MP1764A
Error Detector
Operation Manual
(GPIB Programming)

Fifth Edition

Read this manual after reading the MP1764A operation manual first. Keep it handy so that it can be read when necessary.

Measurement Solutions

ANRITSU CORPORATION

Document No.: M-W0888AE-5.0

MP1764A Error Detector
Operation Manual
(GPIB Programming)

April 1995 (First Edition)
April 2002 (Fifth Edition)

Printed in Japan

WARNING

- *The protective earth terminal of this instrument must be connected to ground. The three-core power cord supplied with the instrument can be plugged into a grounded two pole AC outlet. If no grounded two pole AC outlet is available, the ground pin of the power cord or the earth terminal on the rear panel must be connected to ground before supplying the power to the instrument. Failure to do so could cause dangerous or possibly fatal electric shocks.*
- *Replacing fuses with the power cord still plugged into an AC outlet could also cause electric shocks.*
- ***Supplemental explanation about WARNING on the rear panel***

WARNING ⚠
NO OPERATOR SERVICE-
ABLE PARTS INSIDE.
REFER SERVICING TO
QUALIFIED PERSONNEL.

A supplemental explanation about the WARNING labeled on the rear panel is given in the following:

Disassembly, adjustment, maintenance, or other access inside this instrument by unqualified personnel should be avoided. Maintenance of this instrument should be performed only by Anritsu trained service personnel who are familiar with the risks involved of fire and electric shock. Potentially lethal voltages existing inside this instrument, if contacted accidentally, may result in personal injury or death, or in the possibility of damage to precision components.

■ SAFETY CONSIDERATIONS:

Anritsu uses the following labels to identify safety precautions which should be followed to prevent personal injury or product damage. Please familiarize yourself with them before operating this product.

Labels used in this manual:

WARNING : Indicates that the procedure could result in personal injury if not correctly performed. Do not proceed before you fully understand the explanation given with this symbol and meet the required conditions.

CAUTION : Indicates that the operating procedure could result in damage to the product if not correctly performed. Do not proceed before you fully understand the explanation given with this symbol and meet the required conditions.

Labels or symbols used on/in the product:

⚠ : This international caution symbol indicates that the operator should refer to the operation manual before beginning a procedure.

⏚ : This symbol indicates an earth (ground) terminal. The product should be grounded via the earth terminal if a three prong power cord is not used.

CERTIFICATION

ANRITSU CORPORATION certifies that this instrument has been thoroughly tested and inspected, and found to meet published specifications prior to shipping.
Anritsu further certifies that its calibration measurements are based on the Japanese Electrotechnical Laboratory and Radio Research Laboratory standards.

WARRANTY

All parts of this product are warranted by Anritsu Corporation of Japan against defects in material or workmanship for a period of one year from the date of delivery.
In the event of a defect occurring during the warranty period, Anritsu Corporation will repair or replace this product within a reasonable period of time after notification, free-of-charge, provided that: it is returned to Anritsu; has not been misused; has not been damaged by an act of God; and that the user has followed the instructions in the operation manual.

Any unauthorized modification, repair, or attempt to repair, will render this warranty void.

This warranty is effective only for the original purchaser of this product and is not transferable if it is resold.

ALL OTHER EXPRESSED WARRANTIES ARE DISCLAIMED AND ALL IMPLIED WARRANTIES FOR THIS PRODUCT, INCLUDING THE WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED IN DURATION TO A PERIOD OF ONE YEAR FROM THE DATE OF DELIVERY. IN NO EVENT SHALL ANRITSU CORPORATION BE LIABLE TO THE CUSTOMER FOR ANY DAMAGES, INCLUDING LOST PROFITS, OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES, ARISING OUT OF THE USE OR INABILITY TO USE THIS PRODUCT.

All requests for repair or replacement under this warranty must be made as soon as possible after the defect has been noticed and must be directed to Anritsu Corporation or its representative in your area.

MEMORY BACK-UP BATTERY REPLACEMENT

The power for memory back-up is supplied by a Manganese-dioxide Lithium Battery. This battery should only be replaced by a battery of the same type; since replacement can only be made by Anritsu, contact the nearest Anritsu representative when replacement is required.

At the end of it's life, the battery should be recycled or disposed properly.

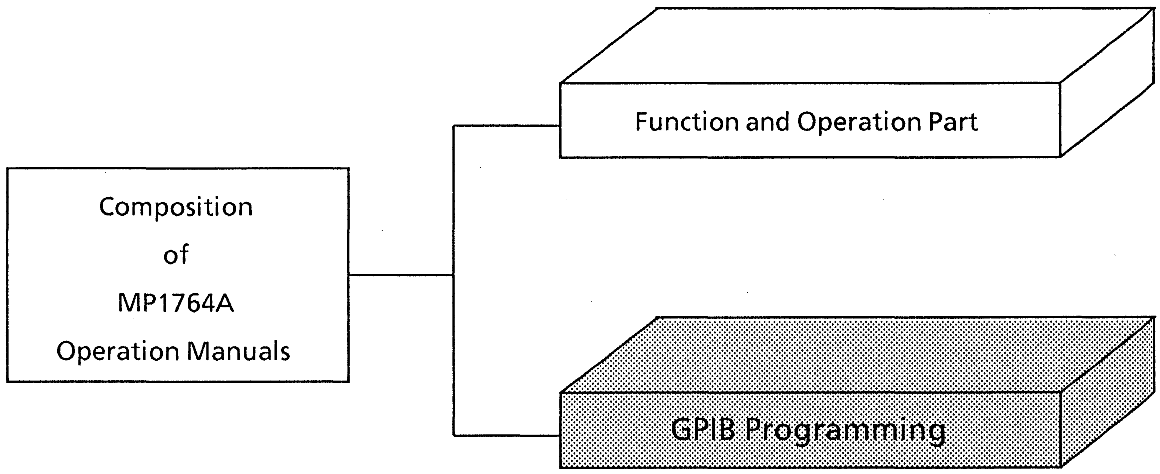
CONCERNING THE DISPOSAL OF THE MP1764A

This instrument employs semiconductors which contain an arsenical compound. Should this instrument be discarded for any reason, an adequate care should be taken so that it is disposed of according to the waste disposal laws of your own country.

(Blank)

Composition of MP1764A Operation Manuals

The MP1764A Error Detector operation manuals are composed of the following two documents. Use them properly according to the usage purpose.



Function and Operation Part	These outline the MP1764A, and describes the preparations before use, the panels, specifications, performances, functions, and operation procedures.
GPIB Programming :	The MP1764A GPIB conforms to IEEE488.2. Remote control by GPIB is explained based on IEEE488.2. An application program example using the ANRITSU PACKET V series of personal computers, HP9000 series HP-BASIC and Quick Basic of Microsoft Corporation are also provided.

(Blank)

TABLE OF CONTENTS

SECTION 1
GENERAL

This section outlines the histotrical development of the GPIB standard and gives a general description of GPIB functions of the MP1764A Error Detector.

TABLE OF CONTENTS

(Blank)

1.1 Development of the GPIB Standard

The MP1764A, when combined with an external controller in a system bus automates measurements on radio communications. For this purpose it is provided with a GPIB interface bus (IEEE std. 488.2-1987) as a standard feature. The GPIB (General Puropose Interrace Bus) was established by the IEEE (Institute of Electric and Electronics Engineers) in 1975 as a standard digital interface bus for programmable measuring instruments. The original version was announced in 1975 under the name IEEE std. 488-1975.

A revised version, called IEEE std. 488-1978, was issued in 1978. As this version only stipulated hardware specifications for the interface side, IEEE std. 728-1982, which stipulated software specifications for the device side, was added in 1982.

Though IEEE std. 728-1982 standardized the formats for sending device messages, it was lacking in its concept of software sharing on the user side. So, in 1987, the IEEE std. 488. 2-1987 (hereafter IEEE 488.2) version, which aimed to overcome the shortcomings, was introduced. This version strengthened the standardization of message exchange protocol, message date code, device input / output formats and common commands.

With the introduction of IEEE 488.2, the name of IEEE std. 488-1978 (hereafter IEEE 488) was changed to IEEE std. 488. 1-1987 (hereafter IEEE 488.1). The table below summarizes the development of the GPIB standard.

Object of standard	Former standard	New standard	Remarks
Hardware	IEEE 488	IEEE 488.1	IEEE 488.1 is indentical to IEEE 488
Software	IEEE 728	IEEE 488.2	IEEE 488.2 is the revised version of IEEE 728

Devices which support IEEE 488.2 must also have compatibility with IEEE 488.1; however, devices which support IEEE 488.1 (IEEE 488) are not guaranteed to be compatible to IEEE 488.2.

1.2 MP1764A GPIB Functions

The MP1764A has the following GPIB functions.

(1) Apart from the power switch and some LOCAL keys, all functions can be controlled.

(2) Readout of all setting conditions

(3) Interrupt function and serial poll operation

(4) Automatic measuring systems can be constructed by combining the MP1764A with a personal computer and other measuring instruments.

(5) GPIB is composed of two ports; GPIB 1 and GPIB 2.

For the last feature (5), see the following description as well as examples.

1.2.1 Overviews of 2-port GPIB functions

MP1764A is provided with two GPIB ports. The port on the GPIB 1 side primarily carries out, as the first interface, MP1764A's remote control through an external host computer; on the other hand, the port on the GPIB 2 side primarily controls, as the second interface, output of measurement data to an external printer. Thus an efficient system makeup can be enabled by means of using the GPIB 1 side as a device port and the GPIB 2 side as a system controller port.

(1) Functions of GPIB 1

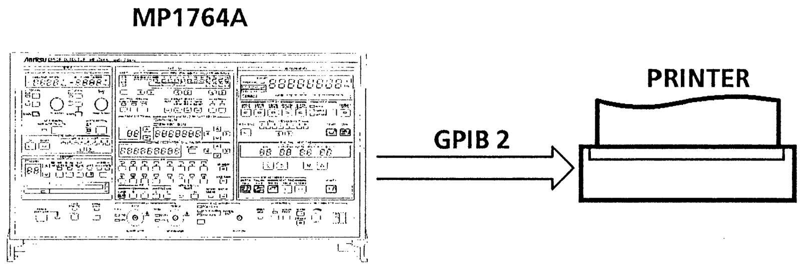
GPIB 1 can be handled similarly to conventional measuring instrument having 1-port GPIB. It functions as a device port when it is in ordinary measurement condition; or it functions as a system controller port to control the MP1763A Pulse Pattern Generator by the system controller's settings in tracking operating.

(2) Functions of GPIB 2

GPIB 2 is used, independent of the GPIB 1 port, as a device control port of individual devices connected to the GPIB 2 port. Thus GPIB 2 always functions as a system controller port, but not as a device port.

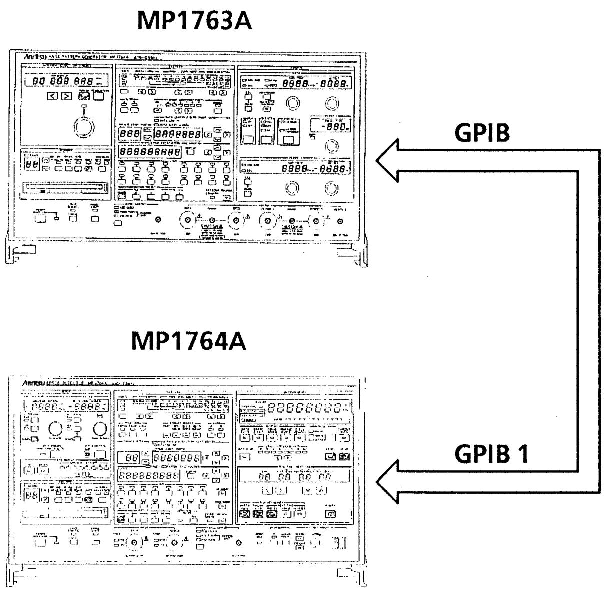
1.2.2 Examples of system makeup using GPIB 1 / GPIB 2

(1) Stand-alone type (1) ... Panel operation



Outputs data measured with **MP1764A** to the printer through panel operation.

(2) Stand-alone type (2) ... Tracking operation

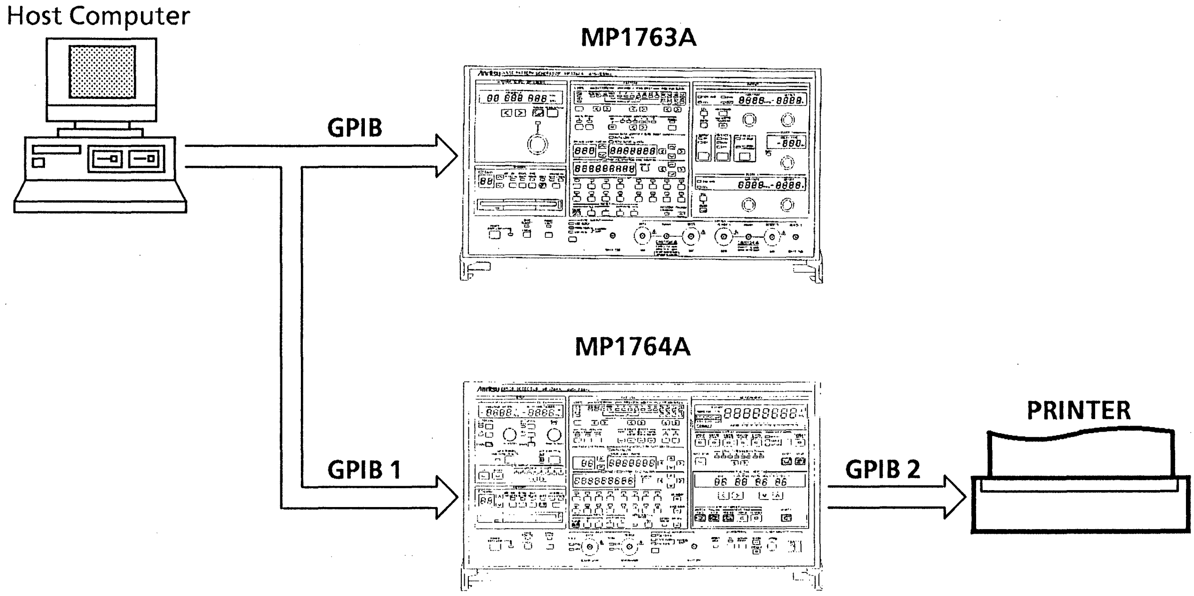


① Some settings for the transmitter are synchronized with the settings for the receiver. During this tracking operation, no external controller can be connected.

② Some settings for the receiver are synchronized with the settings for the transmitter. During this tracking operation, no external controller can be connected.

※ In the tracking operation, either MP1763A or MP1764A can be a master (controller).

(3) Control by the host computer



By means of controlling **MP1763A** and **MP1764A** using the host computer via **GPIB 1** port, data can be output to the printer via **GPIB 2** port.

SECTION 2
SPECIFICATIONS

In this section, interface functions of the MP1764A GPIB specifications are explained. For the device message, see SECTION 9.

TABLE OF CONTENTS

(Blank)

2.1 Interface Functions

IEEE 488.2 sets down a minimum requirement for subsets of the GPIB interface functions specified in IEEE 488.1 that must be provided by measuring intruments used in a GPIB system. The MP1764A GPIB 1 and GPIB 2 provide the subsets listed in the code columns of the tables below.

GPIB 1 Interface Functions

Code	Interface function	IEEE 488.2 standard
SH1	All source handshake functions are provided. Synchronizes the timing of data transmission.	All functions provided as standard. The device must have a complete set of source handshake functions.
AH1	All acceptor handshake functions are provided. Synchoronizes the timing for receiving data.	All functions provided as standard. The device must have a complete set of acceptor handshake functions.
T6	Basic talker functions are provided. The serial poll function is provided. The talk-only function is not provided. The talker can be canceled by MLA.	Davices must have one of the T5, T6, TE5 or TE6 subsets. The talk-only function is out of the scope of the IEEE 488.2 standard.
L4	Basic listener functions are provided. The listen-only function is not provided. The listener can be canceled by MTA.	Devices must have one of the L3, L4, LE3 or LE4 subsets. The listen-only function is out of the scope of the IEEE 488.2 standard.
SR1	All service request and status byte functions are provided.	All functions are provided as standard.
RL1	All remote / local functions are provided. The local lockout function is provided.	RL0 (functions not provided) or RL1 (all functions provided)
PP0	Parallel poll functions are not provided.	PP0 (functions not provided) or PP1 (all functions provided)
DC1	All device clear functions are provided.	All functions provided as standard.
DT1	Device trigger functions are provided.	DT0 (functions not provided) or DT1 (all functions provided)
C1,C2 C3,C4, C7	Controller functions are provided. Can be used as controller only for tracking operation.	C0 (functions not provided) or C4 and C5 or any of C7, C9, C11

GPIB 2 Interface Functions

Code	Interface fnction
SH1	All source handshake functions are provided. Synchronizes the timing of data transmission.
AH1	All acceptor handshake functions are provided. Synchronizes the timing for receiving data.
T6	Basic talker functions are provided. Serial poll functions are provided. The talk-only function is not provided. A talker can be canceled by MLA.
L4	Basic listener functions are provided. The listen-only function is not provided. A listender can be canceled by MTA.
SR0	Service request and status byte functions are not provided.
RL0	Remote / local functions are not provided. Local lockout functions are not provided.
PP0	Parallel poll functions are not provided.
DC0	Device clear functions are not provided.
DT0	Device trigger functions are not provided.
C1,C2,C3,C4, C28	Controller functrions are provided.

2.2 Device Message List

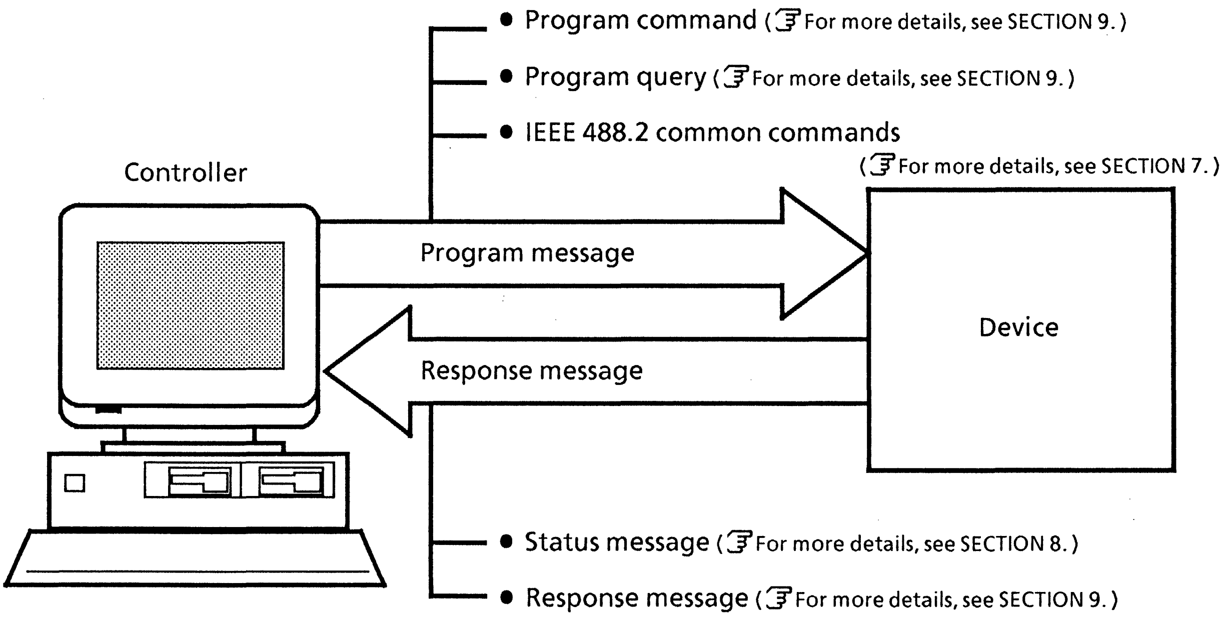
Device messages are message that are transmitted between the controller and the device via the system interface in the bus mode, i.e. when the ATN line, is false. There are two types: program messages and response messages.

Program messages are ASCII data message transferred from controller to device. There are two types of program message: program commands and program queries.

Program commands consist of commands specific to devices which are used exclusively for the control of the MP1764A and IEEE 488.2 common commands. The latter are common commands used for, in addition to the MP1764A, any measuring instrument conforming to the IEEE 488.2 standard.

Program queries are commands used to elecit are response message from a devcice. A program query is transferred from the controller to the device so that the controller can receive a response message from the controller later on.

Reponse messages are ASCII data messages sent from device to controller. Status messages and response messages for program queries are listed on the following pages.



The messages described above are transferred via the input and output buffers of the device. The output buffer is also referred to as an output queue. The following table gives a brief explanation of input and output buffers.

Input buffer	Output buffer
A FIFO (First In First Out) memory area where DAB (program messages or query messages), whose syntax has been analyzed, are temporarily stored before they are executed. The size of the MP1764A is input buffer is 256 bytes.	A FIFO-type queue memory area. All DAB (response messages) output to a device from the controller are all stored in this area until the controller has read each of them. The size of the MP1764A output queue is 256 bytes.

2.2.1 IEEE 488.2 common commands and MP1764A supported commands

The table below lists 39 types of common commands specified in the IEEE 488.2 standard. IEEE 488.2 common commands which are supported by the MP1764A are indicated with ◎ symbol in the table.

Mnemonic	Command name	IEEE488.2 Standard	MP1764A supported commands
*AAD	Accept Address Command	Optional	
*CAL?	Calibration Query	Optional	
*CLS	Clear Status Command	Mandatory	◎
*DDT	Define Device Trigger Command	Optional	
*DDT?	Define Device Trigger Query	Optional	
*DLF	Disable Listener Function Command	Optional	
*DMC	Define Macro Command	Optional	
*EMC	Enable Macro Command	Optional	
*EMC?	Enable Macro Query	Optional	
*ESE	Standard Event Status Enable Command	Mandatory	◎
*ESE?	Standard Event Status Enable Query	Mandatory	◎
*ESR?	Standard Event Status Register Query	Mandatory	◎
*GMC?	Get Macro Contents Query	Optional	
*IDN?	Identification Query	Mandatory	◎
*IST?	Individual Status Query	Optional	
*LMC?	Learn Macro Query	Optional	
*LRN?	Learn Device Setup Query	Optional	
*OPC	Operation Complete Command	Mandatory	◎
*OPC?	Operation Complete Query	Mandatory	◎
*OPT?	Option Identification Query	Optional	
*PCB	Pass Control Back Command	Mandatory if other than CO	
*PMC	Purge Macro Command	Optional	
*PRE	Parallel Poll Register Enable Command	Optional	
*PRE?	Parallel Poll Register Enable Query	Optional	
*PSC	Power On Status Clear Command	Optional	◎
*PSC?	Power On Status Clear Query	Optional	◎
*PUD	Protected User Data Command	Optional	
*PUD?	Protected User Data Query	Optional	
*RCL	Recall Command	Optional	
*RDT	Resource Description Transfer Command	Optional	
*RDT?	Resource Description Transfer Query	Optional	
*RST	Reset Command	Mandatory	◎
*SAV	Save Command	Optional	
*SRE	Service Request Enable Command	Mandatory	◎
*SRE?	Service Request Enable Query	Mandatory	◎

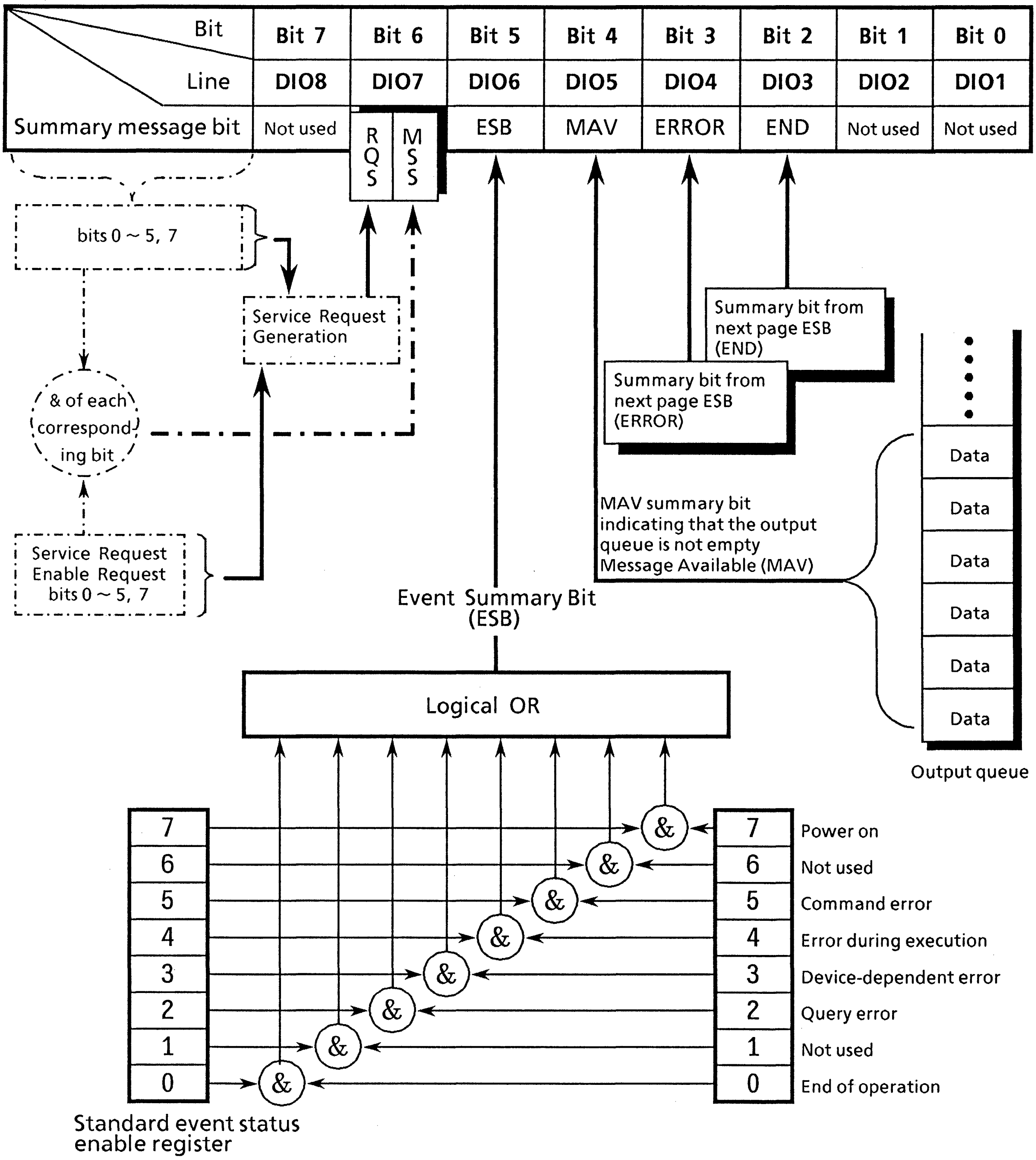
Mnemonic	Command name	IEEE488.2 Standard	MP1764A supported commands
*STB?	Read Status Byte Query	Mandatory	◎
*TRG	Trigger Command	Mandatory if DT1	◎
*TST?	Self Test Query	Mandatory	◎
*WAI	Wait to Continue Command	Mandatory	◎

The IEEE 488.2 common commands are always begin with "*" For more details, see SECTION 7.

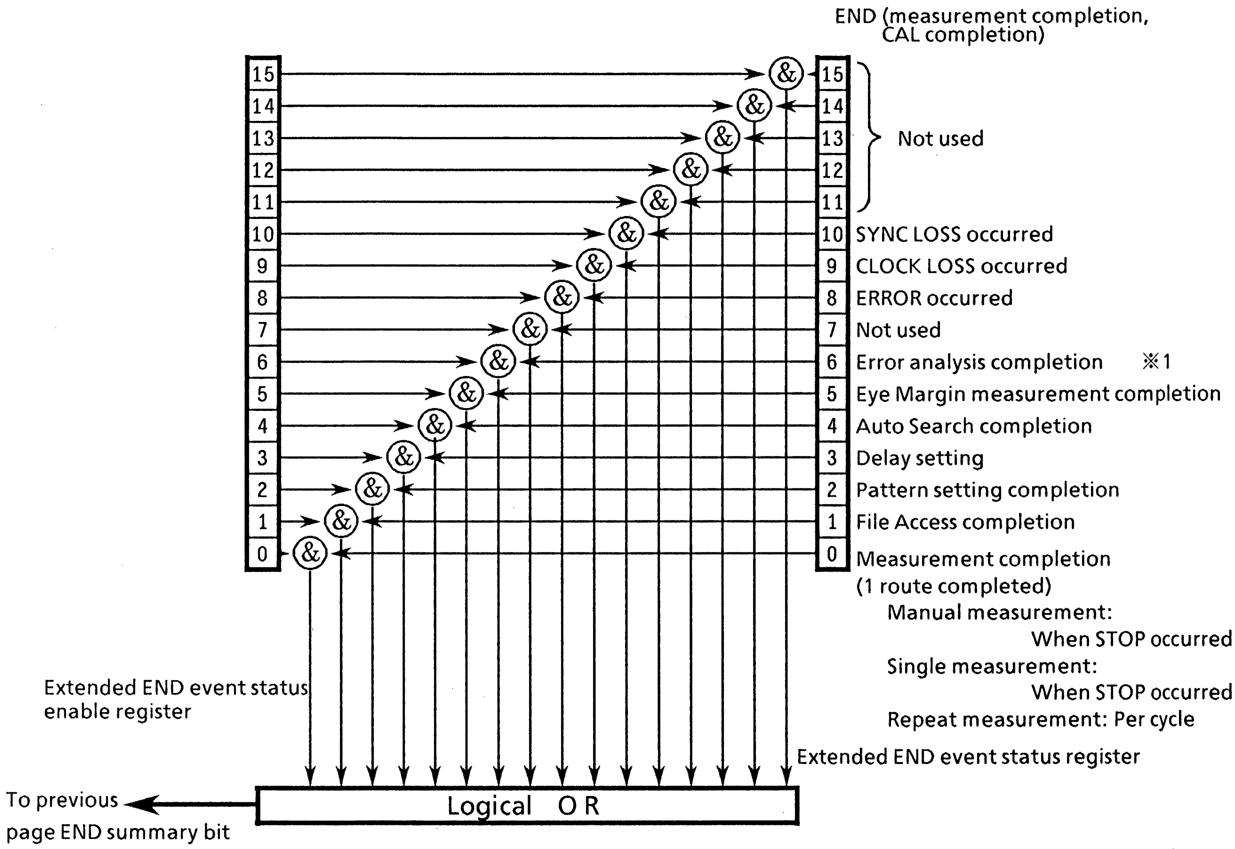
2.2.2 Status messages

The diagram below shows the structure of service-request summary messages for the status byte register used with the MP1764A.

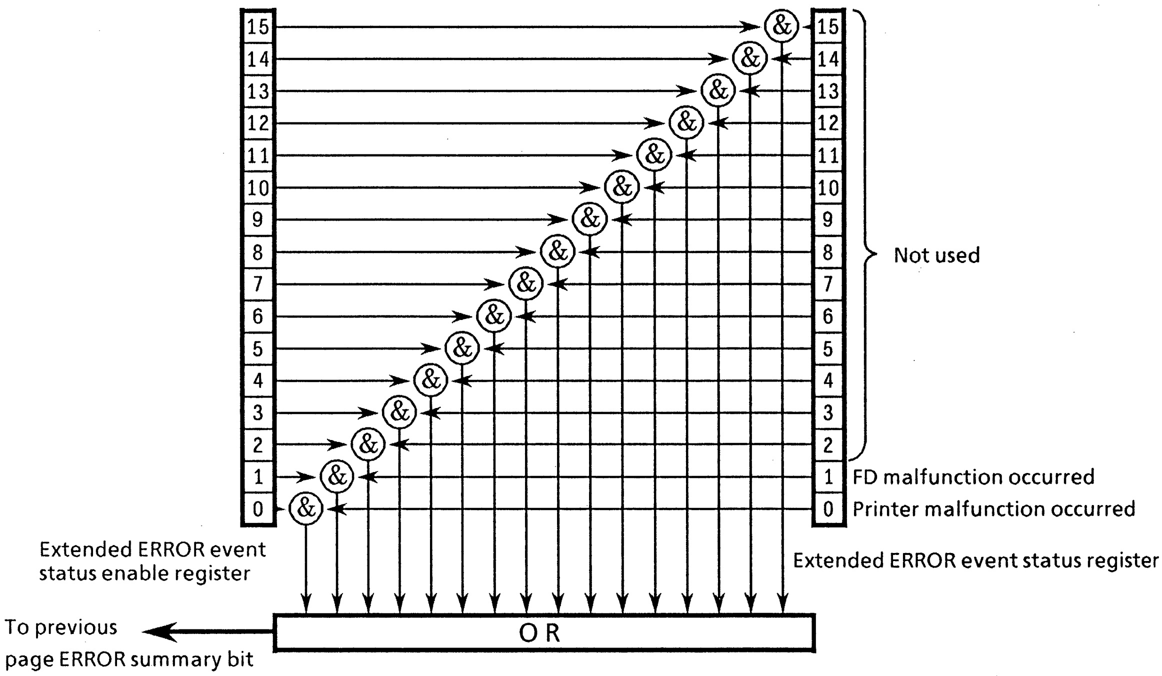
Status Byte Register Summary Bit Composition



Note : (&) means logical AND operation.



※1) Error analysis is allowed only when OPTION-01 is installed.



2.2.3 MP1764A device messages

The device messages consist of fixed program commands of the MP1764A queries and response messages. The device messages list and description are shown in Section 9.

SECTION 3
CONNECTING THE BUS AND SETTING ADDRESS

The remote control of devices connected to the GPIB system interface begins with referring to their addresses as control procedure parameters. This section describes the GPIB cable connections and setting of addresses that must be performed before using the GPIB interface.

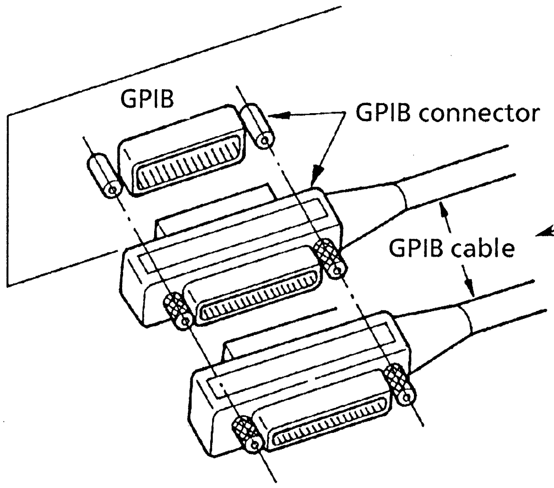
TABLE OF CONTENTS

(Blank)

3.1 Connecting Devices with GPIB Cables

The rear panel has connectors for connecting GPIB cables. The cables must be connected before the power is switched on.

A maximum of 15 devices, including the controller, can be connected to one system. The restrictions indicated at the right of the diagram below should be observed when connecting many devices to one system.



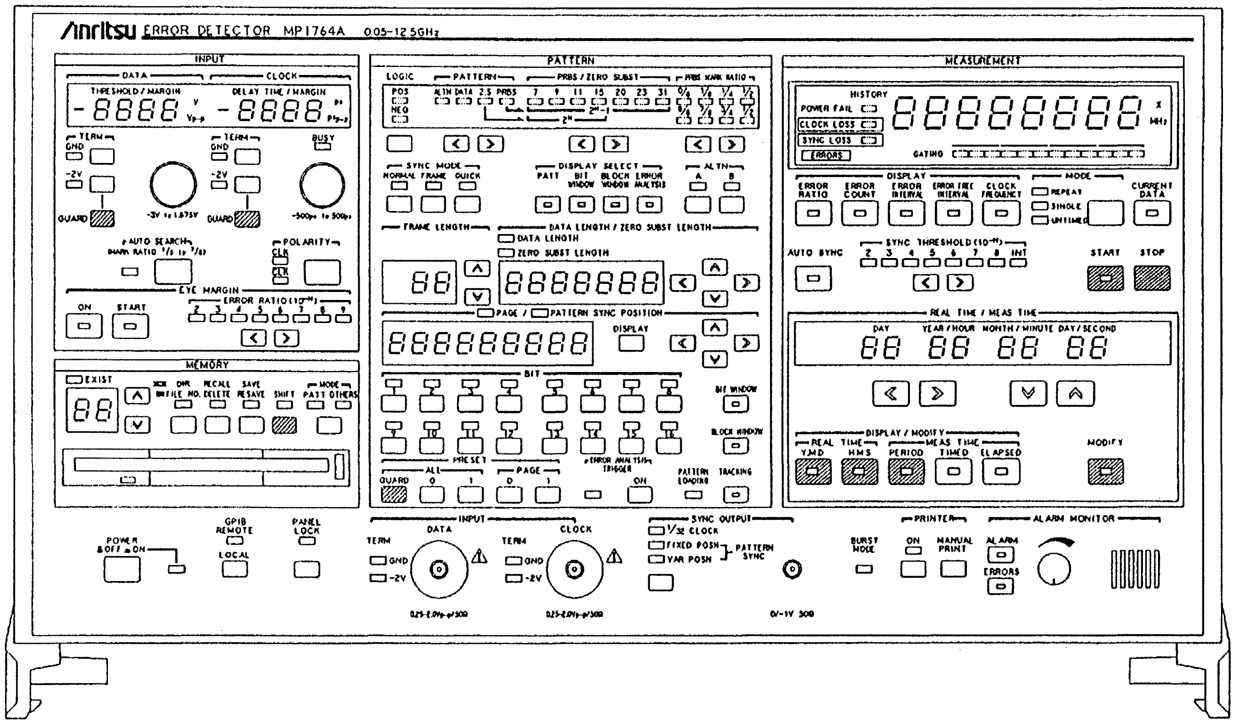
Total cable length	≦ 20 m
Each cable length between devices	≦ 4 m
Number of devices which can be connected	≦ 15

3.2 Procedures for Setting the Address and Checking it

Set the GPIB address for the MP1764A after or before turning on the power. The GPIB 1 address (for device) is factory-set to 0. The address is preset with the GPIB ADDRESS switch on the rear panel. There is no need to set the address if using the default address. To change the address, put the MP1764A in the local state and input the address using the GPIB ADDRESS switch on the rear panel. Devices connected to the GPIB are normally in the local state when the power is turned on.

Note : 1) The system always checks the GPIB "ADDRESS" switch settings when the power is turned on and determines its own address. So, changing the address is always allowed unless the system is in remote state.

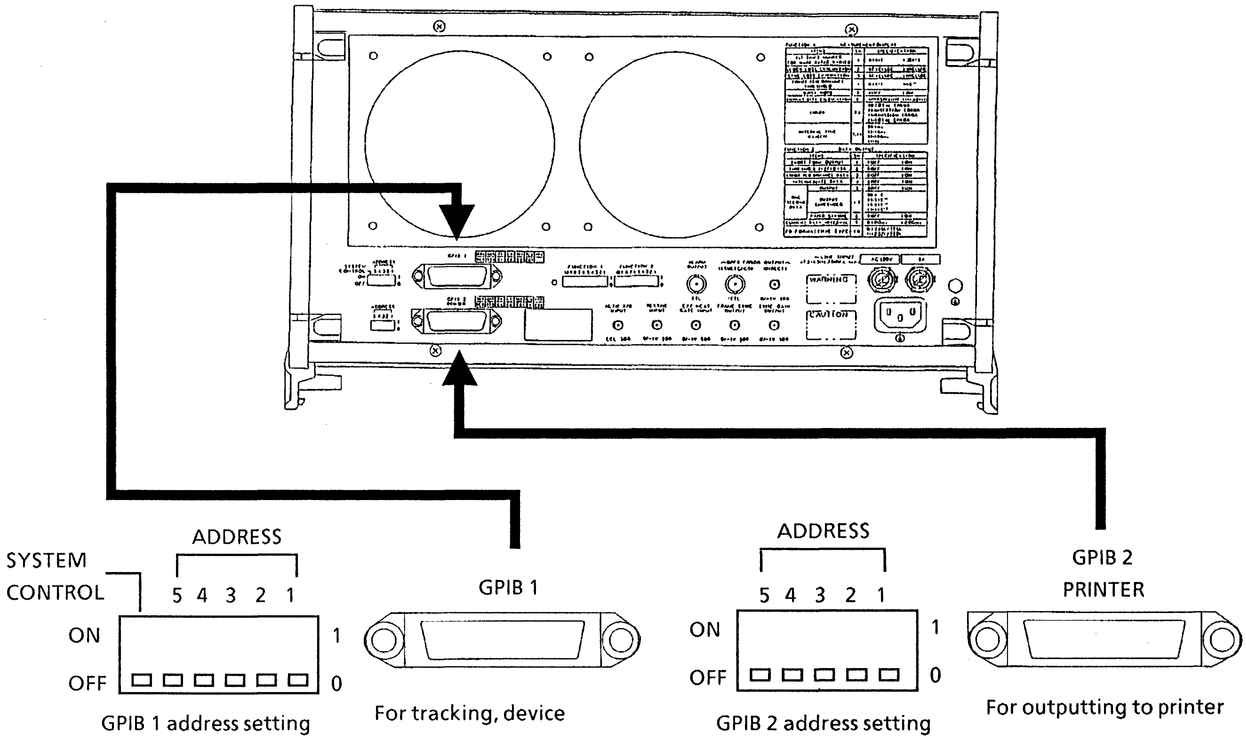
2) To control the system as a device from an external controller, set "SYSTEM CONTROL" of the GPIB 1 address switch to OFF(0).



3.2.1 Address setting

The GPIB addresses for two GPIB ports of MP1764A are set with the DIP switch on the rear panel, respectively.

MP1764A Rear panel

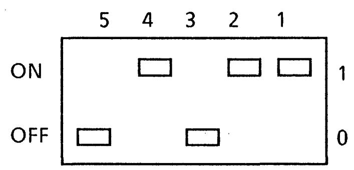


The GPIB 1's and GPIB 2's addresses can be set 0 to 30, respectively. Five switches are weighed differently: "5", "4", "3", "2" and "1" are respectively weighed to 16, 8, 4, 2 and 1.

To set the address to 11, for example, the operation is as follows: Since

$$11 = 8 + 2 + 1,$$

set switches "4", "2" and "1" to ON as shown below.



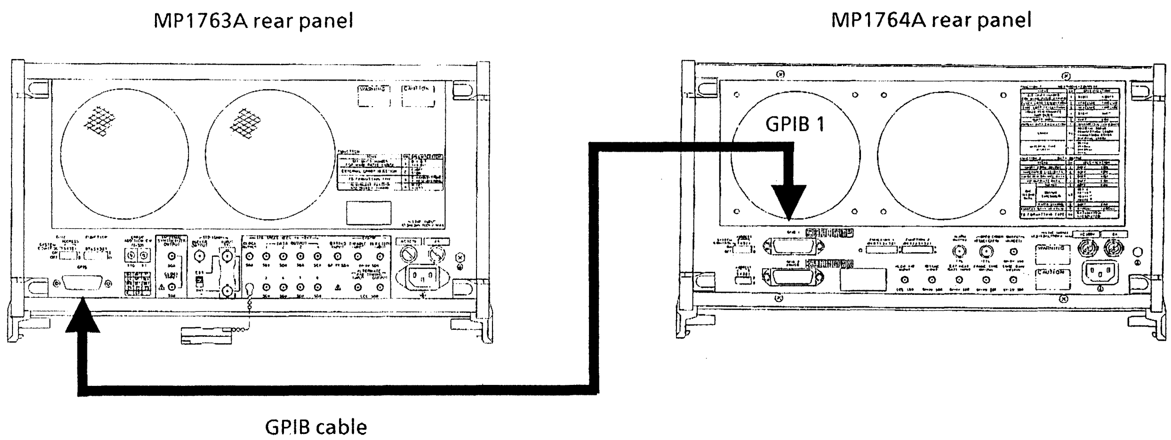
However, address 31, where all the switches are set to ON, is assumed to be address 0.

3.2.2 Connection with MP1763A during the tracking operation

Tracking operation is a function that pattern settings are made to be synchronized with each other between MP1763A and MP1764A. Either MP1763A or MP1764A is made to be a Master and the other is made to be a Slave, and the settings for the Slave are synchronized with those for the Master.

(1) If MP1763A is a Master and controls MP1764A:

When the settings for MP1763A are set to MP1764A via GPIB, the setting and connection are as follows:



a) Like the diagram above, the GPIB connector on the MP1763A's rear panel is connected with the GPIB 1 connector on the MP1764A via the GPIB cable (included).

b) Set "SYSTEM CONTROL" of the GPIB address switch on the MP1763A' rear panel to ON (1).

c) Set the value of the GPIB 1 address switch on the MP1764A's rear panel to that of MP1764A's GPIB address + 2.

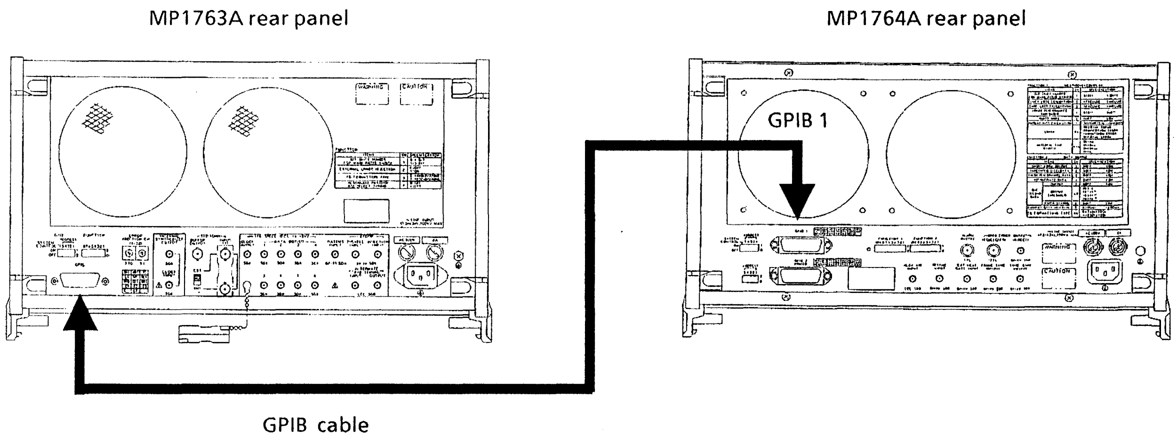
d) Turn on the MP1763A power again.

e) Set the TRACKING key on the MP1763A's front panel to ON.

By now, you are ready to perform pattern-tracking.

(2) If MP1764A is a Master and controls MP1763A:

When the settings for MP1764A are set for MP1763A via GPIB, the setting and connection are as follows:



a) Like the diagram above, the GPIB 1 connector on the MP1764's rear panel is connected with the GPIB connector on the MP1763A via the GPIB cable (included).

b) Set "SYSTEM CONTROL" of the GPIB 1 address switch on the MP1764A's rear panel to ON (1).

c) Set the value of the GPIB address switch on the MP1763A's rear panel to that of MP1764A's GPIB 1 address + 2.

d) Turn on the MP1764A power again.

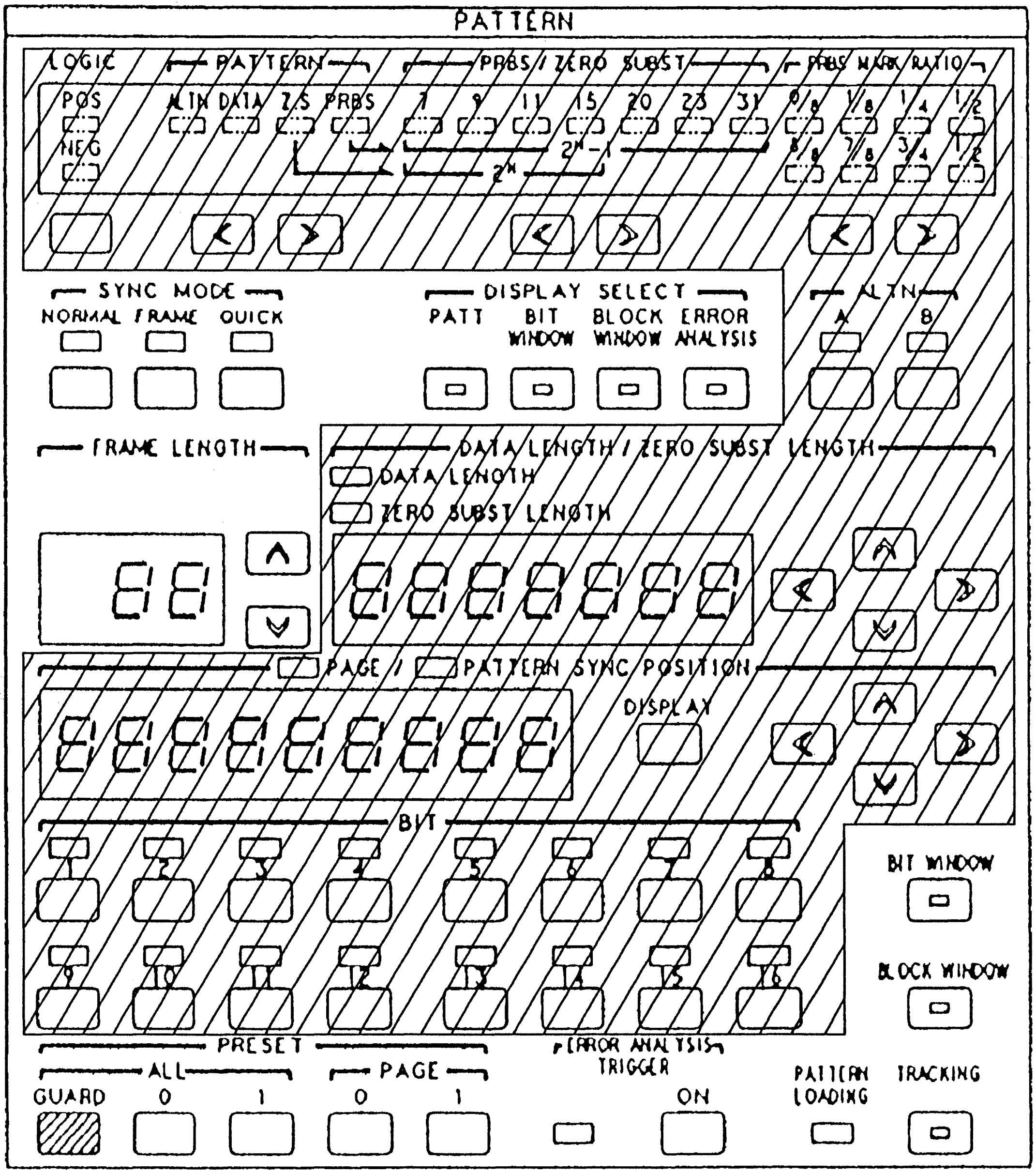
e) Set the TRACKING key on the MP1764A's front panel to ON.

By now, you are ready to perform pattern-tracking.

(3) Items to be tracked between MP1763A and MP1764A

The setting items to be tracked using pattern-tracking function are as follows:

Pattern-setting area on the MP1764A's front panel



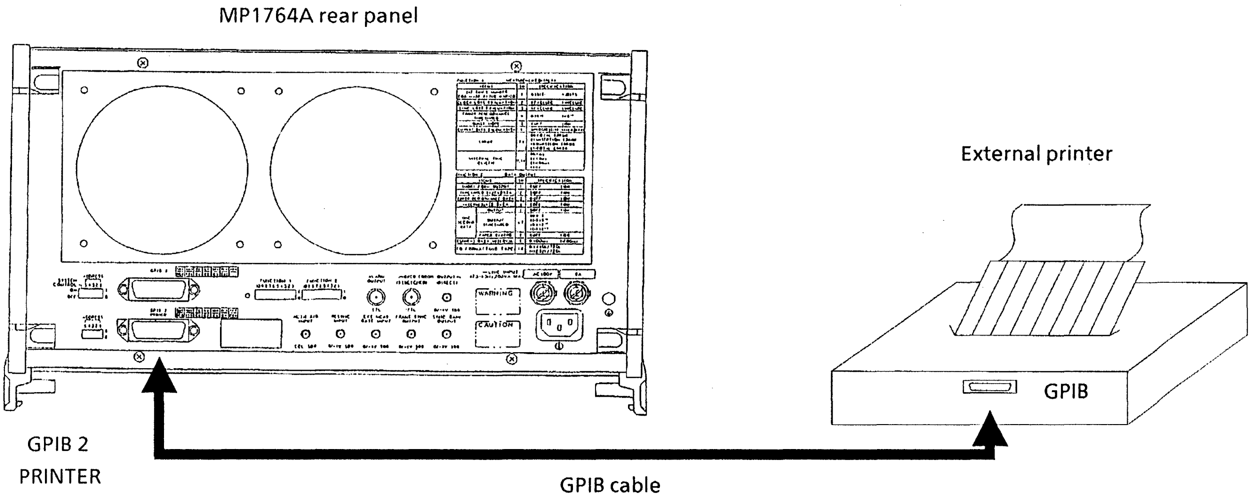
Items to be tracked are pattern-settings for the shaded area shown on the above diagram.

Within the shaded area shown above, however, the area where a setting for MP1763A does not coincide with that for MP1764A (such as error analysis data) cannot be pattern-tracked.

☞ For more information on the setting items to be pattern-tracked, see "APPENDIX D Tracking Items List."

3.2.3 Connection with external printer

The MP1764A is provided with the GPIB port for outputting measured data to an external printer. When the measured data is output to an external printer, the connection and address setting are as follows:



a) Like the diagram above, the GPIB 2, PRINTER connector on the MP1764A's rear panel is connected with the GPIB connector on an external printer via the GPIB cable.

b) Set the GPIB 2, PRINTER address switch on the MP1764A's rear panel.

c) Set the address on an external printer to the MP1764A's GPIB 2, PRINTER address set in b) +2.

d) Set the "PRINTER ON" or "MANUAL PRINT" key on the MP1764A's front panel to ON.

By now, measured data is ready for being output.

If measured data is non-available (e.g., during measurement halt), however, it is not output.

(Blank)

SECTION 4
INITIAL SETTINGS

There are 3 levels of initialization for the GPIB interface system. The first level is bus initialization in which the system bus is in the idle state. The second level is initialization for message exchange in which devices are able to receive program message. The third level is device initialization in which device functions are initialized. These levels of initialization prepare a device for operation.

Control commands by ANRITSU PACKET V series personal computers are applied for formats and use examples in this section.

TABLE OF CONTENTS

(Blank)

The IEEE 488.1 standard stipulates the following two levels for the initialization of the GPIB system.

- Bus initialization

  All interface functions connected to the bus are initialized by **IFC** messages from the controller.

- Device initialization

  The **DCL** GPIB bus command returns all devices to their initial states while the **SDC** GPIB bus command returns designated devices only to their stipulated initial states.

In the IEEE 488.2 standard the initialization levels are divided into three, with bus initialization as the highest level. The second level is initialization for message exchange and third device initilazation. This standard also stipulates that a device must be set to a known state when the power is turned on. The above details are summarized in the table below.

Level	Initialization type	Description
1	Bus initialization	All interface functions connected to the bus are initialized by **IFC** messages from the controller
2	Initialization for the exchange of messages	The **DCL** and **SDC** GPIB bus commands perform initialization for message exchange for all devices and designated devices, respectively, as well as nullifying the function to report the end of operation to the controller.
3	Device initialization	The ***RST** or **INI** reset command resets only specified devices, from among those connected to the GPIB, to their known states regardless of the conditions under which they were previously being used.

For levels 1, 2 and 3, see the following description that focuses the instructions for executing these initializations and their results which mean the items to be initialized. Also, the known states to be set at power-on are described.

IFC @

4.1 Bus Initialization by the IFC Statement

■ **Syntax**

IFC△@ select code

■ **Example**

```
IFC @1
```

■ **Explanation**

The IFC line of the GPIB in the stipulated select code is kept active for approximately 100 μs (electrically low level state).

When **IFC@** is executed, the interface functions of all devices connected to the bus line of the GPIB in the select code are initialized. Only the system controller can send this command.

The initialization of interface functions involves erasing the settings made by the controller and resetting them to their initial states. In the table below, ○ indicates the functions which are initialized; △ indicates the functions which 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	○

Even if the **IFC** statement is True (the level of the IFC line is set to low by execution of the **IFC@** statement), levels 2 and 3 initialization are not performed, so, it does not affect devicer operating conditions (parameter setting, LEDs ON / OFF, etc.).

The following lists the effect of the **IFC** statement on some device functions taken from the table above.

① Talker / listener

All talkers and listeners are put in the idle state (TIDS, LIDS) within 100μs.

② Controller

The controller is put in the idle state (CIDS – Controller Idle State) within 100μs if it is not active (SACS – System Control Active State).

③ Return of control

If the system controller (the device on the GPIB initially designated as controller) has given up its control function to another device, executing **IFC@** returns the control function to the system controller. The system controller's **RESET** key causes it to output an **IFC** message.

④ Service request devices

The IFC statement has no effect on a device sending an **SRQ** message to the controller (the SRQ line in the figure below is set to low level by the device), but it does clear the condition that the controller has put all devices connected to the system bus into serial poll mode.

⑤ Devices in the remote state

The **IFC** statement has no effect on devices in the remote state.

DCL @

4.2 Initialization for Message Exchange by DCL and SDC Bus Commands

■ Syntax

DCL△@ select code [primary address] [secondary address]

■ Example

DCL @1	Initializes all devices under the bus for message exchange (sending DCL).
DCL @1Ø3	Initializes only the device whose address is 3 for message exchange (sending SDC).

■ Explanation

This statement carries out the initialization for message exchange for all devices on the GPIB of the specified select code or that for specified devices only.

The purpose of initialization for message exchange is to prepare devices to receive new commands from the controller when the sections of devices used for the exchange of messages are in an inappropriate state to be controlled by the controller as the result of the execution of other programs, etc. There is no need to change the panel settings, however.

■ When only the select code is specified

This carries out the initialization for message exchange of all devices on the GPIB of the specified select code. **DCL@** sends a **DCL** (Device Clear) bus command to the GPIB.

■ When the address is specified

Performs initialization for message exchange for the specified device. After clearing the listeners on the GPIB of the specified select code, the specified device only is set to listener and an **SDC** (Selected Device Clear) bus command is output.

■ Items to be initialized for message exchange

① Input buffer and output queue	Cleared
② Parser, execution controller and response formatter .	Reset
③ Device commands including ✻RST	All commands that interfere with the execution of these commands are cleared.
④ Coupled-parameter program messages	All commands (in the execution pending sections and queries) are discarded because they are coupled parameters.
⑤ Processing the ✻OPC command	Puts a device in OCIS (Operation Complete Idle State). As a result, the operation complete bit cannot be set in the standard event status register.

⑥ Processing the ∗OPC? query .... Puts a device in OQIS (Operation Complete Query Idle State). As a result, the operation complete bit cannot be set in the output queue. The MAV bit is cleared.

⑦ Automation of system construction .... The ∗ADD and ∗DLF common commands are nullified. (These commands are not supported on the MP1764A)

⑧ Device functions .... Functions for message exchange are put in the idle state. The device continues to wait for a message from the controller.

CAUTION

Device clear is prohibited from carrying out the followings.

① ***Changing the current device settings or stored data.***

② ***Interrupting front panel I / O***

③ ***Changing any other status bit except clearing the MAV bit, when clearing the output queue.***

④ ***Interrupting or having any effect on the device that is currently operating.***

■ Transmission sequence of GPIB bus commands by the DCL@ statement.

The transmission sequence of the DCL and SDC GPIB bus commands by the **DCL@** statement is shown in the table below.

Statement	Bus command transmission sequence (at ATN line "LOW")	Data (at ATN LINE "HIGH")
DCL@ select code	UNL, DCL	———
DCL@ device number	UNL, LISTEN address, [secondary address], SDC	———

*RST

4.3 Device Initialization by the *RST Command

■ Syntax

```
*RST
```

■ Example

`WRITE @1Ø3:"*RST"` Initializes only the device of the address 3 with level 3.

■ Explanation

The ***RST** (Reset) is an IEEE 488.2 common command which resets a device with level 3.

Normally devices are set to various states using the commands specific to each device (device messages). The ***RST** command is one of these and is used to reset a device to a specific known state. The function of nullifying of the end of operation is the same as for level 2.

■ Specifying device number in WRITE@ statement

The device with the specified address is initialized with level 3.

After clearing the listeners on the GPIB of the specified select code while the ATN line is active, only the specified device is set to listener.

When the ATN line is false, the ***RST** command is sent.

■ Device Initialization Items

① Device-dependent functions and states

A device is returned to a known state regardless of its current condition. (See the next page for the list.)

② Processing of the *OPC command

The device is put into OCIS (Operation Complete Idle State). As a result, the operation complete bit cannot be set in the standard event status register.

③ Processing the *OPC? query

The device is put into OQIS (Operation Complete Query Idle State). As a result, the operation complete bit cannot be set in the output queue. The MAV bit is cleared.

④ Macro commands

Disables macro operations and puts a device in a mode in which it cannot receive macro commands. Also, the definition of macros is returned to the state specified by the system designer.

Note : The *RST command does not affect the items listed below.

① IEEE 488.1 interface state

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 device specification

⑧ Macros defined by the DMC (Define Macro Contents) command

⑨ Response messages for the PUD (Protect User Data) query

⑩ Response messages for the RDT (Resource Description Transfer) query

There are also preset parameters, etc specific to the MP1764A for the control of external devices, etc. (Refer to SECTION 8 for items ③, ④ and ⑤. The MP1764A does not support items ⑧ to ⑩.)

The table below shows the initial settings proper to the MP1764A for the functions and status.

Initial Settings

Group	Initial Settings	Notes
Setting States	Initialized	See Appendix C Initial Value List for Initial Values.
GPIB Address	Not initialized	
Time & Date	Not initialized	

INI

4.4 Device Initialization by the INI Command

■ **Syntax**

```
INI
```

■ **Example (program message)**

`WRITE @1Ø3:"INI"` Initializing only the device assigned address 3 with level 3.

■ **Description**

The **INI** command is one of the device messages proper to MP1764A; this command is sent as a program message to the device from the controller to reset the device with level 3.

This command functions the same as the ✻**RST** command.

■ **Specifying a device number in the WRITE@ statement**

Initializes the device assigned a specified address with level 3.

The sequence of sending out commands is as follows; listener(s) is(are) released by the GPIB having a specified selection code while the ATN line is true, then only specified device(s) is(are) set to listener(s). When the ATN line turns to false, the **INI** command is output to the specified listener(s) as a program message.

■ **Device's items to be initialized**

The device's items to be initialized are the same as those of the ✻**RST** command.

4.5 Device Status at Power-on

When the power is switched on:

① The device status is the one when the power was last switched off.

② The input buffer and output queue are cleared.

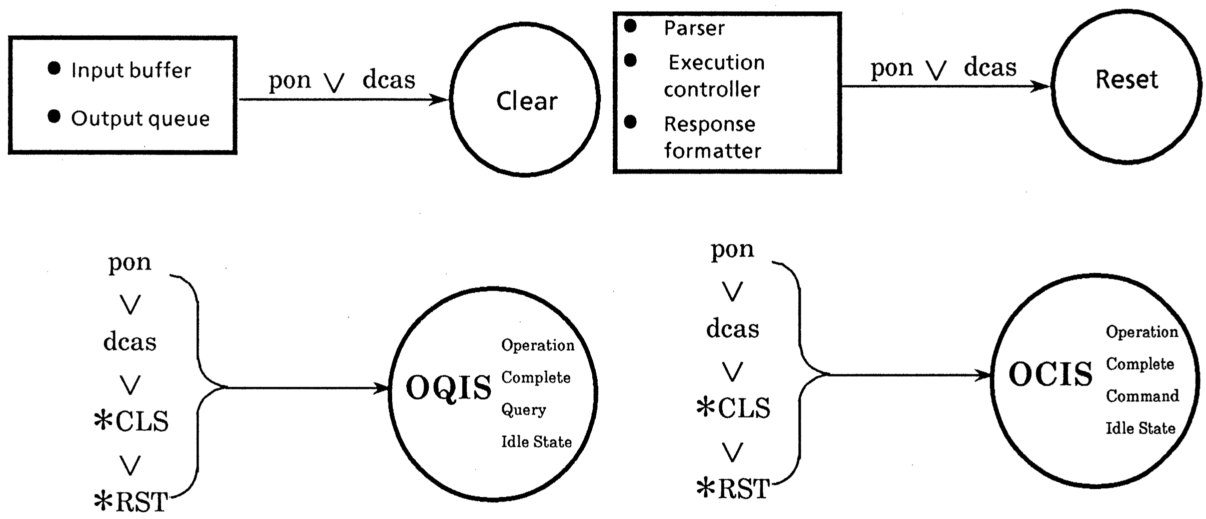
③ The parser, execution control and response formatter are reset.

④ The device is put into the OCIS (Operation Complete Command Idle State).

⑤ The device is put into the OQIS (Operation Complete Query Idle State).

⑥ The MP1764A supports the ***PSC** command. Therefore, when the PSC flag is true and all event status enable registers are cleared. Events can be recorded after the registers have been cleared.

As a special case for ①, the settings are the same as the ones in the Initial Settings Table (in C-1) the first time the MP1764A is switched on after delivery. The diagram below shows the transition states of items ② to ⑤.



■ **Items which do not change at power-on**

① Address

② Related calibration data (The MP1764A has no calibration data.)

③ Data or states which are changed by responses to the common queries listed below.

*IDN?

*OPT? (Not supported by the MP1764A)

*PSC?

*PUD? (Not supported by the MP1764A)

*RDT? (Not supported by the MP1764A)

■ Items related to power-on-status-clear (PSC) flag

The PSC flag has no effect on the Service Request Enable Register, Standard Event Status Enable Register or Extended Event Status Enable Register when it is false. These registers are cleared when it is true or the ∗**PSC** command is not being executed.

■ Items which change at power on

① Current device function state

② Status information

③ ∗**SAV** / ∗**RCL** registers

④ Marco-definition defined by the ∗**DDT** command (not supported by the MP1764A)

⑤ Marco-definition defined by ∗**DMC** command (not supported by the MP1764A)

⑥ Macros enabled by the ∗**EMC** command (not supported by the MP1764A)

⑦ Addresses received by the ∗**PCB** command (not supported by the MP1764A)

SECTION 5
LISTENER INPUT FORMAT

Two types of data message are transmitted between the controller and a device via the system interface when the bus is in the data mode (i.e. the ATN line is false): program messages and response messages. This section describes the format of program messages received by the listener.

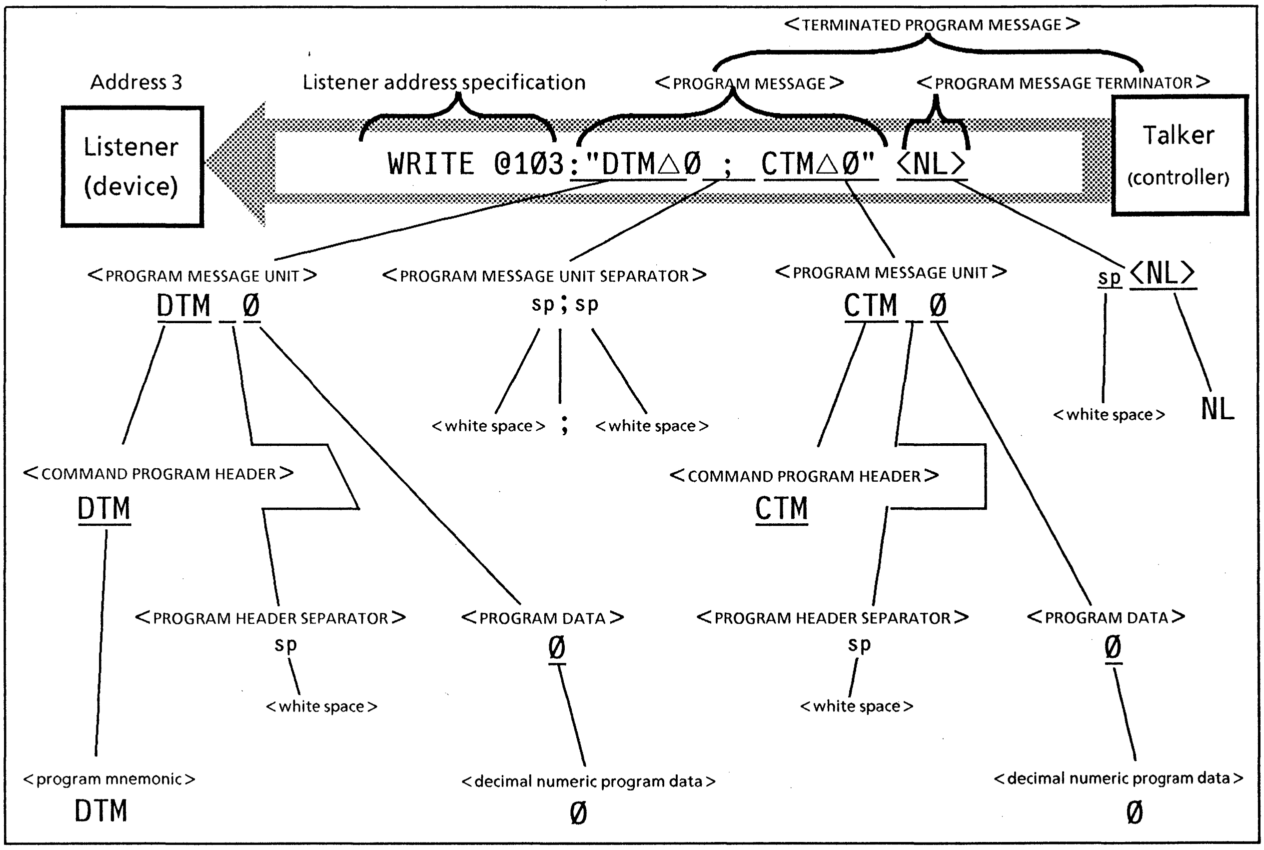
Control commands by ANRITSU PACKET V series personal computers are applied for program examples in this section.

TABLE OF CONTENTS

(Blank)

Program messages comprise a sequence of program message units which are either program commands or program queries.

In the diagram below, in which the data input and clock input termination voltage is set to GND, the controller sends a program message, composed of two program units **DTM△Ø** and **CTM△Ø** linked by a program-message unit separator to a device.



The program message format is a sequence of functional elements which are the minimum requirement for indicating a function. The groups of upper-case alphabetic characters enclosed by < > in the diagram above are examples of functional elements. Functional elements can be further divided into "encoded elements". The groups of lower-case alphabetic characters enclosed by < > in the diagram above are examples of encoded elements.

A diagram indicating the selection of functional elements on a specific path is called a functional syntax diagram, while a diagram indicating the selection of encoded elements on a specific path is called an encoded syntax diagram. The following pages explain program message format using these two diagrams.

Encoded elements represent encoded elements of the actual bus required to send functional element data bytes to a device. Listeners (which receive the functional element data bytes) determine whether they conform to the rules for encoding. If they do not, the listener does not recognize them as functional elements and generates a command error.

5.1 Listener Input Program Message Syntax Notation

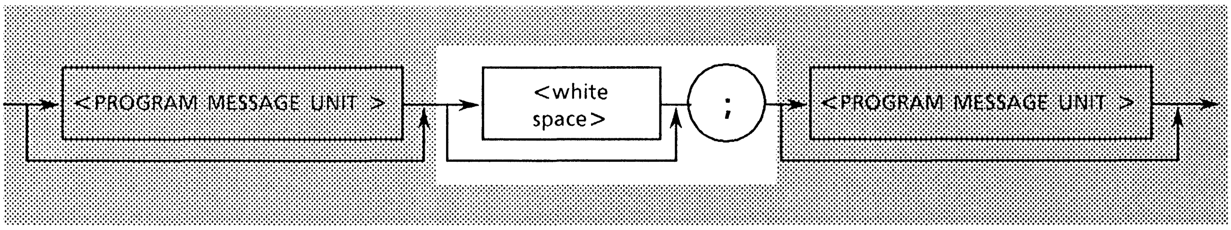
The following explains program message functional element and program data formats (Compound and common commands have been omitted)

5.1.1 Separators, terminators and spaces before headers

(1) Program message unit separators

The format for separating program message units is optional space(s) + semicolon.

Example 1: General format for separating two program message units



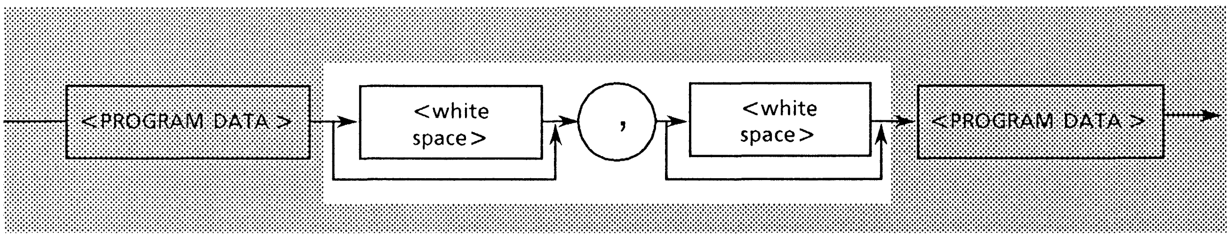
Example 2: 1 space + semicolon

```
DTM Ø△; CTM Ø
```

(2) Program data separators

The format for separating program data items is optional space(s) + comma + optional space(s).

Example 1: General format for separating 2 items of program data



Example 2: Comma only

```
WRT 1,Ø
```

Example 3: Comma + 1 space

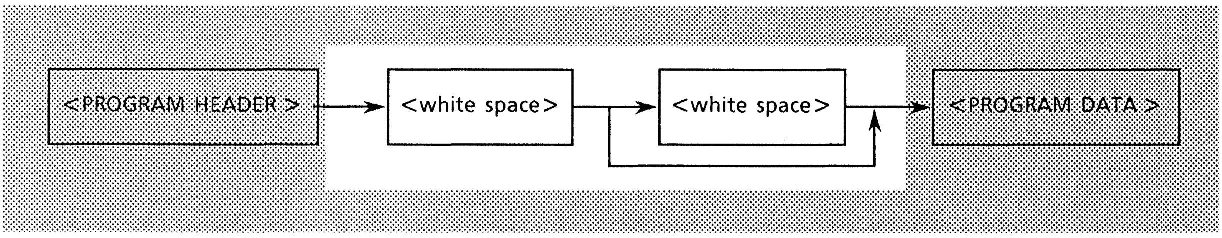
```
WRT 1,△Ø
```

(3) Program header separators

The format for separating a program header from program data is:

1 space + optional space(s).

Example 1: General format for single command program header



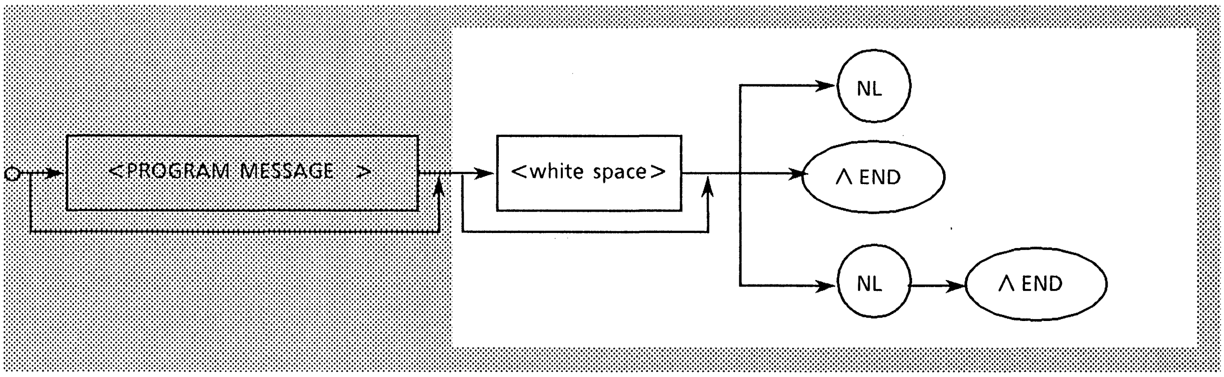
Example 2: 1 space

DTM△Ø

(4) Program message terminators

The format for the terminator at the end of a program message is:
optional space(s) + any of NL, EOI or NL + EOI

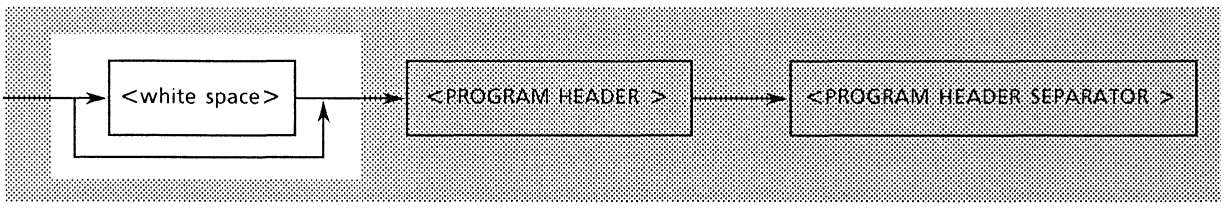
General format:



(5) Spaces before headers

An optional space may be placed before a program header.

General format:

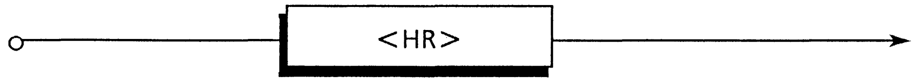


Example: 1 space is placed before the second program header **SPF**.

DTM Ø;△CTM Ø

5.1.2 General format for program command messages

(1) Messages not accompanied by data

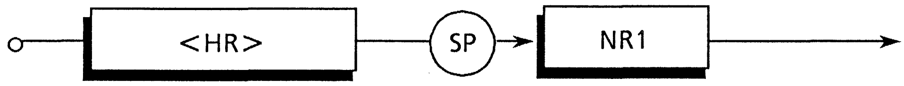


HR : COMMAND PROGRAM HEADER

Examples:

INI Initializes setting

(2) Messages accompanied by integer data

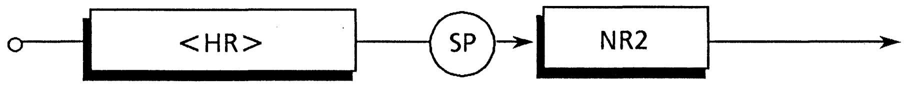


NR1 : integer

Example:

DMS△0 Sets Error measurement display unit Ratio
DMS△1 Sets Error measurement display unit Count
DMS△2 Sets Error measurement display unit EI
DMS△3 Sets Error measurement display unit % EF1
DMS△4 Sets Error measurement display unit CLOCK FREQUENCY

(3) Messages accompanied by real numbers

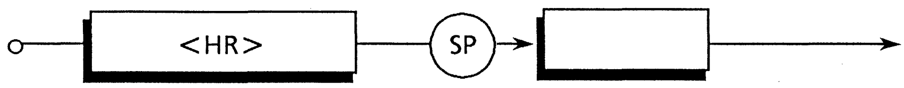


NR2 : real number

Example:

DTH△-3.000 Sets input data threshould level.

(4) Messages accompanied by HEX (hexadecimal)

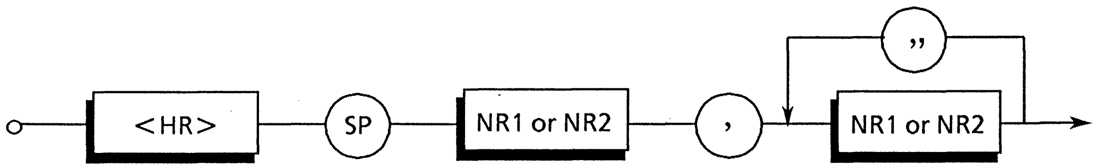


: HEX

Example:

BIT△#H FFFF

(5) Messages accompanied by multiple program data items



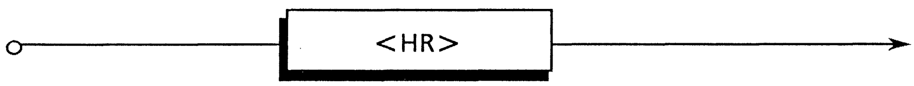
Example:

PRD△99,23,59,59 Sets measurement time to 99 days 23 hours 59 minutes 59 seconds.

5.1.3 General format for query messages

A query program header is indicated by placing a ? at the end of a command program header.

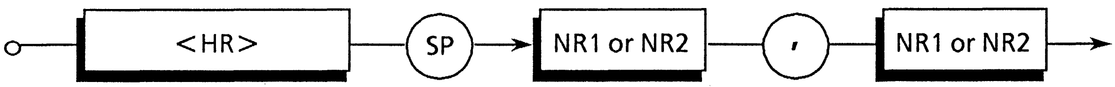
(1) Messages not accompanied by query data



Example :

DTM? Requests data input termination voltage data

(2) Messages accompanied by query data



Example:

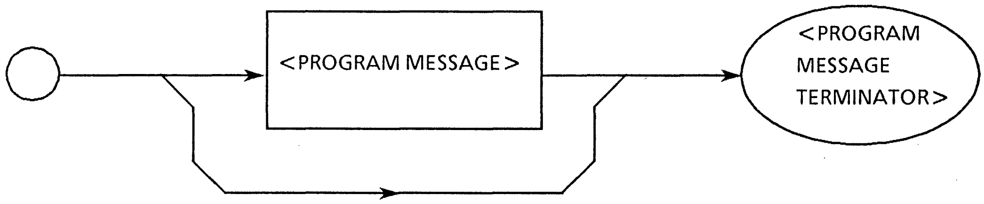
FSH? 1 Requests file information whose file No. is from 51 in the files saving measurement conditions in a floppy disk.

5.2 Functional Elements of Program Messages

A device accepts a program message by detecting the terminator at the end of it. The functional elements of program messages are explained below.

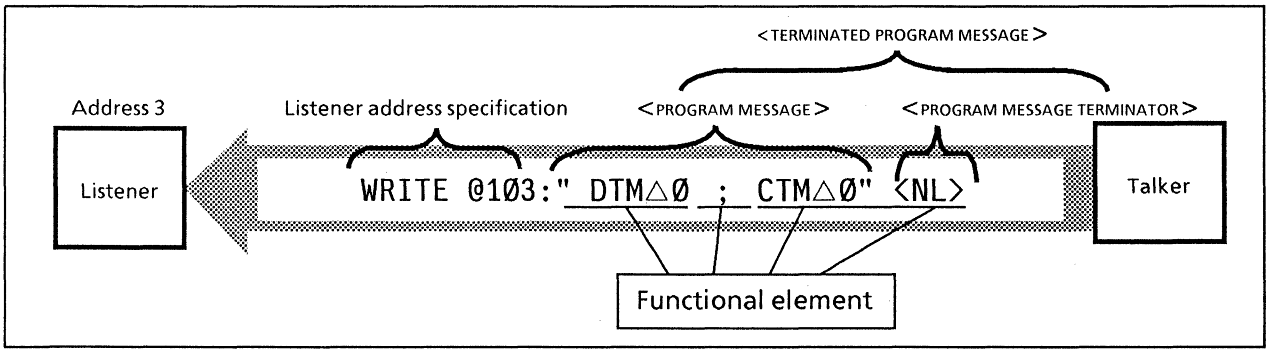
5.2.1 <TERMINATED PROGRAM MESSAGE>

A <TERMINATED PROGRAM MESSAGE> is defined as follows.



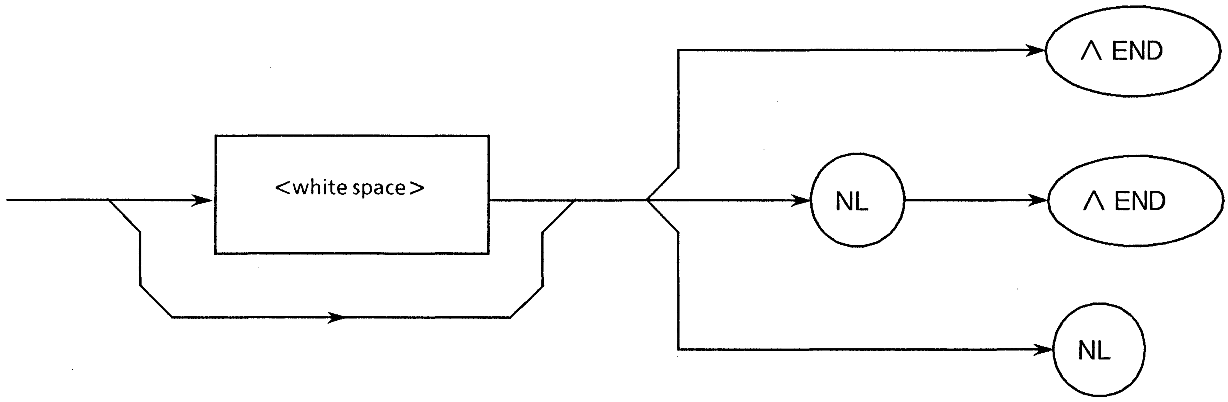
A <TERMINATED PROGRAM MESSAGE> is a data message which has all the functional elements required for transmission from the controller to a listener device. A <PROGRAM MESSAGE TERMINATOR> is attached to the end of a <PROGRAM MESSAGE> to terminate its transmission.

Example: < TERMINATED PROGRAM MESSAGE> which sends 2 commands with a WRITE statement.



5.2.2 <PROGRAM MESSAGE TERMINATOR>

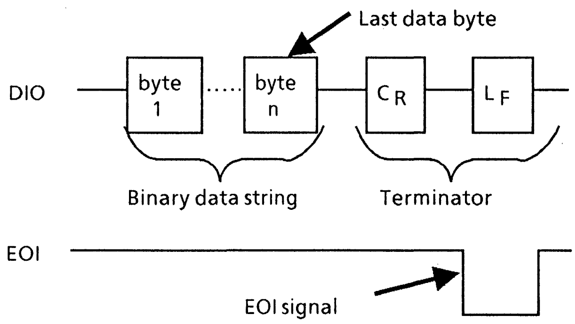
A <PROGRAM MESSAGE TERMINATOR> is defined as follows



A <PROGRAM MESSAGE TERMINATOR> terminates a sequence of one or more <PROGRAM MESSAGE UNIT> elements of a fixed length.

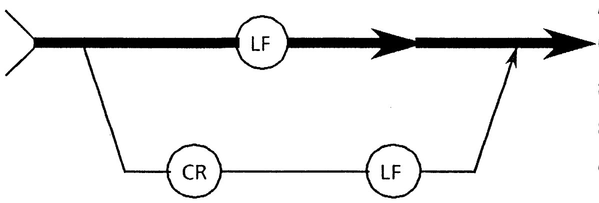
NL: NL is defined as a single ASCII code byte (decimal 10), i.e. the ASCII control code LF (Line Feed) used to return the carriage and bring the print position to the beginning of the next line. It is also called NL (New Line). When a <PROGRAM MESSAGE> is sent by a **WRITE@** statement, there is no need to write the generation of CR.LF code into programs because it is automatically sent by this statement. To generate LF code only, the following statement is executed at the beginning of a program: `TERM IS CHR$(10)`

END: The EOI signal can be generated by making the EOI line (one of GPIB management bus lines) true (low level).



EOI ON / OFF is one statement for controlling the EOI line. The default is **EOI OFF** which means that the EOI line is not controlled. Specifying **EOI ON** causes an EOI signal to be transmitted at the same time as terminator LF when the last data byte of the **WRITE@** statement is transmitted.

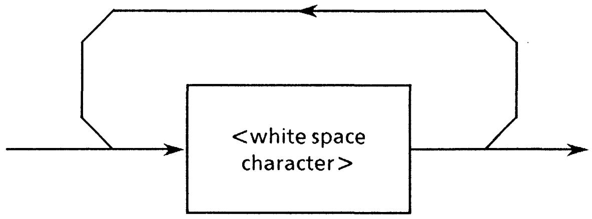
A <PROGRAM MESSAGE> may also be terminated, without sending LF, by using an END signal only.



Note **:**
CR returns the carriage to the beginning of the same line, but is generally ignored on the listener side. However, because there is a lot of equipment already on the market which uses CR and LF code, most controllers are designed to output LF code following CR code.

5.2.3 <white space>

A <white space> is defined as follows.

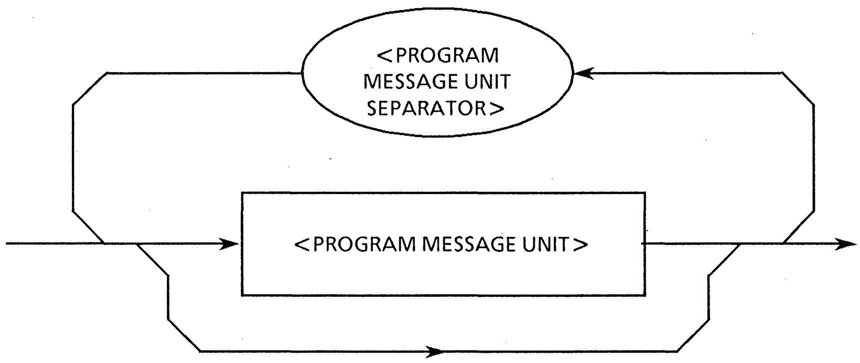


A <white space character> is defined as a single ASCII code byte in the range 00 to 09, 0B to 20 (decimal 0 to 9, 11 to 32).

This range includes ASCII control signals and space signal except new line. A device either treats them as ASCII control signals but as spaces, or skips over them.

5.2.4 <PROGRAM MESSAGE>

A <PROGRAM MESSAGE> is defined as follows.



A <PROGRAM MESSAGE> consists of zeros, or a sequence of one or several <PROGRAM MESSAGE UNIT> elements. <PROGRAM MESSAGE UNIT> elements are either programming commands or data sent from the controller to devices. The <PROGRAM MESSAGE UNIT SEPARATOR> element is used to separate <PROGRAM MESSAGE UNITS>.

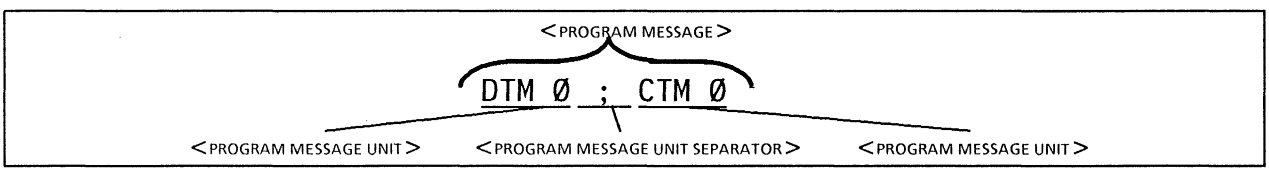
Example 1:

The program message which sets the data input termination voltage to GND.

```
DTM Ø
```

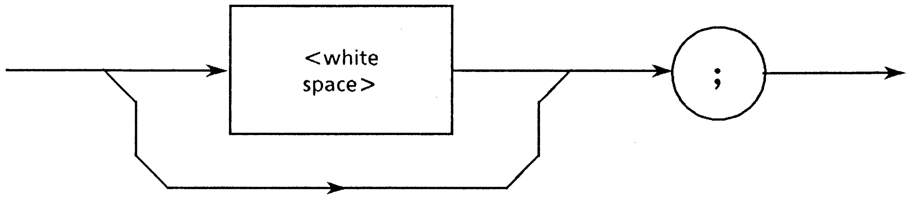
Example 2:

The program message which sets as same as the Example 1, and then sets the clock input termination voltage to GND.

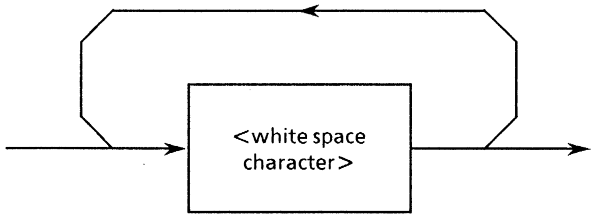


5.2.5 <PROGRAM MESSAGE UNIT SEPARATOR>

A <PROGRAM MESSAGE UNIT SEPARATOR> is defined as follows.



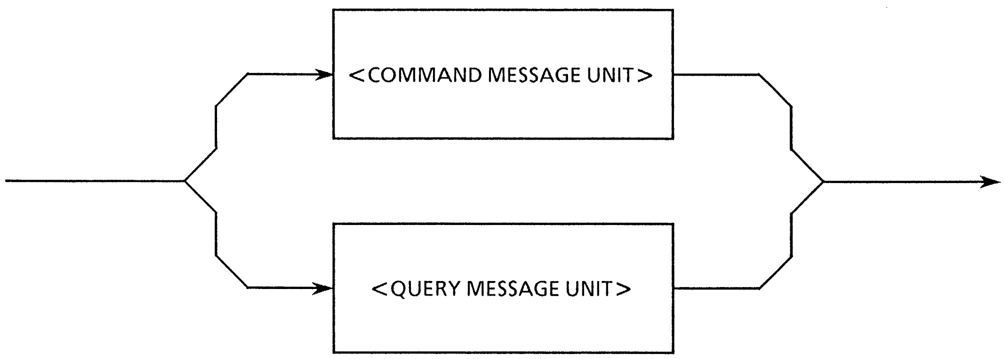
<white space> is defined as follows.



The <PROGRAM MESSAGE UNIT SEPARATOR> separates the <PROGRAM MESSAGE UNIT> elements in a <PROGRAM MESSAGE>. A device interprets a semicolon as the separator of <PROGRAM MESSAGE UNIT> elements so, it skips the <white space characters> before and after the semicolon. <white space characters> make a program easy to read. If there is one after a semicolon, it is the <white space> for the next program header.

5.2.6 <PROGRAM MESSAGE UNIT>

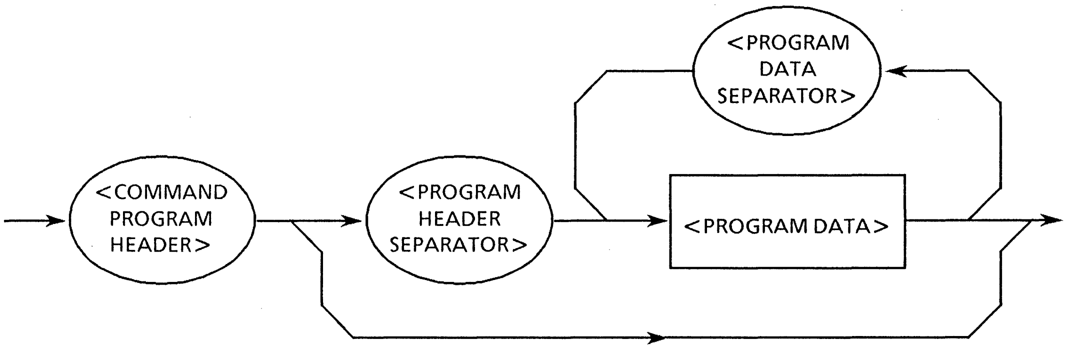
A <PROGRAM MESSAGE UNIT> is defined as follows.

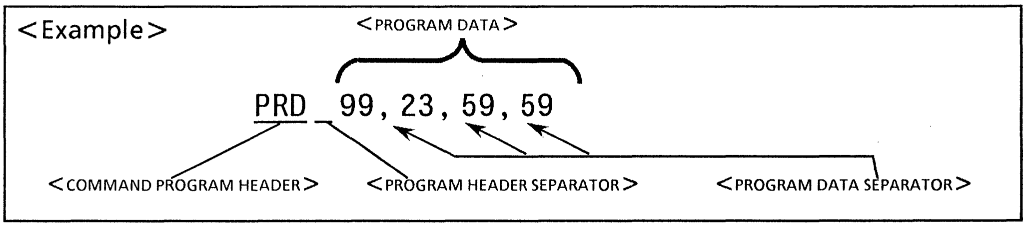


A <PROGRAM MESSAGE UNIT> is either the <COMMAND MESSAGE UNIT> or <QUERY MESSAGE UNIT> received by a device. <COMMAND MESSAGE UNITS> and <QUERY MESSAGE UNITS> are explained in detail on the next page.

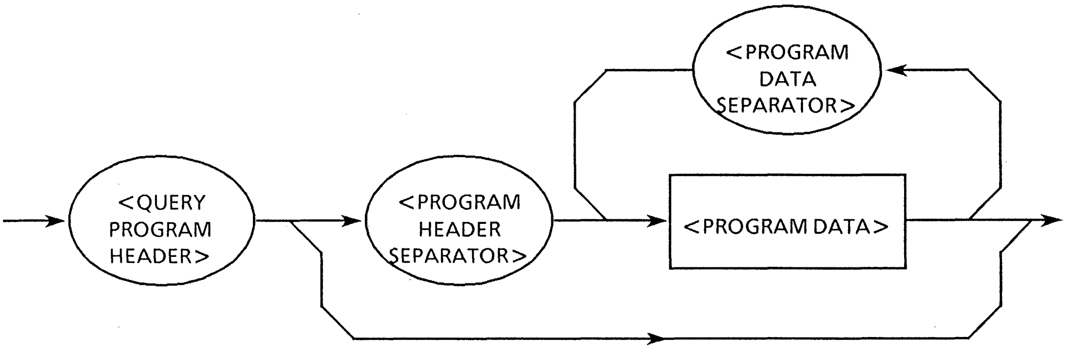
5.2.7 <COMMAND MESSAGE UNIT> and <QUERY MESSAGE UNIT>

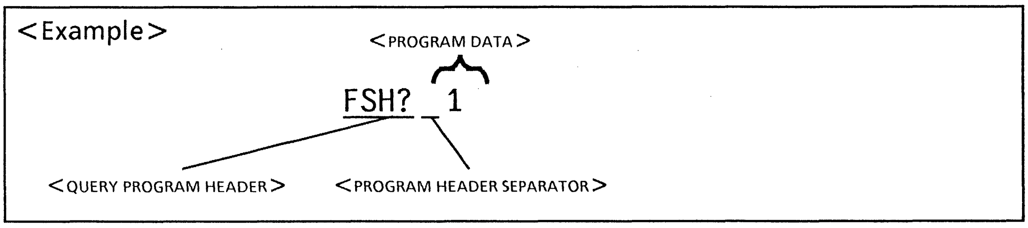
1) A <COMMAND MESSAGE UNIT> is defined as follows.





2) A <QUERY MESSAGE UNIT> is defined as follows.





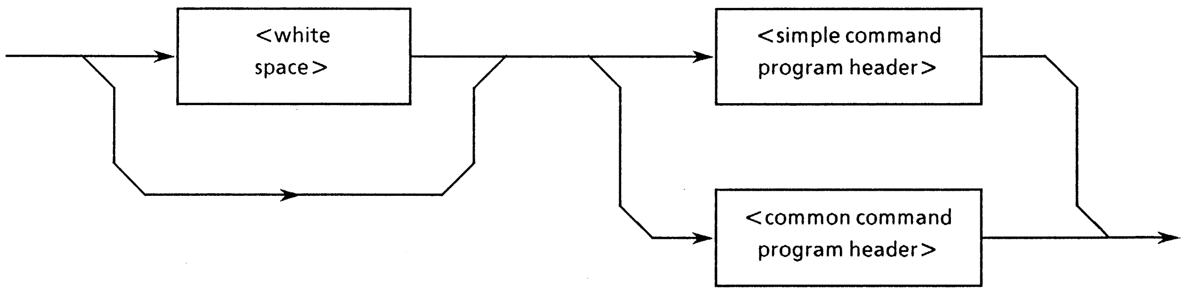
For both <COMMAND MESSAGE UNITS> and <QUERY MESSAGE UNITS>, a space must be inserted between the program header and any program data immediately following it. The application, function and operation of the program data can be seen from the program header. If there is no program data; the application, function or operation to be performed by a device is indicated by the header alone.

The <COMMAND PROGRAM HEADER> is a command by which the controller controls a device. <QUERY PROGRAM HEADER> is a command used for sending a query from the controller to a device so that the controller can receive a response message from it.

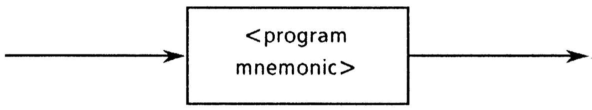
The special feature of the header is that a question mark is always tagged on at the end to indicate that it is a query.

5.2.8 <COMMAND PROGRAM HEADER>

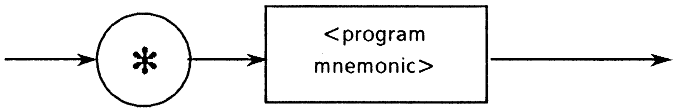
A <COMMAND PROGRAM HEADER> is defined as follows. A <white space> may be placed in front of each header.



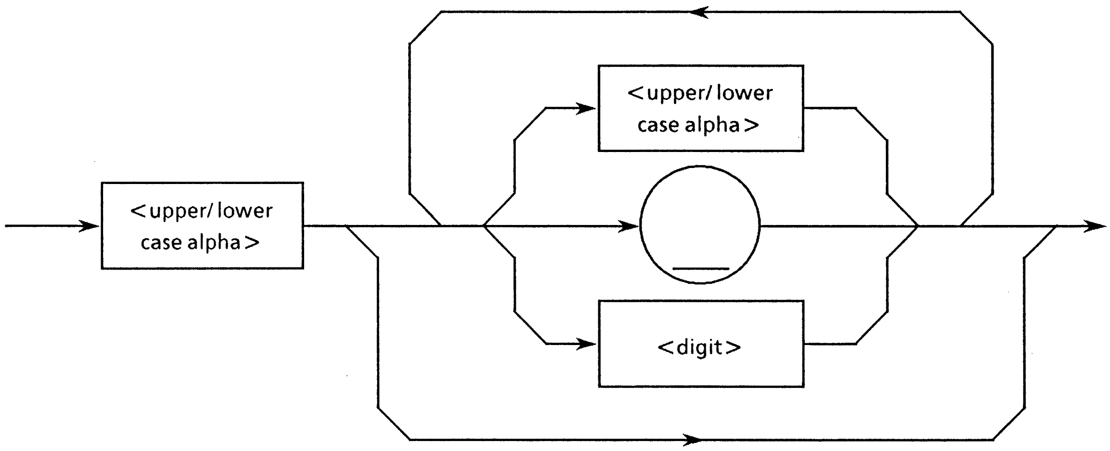
1) A <simple command program header> is defined as follows.



2) A <common command program header> is defined as follows.



3) A <program mnemonic> is defined as follows.



■ <COMMAND PROGRAM HEADER>

Indicates the application, function and operation of a program to be executed by a device. If there is no program data; the application, function and operation to be executed by the device are indicated in the header itself. This is expressed in ASCII code characters by a <program mnemonic>, usually called just mnemonic.

The following explains items 1), 2) and 3) above and the definition of mnemonics.

■ <program mnemonic>

A mnemonic must begin with upper-case or lower-case alphabetic characters. Following that, upper-case alphabetic characters from A to Z, lower-case alphabetic characters, the underline and numbers from 1 to 9 can be used in any combination. The maximum length of a mnemonic is 12 characters but they usually consist of 3 to 4 upper-case alphabetic characters. There are no spaces between characters.

- <upper / lower-case alpha>

  Defined as a single ASCII code byte in the range 41 to 5A, 61 to 7A (decimal 65 to 90, 97 to 122 = A to Z, a to z).

- <digit>

  Digits are defined as single ASCII code byte in the range 30 to 39 (decimal 48 to 57 = numeric 0 to 9).

- (_)

  The underline is defined as the single ASCII code byte 5F (decimal 95).

■ <simple command program header>

The above definition for <program mnemonic> is used as it is.

■ <common command program header>

An asterisk is always placed before the <program mnemonic> in a <common command program header>. The word 'common' is used to indicate that the <common command program header> is applicable to all other measuring instruments conforming to the IEEE 488.2 standard connected to the bus.

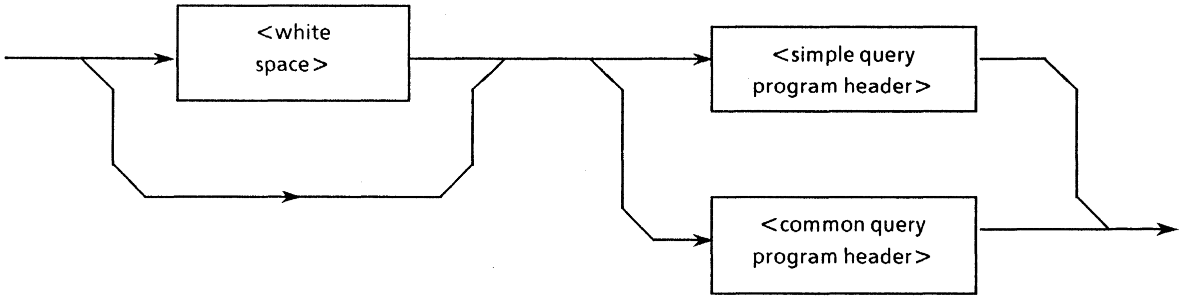
- Example

  The operation (of the device with address 3 connected to the select code 1 GPIB interface) is terminated and it is put in the idle state; then each device is reset to the initial state stipulated for it.

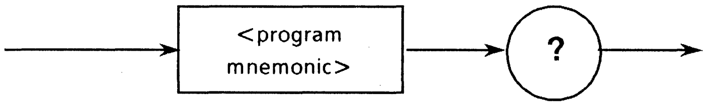
  `WRITE @103:"*RST":` .... *RST is the common IEEE 488.2 command which executes the above.

5.2.9 <QUERY PROGRAM HEADER>

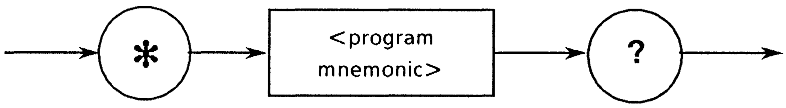
A <QUERY PROGRAM HEADER> is defined as follows. A <white space> is placed before each header.



1) A <simple query program header> is defined as follows.



2) A <common query program header> is defined as follows.



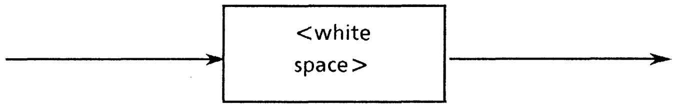
■ <QUERY PROGRAM HEADER>

A <QUERY PROGRAM HEADER> is a command for sending a query from the controller to a device so that the controller can receive a response message from it. A ? is always added at the end of the header to indicate a query.

☞ Except for the ? after it, the format of the <QUERY PROGRAM HEADER> is identical to that of the <COMMAND PROGRAM HEADER>.

5.2.10 <PROGRAM HEADER SEPARATOR>

A <PROGRAM HEADER SEPARATOR> is defined as follows.

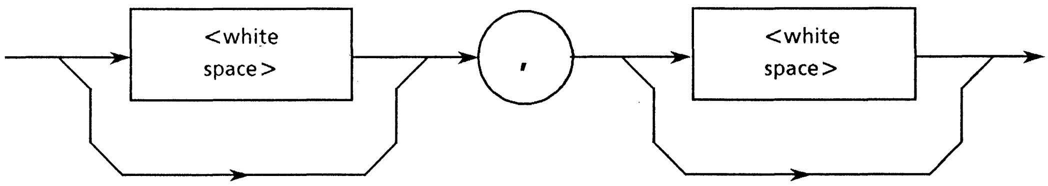


A <PROGRAM HEADER SEPARATOR> is used to separate a <COMMAND PROGRAM HEADER> or <QUERY PROGRAM HEADER> from <PROGRAM DATA>. When there is more than one <white space character> between a program header and program data, the first is interpreted as the separator and the rest are skipped. <white space characters> are used to make a program easy to read.

So, there must always be one header separator between the header and the data to indicate the end of the program header and the start of the program data.

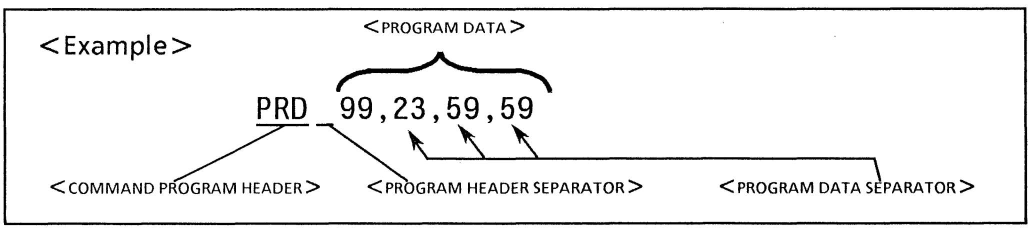
5.2.11 <PROGRAM DATA SEPARATOR>

A <PROGRAM DATA SEPARATOR> is defined as follows.



When a <COMMAND PROGRAM HEADER> or <QUERY PROGRAM HEADER> has many parameters, A <PROGRAM DATA SEPARATOR> is used to separate them.

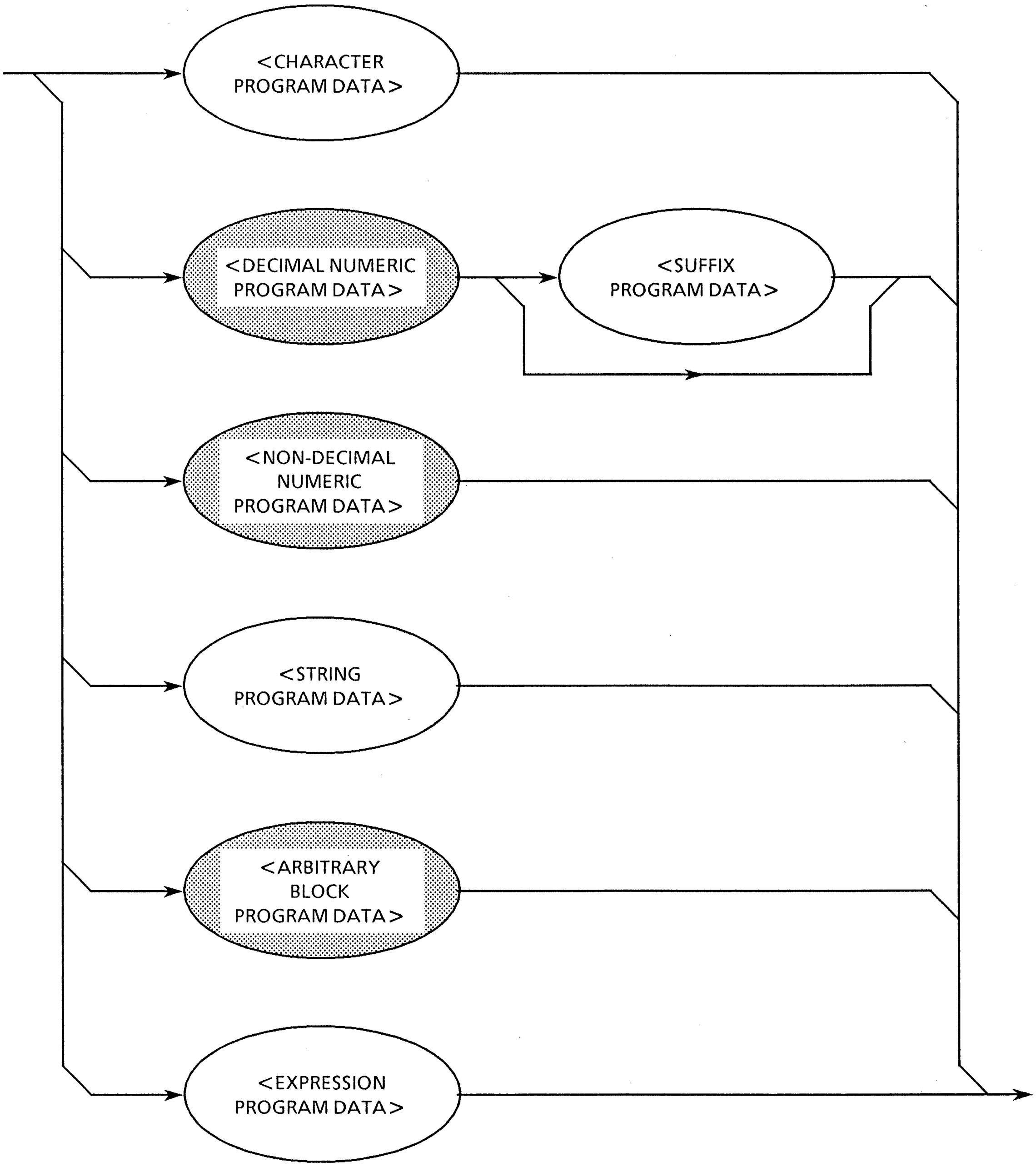
A comma must be used with a <PROGRAM DATA SEPARATOR>, but a white space does not always have to be used. A white space before or after the comma is skipped. They are used to make a program easier to read.



5.3 Program Data Format

The following describes the format of <PROGRAM DATA>.

<PROGRAM DATA> functional elements are used in sending various types of parameter related to the program header. The diagram below shows the different types of program data. The MP1764A accepts the data types in the shaded ovals.



5.3.1 <DECIMAL NUMERIC PROGRAM DATA>

<DECIMAL NUMERIC PROGRAM DATA> is program data for sending numeric contents expressed in decimal notation. There are 3 formats for expressing decimal numbers: integer format, fixed point format and floating point format. The MP1764A does not use the floating point format.

The program data transmission in the integer or fixed point formt used in the MP1764A is described.

Note : The data will processed at any data format in the manner described below.

- Rounding off of numeric elements

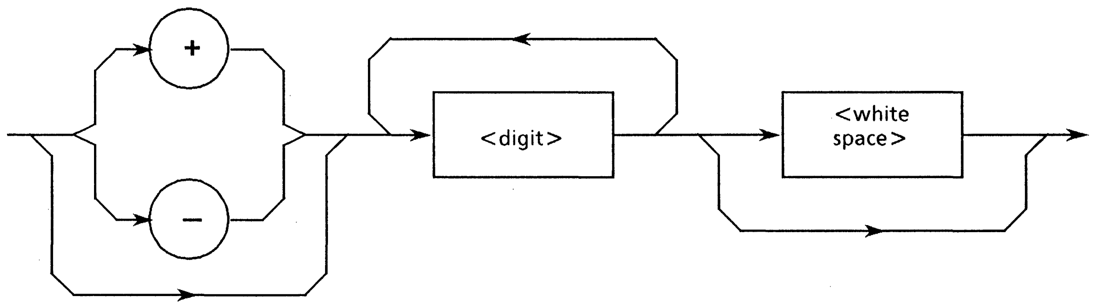
  When a device receives <DECIMAL NUMERIC PROGRAM DATA> elements with more digits than it can handle, it ignores the sign and rounds it off to the nearest whole number.

- Outside-range data

  When a <DECIMAL NUMERIC PROGRAM DATA> element is outside the permissible range for the program header, execution error is reported.

(1) Integer format - NR1 transmission

In the diagram below, an integer NR1, i.e. a decimal number which does not contain a floating point or exponential expression, is transmitted.

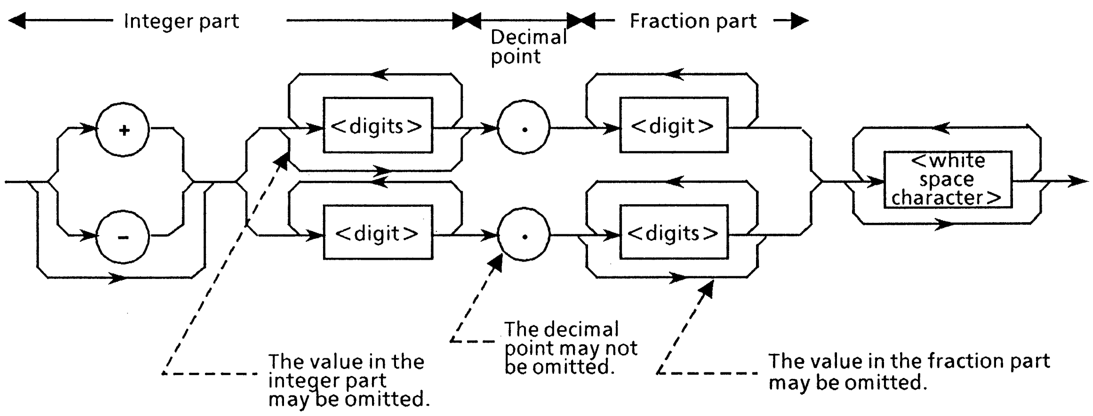


- Zeros can be inserted at the beginning. → 005, +000045
- Spaces cannot be inserted between a + or − sign and a number → +5, +△5 (X)

  X: not allowable
- Spaces can be inserted after a number. → +5△△△
- The + sign is optional. → +5, 5
- Commas may not be used to separate digits → 1,234,567 (X)

(2) Fixed point format - NR2 transmission

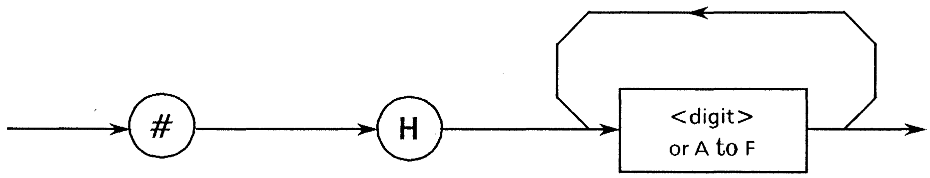
The example below shows the transmission of NR2, a real number with no integer or exponential expressions having digits after the decimal point. The syntax diagram consists of an integer part, the decimal point and a fraction part.



- The numeric expression of the integer format is applied to the integer part.
- No spaces may be inserted between numbers and the decimal point → +753△.123 (X)
  X: not allowable
- Spaces may be inserted after the fraction part → +753.123△△△△
- There need not be any numbers before the decimal point → .05
- A + or − sign can be inserted before the decimal point → +.05, −.05
- A number can end in a decimal point → 12.

5.3.2 <NON-DECIMAL NUMERIC PROGRAM DATA>

<NON-DECIMAL NUMERIC PROGRAM DATA> is program data for sending hexadecimal value data as non-decimal numeric data. The non-decimal data always begins from the # mark. The non-decimal data is defined as a coded syntax diagram shown in the below. When strings except for a specified character string is sent,a command error generates.



Characters followed by #H are received at the device as an unsigned hexadecimal numeric. Characters in the ( ) means corresponding decimal numbers.

Example:

The program message which sets the data input timing voltage to GND.

#HABCD (43, 981 D)

SECTION 6
TALKER OUTPUT FORMAT

Two types of data messages are transmitted between the controller and a device via the system interface when the bus is in the data mode, i.e. when the ATN line is false: program messages and response messages. This section describes the format of the response messages sent by a talker device to the controller.

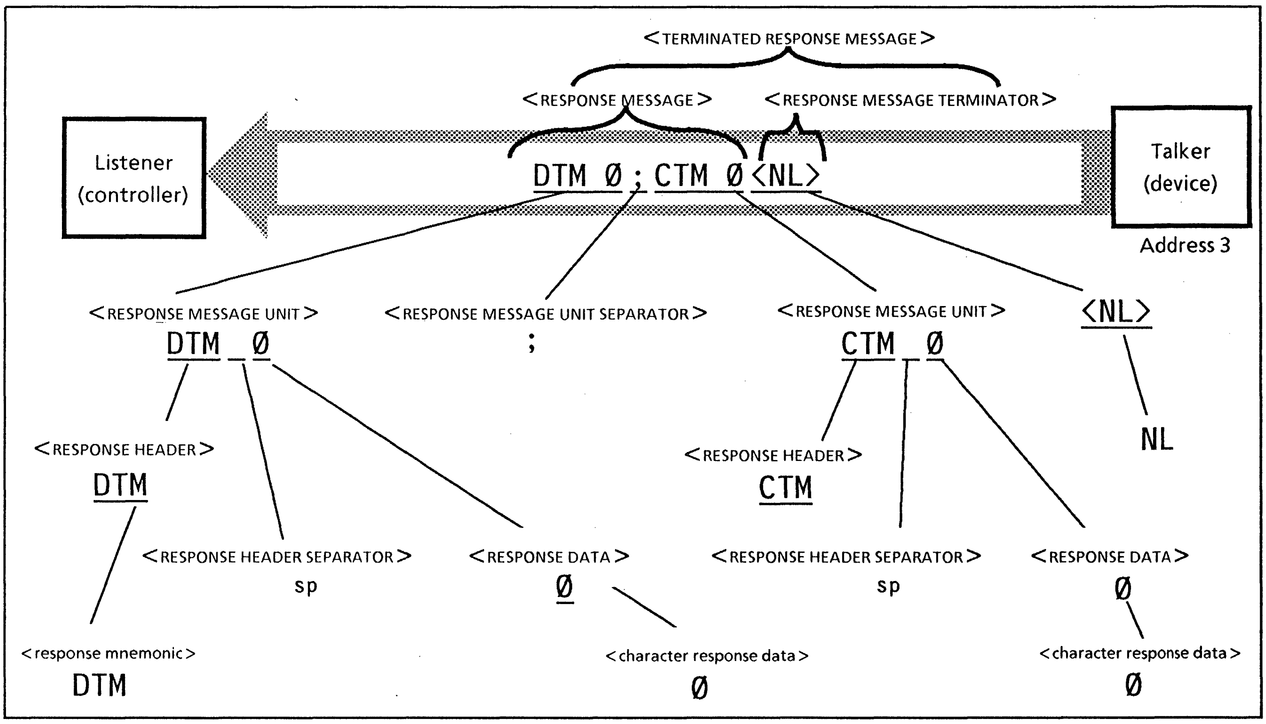
Control commands by ANRITSU PACKET V series personal computers are applied for formats and use examples in this section.

TABLE OF CONTENTS

(Blank)

Response messages convey measured results, setting conditions and status information. Some response messages have a header, and others not.

The diagram below, as an example, shows each response message is sent from a device to a controller as an ASCII character string with a header for a data input termination voltage query message unit **DTM?** and a clock input termination voltage query message unit **CTM?**.



The program for the above would be as follows:

`100 WRITE @103:"DTM? "!`	Data input termination voltage query message request
`110 READ @103:A$!`	When the terminator NL is detected, the response message DTM△Ø is read into A$.
`120 WRITE @103:"CTM? "!`	Clock input termination voltage query message request
`130 READ @103:B$!`	Clock input termination volgage response message CTM△Ø

As for program messages, response messages are made up of a sequence of functional elements which are the minimum unit capable of expressing function. The upper-case alphabetic character items inside < > in the diagram above are examples of functional elements. Functional elements can be further subdivided into coded elements. The lower-case alphabetic character items inside < > in the diagram above are examples of coded elements. Thus, the way of expressing items on functional syntax diagrams is the same for talker and listener.

The following pages explain the talker device output format focussing on the differences between it and the listener device input format.

6.1 Syntax Differences Between Formats of Listener Input and Talker Output

The differences in syntax between listener device input and talker device output formats are:

- Listener format

  There is flexibility in writing programs to make program messages (from the controller) easy to receive by the listener. Consequently, program messages can perform the same function despite differences in message description between them. For example, the free insertion of <white spaces> in separators and terminators makes programs easy to read.

- Talker format

  Strict rules govern the syntax of response messages sent from device to controller to make them easy to receive. Thus, in contrast to the listener format, there is only one notation for each function in the talker format.

The table below summarizes the differences between the listener and talker formats. Space in the table means <white space>.

Item	Listener-input program message syntax	Talker-output response message syntax
Characteristics	(Flexible)	(Strict)
Alphabetic characters	Either upper or lower-case characters can be used. Only upper-case for header.	Upper-case only
Before / after E in NR3 exponent	Optional space(s) + E / e + optional space(s) Not supported by the MS2802A.	Upper-case E only
+ sign in NR3 exponent	Can be omitted. Not supported by the MS2802A.	Cannot be omitted
<white space>	Two or more spaces can be placed before or after a separtor and before a temrinator.	Not used
Message unit	① Header with program data ② Header without program data	① Data with header ② Data without header
Unit separator	Optional space(s) + semicolon	Semicolon only
Space before header	Optional space(s) + header	Header only
Header spearator	Header + 1 or more spaces	Header + one $20*
Data separator	Optional space(s) + comma + optional space(s)	Comma only
Terminator	Optional space(s) + any of NL, EOI or NL + EOI	NL + EOI

* ASCII code byte 20 (decimal 32 = ASCII character SP: space)

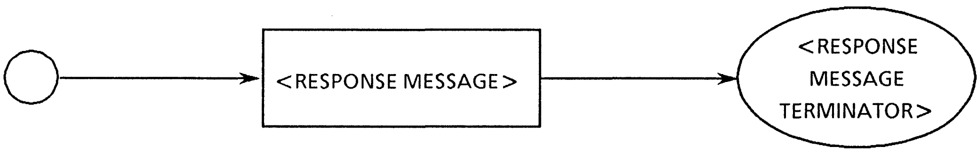
6.2 Functional Elements of Response Message

Response messages output by the talker are accepted by the controller once they have been terminated by the NL END signal. The following describes the functional elements of the response message.

As the rules for syntax diagram notation are the same as for program messages, refer to Section 5 for the details. The explanation of functional elements and encoded elements has been omitted where it would overlap with that for program messages. Refer to Section 5 as required.

6.2.1 <TERMINATED RESPONSE MESSAGE>

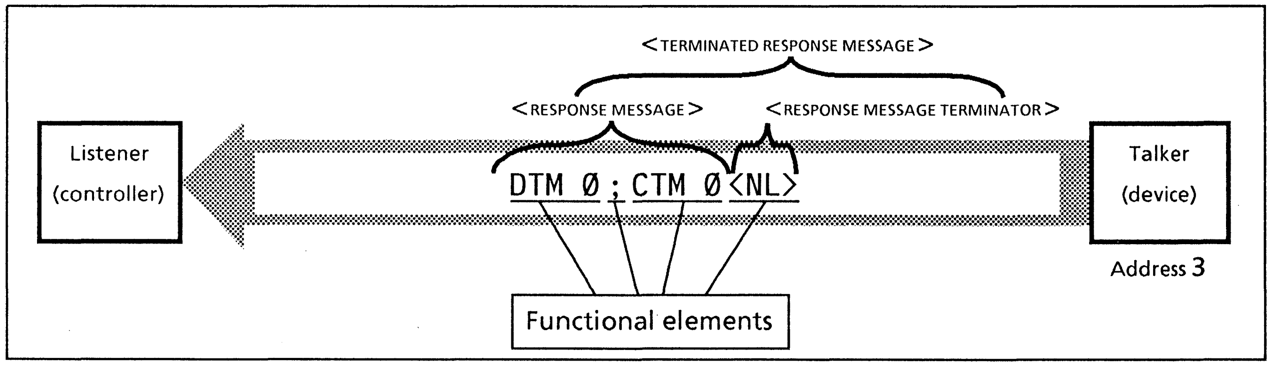
A <TERMINATED RESPONSE MESSAGE> is defined as follows:



A <TERMINATED RESPONSE MESSAGE> is a data message, containing all the functional elements required for transmission, sent from a talker device to the controller.

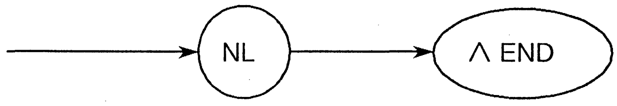
A <RESPONSE MESSAGE TERMINATOR> is attached to the end of a <RESPONSE MESSAGE> to terminate its transmission.

Example: A <TERMINATED RESPONSE MESSAGE> comprising 2 message units



6.2.2 <RESPONSE MESSAGE TERMINATOR>

A <RESPONSE MESSAGE TERMINATOR> is defined as follows.



A <RESPONSE MESSAGE TERMINATOR> is placed after the last <RESPONSE MESSAGE UNIT> to terminate a fixed length sequence consisting of one or more <RESPONSE MESSAGE UNIT> elements.

Executing the following statements listed below for **NL** and **END** at the start of a program outputs terminator LF together with the EOI signal, to indicate the END, when the last data byte is transmitted.

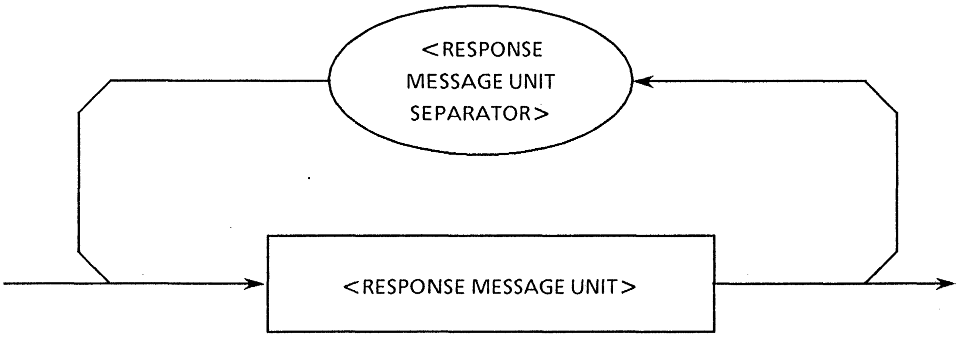
- For NL(LF): `TERM IS CHR$(1Ø)`
- For END (EOI): `EOI ON`

Example: To read the current center frequency setting

```
10 LET ADR=1Ø1
20 TERM IS CHR$(1Ø)!      LF (new line) is assigned as the terminator code.
30 EOI ON!                When the last data byte is transmitted, the EOI signal is sent which makes the EOI line
                          true
40 WRITE @ADR:"DTM?"!     Query to read the data input termination voltage
50 READ @ADR:A$!          EOI signal terminates the reading of response data
60 PRINT A$
70 END
```

6.2.3 <RESPONSE MESSAGE>

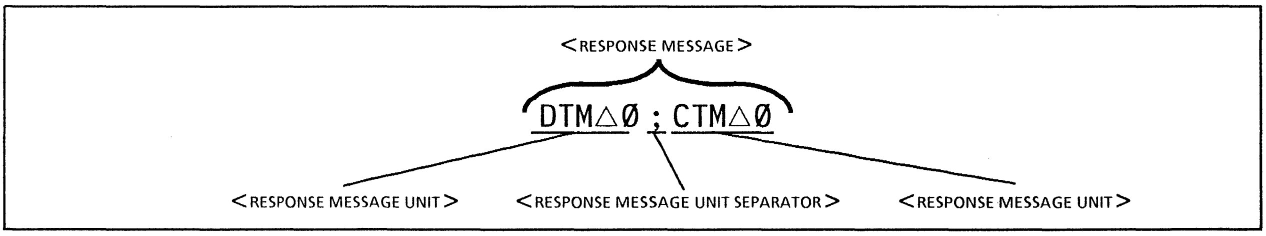
A <RESPONSE MESSAGE> is defined as follows.



A <RESPONSE MESSAGE> consists of one <RESPONSE MESSAGE UNIT> element or a sequence of many <RESPONSE MESSAGE UNIT> elements. A <RESPONSE MESSAGE UNIT> element is a single message sent from a device to the controller. A <RESPONSE MESSAGE UNIT SEPARATOR> element is used to separate <RESPONSE MESSAGE UNIT> elements.

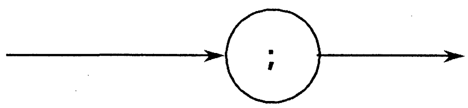
Example:

Attaches the DTM and CTM headers to the data input termination voltage and clock input termination voltage, and transmits them in 1- character fixed format.



6.2.4 <RESPONSE MESSAGE UNIT SEPARATOR>

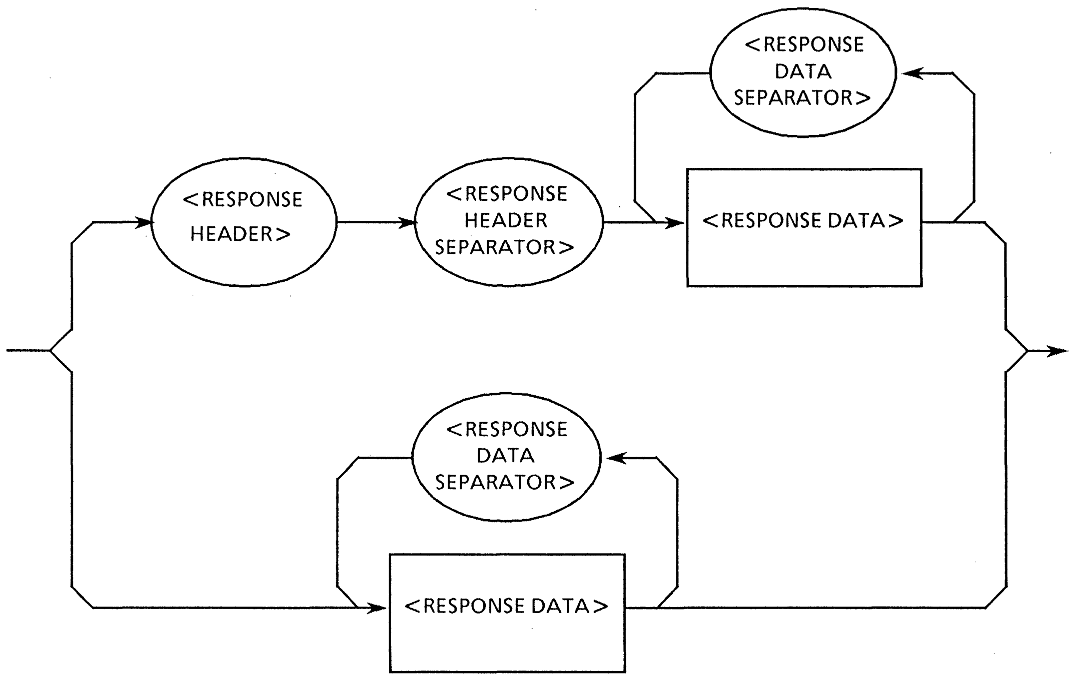
A <RESPONSE MESSAGE UNIT SEPARATOR> is defined as follows.



A semicolon (;) is used as the <RESPONSE MESSAGE SEPARATOR> to separate a sequence of <RESPONSE MESSAGE UNIT> elements that is to be transmitted as one message.

6.2.5 <RESPONSE MESSAGE UNIT>

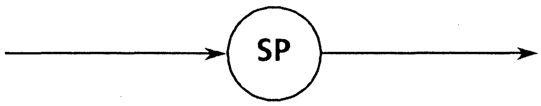
A <RESPONSE MESSAGE UNIT> is defined as follows.



A <RESPONSE MESSAGE UNIT> consists of 2 basic types of syntax. The first is a response message with a header which returns the results of processing data on settings made by program messages. The second is a response message unit without a header which returns only measured results.

6.2.6 <RESPONSE HEADER SEPARATOR>

A <RESPONSE HEADER SEPARATOR> is defined as follows:

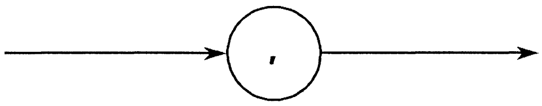


The <RESPONSE HEADER SEPARATOR> is a space after the <RESPONSE HEADER> to separate it from <RESPONSE DATA>. The space, SP, is ASCII code byte 20 (decimal 32).

There is always one space to separate the header from the data in a response message with a header. This space indicates the end of the header and the start of the data.

6.2.7 <RESPONSE DATA SEPARATOR>

A <RESPONSE DATA SEPARATOR> is defined as follows:



A <RESPONSE DATA SEPARATOR> is used to separate <RESPONSE DATA> items when more than one is output.

6.2.8 <RESPONSE HEADER>

With the exception of the following three points, the format of the <RESPONSE HEADER> is the same as that described for the <COMMAND PROGRAM HEADER> in paragraph 5.2.8.

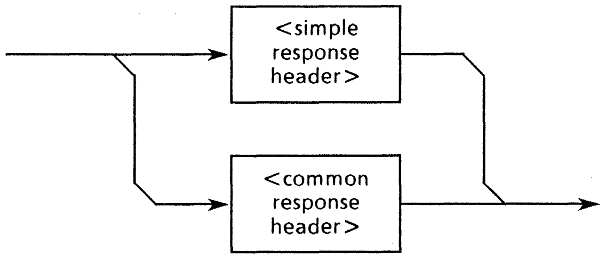
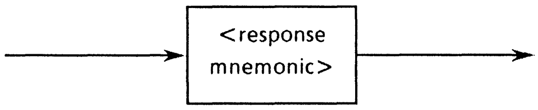
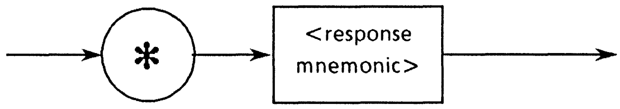
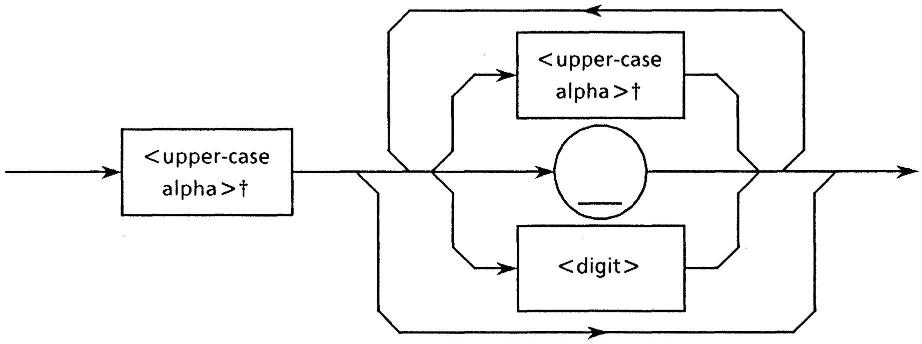
① The <response mnemonic> has a stipulated character set stating that alphabetic characters must be upper-case. Otherwise it is the same as the <program mnemonic> in paragraph 5.2.8.

② Spaces can be placed in front of a program header but cannot be placed in front of a response header.

③ More than one space may be placed after a program header but only one may be placed after a response header.

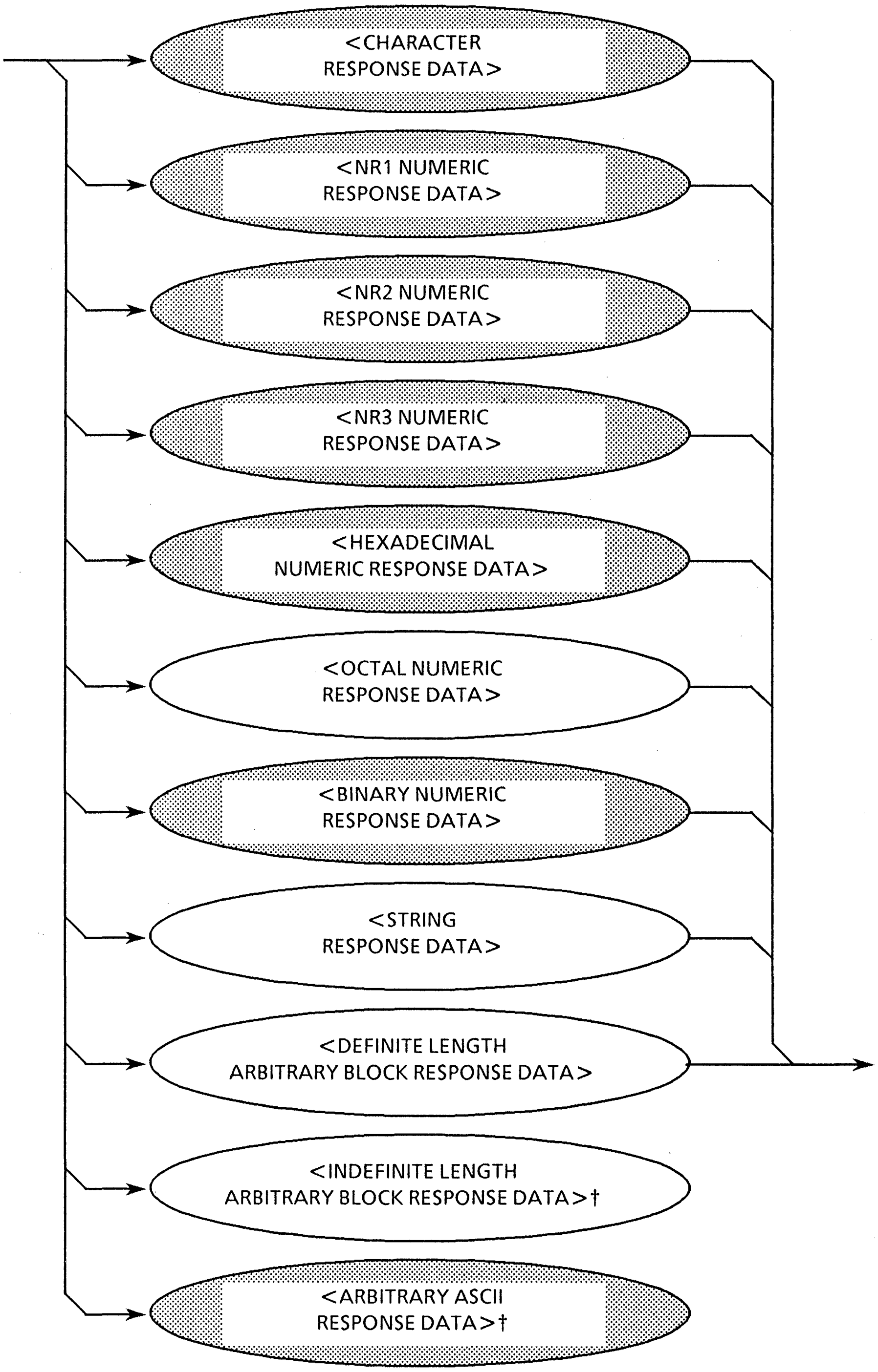
All aspects of the <RESPONSE HEADER> up to the <response mnemonic> are shown on the next page.

☞ For characters used in <response mnemonic>, alphabetic characters are always upper-case characters and other characters are used in the same manner as <response mnemonic>.)

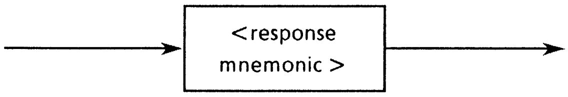
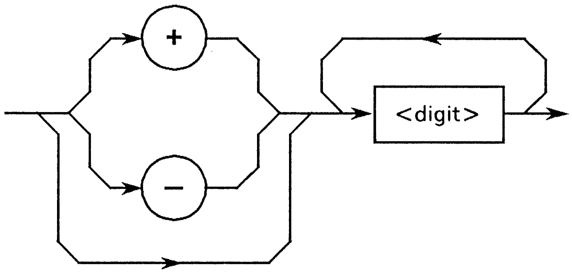
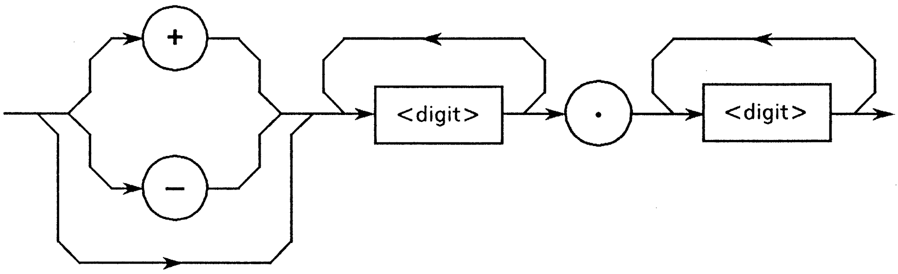
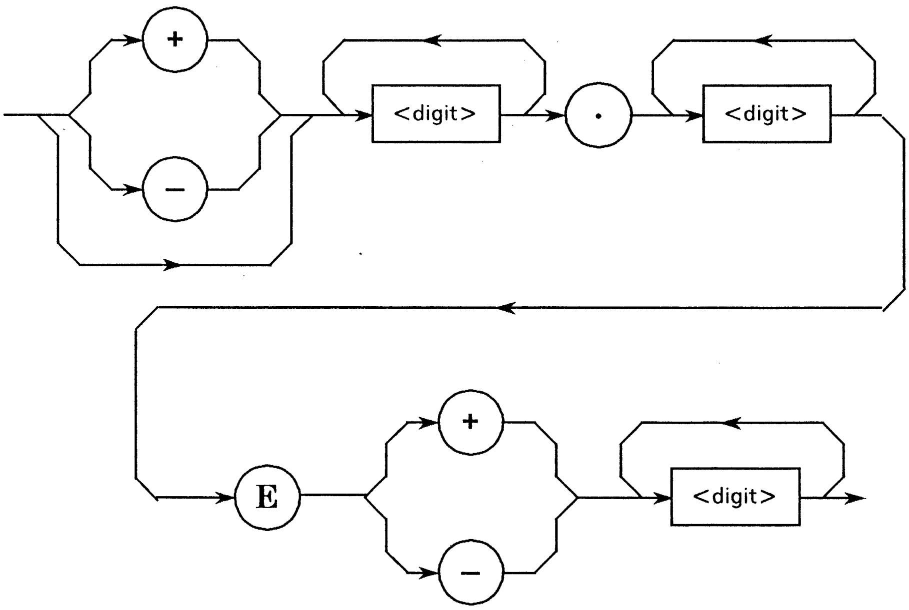
Element	Function
RESPONSE HEADER	The header indicates the function of the response data. Its meaning is shown by a <response mnemonic> which is a combination, of up to 12 characters, of upper-case alphabetic characters, numbers and underlines starting with an upper-case alphabetic character.  1) A <simple response header> is defined as follows:  2) A <common response header> is defined as follows:  3) A <response mnemonic> is defined as follows:  †<upper-case alpha> ASCII code bytes 41 to 5A (decimal 65 to 90 = upper-case alphabetic A to Z)

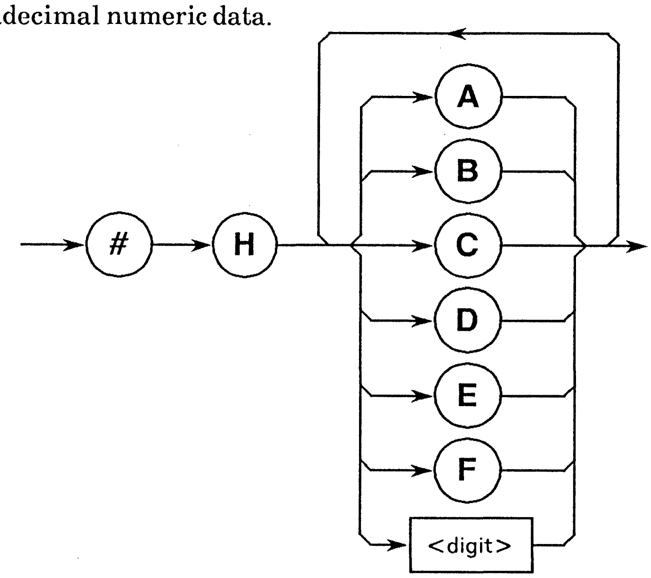
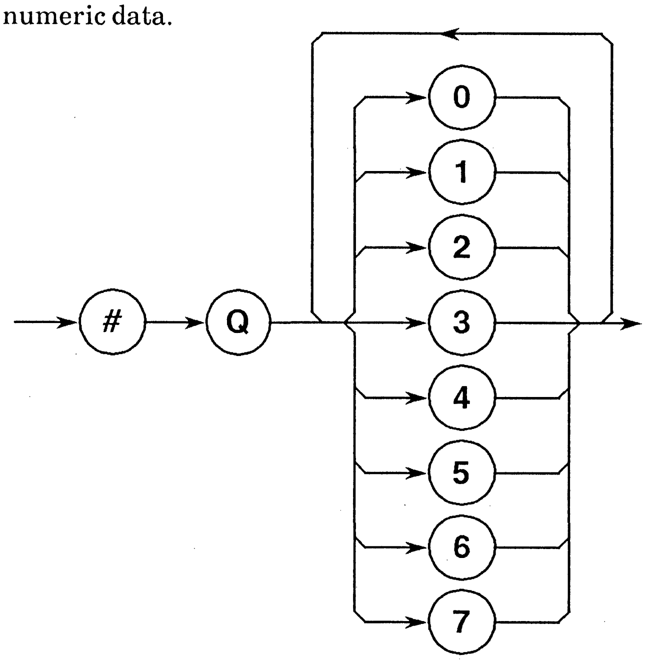
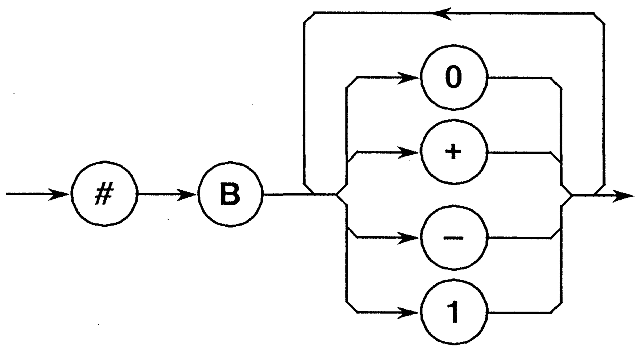
6.2.9 <RESPONSE DATA>

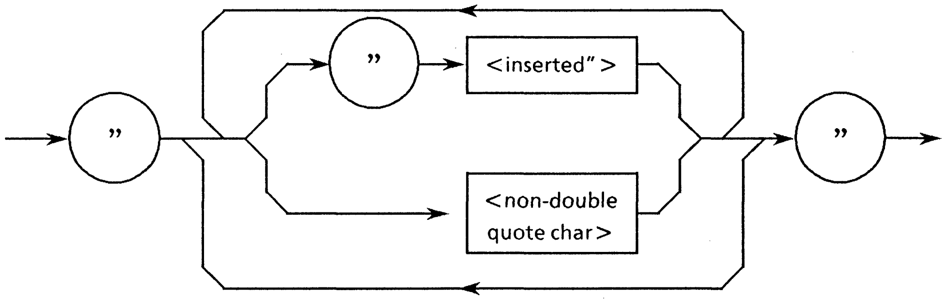
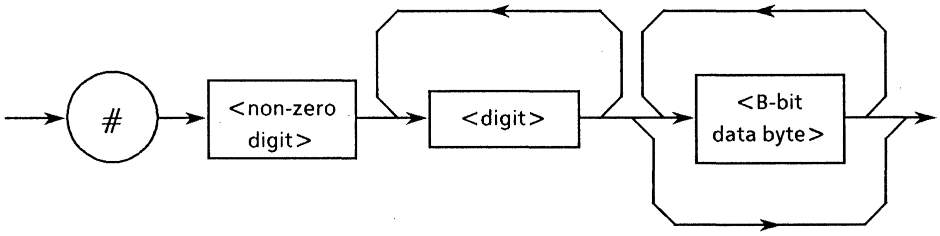
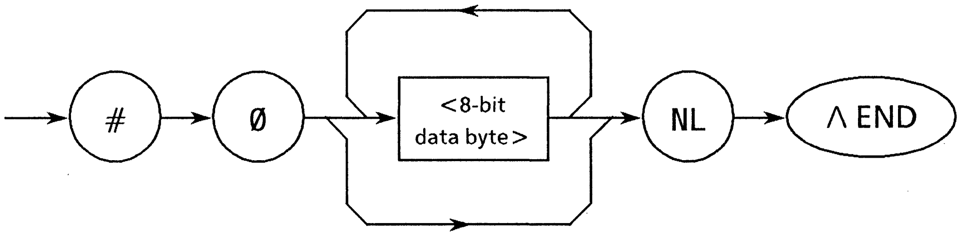
The diagram below shows the 11 types of response data. The MP1764A supports the response data in the shaded ovals below. The type of response data to be returned is determined by the query message.

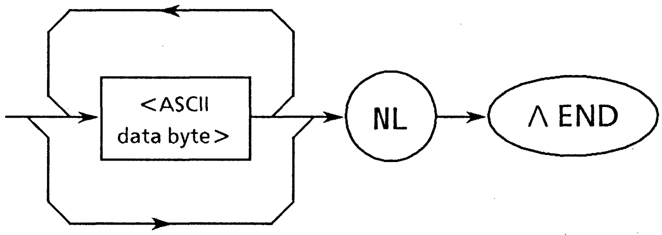


† Both <INDEFINITE LENGTH ARBITRARY BLOCK RESPONSE DATA> and <ARBITRARY ASCII RESPONSE DATA> are terminated by an NL^END in their own last data byte.

Element	Function
(1) CHARACTER RESPONSE DATA <Example> AAT2_AUTO AAT2_MANUAL	Data composed of character strings common with <response mnemonic>. Thus the beginning of the character string is always an upper-case alphabetic character and the character string length is limited to 12 characters. Numeric parameters are not suitable for being used. 
(2) NR1 NUMERIC RESPONSE DATA <Example> 123 +123 −1234	Integer data, i.e. decimal values without a decimal point or exponents. 
(3) NR2 NUMERIC RESPONSE DATA <Example> 12.3 +12.34 −12.345	Fixed-point data, i.e. decimal values without integers or exponents. 
(4) NR3 NUMERIC RESPONSE DATA <Example> 12.3E + 4 +12.34E − 5 −12.345 E + 6 • No lower-case character is allowed for E. • Spaces before and after E are not allowed. • "+" in exponent part cannot be omitted. • "+" in mantissa part can be omitted.	Floating-point data, i.e. decimal values with exponent digits. 

Element	Function
(5) HEXADECIMAL NUMERIC RESPONSE DATA <Example> #HABC123 #H2DC3 #H8301	Hexadecimal numeric data. 
(6) OCTAL NUMERIC RESPONSE DATA <Example> #Q37 #Q26703 #Q30562	Octal numeric data. 
(7) BINARY NUMERIC RESPONSE DATA <Example> #B011101 #B1011 #B1011	Binary numeric data. * For MP1764A, data with "+" and "−" is handled. 

Element	Function
(8) STRING RESPONSE DATA <Example> "This is a text" "Say," "Hello" "."	All the ASCII 7 bit codes are available. Both ends of the character string are always enclosed by double quotation marks. Double quotation marks within a character string are used as two consecutive quotations composed of identical ones. They are suitable for outputting texts to a printer or CRT since CRs, LFs and spaces are available. 
(9) DEFINITE LENGTH ARBITRARY BLOCK RESPONSE DATA <Example> Transferring 11256099D in 4 byte length ↓ #1400ABC123	Fixed-length 8 bit binary block data. It is suitable for transferring a large amount of data, 8 bit extended ASCII codes, non-displayed data and so on. 
(10) INDEFINITE LENGTH ARBITRARY BLOCK RESPONSE DATA <Example> Transferring −250, −50, 120, ... in undefined length ↓ #0FF06FFCE0078	Undefined-length 8 bit binary block data. So, the first data is preceded with #Ø. The last data is terminated by NL∧END. 

Element	Function
(11) ARBITRARY ASCII RESPONSE DATA <Example1> <ASCII Byte> <ASCII Byte> NL∧END <Example2> NL∧END	ASCII data bytes (excluding NL characters) sent without separating them; so, the last data is terminated by NL∧END. 

(Blank)

SECTION 7
COMMON COMMANDS

This section describes the common commands and common query commands specified in the IEEE 488.2 standard. These common commands are not the bus commands used in interface messages. Like device messages, common commands are a type of data message used in the bus data mode, i.e. when the ATN line is false. They can be used for all measuring instruments, including those made by other companies, as long as they conform to the IEEE 488.2 standard. IEEE 488.2 common commands must start with an *.

Control commands by ANRITSU PACKET V series personal computers are applied for formats and use examples in this section.

TABLE OF CONTENTS

(Blank)

7.1 Classification by Function of Common Commands Supported by the MP1764A

The table below shows the classification by function of the IEEE 488.2 common commands supported by the MP1764A. Supported commands are listed on the following pages in alphabetical order.

7.2 The Classification of Commands Supported and the Reference

Commands to be supported for MP1764A shown on the previous page are described for each function group in the table below. Each command is described in alphabetic order from the next page.

Group	Function	Mnemonic
System data	Data specific to each device connected to the GPIB system, e.g. manufacturer, model, serial number, etc.	*IDN?
Internal operation	Device internal control: ① Resetting device in level 3 (See Section 4) ② Device self testing and error detection	*RST *TST?
Synchronization	Synchronization of device to controller by: ① Waiting for a service request ② Waiting for a response from the device output queue ③ Performed by forcing sequential execution.	*OPC *OPC? *WAI
Status and event	A status byte consists of a status summary message. The summary bits of the message are supplied by the standard event register, the output queue and the extended event register or extended queue. Four commands and five queries are available to set or clear the data in the registers and queues, to enable or disable them and to obtain the settings status of the registers.	*CLS *ESE *ESE? *ESR? *PSC *PSC? *SRE *SRE? *STB?
Device trigger	Defines the commands to be executed when the IEEE 488.2 GET bus command is received by a device.	*TRG

*CLS Clear Status Command

(Clear status byte register)

■ **Syntax**

```
*CLS
```

■ **Example**

```
30 WRITE @103:"*CLS"
40 WRITE @103:"DTM△0;CTM△0;*CLS"
```

■ **Explanation**

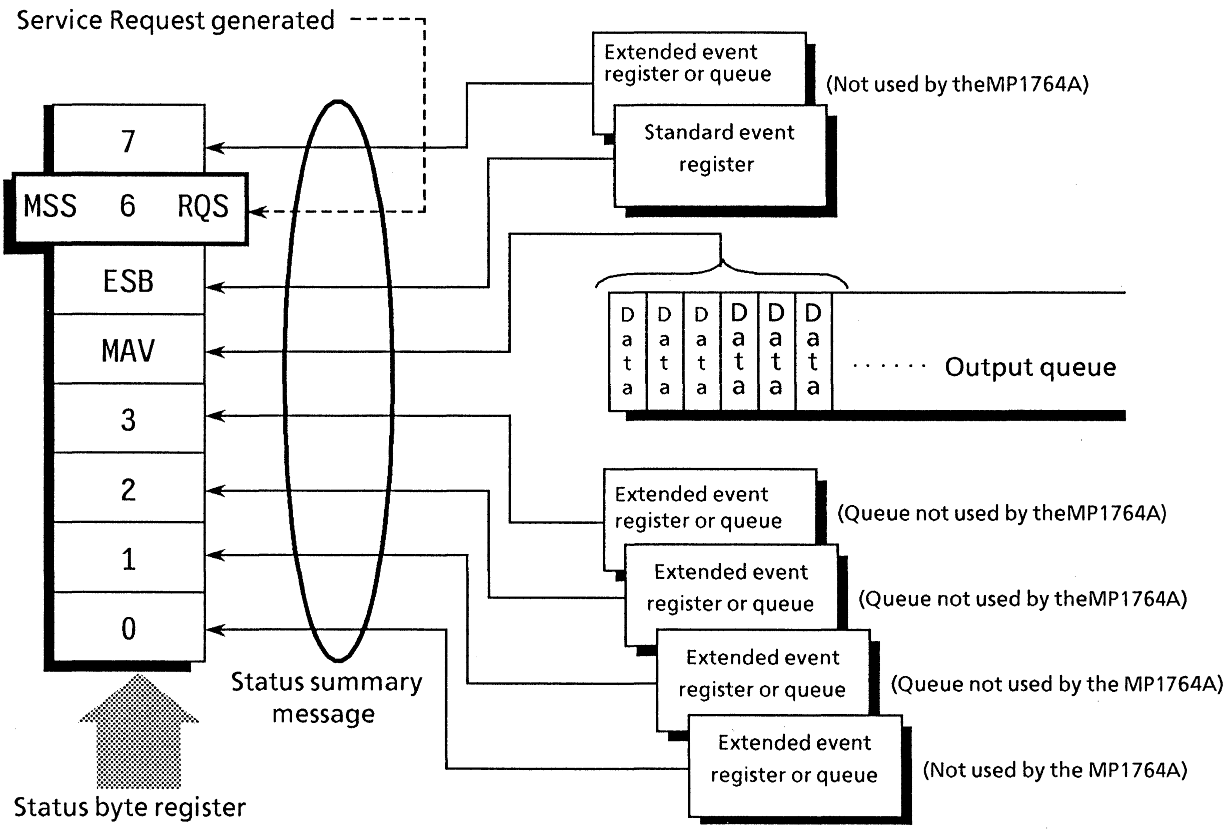
The ***CLS** common command clears all status data structures (i.e their event registers and queues) except for the output queue and its **MAV** summary messages. It also clears the summary messages corresponding to these structures.

In the example below, the output queue and its **MAV** summary messages are also cleared.

```
30 WRITE @103:"DTM△0;CTM△0"
40 WRITE @103:"*CLS;DTM?"
```

That is to say, if a ***CLS** command is sent after a <PROGRAM MESSAGE TERMINATOR> or before <QUERY MESSAGE UNIT> elements, all status bytes are cleared. This command also clears all unread messages in the output queue.

***CLS** has no effect on settings in enable registers.



*ESE Standard Event Status Enable Command

(Sets or clears the standard event status enable register)

■ Syntax

*ESE <HEADER SEPARATOR> <DECIMAL NUMERIC PROGRAM DATA>

In this format:

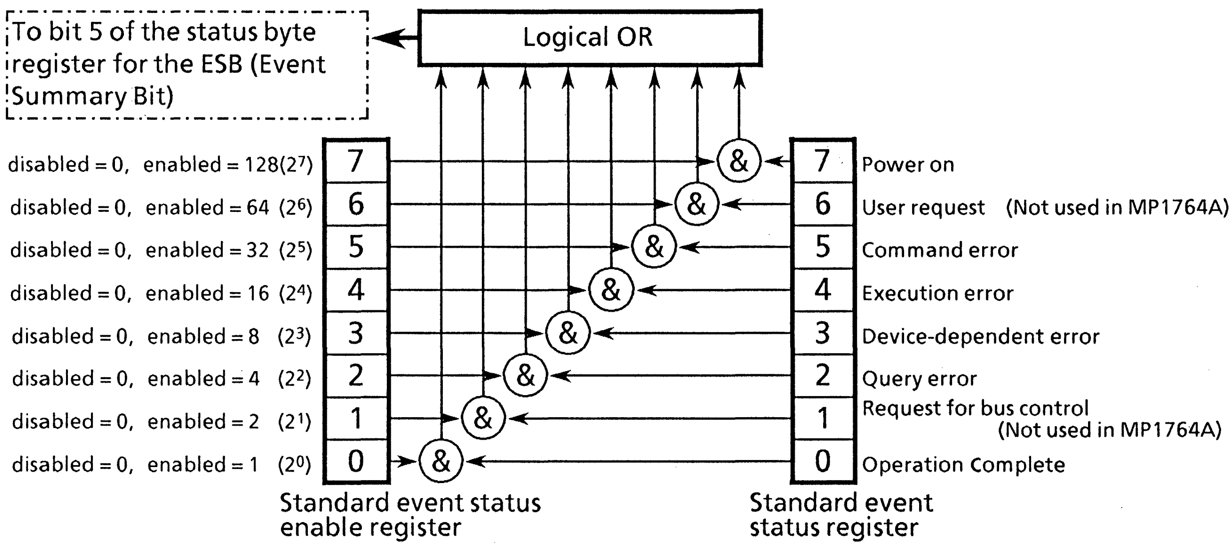
<DECIMAL NUMERIC PROGRAM DATA> = Value rounded to an integer from 0 to 255 (Binary weighted with a base value of 2)

■ Example

```
WRITE @1Ø3:"*ESE 2Ø"!
```
Sets bits 2 and 4 of enable register

■ Explanation

The program data is the sum of weighted bit-digit values when the weighted value for bits to be enabled are selected from among the values $2^0=1$, $2^1=2$, $2^2=4$, $2^3=8$, $2^4=16$, $2^5=32$, $2^6=64$ or $2^7=128$; corresponding to the enable register bits 0, 1, 2, 3, 4, 5, 6 or 7. The value of bits to be disabled is 0.



*ESE? Standard Event Status Enable Query

(Returns current value of standard event status enable register)

■ Syntax

*ESE?

■ Example

20 is the response if *ESE? is sent after executing *ESE 20

■ Explanation

Returns NR1, the value of the standard event status enable register

■ Response message

NR1 = 0 ~ 255

*ESR?: Standard Event Status Register Query

(Returns the current value in the standard event status register)

■ Syntax

```
*ESR?
```

■ Example

```
30 WRITE @103:"*ESR?"
40 READ @103:STEVET
50 PRINT STEVET
```

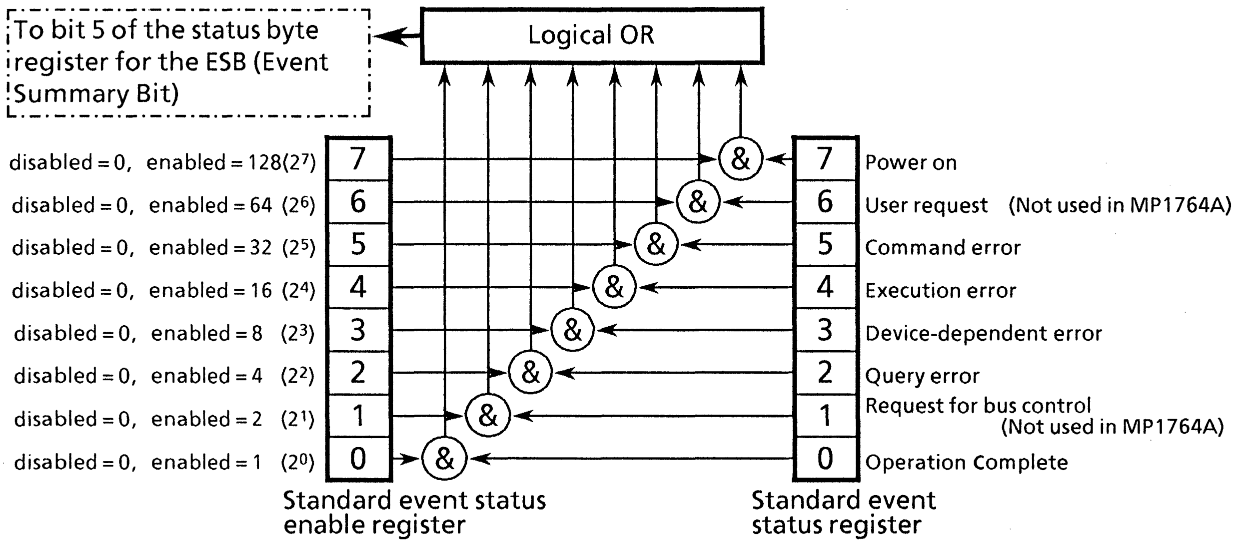
■ Response Message

NR1=0 to 255

■ Explanation

The current value of the standard event status register is returned by NR1. NR1 is the total of weighted bit-digit values of bits (enabled by the standard event status enable register) which are selected from amongs the values $2^0=1$, $2^1=2$, $2^2=4$, $2^3=8$, $2^4=16$, $2^5=32$, $2^6=64$ or $2^7=128$: corresponding to the standard event status register bits 0, 1, 2, 3, 4, 5, 6 or 7.

This register is cleared when the response is read (e.g. line 40).



*IDN? Identification Query

(Returns the manufacturer name, model name etc. of the product.)

■ Syntax

```
*IDN?
```

■ Example

```
30 WRITE @103:"*IDN?"
40 READ @103:IDEN$!
```
Stores names of manufacturer, model, etc.

■ Explanation

Returns manufacturer's name, model name, 0.firmware level

→ 0001

→ MP1762A

→ ANRITSU

If an ***IDN?** common query is sent to a device when the manufacturer is Anritsu, the model is MP1764A, and the firmware version is 1; a response message comprising the four fields shown above is returned.

① Field 1 ..... Manufacturer's name (Anritsu)

② Field 2 ..... Model name (MP1762A)

③ Field 3 ..... (usually 0)

④ Field 4 ..... Firmware version

■ Response message

A Response message comprising the four fields above separated by commas is sent by <ARBITRARY ASCII RESPONSE DATA>.

`<field 1>, <field 2>, <field 3>, <field 4>`

For the example above,

ANRITSU,MP1762A,0,0001

The total length of a response message is ≦72 characters

Note

Even if the real model name is MP1764A, the response message is MP1762A.

*OPC Operation Complete Command

(Sets the status of bit 0 of the standard event status register when device operation is completed)

■ Syntax

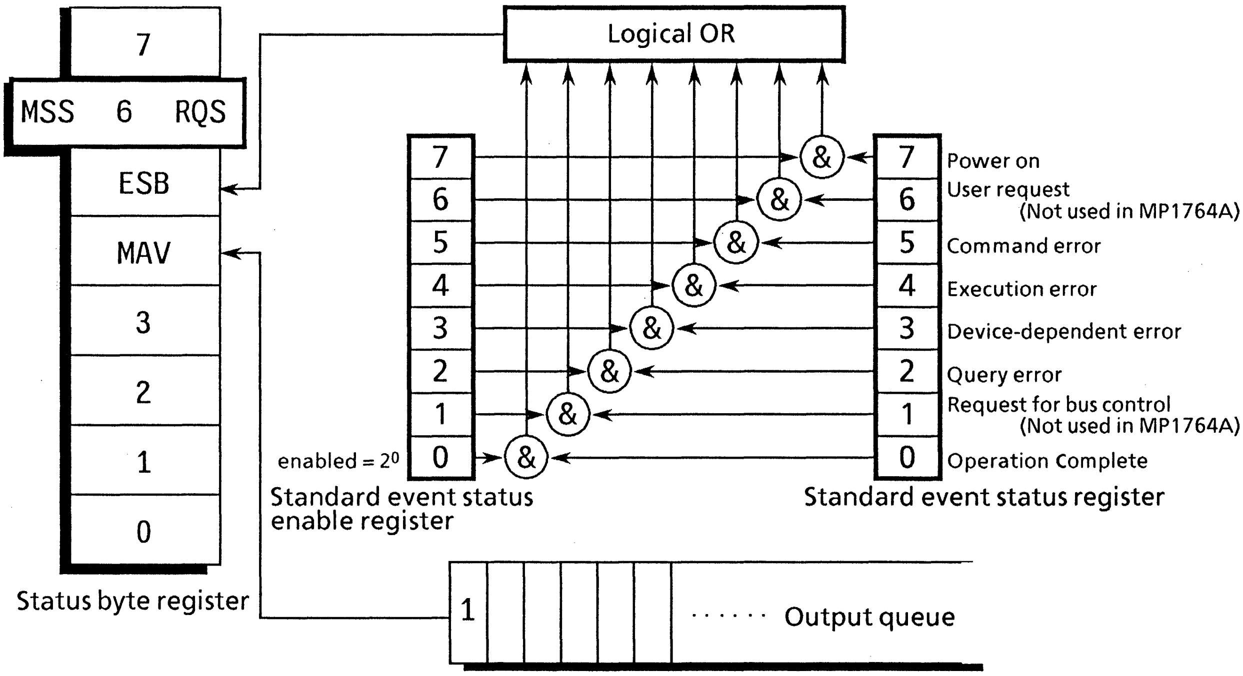
```
*OPC
```

■ Example

```
WRITE @103:"*OPC"
```

■ Explanation

Sets the status of bit 0, i.e. the operation complete bit, of the standard event status register when all pending operations of the selected device have been completed. This is an overlap command.



*OPC? Operation Complete Query

(Sets 1 in the output queue to generate a MAV summary message when device operation has been completed)

■ Syntax

```
*OPC?
```

■ Example

```
WRITE @1Ø3:"*OPC?"
```

■ Explanation

When all pending operations of the selected device have been completed, sets 1 in the output queue and waits for the MAV summary message to be generated.

■ Response message

A 1 is returned by <NR1 NUMERIC RESPONSE DATA>.

*PSC Power-on Status Clear Command

(Specifies whether status enable registers are cleared at power-on, or not.)

■ Syntax

```
*PSC <HEADER SEPARATOR><DECIMAL NUMERIC PROGRAM DATA>
```

In this format:

<DECIMAL NUMERIC PROGRAM DATA> = Ø	:	not cleared
Numbers in range of −32767 to 32767	:	cleared

■ Example

```
WRITE @1Ø3:"*PSC Ø;*SRE 32;*ESE 128"!
```
not cleared and SRQ is on

■ Explanation

The *PSC command specifies whether the three enable registers of service request, standard event status, and parallel poll in status are cleared at power-on, or not.

A value in the <DECIMAL NUMERIC PROGRAM DATA> field controls the logical state of the power-on status flag. When it is rounded to 0, the flag is set to false, so the enable registers are not cleared. When the *PSC Ø is issued, it enables the device to generate the SRQ at power-on. In the above example, the power-on event is reported to the controller.

When the value in the <DECIMAL NUMERIC PROGRAM DATA> field is rounded to an integer other than 0 that is in range of −32767 to 32767, the flag is set to true, so the enable registers are cleared. When the *PSC 1 is issued, it enables the device to clear the registers but not to generate the SRQ.

When the value in the <DECIMAL NUMERIC PROGRAM DATA> field is rounded to an integer that is out of range of −32767 to 32767, the execution error is generated.

*PSC? Power-on Status Clear Query

(Returns the power-on status flag state)

■ Syntax

```
*PSC?
```

■ Example

```
30 WRITE @103:"*PSC?"
40 READ:POWF
```

■ Explanation

When the ***PSC?** common query is issued, 1 is returned when the power-on status flag is true, and 0 is returned when it is false.

■ Response message

NR1 = 1 (Power-on status flag is true.) 0 (Power-on status flag is false.)

*RST Reset Command

(Resets (initializes) device in level 3)

■ **Syntax**

```
*RST
```

■ **Example**

`WRITE @1Ø3:"*RST"` Resets devices in level 3

■ **Explanation**

The ***RST** command resets a device in level 3. (See Section 4)

The items that are reset in level 3 are as follows.

① The functions and conditions specific to a device are reset to a known initial state regardless of the settings up to that point. (See Section 4 for MP1764A initial states)

② Macro operation is inhibited and the device can no longer receive macros. And, macro definition is reset to the state designated by the system designer.

③ The device is put into OCIS (Operation Complete Command Idle State). As a result, the operation complete (end) bit cannot be set in the standard event status register.

④ The device is put into OQIS (Operation Complete Query Idle State). As a result, the operation complete bit cannot be set in the output queue. The MAV bit is cleared.

The ***RST** command has no effect on the following.

① The state of the IEEE 488.1 interface

② Device address

③ Output queue

④ Service request enable register

⑤ Standard event status enable register

⑥ Power-on-status-clear flag setting

*SRE Service Request Enable Command

(Sets status of bits in the service request enable register)

■ Syntax

*SRE <HEADER SEPARATOR> <DECIMAL NUMERIC PROGRAM DATA>

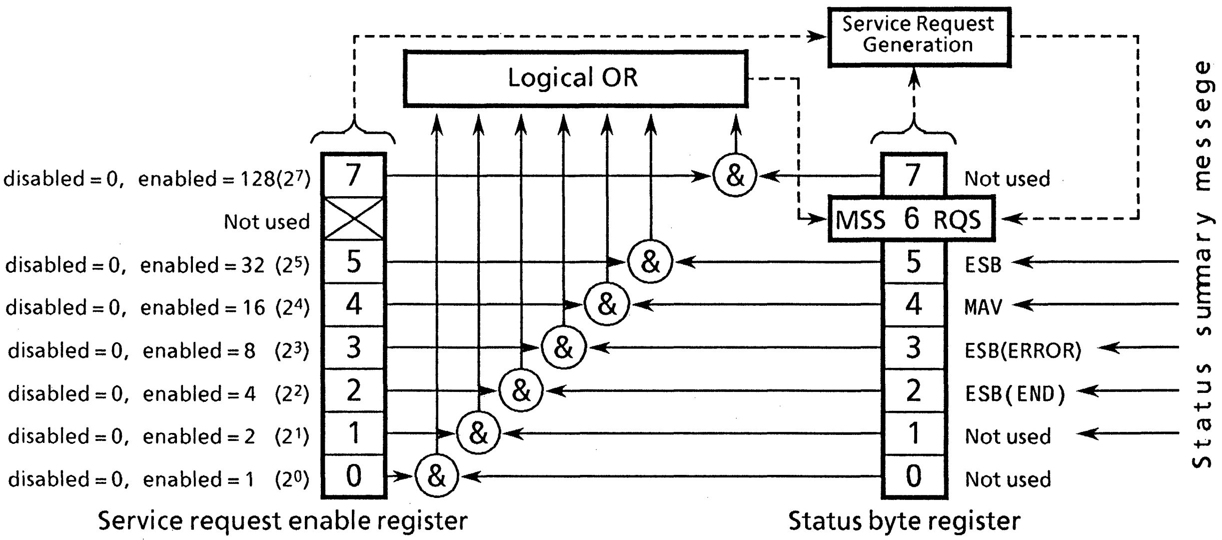
<DECIMAL NUMERIC PROGRAM DATA> = Values rounded to an integer from 0 to 255 (binary weighted with a base value of 2)

■ Example

`WRITE @1Ø3:"*SRE 16"!` Sets bit 4 of the enable register

■ Explanation

The program data is the sum of weighted bit-digit values when the weighted value for bits to be enabled are selected from among the values $2^0=1$, $2^1=2$, $2^2=4$, $2^3=8$, $2^4=16$, $2^5=32$ or $2^7=128$: corresponding to the service request enable register bits 0, 1, 2, 3, 4, 5, 6 or 7. The value of bits to be disabled is 0.



*SRE? Service Request Enable Query

(Returns the current value of the service request enable register)

■ **Syntax**

```
*SRE?
```

■ **Example**

A 16 is sent in response if *SRE? is sent after executing *SRE 16.

■ **Explanation**

NR1, the value of the service request enable register, is returned.

■ **Response message**

As NR1 (bit 6 : RQS bit) cannot be set, NR1 = 0 to 63 or 128 to 191)

*STB? Read Status Byte Command

(Returns the current values of status bytes including MSS bits)

■ Syntax

```
*STB?
```

■ Example

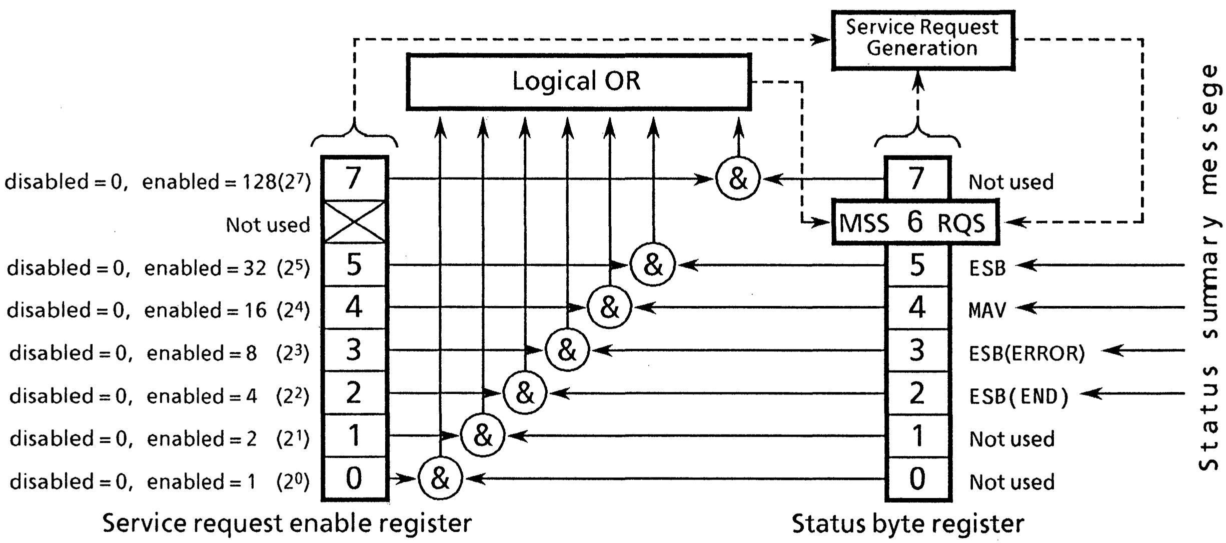
```
30 WRITE @103:"*STB?"
40 READ @103:STBV
50 PRINT STBV
```

■ Explanation

The *STB? query returns the total of the binary weighted values of the status byte register and of the MSS summary message with <NR1 NUMERIC RESPONSE DATA>.

■ Response message

The response message is a <NR1 NUMERIC RESPONSE DATA> integer in the range 0 to 255 representing the total of the binary weighted values of the bits in the status byte register. Status byte register bits 0 to 5 and 7 are weighted to 1, 2, 4, 8, 16, 32 and 128, respectively, and the MSS (Master Summary Status) bit to 64. MSS message indicates that a request has at least one cause.



The table below shows the conditions for the MP1764A's status byte register.

Bit	Bit weight	Bit name	Status-byte-register conditions	
7	128	——	0 = Not used	
6	64	MSS	0 = Service not requested	1 = Service requested
5	32	ESB	0 = Event status not generated	1 = Event status generated
4	16	MAV	0 = No data in output queue	1 = Data in output queue
3	8	ESB(ERROR)	0 = Event status not generated	1 = Event status generated
2	4	ESB(END)	0 = Event status not generated	1 = Event status generated
1	2	——	0 = Not used	
0	1	——	0 = Not used	

*TRG Trigger Command

(The same function as that of IEEE 488.1 GET-Group Execute Trigger-bus command)

■ Syntax

```
*TRG?
```

■ Example

```
WRITE @1Ø3:"*TRG"
```

■ Explanation

The ***TRG** common command has the same function as the IEEE 488.1 **GET** – Group Execute Trigger-bus command. The MP1764A does not support the ***DDT** command.

With the MP1764A, a measurement is started by executing the ***TRG** common command.

```
WRITE @1Ø3:"*TRG"
```

*TST? Self-test Query

(Returns the results of error present/absent in the self-test)

■ Syntax

```
*TST?
```

■ Example

```
3Ø WRITE @1Ø3:"*TST?"
4Ø READ @1Ø3:TEST
5Ø PRINT TEST
```

■ Explanation

The ***TST?** query executes the self-test of the internal cirucit in device(s). The test result is set in the output queue. Data in the output queue indicates whether or not the test has been completed without error occurrence. Opeator intervention is not required to execute the self-test.

When the power is turned on, the MP1764A reports the self-test result.

■ Response message

The response message is sent by <NR1 NUMERIC RESPONSE DATA>. The data range is –32767 to 32767.

NR1 = Ø .... Indicates no errors

NR1 ≠ Ø .... Indicates that errors have occurred

*WAI Wait-Continue Command

(Forces the next command to wait while the device is executing a command)

■ Syntax

```
*WAI
```

■ Example

```
WRITE @1Ø3:"*WAI"
```

■ Explanation

The *WAI common command executes a overlap command as a sequential command.

The overlap command is a command or query that is sent by the controller and allows the next command to be executed even while the device is executing something.

While the device is executing a command, executing the *WAI common command after an overlap command forces the next command to wait and allows it to be executed after the current command has been executed. This action is the same as that of sequential command.

SECTION 8
STATUS STRUCTURE

This section describes device status reports and their data structure as defined in the IEEE 488.2 standard and explains the techniques for synchronizing the controller and devices.

In order to obtain more detailed status information, the IEEE 488.2 standard has more common commands and common queries than the IEEE 488.1 standard.

Refer to Section 7 for a detailed explanation of these common commands and queries.

Control commands by ANRITSU PACKET V series personal computers are applied for formats and use examples in this section.

TABLE OF CONTENTS

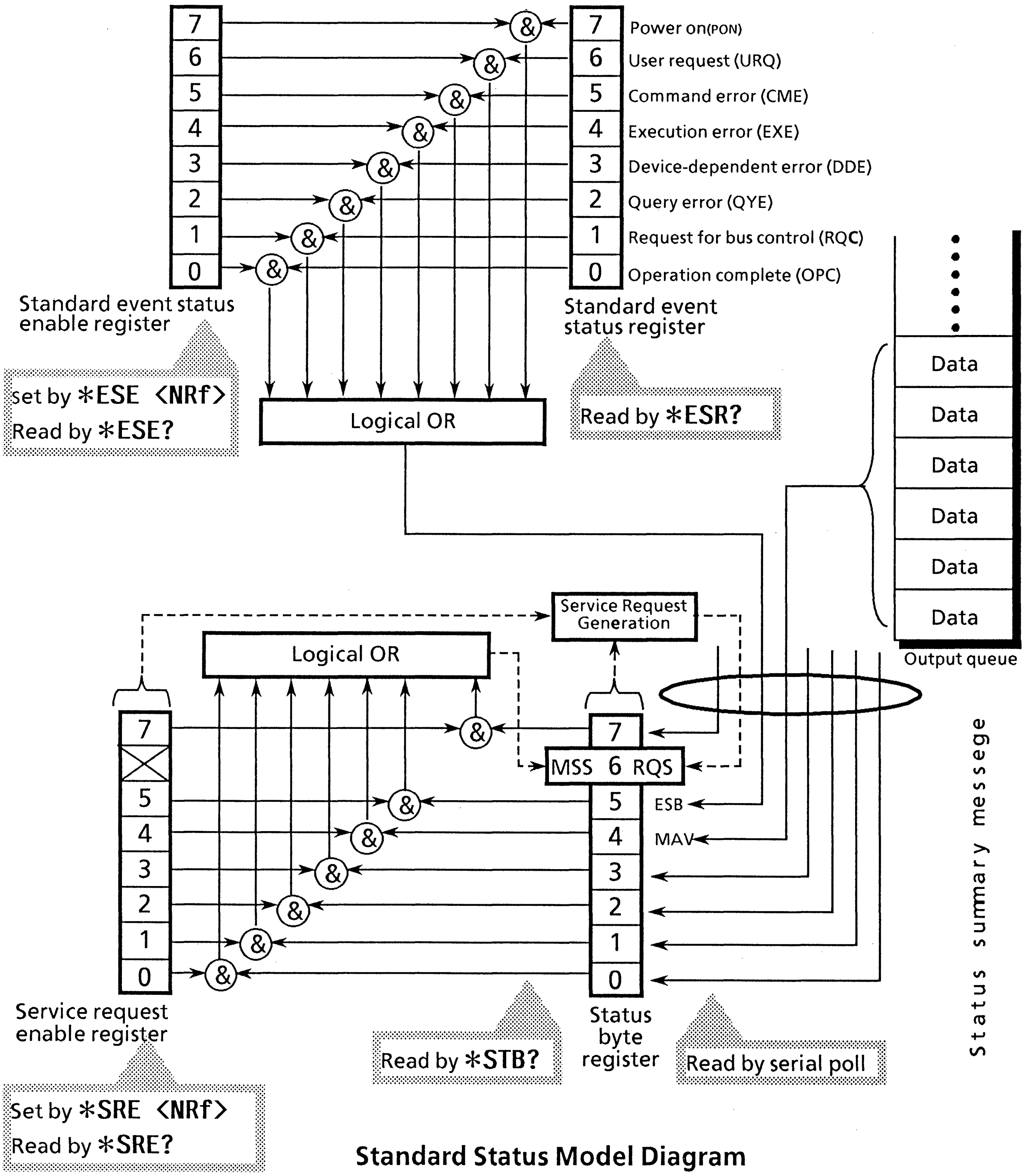
(Blank)

The Status Byte (SB) sent by the controller is based on the IEEE 488.1 standard. The bits comprising it are called a status summary message because they represent a summary of the current data contained in registers and queues.

The following pages explain the status summary message and the structure of the status data that constitutes the status summary message bits as well as techniques for synchronizing the devices and controller, which use these status messages.

8.1 IEEE 488.2 Standard Status Model

The diagram below shows the standard model for the status data structure stipulated in the IEEE 488.2 standard.



Standard Status Model Diagram

The IEEE 488.1 status byte is used in the status model. This status byte is composed of 7 summary message bits given from the status data structure. For creating the summary message bits, there are 2 models for the data structure - the register model and the queue model.

Register model	Queue model
The register model consists of the two registers used for recording events and conditions encountered by a device. These two registers are the Event Status Register and Event Status Enable Register. When the results of the AND operation of both register contents is not 0, the corresponding bit of the status bit becomes 1. In other cases, it becomes 0. And, when the result of their Logical OR is 1, the summary message bit becomes also 1. If the Logical OR result is 0, the summary message bit becomes 0 too.	The queue in the queue model is for sequentially recording the waiting status values and data. The queue structure is such that the relevant bit is set to 1 when there is data in it and 0 when it is empty.

In IEEE 488.2, there are 3 standard models for status data structure - 2 register models and 1 queue model - based on the register model and queue model explained above. They are:

① Standard Event Status Register and Standard Event Status Enable Register

② Status Byte Register and Service Request Enable Register

③ Output queue

Standard Event Status Register	Status Byte Register	Output Queue
The Standard Event Status Register has the structure of the previously described register model. In this register, bits are set for 8 types of standard event encountered by a device, viz. ① Power on, ② User request, ③ Command error, ④ Execution error, ⑤ Device-dependent error, ⑥ Query error, ⑦ Request for bus control and ⑧ Operation complete. The Logical OR output bit is represented by Status Byte Register bit 5 (DIO6) as a summary message for the Event Status Bit (ESB).	The Status Byte Register is a register in which the RQS bit and the 7 summary message bits from the status data structure can be set. It is used together with the Service Request Enable Register. When the results of the OR operation of both register contents is not 0, SRQ becomes ON. To indicate this, bit 6 of the Status Byte Register (DIO7) is reserved by the system as the RQS bit which means that there is a service request for the external controller. The mechanism of SRQ conforms to the IEEE 488.1 standard.	The Output Queue has the structure of the queue model mentioned above. Status Byte Register bit 4 (DIO5) is set as a summary message for Message Available (MAV) to indicate that there is data in the output queue.

8.2 Status Byte (STB) Register

The STB register consists of device STB and RQS (or MSS) messages. The IEEE 488.1 standard defines the method of reporting STB and RQS messages but not the setting and clearing protocols or the meaning of STB. The IEEE 488.2 standard defines the device status summary message and the Master Summary Status (MSS) which is sent to bit 6 together with STB in response to an *STB? common query.

8.2.1 ESB and MAV summary messages

The following is a description of the ESB and MAV summary messages.

(1) ESB summary messages

The ESB (Event Summary Bit) summary message is a message defined by IEEE 488.2, which is represented by bit 5 of the STB register. This bit indicates whether at least one of the events defined in IEEE 488.2 has occurred or not when the service request enable register is set so that events are enabled after the final reading or clearing of the standard event status register. The ESB summary message bit becomes true when the setting permits events to occur if any one of the events recorded in standard event status register is true. Conversely, it is false if none of the recorded events occurs even if events are set to occur.

(2) MAV summary messages

The MAV summary message is a message defined in IEEE 488.2 and represented by bit 4 in the STB register. This bit indicates whether the output queue is empty or not. The MAV summary message bit is set to 1 (true) when a device is ready to receive a request for a response message from the controller and to 0 (false) when the output queue is empty. This message is used to synchronize the exchange of information with the controller. For example, it can be used get the controller to wait till MAV is true after it has sent a query command to a device. While the controller is waiting for a response from the device, it can process other jobs.

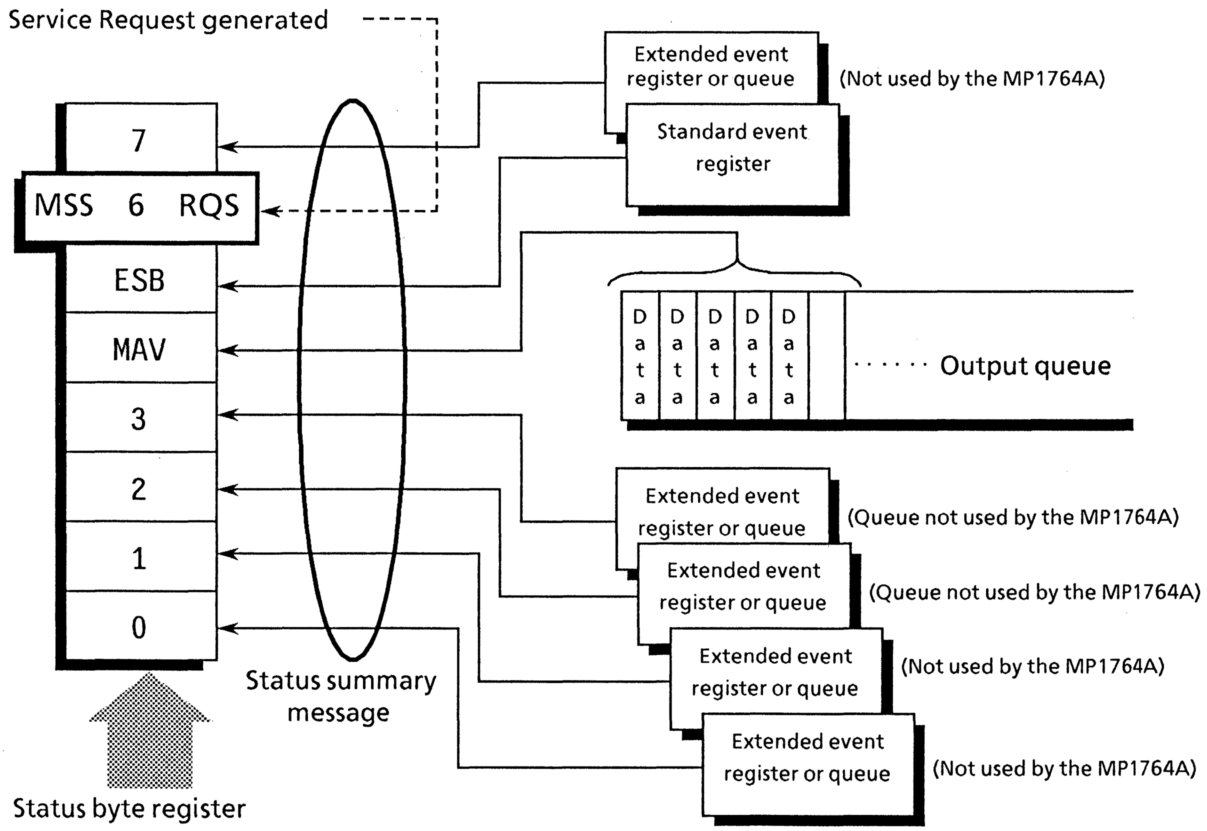
Reading the output queue without first checking MAV will cause all system bus operations to be delayed until the device responds.

8.2.2 Device-dependent summary messages

The IEEE 488.2 standard does not specify whether bits 7 (DIO8) and 3 (DIO4) to 0 (DIO1) of the status byte register are used as status register summary bits, or used to indicate that there is data in a queue. These bits can be used as device-dependent summary messages.

Device-dependent summary messages have the respective status data structures of the register model or the queue model. Thus, the status data structure may be either the register to report events and status in parallel or the queue to report conditions and status in sequence. The summary bit represents a summary of the current status of the corresponding data structure. In the case of the register model, the summary bit is true when there is an event set to permit the occurrence of more than one true; while in the case of the queue model, it is true if the queue is not empty.

As shown below, the MP1764A does not use bits 0, 1 and 7. As it uses bits 2 and 3 as the summary bits of the status register, it has 5 register model types (, where 3 types extended) and one queue model type - an output queue with no extension.



8.2.3 Reading and clearing the STB register

Serial poll or the ***STB?** common query are used to read the contents of STB register. STB messages conforming to IEEE 488.1 can be read by either method, but the value sent to bit 6 is different for each of them.

The STB register can be cleared using the ***CLS** command.

(1) Reading by serial poll

When using the serial poll conforming to IEEE 488.1, the device must return a 7-bit status byte and an RQS message bit which conforms to IEEE 488.1.

According to IEEE 488.1, the **RQS** message indicates whether the device sent **SRQ** as true or not. The value of the status byte is not changed by serial poll. The device must set the RQS message to false immediately after being polled. As a result, if the device is again polled before there is a new cause for a service request, the **RQS** message is false.

(2) Reading by the *STB? common query

The ***STB?** common query requires the device to send the contents of the STB register and one <NR1 NUMERIC RESPONSE DATA> from the **MSS** (Master Summary Status) summary message. The response represents the total binary weighted value of the STB register and the **MSS** summary message. The STB-register bits 0 to 5 and 7 are weighted to 1, 2, 4, 8, 16, 32, and 128; and the MSS to 64, respectively. Thus, excepting the fact that bit 6 represents the MSS summary message instead of the **RQS** message, the response to ***STB?** is identical to that for serial poll.

(3) Definition of MSS (Master Summary Status)

MSS indicates that there is at least one cause for a service request. The MSS message is represented at bit 6 in a device response to the ***STB?** query but it is not produced as a response to serial poll. In addition, it is not part of the status byte specified by IEEE 488.1. **MSS** is produced by the logical OR operation of STB register with SRQ enable (SRE) register. In concrete terms, MSS is defined as follows.

(STB Register bit0 AND SRE Register bit0)

OR

(STB Register bit1 AND SRE Register bit1)

OR

:

:

(STB Register bit5 AND SRE Register bit5)

OR

(STB Register bit7 AND SRE Register bit7)

As bit-6 status of the STB and SRQ enable registers are ignored in the definition of MSS, it can be considered that bit-6 status are always being 0 when calculating the value of MSS.

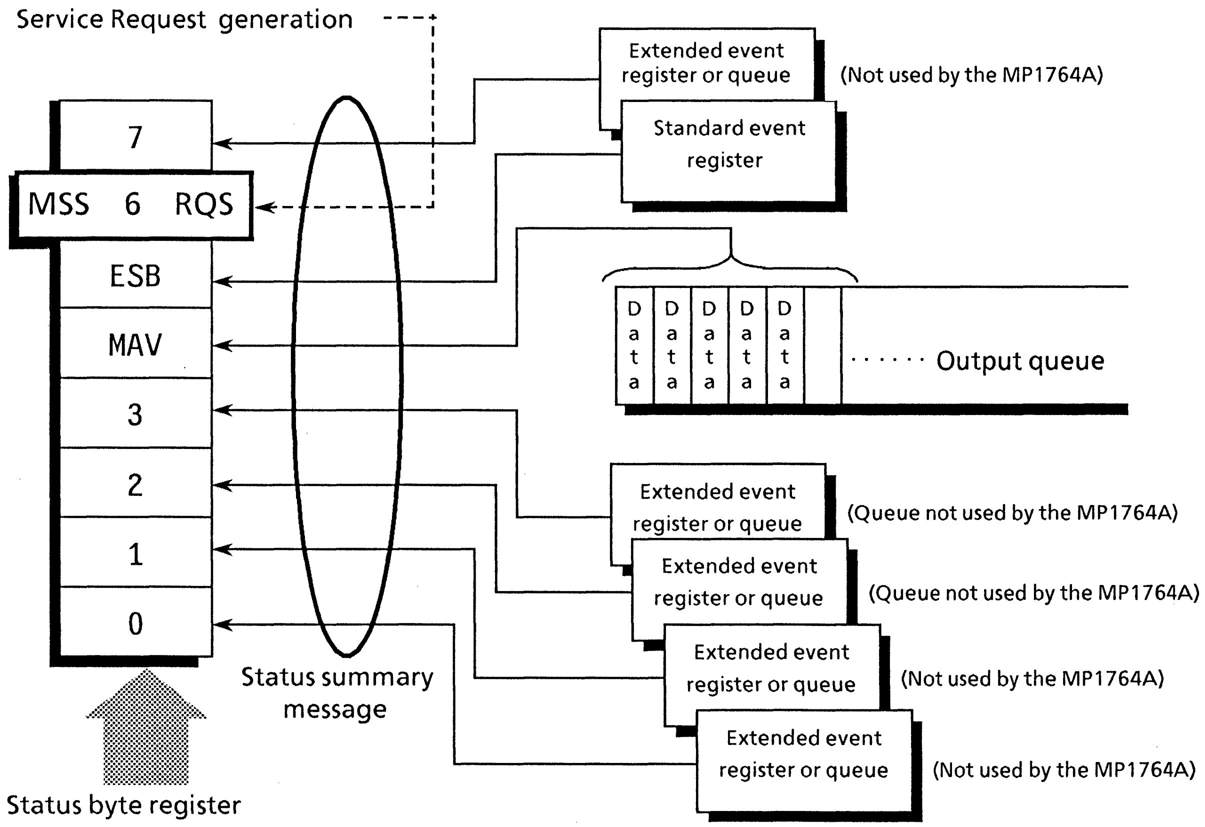
(4) Clearing the STB register by the ∗CLS common command

With the exception of the output queue and its **MAV** summary message, the **∗CLS** common command clears all status data structures (status event registers and queues) as well as the summary messages corresponding to them.

In the following case, the output queue and its **MAV** summary message are both cleared.

```
30 WRITE @103:"DTM△0;CTM△0"
40 WRITE @103:"*CLS;DTM?"
```

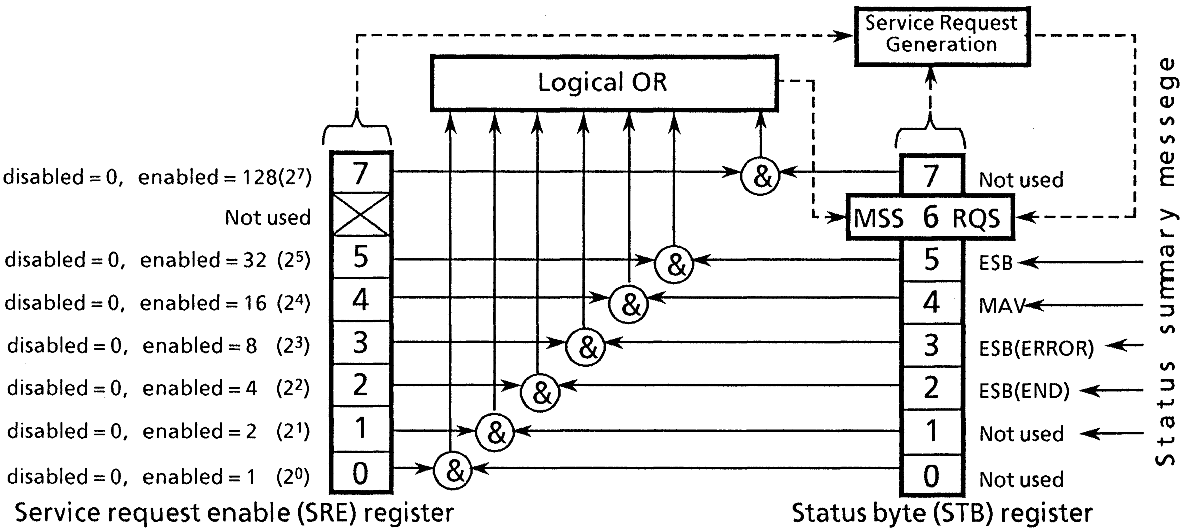
That is to say, sending a **∗CLS** command (after a <PROGRAM MESSAGE TERMINATOR> or before <QUERY MESSAGE UNIT> elements) clears all status bytes. This clears all unread messages in the output queue and sets the **MAV** message to false. The **MSS** message is also set to false when a response is made to **∗STB?**. The **∗CLS** command does not affect settings in the enable registers.



8.3 Enabling SRQ

All types of summary message in the STB register can be enabled or disabled for service requests by using the SRQ enable function. The service request enable (SRE) register is used for this function to select summary messages as shown in the diagram below.

Bits in the service request enable register correspond to bits in the status byte register. If a bit in the status byte corresponding to an enabled bit in the service request enable register is set to 1, a device makes a service request to the controller with the RQS bit set to 1. For example, if bit 4 (MAV) in the service request enable register is enabled, the device makes a request for service to the controller each time the MAV bit is set to 1 when there is data in the output queue.



(1) Reading the SRE register

The contents of the SRE register are read using the *SRE? common query. The response message to this query is a <NR1 NUMERIC RESPONSE DATA> integer from 0 to 255 which is the sum of the bit digit weighted values in the SRE register. SRE register bits 0 to 5 and 7 are respectively weighted to 1, 2, 4, 8, 16, 32 and 128. The unused bit 6 must always be set to 0.

(2) Updating the SRE register

The SRE register is written to using the *SRE common command. <DECIMAL NUMERIC PROGRAM DATA> elements follow the *SRE common command. <DECIMAL NUMERIC PROGRAM DATA> is a rounded integer expressed in binary which represents the sum of the binary weighted value of each bit of SRE register. A bit value of 1 indicates enabled and a bit value of 0 disabled. The value of bit 6 must always be ignored.

(3) Clearing the SRE register

The SRE register can be cleared by executing the *SRE common command or turn the power off and it on again.

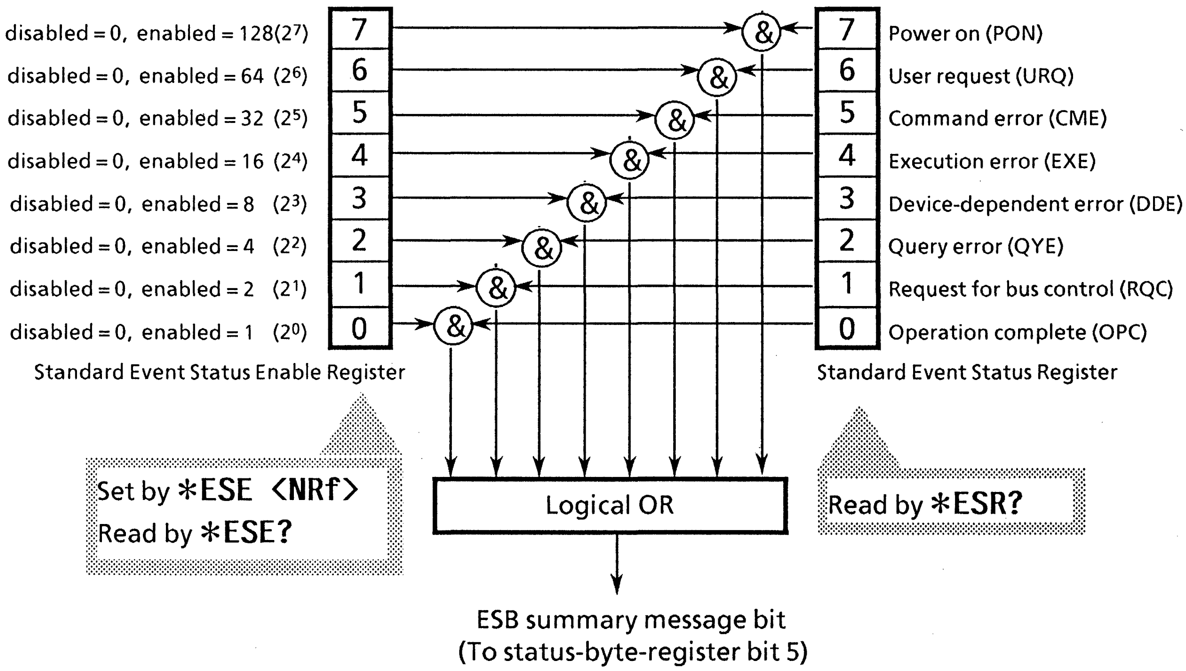
Using the *SRE common command, the SRE register is cleared by setting the value of the <DECIMAL NUMERIC PROGRAM DATA> element to 0. Clearing the register stops status information from generating rsv local messages, and service requests are no longer generated.

The MP1764A has the *PSC command. Therefore, if the PSC flag is ture when power is turned on, the SRE register is cleared.

8.4 Standard Event Status Register

8.4.1 Bit definition

The standard event status register must be available on all devices conforming to the IEEE 488.2 standard. The diagram below shows the operation of the standard event status register model. Because the operation of the model is the same as that for the other models explained up till now, the following only explains the meaning of each bit in the standard event status register as defined in the IEEE 488.2 standard.



Bit	Event name	Description
7	PON – Power on	The power is turned to on
6	URQ – User Request	Request for local control (rtl). This bit is produced regardless of whether a device is in remote or local mode. It is not used for the MP1764A so, it is always set to 0.
5	CME – Command Error	An illegal program message, a misspelt command or a **GET** command within a program is received. (Syntax error in header or parameter, or missing or too many parameters)
4	EXE – Execution Error	A legal program message, which cannot be executed, is received (Out of range for the parameter)
3	DDE – Device-dependent Error	An error caused by other than CME, EXE or QYE occurred. (The current device status cannot accept the request.)
2	QYE – Query Error	An attempt is made to read data in the output queue though there is none there, or data is lost from the output queue due to any reason, e.g. overflow etc..
1	RQC – Request Control	A device is requesting control of the bus. This bit is not used on the MP1764A so, it is always set to 0
0	OPC – Operation Complete	A device has completed operations which were pending and is ready to receive new commands. This bit is only set in response to the ***OPC** command.

8.4.2 Query error details

No.	Item	Description
1	Incomplete program messages	If a device receives an MTA from the controller before it receives the terminator of the program message it is receiving, it aborts the incomplete program message and waits for the next one. In order to abort the incomplete message, the device clears its input buffer and output queue, reports a query error and sets bit 2 in the standard status register to indicate the query error.
2	Interruption of response message	If a device receives an MLA from the controller before it has sent the terminator of the response message it is sending, it automatically interrupts the response message and waits for the next program message. In order to interrupt the response message, the device clears its output queue, reports a query error and sets bit 2 in the standard status register to indicate the query error.
3	Sending the next program message without reading the previous response message	When a device becomes unable to send a response message because the controller has sent another program message immediately following a program or query message, the device aborts the response message and waits for the next program message. It then reports a query error as in No. 2 above.
4	Output queue overflow	When several program and query messages are executed in succession, there may be too many response messages for the output queue (256 bytes). If further query messages are received when the output queue is full, the output queue cannot send responses to them because an overflow situation exists in it. If there is an overflow in the output queue, the device clears it and resets the section where response messages are created. Then it sets bit 2 in the standard event status register to indicate a query error.

8.4.3 Reading, writing to and clearing the standard event status register

Reading	The register is destructively read by the *ESR? common query, i.e. it is cleared after being read. The response message is an NR1 value obtained by binary weighting the event bit and converting it to a decimal number.
Writing	With the exception of clearing, writing operations cannot be performed externally.
Clearing	The register is only cleared in the following cases. ① A *CLS command is received ② The power is turned on when the power-on-status-clear flag is true. ③ An event is read for the *ESR? query command

8.4.4 Reading, writing to and clearing the standard event status enable register

Reading	The register is non-destructively read by the *ESE? common query, i.e. it is not cleared after being read. The response message is returned by NR1 after having been binary weighted and converted to decimal.
Writing	The register is written to by the *ESE common command. As bits 0 to 7 of the register are respectively binary weighted to 1, 2, 4, 8, 16, 32, 64 and 128; data to be written is sent by <DECIMAL NUMERIC PROGRAM DATA> which is the digit total of the bits selected from these bits.
Clearing	The register is cleared in the following cases. ① A *ESE command with a data value of 0 is received ② The power is turned on when the power-on-status-clear flag is true. The event status enable register is not affected by the following. ① Changes of the status of the IEEE 488.1 device clear function ② A *RST common command is received ③ A *CLS common command is received

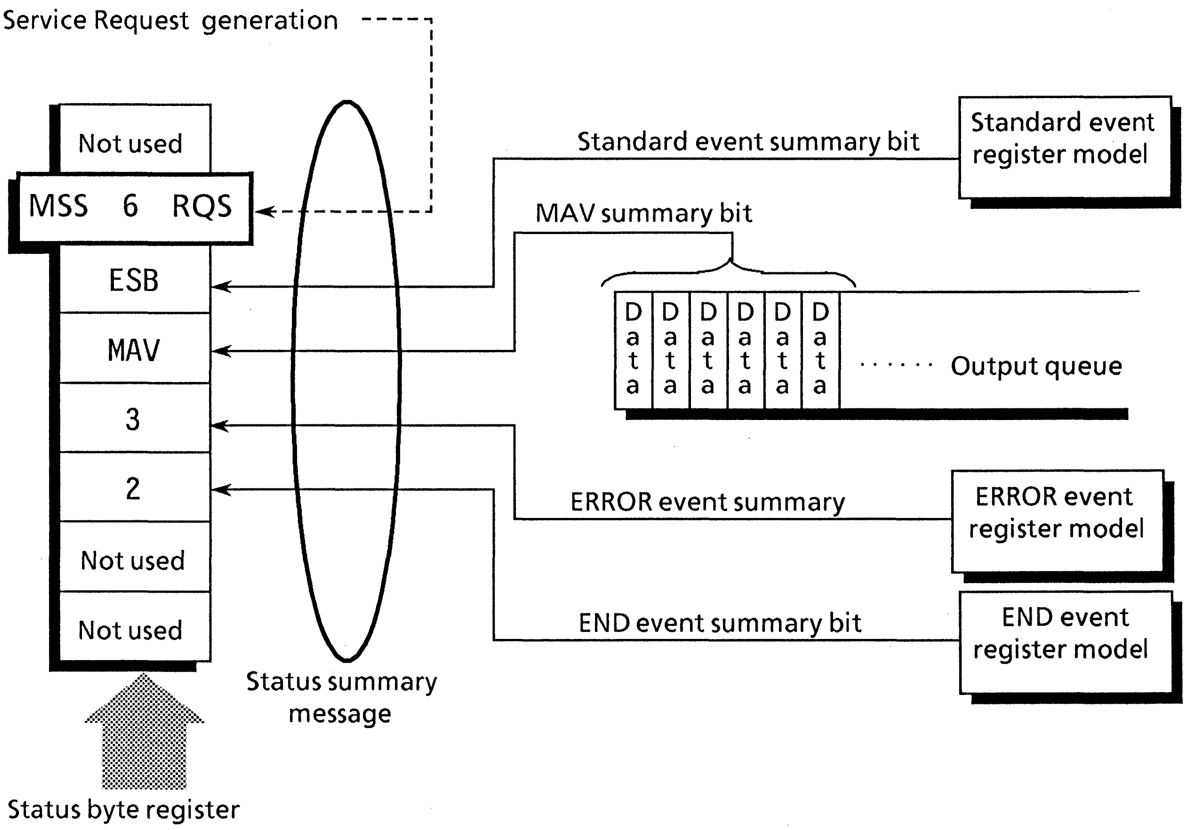
8.5 Extended Event Status Register

The register models of the status byte register, standard event status register and enable registers are mandatory for equipment conforming to the IEEE 488.2 standard.

In IEEE 488.2, status-byte-register bits 7 (DIO8), 3 (DIO4) to 0 (DIO1) are assigned to status- summary bits supplied by the extended-register and extended-queue models.

For the MP1764A, as shown in the diagram below, bits 0, 1 and 7 are unused and bits 2 and 3 are assigned to the END and ERROR summary bits as the status-summary bits supplied by the extended-register model.

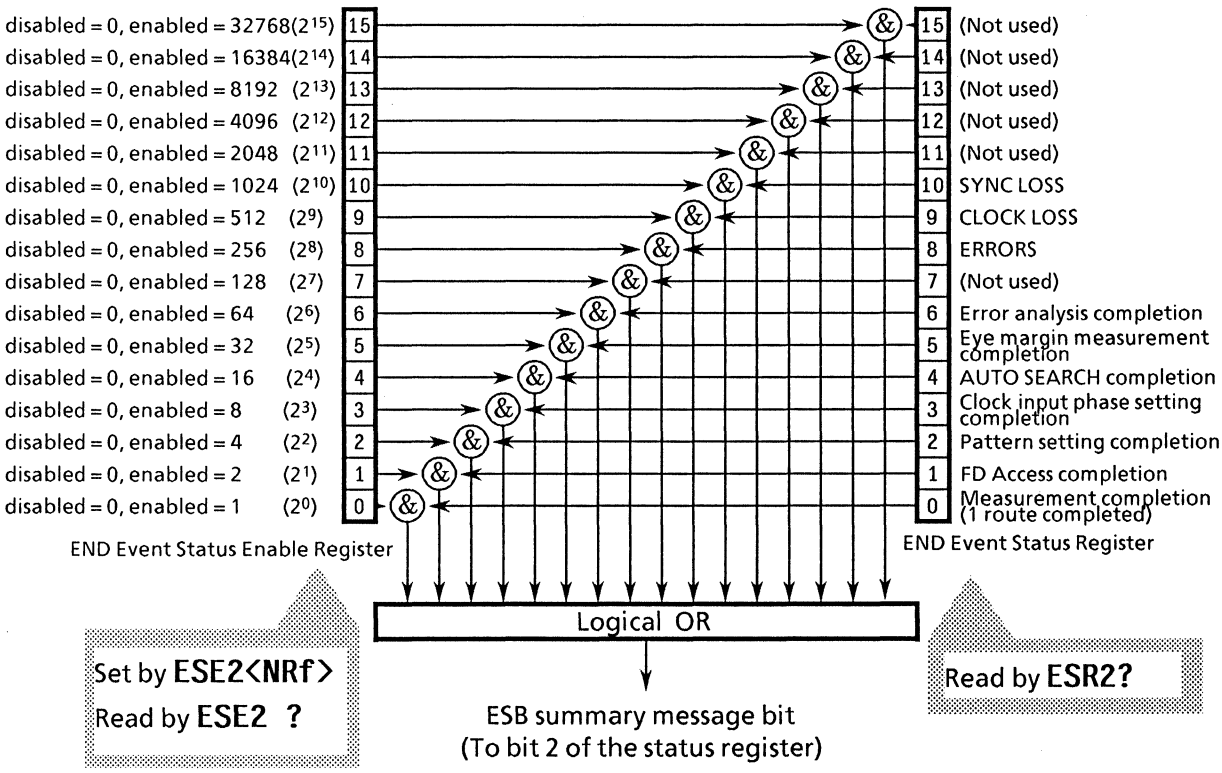
As the queue model is not extended, there is only one type of queue - the output queue.



The following pages describe bit definition, the reading, writing to and clearing of registers for the END and ERROR extended event register models.

8.5.1 Bit definition of END event status register

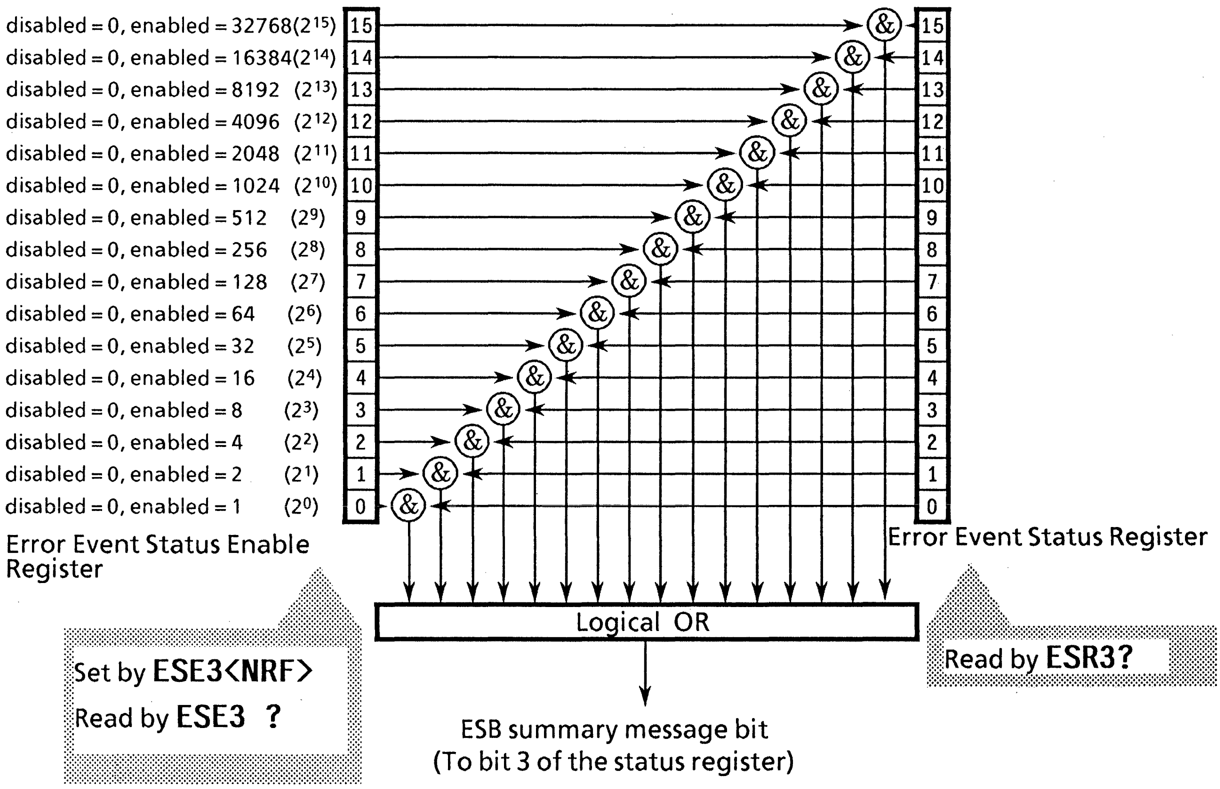
The following describes the operation of the END event status register model, the naming of its event bits and what they mean.



Bit	Event name	Description
15	(Not used)	(Not used)
14	(Not used)	(Not used)
13	(Not used)	(Not used)
12	(Not used)	(Not used)
11	(Not used)	(Not used)
10	SYNC LOSS	Synchronous loss has been occurred, or recovered.
9	CLOCK LOSS	Clock breakdown has been occurred, or recovered.
8	ERRORS	Error has been detected from error-free state.
7	(Not used)	(Not used)
6	Error analysis completion	Error analysis has been completed (only when OPTION-01 is installed).
5	Eye margin measurement completion	Eye margin measurement has been completed.
4	AUTO SEARCH completion	Auto search has been completed.
3	Clock input phase setting completion	The servo circuit used for setting clock input phase has been turned from BUSY to READY state.
2	Pattern setting completion	Programmable pattern setting has been completed.
1	FD Access completion	Accessing the floppy disk has been completed.
0	Measurement completion	When manual measurement, at operator stop. When single measurement, at measurement completion. When repeat measurement, at every measurement completion.

8.5.2 Bit definition of ERROR event status register

The following describes the operation of the ERROR event status register model, the naming of its event bits and what they mean.



Bit	Event name	Description
15	(Not used)	(Not used)
14	(Not used)	(Not used)
13	(Not used)	(Not used)
12	(Not used)	(Not used)
11	(Not used)	(Not used)
10	(Not used)	(Not used)
9	(Not used)	(Not used)
8	(Not used)	(Not used)
7	(Not used)	(Not used)
6	(Not used)	(Not used)
5	(Not used)	(Not used)
4	(Not used)	(Not used)
3	(Not used)	(Not used)
2	(Not used)	(Not used)
1	FD malfunction occurred	FD abnormal status has occurred.
0	Printer malfunction occurred	Printer abnormal status has occurred.

8.5.3 Reading, writing to and clearing the extended event status register

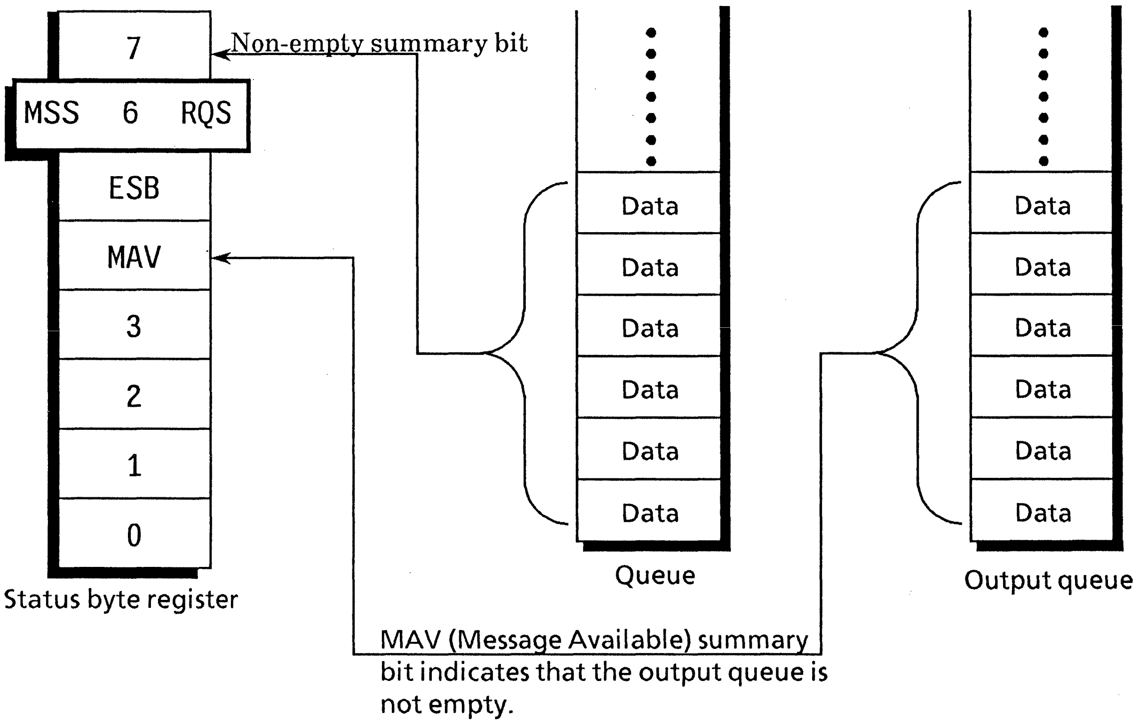
Reading	The register is destructively read by the a query, i.e. it is cleared after being read. The END and ERROR event status registers are read by the ESR2? and ESR3? queries. The read value, <NR1>, is obtained by binary weighting the event bit and converting it to decimal.
Writing	With the exception of clearing, writing operations cannot be performed externally.
Clearing	The register is cleared in the following cases. ①A *CLS command is received ②The power is turned on when the power-on-status-clear flag is true. ③An event is read for a query command

8.5.4 Reading, writing to and clearing the extended event status enable register

Reading	The register is non-destructively read by a query, i.e. it is not cleared after being read. The END and ERROR event status enable registers are read by the ESE2? and ESE3? queries. The read value, returned by <NR1>, is obtained by binary weighting the event bit and converting it to decimal.
Writing	The END and ERROR event status enable registers are written to by the ESE2 and ESE3 program commands. As bits 0 to 7 of the registers are respectively binary weighted to 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, and 32768 data to be written is sent by <DECIMAL NUMERIC PROGRAM DATA>, the digit total weighted value of the bits selected from among them.
Clearing	The register is cleared in the following cases. ① ESE2 and ESE3 program commands with a data values of 0 are received by the END and ERROR event status enable registers. ② The power is turned on when the power-on-status-clear flag is true. The extended event status enable register is not affected by the followings: ① Changes of the status of the IEEE 488.1 device clear function ② A *RST common command is received ③ A *CLS common command is received

8.6 Queue Model

The status-data-structure queue model is shown at the right of the diagram below. A queue is data structure including data lists arranged in sequence which provides a means of reporting sequential status and other information. The existence of such information in the queue is indicated by summary messages. The queue contents are read by the handshake when a device is in TACS (Talker Active State).



The output queue, which is mandatory, is the queue that outputs the MAV summary message to bit 4 of the status byte. A queue (which can output the MAV summary message to any of bits 0 to 3 or 7 of the status byte register) is an option and is simply called a "queue".

As the summary messages from the register model can also be connected to bits 0 to 3 or 7 of the status byte register, the types of summary messages vary with the device.

Though Anritsu assigns bit 7 of the status byte register for the use of summary message bits from "queues", it is not used when the output queue is sufficient.

The output queue is compared with an ordinary queue on the next page.

Comparison of Output and Ordinary Queues

Item	Output queue	Ordinary Queue
Data input / output operation	FIFO (First-In First-Out)	Need not always be FIFO
Read	Can only be read through the protocol defined in SECTION 6. The type of response message unit read is determined by the query.	Read by device-dependent query commands. The response messages read must be of the same type.
Writing	<PROGRAM MESSAGE> elements cannot be written directly to the output queue. They can only be sent to or from the system interface by the protocol specified by IEEE 488.2 message exchange.	<PROGRAM MESSAGE> elements cannot be written directly to a queue. They indicate encoded device information.
Summary message	Is true (1) when the output queue is not empty and false (0) when the output queue is empty. The MAV summary message is used to synchronize the exchange of information between a device and the controller.	Is true (1) when the queue is not empty and false (0) when the queue is empty.
Clearing	The output queue is cleared in the following cases: ① All items in it have been read ② A **DCL** bus command is received to initialize message exchange ③ PON is true at power on	A queue is cleared in the following cases: ① All items in it have been read ② A ***CLS** command is received ③ Other device-dependent methods are used

8.7 Techniques for Synchronizing Devices with the Controller

There are 2 ways of synchronizing devices with the controller.

① Enforcing the sequential execution: (Using the *WAI? command)

② Wait for a response from the device's output queue: (Using the *OPC? query)

③ Wait for a service request: (Using the *OPC command / *OPC? query)

8.7.1 Enforcing the sequential execution

There are two types of commands specific to devices: sequential commands and overlap commands.

- Sequential command

This is a command or query that is sent by the controller and does not allow the next command to be executed while the device is executing something.

- Overlap command

This is a command or query that is sent by the controller and allows the next command to be executed even while the device is executing something.

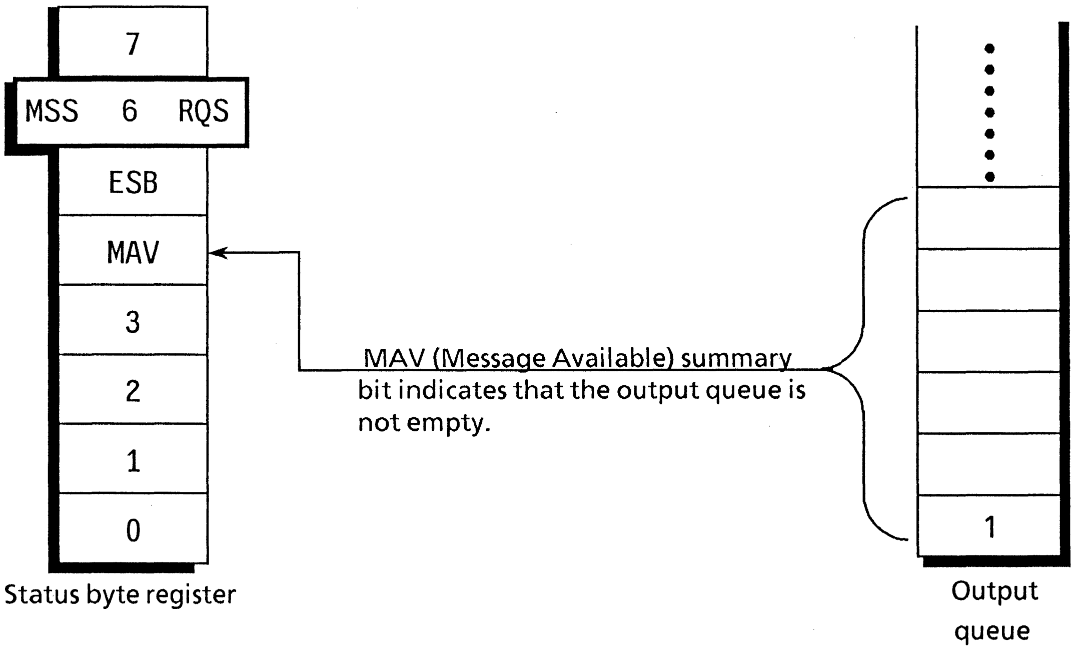
Enforcing the sequential execution is a synchronizing technique used to enforce a command that natively acts as an overlap command to be executed sequentially and not to perform the next process until one process has been completed. In this technique, the *WAI command is used.

8.7.2 Wait for a response from the output queue

Executing the ∗**OPC?** query sets a 1 in the output queue to generate a MAV summary message when a device has completed all of its pending operations.

In this technique, a device is synchronized with the controller by reading the 1 set in the output queue as described above or the MAV summary message bit.

As the MAV summary message bit is used in the "wait for a service request" technique, it will be explained in the next paragraph. The following explains synchronization by reading the output queue.



<Reading output-queue>

① Send overlap command. .... [The next ∗**OPC?** is used to confirm the completion of the final command. For this reason, it is normal to send ∗**OPC?** after executing an overlap command. However, even for sequential commands, if it is necessary to confirm the end of execution of the final command from the execution sequence or execution route, this is done by the ∗**OPC?** command.]

⇩

② Send ∗**OPC?**

⇩

③ Read output queue. .... [The 1 which is read is ignored, and it goes on to the next operation]

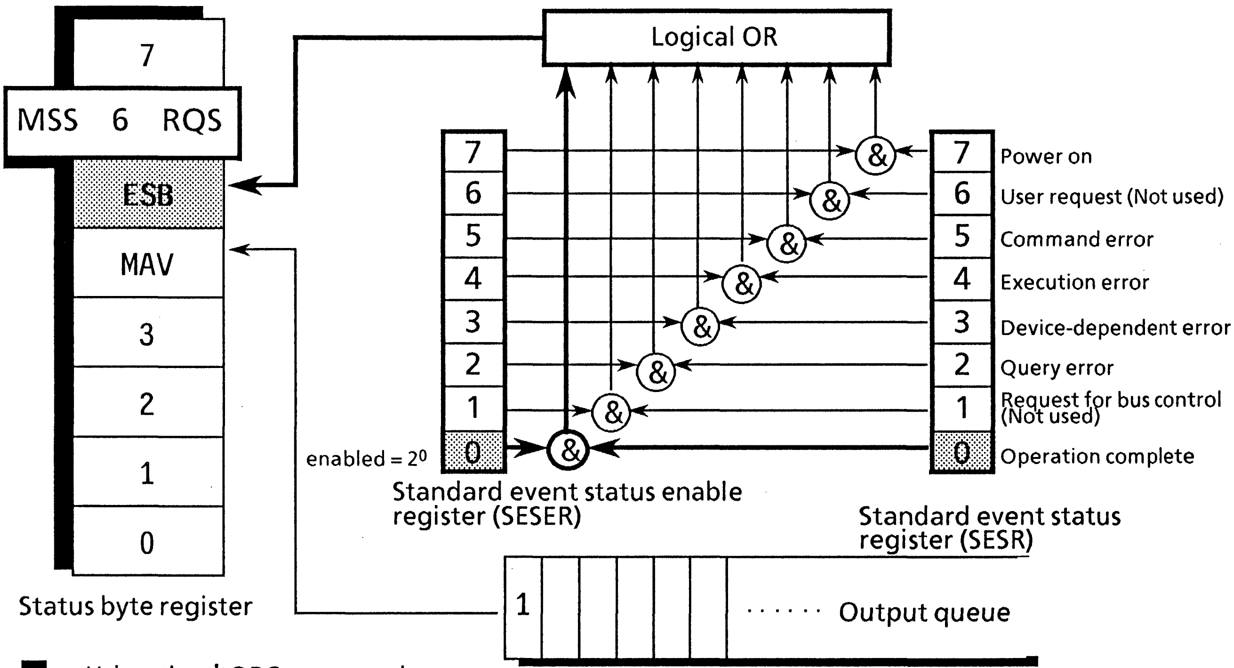
⇩

To the next operation

8.7.3 Wait for a service request

In this technique, the controller is momentarily interrupted by an SRQ signal from a device to process a status message from the device.

In a normal interrupt, the device would make a request to the controller at any time regardless of what the controller is doing. However, in using it as a technique for synchronizing the device with the controller, the controller sends an ***OPC** command or an ***OPC?** query to the device to check whether the device's operation has been completed or not. While waiting for the SRQ signal from the operation complete event, the controller carries on with some other useful task, and when it detects the operation complete event, the controller processes the designated task.



■ <Using the *OPC command>

①Enable the 2^0 bit of the standard event status enable register. `WRITE @1Ø3:"*ESE 1"`

⇩

②Enable the 2^5 bit of the service request enable register. `WRITE @1Ø3:"*SRE 32"`

⇩

③Make the device execute the specified operation. `WRITE @1Ø3:<PROGRAM MESSAGE>`

⇩

④Execute the *OPC command. (Since it is an overlap command, the next command is also executed) `WRITE @1Ø3:"*OPC"`

⇩

⑤Wait for an SRQ interrupt. (ESB summary message) .... Value of the status byte: $2^6 + 2^5 = 96$

■ <Using the *OPC? query>

① Enable the 2^4 bit of the service request enable register. `WRITE @1Ø3:"*SRE 16"`

⇩

② Make the device execute the specified operation. `WRITE @1Ø3:<PROGRAM MESSAGE>`

⇩

③ Send the *OPC? query. (Wait until the operation in ② has been completed) `WRITE @1Ø3:"*OPC?"`

⇩

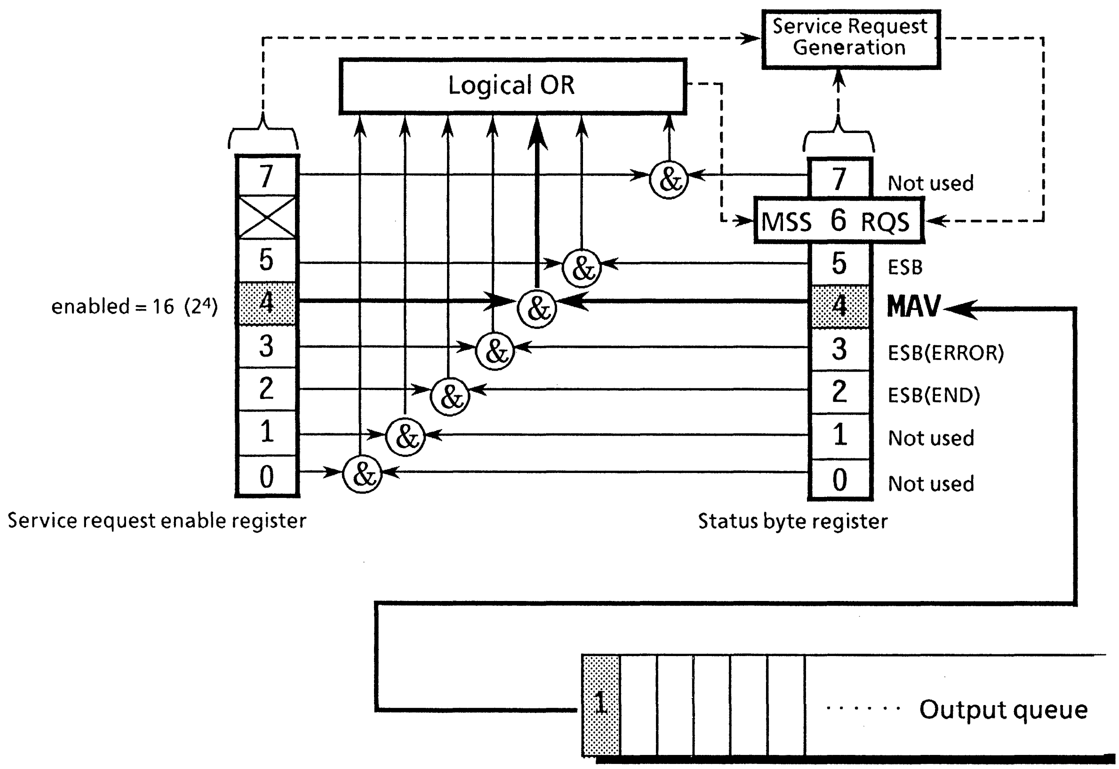
④ Read the ASCII character 1 in the output queue and discard it.

⇩

⑤ Wait for the SRQ interrupt (MAV summary message) ···· Value of the status byte: $2^6 + 2^4 = 80$

⇩

To the next operation



SECTION 9
DETAILS OF DEVICE MESSAGES

This section explains the details of the device messages in the table.

Formats and usage example in this section are explained in the HP-BASIC of the Hewlett-Packard HP9000 Series.

TABLE OF CONTENTS

(Blank)

This section explains each device message by group. Each group is corresponded to the front and rear panel of MP1764A. Groups are specified according to the setting or request contents.

9.1 Table of Device Messages

Control messages and data request messages that are stipulated in the MP1764A specifications are explained in the listing order.

Check the details of each command by referring to the page numbers listed in the last column of the table under "Device message details".

9.1.1 Table of Device Messages (in the Alphabetic order)

An alphabetic list of each control message and data request message is shown in Table 9-1.

Table 9-1 Table of Device Messagess (Alphabetic order)

Function	Control message		Data request message	Device message details	
	Header part	Numeric data part	Header part	Section	Page
Number of pages	ADR	NR1 format	ADR?	PATTERN	P9-64
Pattern data preset (All pages, all bits)	ALL	NR1 format	–	PATTERN	P9-85
Alarm monitor (alarm detection)	ALM	NR1 format	ALM?	Others	P9-129
Alternate pattern A / B switch selection	ALT	NR1 format	ALT?	PATTERN	P9-59
Alarm measurement result	–	–	AMD?	MEASUREMENT	P9-114
Pattern . bit	BIT	NR1 format HEX format	BIT?	PATTERN	P9-65
BURST measurement mode	BST	NR1 format	BST?	Others	P9-137
Intermediate data calculation	CAL	NR1 format	CAL?	Others	P9-138
Bit window pattern	CHM	NR1 format HEX format	CHM?	PATTERN	P9-67
Clock loss state	–	–	CLI?	MEASUREMENT	P9-94
Clock loss processing function	CLS	NR1 format	CLS?	Others	P9-134
Clock input phase (delay)	CPA	NR1 format	CPA?	INPUT	P9-28
Clock input polarity	CPL	NR1 format	CPL?	INPUT	P9-34
Clock input termination voltage	CTM	NR1 format	CTM?	INPUT	P9-31
Intermediate result display	CUR	NR1 format	CUR?	MEASUREMENT	P9-98
Floppy data delete	DEL	NR1 format	–	MEMORY	P9-44
Measurement data length	DLN	NR1 format	DLN?	PATTERN	P9-61
Delay status	–	–	DLY?	INPUT	P9-32
Measurement result display mode	DMS	NR1 format	DMS?	MEASUREMENT	P9-97
1-second data print threshold selection	DOT	NR1 format	DOT?	Others	P9-146
Display selection	DSP	NR1 format	DSP?	PATTERN	P9-58
Data input threshold voltage	DTH	NR2 format	DTH?	INPUT	P9-26
Data input termination voltage	DTM	NR1 format	DTM?	INPUT	P9-30

Table 9-1 Table of Device Messagess (Alphabetic order: contd.)

Function	Control message		Data request message	Device message details	
	Header part	Numeric data part	Header part	Section	Page
Error analysis data *1	–	–	EAB?	PATTERN	P9-72
Error analysis page *1	EAP	NR1 format	EAP?	PATTERN	P9-74
Error analysis trigger *1	EAT	NR1 format	EAT?	PATTERN	P9-77
Error count measurement results	–	–	EC?	MEASUREMENT	P9-109
Clears measurement data from buffer	EDC	–	–	MEASUREMENT	P9-117
Stores measurement data in the buffer	EDS	–	–	MEASUREMENT	P9-116
%EFI measurement results	–	–	EFI?	MEASUREMENT	P9-111
EI measurement results	–	–	EI?	MEASUREMENT	P9-110
EI, %EFI interval time	EIT	NR1 format	EIT?	Others	P9-140
Eye margin measurement display switching	EME	NR1 format	EME?	INPUT	P9-35
Measurement data output	–	–	END?	MEASUREMENT	P9-118
Error performance data print selection	EPF	NR1 format	EPF?	Others	P9-143
Error ratio measurement results	–	–	ER?	MEASUREMENT	P9-108
Error detection status	–	–	ERS?	MEASUREMENT	P9-96
Starts Eye margin measurement	EST	NR1 format	EST?	INPUT	P9-36
Error performance data threshold selection	ETH	NR1 format	ETH?	Others	P9-136
Error detection mode selection	ETY	NR1 format	ETY?	Others	P9-139
Eye margin measurement (Error ratio selection)	EYT	NR1 format	EYT?	INPUT	P9-37
FD error messages	–	–	FDE?	MEMORY	P9-50
FD format	FDF	–	–	MEMORY	P9-51
File No. / direct mode switching	FIL	NR1 format	FIL?	MEMORY	P9-42
Frame length	FLN	NR1 format	FLN?	PATTERN	P9-60
Memory FD mode	–	–	FMD?	MEMORY Others	P9-39
Data print format	FMT	NR1 format	FMT?	Others	P9-141

Table 9-1 Table of Device Messagess (Alphabetic order: contd.)

Function	Control message		Data request message	Device message details	
	Header part	Numeric data part	Header part	Section	Page
Clock frequency measurement result	–	–	FRQ?	MEASUREMENT	P9-112
File contents retrieving	–	–	FSH?	MEMORY	P9-40
GPIB 2 address	GPA	NR1 format	GPA?	Others	P9-132
Bit window preset (All pages, all bits)	HAL	NR1 format	–	PATTERN	P9-89
Bit window preset (1 page, all bits)	HPS	NR1 format	–	PATTERN	P9-90
Clears measurement intermediate data from buffer	IMC	–	–	MEASUREMENT	P9-122
Intermediate measurement data output	–	–	IMD?	MEASUREMENT	P9-123
Stores intermediate measurement data in buffer	IMS	–	–	MEASUREMENT	P9-121
Intermediate measurement data print	ITM	NR1 format	ITM?	Others	P9-144
Measurement interval time	ITV	NR1 format	ITV?	Others	P9-148
Pattern logic	LGC	NR1 format	LGC?	PATTERN	P9-53
Floppy disk access status	–	–	MAC?	MEMORY	P9-49
Block window preset (All pages, all bits)	MAL	NR1 format	–	PATTERN	P9-87
Memory function switching	MEM	NR1 format	MEM?	MEMORY	P9-47
Block window pattern	MGB	NR1 format HEX format	MGB?	PATTERN	P9-70
Block window ON / OFF	MGE	NR1 format	MGE?	PATTERN	P9-76
Measurement mode	MOD	NR1 format	MOD?	MEASUREMENT	P9-99
Alarm monitor (error detection)	MON	NR1 format	MON?	Others	P9-130
Block window preset (1 page, all data)	MPS	NR1 format	–	PATTERN	P9-88
Number of bytes of block window data output	–	–	MRD?	PATTERN	P9-83
PRBS mark ratio	MRK	NR1 format	MRK?	PATTERN	P9-56
Bit window ON / OFF	MSE	NR1 format	MSE?	PATTERN	P9-75
Bit window page	MSK	NR1 format	MSK?	PATTERN	P9-69

Table 9-1 Table of Device Messagess (Alphabetic order: contd.)

Function	Control message		Data request message	Device message details	
	Header part	Numeric data part	Header part	Section	Page
Measurement status	–	–	MSR?	MEASUREMENT	P9-103
Number of bytes of block window data input	MWT	NR1 format	–	PATTERN	P9-81
1-second data print	OSC	NR1 format	OSC?	Others	P9-145
1-second data measurement result	–	–	OSD?	MEASUREMENT	P9-113
Number of pages	PAG	NR1 format	PAG?	PATTERN	P9-64
Eye margin measurement result (Phase)	–	–	PHM?	INPUT	P9-29
PAGE / PATTERN SYNC POSITION switch	PPD	NR1 format	PPD?	PATTERN	P9-92
Measurement period	PRD	NR1 format	PRD?	MEASUREMENT	P9-107
Printer function	PRN	NR1 format	PRN?	Others	P9-127
Manual print	PSA	–	–	Others	P9-128
PATTERN SYNC POSITION	PSP	NR1 format	PSP?	PATTERN	P9-91
Pattern data preset (1 page, all bits)	PST	NR1 format	–	PATTERN	P9-86
Paper saving function	PSV	NR1 format	PSV?	Others	P9-147
Number of steps of ZERO SUBST and PRBS	PTN	NR1 format	PTN?	PATTERN	P9-55
Measurement pattern	PTS	NR1 format	PTS?	PATTERN	P9-54
Floppy data recall	RCL	NR1 format	–	MEMORY	P9-43
Number of bytes of pattern data output	–	–	RED?	PATTERN	P9-80
Floppy data resave	RSV	NR1 format	–	MEMORY	P9-46
Internal timer setting	RTM	NR1 format	RTM?	MEASUREMENT	P9-106
Floppy data save	SAV	NR1 format	–	MEMORY	P9-45
Number of mark ratio AND bit shifts	SFT	NR1 format	SFT?	Others	P9-133
Sync loss state	–	–	SLI?	MEASUREMENT	P9-95
Sync loss processing	SLS	NR1 format	SLS?	Others	P9-135
Sync output selection	SOP	NR1 format	SOP?	Others	P9-131
Automatic phase threshold search	SRH	NR1 format	SRH?	INPUT	P9-33

Table 9-1 Table of Device Messagess (Alphabetic order: contd.)

Function	Control message		Data request message	Device message details	
	Header part	Numeric data part	Header part	Section	Page
Measurement start and restart	STA	–	–	MEASUREMENT	P9-100
Measurement stop	STO	–	–	MEASUREMENT	P9-101
Automatic synchronous threshold	SYE	NR1 format	SYE?	MEASUREMENT	P9-104
Synchronous method	SYM	NR1 format	SYM?	PATTERN	P9-57
Automatic synchroniztion	SYN	NR1 format	SYN?	MEASUREMENT	P9-103
Eye margin measurement result (threshold)	–	–	THM?	INPUT	P9-27
Threshold EI, EFI data print	THR	NR1 format	THR?	Others	P9-142
Real time / Measurement-time display switching	TIM	NR1 format	TIM?	MEASUREMENT	P9-105
Number of pattern data input bytes	WRT	NR1 format	–	PATTERN	P9-78
ZERO SUBST length	ZLN	NR1 format	ZLN?	PATTERN	P9-63

*1) Option 01: A command which is effective only when an error analysis function is provided.

9.1.2 Device Messages (Panel correspondence)

Figures 9-1 (1) to (7) and Table 9-2 (1) to (7) show the correspondence of control messages and data arequest messages to the panel keys.

- INPUT section

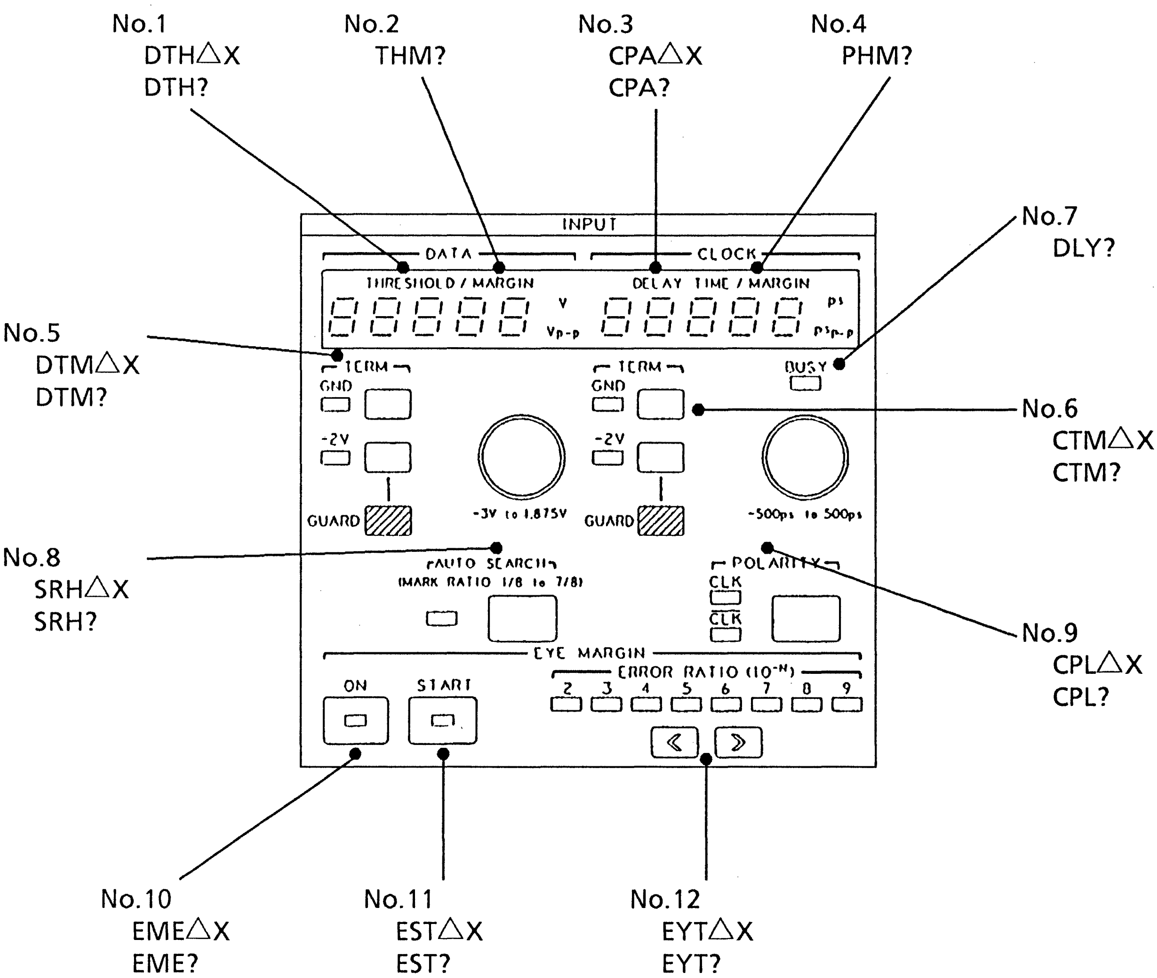
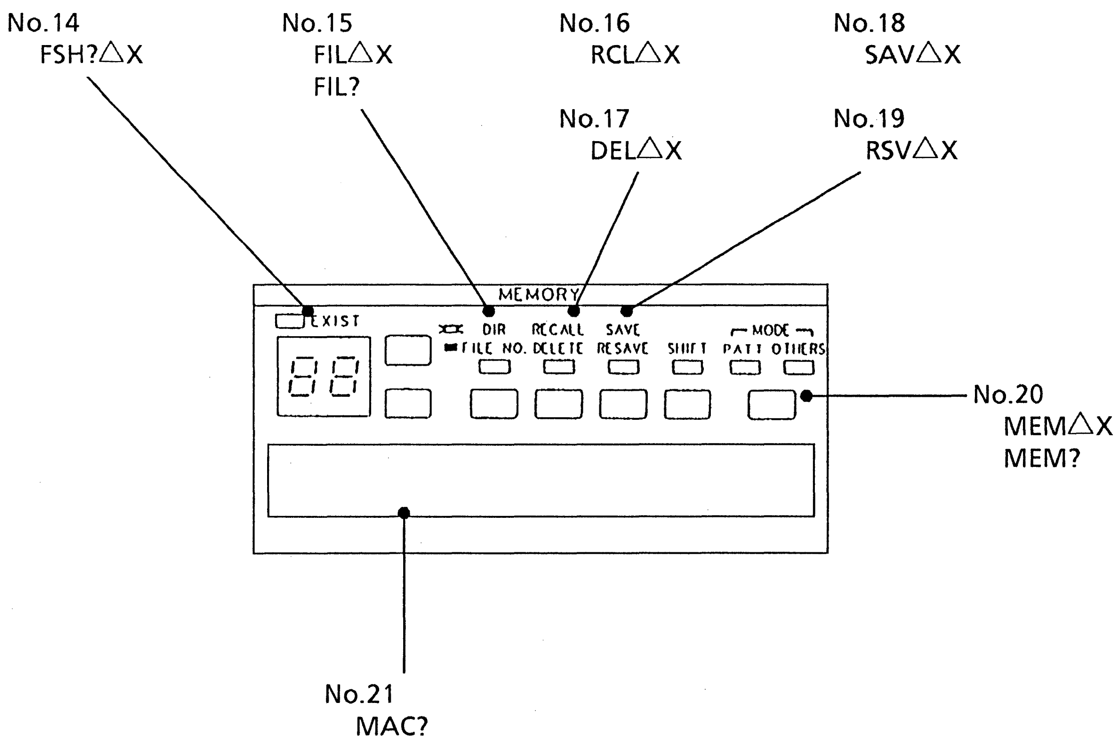


Fig. 9-1-(1) INPUT Section

Table 9-2-(1) Table of Device Messages (INPUT section)

Function	Control message		Data request message	Device message details	
	Header part	Numeric data part	Header part	Item No.	Page
● INPUT section					
Data input threshold voltage	DTH	NR2 format	DTH?	1	P9-26
Eye margin measurement result (threshold)	–	–	THM?	2	P9-27
Clock input phase (delay)	CPA	NR1 format	CPA?	3	P9-28
Eye margin measurement result (phase)	–	–	PHM?	4	P9-29
Data input termination voltage	DTM	NR1 format	DTM?	5	P9-30
Clock input termination voltage	CTM	NR1 format	CTM?	6	P9-31
Delay status	–	–	DLY?	7	P9-32
Automatic phase threshold search	SRH	NR1 format	SRH?	8	P9-33
Clock input polarity	CPL	NR1 format	CPL?	9	P9-34
Eye margin measurement display switching	EME	NR1 format	EME?	10	P9-35
Eye margin measurement start	EST	NR1 format	EST?	11	P9-36
Eye margin measurment (error ratio selection)	EYT	NR1 format	EYT?	12	P9-37

● MEMORY Section



- FD mode : No. 13 FMD?
- FD error message : No. 22 FDE?
- FD format : No. 23 FDF

Fig. 9-1-(2) MEMORY Section

Table 9-2-(2) Table of Device Messagess (MEMORY section)

Function	Control message		Data request message	Device message details	
	Header part	Numeric data part	Header part	Item No.	Page
● MEMORY section					
Memory FD mode	–	–	FMD?	13	P9-39
File contents search	–	–	FSH?	14	P9-40
File No. / directory mode switching	FIL	NR1 format	FIL?	15	P9-42
Floppy data recall	RCL	NR1 format	–	16	P9-43
Floppy datas delete	DEL	NR1 format	–	17	P9-44
Floppy data save	SAV	NR1 format	–	18	P9-45
Floppy data resave	RSV	NR1 format	–	19	P9-46
Memory function switching	MEM	NR1 format	MEM?	20	P9-47
Floppy access condition	–	–	MAC?	21	P9-49
FD error message	–	–	FDE?	22	P9-50
FD format	FDF	–	–	23	P9-51

- PATTERN Section

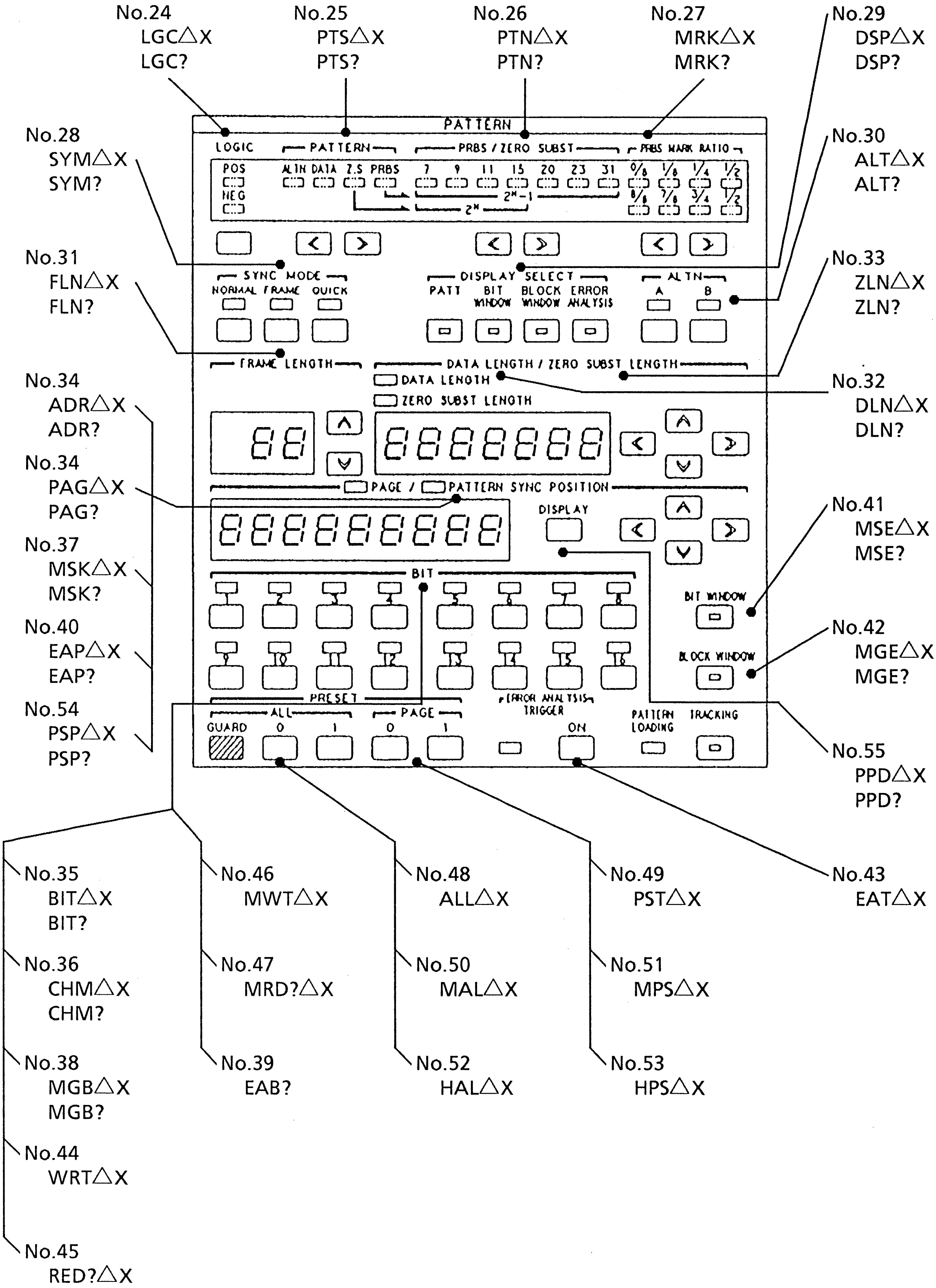


Fig. 9-1-(3) PATTERN Section

Table 9-2-(3) Table of Device Messagess (PATTERN section)

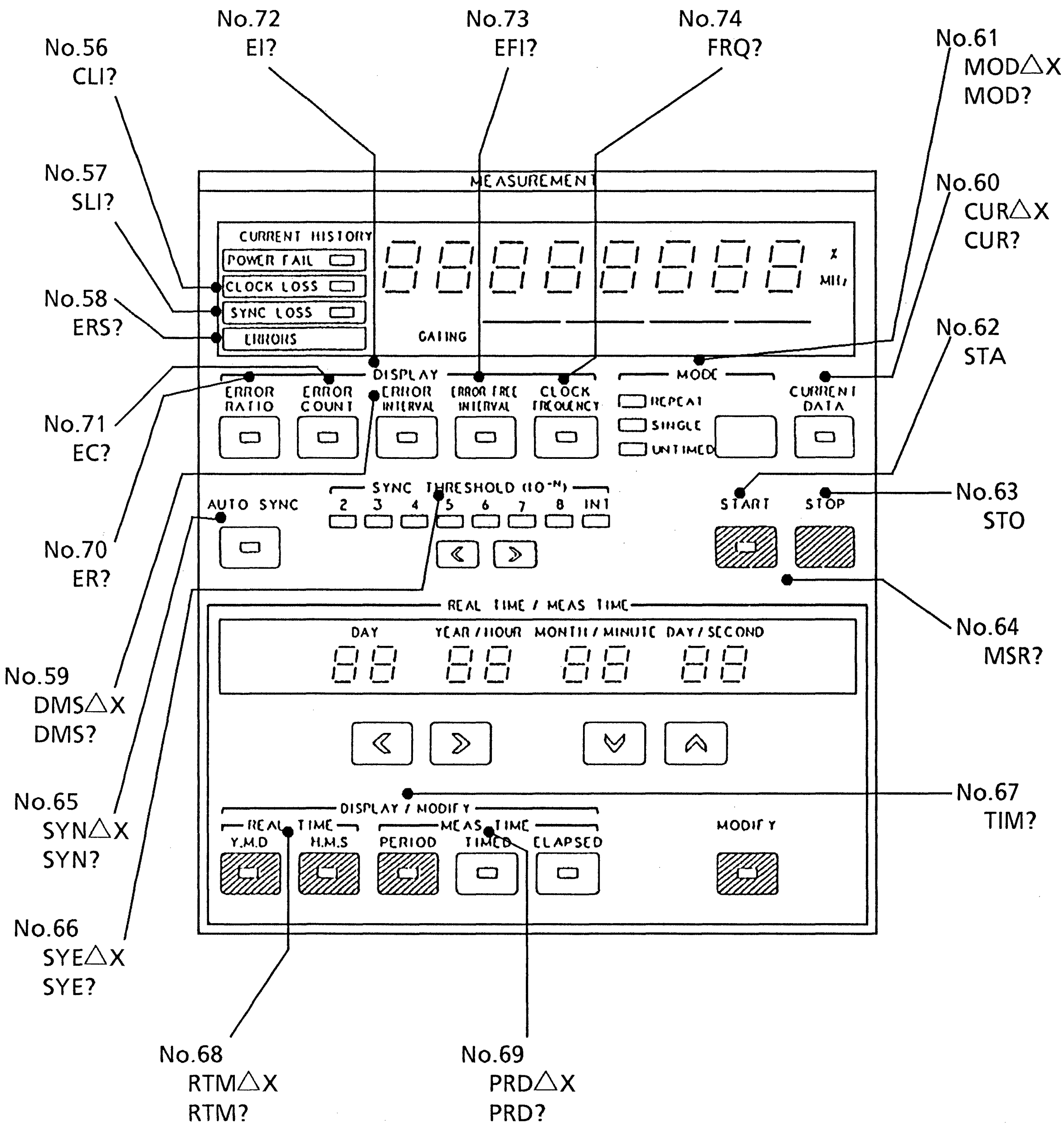
Function	Control message		Data request message	Device message details	
	Header part	Numeric data part	Header part	Item No.	Page
● PATTERN section					
Pattern logic	LGC	NR1 format	LGC?	24	P9-53
Measurement pattern	PTS	NR1 format	PTS?	25	P9-54
Number of steps of ZERO SUBST and PRBS	PTN	NR1 format	PTN?	26	P9-55
PRBS mark ratio	MRK	NR1 format	MRK?	27	P9-56
Synchronous method	SYM	NR1 format	SYM?	28	P9-57
Display selection	DSP	NR1 format	DSP?	29	P9-58
Alternate pattern A / B switching	ALT	NR1 format	ALT?	30	P9-59
Frame length	FLN	NR1 format	FLN?	31	P9-60
Measurement data length	DLN	NR1 format	DLN?	32	P9-61
ZERO SUBST length	ZLN	NR1 format	ZLN?	33	P9-63
Number of pages	PAG ADR	NR1 format HEX format	PAG? ADR?	34	P9-64
Pattern bit	BIT	NR1 format HEX format	BIT?	35	P9-65
Bit window pattern	CHM	NR1 format HEX format	CHM?	36	P9-67
Bit window page	MSK	NR1 format	MSK?	37	P9-69
Block window pattern	MGB	NR1 format HEX format	MGB?	38	P9-70
Error analysis data *1	–	–	EAB?	39	P9-72
Error analysis page *1	EAP	NR1 format	EAP?	40	P9-74
Bit window ON / OFF	MSE	NR1 format	MSE?	41	P9-75
Block window ON / OFF	MGE	NR1 format	MGE?	42	P9-76
Error analysis trigger *1	EAT	NR1 format	EAT?	43	P9-77
Number of pattern data input bytes	WRT	NR1 format	–	44	P9-78
Number of pattern data output bytes	–	–	RED?	45	P9-80

Table 9-2-(3) Table of Device Messagess (PATTERN section: contd.)

Function	Control message		Data request message	Device message details	
	Header part	Numeric data part	Header part	Item No.	Page
● PATTERN section (contd.)					
Number of bytes of block window data input	MWT	NR1 format	–	46	P9-81
Number of bytes of block window data output	–	–	MRD?	47	P9-83
Pattern data preset (all pages, all bits)	ALL	NR1 format	–	48	P9-85
Pattern data preset (1 page, all bits)	PST	NR1 format	–	49	P9-86
Block window preset (all pages, all bits)	MAL	NR1 format	–	50	P9-87
Block window preset (1 page, all bits)	MPS	NR1 format	–	51	P9-88
Bit window preset (all pages, all bits)	HAL	NR1 format	–	52	P9-89
Bit window preset (1 page, all bits)	HPS	NR1 format	–	53	P9-90
Pattern sync trigger position	PSP	NR1 format	PSP?	54	P9-91
Page / pattern sync trigger position display switch	PPD	NR1 format	PPB?	55	P9-92

*1) Option 01: A command which is effective only when an error analysis function is provided.

- MEASUREMENT Section



· One-second data output	: No.75	OSD?△X
· Alarm measurement results	: No.76	AMD?△X
· Result data buffer clear	: No.78	EDC
buffer store	: No.77	EDS
output	: No.79	END?△X
· Intermediate data buffer clear	: No.81	IMC
buffer store	: No.80	IMS
output	: No.82	IMD?△X

Fig. 9-1-(4) MEASUREMENT Section

Table 9-2-(4) Table of Device Messagess (MEASUREMENT section)

Function	Control message		Data request message	Device message details	
	Header part	Numeric data part	Header part	Item No.	Page
● MEASUREMENT section					
Clock off status	–	–	CLI?	56	P9-94
Synchronous off status	–	–	SLI?	57	P9-95
Error detection status	–	–	ERS?	58	P9-96
Measurement result display mode	DMS	NR1 format	DMS?	59	P9-97
Intermediate result display function	CUR	NR1 format	CUR?	60	P9-98
Measurement mode	MOD	NR1 format	MOD?	61	P9-99
Measurement start and restart	STA	–	–	62	P9-100
Measurement stop	STO	–	–	63	P9-101
Measurement condition	–	–	MSR?	64	P9-102
Automatic synchronization	SYN	NR1 format	SYN?	65	P9-103
Automatic synchronous threshold	SYE	NR1 format	SYE?	66	P9-104
Real time / Measurement time display switching	TIM	NR1 format	TIM?	67	P9-105
Internal timer setting	RTM	NR1 format	RTM?	68	P9-106
Measurement period setting	PRD	NR1 format	PRD?	69	P9-107
Error rate measurement result	–	–	ER?	70	P9-108
Number of errors measurement result	–	–	EC?	71	P9-109
Number of EIs measurement result	–	–	EI?	72	P9-110
%EFI measurement result	–	–	EFI?	73	P9-111
Clock frequency data	–	–	FRQ?	74	P9-112
1-second data measurement result	–	–	OSD?	75	P9-113
Alarm measurement result	–	–	AMD?	76	P9-114
Stores measurement data in buffer	EDS	–	–	77	P9-116
Clears measurement data from buffer	EDC	–	–	78	P9-117

Table 9-2-(4) Table of Device Messagess (MEASUREMENT section: contd.)

Function	Control message		Data request message	Device message details	
	Header part	Numeric data part	Header part	Item No.	Page
● MEASUREMENT section (contd.)					
Measurement termination data output	–	–	END?	79	P9-118
Stores intermediate measurement data in buffer	IMS	–	–	80	P9-121
Clears intermediate measurement data from buffer	IMC	–	–	81	P9-122
Outputs intermediate data during measurement	–	–	IMD?	82	P9-123

• Other section

(Front panel)

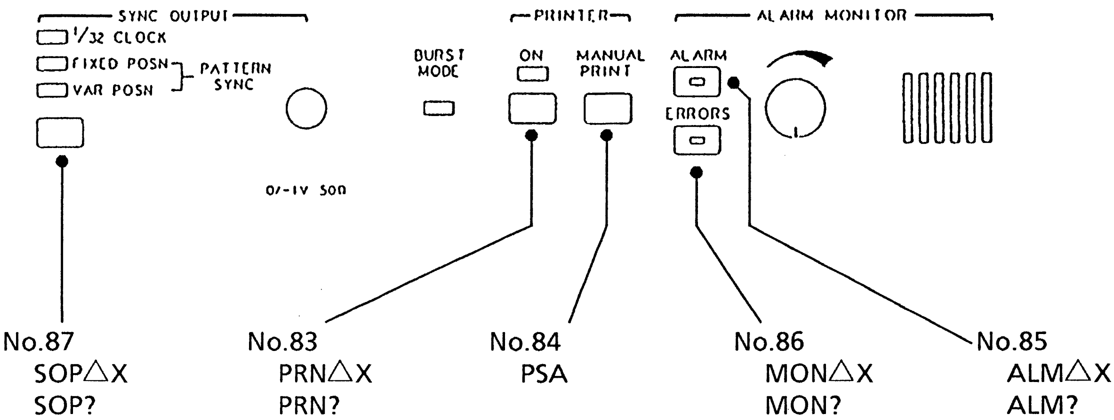


Fig. 9-1-(5) Other section (Front panel)

(Rear panel)

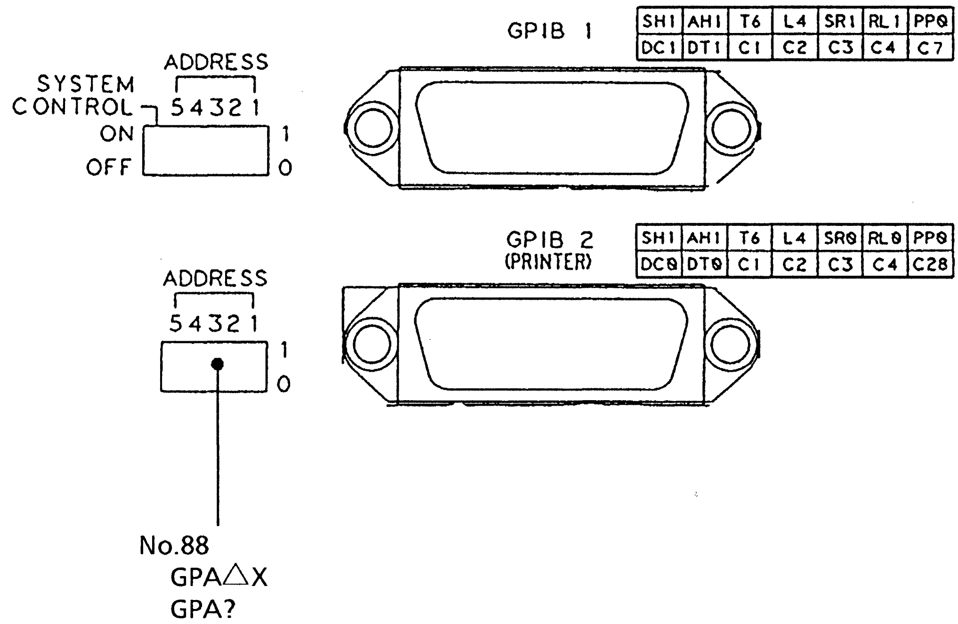


Fig. 9-1-(6) Other section (Rear panel. GPIB)

Notes

There are two setting methods for the GPIB 2 address; one is set via GPIB, and other is set by setting address SW on the rear panel.

When an address is set via GPIB using the GPA command, the set content is held while the mainframe is in remote status. (Except when one of the initialize commands, INI, or *RST, is issued.)

However, if a local state has been made, an address SW status on the rear panel has priority, so the set contents of the remote status become invalid.

Table 9-2-(5) Table of Device Messagess (Other section: front panel)

Function	Control message		Data request message	Device message details	
	Header part	Numeric data part	Header part	Item No.	Page
● Other section (front panel)					
Printer function	PRN	NR1 format	PRN?	83	P9-127
Manual print	PSA	NR1 format	–	84	P9-128
Alarm monitor (alarm detection)	ALM	NR1 format	ALM?	85	P9-129
Alarm monitor (Error detection)	MON	NR1 format	MON?	86	P9-130
Sync output selection	SOP	NR1 format	SOP?	87	P9-131

Table 9-2-(6) Table of Device Messagess (Other section: rear panel GPIB)

Function	Control message		Data request message	Device message details	
	Header part	Numeric data part	Header part	Item No.	Page
● Other section (rear panel GPIB)					
GPIB2 address	GPA	NR1 format	GPA?	88	P9-132

(Rear panel function SW)

No.97 FMT△X FMT?

No.98 THR△X THR?

No.99 EPF△X EPF?

No.100 ITM△X ITM?

No.101 OSC△X OSC?

No.102 DOT△X DOT?

No.103 PSV△X PSV?

No.104 ITV△X ITV?

No.13 FMD?

FUNCTION 1 MEASUREMENT DISPLAY

ITEMS	SW	SPECIFICATION
BIT SHIFT NUMBER FOR MARK RATIO VARIED	1	0:1BIT 1:3BITS
CLOCK LOSS EVALUATION	2	0:EXCLUDE 1:INCLUDE
SYNC LOSS EVALUATION	3	0:EXCLUDE 1:INCLUDE
ERROR PERFORMANCE THRESHOLD	4	$0:10^{-3}$ $1:10^{-4}$
BURST MODE	5	0:OFF 1:ON
CURRENT DATA CALCULATION	6	0:PROGRESSIVE 1:IMMEDIATE
ERROR	7,8	00:TOTAL ERROR 01:INSERTION ERROR 10:OMISSION ERROR 11:TOTAL ERROR
INTERVAL TIME (EI,%EFI)	9,10	00:1ms 01:10ms 01:100ms 11:1s

FUNCTION 2 DATA OUTPUT

ITEMS		SW	SPECIFICATION
SHORT FORM OUTPUT		1	0:OFF 1:ON
THRESHOLD EI,EFI,DATA		2	0:OFF 1:ON
ERROR PERFORMANCE DATA		3	0:OFF 1:ON
INTERMEDIATE DATA		4	0:OFF 1:ON
ONE SECOND DATA	OUTPUT	5	0:OFF 1:ON
	OUTPUT THRESHOLD	6,7	00:> 0 $01:>10^{-6}$ $10:>10^{-4}$ $11:>10^{-3}$
	PAPER SAVING	8	0:OFF 1:ON
CURRENT DATA INTERVAL		9	0:100ms 1:200ms
FD FORMATTING TYPE		10	0:1440k/720k 1:1232k/720k

No.89 SFT△X SFT?

No.90 CLS△X CLS?

No.91 SLS△X SLS?

No.92 ETH△X ETH?

No.93 BST△X BST?

No.94 CAL△X CAL?

No.95 ETY△X ETY?

No.96 EIT△X EIT?

Fig. 9-1-(7) Other section (rear panel function SW)

NOTE

When the rear function SW value differs from that set in the remote status, the processing is done as follows:

- The values set in remote mode are maintained while the mainframe is in the remote status. (When an initialize command INI, or *RST is issued, this is initialized.)

  However, if the mainframe becomes local state, function SW status on the rear panel has priority, so the set contents of the remote status become invalid.

Table 9-2-(7) Table of Device Messagess (Other section: rear panel FUNCTION switch)

Function	Control message		Data request message	Device message details	
	Header part	Numeric data part	Header part	Item No.	Page
● Other section (Rear panel FUNCTION switch) Number of shifts of the mark ratio AND shifts	SFT	NR1 format	SFT?	89	P9-133
Clock loss processing	CLS	NR1 format	CLS?	90	P9-134
Synchronous off processing	SLS	NR1 format	SLS?	91	P9-135
Error performance data threshold selection	ETH	NR1 format	ETH?	92	P9-136
BURST measurement mode	BST	NR1 format	BST?	93	P9-137
Intermediate data calculation	CAL	NR1 format	CAL?	94	P9-138
Error detection mode selection	ETY	NR1 format	ETY?	95	P9-139
EI, %EFI interval time	EIT	NR1 format	EIT?	96	P9-140
Data print format	FMT	NR1 format	FMT?	97	P9-141
Threshold EI, EFI data print	THR	NR1 format	THR?	98	P9-142
Error performance data print	EPF	NR1 format	EPF?	99	P9-143
Intermediate data print	ITM	NR1 format	ITM?	100	P9-144
1-second data print	OSC	NR1 format	OSC?	101	P9-145
1-second data print threshold	DOT	NR1 format	DOT?	102	P9-146
Paper saving function	PSV	NR1 format	PSV?	103	P9-147
Measurement interval time	ITV	NR1 format	ITV?	104	P9-148
Memory FD mode	–	–	FMD?	13	P9-39

9.1.3 Detailed Explanation of Device Messages

MP1764A control messages and data request messages are explained in this section.

The explanation below is already described in HP-BASIC of the Hewlett-Pacckard HP9000 Series.

• INPUT section

Each control message in the INPUT section is explained in the following pages. The triangle marks (△) indicates a spece.

1) DTH Data input threshold voltage (**D**ata **TH**reshold)

■ Function

Setting of data input threshold voltages.
Set resolution is 0.001 V.

Header	Program	Query	Response (Number of characters)
DTH	DTH△m	DTH?	DTH△m (FIX 6)

■ Value of m

The range of the data input threshold voltages is set 1.875 to −3.000 V.

Range of numeric values: Max. 1.875
Min. −3.000
Step 0.001

■ Command type

Sequential command

■ Usage restrictions

The command is invalid in the following conditions.

Program: During automatic phase threshold search (AUTO SEARCH) is ON
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ Usage example

Program: When setting the data input threshold voltage to −3.000 V:

```
OUTPUT△700;"DTH△-3"
```

Query: When the data input threshold voltage is 1.000 V:

```
OUTPUT△700;"DTH?"
ENTER△700;B$
PRINT△B$
        ↓
DTH△1.000 (CR/LF) is output.
```

When the data input threshold voltage is −3.000 V:

```
OUTPUT△700;"DTH?"
ENTER△700;B$
PRINT△B$
        ↓
DTH△-3.000 (CR/LF) is output.
```

2) THM?

Eye margin measurement result (threshold)
(THreshold Margin?)

■ **Function**

A threshold margin measurement result is output from the Eye-margin measurement results data.
Measured resolution is 0.001 Vp-p.

Header	Program	Query	Response (Number of characters)
THM	None	THM?	THM△m (FIX 6)

■ **Value of m**

The value of the Eye margin from 0.000 Vp-p to 4.875 Vp-p is output.

Range of numeric values: Max. 4.875
Min. 0.000
Resolution 0.001

Here, if there is no Eye margin result, the following data is output.
−9.999

■ **Command type**

Sequential command

■ **Usage restrictions**

The command is invalid in the following conditions.

Query: None

■ **Usage example**

Query: When the Eye margin measurement result is 1.875 Vp-p:

```
OUTPUT△7ØØ;"THM?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

THM△△1.875 (CR / LF) is output.

If there is no Eye margin measurement result:

```
OUTPUT△7ØØ;"THM?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

THM△−9.999 (CR / LF) is output.

■ **Note**

The Eye margin measurement result becomes the object for non-battery back up. Therefore, care must be taken because the measurement data is erased at power OFF.

3) CPA — Clock input phase (delay) (**C**lock **P**hase **A**djust)

■ **Function**

Clock phase to be input is specified.
The setting resolution is 1 ps.

Header	Program	Query	Response (Number of characters)
CPA	CPA△m	CPA?	CPA△m (FIX 4)

■ **Value of m**

The range of clock input phase voltages from −500 ps to 500 ps is set.

Range of numeric values: Max. 500
Min. −500
Step 1

■ **Command type**

Sequential command

■ **Usage restrictions**

The command is invalid in the following conditions.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ **Usage example**

Program: When the clock input phase is set to 0 ps:

```
OUTPUT△700;"CPA△0"
```

Query: When the clock input phase is 0 ps:

```
OUTPUT△700;"CPA?"
ENTER△700;B$
PRINT△B$
        ↓
CPA△△△△0 (CR / LF) is output.
```

When the the clock input phase is −20 ps:

```
OUTPUT△700;"CPA?"
ENTER△700;B$
PRINT△B$
        ↓
CPA△△-20 (CR / LF) is output.
```

4) PHM? Eye margin measurement result (phase) (PHase Margin?)

■ Function

Phase margin measurement result is output from Eye margin measurement data.
The measured resolution is 1 psp-p.

Header	Program	Query	Response (Number of characters)
PHM	None	PHM?	PHM△m (FIX 6)

■ Value of m

The value of the Eye margin measurement result from 0 to 1000 psp-p is output.

Range of numeric values: Max. 1000
Min. 0
Resolution 1

Here, if there is no Eye margin result, the following data is output.
−9999

■ Command type

Sequential command

■ Usage restrictions

The command is invalid in the following condition:

Query: None

■ Usage example

Query: When the Eye margin measurement result is 100 psp-p:

```
OUTPUT△700;"PHM?"
ENTER△700;B$
PRINT△B$
```

↓

PHM△△△100 (CR / LF) is output.

If there is no Eye margin measurement result:

```
OUTPUT△700;"PHM?"
ENTER△700;B$
PRINT△B$
```

↓

PHM△-9999 (CR / LF) is output.

■ Note

The Eye margin measurement result becomes the object for non-battery back up. Therefore, care must be taken because the measurement data is erased at power OFF.

5) DTM Data input termination voltage (Data TerMination)

■ Function — Data input termination voltage is specified.

Header	Program	Query	Response (Number of characters)
DTM	DTM△m	DTM?	DTM△m (FIX 1)

■ Value of m

Ø : GND
1 : −2 V

■ Command type — Sequential command

■ Usage restrictions — The command is invalid in the following conditions.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ Usage example

Program: When data input termination voltage is set to GND:

```
OUTPUT△7ØØ;"DTM△Ø"
```

Query: When data input termination voltage is set to −2 V:

```
OUTPUT△7ØØ;"DTM?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

DTM△1 (CR / LF) is output.

CAUTION

If the data input termination voltage is set to GND / −2 V, differ from the data signal conditions input, the mainframe and any equipment under test will be damaged. Care must be taken to set it correctly.

6) CTM

Clock input termination voltage
(Clock TerMination)

■ Function

Clock input termination voltage is set.

Header	Program	Query	Response (Number of characters)
CTM	CTM△m	CTM?	CTM△m (FIX 1)

■ Value of m

Ø : GND
1 : −2V

■ Command type

Sequential command

■ Usage restrictions

The command is invalid in the following conditions:

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ Usage example

Program: When clock input termination voltage is set to GND

```
OUTPUT△7ØØ;"CTM△Ø"
```

Query: When the set value of clock input termination voltage is −2 V

```
OUTPUT△7ØØ;"CTM?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

CTM△1 (CR / LF) is output.

CAUTION

If the data input termination voltage is set to GND / −2 V, differ from the data signal conditions input, the mainframe and any epuipment under test will be damaged. Care must be taken to set itcorrectly.

7) DLY? Delay status (DeLaY?)

■ Function

Operation status of the servo motor used to set clock input phases is output.

Header	Program	Query	Response (Number of characters)
DLY	None	DLY?	DLY△m (FIX 1)

■ Value of m

Ø : READY status
1 : BUSY status

■ Command type

Sequential command

■ Usage restrictions

The command is invalid in the following condition.

Query: None

■ Usage example

Query: When the servo motor used to set the clock input phase is not operating:

```
OUTPUT△7ØØ;"DLY?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

DLY△Ø (CR / LF) is output.

■ Note

READY means the servo motor used to set the clock input phase is not operating.
BUSY means the servo motor used to sest the clock input phase is operating.
Therefore, the mainframe sets the clock input phase in BUSY status.

8) SRH

Automatic phase threshold search function
(auto SeaRcH)

■ **Function**

Phases between clock / data input phase and threshold voltages are set to a suitable status.

Header	Program	Query	Response (Number of characters)
SRH	SRH△m	SRH?	SRH△m (FIX 1)

■ **Value of m**

Ø : Automatic phase threshold search OFF
1 : Automatic phase threshold search ON
2 : Failure in automatic phase threshold search

Item 2 above is effective only in response to a query

■ **Command type**

Overlap command

■ **Usage restrictions**

The command is invalid in the following conditions:

Program: When the automatic synchronous function is OFF
If the mark ratio of the measurement pattern is not within 1 / 8 to 7 / 8
When clock loss occurs
When a floppy disk is being accessed
During automatic phase threshold search operation
When ON is set during the automatic phase threshold search failure status
While Eye margin measurement is being executed

Query: None

■ **Usage example**

Program: When setting ON the automatic phase threshold search:

```
OUTPUT;"SRH△1"
```

Query: If an automatic phase threshold search function failed

```
OUTPUT△7ØØ;"SRH?"
ENTER△7ØØ;B$
PRINT△B$
        ↓
SRH△2 (CR / LF) is output.
```

■ **Note**

① Automatic phase threshold search judges the following cases as failures and blinks the AUTO SEARCH key LED on the front panel. The status then becomes SRH△2.
(1) When a clock loss occurs during operation
(2) When synchronous operation cannot be accessed under the search setting status.

② In order to clear the failure of the automatic phase threshold search, set to OFF.
The automatic phase threshold search funciton cannot set to ON again under failed status.

9) CPL Clock input polarity **(Clock PoLarity)**

■ **Function** Clock input polarity is set.

Header	Program	Query	Response (Number of characters)
CPL	CPL△m	CPL?	CPL△m (FIX 1)

■ **Value of m**
Ø : CLOCK
1 : $\overline{\text{CLOCK}}$

■ **Command type** Sequential command

■ **Usage restrictionss** The command is invalid in the following conditions:

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ **Usage example**

Program: When clock input polarity is set to CLOCK
```
OUTPUT△7ØØ;"CPL△Ø"
```

Query: When clock input polarity set value is $\overline{\text{CLOCK}}$
```
OUTPUT△7ØØ;"CPL?"
ENTER△7ØØ;B$
PRINT△B$
```
↓
CPL△1 (CR / LF) is output.

10) EME

Eye margin measurement display switching
(Eye Margin Enable)

■ Function

Contents of 7seg display for clocks and INPUT section data are switched.

Header	Program	Query	Response (Number of characters)
EME	EME△m	EME?	EME△m (FIX 1)

■ Value of m

Ø : The display values of the 7seg display denote the set values of data input threshold voltage and clock input phase.?

1 : The display values of the 7seg display denote the Eye margin measurement results.

■ Command type

Sequential command

■ Usage restrictions

The command is invalid in the following conditions:

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ Usage example

Program: When the display values of the 7 seg-display are setting values of data input threshold voltage set values or clock input phase values

```
OUTPUT△7ØØ;"EME△Ø"
```

Query: When the values in the 7seg display is the Eye margin measurement result

```
OUTPUT△7ØØ;"EME?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

EME△1 (CR / LF) is output.

■ Note

This setting is for effective Eye margin measurement during panel operation.
When the Eye margin measurement is controlled through GPIB command, settings and queries can be made without regard to the eye margin measurement display.

11) EST

Eye margin measurement start **(Eye margin STart)**

■ Function

Eye margin start and stop is set.

Header	Program	Query	Response (Number of characters)
EST	EST△m	EST?	EST△m (FIX 1)

■ Value of m

Ø : Eye margin measurement stop
1 : Eye margin measurement start
2 : Eye margin measurement failure

2 is effective only in response to a query.

■ Command type

Overlap command

■ Usage restrictions

The command is invalid in the following conditions.

Program: When the automatic synchronous function is OFF
If the mark ratio of the measurement pattern is not within 1/8 to 7/8
When clock loss occurs
When a floppy disk is being accessed
During automatic phase threshold search operation
When ON is set during Eye margin measurement failure status
While Eye margin measurement is being executed

Query: None

■ Usage example

Program: When setting the Eye margin measurement to OFF

```
OUTPUT;"EST△Ø"
```

Query: When the Eye margin measurement has failed:

```
OUTPUT△7ØØ;"EST?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

EST△2 (CR/LF) is output.

■ Note

Eye margin measurement function judges the following cases as failures and blinks the EYE MARGIN START key LED on the front panel. The status then becomes EST△2.

(1) When a clock loss occurs during operation.
(2) When synchronous operation cannot be accessed under the Eye margin setting status.

The Eye margin cannot start again under failed status.
To clear the Eye margin failure status, specify OFF setting.
Eye margin measurment cannot restart in the failure status.

12) EYT

Eye margin measurement (error ratio selection)
(EYe margin Threshold)

■ Function

The Eye margin measurement range can be specified by error ratio. The Eye margin can be measured within the specified error ratio.

Header	Program	Query	Response (Number of characters)
EYT	EYT△m	EYT?	EYT△m (FIX 1)

■ Value of m

Ø : ≦1.0E-2
1 : ≦1.0E-3
2 : ≦1.0E-4
3 : ≦1.0E-5
4 : ≦1.0E-6
5 : ≦1.0E-7
6 : ≦1.0E-8
7 : ≦1.0E-9

■ Command type

Sequential command

■ Usage restrictions

The command is invalid in the following conditions:

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ Usage example

Program: When the eye margin measurement error ratio is ≦1.0E-5:
OUTPUT;"EYT△3"

Query: When the eye margin measurement error ratio is ≦1.0E-7:
OUTPUT△7ØØ;"EYT?"
ENTER△7ØØ;B$
PRINT△B$
↓
EYT△5 (CR / LF) is output.

● MEMORY section

Each control message in the MEMORY SECTION is described in the following pages.

(△) indicates a space.

13) FMD? Memory FD mode (memory Fd MoDe?)

■ **Function** Floppy disk format type is output.

Header	Program	Query	Response (Number of characters)
FMD	None	FMD?	FMD△m (FIX 1)

■ **Value of m**
Ø : 1440 k
1 : 720 k
2 : 1232 k
3 : 640 k

■ **Command type** Sequential command

■ **Usage restrictions** The command is invalid in the following condition:
Query: None

■ **Usage example** Query: When the inserted FD is 2DD and is 1440 kB / 720 k formatted:

```
OUTPUT△△7ØØ;"FMD?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

FMD△1 (CR / LF) is output.

■ **Note** Selecting MS-DOS format is done at the rear FUNCTION2 SW10.
Since this setting is decided at the power on status, it cannot be changed after that.

- When FUNCTION2 SW10 is 'OFF'

Capacity at the time of formatting	Sector length [byte / sector]	Number of sectors [sector / track]	Number of tracks [track / site]
1440 kB	512	18	80
720 kB	512	9	80

- When FUNCTION2 is 'ON'

Capacity at the time of formatting	Sector length [byte / sector]	Number of sectors [sector / track]	Number of tracks [track / site]
1232 kB	1024	8	77
640 kB	512	8	80

When this Query is executed without inserting a floppy disk, data item 1440 kB or 1232 kB is output according to the current FDD conditions (FUNCTION2 SW10).

14) FSH? File contents search (File SearcH?)

■ **Function**

Data information saved in a floppy disk is output.

There are three types of objective file names as shown below:

TT∗∗.PTN

RR∗∗.PTN RR∗∗.OTH

Header	Program	Query	Response (Number of characters)
FSH	None	FSH?△m1	FSH△m2,m3,m4,m5, m2 (FIX 7) m3 (FIX 7) m4 (FIX 2) m5 (each FIX 2)

■ **Value of m**

m1 : Selection of first half of a file number.
 0 : First half (No. 0 to 49)
 1 : Last half (No. 50 to 99)
m2 : Unused size
m3 : Used size
m4 : Number of files
m5 : File number of the first half or last half (Only for the objective file name)

■ **Command type**

Sequential command

■ **Usage restrictions**

This command is invalid in the following set condition:

Query: None

■ **Usage example**

Query: When there are file numbers from 1 to 10 in the floppy disk (unused size and used size is an example)

```
OUTPUT△700;"FSH?△0"
ENTER△700;B$
PRINT△B$
```

↓

FSH△△72294,△△18132,10,01,02,03,04,05,06,07,08,09,10 (CR/LF) is output.

When the file cannoot be found on the floppy disk

```
OUTPUT△700;"FSH?△0"
ENTER△700;B$
PRINT△B$
```

↓

FSH△△723968,△△△6144,△0,--(CR/LF) is output.

■ Note

File contents are searched according to the setting status of the memory mode switching (PATTERN / OTHERS).

(1) When the memory mode swtiching is PATT
TT**.PTN or RR**.PTN file is searched.
(2) When the memory mode switching is OTHERS
RR**.OTH file is searched.

If a PATT file saved by MP1763A and a PATT file saved by MP1764A are stored in a floppy disk, the file saved by MP1764A is output prior to the other one.

Searching file contents is output from the nearest directory.
This mainframe does not have an insertion detection function for a floppy, so previous directory information cannot be updated at floppy disk exchange.
Therefore, when inserting or changing floppy disks, always update directory information by selecting the DIR mode.

15) FIL

File No. and directory mode switching
(FILe no. / directory mode)

■ Function

Sets swtiching between file No. mode, and directory mode.

Header	Program	Query	Response (Number of characters)
FIL	FIL△m	FIL?	FIL△m (FIX 1)

■ Value of m

Ø : File No. mode
1 : DIR mode

■ Command type

Sequential command

■ Usage restrictions

The command is invalid in the following set conditions.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ Usage example

Program: When DIR mode is set

```
OUTPUT△7ØØ;"FIL△1"
```

The inserted floppy disk is accessed and the DIR information is saved in internal memory.

Query: When File No. mode is selected

```
OUTPUT△7ØØ;"FIL?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

FIL△Ø (CR / LF) is output.

■ Note

If any errors occur during file accessing, the error nformation is displayed on the MEMORY INDICATOR.
This error indication is cleared by the following settings.

File No. / directory mode switching	(No. 15)
Floppy data recall	(No. 16)
Floppy data delete	(No. 17)
Floppy data save	(No. 18)
Floppy data recall	(No. 19)
Memory mode switching	(No. 20)
FD format	(No. 23)

16) RCL　　Floppy data recall (ReCaLl)

■ **Function**　Sets the contents of a floppy disk for this system.

Header	Program	Query	Response (Number of characters)
RCL	RCL△m	None	None

■ **Value of m**

File names are set within the range 0 to 99.

Range of numeric values: Max. 99
Min. 0
Step 1

■ **Command type**

Overlap command

■ **Usage restrictions**

The command is invalid in the following conditions.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

■ **Usage example**

Program: When the contents of file number 9 are specified
OUTPUT△700;"RCL△9"

■ **Note**

If the specified file cannot be found, an error occurs and the error information is displayed on the MEMORY INDICATOR.
This error indication is cleared by the following settings.

File No. / directory mode switching	(No. 15)
Floppy data recall	(No. 16)
Floppy data delete	(No. 17)
Floppy data save	(No. 18)
Floppy data recall	(No. 19)
Memory mode switching	(No. 20)
FD format	(No. 23)

Also, when an error occurs, Expansion event status resister ESR3 (ERROR) bit 1 is raised as an FD error occurrence bit.

The following files are read according to the memory mode setting status.

PATT mode: RR**.PTN or TT**.PTN
OTHERS mode: RR**.OTH

17) DEL — Floppy data delete (DELete)

■ **Function** — A specified file is deleted in a floppy disk.

Header	Program	Query	Response (Number of characters)
DEL	DEL△m	None	None

■ **Value of m** — File names are set within the range 0 to 99.

Range of values: Max. 99
Min. 0
Step 1

■ **Command type** — Overlap command

■ **Usage restrictions** — The command is invalid in the following set conditions.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

■ **Usage example** — Program: When the contents of file number 9 are specified

`OUTPUT△700;"DEL△9"`

■ **Note** — If there is no specified file, an error occurs and the error information is displayed on the MEMORY INDICATOR.
This error indication is cleared by the following setting.

① File No. / directory mode switching (No. 15)
Floppy data recall (No. 16)
Floppy data delete (No. 17)
Floppy data save (No. 18)
Floppy data recall (No. 19)
Memory mode switching (No. 20)
FD format (No. 23)

Also, when an error occurs, Expansion event status resister ESR3 (ERROR) bit1 is raised as a FD error occurrence bit.

Files saved using MP1763A cannot be deleted by this equipment.

The following files are read according to the memory mode setting status.

PATT mode: RR**.PTN
OTHERS mode: RR**.OTH

18) SAV Floppy data save (SAVe)

■ Function

Setting contents for this equipment are saved to a floppy disk.

Header	Program	Query	Response (Number of characters)
SAV	SAV△m	None	None

■ Value of m

File names are set within the range 0 to 99.

Range of numeric values: Max. 99
Min. 0
Step 1

■ Command type

Overlap command

■ Usage restrictions

The command is invalid in the following conditions.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

■ Usage example

Program: When saving the setting contents of this equipment using file number 9
OUTPUT△7ØØ;"SAV△9"

■ Note

If the specified file cannot be found, an error occurs and error information is displayed on the MEMORY INDICATOR.
This error indication is cleared by the following settings.

File No. / directory mode switching	(No. 15)
Floppy data recall	(No. 16)
Floppy data delete	(No. 17)
Floppy data save	(No. 18)
Floppy data recall	(No. 19)
Memory mode switching	(No. 20)
FD format	(No. 23)

Also, when an error occurs, expansion event status resister ESR3 (ERROR) bit1 is raised as an FD error occurrence bit.

The following files are saved according to the memory mode setting status.

PATT mode: RR∗∗.PTN or TT∗∗.PTN
OTHERS mode: RR∗∗.OTH

19) RSV Floppy data resave (ReSaVe)

■ **Function**

The contents of a floppy disk are specified into the mainframe.

Header	Program	Query	Response (Number of characters)
RSV	RSV△m	None	None

■ **Value of m**

File names are set within the range 0 to 99.

Range of numeric values: Max. 99
Min. 0
Step 1

■ **Command type**

Overlap command

■ **Usage restrictions**

The command is invalid in the following set conditions.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

■ **Usage example**

Program: When overwriting the contents of file No. 9
OUTPUT△700;"RSV△9"

■ **Note**

If the specified file cannot be found, an error occurs and error information is displayed on the MEMORY INDICATOR.
This error indication is cleared in the following setting.

File No. / directory mode switching	(No. 15)
Floppy data recall	(No. 16)
Floppy data delete	(No. 17)
Floppy data save	(No. 18)
Floppy data recall	(No. 19)
Memory mode switching	(No. 20)
FD format	(No. 23)

Also, when an error occurs, expansion event status resister ESR3 (ERROR) bit1 is raised as an FD error occurrence bit.

The following files are resaved according to the memory mode setting status.

PATT mode: RR＊＊.PTN or TT＊＊.PTN
OTHERS mode: RR＊＊.OTH

20) MEM   Memory function switching (**MEMory mode**)

■ Function   PATT / OTHERS switching is specified.

Header	Program	Query	Response (Number of characters)
MEM	MEM△m	MEM?	MEM△m (FIX 1)

■ Value of m

Ø : PATT mode
1 : OTHERS mode

■ Command type   Sequential command

■ Usage restrictions   The command is invalid in the following condition.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ Usage example

Program: When memory mode PATT is specified

```
OUTPUT△7ØØ;"MEM△Ø"
```

Query: When memory mode OTHERS is specified

```
OUTPUT△7ØØ;"MEM?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

MEM△1 (CR / LF) is output.

■ Note

PATT mode denotes PATTern mode.
In this case, the objective content is the contents of the PATTERN section.
OTHERS MODE includes contents other than measurement data in PATTERN mode.

If any errors occur during the file accessing, error information is displayed on the MEMORY INDICATOR.
This error indication is cleared by the following settings.

File No. / directory mode switching (No. 15)
Floppy data recall (No. 16)
Floppy data delete (No. 17)
Floppy data save (No. 18)
Floppy data recall (No. 19)
Memory mode switching (No. 20)
FD format (No. 23)

Also, when an error occurs, expansion event status resister ESR3 (ERROR) bit 1 is raised as an FD error occurrence bit.

The following files are resaved according to the memory mode setting status.

PATT mode:	RR**.PTN or TT**.PTN
OTHERS mode:	RR**.OTH

21) MAC? Floppy disk access status (**M**emory **A**ccess **C**ondition?)

■ Function — Floppy disk access condition is output.

Header	Program	Query	Response (Number of characters)
MAC	None	MAC?	MAC△m (FIX 1)

■ Value of m
- Ø : Non access condition
- 1 : Access condition

■ Command type — Sequential command

■ Usage restrictions — The command is invalid in the following condition.

Query: None

■ Usage example

Query: When a floppy disk is not accessed

```
OUTPUT△7ØØ;"MAC?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

MAC△Ø (CR / LF) is output.

22) FDE? FD error message (FD Error message?)

■ **Function**

FD error information is output.

Header	Program	Query	Response (Number of characters)
FDE	None	FDE?	FDE△m (FIX 2)

■ **Value of m**

Error message
- Ø : E0 (Error due to wrong format)
- 1 : E1 (An attempt was made to write to a write-protected file)
- 2 : E2 (Insufficient write area)
- 3 : E3 (The file name specified for reading could not be found)
- 4 : E4 (Saving was attempted with a file name already in use)
- 5 : E5 (Write error)
- 6 : E6 (Read error)
- 7 : E7 (DMA transfer error)
- 8 : E8 (Other error)
- 9 : E9 (Hardware trouble error)
- 1Ø : No error

■ **Command type**

Sequential command

■ **Usage restrictions**

The command is invalid in the following condition.

Query: None

■ **Usage example**

Query: When a hardware trouble error occurs

```
OUTPUT△7ØØ;"FDE?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

FDE△9 (CR / LF) is output.

■ **Note**

This error indication is cleared in the following settings.

File No. / directory mode switching	(No. 15)
Floppy data recall	(No. 16)
Floppy data delete	(No. 17)
Floppy data save	(No. 18)
Floppy data recall	(No. 19)
Memory mode switching	(No. 20)
FD format	(No. 23)

Also, when an error occurs, expansion event status register ESR3 (ERROR) bit 1 is raised as an FD error occurrence bit.

23) FDF   FD format (FD Format)

■ **Function**

Floppy disk is formatted.
Select the formatting type with the FUNCTION2 SW10 on the rear panel.
2HD or 2DD is decided automatically according to the FD inserted.

Header	Program	Query	Response (Number of characters)
FDF	FDF	None	None

■ **Value of m**

None

■ **Command type**

Overlap command

■ **Usage restrictions**

The command is invalid in the following condition.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed
When the directory mode is switched, from File No. / Directory Mode.

■ **Usage example**

Program: Floppy disk can be formatted.
OUTPUT△700;"FDF"

■ **Note**

If any errors occur during file accessing, error information is displayed on the MEMORY INDICATOR.
This error indication is cleared by the following settings.

File No. / directory mode switching	(No. 15)
Floppy data recall	(No. 16)
Floppy data delete	(No. 17)
Floppy data save	(No. 18)
Floppy data recall	(No. 19)
Memory mode switching	(No. 20)
FD format	(No. 23)

Also, when an error occurs, expansion event status register ESR3 (ERROR) bit 1 is raised as a FD error occurrence bit.

• PATTERN section

Each control message in the PATTERN section is explained on thefollowing pages.

The triangle marks (△) indicates a space.

24) LGC Pattern logic (LoGiC mode)

■ **Function**

Data logic is specified.

The relationship between logical data output and actual data output is different in the ALTERNATE / DATA / ZERO SUBST and in PRBS. (Refer to the Function and Operation Instruction Manual)

Header	Program	Query	Response (Number of characters)
LGC	LGC△m	LGC?	LGC△m (FIX 1)

■ **Value of m**

0 : Positive
1 : Negative

■ **Command type**

Sequential command

■ **Usage restrictions**

The command is invalid in the following condition.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ **Usage example**

Program: When the positive logic is specified

```
OUTPUT△700;"LGC△0"
```

Query: When the negative logic is specified

```
OUTPUT△700;"LGC?"
ENTER△700;B$
PRINT△B$
```

↓

LGC△1 (CR / LF) is output.

■ **Note**

When a pattern logic is specified in PRBS mode, the pattern mark ratio is switched corresponding to the logic.

- Positive logic 0 / 8, 1 / 8, 1 / 4, 1 / 2
- Negative logic 8 / 8, 7 / 8, 3 / 4, $\overline{1/2}$

25) PTS Measurement pattern (PaTtern Select)

■ Function
Measurement pattern is specified.

Header	Program	Query	Response (Number of characters)
PTS	PTS△m	PTS?	PTS△m (FIX 1)

■ Value of m
Ø : ALTERNATE
1 : DATA
2 : ZERO SUBST
3 : PRBS

■ Command type
Overlap command

■ Usage restrictions
The command is invalid in the following condition.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ Usage example
Program: When measurement pattern is specified as alternate
```
OUTPUT△7ØØ;"PTS△Ø"
```

Query: When the measurement pattern is specified as Data
```
OUTPUT△7ØØ;"PTS?"
ENTER△7ØØ;B$
PRINT△B$
```
↓
PTS△1 (CR / LF) is output.

■ Note
When the measurement pattern is switched, the previous setting status is returned.
For example, when the PRBS pattern is specified, the step numbers and pattern mark ratio of the PRBS pattern is returned.

26) PTN

ZERO SUBST / PRBS step numbers
(zero subst / prbs PaTterN mode)

■ Function

ZERO SUBST / PRBS pattern is specified.

Header	Program	Query	Response (Number of characters)
PTN	PTN△m	PTN?	PTN△m (FIX 1)

■ Value of m

ZERO SUBST
2 : 2^7
3 : 2^9
5 : 2^{11}
6 : 2^{15}

PRBS
2 : 2^7-1
3 : 2^9-1
5 : $2^{11}-1$
6 : $2^{15}-1$
7 : $2^{20}-1$
8 : $2^{23}-1$
9 : $2^{31}-1$

■ Command type

Overlap command

■ Usage restrictions

The command is invalid in the following conditions.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed
When the measurement pattern is Alternate.Data

Query: The following is invalid, and ERR (CR / LF) is output.
When the measurement patterns are Alternate, Data

■ Usage example

Program: When the measurement pattern is specified as PRBS 2^7-1

```
OUTPUT△700;"PTN△2"
```

Query: When the measurement pattern logic is specified as PRBS $2^{31}-1$

```
OUTPUT△700;"PTN?"
ENTER△700;B$
PRINT△B$
```

↓

PTN△9 (CR / LF) is output.

: When the measurement pattern is specified as DATA

```
OUTPUT△700;"PTN?"
ENTER△700;B$
PRINT△B$
```

↓

ERR (CR / LF) is output.

27) MRK Pattern mark ratio (MaRK ratio mode)

■ **Function** The mark ratio for the PRBS pattern is specified.

Header	Program	Query	Response (Number of characters)
MRK	MRK△m	MRK?	MRK△m (FIX 1)

■ **Value of m**

	Positive logic	Negative logic
Ø :	0/8	8/8
1 :	1/8	7/8
2 :	1/4	3/4
3 :	1/2	1/2

■ **Command type** Sequential command

■ **Usage restrictions** The command is invalid in the following condition.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed
When measurement patterns are Alternate, Data, Zero subst

Query: The following case is invalid and ERR (CR / LF) is output.
When measurement patterns are Alternate, Data, Zero subst

■ **Usage example**

Program; When the pattern mark ratio is specified as 0 / 8

```
OUTPUT△7ØØ;"MRK△Ø"
```

Query: When the pattern mark ratio is specified as 1 / 8

```
OUTPUT△7ØØ;"MRK?"
ENTER△7ØØ;B$
PRINT△B$
        ↓
MRK△1 (CR / LF) is output.
```

: When the measurement pattern is specified as DATA

```
OUTPUT△7ØØ;"MRK?"
ENTER△7ØØ;B$
PRINT△B$
        ↓
ERR (CR / LF) is output.
```

28) SYM Synchronous method (SYnc Mode)

■ Function — Synchronous method is specified.

Header	Program	Query	Response (Number of characters)
SYM	SYM△m	SYM?	SYM△m (FIX 1)

■ Value of m

Ø : NORMAL
1 : FRAME
2 : QUICK

■ Command type — Sequential command

■ Usage restrictions — The command is invalid in the following condition.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed
The FRAME is invalid in the following case
When measurement pattern is DATA, and data length is 127 or less
QUICK selection in the ALTERNATE measurement pattern

Query: The following case is invalid, and ERR (CR / LF) is output:
When the measurement patter is PRBS

■ Usage example

Program: When NORMAL synchronous mode is specified

```
OUTPUT△7ØØ;"SYM△Ø"
```

Query: When synchronous mode is specified as FRAME

```
OUTPUT△7ØØ;"SYM?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

SYM△1 (CR / LF) is output.

: When measurement pattern is specified as PRBS

```
OUTPUT△7ØØ;"SYM?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

ERR (CR / LF) is output.

29) DSP Display switching (DiSPlay select)

■ **Function**

Pattern part display switching is specified.

Header	Program	Query	Response (Number of characters)
DSP	DSP△m	DSP?	DSP△m (FIX 1)

■ **Value of m**

Ø : PATTERN
1 : BIT WINDOW
2 : BLOCK WINDOW
3 : ERROR ANALYSIS

■ **Command type**

Sequential command

■ **Usage restrictions**

The command is invalid in the following condition.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed
The BLOCK WINDOW is invalid in the following cases:
When the measurement pattern is PRBS
When the measurement pattern is DATA, and the bit length is not multiple of 32
When the synchronous mode is QUICK
The ERROR ANALYSIS is invalid in the following cases:
When no OPTION-01 is mounted
When the measurement pattern is ALTERNATE
When the synchronous mode is QUICK

Query: None

■ **Usage example**

Program: When display switching is specified as PATTERN

```
OUTPUT△7ØØ;"DSP△Ø"
```

Query: When display switching is specified as BIT WINDOW

```
OUTPUT△7ØØ;"DSP?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

DSP△1 (CR / LF) is output.

30) ALT

Alternate pattern A / B display switching
(ALTernate a / b)

■ Function

The alternate pattern time for A / B display switching is specified.

Header	Program	Query	Response (Number of characters)
ALT	ALT△m	ALT?	ALT△m (FIX 1)

■ Value of m

Ø : Pattern A
1 : Pattern B

■ Command type

Sequential command

■ Usage restrictions

The command is invalid in the following condition.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed
When the measurement patterns are DATA, ZERO SUBST, and PRBS

Query: The following case is invalid and ERR (CR / LF) is output.
When the measurement patterns are DATA, ZERO SUBST and PRBS

■ Usage example

Program: When the alternate display is switched to pattern A

```
OUTPUT△7ØØ;"ALT△Ø"
```

Query: When the alternate display is switched to pattern A

```
OUTPUT△7ØØ;"ALT?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

ALT△1 (CR / LF) is output.

: When the measurement pattern is specified as PRBS

```
OUTPUT△7ØØ;"ALT?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

ERR (CR / LF) is output.

31) FLN FRAME length (Frame LeNgth)

■ **Function** Frame length is specified.

Header	Program	Query	Response (Number of characters)
FLN	FLN△m	FLN?	FLN△m (FIX 2)

■ **Value of m** Frame bit length values are specified within the range 4 to 32 bits.

Max.: 32
Min.: 4
Step: 4

■ **Command type** Sequential command

■ **Usage restrictions** The command is invalid in the following condition.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed
When measurement pattern is PRBS
When a synchronous mode is NORMAL or QUICK
When a frame bit length is not in steps of 4

Query: In the following cases, the command is invalid and ERR (CR / LF) is output.
When the measurement pattern is PRBS
When the synchronous mode is NORMAL or QUICK

■ **Usage example**

Program: When the frame bit length is set to 4 bits

```
OUTPUT△700;"FLN△4"
```

Query: When the frame bit length is specified as 32 bits

```
OUTPUT△700;"FLN?"
ENTER△700;B$
PRINT△B$
```

↓

FLN△32 (CR / LF) is output.

: When the measurement pattern is specified as PRBS

```
OUTPUT△700;"FLN?"
ENTER△700;B$
PRINT△B$
```

↓

ERR (CR / LF) is output.

32) DLN — Measurement data length (Data LeNgth)

■ **Function**

Measurement data length is specified when the measurement pattern is ALTERNATE or DATA.

Header	Program	Query	Response (Number of characters)
DLN	DLN△m	DLN?	DLN△m (FIX 7)

■ **Value of m**

Data length is specified in the following range.

- Alternate pattern
  - Max.: 4194304
  - Min.: 128
  - Step: 128
- Data pattern
  - Max.: 8388608
  - Min.: 2
  - Step: The following are classified according to the data length.

Data length		Step
2 ~	65536 bits / step	1 bit
65536 ~	131072 bits / step	2 bits
131072 ~	262144 bits / step	4 bits
262144 ~	524288 bits / step	8 bits
524288 ~	1048576 bits / step	16 bits
1048576 ~	2097152 bits / step	32 bits
2097152 ~	4194304 bits / step	64 bits
4194304 ~	8388608 bits / step	128 bits

■ **Command type**

Overlap command

■ **Usage restrictions**

The command is invalid in the following conditions.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed
When the measurement patterns are ZERO SUBST or PRBS

Query: The following case is invalid, and ERR (CR / LF) is output
When a measurement pattern is ZERO SUBST or PRBS

■ Usage example

Program: When the data length is specified as 4 bits

OUTPUT△700;"DLN△4"

Query: When the data length was specified as 32 bits

OUTPUT△700;"DLN?"
ENTER△700;B$
PRINT△B$

↓

DLN△△△△△△32 (CR / LF) is output.

: When the measurement pattern was specified as PRBS

OUTPUT△700;"DLN?"
ENTER△700;B$
PRINT△B$

↓

ERR (CR / LF) is output.

■ Note

When no data value is specified, the value is optimized in the following way.

The input value is rounded down to the nearest value.				
Example)	● Data length input		● Data length to be specified	
	131075	→→→	131072	

33) ZLN

ZERO SUBST length (**Z**ero subst **L**e**N**gth)

■ Function

Zero substitution bit length is specified when the measurement pattern is ZERO SUBST.

Header	Program	Query	Response (Number of characters)
ZLN	ZLN△m	ZLN?	ZLN△m (FIX 5)

■ Value of m

Zero substitution bit length can be specified within the following range.

Maximum value: The range is decided according to the number of steps ZERO SUBST.

2^7 : 127
2^9 : 511
2^{11} : 2047
2^{15} : 32767

Minimum value: 1
Step: 1

■ Command type

Overlap command

■ Usage restrictions

The command is invalid in the following conditions.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed
When the measurement pattern is ALTERNATE, DATA, or PRBS
When synchronous mode is QUICK

Query: The following case is invalid and ERR (CR / LF) is output.
When the measurement pattern is ALTERNATE, DATA, or PRBS
When synchronous mode is QUICK

■ Usage example

Program: When ZERO SUBST length is specified as 1 bit

```
OUTPUT△700;"ZLN△1"
```

Query: When ZERO SUBST length has been specified as 127 bits

```
OUTPUT△700;"ZLN?"
ENTER△700;B$
PRINT△B$
↓
```

ZLN△△△127 (CR / LF) is output.

: When the measurement pattern was specified as PRBS

```
OUTPUT△700;"ZLN?"
ENTER△700;B$
PRINT△B$
↓
```

ERR (CR / LF) is output.

34) ADR PAG

Number of pages (ADdRess / PAGe)

■ Function

Number of pages is specified.

Header	Program	Query	Response (Number of characters)
ADR	ADR△m	ADR?	ADR△m (FIX 9)
PAG	PAG△m	PAG?	PAG△m (FIX 9)

■ Value of m

Number of pages is specified within the following range.
Maximum value: 1342177278
Minimum value: 1
Step: 1

■ Command type

Sequential command

■ Usage restrictions

The command is invalid in the following condition.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ Usage example

Program: When the a page number is specified as 1
OUTPUT△700;"PAG△1"

Query: When number of pages has been specified as 16000
OUTPUT△700;"PAG?"
ENTER△700;B$
PRINT△B$
↓
PAG△△△△△16000 (CR / LF) is output.

■ Note

There are two commands; "ADR" and "PAG", which have the same function.
However, the maximum settable page numbers differ according to the specified values of the measurement pattern and data length.

Also, when a settable maximum page number is specified within the above maximum value of 1342177278, then, this becomes the maximum value at that time.

Example) The data length is 32, number of pages displayed is 1, and maximum page numbers is 2, so if PAG△3 is input, the number of pages becomes 2.

The maximum page value is the value in which the data length is divided by 16, and if there is any remainder, 1 is added to the quotient.

35) BIT — Pattern bit (pattern BIT)

■ **Function** — Bit pattern is specified.

Header	Program	Query	Response (Number of characters)
BIT	• NR1 format BIT△m • HEX format BIT△#Hm	BIT?	Bit contents of 8 pages max. can be output in the following format until the maximum pattern specified pages is reached. PAG△*********: BIT△#H****,#H****,..., #H**** Bit pattern is indicated in HEX notation together with output head page. Maximum 8 pages (by FIX 4)

■ **Value of m** — Bit pattern is specified in the following range.

- NR1 format
  - Maximum value: 65535
  - Minimum value: 0
  - Step: 1
- HEX format
  - Maximum value: FFFF
  - Minimum value: 0
  - Step: 1

■ **Command type** — Overlap command

■ **Usage restrictions** — The command is invalid in the following condition.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessedWhen measurement pattern is ZERO SUBST, or PRBS
When synchronous mode is QUICK

Query: The command is invalid in the following case and ERR (CR / LF) is output.
When the synchronous mode is QUICK

■ Usage example

Program: When three pages of bit pattern are specified from the current specified page

```
OUTPUT△700;"BIT△10,20,30"
OUTPUT△700;"BIT△#HFFFF,#H1000,#H2000"
```

The pattern bit of a continuation page can be specified by separating the data and data space with a comma.

After specifying page numbers once, if a 4 page pattern bit is to be specified from that specified page

```
OUTPUT 700;"PAG△10;BIT△10,20,30,40"
OUTPUT 700;"PAG△10;BIT△#HFFFF,#H1000,
            #H2000,#H3000"
```

Query: When the display page is 1 and maximum settable pages are 10, the data from page 1 to 8 is read.

```
OUTPUT△700;"BIT?"
ENTER△700;B$
PRINT△B$
              ↓
PAG△△△△△△△△△1; BIT△#H0000,#H0000,
                #H0000,#H0000,#H0000,
                #H0000,#H0000,#H0000
```

(CR / LF) is output.

: When the synchronous mode is QUICK

```
OUTPUT△700;"BIT?"
ENTER△700;B$
PRINT△B$
              ↓
```

ERR (CR / LF) is output.

■ Note

Continuation page pattern bits can be specified by separating with comma (,) between data and data space of NR1 or HEX part. (8 pages max.)

By setting bit1 of the bit indicator to LSB, and bit16 to MSB, response is performed.
For example, if 32768 is specified, the upper bit (bit16) becomes 1.

36) CHM

BIT WINDOW patter bit
(bit window CH Mask pattern)

■ Function

BIT WINDOW pattern is specified.

Header	Program	Query	Response (Number of characters)
CHM	● NR1 format CHM△m ● HEX format CHM△#Hm	CHM?	Bit contents of 8 pages max.can be output in the following format until the maximum pattern specified pages is reached. $\underbrace{\texttt{MSK△*;CHM△\#H****,\#H****}}$ Bit pattern is indicated in HEX notation together with header page. Maximum 2 pages (by FIX 4)

■ Value of m

BIT WINDOW pattern is specified within the following range.

● NR1 format
Maximum value: 65535
Minimum value: 0
Step: 1

● HEX format
Maximum value: FFFF
Minimum value: 0
Step: 1

■ Command type

Sequential command

■ Usage restrictions

The command is invalid in the following conditions.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ Usage example

Program: When bit patters for 2 pages is specified while the current specified page is 1

```
OUTPUT△700;"CHM△10,20"
OUTPUT△700;"CHM△#HFFFF,#H1000"
```

The pattern bit of a continuation page can be specified by separating the data and data space with a comma.
After specifying page numbers once, if a pattern bit of the maximum numbers of pages is specified from the specified pages

```
OUTPUT 700;"MSK△1;CHM△10,20"
OUTPUT 700;"MSK△1;CHM△#HFFFF,#H1000"
```

Query: When the display page is specified as 1
Data of page 1 and page 2 are read.

```
OUTPUT△700;"CHM?"
ENTER△700;B$
PRINT△B$
```

↓

`MSK△1;CHM△#H0000,#H0000` (CR / LF) is output.

■ Note

Continuation page pattern bits can be specified by separating with commas (,) between data and data space of NR1 or HEX. (Maximum 2 pages)

37) MSK BIT WINDOW page **(bit window MaSK page)**

■ Function — Page numbers and pattern synchronous position are specified.

Header	Program	Query	Response (Number of characters)
MSK	MSK△m	MSK?	MSK△m (FIX 10)

■ Value of m — Page number are specified within the following range.

Maximum: 2
Minimum: 1
Step: 1

■ Command type — Sequential command

■ Usage restrictions — The command is invalid in the following condition.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ Usage example — Program: When page 1 is specified

```
OUTPUT△700;"MSK△1"
```

Query: When 2 pages have been specified

```
OUTPUT△700;"MSK?"
ENTER△700;B$
PRINT△B$
```

↓

MSK△2 (CR / LF) is output.

38) MGB

BLOCK WINDOW pattern bit
(block window Mask Gate pattern Bit)

■ Function

BLOCK WINDOW patter bit is specified.

Header	Program	Query	Response (Number of characters)
MGB	• NR1 format MGB△m • HEX format MGB△#Hm	MGB?	Bit contents of 8 pages max. can be output in the following format until the maximum pattern specified pages is reached. PAG△*********; MGB△#H****,#H****,..., #H**** Bit pattern is indicated in HEX notation together with header page. Maximum 8 pages (FIX 4 each)

■ Value of m

Bit pattern is specified in the following range.

• NR1 format		• HEX format	
Maximum:	65535	Maximum:	FFFF
Minimum:	0	Minimum:	0
Step:	1	Step:	1

※ When 1 to 655356 is specified, measurement range is MASKed in 32-bit units.

■ Command type

Overlap command

■ Usage restrictions

The command is invalid in the following condition.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed
When the measurement pattern is PRBS
When the synchronous format is QUICK
When the data length is not a multiple of 32

Query: The following cases are invalid and ERR (CR / LF) is output.
When the measurement pattern is PRBS
When the synchronous format is QUICK
When the data length is not a multiple of 32

■ Usage example

Program: When a bit pattern of 3 pages is specified from the current specified pages

```
OUTPUT△7ØØ;"MGB△ØØ,Ø1,ØØ"
OUTPUT△7ØØ;"MGB△#HFFFF,#H1ØØØ,#H2ØØØ"
```

Continuation page pattern bits can be specified by separating with commas (,) between data and data spaces.

After specifying page numbers once, if a pattern bit of 4 pages is specified from the specified pages

```
OUTPUT 7ØØ;"PAG△1Ø;MGB△1Ø,2Ø,3Ø,4Ø"
OUTPUT 7ØØ;"PAG△1Ø;MGB△#HFFFF,#H1ØØØ,
              #H2ØØØ,#H3ØØØ"
```

Query: When page is specified as 1
Data from page 1 to 8 is read.

```
OUTPUT△7ØØ;"MGB?"
ENTER△7ØØ;B$
PRINT△B$
             ↓
PAG△△△△△△△△△△1; MGB△#HFFFF,#HFFFF,
                #HFFFF,#HFFFF,#HØØØØ,
                #HØØØØ,#HØØØØ,#HØØØØ,
                #HØØØØ
```

(CR / LF) is output.

■ Note

Continuation page pattern bits can be specified by separating with commas (,) between data and data space of NR1 or HEX. (Maximum 8 pages)

The BLOCK WINDOW pattern is specified in units of 32 bits.
Therefore, data of 32 bits such as page 1 and page 2, page 3 and page 4 is specified with ALL'0s' or ALL'1s'.
For example, 1 to 65535 is specified, 32 bits values are specified by ALL '1' and the measurement range is MASKed.

For ALTERNATE pattern, the A or B BLOCK WINDOW pattern must be specified by using the A/B display switch (No. 30).

39) EAB? Error analysis data (Error Analysis pattern Bit?)

■ **Function** Error analysis data results can be read. (Effective only for Option-01)

Header	Program	Query	Response (Number of characters)
None	None	EAB?	The result of analysis of the specified page numbers is output in the following format. EAP△**;EAB△*********, #B**************** Page (FIX 2) Monitor data (page value) (FIX 9) Error analysis data (FIX 16) 0 or 1 : No error + : Inserted error _ : Erased error

■ **Value of m**

Page: Display switching from 1 to 16

Monitor data: Page position from the header of the pattern specified

Error analysis data: Error analysis result

0 or 1: No error

+ sign: Inserted error (Changed error from 0 to 1)

− sign: Sending error (Changed error from 1 to 0)

■ **Command type** Sequential command

■ **Usage restrictions** The command is invalid in the following conditions.

Query: The following cases is invalid and ERR (CF / LF) is output.
When the synchronous mode is QUICK
When there is no measurement data
When the measurement data is invalid
When the measurement pattern is ATTERNATE

■ **Usage example**

Query: When the error analysis result is as shown below with display page 10, and 218th monitor page

bit 1 bit 16
Ø11Ø11Ø1ØØØØ1ØØ1
Error bit

```
OUTPUT△7ØØ;"EAB?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

EAP△1Ø;EAB△△△△△△△218,
#B1ØØ1ØØØØ1Ø11Ø+1Ø (CR / LF) is output.

bit 16 bit 1

■ Note

Error analysis results are fetched in units of 16 bits. Page setting is not performed automatically. Therefore, when each data page is fetched, the required error analysis page must be specified if necessary.
(Pages are specified by the EAP command on the next page)
The error analysis data cannot backed up in the built-in memory.
Keep this in mind.

40) EAP

Error analysis page **(Error Analysis Page)**

■ Function

Error analysis page is switched. (Effective only for option-01)

Header	Program	Query	Response (Number of characters)
EAP	EAP△m	EAP?	EAP△m (FIX 2)

■ Value of m

The error analysis result is displayed on the BIT indicator in units of 16 bits.

Range of numeric values: Maximum 16
Minimum 1
Step 1

■ Command type

Sequential command

■ Usage restrictions

The command is invalid in the following condition.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed
When no error analysis OPTION-01 is mounted
When the synchronous mode is QUICK

Query: The following case is invalid and ERR (CR / LF) is output.
When the synchronous mode is QUICK
The following case becomes invalid.
When no error analysis OPTION-01 is mounted

■ Usage example

Program: When error analysis page is specified as 5

```
OUTPUT△700;"EAP△5"
```

Query: When error analysis page has been specified as 1

```
OUTPUT△700;"EAP?"
ENTER△700;B$
PRINT△B$
```

↓

EAP△△1 (CR / LF) is output.

41) MSE BIT WINDOW ON / OFF **(bit window ch MaSk Enable)**

■ Function BIT WINDOW ON / OFF is controlled.

Header	Program	Query	Response (Number of characters)
MSE	MSE△m	MSE?	MSE△m (FIX 1)

■ Value of m BIT WINDOW ON / OFF is controlled.

Ø : OFF

1 : ON

■ Command type Sequential command

■ Usage restrictions The command is invalid in the following condition.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ Usage example Program: When the BIT WINDOW is specified as ON

```
OUTPUT△7ØØ;"MSE△1"
```

Query: When the BIT WINDOW has been specified as OFF

```
OUTPUT△7ØØ;"MSE?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

MSE△Ø (CR / LF) is output.

42) MGE

BLOCK WINDOW ON / OFF
(block window Meas. Gate Enable)

■ Function

BLOCK WINDOW ON / OFF is controlled.

Header	Program	Query	Response (Number of characters)
MGE	MGE△m	MGE?	MGE△m (FIX 1)

■ Value of m

BLOCK WINDOW ON / OFF is controlled.
Ø : OFF
1 : ON

■ Command type

Sequential command

■ Usage restrictions

The command is invalid in the following condition.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed
When measurement pattern is PRBS
When the measurement pattern is ALTERNATE or DATA, and the data length is not a multiple of 32
When the synchronous mode is QUICK

Query: The following cases is invalid, and ERR (CR / LF) is output.
When the measurement pattern is PRBS
When the measurement pattern is ALTN or DATA, and the data length is not multiple of 32
When the synchronous mode is QUICK

■ Usage example

Program: When the BLOCK WINDOW is specified as ON
```
OUTPUT△700;"MGE△1"
```

Query: When the BLOCK WINDOW has been specified as OFF
```
OUTPUT△700;"MGE?"
ENTER△700;B$
PRINT△B$
```
↓
MGE△Ø (CR / LF) is output.

43) EAT

Error analysis trigger (**Error Analysis Trigger**)

■ Function

Error analysis trigger is specified. (Effective only when option-01 is mounted)

Header	Program	Query	Response (Number of characters)
EAT	EAT△m	EAT?	EAT△m (FIX 1)

■ Value of m

Error trigger condition is specified.

Ø : OFF (Forced termination)

1 : Programing time
  START
 Query time
  AWAITING (trigger wait status)

2 : Programming time
  Invalid
 Query time
  TRIGGERED (triggered status)

■ Command type

Sequential command

■ Usage restrictions

The command is invalid in the following condition.

Program: During automatic phase threshold search operation
 While Eye margin measurement is being executed
 When a floppy disk is being accessed
 When the synchronous mode is QUICK
 When OPTION-01 is not mounted
 When the measurement pattern is ALTERNATE

Query: The following case is invalid, and ERR (CR / LF) is output.
  When the synchronous mode is QUICK
 The following case is invalid.
  OPTION-01 is not mounted.

■ Usage example

Program: When the error analysis trigger is specified as START

```
OUTPUT△7ØØ;"EAT△1"
```

Query: When the error analysis has been triggered

```
OUTPUT△7ØØ;"EAT?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

EAT△2 (CR / LF) is output.

44) WRT

Number of pattern data input bytes
(pattern data WRiTe)

■ Function

Number of bytes of pattern data to be DMA transferred and start address of the pattern data.

Header	Program	Query	Response (Number of characters)
WRT	WRT△m1,m2	None	None

■ Value of m

m1 : Number of pattern transfer bytes
Range of numeric values: Maximum 1048376
Minimum 1
Step 1

m2 : Pattern input head address
Range of numeric values: Maximum 524287
Minimum 0
Step 1

■ Command type

Sequential command

■ Usage restrictions

The command is invalid in the following condition.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed
When the synchronous mode is QUICK

■ Usage example

Program: When the measurement pattern is DATA, and data from pages 1 to 10 is specified

```
DIM△B(9)
READ△B(*)
DATA△1,2,4,8,16,32,64,128,256,512
OUTPUT△700;"WRT△20,0"
OUTPUT△700△USING△"W";B(*)
```

Data from page 1 to 10 is specified.

■ Note

This equipment specify necessary byte numbers and input header address of pattern data to transfer them in the DMA mode, and also specifies DMA switching and internal RAM area storage.

The relationship between the pattern header address and actual pages becomes as below:

(Pattern header address + 1) = Actual page numbers

When pattern data sending is complete, the DMDA mode is cleared. For DMA transfer of pattern data, see appendix "Pattern Data DMA Sending".

The settable number of the bytes is decided by the data length and header address. However, when the data over the settable range are transferred, the data become invalid.

45) RED?

Number of bytes of pattern data output
(pattern data REaD?)

■ Function

Number of bytes and start address of the DMA transferred pattern data are specified.

Header	Program	Query	Response (Number of characters)
RED	None	RED?△m1,m2	Data pattern string (by m1)

■ Value of m

m1 : Number of pattern sending bytes
Range of numeric values: Maximum 1048376
Minimum 1
Step 1

m2 : Pattern output head address
Range of numeric value: Maximum 524287
Minimum 0
Step 1

When the measurement pattern is ALTERNATE, the maximum value becomes half of the above maximum value.

■ Command type

Sequential command

■ Usage restrictions

The command is invalid in the following condition and ERR (CR / LF) is output.

Query: When the synchronous mode is QUICK
Measurement patterns is ZERO SUBST or PRBS

■ Usage example

Query: When the measurement pattern is DATA, and data from page 1 to 10 is specified

```
DIM△B(9)
OUTPUT△700;"RED?△20,0"
ENTER△700△USING△"W";B(*)
PRINT△B(*)
```

Data from page 1 to 10 is printed.

■ Note

This equipment specify necessary byte numbers and input header address of pattern data to transfer them in the DMA mode, and also specifies DMA switching and internal RAM area storage.

The relationship between the pattern header address and actual pages becomes as below:
(Pattern header address + 1) = Actual page numbers

When pattern data sending is complete, the DMA mode is cleared.
For DMA transfer of pattern data, see appendix "Pattern Data DMA Sending".

The number of the bytes to be output is decided by the data length and header address. However, when a query over the valid number of the data is performed, only the valid number of the bytes is output.

46) MWT

Number of BLOCK WINDOW data input bytes
(block window Meas. gate pattern data WriTe)

■ Function

Number of bytes to be DMA transferred and the start address of the BLOCK WINDOW pattern data are specified

Header	Program	Query	Response (Number of characters)
MWT	MWT△m1,m2	None	None

■ Value of m

m1 : Number of sending bytes of the BLOCK WINDOW pattern (32-page data)
Range of numeric values: Maximum 32768
Minimum 1
Step 1

m2 : Input header address of the BLOCK WINDOW PATTERN
Range of numeric values: Maximum 16383
Minimum 0
Step 1

When the measurement pattern is ALTERNATE, the maximum value becomes half of the above maximum value.

■ Command type

Sequential command

■ Usage restrictions

The command is invalid in the following condition.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed
When the synchronous mode is QUICK

■ Usage example

Program: When the measurement pattern is DATA and data from page 1 to 32 is specified

```
DIM△B(Ø)
READ△B(*)
DATA△1
OUTPUT△7ØØ;"MWT△2,Ø"
OUTPUT△7ØØ△USING△"W";B(*)
```

Data from page 1 to 32 is specified. (Page 17 and 18 are masked)

Note

This equipment specify necessary byte numbers and input header address of pattern data to transfer them in the DMA mode, and also specifies DMA switching and internal RAM area storage.

The relationship between the pattern head address and actual pages becomes as below:

(Pattern head address $\times$ 32 + 1) = Actual page numbers

When pattern data sending is complete, the DMDA mode is cleared. For DMA transfer of pattern data, see Appendix "Pattern Data DMA Sending".

Since the BLOCK WINDOW data is in units of 32 bits, when the DMA transfer is carried out, 32 bits are handled as 1 bit.
When the setting value of each bit is 0, this bit becomes the object block for measurement, and when 1, then this is masked.

The significance of the setting values is as follows:

Setting value	BLOCK WINDOW setting page (For head address is 0)
1	17, 18
2	19, 20
4	21, 22
8	23, 24
16	25, 26
32	27, 28
64	29, 30
128	31, 32
256	1, 2
512	3, 4
1024	5, 6
2048	7, 8
4096	9, 10
8192	11, 12
16384	13, 14
32768	15, 16

If the header address is assumed to be N, the page setting becomes the above page value + (32 $\times$ N).

For ALTERNATE pattern, pattern A or B can be obtained using the ALTERNATE switching.

The settable number of the bytes is decided by the data length and header address. However, when the data over the settable range are transferred, the data become invalid.

47) MRD?

Number of output bytes of BLOCK WINDOW data
(block window Meas. gate pattern data ReaD?)

■ Function

Number of bytes and start address of the DMA transferred BLOCK WINDOW pattern data can be specified.

Header	Program	Query	Response (Number of characters)
MRD	None	MRD?△m1,m2	Data pattern string (by m1)

■ Value of m

m1 : Number of transfer bytes of BLOCK WINDOW pattern
Range of numeric values: Maximum 32768
Minimum 1
Step 1

m2 : Head address of the BLOCK WINDOW pattern output
Range of numeric value: Maximum 16383
Minimum 0
Step 1

When the measurement pattern is ALTERNATE, the maximum value becomes half of the above maximum value.

■ Command type

Sequential command

■ Usage restrictions

The command is invalid in the following conditions.

Query: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed
When the synchronous mode is QUICK
When the measurement pattern is PRBS
When the measurement pattern is ALTERNATE or DATA, and the datalength is not a multiple of 32

■ Usage example

Query: When the measurement pattern is DATA and the data from page 1 to 32 is read

```
DIM△B(Ø)
OUTPUT△7ØØ;"MRD?△2,Ø"
ENTER△7ØØ△USING△"W";B(*)
PRINT△B(*)
```

Data from page 1 to 32 is printed.

Note

This equipment specify necessary byte numbers and input header address of pattern data to transfer them in the DMA mode, and also specifies DMA switching and internal RAM area storage.

The relationship between the pattern header address and actual pages becomes as below:

(Pattern header address $\times$ 32 + 1) = Actual page numbers

When pattern data sending is complete, the DMDA mode is cleared. For DMA transfer of pattern data, see Appendix "Pattern Data DMA Sending".

Since the BLOCK WINDOW data is in units of 32 bits, when the DMA transfer is carried out, 32 bits are handled as 1 bit.

When the setting value of each bit is 0, this bit becomes the object block for measurement, and when 1, then this is masked.

Significance of the setting values is as follows:

Setting value	BLOCK WINDOW setting page (For head address is 0)
1	17, 18
2	19, 20
4	21, 22
8	23, 24
16	25, 26
32	27, 28
64	29, 30
128	31, 32
256	1, 2
512	3, 4
1024	5, 6
2048	7, 8
4096	9, 10
8192	11, 12
16384	13, 14
32768	15, 16

When the header address is assumed to be N, the page setting becomes the above page value + (32 $\times$ N).

For ALTERNATE pattern, pattern A or B can be obtained using the ALTERNATE switching.

The number of the bytes to be output are decided by the data length and header address. However, when a query over the valid number of the data is performed, only the valid number of the bytes is output.

48) ALL — Pattern data preset (All pages, all bits) **(preset ALL)**

■ **Function**

All pages and all bits of the pattern data are set to 0 or 1.

Header	Program	Query	Response (Number of characters)
ALL	ALL△m1	None	None

■ **Value of m**

m1 :

- 0 : All pages clear
- 1 : All page set

■ **Command type**

Overlap command

■ **Usage restrictions**

The command is invalid in the following condition.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed
When the synchronous mode is QUICK
When the measurement pattern is ZERO SUBST or PRBS

■ **Usage example**

Program: When the measurement pattern is DATA, and all pages are cleared

```
OUTPUT△700;"ALL△0"
```

The data from all pages are cleared.

■ **Note**

For ALTERNATE pattern, pattern A or B can be preset according to the A / B display switching condition (No. 30).

For example, if this command is executed when pattern A is displayed, then only pattern A is preset.

49) PST Pattern data preset (1 page, all bits) (PreSeT)

■ **Function**

All bits of 1 pattern data page are specified to 0 or 1.

Header	Program	Query	Response (Number of characters)
PST	PST△m1	None	None

■ **Value of m**

m1 :
1 : 1 page clear
0 : 1 page set

■ **Command type**

Overlap command

■ **Usage restrictions**

The command is invalid in the following condition.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed
When the synchronous mode is QUICKWhen the measurement pattern is ZERO SUBST or PRBS

■ **Usage example**

Program: When the measurement pattern is DATA and 1 page is cleared
OUTPUT△7ØØ;"PST△Ø"

Data from 1 page is cleared.

■ **Note**

For ALTERNATE pattern, pattern A or B can be preset according to the A / B display switching condition (No. 30).

For example, if this command is executed when pattern A is displayed, then only pattern B is preset.

50) MAL

BLOCK WINDOW data preset (All pages, all bits)
(block window Meas. gate pattern preset ALl)

■ Function

All bits of all pages of the BLOCK WINDOW pattern data are specified to 0 or 1.

Header	Program	Query	Response (Number of characters)
MAL	MAL△m1	None	None

■ Value of m

m1 :
0 : All pages clear
1 : All pages set

■ Command type

Overlap command

■ Usage restrictions

The command is invalid in the following condition.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed
When the synchronous mode is QUICK
When the measurement pattern is ALTERNATE or DATA, and the data length is not a multiple of 32

■ Usage example

Program: When the measurement pattern is DATA, and the data of all pages of the BLOCK WINDOW is cleared

```
OUTPUT△7ØØ;"MAL△Ø"
```

All the page data is cleared.

■ Note

For ALTERNATE pattern, pattern A or B can be preset according to the A / B display switching condition (No. 30).

For example, if this command is executed when pattern A is displayed, then only pattern A is preset.

51) MPS

BLOCK WINDOW data preset (1 page, all bits)
(block window Meas. gate pattern PreSet)

■ **Function**

All bits of 1 page of the BLOCK WINDOW pattern data are specified to 0 or 1.

Header	Program	Query	Response (Number of characters)
MPS	MPS△m1	None	None

■ **Value of m**

m1 :
- 0 : 1 page clear
- 1 : 1 page set

■ **Command type**

Overlap command

■ **Usage restrictions**

The command is invalid in the following condition.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed
When the synchronous mode is QUICK
When measurement pattern is ALTERNATE or DATA, and the data length is not a multiple of 32

■ **Usage example**

Program: When the measurement pattern is DATA and data of 1 page of the BLOCK WINDOW is cleared

```
OUTPUT△7ØØ;"MPS△Ø"
```

Data of 1 page is cleared.

■ **Note**

For ALTERNATE pattern, pattern A or B can be preset according to the A / B display switching condition (No. 30).

For example, if this command is executed when pattern A is displayed, then only pattern A is preset.

52) HAL

BIT WINDOW data preset (All pages, all bits)
(bit window cH mask pattern preset ALl)

■ Function

All bits of all pages of BIT WINDOW pattern data are specified to 0 or 1.

Header	Program	Query	Response (Number of characters)
HAL	HAL△m1	None	None

■ Value of m

m1 :
- 0 : All pages clear
- 1 : All pages set

■ Command type

Overlap command

■ Usage restrictions

The command is invalid in the following condition.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

■ Usage example

Program: When the measurement data is DATA and the data from all pages of the BIT WINDOW data is cleared

```
OUTPUT△700;"HAL△0"
```

The data from all pages is cleared.

53) HPS

BIT WINDOW data preset (1 page, all bits)
(bit window cH mask pattern PreSet)

■ Function

All bits on 1 page of BIT WINDOW PATTERN data are set to 0 or 1.

Header	Program	Query	Response (Number of characters)
HPS	HPS△m1	None	None

■ Value of m

m1 :
- 0 : 1 page clear
- 1 : 1 page set

■ Command type

Overlap command

■ Usage restrictions

The command is invalid in the following condition.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

■ Usage example

Program: When the measurement pattern is DATA and data from 1 page of the BIT WINDOW is cleared

```
OUTPUT△700;"HPS△0"
```

The data from 1 page is cleared.

54) PSP

Pattern sync trigger position (**P**attern **S**ync **P**osition)

■ Function

When a pattern sync trigger is Variable, the trigger position is set.

Header	Program	Query	Response (Number of characters)
PSP	PSP△m	PSP?	PSP△m (FIX 9)

■ Value of m

Pattern sync position is set in the range below. State of not lock off

Maximum value: 1342177278
Minimum value: 1
Step: 1

■ Command type

Sequential command

■ Usage restrictions

The command is invalid in the following condition.

Program: During automatic threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ Usage example

Program: When the pattern sync trigger position is set to the first page

```
OUTPUT△7ØØ;"PSP△1"
```

Query: When the pattern sync trigger position is set to the 16000th page

```
OUTPUT△7ØØ;"PSP?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

Outputs PSP△△△△△16ØØØ (CR / LF).

■ Note

The maximum page number of the settable pattern sync trigger position depends on the set measurement pattern or data length setting value.
When a page number over the settable maximum page number in the range of less than the maximum value of the above m 1342177278 is input, the input page number is changed to the then maximum page number.

Example) If PSP△3 is input when a data length = 32, trigger position = 1, and trigger position maximum number = 2, the trigger position becomes 2.

The maximum value of the trigger position is the quotient of data length ÷ 16 when the remainder does not exist, and is the quotient + 1 when the remainder exists.

55) PPD

Page pattern sync trigger position display switch
(Page / Pattern sync positon Display)

■ **Function**

Display contents of the 7 segment display are switched to a page and pattern sync trigger position.

Header	Program	Query	Response (Number of characters)
PPD	PPD△m	PPD?	PPD△m (FIX 1)

■ **Value of m**

Ø : Page number
1 : Pattern sync trigger position

■ **Command type**

Sequential command

■ **Usage restrictions**

The command is invalid in the following condition.

Program: During automatic threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ **Usage example**

Program: When the page number is displayed

```
OUTPUT△7ØØ;"PPD△Ø"
```

Query: When the pattern sync trigger position is displayed

```
OUTPUT△7ØØ;"PPD?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

Outputs PPD△1 (CR / LF).

● MEASUREMENT section

Each control message of the MEASUREMENT section is explained on the following pages.

The triangle marks (△) indicates a space.

56) CLI? Clock loss status (Clock Loss Intervals?)

■ **Function** Clock input status is read.

Header	Program	Query	Response (Number of characters)
CLI	None	CLI?	CLI△m (FIX 1)

■ **Value of m**

Ø : Not clock loss status
1 : Clock loss status

■ **Command type** Sequential command

■ **Usage restrictions** The command is invalid in the following condition.

Query: None

■ **Usage example** Query: When status is not clock loss

```
OUTPUT△7ØØ;"CLI?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

CLI△Ø (CR / LF) is output.

57) SLI? Sync loss state (Sync Loss Intervals?)

■ Function — Sync loss status can be read.

Header	Program	Query	Response (Number of characters)
SLI	None	SLI?	SLI△m (FIX 1)

■ Value of m

Ø : Not sync loss status
1 : Sync loss status

■ Command type — Sequential command

■ Usage restrictions — The command is invalid in the following condition.

Query: None

■ Usage example

Query: When status is not sync loss

```
OUTPUT△7ØØ;"SLI?"
ENTER△7ØØ;B$
PRINT△B$
        ↓
SLI△Ø (CR/LF) is output.
```

58) ERS? Error detection status (ERrorS?)

■ Function — Error detection status is read.

Header	Program	Query	Response (Number of characters)
ERS	None	ERS?	ERS△m (FIX 1)

■ Value of m
Ø : Not error detected status
1 : Error detected status

■ Command type — Sequential command

■ Usage restrictions — The command is invalid in the following condition.
Query: None

■ Usage example — Query: When status is not error detected status

```
OUTPUT△7ØØ;"ERS?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

ERS△Ø (CR / LF) is output.

59) DMS Measurement display mode (**Display or MeaSurement**)

■ **Function** Data to be displayed on the measurement display is selected.

Header	Program	Query	Response (Number of characters)
DMS	DMS△m	DMS?	DMS△m (FIX 1)

■ **Value of m**

Ø : ERROR RATIO
1 : ERROR COUNT
2 : ERROR INTERVAL
3 : ERROR FREE INTERVAL
4 : CLOCK FREQUENCY

■ **Command type** Sequential command

■ **Usage restrictions** The command is invalid in the following condition.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ **Usage example**

Program: When the display mode is specified as ERROR RATIO

```
OUTPUT△7ØØ;"DMS△Ø"
```

Query: When the display mode has been specified as ERROR COUNT

```
OUTPUT△7ØØ;"DMS?"
ENTER△7ØØ;B$
PRINT△B$
      ↓
```

DMS△1 (CR / LF) is output.

60) CUR

Intermediate measurement result display function
(CURrent data)

■ Function

Functions for displaying intermediate measurement results are controlled.

Header	Program	Query	Response (Number of characters)
CUR	CUR△m	CUR?	CUR△m (FIX 1)

■ Value of m

Ø : OFF
1 : ON

■ Command type

Sequential command

■ Usage restrictions

The command is invalid in the following condition.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ Usage example

Program: When the intermediate measurement result display is specified as OFF

```
OUTPUT△7ØØ;"CUR△Ø"
```

Query: When the measurement intermediate result display has been specified as ON

```
OUTPUT△7ØØ;"CUR?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

CUR△1 (CR / LF) is output.

61) MOD Measurement mode (measurement MODe)

■ Function

Measurement mode is specified.

Header	Program	Query	Response (Number of characters)
MOD	MOD△m	MOD?	MOD△m (FIX 1)

■ Value of m

Ø : REPEAT
1 : SINGLE
2 : UNTIMED

■ Command type

Sequential command

■ Usage restrictions

The command is invalid in the following conditions.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ Usage example

Program: When the measurement mode is specified as REPEAT

```
OUTPUT△7ØØ;"MOD△Ø"
```

Query: When the measurement mode has been specified as SINGLE

```
OUTPUT△7ØØ;"MOD?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

MOD△1 (CR / LF) is output.

62) STA Measurement start (STArt)

■ Function

Measurement start is specified.
During operation, this becomes the re-start specification.

Header	Program	Query	Response (Number of characters)
STA	STA	None	None

■ Command type

Sequential command

■ Usage restrictions

The command is invalid in the following condition.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

■ Usage example

Program: When measurement start or restart is specified
```
OUTPUT△7ØØ;"STA"
```

63) STO Measurement end (STOp)

■ **Function** Measurement termination or stop is specified.

Header	Program	Query	Response (Number of characters)
STO	STO	None	None

■ **Command type** Sequential command

■ **Usage restrictions** The command is invalid in the following condition.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

■ **Usage example** Program: When measurement termination or stop is specified

```
OUTPUT△700;"STO"
```

64) MSR?

Measurement status
(MeaSuRement in progress or stop?)

■ **Function**

Measurement status (during measurement or measurement stop) can be read.

Header	Program	Query	Response (Number of characters)
MSR	None	MSR?	MSR△m (FIX 1)

■ **Value of m**

Ø : Measurement stopped
1 : During measurement

■ **Command type**

Sequential command

■ **Usage restrictions**

The command is invalid in the following condition.

Query: None

■ **Usage example**

Query: When measurement is stopped

```
OUTPUT△700;"MSR?"
ENTER△700;B$
PRINT△B$
```

↓

MSR△Ø (CR / LF) is output.

65) SYN — Automatic synchronization (auto SYNc)

■ Function — Automatic synchronous function is controlled.

Header	Program	Query	Response (Number of characters)
SYN	SYN△m	SYN?	SYN△m (FIX 1)

■ Value of m

Ø : OFF
1 : ON

■ Command type

Sequential command

■ Usage restrictions

The command is invalid in the following condition.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ Usage example

Program: When the automatic synchronous function is specified as OFF

```
OUTPUT△7ØØ;"SYN△Ø"
```

Query: When the automatic synchronous function has been specified as ON

```
OUTPUT△7ØØ;"SYN?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

SYN△1(CR / LF) is output.

66) SYE

Automatic synchronous threshold
(auto SYnc thrEshold)

■ Function

Automatic synchronous threshold is specified.

Header	Program	Query	Response (Number of characters)
SYE	SYE△m	SYE?	SYE△m (FIX 1)

■ Value of m

Ø : 1E-2
1 : 1E-3
2 : 1E-4
3 : 1E-5
4 : 1E-6
5 : 1E-7
6 : 1E-8
8 : INT

■ Command type

Sequential command

■ Usage restrictions

The command is invalid in the following condition.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ Usage example

Program: When the automatic synchronous threshold is specified as 1E-2

```
OUTPUT△7ØØ;"SYE△Ø"
```

Query: When the automatic synchronous threshold has been specified as 1E-3

```
OUTPUT△7ØØ;"SYE?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

SYE△1 (CR / LF) is output.

■ Note

For the setting values of the automatic synchronous threshold, refer to the separate "Instruction Manual" booklet.

67) TIM

Real time / measurement time display switching
(real TIMe measurement time)

■ **Function**

Display contents of the time display is switched.

Header	Program	Query	Response (Number of characters)
TIM	TIM△m	TIM?	TIM△m (FIX 1)

■ **Value of m**

Ø : Y. M. D
1 : H. M. S
2 : PERIOD
3 : TIMED
4 : ELAPSED

■ **Command type**

Sequential command

■ **Usage restrictions**

The command is invalid in the following condition.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ **Usage example**

Program: When the display items are specified to Y. M. D (year. month. day)

```
OUTPUT△7ØØ;"TIM△Ø"
```

Query: When the display items has been specified as H. M. S (hour. minute. second)

```
OUTPUT△7ØØ;"TIM?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

TIM△1 (CR / LF) is output.

68) RTM   Built-in timer setting (Real TiMe setting)

■ Function   Built-in timer is set.

Header	Program	Query	Response (Number of characters)
RTM	RTM△m1,m2,m3,m4,m5,m6	RTM?	RTM△m1,m2,m3,m4,m5,m6 (each FIX 2)

■ Value of m

m1 : Year 0 - 99
m2 : Month 1 - 12
m3 : Day 1 - 31 (a maximum value of the day varies according to the month.)
m4 : Hour 0 - 23
m5 : Minute 0 - 59
m6 : Second 0 - 59

■ Command type   Sequential command

■ Usage restrictions   The command is invalid in the following condition.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ Usage example

Program: When the built-in timer is set to 59 seconds, 59 minutes, 13 hours, 27 days, January, and 1995 years

```
OUTPUT△700;"RTM△95,01,27,13,59,59"
```

Query: When the built-in timer has been set to 0 seconds, 0 minutes, 14 hours, 27 days, January, and 1995 years

```
OUTPUT△700;"RTM?"
ENTER△700;B$
PRINT△B$
```

↓

RTM△95,01,27,14,00,00 (CR / LF) is output.

■ Note   The NR1 part of m1 to m6 set in the program cannot be omitted.

69) PRD Measurement Period Setting (measurement PeRioD)

■ **Function** Measurement period is specified.

Header	Program	Query	Response (Number of characters)
PRD	PRD△m1,m2,m3,m4	PRD?	PRD△m1,m2,m3,m4 (each FIX 2)

■ **Value of m**

m1 : Day 0 - 99
m2 : Hour 0 - 23
m3 : Minute 0 - 59
m4 : Second 0 - 59

Here, all zeros cannot specified.

■ **Command type** Sequential command

■ **Usage restrictions** The command is invalid in the following condition.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed
When the measurement mode is UNTIME

Query: None

■ **Usage example**

Program: When the measurement period is specified as 99 days, 13 hours, 59 minutes, and 59 seconds

```
OUTPUT△700;"PRD△99,13,59,59"
```

Query: When the measurement period has been specified to 99 days, 14 hours, 0 minute, 0 second

```
OUTPUT△700;"PRD?"
ENTER△700;B$
PRINT△B$
```

↓

PRD△99,14,00,00 (CR/LF) is output.

■ **Note** NR1 values such as m1 to m4 set in the program cannot be omitted.

70) ER? Error ratio measurement result (ERror ratio?)

■ **Function**

Error ratio measurement result is output according to the output format.

However, the value of the error measurement ratio output here is the one displayed on the 7-segment display unit (End data or current data).

Header	Program	Query	Response (Number of characters)
ER	None	ER?	ER△△*.****E-** (FIX 10) ER△△*.****E-* (FIX 9) * denotes error ratio, 4th decimal place with exponent. Exponent is 2 digits or 1 digit.

■ **Command type** Sequential command

■ **Usage restrictions** The command is invalid in the following condition.

Query: When the error display unit displays '–', the following value is output.
ER△△Ø.ØØØØE-ØØ

■ **Usage example**

Query: When the error ratio measurement result is 1.05×10^{-6}

```
OUTPUT△7ØØ;"ER?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

ER△△1.Ø5ØØE-6 (CR / LF) is output.

: When the error ratio measurement result is 1.05×10^{-10}

```
OUTPUT△7ØØ;"ER?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

ER△△1.Ø5ØØE-1Ø (CR / LF) is output.

: When the error ratio measurement result is "–"

```
OUTPUT△7ØØ;"ER?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

ER△△Ø.ØØØØE-ØØ (CR / LF) is output.

71) EC? Error count measurement result (Error Count?)

■ Function

Error ratio measurement result is output according to the output format.

However, the number output here is the one displayed on the 7-segment display unit (End data or current data).

Header	Program	Query	Response (Number of characters)
EC	None	EC?	•When the number of errors are less than 1E+8 (FIX 10) EC△△******** * denotes error numbers with 8 digits fixed length. •When the number of errors 1E+8 or more EC△△*.****E** * denotes error numbers and 4th decimal place with fixed length exponent.

■ Command type

Sequential command

■ Usage restrictions

The command is invalid in the following condition.

Query: When the error display unit displays '−', the following value is output.
EC△△1.ØØØØE-99 (11 characters)

■ Usage example

Query: When the error numbers measurement result is 1.05×10^6

```
OUTPUT△7ØØ;"EC?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

EC△△△△1Ø5ØØØØ (CR / LF) is output.

: When the error numbers measurement result is 1.05×10^9

```
OUTPUT△7ØØ;"EC?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

EC△△1.Ø5ØØEØ9 (CR / LF) is output.

: When the error numbers measurement result is "−"

```
OUTPUT△7ØØ;"EC?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

EC△△1.ØØØØE-99 (CR / LF) is output.

72) EI? EI measurement result (Error Interval?)

■ **Function**

Error interval numbers measurement result is output according to the output format.

However, the error number output here is the one displayed on the 7-segment display unit (End data or current data).

Header	Program	Query	Response (Number of characters)
EI	None	EI?	•When the number of errors are less than 1E + 8 (FIX 10) EI△△******** * denotes number of EIs with 8 digits fixed length. •When the number of EIs is 1E + 8 or more EI△△*.****E** * denotes the number of EIs and 4th decimal place with fixed length exponent .

■ **Command type** Sequential command

■ **Usage restrictions** The command is invalid in the following condition.

Query: When the error display unit displays '–', the following value is output.
EI△△1.0000E-99 (11 characters)

■ **Usage example**

Query: When the EI numbers measurement result is 1.05×10^6

```
OUTPUT△700;"EI?"
ENTER△700;B$
PRINT△B$
```

↓

EI△△△△1050000 (CR / LF) is output.

: When the EI numbers measurement result is 1.05×10^9

```
OUTPUT△700;"EI?"
ENTER△700;B$
PRINT△B$
```

↓

EI△△1.0500E09 (CR / LF) is output.

: When the error measurement result is "–"

```
OUTPUT△700;"EI?"
ENTER△700;B$
PRINT△B$
```

↓

EI△△1.0000E-99 (CR / LF) is output.

73) EFI?

Error free interval ratio measurement results
(Error Free Interval?)

■ Function

Error free interval ratio measurement result is output according to the output format.

However, the error number output here is the one displayed on the 7-segment display unit (End data or current data).

Header	Program	Query	Response (Number of characters)
EFI	None	EFI?	EFI△△***.**** (FIX 10) * denotes %EFI, and 4th decimal place with fixed length.

■ Command type

Sequential command

■ Usage restrictions

The command is invalid in the following condition.

Query: When the error display unit displays '—', the following value is output.
EFI△△999.9999 (CR / LF)

■ Usage example

Query: When the EFI measurement result is 99.01%

```
OUTPUT△7ØØ;"EFI?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

EFI△△△△99.Ø1ØØ (CR / LF) is output.

: When the EFI measurement result is '—'

```
OUTPUT△7ØØ;"EFI?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

EFI△△△999.9999 (CR / LF) is output.

74) FRQ?

Clock frequency measurement result
(clock FReQuency?)

■ **Function**

Clock frequency measurement result is output according to the output format.

However, the error numbers output here is the one displayed on the 7-segment display unit (End data or current data).

Header	Program	Query	Response (Number of characters)
FRQ	None	FRQ?	FRQ△△*****.*** (FIX 10) * denotes clock frequency and 3rd decimal place fixed length (MHz)

■ **Command type**

Sequential command

■ **Usage restrictions**

The command is invalid in the following condition.

Query: When clock off occurs, the following value is output.
FRQ△△△△△△△0.000 (CR / LF)

■ **Usage example**

Query: When the clock frequency is 50 MHz

```
OUTPUT△700;"FRQ?"
ENTER△700;B$
PRINT△B$
```

↓

FRQ△△△△50.000 (CR / LF) is output.

: When clock off occurs

```
OUTPUT△700;"FRQ?"
ENTER△700;B$
PRINT△B$
```

↓

FRQ△△△△△△0.000 (CR / LF) is output.

75) OSD? 1-second data measurement result (One-Second Data?)

■ **Function** One-second measurement result is output according to the output format.

Header	Program	Query	Response (Number of characters)
OSD	None	OSD?	●When 1-second error number is less than 1E + 8 OSD△$\underbrace{\text{*.****E-**}}_{\text{Average 1-second error ratio}}$, △$\underbrace{\text{********}}_{\text{Number of 1-second errors}}$ ●When the 1-second error number is 1E + 8 or more OSD△*.****E-**, *.****E** (FIX 19 or 20)

■ **Command type** Sequential command

■ **Usage restrictions** The command is invalid in the following condition, and ERR (CR/LF) is output.

Query: When the 1-second data print function is OFF

The data written in the example below is assumed to be none, and data the same as in the '−' value in ER? or EC? is output.

Query: No measurement data

■ **Usage example**

Query: When the 1-second error ratio is 1.0E-4, and the 1-second error number is 1000

```
OUTPUT△700;"OSD?"
ENTER△700;B$
PRINT△B$
```

↓

OSD△1.0000E-4,△△△△△1000 (CR/LF) is output.

: When the 1-second error ratio is 1.0E-10, and the 1-second error number is 1

```
OUTPUT△700;"OSD?"
ENTER△700;B$
PRINT△B$
```

↓

OSD△1.0000E-10,△△△△△△△△1 (CR/LF) is output.

: When EFI ratio measurement result is '−'

```
OUTPUT△700;"OSD?"
ENTER△700;B$
PRINT△B$
```

↓

OSD△0.0000E-00,1.0000E-99 (CR/LF) is output.

76) AMD? Alarm measurement result (AlarM Data?)

■ **Function**

Alarm measurement data is output according to the output format.

Header	Program	Query	Response (Number of characters)
AMD	None	AMD?m	See the following description.

■ **Value of m**

m : Select alarm measurement items.

- Ø : all items 1 to 6 below are output
- 1 : Power fail time
- 2 : Power fail recovery time
- 3 : Clock loss time
- 4 : Clock loss recovery time
- 5 : Sync loss time
- 6 : Sync loss recover time

■ **Output format**

When all items are output: ✱✱-✱✱-✱✱△✱✱:✱✱:✱✱

Year Day Minute

Month Hour Second

The above is output from items 1 to 6 in this order by punctuating with commas (,).

When items 1 to 6 are output separately:

The each item of data is output in the following format.

✱✱-✱✱-✱✱△✱✱:✱✱:✱✱

Year Day Minute

Month Hour Second

■ **Command type**

Sequential command

■ **Usage restrictions**

The command is invalid in the following condition, and 99-99-99△99:99:99 (CR / LF) is output.

Query: Alarm occurrence and recovery time when no alarms occur
Alarm recovery time when the alarm is not recovered
Alarm occurrence and recovery time which occurred when measurement stopped

■ Usage example

Query: When requesting the power fail time

```
OUTPUT△7ØØ;"AMD?△1"
ENTER△7ØØ;B$
PRINT△B$
```

↓

95-Ø1-28△13:52:59 (CR / LF) is output.

When requesting in invalid condition

```
OUTPUT△7ØØ;"AMD?△2"
ENTER△7ØØ;B$
PRINT△B$
```

↓

99-99-99△99:99:99 (CR / LF) is output.

77) EDS

Storing measurement end data in buffer
(End-Data buffer Store)

■ Function — Measurement end data is stored in buffer.

Header	Program	Query	Response (Number of characters)
EDS	EDS	None	None

■ Command type — Sequential command

■ Usage restrictions — The command is invalid in the following conditions.

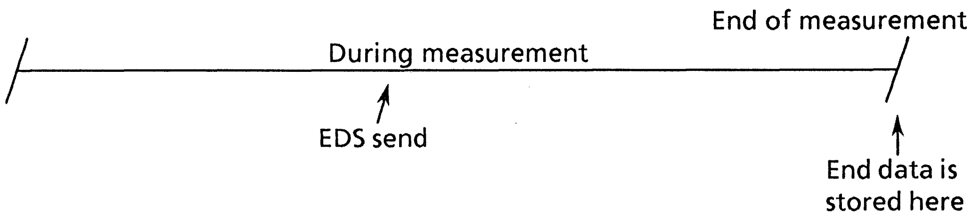
Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed
When measurement is not executed

■ Usage example — Program: When measurement end data is stored in buffer

OUTPUT△7ØØ;"EDS"

■ Note — The measurement end data is stored at the time that the current measurement data is terminated.

Before storing the measurement data, clear a buffer area.



78) EDC

Clearing buffer of measurement end data
(End-Data buffer Clear)

■ Function

The measurement end data buffer is cleared.

Header	Program	Query	Response (Number of characters)
EDC	EDC	None	None

■ Command type

Sequential command

■ Usage restrictions

The command is invalid in the following conditions.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

■ Usage example

Program: When the measurement end data buffer is cleared
OUTPUT△7ØØ;"EDC"

■ Note

By executing EDC, all the data stored by the EDS operation is erased.

79) END? Measurement end data read (**END-data buffer read?**)

■ Function

Measurement end data is read.

Header	Program	Query	Response (Number of characters)
END	None	END?m1,m2	Measurement end data is output according to the following output format.

■ Value of m

m1 : Data type
- 0 : Time data
- 1 : Error measurement data
- 2 : Alarm measurement data
- 3 : Threshold EI, EFI data
- 4 : Error performance data

m2 : Output numbers
- 0 : All items output
- =>1 : 1 data item output (See the output format below)

■ Output format

m1= 0 : Time data

m2=1 :	Measurement start time	(FORM 1)
2 :	Measurement end time	(FORM 1)
3 :	Measurement period	(FORM 1)
4 :	Measurement timed time	(FORM 1)

m1= 1 : Alarm measurement data

m2=1 :	POWER FAIL intervals	(FORM 2)
2 :	CLOCK LOSS intervals	(FORM 2)
3 :	SYNC LOSS intervals	(FORM 2)

m1= 2 : Error measurement data

m2=1 :	ERROR RATIO	(FORM 3)
2 :	ERROR COUNT	(FORM 4)
3 :	EI	(FORM 4)
4 :	%EFI	(FORM 5)

m1= 3 : Threshold EI, EFI data

m2=1 :	$>10^{-3}$	(FORM 6)
2 :	$>10^{-4}$	(FORM 6)
3 :	$>10^{-5}$	(FORM 6)
4 :	$>10^{-6}$	(FORM 6)
5 :	$>10^{-7}$	(FORM 6)
6 :	$>10^{-8}$	(FORM 6)
7 :	$<=10^{-8}$	(FORM 6)

m1 = 4 : Error performance data

m2 = 1	: ES	(FORM 5)
2	: EFS	(FORM 5)
3	: SES	(FORM 5)
4	: DM	(FORM 5)
5	: US	(FORM 5)

(FORM 1) Time data type

--**△**:**:** (No. of chars. 17)

Year, Month, Day, Hour, Minute, Second

(FORM 2) Numerical value data type

********** (No. of chars. 10)

(FORM 3) Exponent data type

*.****E-** (No. of chars. 10)

*.****E-* (No. of chars. 9)

(FORM 4) Numeric value and exponent data type

- Less than 1E + 8

  △******** (No. of chars. 9)

- 1E + 8 or more

  *.****E** (No. of chars. 9)

(FORM 5) %data type

.* (No. of chars. 8)

(FORM 6) Mixed data type

- Less than 1E + 8

  △********,***.**** (No. of chars. 18)

- 1E + 8 or more

  *.****E**,***.**** (No. of chars. 18)

■ Command type

Sequential command

■ Usage restrictions

The following condition is invalid and ERR (CR / LF) is output.

Query: When no data is in buffer

■ Usage example

Query: When only the measurement start time of the time data is read from measurement end data

```
OUTPUT△700;"END?△0,1"
ENTER△700;B$
PRINT△B$
```

↓

95-01-30△15:24:59 (CR / LF) is output.

80) IMS

Storing measurement intermediate data to buffer
(InterMediate-data buffer Store)

■ Function

Intermediate measurement data is stored in buffer.

Header	Program	Query	Response (Number of characters)
IMS	IMS	None	None

■ Command type

Sequential command

■ Usage restrictions

The command is invalid in the following conditions.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed
When measurement is not executed

■ Usage example

Program: When intermediate measurement data is stored in buffer
OUTPUT△700;"IMS"

81) IMC

Clearing the buffer for measurement intermediate data
(InterMediate-data buffer Clear)

■ Function

Intermediate measurement buffer is cleared of data.

Header	Program	Query	Response (Number of characters)
IMC	IMC	None	None

■ Command type

Sequential command

■ Usage restrictions

The command is invalid in the following conditions.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

■ Usage example

Program: When an intermediate measurement data buffer is cleared
OUTPUT△7ØØ;"IMC"

■ Note

By clearing the buffer, measurement data stored by the IMS operation is erased.

82) IMD?

Reading intermediate measurement data
(InterMediate-Data buffer read?)

■ Function

Intermediate measurement data is read.

Header	Program	Query	Response (Number of characters)
IMD	None	IMD?m1,m2	Intermediate measurement data is output according to the output format below.

■ Value of m

m1 : Data type
- 0 : Time data
- 1 : Alarm measurement data
- 2 : Error measurement data
- 3 : Threshold EI, EFI data
- 4 : Error performance data

m2 : Output numbers
- 0 : All items output
- => 1 : 1 item data output (See the output format below)

■ Output format

m1= 0 : Time data

m2=1	: Measurement start time	(FORM 1)
2	: Measurement intermediate time	(FORM 1)
3	: Measurement elapsed time	(FORM 1)
4	: Measurement timed time	(FORM 1)

m1= 1 : Alarm measurement data

m2=1	: POWER FAIL intervals	(FORM 2)
2	: CLOCK LOSS intervals	(FORM 2)
3	: SYNC LOSS intervals	(FORM 2)

m1= 2 : Error measurement data

m2=1	: ERROR RATIO	(FORM 3)
2	: ERROR COUNT	(FORM 4)
3	: EI	(FORM 4)
4	: %EFI	(FORM 5)

m1= 3 : Threshold EI, EFI data

m2=1	: $>10^{-3}$	(FORM 6)
2	: $>10^{-4}$	(FORM 6)
3	: $>10^{-5}$	(FORM 6)
4	: $>10^{-6}$	(FORM 6)
5	: $>10^{-7}$	(FORM 6)
6	: $>10^{-8}$	(FORM 6)
7	: $<=10^{-8}$	(FORM 6)

m1 = 4 : Error performance data

m2 = 1 : ES (FORM 5)
2 : EFS (FORM 5)
3 : SES (FORM 5)
4 : DM (FORM 5)
5 : US (FORM 5)

(FORM 1) Time data type

--**△**:**:** (No. of chars. 17)

Year Day Minute

Month Hour Second

(FORM 2) Numerical value data type

********** (No. of chars. 10)

(FORM 3) Exponent data type

*.****E-** (No. of chars. 10)
*.****E-* (No. of chars. 9)

(FORM 4) Numeric value and exponent data type

- Less than 1E + 8

△******** (No. of chars. 9)

- 1E + 8 or more

*.****E** (No. of chars. 9)

(FORM 5) %data type

.* (No. of chars. 8)

(FORM 6) Mixed data type

- Less than 1E + 8

△********,***.**** (No. of chars. 18)

- 1E + 8 or more

*.****E**,***.**** (No. of chars. 18)

■ Command type Sequential command

■ Usage restrictions The following condition is invalid and ERR (CR / LF) is output.

Query: When no data is in buffer
If measurement period is 1 minute, intermediate data is not generated.

■ Usage example

Query: When only the measurement start time of the time data is read from intermediate measurement data

```
OUTPUT△700;"IMD?△0,1"
ENTER△700;B$
PRINT△B$
```

↓

95-01-30△15:24:59 (CR / LF) is output.

• Other section

Each control message in the OTHER section is explained in the following pages.

The triangle marks (△) indicates a spaces.

■ Note

When remote mode is selected, the contents of command specification have priority; when local mode is selected, then the rear panel function switching in this section is selected.
Keep in mind this.

83) PRN Printer function (PRiNter enable)

■ Function Printer output ON / OFF is controlled.

Header	Program	Query	Response (Number of characters)
PRN	PRN△m	PRN?	PRN△m (FIX 1)

■ Value of m
Ø : OFF
1 : ON

■ Command type Sequential command

■ Usage restrictions The command is invalid in the following conditions.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ Usage example

Program: When a printer is set to OFF
```
OUTPUT△7ØØ;"PRN△Ø"
```

Query: When a printer was set to ON
```
OUTPUT△7ØØ;"PRN?"
ENTER△7ØØ;B$
PRINT△B$
```
↓
PRN△1 (CR / LF) is output.

84) PSA Manual print (Printer StArt)

■ Function Manual printing start is specified.

Header	Program	Query	Response (Number of characters)
PSA	PSA	None	None

■ Command type Sequential command

■ Usage restrictions The command is invalid in the following conditions.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed
In the first 1 second after measurement start
When measurement is stopped
When the printer function is OFF

■ Usage example Program: When manual printing starts

OUTPUT△700;"PSA"

85) ALM Alarm monitor function (ALarm Monitor on / off)

■ Function — Alarm monitoring ON / OFF is controlled.

Header	Program	Query	Response (Number of characters)
ALM	ALM△m	ALM?	ALM△m (FIX 1)

■ Value of m

Ø : OFF
1 : ON

■ Command type — Sequential command

■ Usage restrictions — The command is invalid in the following conditions.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ Usage example

Program: When alarm monitor function is set to OFF
OUTPUT△7ØØ;"ALM△Ø"

Query: When alarm monitor function was set to ON
OUTPUT△7ØØ;"ALM?"
ENTER△7ØØ;B$
PRINT△B$
↓
ALM△1 (CR / LF) is output.

86) MON — Error monitor function **(error MONitor on / off)**

■ **Function** — Error monitor function ON / OFF is controlled.

Header	Program	Query	Response (Number of characters)
MON	MON△m	MON?	MON△m (FIX 1)

■ **Value of m**

Ø : OFF
1 : ON

■ **Command type** — Sequential command

■ **Usage restrictions** — The command is invalid in the following conditions.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ **Usage example**

Program: When error monitor function is specified as OFF

```
OUTPUT△7ØØ;"MON△Ø"
```

Query: When error monitor function was specified as ON

```
OUTPUT△7ØØ;"MON?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

MON△1 (CR / LF) is output.

87) SOP Synchronous signal output selection (**S**ync **O**ut**P**ut)

■ Function

Synchronous signal output is controlled.

Header	Program	Query	Response (Number of characters)
SOP	SOP△m	SOP?	SOP△m (FIX 1)

■ Value of m

Ø : 1 / 32 CLOCK
1 : PATTERN SYNC (FIXED)
2 : PATTERN SYNC (VARIABLE)

■ Command type

Sequential command

■ Usage restrictions

The command is invalid in the following conditions.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ Usage example

Program: When synchronous signal output is specified as 1 / 32 CLOCK
OUTPUT△7ØØ;"SOP△Ø"

Query: When synchronous signal output was specified as PATTERN SYNC(FIXED)
OUTPUT△7ØØ;"SOP?"
ENTER△7ØØ;B$
PRINT△B$
↓
SOP△1 (CR / LF) is output.

88) GPA GPIB address (GPib 2 Address)

■ **Function** GPIB 2 (output port exclusively used for printer) address is specified.

Header	Program	Query	Response (Number of characters)
GPA	GPA△m	GPA?	GPA△m (FIX 2)

■ **Value of m** GPIB 2 address 1 to 30 is specified.

Range of numeric values Maximum: 30
Minimum: 0
Step: 1

■ **Command type** Sequential command

■ **Usage restrictions** The command is invalid in the following conditions.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ **Usage example** Program: When GPIB 2 address is specified as 0

```
OUTPUT△700;"GPA△0"
```

Query: When GPIB 2 address was specified as 1

```
OUTPUT△700;"GPA?"
ENTER△700;B$
PRINT△B$
```

↓

GPA△△1 (CR / LF) is output.

89) SFT

Number of shifts for mark ratio AND bit
(mark ratio and bit ShiFT)

■ **Function**

The number of shifts of the PRBS mark ratio AND bit is specified.

Header	Program	Query	Response (Number of characters)
SFT	SFT△m	SFT?	SFT△m (FIX 1)

■ **Value of m**

Ø : 1 bit shift
1 : 3 bits shift

■ **Command type**

Sequential command

■ **Usage restrictions**

The command is invalid in the following conditions.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed
When measurement pattern is ALTERNATE, DATA, or ZERO SUBST

Query: In the following case the command is invalid and ERR (CR / LF) is output.
When the measurement pattern is ALTERNATE, DATA, or ZERO SUBST

■ **Usage example**

Program: When the number of shifts for the mark ratio AND bit is specified as 1 bit

```
OUTPUT△7ØØ;"SFT△Ø"
```

Query: When the number of shifts for the mark ratio AND bit was specified as 1 bit

```
OUTPUT△7ØØ;"SFT?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

SFT△1 (CR / LF) is output.

: When the measurement patter was specified as ALTERNATE, DATA, or ZERO SUBST

```
OUTPUT△7ØØ;"SFT?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

ERR (CR / LF) is output.

90) CLS Clock loss processing function (CLock loSs)

■ **Function** Clock loss processing function is selected.

Header	Program	Query	Response (Number of characters)
CLS	CLS△m	CLS?	CLS△m (FIX 1)

■ **Value of m**
Ø : EXCLUDE
1 : INCLUDE

■ **Command type** Sequential command

■ **Usage restrictions** The command is invalid in the following conditions.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ **Usage example**
Program: When the clock loss status is to be excluded from calculation
```
OUTPUT△7ØØ;"CLS△Ø"
```
Query: When the clock loss status is to be included in calculation
```
OUTPUT△7ØØ;"CLS?"
ENTER△7ØØ;B$
PRINT△B$
```
↓
CLS△1 (CR / LF) is output.

■ **Note**
EXCLUDE means the clock loss is executed from the calculation.
INCLUDE means the clock loss is included in the calculation.

91) SLS Sync loss processing (Sync LoSs)

■ **Function** Sync loss processing function is selected.

Header	Program	Query	Response (Number of characters)
SLS	SLS△m	SLS?	SLS△m (FIX 1)

■ **Value of m**
Ø : EXCLUDE
1 : INCLUDE

■ **Command type** Sequential command

■ **Usage restrictions** The command is invalid in the following conditions.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ **Usage example**
Program: When a sync loss is to be excluded from calculation

```
OUTPUT△7ØØ;"SLS△Ø"
```

Query: When a sync loss is to be included in calculation

```
OUTPUT△7ØØ;"SLS?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

SLS△1 (CR / LF) is output.

■ **Note** EXCLUDE means the sync loss is excluded from calculation.
INCLUDE means the sync loss is included in calculation.

92) ETH

Error performance threshold selection function
(Error performance THreshold)

■ Function

Print threshold for the error performance data is specified.

Header	Program	Query	Response (Number of characters)
ETH	ETH△m	ETH?	ETH△m (FIX 1)

■ Value of m

Ø : 1.0E-3
1 : 1.0E-4

■ Command type

Sequential command

■ Usage restrictions

The command is invalid in the following conditions.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ Usage example

Program: When a print threshold for an error performance is specified as 1.0E-3

```
OUTPUT△700;"ETH△0"
```

Query: When a print threshold for an error performance was specified as 1.0E-4

```
OUTPUT△700;"ETH?"
ENTER△700;B$
PRINT△B$
```

↓

ETH△1 (CR / LF) is output.

93) BST

BURST measurement mode (**B**ur**ST** mode)

■ Function

BURST measurement mode is specified.

Header	Program	Query	Response (Number of characters)
BST	BST△m	BST?	BST△m (FIX 1)

■ Value of m

Ø : OFF
1 : ON

■ Command type

Sequential command

■ Usage restrictions

The command is invalid in the following conditions.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ Usage example

Program: When a BURST mode is specified as OFF

```
OUTPUT△7ØØ;"BST△Ø"
```

Query: When a BURST mode was specified as OFF

```
OUTPUT△7ØØ;"BST?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

BST△1 (CR / LF) is output.

■ Note

When the BURST mode is ON, the alarm monitor method is changed as follows.

- Clock loss: When no clock is input for 100 msecs, clock loss occurs.
  Clock loss does not occur in other cases.
- Sync loss: This cannot be detected.

94) CAL

Intermediate measurement data calculation function
(current data CALculation)

■ **Function**

Intermediate measurement data (current data) calculation is specified.

Header	Program	Query	Response (Number of characters)
CAL	CAL△m	CAL?	CAL△m (FIX 1)

■ **Value of m**

Ø : Cumulative data
1 : Immediate data

■ **Command type**

Sequential command

■ **Usage restrictions**

The command is invalid in the following conditions.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ **Usage example**

Program: When a intermediate measurement data is calculated in the cumulative data

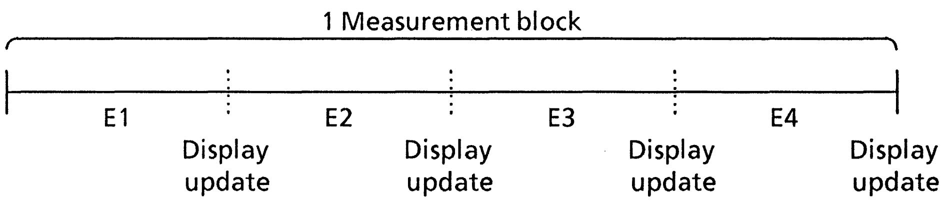
OUTPUT△7ØØ;"CAL△Ø"

Query: When intermediate measurement data calculation was specified as the immediate calculation data

OUTPUT△7ØØ;"CAL?"
ENTER△7ØØ;B$
PRINT△B$
↓
CAL△1 (CR / LF) is output.

■ **Note**

Cumulative data is added according to the current data updating period (100 msec / 200 msec).



- Cumulative data = E1 + E2 + E3 + E4 + · · · · · ·
- Immediate data = E1
  E2
  E3
  E4

Immediate data is the measurement data at the display update period (100 msec / 200 msec) block. (Data is not added.)

95) ETY

Error detection mode selection (**E**rror **TY**pe)

■ Function

Error monitor mode is selected.

Header	Program	Query	Response (Number of characters)
ETY	ETY△m	ETY?	ETY△m (FIX 1)

■ Value of m

Ø : Total error
1 : Insertion error
2 : Omission error

■ Command type

Sequential command

■ Usage restrictions

The command is invalid in the following conditions.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ Usage example

Program: When the error monitor mode is specified as total error
OUTPUT△7ØØ;"ETY△Ø"

Query: When the error monitor mode was specified as insertion error
OUTPUT△7ØØ;"ETY?"
ENTER△7ØØ;B$
PRINT△B$
↓
ETY△1 (CR / LF) is output.

96) EIT
EI / %EFI interval time **(Ei, %efi Interval Time)**

■ Function

EI, %EFI interval time is specified.

Header	Program	Query	Response (Number of characters)
EIT	EIT△m	EIT?	EIT△m (FIX 1)

■ Value of m

Ø : 1 msec
1 : 10 msec
2 : 100 msec
3 : 1 sec

■ Command type

Sequential command

■ Usage restrictions

The command is invalid in the following conditions.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ Usage example

Program: When an interval time is specified as 1 msec

```
OUTPUT△7ØØ;"EIT△Ø"
```

Query: When an interval time was specified as 10 msec

```
OUTPUT△7ØØ;"EIT?"
ENTER△7ØØ;B$
PRINT△B$
          ↓
EIT△1 (CR/LF) is output.
```

97) FMT Data print format (output data ForMaT)

■ **Function** Printing format is specified.

Header	Program	Query	Response (Number of characters)
FMT	FMT△m	FMT?	FMT△m (FIX 1)

■ **Value of m**
Ø : Standard format
1 : Abridged (short) format

■ **Command type** Sequential command

■ **Usage restrictions** The command is invalid in the following conditions.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ **Usage example**

Program: When a print format is specified as standard format
```
OUTPUT△7ØØ;"FMT△Ø"
```

Query: When a print format is specified as Abridged (short) format
```
OUTPUT△7ØØ;"FMT?"
ENTER△7ØØ;B$
PRINT△B$
```
↓
FMT△1 (CR / LF) is output.

98) THR

Threshold EI, %EFI data print selection
(THReshold ei / %efi data output)

■ Function

Printing for threshold EI, %EFI data is specified.

Header	Program	Query	Response (Number of characters)
THR	THR△m	THR?	THR△m (FIX 1)

■ Value of m

Ø : Do not print
1 : Print

■ Command type

Sequential command

■ Usage restrictions

The command is invalid in the following conditions.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ Usage example

Program: When threshold EI, %EFI data is not printed

```
OUTPUT△7ØØ;"THR△Ø"
```

Query: When threshold EI, %EFI data is printed

```
OUTPUT△7ØØ;"THR?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

THR△1 (CR / LF) is output.

99) EPF

Error performance data print selection
(Error PerFormance data output)

■ Function

Printing of error performance data is specified.

Header	Program	Query	Response (Number of characters)
EPF	EPF△m	EPF?	EPF△m (FIX 1)

■ Value of m

Ø : Do not print
1 : Print

■ Command type

Sequential command

■ Usage restrictions

The command is invalid in the following conditions.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ Usage example

Program: When error performance data is not printed
```
OUTPUT△7ØØ;"EPF△Ø"
```

Query: When error performance data is printed
```
OUTPUT△7ØØ;"EPF?"
ENTER△7ØØ;B$
PRINT△B$
```
↓
EPF△1 (CR / LF) is output.

100) ITM

Intermediate data print selection
(InTerMediate data output)

■ Function

Intermediate data printing is specified.

Header	Program	Query	Response (Number of characters)
ITM	ITM△m	ITM?	ITM△m (FIX 1)

■ Value of m

Ø : Do not print
1 : Print

■ Command type

Sequential command

■ Usage restrictions

The command is invalid in the following conditions.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ Usage example

Program: When intermediate data is not printed

```
OUTPUT△7ØØ;"ITM△Ø"
```

Query: When intermediate data is printed

```
OUTPUT△7ØØ;"ITM?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

ITM△1 (CR / LF) is output.

101) OSC

1-second data print selection
(One-SeCond data output)

■ Function

1-second data printing is specified.

Header	Program	Query	Response (Number of characters)
OSC	OSC△m	OSC?	OSC△m (FIX 1)

■ Value of m

Ø : Do not print
1 : Print

■ Command type

Sequential command

■ Usage restrictions

The command is invalid in the following conditions.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ Usage example

Program: When 1-second data is not printed
OUTPUT△7ØØ;"OSC△Ø"

Query: When 1-second data is printed
OUTPUT△7ØØ;"OSC?"
ENTER△7ØØ;B$
PRINT△B$
↓
OSC△1 (CR / LF) is output.

102) DOT

1-second data print threshold selection
(Data Output Threshold)

■ Function

1-second data threshold is selected.

Header	Program	Query	Response (Number of characters)
DOT	DOT△m	DOT?	DOT△m (FIX 1)

■ Value of m

Ø : Error>0
1 : Error>1.0E-6
2 : Error>1.0E-4
3 : Error>1.0E-3

■ Command type

Sequential command

■ Usage restrictions

The command is invalid in the following conditions.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ Usage example

Program: When a 1-second data print threshold is specified as error>0

```
OUTPUT△7ØØ;"DOT△Ø"
```

Query: When a 1-second data print threshold was specified as error>1.0E-6

```
OUTPUT△7ØØ;"DOT?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

DOT△1 (CR / LF) is output.

103) PSV Paper saving function (Paper SaVe)

■ Function Paper saving function is selected.

Header	Program	Query	Response (Number of characters)
PSV	PSV△m	PSV?	PSV△m (FIX 1)

■ Value of m

Ø : OFF
1 : ON

■ Command type Sequential command

■ Usage restrictions The command is invalid in the following conditions.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ Usage example

Program: When the paper saving function is specified as OFF

```
OUTPUT△7ØØ;"PSV△Ø"
```

Query: When the paper saving function was specified as ON

```
OUTPUT△7ØØ;"PSV?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

PSV△1 (CR / LF) is output.

104) ITV

Measurement interval time
(measurement InTerVal time)

■ **Function**

Measurement interval time is selected.

Header	Program	Query	Response (Number of characters)
ITV	ITV△m	ITV?	ITV△m (FIX 1)

■ **Value of m**

Ø : 100 msec
1 : 200 msec

■ **Command type**

Sequential command

■ **Usage restrictions**

The command is invalid in the following conditions.

Program: During automatic phase threshold search operation
While Eye margin measurement is being executed
When a floppy disk is being accessed

Query: None

■ **Usage example**

Program: When a measurement interval time is specified as 100 msec

```
OUTPUT△7ØØ;"ITV△Ø"
```

Query: When a measurement interval time was specified as 200 msec

```
OUTPUT△7ØØ;"ITV?"
ENTER△7ØØ;B$
PRINT△B$
↓
```

ITV△1 (CR / LF) is output.

SECTION 10
EXAMPLE OF PROGRAM CREATION

TABLE OF CONTENTS

(Blank)

This section describes examples of how to create MP1764A GPIB programs.

The sample programs which appear in this section were written for the HP9000 series computer of Hewlett-Packard and for a PC-compatible computer with GPIB interface card of National Instruments (N.I).

The program for the HP9000 were written in HP-BASIC while those for IBM-PC-compatible were written in Microsoft QUICK BASIC Version 4.50.

The programs were verified by running them on the HP9000-200 / 300 using HP-BASIC V5.12 and DECpc computer with GPIB interface card of N.I, using Microsoft QUICK-BASIC Version 4.50.

The program examples described here are:

(1) Input signal setting

(2) Automatic threshold search (Auto search) setting

(3) Eye margin measurement

(4) Measurement pattern, BIT WINDOW, and BLOCK WINDOW setting

(5) Error analysis

(6) Measurement result display (displayed using serial polling)

(7) Measurement result display (displayed using request command)

(8) Intermediate measurement data display

(9) Reading file information from floppy disk

(10) Floppy disk operation

(11) Status byte checking

(12) DMA transfer for pattern data

(13) DMA transfer for BLOCK WINDOW

Table 10-1 shows the preparations that must be made for each controller prior to sample program execution.

Table 10-1 Preparation for Sample Program Execution (1 / 2)

Controller	Preparation for program execution
HP9000	• Set the GPIB address of MP1764A as "1". • Set the GPIB address of MP1763A as "2". • Connects the MP1764A, MP1763A, and HP9000 with GPIB cables.
DEC pc	• Set the GPIB address of MP1764A as "1". • Set the GPIB address of MP1763A as "2". • Set IBCONF as follows. ① <Board Characteristics> Board : GPIB Ø (Defines board as "GPIBØ") Primary GPIB Address: 0 Secondary GPIB Address: NONE Timeout setting: 1000 sec Terminate Read on EOS: Yes Set EOI with EOS on Writes: Yes Type of Compare on EOS: 7-Bit EOS byte: ØAH Send EOI at end of Write: Yes System Controller: Yes Assert REN when SC: No Enable Auto Serial Polling: Yes Enable CIC Protocol: No Bus timing: 500 nsec Cable Length for High Speed: off Parallel Poll Duration: Default Use this GPIB interface: Yes Base I / O Address: Ø2cØh Interrupt Level: 11 DMA Channel: 5 DMA Transfer Mode: Demand

Table 10-1 Preparation for Sample Program Execution (2 / 2)

Controller	Preparation for program execution
DEC pc	② <Device Characteristics> Device : ED (Defines device name as "ED") Primary GPIB Address 1 Secondary GPIB Address NONE Timeout setting 1000 sec Serial Poll Timeout 1 sec Terminate Read on EOS Yes Set EOI with EOS on Writes Yes Type of compare on EOS 7-Bit EOS byte ØAh Send EOI at end of Write Yes Enable Repeat Addressing No ③ <Device Characteristics> Device : PPG (Defines device name as "PPG") Primary GPIB Address 2 Secondary GPIB Address NONE Timeout setting 1000 sec Serial Poll Timeout 1 sec Terminate Read on EOS Yes Set EOI with EOS on Writes Yes Type of compare on EOS 7-Bit EOS byte ØAh Send EOI at end of Write Yes Enable Repeat Addressing No ④ Devices ② and ③are connected to the GPIBØ of device ① using the GPIB Device Map. ● Connects MP1764A, MP1763A, and DECpc with GPIB cables.

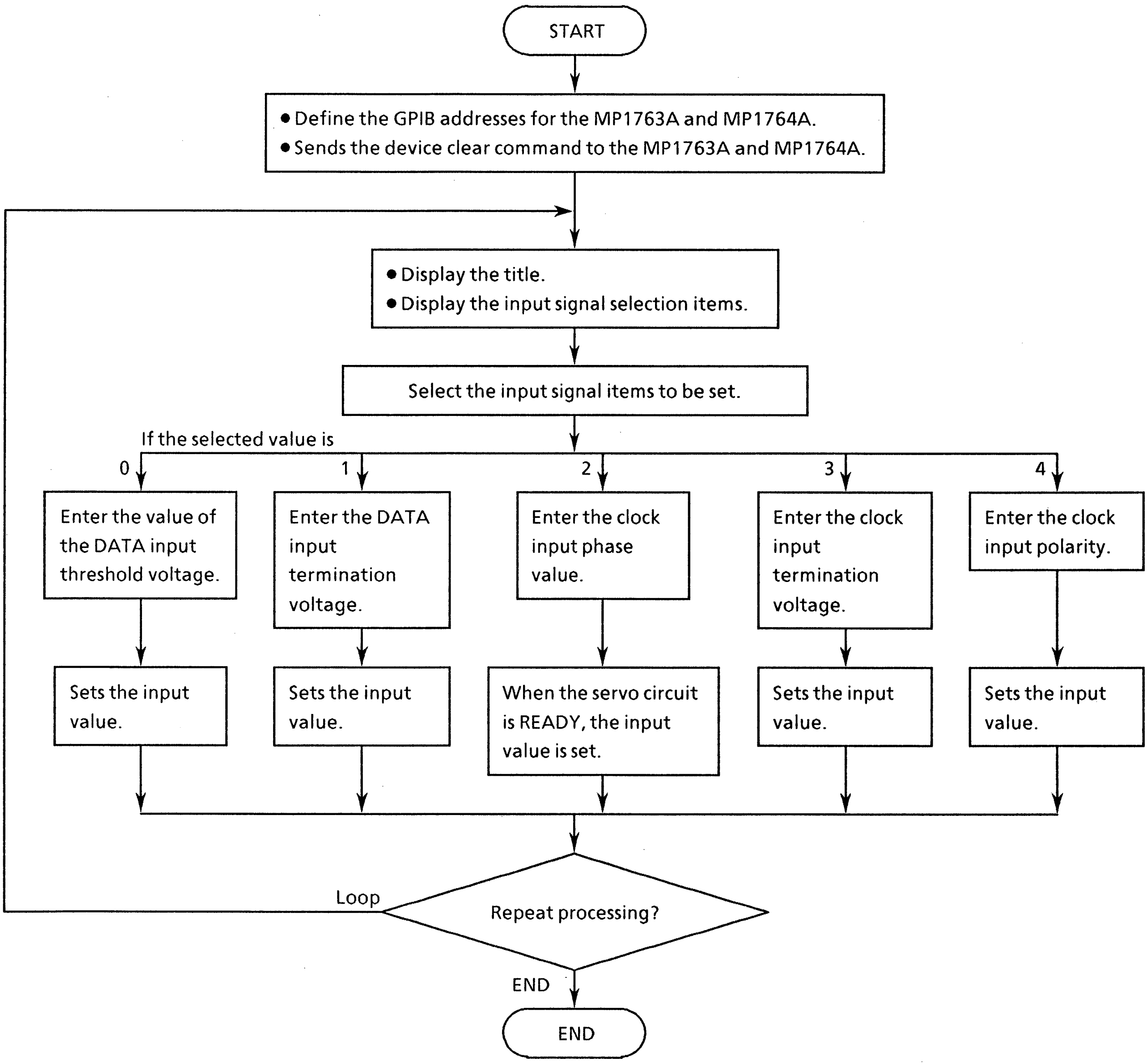
10.1 Example of Program creation Using HP9000

(1) Setting input signals

This program controls input signal conventions and characteristics.

Input signals (DATA, CLOCK voltage, phase, polarity, etc.) are selected according to a message, and are set in the MP1764A.

Note that when setting the clock input phase, the program enters a delay state to ensure that the instrument is READY before applying the delay value.



- Program list

```
10    !**********************************************************************
20    !*                                                                   *
30    !*          MP1762A/MP1764A INPUT SIGNAL SAMPLE PROGRAM              *
40    !*                                                INP_SET.BAS        *
50    !**********************************************************************
60    !
70    Add=701                                      !MP1762A/MP1764A ADDRESS
80    CLEAR Add                                    !DEVICE CLEAR
90    !
100   LOOP
110       CLEAR SCREEN
120       !
130       PRINT "** MP1762A/MP1764A INPUT SIGNAL SAMPLE PROGRAM **"
140       PRINT
150       PRINT "INPUT SIGNAL * DATA THRESHOLD      = [0] "
160       PRINT "              * DATA TERMINATION    = [1] "
170       PRINT "              * CLOCK PHASE ADJUST  = [2] "
180       PRINT "              * CLOCK TERMINATION   = [3] "
190       PRINT "              * CLOCK POLARITY      = [4] "
200       PRINT
210       INPUT "Choose function [0 to 4]:",Sel$
220       !
230       IF Sel$<>"0" AND Sel$<>"1" AND Sel$<>"2" AND Sel$<>"3" AND Sel$<>"4" T
HEN
240           PRINT "Wrong chosen number!"
250           PRINT "Please enter correct number"
260       END IF
270       !
280       SELECT Sel$
290       !
300           CASE "0"
310               PRINT "Please type number for the DATA THRESHOLD"
320               INPUT "Possible data range is -3.000 to +1.875V STEP 0.001V",D
th$
330               OUTPUT Add;"DTH "&Dth$
340               !
350           CASE "1"
360               INPUT "Choose DATA TERMINATION.[GND:0, -2V:1]",Dtm$
380               OUTPUT Add;"DTM "&Dtm$
390               !
400           CASE "2"
410               PRINT "Please type number for the CLOCK PHASE ADJUST"
420               INPUT "Possible data range is -500 to +500ps STEP 1ps",Cpa$
430               !
440               LOOP
450                   OUTPUT Add;"DLY?"              !REQUEST Delay unlock
460                   ENTER Add;Dly$
470               EXIT IF Dly$="DLY 0"
480               END LOOP
490               !
500               OUTPUT Add;"CPA "&Cpa$
510               !
520           CASE "3"
530               INPUT "Choose CLOCK TERMINATION.[GND:0, -2V:1]",Ctm$
540               OUTPUT Add;"CTM "&Ctm$
550               !
560           CASE "4"
570               INPUT "Choose CLOCK POLARITY.[CLK:0, NCLK:1]",Cpl$
580               OUTPUT Add;"CPL "&Cpl$
590               !
600       END SELECT
610       !
620       INPUT "Do you set another data?[Yes:0, No:1]",Loop$
630   EXIT IF Loop$="1"
640   END LOOP
650   END
```

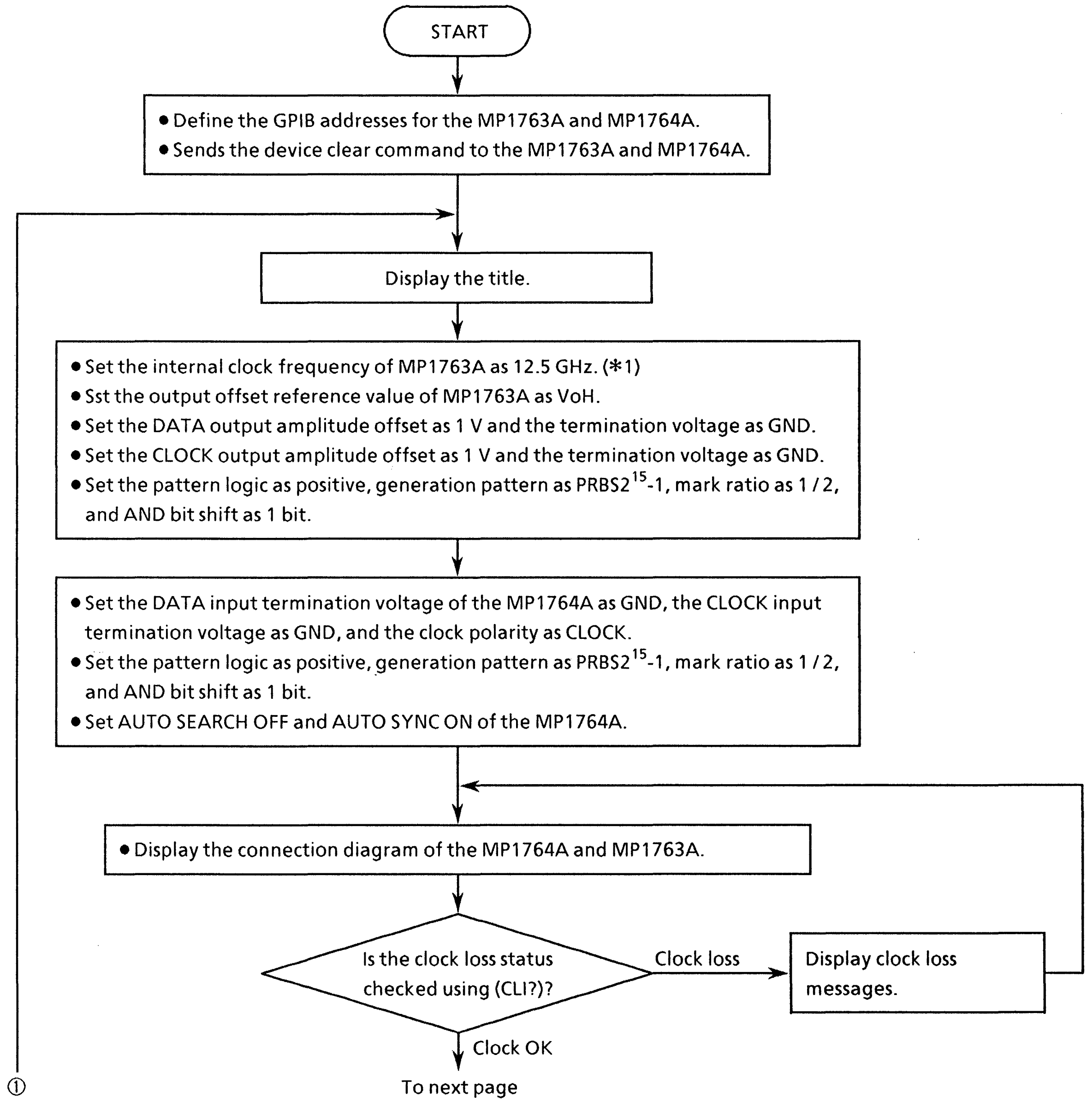
(2) Automatic input threshold search (Auto search) setting

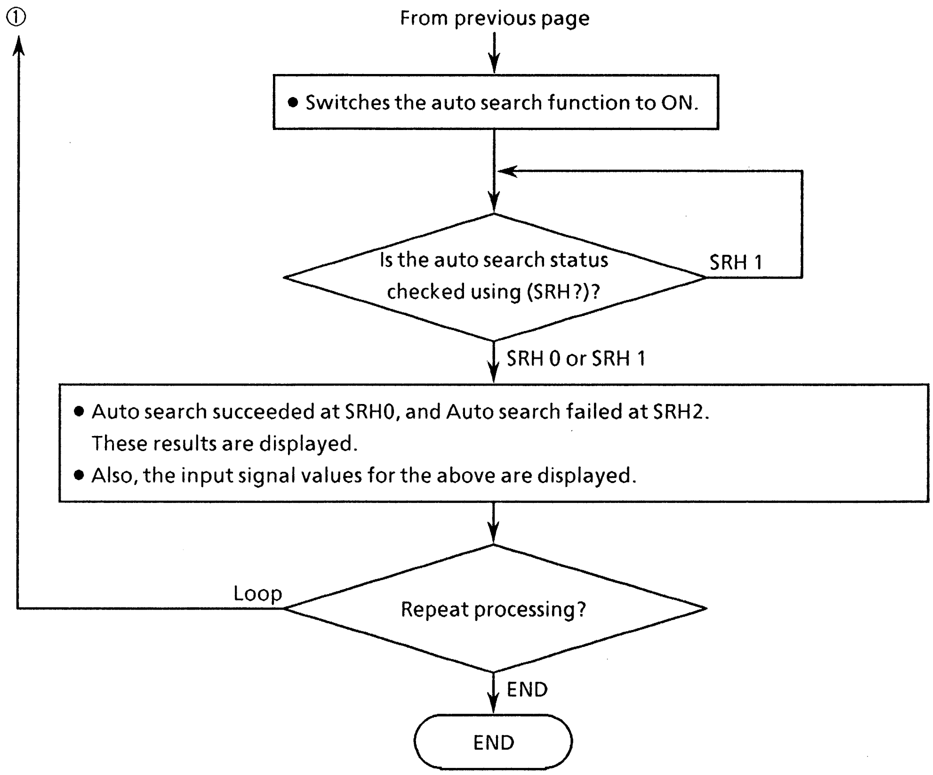
This program executes "auto search" after the MP1764A has been connected to the MP1763A.

First, establish the conditions required to execute Auto search in the MP1764A and MP1763A.

Next, confirm that the clock is not lost because the Auto search becomes invalid if the clock is lost.

Then, turn the Auto search function switch ON. Read using the request command (SRH?), and check whether the Auto search operation succeeded. Then, display the data. The values of the input signals at that time are displayed.





*1 The MP1763A internal clock frequency setting is effective only when the MP1763A OPTION-01 internal synthesizer is installed.

- Program list

```
10     !*********************************************************************
20     !*                                                                   *
30     !*         MP1762A/MP1764A AUTO SEARCH SAMPLE PROGRAM                *
40     !*                                                       AUTO_SRCH   *
50     !*********************************************************************
60     !
70     Add1=701                                    !MP1762A/MP1764A ADDRESS
80     Add2=702                                    !MP1761A/MP1763A ADDRESS
90     CLEAR Add1                                  !DEVICE CLEAR(ED)
100    CLEAR Add2                                  !DEVICE CLEAR(PPG)
110    !
120    !
130    LOOP
140        !
150        CLEAR SCREEN
160        PRINT "** MP1762A/MP1764A AUTO SEARCH SAMPLE PROGRAM ** "
170        PRINT
180        !
190        GOSUB D_set                             !DATA SETTING
200        GOSUB Clock                             !CHECK CLOCK LOSS
210        GOSUB Srch                              !AUTO SEARCH ON
220        GOSUB Result                            !DISPLAY RESULT
230        !
240        INPUT "   Next data set[Yes:0, No:1]",Loop$
250    EXIT IF Loop$="1"
260    END LOOP
270    !
280    STOP
290    !
300    !*********** MP1761A,MP1763A/MP1762A,MP1764A DATA SETTING *********
310 D_set:!
320    !MP1761A/MP1763A DATA SETTING
330    !
340    OUTPUT Add2;"CLK 1;RES 1;FRQ 12500"         !FREQUENCY
350    OUTPUT Add2;"OFS 0"                         !VOH
360    OUTPUT Add2;"DAP 1;DOS 1;DTM 0"             !DATA SET
370    OUTPUT Add2;"CDL 100;CAP 1;COS 1"           !CLOCK SET
380    OUTPUT Add2;"LGC 0;PTS 3;PTN 6;MRK 3;SFT 0" !PATTERN
390    !
400    !MP1762A/MP1764A DATA SETTING
410    !
420    OUTPUT Add1;"DTM 0;CTM 0;CPL 0"             !INPUT
430    OUTPUT Add1;"LGC 0;PTS 3;PTN 6;MRK 3;SFT 0" !PATTERN
440    OUTPUT Add1;"SRH 0;SYN 1"
450    !
460    RETURN
470    !
480    !
490    !******************** Check Connection ***************************
500 Clock:    !
510           !
520    LOOP
530        !
540        GOSUB Connect
550        !
560        OUTPUT Add1;"CLI?"                      !CHECK CLOCK LOSS
570        ENTER Add1;Cli$
580        IF Cli$="CLI 1" THEN
590            PRINT "******** CLOCK LOSS ********* "
600        END IF
610    EXIT IF Cli$="CLI 0"
```

```
620    END LOOP
630    !
640    RETURN
650    !
660    !
670    !********************** AUTO SEARCH FUNCTION *************************
680 Srch:  !
690    !
700    OUTPUT Add1;"SRH 1"
710    !
720    LOOP
730        !
740        OUTPUT Add1;"SRH?"
750        ENTER Add1;Srh$
760        !
770    EXIT IF Srh$="SRH 0" OR Srh$="SRH 2"
780    END LOOP
790    !
800    IF Srh$="SRH 0" THEN
810        PRINT "******** AUTO SEARCH OK *********** "
820    ELSE
830        PRINT "******** Failed in AUTO SEARCH ********* "
840    END IF
850    !
860    RETURN
870    !
880    !
890    !*********************** DISPLAY RESULT FUNCTION *********************
900 Result:  !
910    !
920    OUTPUT Add1;"DTH?"
930    ENTER Add1;Dth$
940    OUTPUT Add1;"DTM?"
950    ENTER Add1;Dtm$
960    IF Dtm$="DTM 0" THEN
970        Dtm$="GND"
980    ELSE
990        Dtm$="-2V"
1000   END IF
1010   !
1020   OUTPUT Add1;"CPA?"
1030   ENTER Add1;Cpa$
1040   !
1050   OUTPUT Add1;"CTM?"
1060   ENTER Add1;Ctm$
1070   IF Ctm$="CTM 0" THEN
1080       Ctm$="GND"
1090   ELSE
1100       Ctm$="-2V"
1110   END IF
1120   !
1130   OUTPUT Add1;"CPL?"
1140   ENTER Add1;Cpl$
1150   IF Cpl$="CPL 0" THEN
1160       Cpl$="CLK"
1170   ELSE
1180       Cpl$="NCLK"
1190   END IF
1200   !
1210   PRINT "DATA THRESHOLD ="&Dth$[5,10]&" V"
1220   PRINT "DATA TERMINATION = "&Dtm$
1230   PRINT "CLOCK PHASE ADJUST = "&Cpa$[6,9]&" ps"
1240   PRINT "CLOCK TERMINATION = "&Ctm$
1250   PRINT "CLOCK POLARITY = "&Cpl$
1260   PRINT
1270   !
```

```
1290  !
1300  !
1310  !*********************** DISPLAY CONNECTION *****************************
1320 Connect: !
1330  !
1340  PEN 3
1350  VIEWPORT 70,140,50,100
1360  SHOW 0,70,0,50
1370  !
1380  CLIP 0,70,5,70
1390  FRAME
1400  !
1410  CSIZE 3,.4
1420  MOVE 25,45
1430  LABEL "<< CONNECTION >>"
1440  !
1450  CSIZE 3,.35
1460  MOVE 6,39
1470  LABEL " MP1761A/MP1763A             MP1762A/MP1764A"
1480  !
1490  MOVE 7,20
1500  RECTANGLE 25,18
1510  !
1520  MOVE 38,20
1530  RECTANGLE 25,18
1540  !
1550  MOVE 26,14
1560  IDRAW 0,9
1570  !
1580  FOR I=0 TO PI*2 STEP PI/12
1590      IDRAW .2*COS(I),.2*SIN(I)
1600  NEXT I
1610  !
1620  MOVE 26,14
1630  IDRAW 21,0
1640  IDRAW 0,9
1650  !
1660  FOR I=0 TO PI*2 STEP PI/12
1670      IDRAW .2*COS(I),.2*SIN(I)
1680  NEXT I
1690  !
1700  MOVE 21,17
1710  IDRAW 0,6
1720  !
1730  FOR I=0 TO PI*2 STEP PI/12
1740      IDRAW .2*COS(I),.2*SIN(I)
1750  NEXT I
1760  !
1770  MOVE 21,17
1780  IDRAW 21,0
1790  IDRAW 0,6
1800  !
1810  FOR I=0 TO PI*2 STEP PI/12
1820      IDRAW .2*COS(I),.2*SIN(I)
1830  NEXT I
1840  !
1850  MOVE 16,25
1860  CSIZE 2.3,.5
1870  LABEL "DATA CLOCK1         DATA  CLOCK"
1880  !
1890  INPUT "Are you ready ? Press return key to start.",A
1900  !
1910  RETURN
1920  !
1930  !
1940  END
```

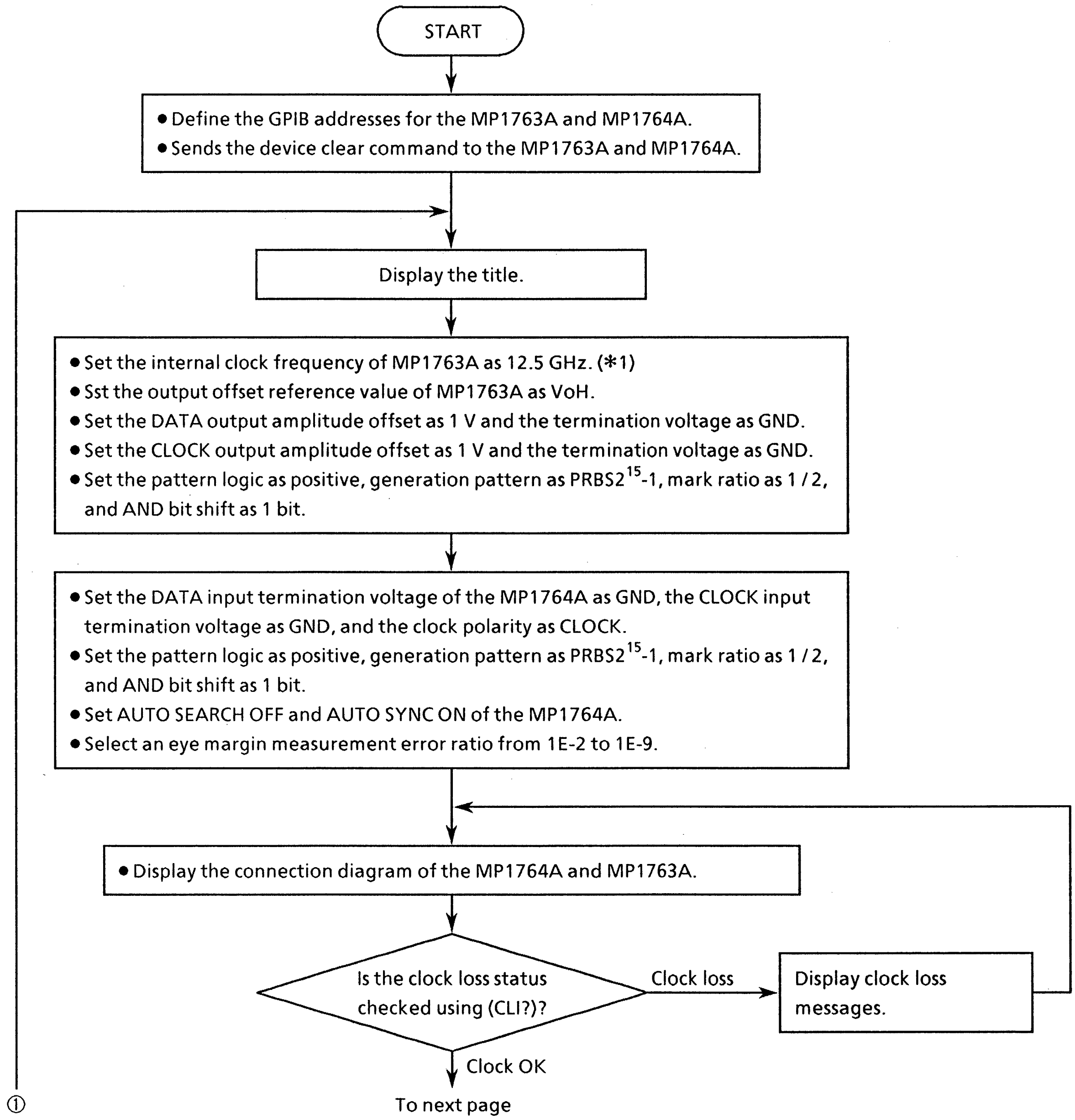
(3) Eye margin measurement

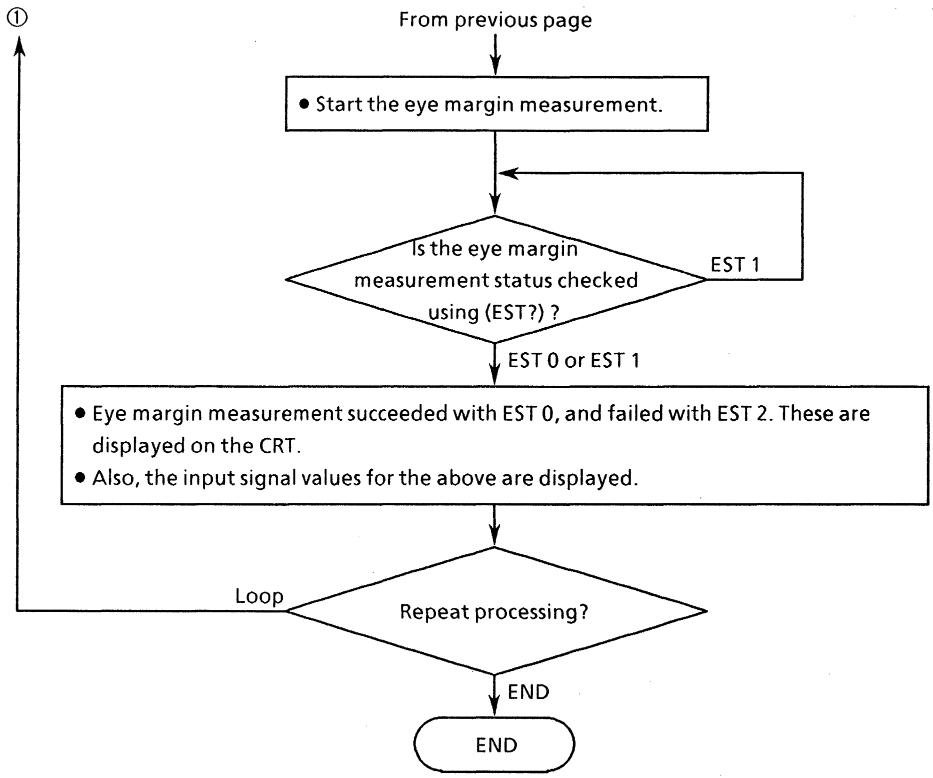
This program executes "eye margin measurement" after the MP1764A has been connected to the MP1763A.

First, establish the conditions for eye margin measurement in the MP1764A and MP1763A.

Next, confirm that the clock is not lost, then start the eye margin measurement. Read by using the request command (EST?) to check if the eye margin measurement succeeded, and display the data.

The values of the input signals at that time are also displayed.





*1 The MP1763A internal clock frequency setting is effective only when the MP1763A OPTION-01 internal synthesizer is installed.

- Program list

```
10    !***********************************************************************
20    !*                                                                    *
30    !*          MP1762A/MP1764A EYE MARGIN SAMPLE PROGRAM                 *
40    !*                                                    EYE_MRGN        *
50    !***********************************************************************
60    !
70    Add1=701                                    !MP1762A/MP1764A ADDRESS
80    Add2=702                                    !MP1761A/MP1763A ADDRESS
90    CLEAR Add1                                  !DEVICE CLEAR(ED)
100   CLEAR Add2                                  !DEVICE CLEAR(PPG)
110   !
120   !
130   LOOP
140       !
150       CLEAR SCREEN
160       PRINT "** MP1762A/MP1764A EYE MARGIN SAMPLE PROGRAM ** "
170       PRINT
180       !
190       GOSUB D_set                             !DATA SETTING
200       GOSUB Clock                             !CHECK CLOCK LOSS
210       GOSUB Eye_mrgn                          !EYE MARGIN START
220       GOSUB Result                            !DISPLAY RESULT
230       !
240       INPUT "   Next data set[Yes:0, No:1]",Loop$
250   EXIT IF Loop$="1"
260   END LOOP
270   !
280   STOP
290   !
300   !************ MP1761A,MP1763A/MP1762A,MP1764A DATA SETTING **********
310 D_set:!
320   !MP1761A/MP1763A DATA SETTING
330   !
340   OUTPUT Add2;"CLK 1;RES 1;FRQ 12500"         !FREQUENCY
350   OUTPUT Add2;"OFS 0"                         !VOH
360   OUTPUT Add2;"DAP 1;DOS 1;DTM 0"             !DATA SET
370   OUTPUT Add2;"CDL 100;CAP 1;COS 1"           !CLOCK SET
380   OUTPUT Add2;"LGC 0;PTS 3;PTN 6;MRK 3;SFT 0"     !PATTERN
390   !
400   !MP1762A/MP1764A DATA SETTING
410   !
420   OUTPUT Add1;"DTM 0;CTM 0;CPL 0"             !INPUT
430   OUTPUT Add1;"LGC 0;PTS 3;PTN 6;MRK 3;SFT 0"     !PATTERN
440   OUTPUT Add1;"SRH 0;SYN 1,SYM 0"
450   PRINT " ** SELECT EYE MARGIN ERROR RATIO ** "
460   PRINT "     ERROR RATIO [ 0 to 7 ]          "
470   PRINT "  0:<=1.0E-2         4:<=1.0E-6      "
480   PRINT "  1:<=1.0E-3         5:<=1.0E-7      "
490   PRINT "  2:<=1.0E-4         6:<=1.0E-8      "
500   PRINT "  3:<=1.0E-5         7:<=1.0E-9      "
510   INPUT "    Select number of the ERROR RATIO = ",Ert$
520   OUTPUT Add1;"EME 1;EYT "&Ert$
530   !
540   RETURN
550   !
560   !
570   !********************* Check Connection *****************************
580 Clock:    !
590           !
600   LOOP
610       !
```

```
620         GOSUB Connect
630         !
640         OUTPUT Add1;"CLI?"                                !CHECK CLOCK LOSS
650         ENTER Add1;Cli$
660         IF Cli$="CLI 1" THEN
670             PRINT "******** CLOCK LOSS ********* "
680         END IF
690     EXIT IF Cli$="CLI 0"
700     END LOOP
710     !
720     RETURN
730     !
740     !
750     !********************** EYE MARGIN FUNCTION *************************
760 Eye_mrgn:!
770     !
780     OUTPUT Add1;"EST 1"
790     !
800     LOOP
810         !
820         OUTPUT Add1;"EST?"
830         ENTER Add1;Est$
840         !
850     EXIT IF Est$="EST 0" OR Est$="EST 2"
860     END LOOP
870     !
880     IF Est$="EST 0" THEN
890         PRINT "******** EYE MARGIN OK *********** "
900     ELSE
910         PRINT "******** Failed in EYE MARGIN ********* "
920     END IF
930     !
940     RETURN
950     !
960     !
970     !********************** DISPLAY RESULT FUNCTION ********************
980 Result:  !
990     !
1000    OUTPUT Add1;"DTH?"
1010    ENTER Add1;Dth$
1020    !
1030    OUTPUT Add1;"THM?"
1040    ENTER Add1;Thm$
1050    IF Thm$="THM -9.999" THEN
1060        Thm$="THM No data "
1070    END IF
1080    !
1090    OUTPUT Add1;"DTM?"
1100    ENTER Add1;Dtm$
1110    IF Dtm$="DTM 0" THEN
1120        Dtm$="GND"
1130    ELSE
1140        Dtm$="-2V"
1150    END IF
1160    !
1170    OUTPUT Add1;"CPA?"
1180    ENTER Add1;Cpa$
1190    !
1200    OUTPUT Add1;"PHM?"
1210    ENTER Add1;Phm$
1220    IF Phm$="PHM -9999" THEN
1230        Phm$="PHM No data "
1240    END IF
1250    !
1260    OUTPUT Add1;"CTM?"
1270    ENTER Add1;Ctm$
```

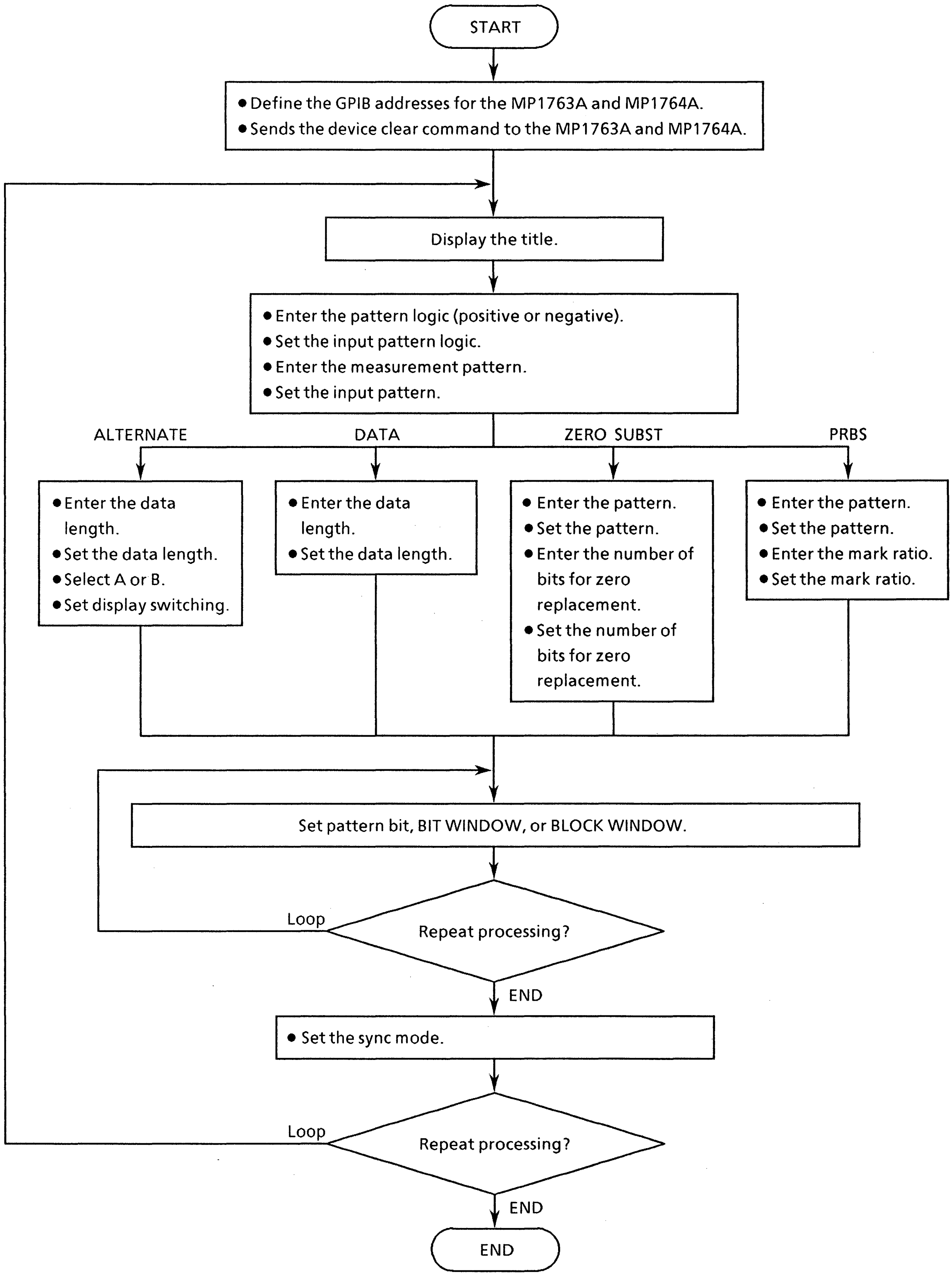
```
1280  IF Ctm$="CTM 0" THEN
1290       Ctm$="GND"
1300  ELSE
1310       Ctm$="-2V"
1320  END IF
1330  !
1340  OUTPUT Add1;"CPL?"
1350  ENTER Add1;Cpl$
1360  IF Cpl$="CPL 0" THEN
1370       Cpl$="CLK"
1380  ELSE
1390       Cpl$="NCLK"
1400  END IF
1410  !
1420  PRINT "DATA THRESHOLD ="&Dth$[5,10]&" V"
1430  PRINT "THRESHOLD MARGIN = "&Thm$[5,10]&" Vp-p"
1440  PRINT "DATA TERMINATION = "&Dtm$
1450  PRINT "CLOCK PHASE ADJUST = "&Cpa$[6,9]&" ps"
1460  PRINT "PHASE MARGIN = "&Phm$[5,10]&" psp-p"
1470  PRINT "CLOCK TERMINATION = "&Ctm$
1480  PRINT "CLOCK POLARITY = "&Cpl$
1490  PRINT
1500  !
1510  RETURN
1520  !
1530  !
1540  !*********************** DISPLAY CONNECTION ***************************
1550 Connect: !
1560  !
1570  PEN 3
1580  VIEWPORT 70,140,50,100
1590  SHOW 0,70,0,50
1600  !
1610  CLIP 0,70,5,70
1620  FRAME
1630  !
1640  CSIZE 3,.4
1650  MOVE 25,45
1660  LABEL "<< CONNECTION >>"
1670  !
1680  CSIZE 3,.35
1690  MOVE 6,39
1700  LABEL " MP1761A/MP1763A                MP1762A/MP1764A"
1710  !
1720  MOVE 7,20
1730  RECTANGLE 25,18
1740  !
1750  MOVE 38,20
1760  RECTANGLE 25,18
1770  !
1780  MOVE 26,14
1790  IDRAW 0,9
1800  !
1810  FOR I=0 TO PI*2 STEP PI/12
1820       IDRAW .2*COS(I),.2*SIN(I)
1830  NEXT I
1840  !
1850  MOVE 26,14
1860  IDRAW 21,0
1870  IDRAW 0,9
1880  !
1890  FOR I=0 TO PI*2 STEP PI/12
1900       IDRAW .2*COS(I),.2*SIN(I)
1910  NEXT I
1920  !
1930  MOVE 21,17
```

```
1940  IDRAW 0,6
1950  !
1960  FOR I=0 TO PI*2 STEP PI/12
1970      IDRAW .2*COS(I),.2*SIN(I)
1980  NEXT I
1990  !
2000  MOVE 21,17
2010  IDRAW 21,0
2020  IDRAW 0,6
2030  !
2040  FOR I=0 TO PI*2 STEP PI/12
2050      IDRAW .2*COS(I),.2*SIN(I)
2060  NEXT I
2070  !
2080  MOVE 16,25
2090  CSIZE 2.3,.5
2100  LABEL "DATA CLOCK1          DATA  CLOCK"
2110  !
2120  INPUT "Are you ready ? Press return key to start.",A
2130  !
2140  RETURN
2150  !
2160  !
2170  END
```

(4) Measurement pattern, BIT WINDOW, and BLOCK WINDOW setting

This program is used to control the characteristics and features of measurement patterns, BIT WINDOW and BLOCK WINDOW.

Select the pattern logic and measurement pattern, then set the settings for the selected pattern.



- Program list

```
10    !*********************************************************************
20    !*                                                                   *
30    !*              MP1762A/MP1764A PATTERN SAMPLE PROGRAM               *
40    !*                                                     ED_PAT        *
50    !*********************************************************************
60    !
70    Add=701                                        !MP1762A/MP1764A ADDRESS
80    !
90    CLEAR Add                                      !DEVICE CLEAR
100   !
110   LOOP
120       CLEAR SCREEN
130       PRINT "** MP1762A/MP1764A PATTERN SAMPLE PROGRAM ** "
140       PRINT
150       !
160       GOSUB Logic_set
170       GOSUB Pattern_set
180       !
190       INPUT "Next data set[Yes:0, No:1]",Loop$
200   EXIT IF Loop$="1"
210   END LOOP
220   !
230   STOP
240   !
250   !****************** MP1762A/MP1764A LOGIC SETTING **********************
260 Logic_set:   !
270   !
280   LOOP
290       INPUT "Choose LOGIC [Positive:0, Negative:1]",Lgc$
300       IF Lgc$<>"0" AND Lgc$<>"1" THEN
310           PRINT "Wrong chosen number!! Please select a correct LOGIC"
320       END IF
330   EXIT IF Lgc$="0" OR Lgc$="1"
340   END LOOP
350   CLEAR SCREEN
360   OUTPUT Add;"LGC "&Lgc$
370   !
380   RETURN
390   !
400   !****************** MP1762A/MP1764A PATTERN SETTING ********************
410 Pattern_set:   !
420   !
430   LOOP
440       INPUT "Choose measure PATTERN [Alternate:0, Data:1, Zero subst:2, PRBS
:3]",Patt$
450       IF Patt$<>"0" AND Patt$<>"1" AND Patt$<>"2" AND Patt$<>"3" THEN
460           PRINT "Wrong chosen number!! Please select a correct PATTERN"
470       END IF
480   EXIT IF Patt$="0" OR Patt$="1" OR Patt$="1" OR Patt$="2" OR Patt$="3"
490   END LOOP
500   CLEAR SCREEN
510   OUTPUT Add;"PTS "&Patt$
520   !
530   SELECT Patt$
540       CASE "0"
550           LOOP
560               INPUT "Choose SYNC MODE [Normal:0, Frame:1]",Sync$
570               IF Sync$<>"0" AND Sync$<>"1" THEN
580                   PRINT "Wrong chosen number!! Please select a correct SYNC
MODE"
590               END IF
```

```
600            EXIT IF Sync$="0" OR Sync$="1"
610            END LOOP
620            CLEAR SCREEN
630            OUTPUT Add;"SYM "&Sync$
640            GOSUB Altn
650        CASE "1"
660            LOOP
670                INPUT "Choose SYNC MODE [Normal:0, Frame:1, Quick:2]",Sync$
680                IF Sync$<>"0" AND Sync$<>"1" AND Sync$<>"2" THEN
690                    PRINT "Wrong chosen number!! Please select a correct SYNC
MODE"
700                END IF
710            EXIT IF Sync$="0" OR Sync$="1" OR Sync$="2"
720            END LOOP
730            CLEAR SCREEN
740            OUTPUT Add;"SYM "&Sync$
750            GOSUB Dat
760        CASE "2"
770            LOOP
780                INPUT "Choose SYNC MODE [Normal:0, Frame:1, Quick:2]",Sync$
790                IF Sync$<>"0" AND Sync$<>"1" AND Sync$<>"2" THEN
800                    PRINT "Wrong chosen number!! Please select a correct SYNC
MODE"
810                END IF
820            EXIT IF Sync$="0" OR Sync$="1" OR Sync$="2"
830            END LOOP
840            CLEAR SCREEN
850            OUTPUT Add;"SYM "&Sync$
860            GOSUB Zer
870        CASE "3"
880            GOSUB Prbs
890            !
900    END SELECT
910    !
920    RETURN
930    !
940    !********************* ALTERNATE PATTERN SETTING ************************
950 Altn:  !
960    LOOP
970        INPUT "Set alternate pattern length [128 to 4194304]",Alt_dln$
980        Alt_dln=VAL(Alt_dln$)
990        IF Alt_dln<128 OR Alt_dln>4194304 THEN
1000           PRINT "Wrong input ALTERNATE LENGTH!! Please set a correct number"
1010       END IF
1020   EXIT IF Alt_dln>=128 AND Alt_dln<=4194304
1030   END LOOP
1040   CLEAR SCREEN
1050   OUTPUT Add;"DLN "&Alt_dln$
1060   !
1070   LOOP
1080       INPUT "Choose ALTERNATE A or B [A:0, B:1] ",Alt_dsp$
1090       IF Alt_dsp$<>"0" AND Alt_dsp$<>"1" THEN
1100           PRINT "Wrong input ALTERNATE DISPLAY!! Please set a correct number
"
1110       END IF
1120   EXIT IF Alt_dsp$="0" OR Alt_dsp$="1"
1130   END LOOP
1140   CLEAR SCREEN
1150   OUTPUT Add;"ALT "&Alt_dsp$
1160   !
1170   LOOP
1180       LOOP
1190           INPUT "Which PATTERN do you set?[Pattern:0, Bit_window:1, Block_wi
ndow:2]",Dsp_sel$
1200           IF Dsp_sel$<>"0" AND Dsp_sel$<>"1" AND Dsp_sel$<>"2" THEN
1210               PRINT "Wrong input setting DISPLAY SELECT!! Please set a corre
```

```
ct number"
1220            END IF
1230        EXIT IF Dsp_sel$="0" OR Dsp_sel$="1" OR Dsp_sel$="2"
1240        END LOOP
1250        CLEAR SCREEN
1260        OUTPUT Add;"DSP "&Dsp_sel$
1270        SELECT Dsp_sel$
1280            CASE "0"
1290                GOSUB Bit_patt
1300            CASE "1"
1310                GOSUB Bit_win
1320            CASE "2"
1330                GOSUB Block_win
1340        END SELECT
1350        INPUT "Do you wish to continue to set another ALTERNATE pattern?[Yes:0
, No:1]",Cont_alt$
1360    EXIT IF Cont_alt$="1"
1370    END LOOP
1380    CLEAR SCREEN
1390    RETURN
1400    !
1410    !************************ DATA PATTERN SETTING ************************
1420 Dat: !
1430    LOOP
1440        INPUT "Set data pattern length [2 TO 8388608]",Dat_dln$
1450        Dat_dln=VAL(Dat_dln$)
1460        IF Dat_dln<2 OR Dat_dln>8388608 THEN
1470            PRINT "Wrong input DATA LENGTH!! Please set a correct number"
1480        END IF
1490    EXIT IF Dat_dln>=2 AND Dat_dln<=8388608
1500    END LOOP
1510    CLEAR SCREEN
1520    OUTPUT Add;"DLN "&Dat_dln$
1530    !
1540    LOOP
1550        LOOP
1560            IF Dat_dln MOD 32=0 THEN
1570                INPUT "Which PATTERN do you set?[Pattern:0, Bit_window:1, Bloc
k_window:2]",Dsp_sel$
1580                Flg=0 !TRUE
1590                IF Dsp_sel$<>"0" AND Dsp_sel$<>"1" AND Dsp_sel$<>"2" THEN
1600                    PRINT "Wrong input setting DISPLAY SELECT!! Please set a c
orrect number"
1610                    Flg=1 !FALSE
1620                END IF
1630            ELSE
1640                INPUT "Which PATTERN do you set?[Pattern:0, Bit_window:1]",Dsp
_sel$
1650                Flg=0 !TRUE
1660                IF Dsp_sel$<>"0" AND Dsp_sel$<>"1" THEN
1670                    PRINT "Wrong input setting DISPLAY SELECT!! Please set a c
orrect number"
1680                    Flg=1 !FALSE
1690                END IF
1700            END IF
1710        EXIT IF Flg=0
1720        END LOOP
1730        CLEAR SCREEN
1740        OUTPUT Add;"DSP "&Dsp_sel$
1750        !
1760        SELECT Dsp_sel$
1770            CASE "0"
1780                GOSUB Bit_patt
1790            CASE "1"
1800                GOSUB Bit_win
1810            CASE "2"
```

```
1820                GOSUB Block_win
1830          END SELECT
1840          INPUT "Do you wish to continue to set another DATA pattern?[Yes:0, No:
1]",Cont_dat$
1850     EXIT IF Cont_dat$="1"
1860     END LOOP
1870     CLEAR SCREEN
1880     RETURN
1890     !
1900     !************************* ZERO SUBST PATTERN SETTING *******************
1910 Zer: !
1920     LOOP
1930          INPUT "Set zero subst pattren [2^7:0, 2^9:1, 2^11:2, 2^15:3]",Zsub_dan
$
1940          IF Zsub_dan$<>"0" AND Zsub_dan$<>"1" AND Zsub_dan$<>"2" AND Zsub_dan$<
>"3" THEN
1950              PRINT "Wrong input ZERO SUBST. PATTERN!! Please set a correct numb
er"
1960          END IF
1970     EXIT IF Zsub_dan$="0" OR Zsub_dan$="1" OR Zsub_dan$="2" OR Zsub_dan$="3"
1980     END LOOP
1990     CLEAR SCREEN
2000     SELECT Zsub_dan$
2010          CASE "0"
2020              OUTPUT Add;"PTN 2"
2030              LOOP
2040                  INPUT "Set zero subst length [1 to 127]",Zsub_len$
2050                  Zsub_len=VAL(Zsub_len$)
2060                  IF Zsub_len<1 OR Zsub_len>127 THEN
2070                      PRINT "Wrong input ZERO SUBST LENGTH!! Please set a correc
t number"
2080                  END IF
2090              EXIT IF Zsub_len>=1 AND Zsub_len<=127
2100              END LOOP
2110              CLEAR SCREEN
2120              OUTPUT Add;"ZLN "&Zsub_len$
2130              !
2140          CASE "1"
2150              OUTPUT Add;"PTN 3"
2160              LOOP
2170                  INPUT "Set zero subst length [1 to 511]",Zsub_len$
2180                  Zsub_len=VAL(Zsub_len$)
2190                  IF Zsub_len<1 OR Zsub_len>511 THEN
2200                      PRINT "Wrong input ZERO SUBST LENGTH!! Please set a correc
t number"
2210                  END IF
2220              EXIT IF Zsub_len>=1 AND Zsub_len<=511
2230              END LOOP
2240              CLEAR SCREEN
2250              OUTPUT Add;"ZLN "&Zsub_len$
2260              !
2270          CASE "2"
2280              OUTPUT Add;"PTN 5"
2290              LOOP
2300                  INPUT "Set zero subst length [1 to 2047]",Zsub_len$
2310                  Zsub_len=VAL(Zsub_len$)
2320                  IF Zsub_len<1 OR Zsub_len>2047 THEN
2330                      PRINT "Wrong input ZERO SUBST LENGTH!! Please set a correc
t number"
2340                  END IF
2350              EXIT IF Zsub_len>=1 AND Zsub_len<=2047
2360              END LOOP
2370              CLEAR SCREEN
2380              OUTPUT Add;"ZLN "&Zsub_len$
2390              !
2400          CASE "3"
```

```
2410            OUTPUT Add;"PTN 6"
2420            LOOP
2430                 INPUT "Set zero subst length [1 to 32767]",Zsub_len$
2440                 Zsub_len=VAL(Zsub_len$)
2450                 IF Zsub_len<1 OR Zsub_len>32767 THEN
2460                      PRINT "Wrong input ZERO SUBST LENGTH!! Please set a correc
t number"
2470                 END IF
2480            EXIT IF Zsub_len>=1 AND Zsub_len<=32767
2490            END LOOP
2500            CLEAR SCREEN
2510            OUTPUT Add;"ZLN "&Zsub_len$
2520            !
2530  END SELECT
2540       !
2550  LOOP
2560       LOOP
2570            INPUT "Which PATTERN do you set?[Bit_window:1, Block_window:2]",Ds
p_sel$
2580            IF Dsp_sel$<>"1" AND Dsp_sel$<>"2" THEN
2590                 PRINT "Wrong input setting DISPLAY SELECT!! Please set a corre
ct number"
2600            END IF
2610       EXIT IF Dsp_sel$="1" OR Dsp_sel$="2"
2620       END LOOP
2630       CLEAR SCREEN
2640       OUTPUT Add;"DSP "&Dsp_sel$
2650       !
2660       SELECT Dsp_sel$
2670            CASE "1"
2680                 GOSUB Bit_win
2690            CASE "2"
2700                 GOSUB Block_win
2710       END SELECT
2720       INPUT "Do you wish to continue to set another ZERO SUBST.pattern?[Yes:
0, No:1]",Cont_zsub$
2730  EXIT IF Cont_zsub$="1"
2740  END LOOP
2750  CLEAR SCREEN
2760  RETURN
2770  !
2780  !************************* PRBS PATTERN SETTING *************************
*
2790 Prbs:  !
2800  LOOP
2810       LOOP
2820            PRINT "Select PRBS pattern [2^7-1:0, 2^9-1:1, 2^11-1:2, 2^15-1:3]"
2830            PRINT "                    [2^20-1:4, 2^23-1:5, 2^31-1:6          ]"
2840            PRINT
2850            INPUT "Set PRBS pattren ",Prbs_dan$
2860            IF Prbs_dan$<>"0" AND Prbs_dan$<>"1" AND Prbs_dan$<>"2" AND Prbs_d
an$<>"3" AND Prbs_dan$<>"4" AND Prbs_dan$<>"5" AND Prbs_dan$<>"6" THEN
2870                 PRINT "Wrong input PRBS PATTERN!! Please set a correct number"
2880            END IF
2890       EXIT IF Prbs_dan$="0" OR Prbs_dan$="1" OR Prbs_dan$="2" OR Prbs_dan$="
3" OR Prbs_dan$="4" OR Prbs_dan$="5" OR Prbs_dan$="6"
2900       END LOOP
2910       CLEAR SCREEN
2920       SELECT Prbs_dan$
2930            CASE "0"
2940                 OUTPUT Add;"PTN 2"
2950            CASE "1"
2960                 OUTPUT Add;"PTN 3"
2970            CASE "2"
2980                 OUTPUT Add;"PTN 5"
2990            CASE "3"
```

```
3000                OUTPUT Add;"PTN 6"
3010            CASE "4"
3020                OUTPUT Add;"PTN 7"
3030            CASE "5"
3040                OUTPUT Add;"PTN 8"
3050            CASE "6"
3060                OUTPUT Add;"PTN 9"
3070        END SELECT
3080        !
3090        LOOP
3100            PRINT
3110            PRINT "Mark ratio (Positive)[0/8:0, 1/8:1, 1/4:2, 1/2:3]"
3120            PRINT "            (Negative)[8/8:0, 7/8:1, 3/4:2, 1/2:3]"
3130            PRINT
3140            INPUT "Choose MARK RATIO",Mrk$
3150            IF Mrk$<>"0" AND Mrk$<>"1" AND Mrk$<>"2" AND Mrk$<>"3" THEN
3160               PRINT "Wrong input setting PRBS MARK RATIO!! Please set a corre
ct number"
3170            END IF
3180        EXIT IF Mrk$="0" OR Mrk$="1" OR Mrk$="2" OR Mrk$="3"
3190        END LOOP
3200        CLEAR SCREEN
3210        OUTPUT Add;"MRK "&Mrk$
3220        !
3230        LOOP
3240            INPUT "Do you wish to set BIT-WINDOW PATTERN?[Yes:0, No:1]",Dsp_se
l$
3250            IF Dsp_sel$<>"0" AND Dsp_sel$<>"1" THEN
3260                PRINT "Wrong input setting DISPLAY SELECT!! Please set a corre
ct number"
3270            END IF
3280        EXIT IF Dsp_sel$="0" OR Dsp_sel$="1"
3290        END LOOP
3300        CLEAR SCREEN
3310        !
3320        SELECT Dsp_sel$
3330            CASE "0"
3340                OUTPUT Add;"DSP 1"
3350                GOSUB Bit_win
3360        END SELECT
3370        INPUT "Do you wish to continue to set another PRBS pattern?[Yes:0, No:
1]",Cont_prbs$
3380    EXIT IF Cont_prbs$="1"
3390    END LOOP
3400    CLEAR SCREEN
3410    RETURN
3420    !
3430    !************************** BIT PATTERN SETTING **************************
3440 Bit_patt: !
3450    !
3460    DIM Bit$[255]
3470    LOOP
3480        PRINT "You aer able to choice data format of HEXadecimal or DECimal"
3490        PRINT "Default data format is HEXdecimal"
3500        INPUT "Which do you choice foramt [HEXdecimal:0, DECimal:1]",Fmt$
3510        INPUT "Where do you set start page[1 to LENGTH/16]",Page$
3520        !
3530        Bit$=""
3540        A$="0"
3550        FOR K=0 TO 7
3560            PRINT "<Do you set bit-pattern of = "&VAL$(VAL(Page$)+K)&" PAGE? [
Yes:0, No:1]"
3570            INPUT "",A$
3580            IF A$="1" THEN
3590                IF K=0 THEN
3600                    GOTO Jump_out
```

```
3610                 ELSE
3620                     GOTO Jump
3630                 END IF
3640             END IF
3650             PRINT
3660             IF K<>0 THEN
3670                 Bit$=Bit$&","
3680             END IF
3690             IF Fmt$="1" THEN
3700                 LOOP
3710                     PRINT "Enter "&VAL$(VAL(Page$)+K)&"PAGE pattern [ 0 to 655
35 ]"
3720                     INPUT B$
3730                     B=VAL(B$)
3740                 EXIT IF B>=0 AND B<=65535
3750                     PRINT "Wrong number!!Please input a correct number"
3760                 END LOOP
3770                 CLEAR SCREEN
3780             ELSE
3790                 LOOP
3800                     PRINT "Enter "&VAL$(VAL(Page$)+K)&"PAGE pattern [ 0 to FFF
F ]"
3810                     INPUT B$
3820                     B=DVAL(B$,16)
3830                     B$="#H"&B$
3840                 EXIT IF B>=0 AND B<=65535
3850                     PRINT "Wrong number!!Please input a correct number"
3860                 END LOOP
3870                 CLEAR SCREEN
3880             END IF
3890             Bit$=Bit$&B$
3900         NEXT K
3910 Jump: !
3920         PRINT
3930         PRINT "You set data is :PAG "&Page$&"; BIT "&Bit$
3940         OUTPUT Add;"PAG "&Page$&";BIT "&Bit$
3950         !
3960 Jump_out:!
3970         INPUT "Do you want to continue setting BIT-PATTERN? [Yes:0, No:1]",Loo
p$
3980     EXIT IF Loop$="1"
3990     CLEAR SCREEN
4000     END LOOP
4010     CLEAR SCREEN
4020     RETURN
4030     !
4040     !************************** BIT WINDOW SETTING ***************************
4050 Bit_win: !
4060     !
4070     DIM Bit_win$[255]
4080     LOOP
4090         PRINT "You aer able to choice data format of HEXadecimal or DECimal"
4100         PRINT "Default data format is HEXdecimal"
4110         INPUT "Which do you choice foramt [HEXdecimal:0, DECimal:1]",Fmt$
4120         INPUT "Where do you set start page[1 to 2]",Page$
4130         IF Page$<>"1" AND Page$<>"2" THEN
4140             GOTO Jump1
4150         END IF
4160         !
4170         Bit_win$=""
4180         A$="0"
4190         FOR K=0 TO 1
4200             IF VAL(Page$)+K>2 THEN
4210                 GOTO Jump1
4220             END IF
4230             PRINT "<Do you set bit-WINDOW-pattern of = "&VAL$(VAL(Page$)+K)&"
```

```
PAGE? [Yes:0, No:1]"
4240            INPUT A$
4250            IF A$="1" THEN
4260                IF K=0 THEN
4270                    GOTO Jump_out1
4280                ELSE
4290                    GOTO Jump1
4300                END IF
4310            END IF
4320            PRINT
4330            IF K<>0 THEN
4340                Bit_win$=Bit_win$&","
4350            END IF
4360            IF Fmt$="1" THEN
4370                LOOP
4380                    PRINT "Enter "&VAL$(VAL(Page$)+K)&"PAGE BIT-WINDOW pattern
 [ 0 to 65535 ]"
4390                    INPUT B$
4400                    B=VAL(B$)
4410                    PRINT B
4420                EXIT IF B>=0 OR B<=65536
4430                    PRINT "Wrong number!!Please input a correct number"
4440                END LOOP
4450                CLEAR SCREEN
4460            ELSE
4470                LOOP
4480                    PRINT "Enter "&VAL$(VAL(Page$)+K)&"PAGE BIT-WINDOW pattern
 [ 0 to FFFF ]"
4490                    INPUT B$
4500                    PRINT B$
4510                    B=DVAL(B$,16)
4520                    PRINT B
4530                    B$="#H"&B$
4540                EXIT IF B>=0 AND B<=65535
4550                END LOOP
4560                CLEAR SCREEN
4570            END IF
4580            Bit_win$=Bit_win$&B$
4590        NEXT K
4600 Jump1: !
4610        PRINT
4620        PRINT "You set data is :MSK "&Page$&"; CHM "&Bit_win$
4630        OUTPUT Add;"MSK "&Page$&";CHM "&Bit_win$
4640        !
4650 Jump_out1:!
4660        LOOP
4670            INPUT "Do you wish to set BIT-WINDOW ON/OFF[OFF:0, ON:1]",Ena$
4680        EXIT IF Ena$="0" OR Ena$="1"
4690            PRINT "Wrong number!!Please input a correct number"
4700        END LOOP
4710        CLEAR SCREEN
4720        OUTPUT Add;"MSE "&Ena$
4730        INPUT "Do you want to continue setting BIT-WINDOW-PATTERN? [Yes:0, No:
1]",Loop$
4740   EXIT IF Loop$="1"
4750   CLEAR SCREEN
4760   END LOOP
4770   CLEAR SCREEN
4780   RETURN
4790   !
4800   !************************** BLOCK WINDOW SETTING ************************
******
4810 Block_win: !
4820   !
4830   DIM Block_win$[255]
4840   LOOP
```

```
4850        PRINT "You aer able to choice data format of HEXadecimal or DECimal"
4860        PRINT "Default data format is HEXdecimal"
4870        INPUT "Which do you choice foramt [HEXdecimal:0, DECimal:1]",Fmt$
4880        INPUT "Where do you set start page[1 to LENGTH/16]",Page$
4890        !
4900        Block_win$=""
4910        A$="0"
4920        FOR K=0 TO 7
4930            PRINT "<Do you set BLOCK-WINDOW-pattern of = "&VAL$(VAL(Page$)+K)&
" PAGE? [Yes:0, No:1]"
4940            INPUT A$
4950            IF A$="1" THEN
4960                IF K=0 THEN
4970                    GOTO Jump_out2
4980                ELSE
4990                    GOTO Jump2
5000                END IF
5010            END IF
5020            PRINT
5030            IF K<>0 THEN
5040                Block_win$=Block_win$&","
5050            END IF
5060            IF Fmt$="1" THEN
5070                LOOP
5080                    PRINT "Enter "&VAL$(VAL(Page$)+K)&"PAGE BLOCK-WINDOW patte
rn [ 0 to 65535 ]"
5090                    INPUT B$
5100                    B=VAL(B$)
5110                EXIT IF B<0 OR B>65535
5120                    PRINT "Wrong number!!Please input a correct number"
5130                END LOOP
5140                CLEAR SCREEN
5150            ELSE
5160                LOOP
5170                    PRINT "Enter "&VAL$(VAL(Page$)+K)&"PAGE BLOCK-WINDOW patte
rn [ 0 to FFFF ]"
5180                    INPUT B$
5190                    B=DVAL(B$,16)
5200                    B$="#H"&B$
5210                EXIT IF B>=0 AND B<=65535
5220                END LOOP
5230                CLEAR SCREEN
5240            END IF
5250            Block_win$=Block_win$&B$
5260        NEXT K
5270 Jump2: !
5280        PRINT
5290        PRINT "You set data is :PAG "&Page$&"; MGB "&Block_win$
5300        OUTPUT Add;"PAG "&Page$&";MGB "&Block_win$
5310        !
5320 Jump_out2:!
5330        LOOP
5340            INPUT "Do you wish to set BLOCK-WINDOW ON/OFF[OFF:0, ON:1]",Ena$
5350        EXIT IF Ena$="0" OR Ena$="1"
5360            PRINT "Wrong number!!Please input a correct number"
5370        END LOOP
5380        CLEAR SCREEN
5390        INPUT "Do you want to continue setting BLOCK-WINDOW-PATTERN? [Yes:0, N
o:1]",Loop$
5400    EXIT IF Loop$="1"
5410    END LOOP
5420    CLEAR SCREEN
5430    RETURN
5440    !
5450    PAUSE
5460    END
```

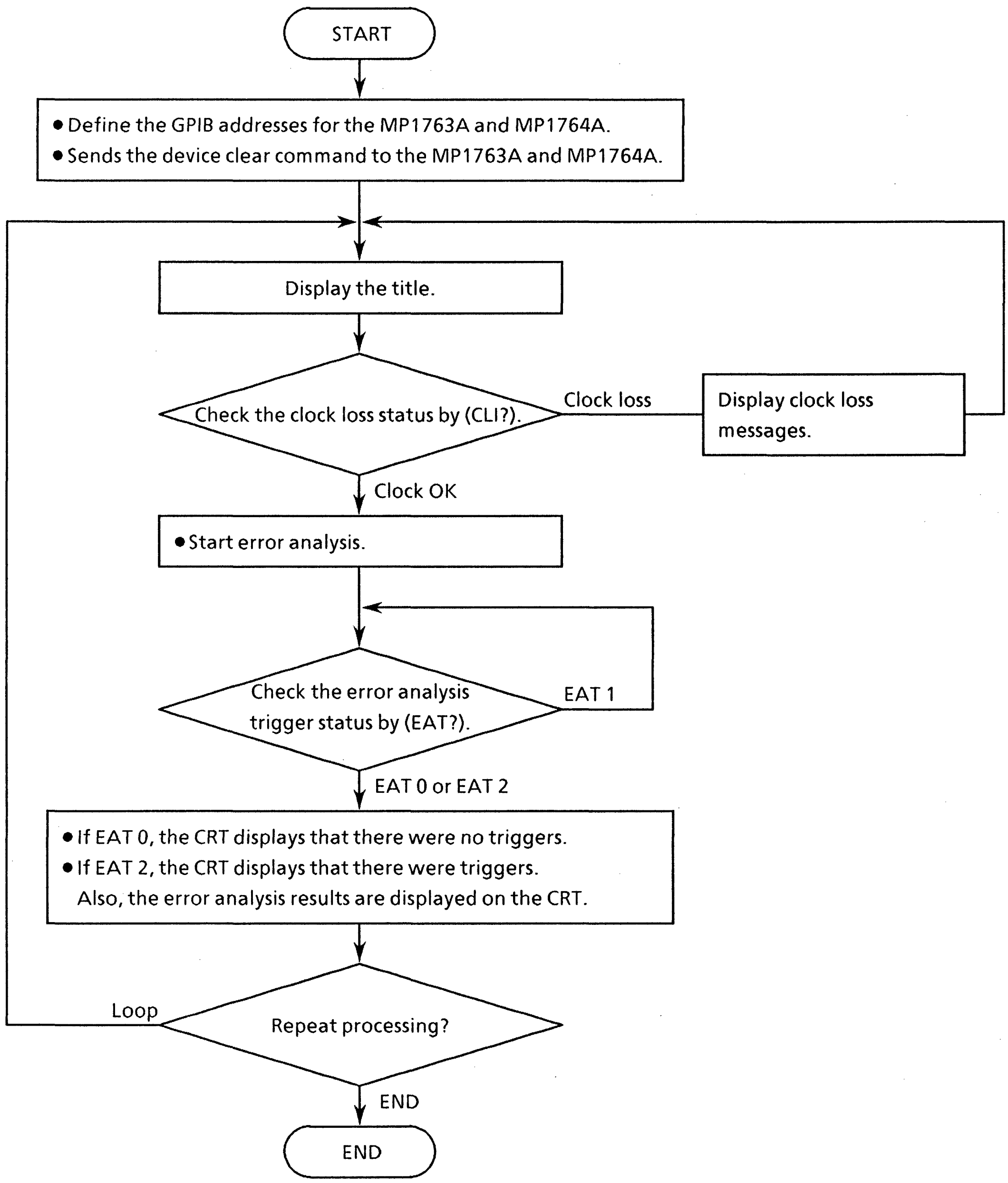
(5) Error analysis

This program executes the error analysis function.

Confirm that the clock is not lost because error analysis becomes invalid if the clock is lost. Start the error analysis.

Read by using the request command (EAT?) to check if the error analysis is being executed, and display them on the CRT.

Display the error analysis results on the CRT.



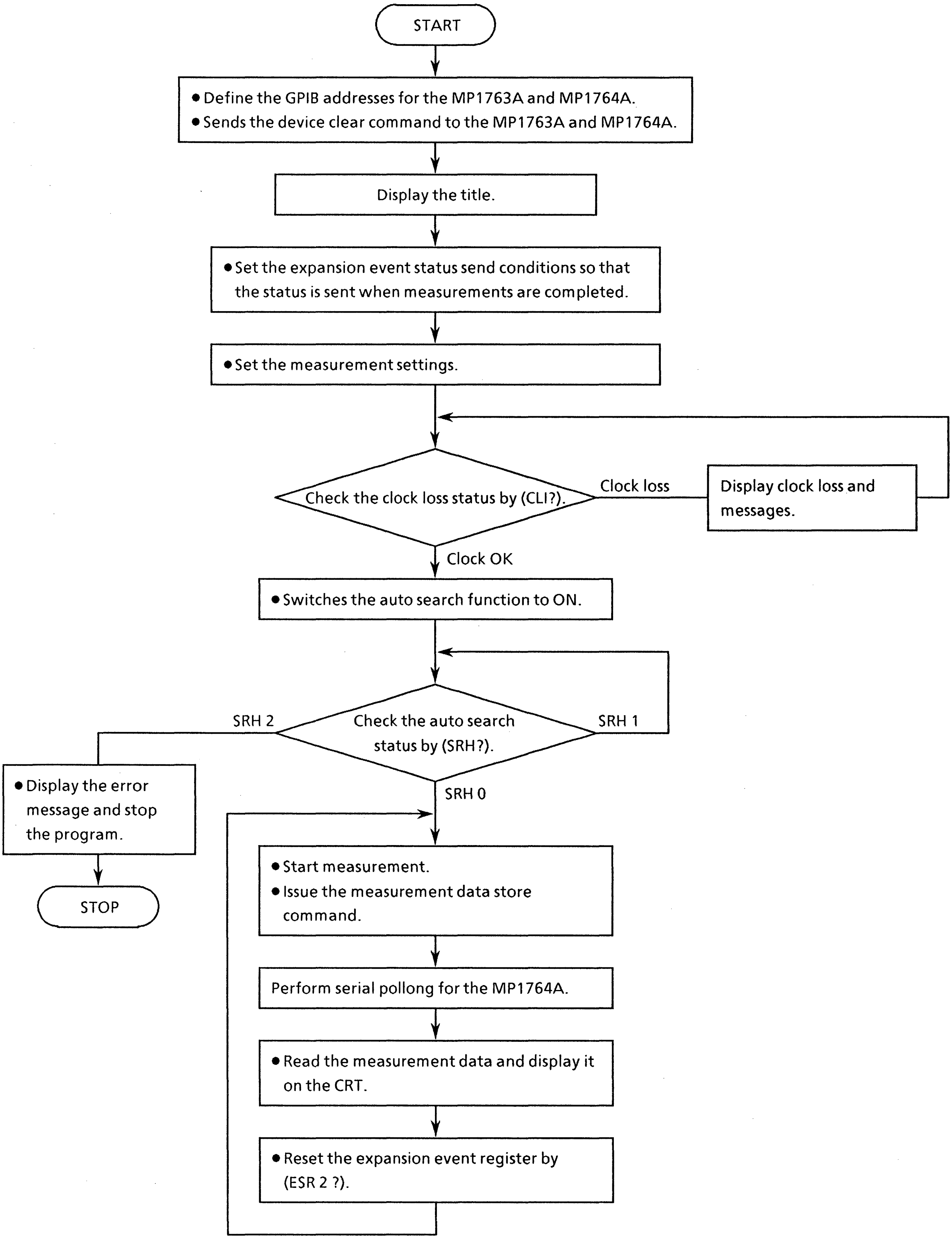
- Program list

```
10    !*********************************************************************
20    !*                                                                   *
30    !*             MP1762A/MP1764A ERROR ANALYSIS PROGRAM               *
40    !*                                                     ED_ANA       *
50    !*********************************************************************
60    !
70    Add=701                                   !MP1762A/MP1764A ADDRESS
80    CLEAR Add                                 !DEVICE CLEAR(ED)
90    !
100   LOOP
110       !
120       CLEAR SCREEN
130       PRINT "** MP1762A/MP1764A ERROR ANALYSIS SAMPLE PROGRAM ** "
140       PRINT
150       !
160       GOSUB Clock                            !CHECK CLOCK LOSS
170       GOSUB Trig                             !ERROR ANALYSIS TRIGGER
180       GOSUB Result                           !DISPLAY RESULT
190       !
200       INPUT "    Try again?[Yes:0, No:1]",Loop$
210   EXIT IF Loop$="1"
220   END LOOP
230   !
240   STOP
250   !
260   !******************** Check Clock loss ***************************
270 Clock:    !
280           !
290   LOOP
300       !
310       OUTPUT Add;"CLI?"                      !CHECK CLOCK LOSS
320       ENTER Add;Cli$
330       IF Cli$="CLI 1" THEN
340           PRINT "******** CLOCK LOSS ********* "
350           WAIT .5
360           CLEAR SCREEN
370       END IF
380   EXIT IF Cli$="CLI 0"
390   END LOOP
400   !
410   RETURN
420   !
430   !
440   !******************** ERROR ANALYSIS TRIGGER *********************
450 Trig:   !
460   !
470   OUTPUT Add;"EAT 1"
480   !
490   LOOP
500       !
510       OUTPUT Add;"EAT?"
520       ENTER Add;Eat$
530       !
540       IF Eat$="EAT 1" THEN
550           PRINT "AWITTING TRIGGER!!"
560           WAIT .5
570           CLEAR SCREEN
580       END IF
590   EXIT IF Eat$="EAT 0" OR Eat$="EAT 2"
600   END LOOP
610   !
```

```
620    IF Eat$="EAT 0" THEN
630        PRINT "******** NO TRRIGER FOUND ********* "
640    ELSE
650        PRINT "*********** TRIGGERD ************** "
660    END IF
670    !
680    RETURN
690    !
700    !
710    !********************** DISPLAY RESULT FUNCTION *********************
720 Result:  !
730    !
740    DIM Ana$(16)[255]
750    CSIZE 3,.7
760    MOVE 20,90
770    LABEL "<< ERROR ANALYSIS RESULT >> "
780    !
790    CSIZE 3,.5
800    MOVE 0,85
810    LABEL "<< PAGE >>"
820    !
830    FOR J=1 TO 16
840        CSIZE 3,.5
850        MOVE 0,(80-J*4)
860        LABEL "        "&VAL$(J)
870    NEXT J
880    !
890    CSIZE 3,.5
900    MOVE 20,85
910    LABEL "<< MONITOR >>"
920    !
930    CSIZE 3,.5
940    MOVE 50,85
950    LABEL "<<   ERROR ANALYSIS DATA   >>"
960    !
970    CSIZE 3,.5
980    MOVE 50,80
990    LABEL " BIT16                BIT1"
1000   !
1010   FOR I=1 TO 16
1020       OUTPUT Add;"EAP "&VAL$(I)
1030       OUTPUT Add;"EAB?"
1040       ENTER Add;Ana$(I)
1050       CSIZE 3,.5
1060       MOVE 20,(80-I*4)
1070       LABEL Ana$(I)[12,20]
1080       !
1090       CSIZE 3,.7
1100       MOVE 52,(80-I*4)
1110       LABEL Ana$(I)[24,41]
1120       !
1130   NEXT I
1140   !
1150   RETURN
1160   !
1170   !
1180   END
```

(6) Measurement results display (displays the measurement results using serial polling)

This program displays the measurement results on the CRT using serial polling.



- Program list

```
10    !************************************************************************
20    !*                                                                      *
30    !*           MP1762A/MP1764A MEASUREMENT RESULT SAMPLE PROGRAM          *
40    !*                                                     ED_MEAS1         *
50    !************************************************************************
60    !
70    Add=701                                       !MP1762A/MP1764A ADDRESS
80    CLEAR Add                                     !DEVICE CLEAR(ED)
90    !
100   LOOP
110       !
120       CLEAR SCREEN
130       PRINT "** MP1762A/MP1764A MEASUREMENT RESULT SAMPLE PROGRAM ** "
140       PRINT
150       !
160       OUTPUT Add;"STO"
170       GOSUB Status_set                          !STATUS RESISTOR SET
180       GOSUB Meas_cnd                            !MEAS.CONDITION SET
190       GOSUB Closs                               !CHECK CLOCK LOSS
200       GOSUB Auto_srh                            !AUTO SEARCH ON
210       GOSUB Polling                             !SERIAL POLE
220       !
230       INPUT "   Try again?[Yes:0, No:1]",Loop$
240   EXIT IF Loop$="1"
250   END LOOP
260   !
270   STOP
280   !
290   !********************** STATUS SETTING *******************************
300 Status_set:!
310   !
320   OUTPUT Add;"*SRE 4"                           !ESR2 ENABLE
330   OUTPUT Add;"ESE2 1"                           !MEAS. END
340   !
350   RETURN
360   !
370   !***************** MEASUREMENT CONDITION SET ************************
380 Meas_cnd:  !
390   !
400   DIM Prd$[255]
410   LOOP
420       !
430       LOOP
440           INPUT "MEAS.MODE?[Repeat:0, Single:1,Untimed:2]",M_mode$
450       EXIT IF M_mode$="0" OR M_mode$="1" OR M_mode$="2"
460           PRINT "Wrong chosen number!! Please select a correct MEAS.MODE"
470       END LOOP
480       OUTPUT Add;"MOD "&M_mode$
490       CLEAR SCREEN
500       !
510       IF M_mode$<>"2" THEN
520           INPUT "MEAS.TIME=[DAY,HOUR,MINUTE,SECOND]",Prd1$,Prd2$,Prd3$,Prd4$
530       END IF
540       OUTPUT Add;"PRD "&Prd1$&","&Prd2$&","&Prd3$&","&Prd4$
550       !LEAR SCREEN
560       !
570       LOOP
580           INPUT "AUTO SYNC CONDITION=[OFF:0, ON:1]",Auto_sync$
590       EXIT IF Auto_sync$="0" OR Auto_sync$="1"
600           PRINT "Wrong chosen number!! Please select a correct AUTO SYNC CON
DITON"
```

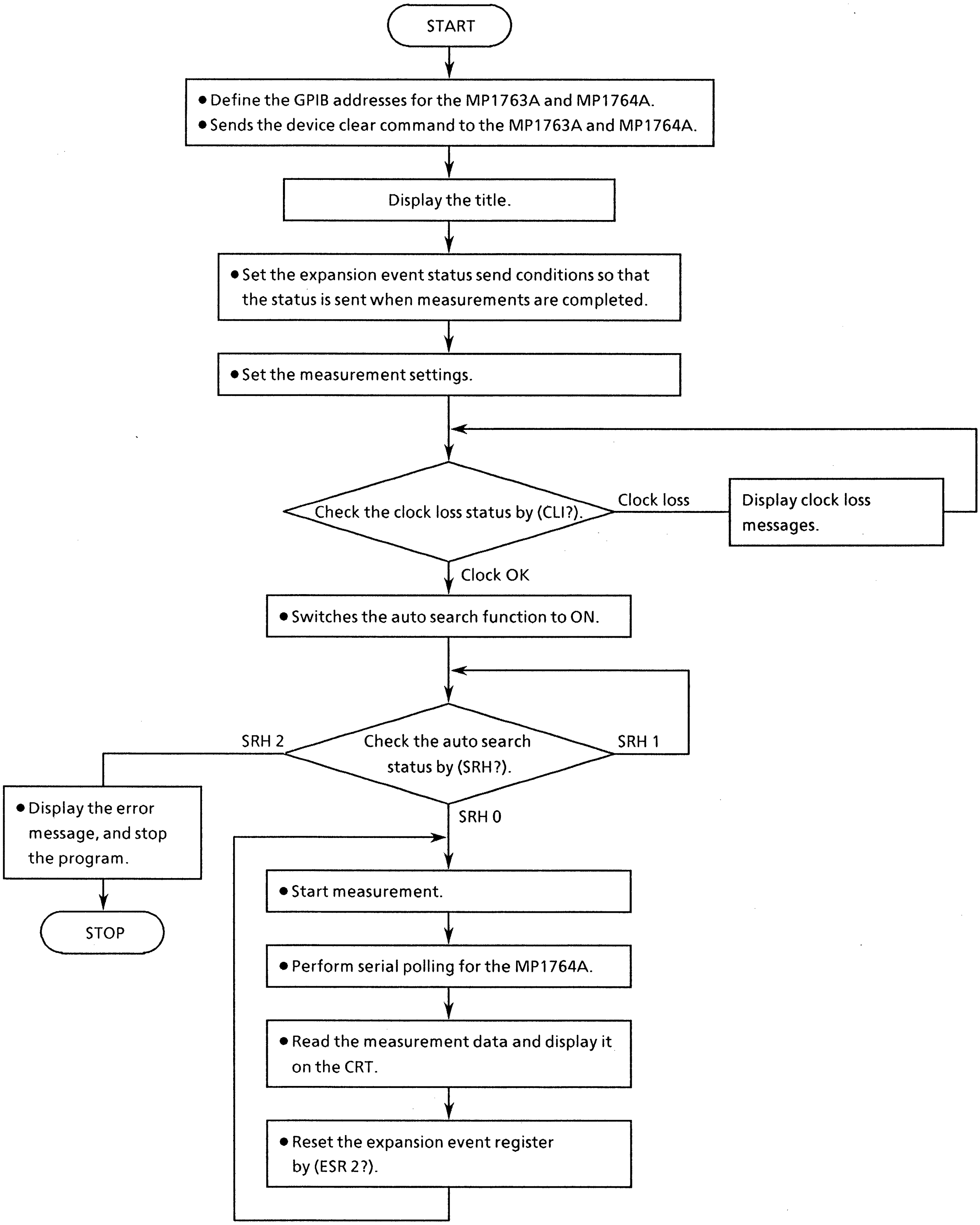
```
610         END LOOP
620         OUTPUT Add;"SYN "&Auto_sync$
630         CLEAR SCREEN
640         !
650         IF Auto_sync$="1" THEN
660              LOOP
670                   PRINT "SYNC THRESHOLD=[1E-2:0, 1E-3:1, 1E-4:2]"
680                   PRINT "                     [1E-5:3, 1E-6:4, 1E-7:5]"
690                   PRINT "                     [IE-8:6,        , INT :8]"
700                   PRINT
710                   INPUT Sync_th$
720              EXIT IF Sync_th$="0" OR Sync_th$="1" OR Sync_th$="2" OR Sync_th$="
3" OR Sync_th$="4" OR Sync_th$="5" OR Sync_th$="6" OR Sync_th$="8"
730                   PRINT "Wrong chosen number!! Please select a correct SYNC THRE
SHOLD"
740              END LOOP
750         OUTPUT Add;"SYE "&Sync_th$
760         CLEAR SCREEN
770         END IF
780         INPUT "Do you change meas.condition?[Yes:0, No:1]",M_cond$
790    EXIT IF M_cond$="1"
800    END LOOP
810    CLEAR SCREEN
820    RETURN
830    !
840    !
850    !************************ CHECK CLOCK LOSS ************************
860 Closs:!
870    !
880    LOOP
890         OUTPUT Add;"CLI?"
900         ENTER Add;Cli$
910         IF Cli$="CLI 1" THEN
920              PRINT "** CLOCK LOSS **"
930              WAIT .5
940         END IF
950         CLEAR SCREEN
960    EXIT IF Cli$="CLI 0"
970    END LOOP
980    !
990    RETURN
1000   !
1010   !********************* AUTO SEARCH FUNCTION **********************
1020 Auto_srh:!
1030   !
1040   OUTPUT Add;"SRH 1"
1050   !
1060   LOOP
1070        OUTPUT Add;"SRH?"
1080        ENTER Add;Srh$
1090        IF Srh$="SRH 1" THEN
1100             PRINT "** SEARCHING **"
1110             WAIT .5
1120        END IF
1130        CLEAR SCREEN
1140   EXIT IF Srh$="SRH 0" OR Srh$="SRH 2"
1150   END LOOP
1160   !
1170   IF Srh$="SRH 2" THEN
1180        PRINT "Failed in auto search!! Program STOP!!"
1190        STOP
1200   END IF
1210   RETURN
1220   !
1230   !
1240   !***************************** POLLING ****************************
```

```
1250 Polling:!
1260  OUTPUT Add;"MOD?"
1270  ENTER Add;Mod$
1280  IF Mod$="MOD 0" THEN
1290       OUTPUT Add;"ESR2?"
1300       ENTER Add;Esr2$
1310       OUTPUT Add;"STA"
1320  END IF
1330  !
1340  LOOP
1350       IF Mod$<>"MOD 0" THEN
1360            OUTPUT Add;"STA"
1370       END IF
1380       !
1390       OUTPUT Add;"EDS"
1400       WAIT .1
1410       LOOP
1420            A=SPOLL(Add)
1430       EXIT IF BIT(A,2)=1
1440            WAIT .1
1450       END LOOP
1460       GOSUB Result_read
1470       OUTPUT Add;"ESR2?"
1480       ENTER Add;Esr2$
1490  END LOOP
1500  !
1510  RETURN
1520  !
1530  !********************** DISPLAY RESULT FUNCTION ********************
1540 Result_read: !
1550  !
1560  DIM Tme$[255],Arm$(3)[255],Err$(4)[255]
1570  DIM Arm_dat$[255],Err_dat$[255]
1580  !
1590  OUTPUT Add;"END? 0,0"
1600  ENTER Add;Tme$
1610  OUTPUT Add;"END? 1,0"
1620  ENTER Add;Arm_dat$
1630  OUTPUT Add;"END? 2,0"
1640  ENTER Add;Err_dat$
1650  Arm$(1)=""
1660  Arm$(2)=""
1670  Arm$(3)=""
1680  K=1
1690  IF Arm_dat$<>"ERR" THEN
1700       Max_len=LEN(Arm_dat$)
1710       FOR J=1 TO 3
1720            LOOP
1730            EXIT IF K=(Max_len+1) OR Arm_dat$[K,K]=","
1740                 Arm$(J)=Arm$(J)&Arm_dat$[K,K]
1750                 K=K+1
1760            END LOOP
1770            K=K+1
1780       NEXT J
1790  ELSE
1800       FOR J=1 TO 3
1810            Arm$(J)=" NO DATA   "
1820       NEXT J
1830  END IF
1840  !
1850  Err$(1)=""
1860  Err$(2)=""
1870  Err$(3)=""
1880  Err$(4)=""
1890  L=1
1900  IF Err_dat$<>"ERR" THEN
```

```
1910        Max_len2=LEN(Err_dat$)
1920        FOR M=1 TO 4
1930            LOOP
1940            EXIT IF Err_dat$[L,L]="," OR L=(Max_len2+1)
1950                Err$(M)=Err$(M)&Err_dat$[L,L]
1960                L=L+1
1970            END LOOP
1980            L=L+1
1990        NEXT M
2000    ELSE
2010        FOR M=1 TO 4
2020            Err$(M)=" NO DATA "
2030        NEXT M
2040    END IF
2050    !
2060    !
2070    IF Tme$="ERR" THEN
2080        GOTO Jump
2090    END IF
2100    !
2110    PRINT " << START TIME >>   "&Tme$[1,17]&"  << STOP TIME >>   "&Tme$[19,35]
2120    !
2130    PRINT " << ARLAM DATA >>  "
2140    PRINT " << POWER FAIL INTVL>> "&Arm$(1)
2150    PRINT " << CLOCK LOSS INTVL>> "&Arm$(2)
2160    PRINT " << SYNC LOSS INTVL >> "&Arm$(3)
2170    !
2180    PRINT " << ERROR DATA >>"
2190    PRINT " << ERROR RATIO >>      "&Err$(1)
2200    PRINT " << ERROR COUNT      >> "&Err$(2)
2210    PRINT " << EI               >> "&Err$(3)
2220    PRINT " << %EFI             >>  "&Err$(4)
2230    !
2240    PRINT
2250   Jump: !
2260    RETURN
2270    !
2280    !
2290    END
```

(7) Measurement results display (Display the measurement results by the request command)

This program displays the measurement results data by using the request command.



- Program list

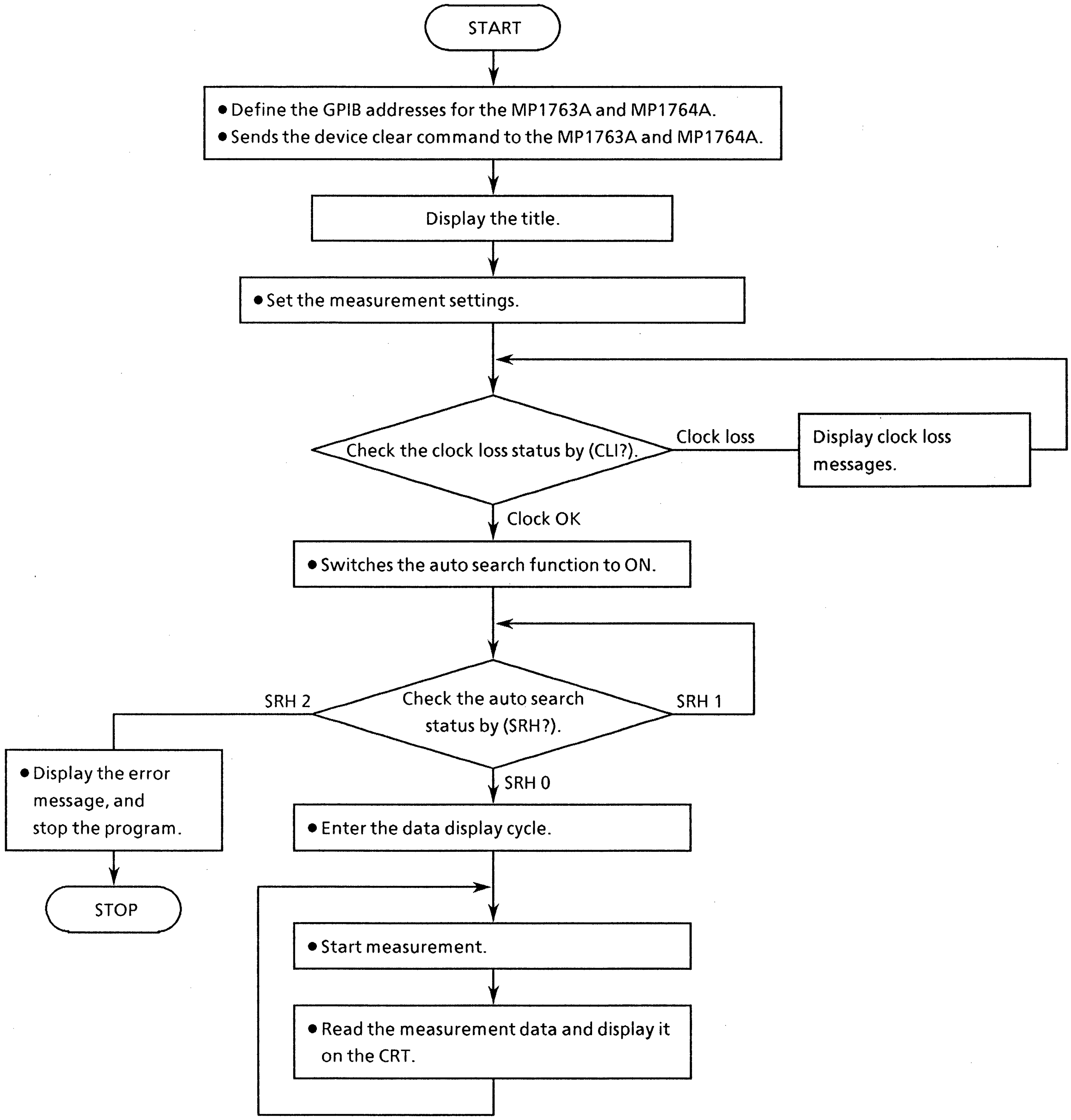
```
10     !************************************************************************
20     !*                                                                      *
30     !*           MP1762A/MP1764A MEASUREMENT RESULT SAMPLE PROGRAM          *
40     !*                                                          ED_MEAS2    *
50     !************************************************************************
60     !
70     Add=701                                     !MP1762A/MP1764A ADDRESS
80     CLEAR Add                                   !DEVICE CLEAR(ED)
90     !
100    LOOP
110        !
120        CLEAR SCREEN
130        PRINT "** MP1762A/MP1764A MEASUREMENT RESULT SAMPLE PROGRAM ** "
140        PRINT
150        !
160        OUTPUT Add;"STO"
170        GOSUB Status_set                        !STATUS RESISTOR SET
180        GOSUB Meas_cnd                          !MEAS.CONDITION SET
190        GOSUB Closs                             !CHECK CLOCK LOSS
200        GOSUB Auto_srh                          !AUTO SEARCH ON
210        GOSUB Polling                           !SERIAL POLE
220        !
230        INPUT "   Try again?[Yes:0, No:1]",Loop$
240    EXIT IF Loop$="1"
250    END LOOP
260    !
270    STOP
280    !
290    !********************** STATUS SETTING *****************************
300 Status_set:!
310    !
320    OUTPUT Add;"*SRE 4"                         !ESR2 ENABLE
330    OUTPUT Add;"ESE2 1"                         !MEAS. END
340    !
350    RETURN
360    !
370    !**************** MEASUREMENT CONDITION SET ***********************
380 Meas_cnd:  !
390    !
400    DIM Prd$[255]
410    LOOP
420        !
430        LOOP
440            INPUT "MEAS.MODE?[Repeat:0, Single:1,Untimed:2]",M_mode$
450        EXIT IF M_mode$="0" OR M_mode$="1" OR M_mode$="2"
460            PRINT "Wrong chosen number!! Please select a correct MEAS.MODE"
470        END LOOP
480        OUTPUT Add;"MOD "&M_mode$
490        CLEAR SCREEN
500        !
510        IF M_mode$<>"2" THEN
520            INPUT "MEAS.TIME=[DAY,HOUR,MINUTE,SECOND]",Prd1$,Prd2$,Prd3$,Prd4$
530        END IF
540        OUTPUT Add;"PRD "&Prd1$&","&Prd2$&","&Prd3$&","&Prd4$
550        !LEAR SCREEN
560        !
570        LOOP
580            INPUT "AUTO SYNC CONDITION=[OFF:0, ON:1]",Auto_sync$
590        EXIT IF Auto_sync$="0" OR Auto_sync$="1"
600            PRINT "Wrong chosen number!! Please select a correct AUTO SYNC CON
DITON"
```

```
610         END LOOP
620         OUTPUT Add;"SYN "&Auto_sync$
630         CLEAR SCREEN
640         !
650         IF Auto_sync$="1" THEN
660             LOOP
670                 PRINT "SYNC THRESHOLD=[1E-2:0, 1E-3:1, 1E-4:2]"
680                 PRINT "                      [1E-5:3, 1E-6:4, 1E-7:5]"
690                 PRINT "                      [IE-8:6,         , INT :8]"
700                 PRINT
710                 INPUT Sync_th$
720             EXIT IF Sync_th$="0" OR Sync_th$="1" OR Sync_th$="2" OR Sync_th$="3" OR Sync_th$="4" OR Sync_th$="5" OR Sync_th$="6" OR Sync_th$="8"
730                 PRINT "Wrong chosen number!! Please select a correct SYNC THRESHOLD"
740             END LOOP
750         OUTPUT Add;"SYE "&Sync_th$
760         CLEAR SCREEN
770         END IF
780         INPUT "Do you change meas.condition?[Yes:0, No:1]",M_cond$
790     EXIT IF M_cond$="1"
800     END LOOP
810     CLEAR SCREEN
820     RETURN
830     !
840     !
850     !************************ CHECK CLOCK LOSS ************************
860 Closs:!
870     !
880     LOOP
890         OUTPUT Add;"CLI?"
900         ENTER Add;Cli$
910         IF Cli$="CLI 1" THEN
920             PRINT "** CLOCK LOSS **"
930             WAIT .5
940         END IF
950         CLEAR SCREEN
960     EXIT IF Cli$="CLI 0"
970     END LOOP
980     !
990     RETURN
1000    !
1010    !********************** AUTO SEARCH FUNCTION ***********************
1020 Auto_srh:!
1030    !
1040    OUTPUT Add;"SRH 1"
1050    !
1060    LOOP
1070        OUTPUT Add;"SRH?"
1080        ENTER Add;Srh$
1090        IF Srh$="SRH 1" THEN
1100            PRINT "** SEARCHING **"
1110            WAIT .5
1120        END IF
1130        CLEAR SCREEN
1140    EXIT IF Srh$="SRH 0" OR Srh$="SRH 2"
1150    END LOOP
1160    !
1170    IF Srh$="SRH 2" THEN
1180        PRINT "Failed in auto search!! Program STOP!!"
1190        STOP
1200    END IF
1210    RETURN
1220    !
1230    !
1240    !***************************** POLLING ***************************
```

```
1250 Polling: !
1260  OUTPUT Add;"MOD?"
1270  ENTER Add;Mod$
1280  IF Mod$="MOD 0" THEN
1290       OUTPUT Add;"ESR2?"
1300       ENTER Add;Esr2$
1310       OUTPUT Add;"STA"
1320  END IF
1330  !
1340  LOOP
1350       IF Mod$<>"MOD 0" THEN
1360            OUTPUT Add;"STA"
1370       END IF
1380       !
1390       OUTPUT Add;"EDS"
1400       WAIT .1
1410       LOOP
1420            A=SPOLL(Add)
1430       EXIT IF BIT(A,2)=1
1440            WAIT .1
1450       END LOOP
1460       GOSUB Result_read
1470       OUTPUT Add;"ESR2?"
1480       ENTER Add;Esr2$
1490  END LOOP
1500  !
1510  RETURN
1520  !
1530  !*********************** DISPLAY RESULT FUNCTION **********************
1540 Result_read: !
1550  !
1560  DIM Err$(4)[255]
1570  !
1580  OUTPUT Add;"ER?"
1590  ENTER Add;Err$(1)
1600  OUTPUT Add;"EC?"
1610  ENTER Add;Err$(2)
1620  OUTPUT Add;"EI?"
1630  ENTER Add;Err$(3)
1640  OUTPUT Add;"EFI?"
1650  ENTER Add;Err$(4)
1660  !
1670  PRINT " << ERROR DATA >>"
1680  PRINT " << ERROR RATIO >>      "&Err$(1)
1690  PRINT " << ERROR COUNT      >> "&Err$(2)
1700  PRINT " << EI               >> "&Err$(3)
1710  PRINT " << %EFI             >> "&Err$(4)
1720  !
1730  PRINT
1740  RETURN
1750  !
1760  !
1770  END
```

(8) Intermediate measurement data display

This program displays the intermediate measurement data on the CRT.



- Program list

```
10     !*********************************************************************
20     !*                                                                   *
30     !*    MP1762A/MP1764A MEASUREMENT INTERMEDIATE DATA SAMPLE PROGRAM   *
40     !*                                                       ED_MEAS3    *
50     !*********************************************************************
60     !
70     Add=701                                      !MP1762A/MP1764A ADDRESS
80     CLEAR Add                                    !DEVICE CLEAR(ED)
90     !
100    LOOP
110        !
120        CLEAR SCREEN
130        PRINT "** MP1762A/MP1764A MEASUREMENT INTERMEDIATE DATA SAMPLE PROGRAM
 **"
140        PRINT
150        !
160        OUTPUT Add;"STO"
170        GOSUB Meas_cnd                             !MEAS.CONDITION SET
180        GOSUB Closs                                !CHECK CLOCK LOSS
190        GOSUB Auto_srh                             !AUTO SEARCH ON
200        GOSUB Int_dat                              !INTERMEDITAE DATA
210        !
220        INPUT "    Try again?[Yes:0, No:1]",Loop$
230    EXIT IF Loop$="1"
240    END LOOP
250    !
260    STOP
270    !
280    !***************** MEASUREMENT CONDITION SET ***********************
290 Meas_cnd:  !
300    !
310    DIM Prd$[255]
320    LOOP
330        !
340        LOOP
350            INPUT "MEAS.MODE?[Repeat:0, Single:1,Untimed:2]",M_mode$
360        EXIT IF M_mode$="0" OR M_mode$="1" OR M_mode$="2"
370            PRINT "Wrong chosen number!! Please select a correct MEAS.MODE"
380        END LOOP
390        OUTPUT Add;"MOD "&M_mode$
400        CLEAR SCREEN
410        !
420        IF M_mode$<>"2" THEN
430            INPUT "MEAS.TIME=[DAY,HOUR,MINUTE,SECOND]",Prd1$,Prd2$,Prd3$,Prd4$
440        END IF
450        OUTPUT Add;"PRD "&Prd1$&","&Prd2$&","&Prd3$&","&Prd4$
460        !LEAR SCREEN
470        !
480        LOOP
490            INPUT "AUTO SYNC CONDITION=[OFF:0, ON:1]",Auto_sync$
500        EXIT IF Auto_sync$="0" OR Auto_sync$="1"
510            PRINT "Wrong chosen number!! Please select a correct AUTO SYNC CON
DITON"
520        END LOOP
530        OUTPUT Add;"SYN "&Auto_sync$
540        CLEAR SCREEN
550        !
560        IF Auto_sync$="1" THEN
570            LOOP
580                PRINT "SYNC THRESHOLD=[1E-2:0, 1E-3:1, 1E-4:2]"
590                PRINT "                   [1E-5:3, 1E-6:4, 1E-7:5]"
```

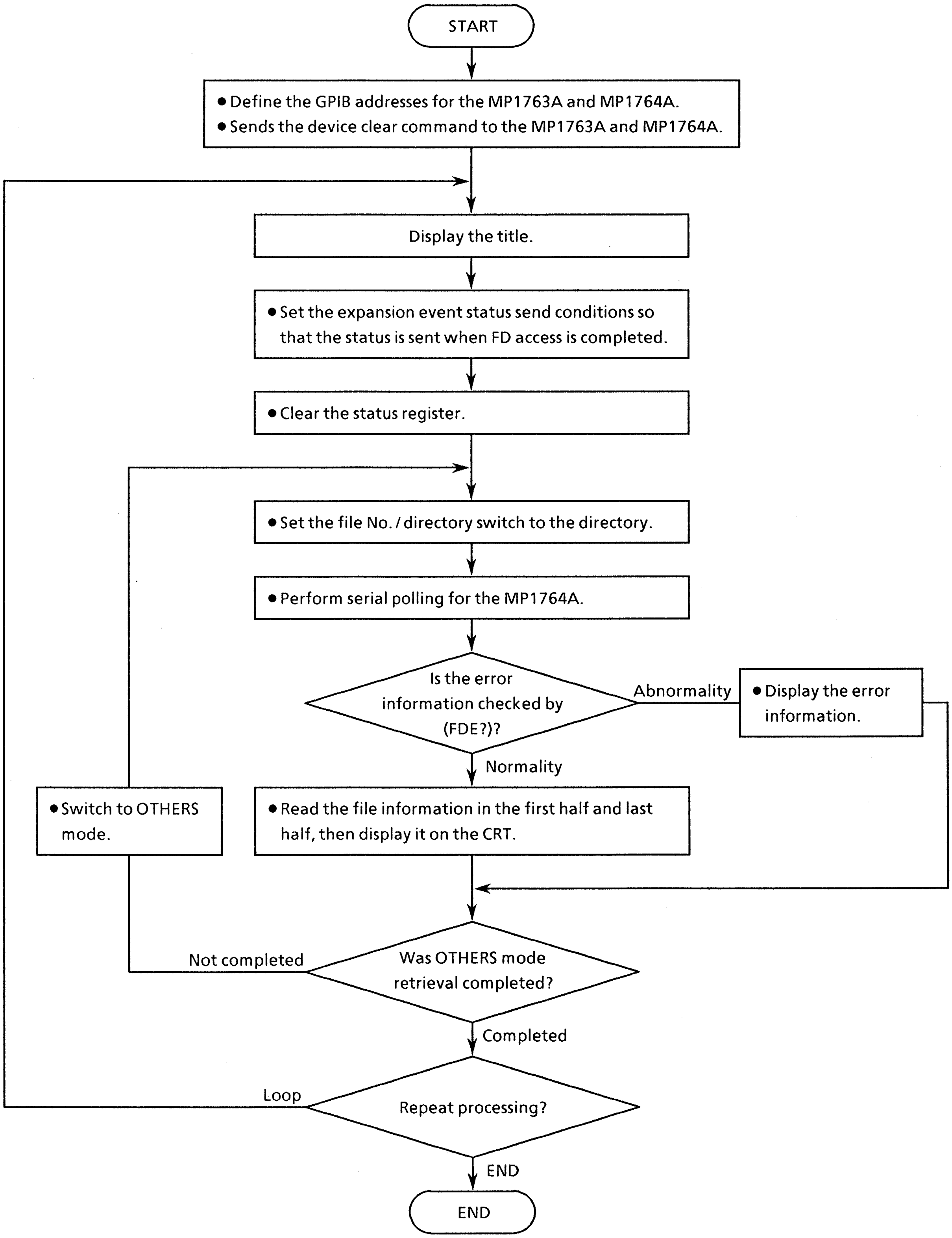
```
600                PRINT "                [IE-8:6,        , INT :8]"
610                PRINT
620                INPUT Sync_th$
630            EXIT IF Sync_th$="0" OR Sync_th$="1" OR Sync_th$="2" OR Sync_th$="
3" OR Sync_th$="4" OR Sync_th$="5" OR Sync_th$="6" OR Sync_th$="8"
640                PRINT "Wrong chosen number!! Please select a correct SYNC THRE
SHOLD"
650            END LOOP
660        OUTPUT Add;"SYE "&Sync_th$
670        CLEAR SCREEN
680        END IF
690        INPUT "Do you change meas.condition?[Yes:0, No:1]",M_cond$
700    EXIT IF M_cond$="1"
710    END LOOP
720    CLEAR SCREEN
730    RETURN
740    !
750    !
760    !************************* CHECK CLOCK LOSS *************************
770 Closs:!
780    !
790    LOOP
800        OUTPUT Add;"CLI?"
810        ENTER Add;Cli$
820        IF Cli$="CLI 1" THEN
830            PRINT "** CLOCK LOSS **"
840            WAIT .5
850        END IF
860        CLEAR SCREEN
870    EXIT IF Cli$="CLI 0"
880    END LOOP
890    !
900    RETURN
910    !
920    !********************** AUTO SEARCH FUNCTION ***********************
930 Auto_srh:!
940    !
950    OUTPUT Add;"SRH 1"
960    !
970    LOOP
980        OUTPUT Add;"SRH?"
990        ENTER Add;Srh$
1000       IF Srh$="SRH 1" THEN
1010           PRINT "** SEARCHING **"
1020           WAIT .5
1030       END IF
1040       CLEAR SCREEN
1050   EXIT IF Srh$="SRH 0" OR Srh$="SRH 2"
1060   END LOOP
1070   !
1080   IF Srh$="SRH 2" THEN
1090       PRINT "Failed in auto search!! Program STOP!!"
1100       STOP
1110   END IF
1120   CLEAR SCREEN
1130   RETURN
1140   !
1150   !
1160   !*********************** INTERMEDIATE DATA ***************************
1170 Int_dat:!
1180   OUTPUT Add;"MOD?"
1190   ENTER Add;Mod$
1200   IF Mod$="MOD 0" THEN
1210       OUTPUT Add;"ESR2?"
1220       ENTER Add;Esr2$
1230       OUTPUT Add;"STA"
```

```
1240  END IF
1250  !
1260  INPUT " CYCLE TIME ? ",Int_time
1270  LOOP
1280       IF Mod$<>"MOD 0" THEN
1290            OUTPUT Add;"STA"
1300       END IF
1310       !
1320       OUTPUT Add;"IMS"
1330       WAIT Int_time
1340       GOSUB Int_dat_read
1350  END LOOP
1360  !
1370  RETURN
1380  !
1390  !*********************** DISPLAY RESULT FUNCTION *********************
1400 Int_dat_read: !
1410  !
1420  DIM Tme$[255],Arm$(3)[255],Err$(4)[255]
1430  DIM Arm_dat$[255],Err_dat$[255]
1440  !
1450  OUTPUT Add;"IMD? 0,0"
1460  ENTER Add;Tme$
1470  OUTPUT Add;"IMD? 1,0"
1480  ENTER Add;Arm_dat$
1490  OUTPUT Add;"IMD? 2,0"
1500  ENTER Add;Err_dat$
1510  PRINT Err_dat$
1520  Arm$(1)=""
1530  Arm$(2)=""
1540  Arm$(3)=""
1550  K=1
1560  IF Arm_dat$<>"ERR" THEN
1570       Max_len=LEN(Arm_dat$)
1580       FOR J=1 TO 3
1590            LOOP
1600            EXIT IF K=(Max_len+1) OR Arm_dat$[K,K]=","
1610                 Arm$(J)=Arm$(J)&Arm_dat$[K,K]
1620                 K=K+1
1630            END LOOP
1640            K=K+1
1650       NEXT J
1660  ELSE
1670       FOR J=1 TO 3
1680            Arm$(J)=" NO DATA   "
1690       NEXT J
1700  END IF
1710  !
1720  Err$(1)=""
1730  Err$(2)=""
1740  Err$(3)=""
1750  Err$(4)=""
1760  L=1
1770  IF Err_dat$<>"ERR" THEN
1780       Max_len2=LEN(Err_dat$)
1790       FOR M=1 TO 4
1800            LOOP
1810            EXIT IF Err_dat$[L,L]="," OR L=(Max_len2+1)
1820                 Err$(M)=Err$(M)&Err_dat$[L,L]
1830                 L=L+1
1840            END LOOP
1850            L=L+1
1860       NEXT M
1870  ELSE
1880       FOR M=1 TO 4
1890            Err$(M)=" NO DATA "
```

```
1900        NEXT M
1910   END IF
1920   !
1930   !
1940   IF Tme$="ERR" THEN
1950        GOTO Jump
1960   END IF
1970   !
1980   PRINT " << START TIME >>    "&Tme$[1,17]&"   << INT TIME >>    "&Tme$[19,35]
1990   !
2000   PRINT " << ARLAM DATA >>  "
2010   PRINT " << POWER FAIL INTVL>> "&Arm$(1)
2020   PRINT " << CLOCK LOSS INTVL>> "&Arm$(2)
2030   PRINT " << SYNC LOSS INTVL >> "&Arm$(3)
2040   !
2050   PRINT " << ERROR DATA >>"
2060   PRINT " << ERROR RATIO >>      "&Err$(1)
2070   PRINT " << ERROR COUNT      >> "&Err$(2)
2080   PRINT " << EI               >> "&Err$(3)
2090   PRINT " << %EFI             >>  "&Err$(4)
2100   !
2110   PRINT
2120 Jump: !
2130   RETURN
2140   !
2150   !
2160   END
```

(9) Reading file information from floppy disk

This program checks file information stored on floppy disk.



- Program list

```
10 !*******************************************************************
20 !*                                                                 *
30 !*   MP1762A/MP1764A FLOPPY DISK OPERATION SAMPLE PROGRAM          *
40 !*                                                    ED_FD1       *
50 !*******************************************************************
60 !
70 Add=701                                  !MP1762A/MP1764A GPIB ADDRESS
80 CLEAR Add                                !DEVICE CLEAR
90 !
100 DIM Fil_pat$(2)[255],Fil_oths$(2)[255],Fde$[255]
110!
120 LOOP
130      !
140      CLEAR SCREEN
150      PRINT "** MP1762A/MP1764A FD OPERATION SAMPLE PROGRAM ** "
160      !
170      GOSUB Status_set                   !STATUS RESISITOR SET
180      GOSUB Fd_ope                       !FD OPERATION SET
190      !
200      INPUT "    Try again?[Yes:0, No:1]",Loop$
210 EXIT IF Loop$="1"
220 END LOOP
230 !
240 STOP
250 !
260 !***************** STATUS RESISTOR SET ************************
270 Status_set:!
280 !
290 OUTPUT Add;"*SRE 4"                     !ESR2 ENABLE
300 OUTPUT Add;"ESE2 2"                     !FD ACCESS END
310 !
320 RETURN
330 !
340 !***************** FD OPERATION SET ***************************
350 Fd_ope:!
360 !
370 OUTPUT Add;"*STB?"
380 ENTER Add;Stb$
390 OUTPUT Add;"ESR2?"
400 ENTER Add;Esr2$
410 !
420 FOR I=0 TO 1
430      OUTPUT Add;"MEM "&VAL$(I)
440      !
450      OUTPUT Add;"FIL 1"
460      !
470      LOOP
480          A=SPOLL(Add)                   !SERIAL POLLING
490      EXIT IF BIT(A,2)=1                 !ESR2 ENABLE
500          WAIT .1
510      END LOOP
520      GOSUB Fd_dir_dsp                   !FD DIR INFORMATION
530      OUTPUT Add;"ESR2?"
540      ENTER Add;Esr2$
550 NEXT I
560 !
570 RETURN
580 !
590 !***************** FD DIR INFORMATION *************************
600 Fd_dir_dsp:!
610 !
```

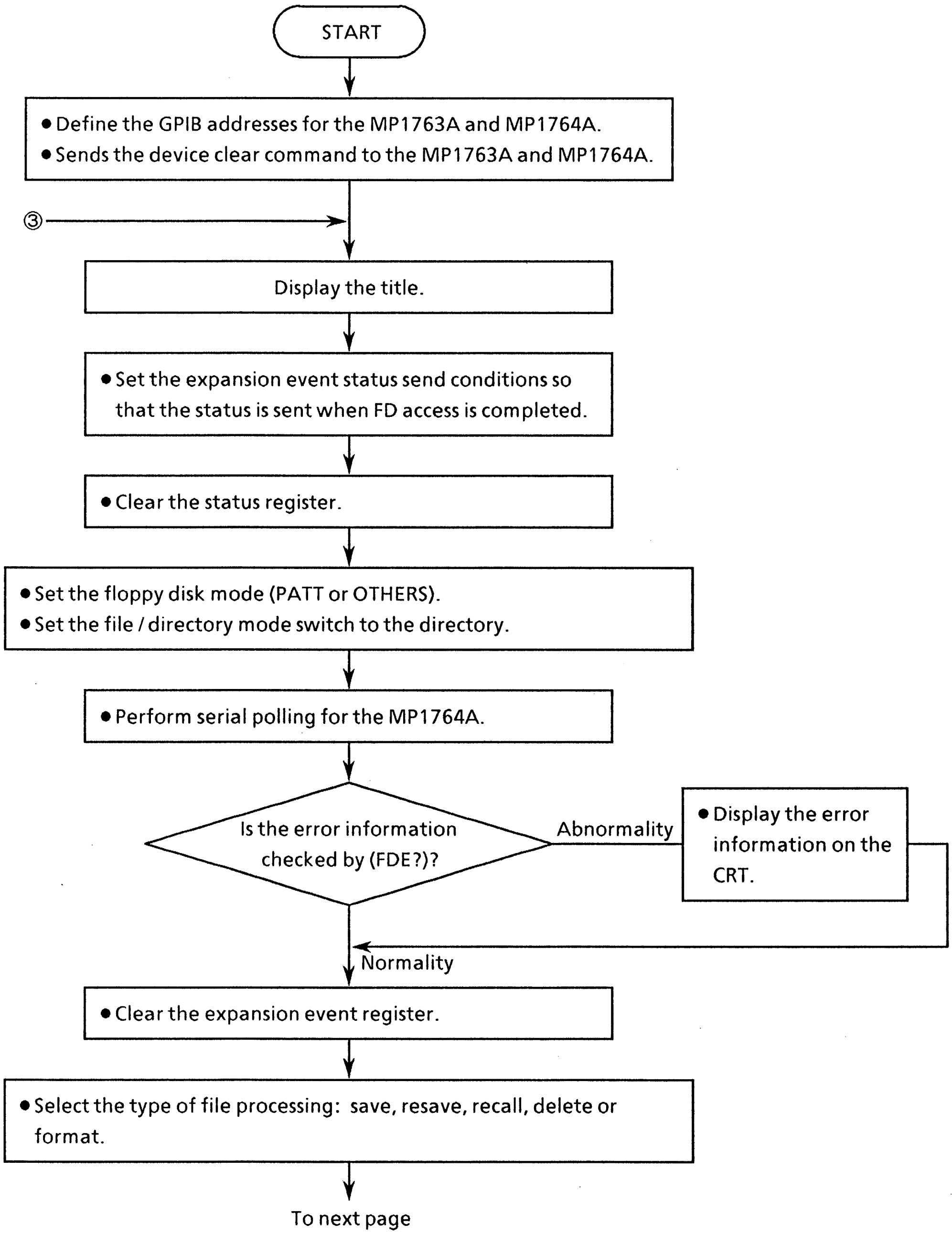
```
620 OUTPUT Add;"FDE?"
630 ENTER Add;Fde$
640 IF Fde$="FDE 10" THEN
650     OUTPUT Add;"FSH? 0"
660     IF I=0 THEN
670         ENTER Add;Fil_pat$(1)
680     ELSE
690         ENTER Add;Fil_oths$(1)
700     END IF
710     OUTPUT Add;"FSH? 1"
720     IF I=0 THEN
730         ENTER Add;Fil_pat$(2)
740     ELSE
750         ENTER Add;Fil_oths$(2)
760     END IF
770     !
780     IF I=0 THEN
790         PRINT " << Unused size      >> "&Fil_pat$(1)[5,11]
800         PRINT " << Used size        >> "&Fil_pat$(1)[13,19]
810         PRINT " << PATTERN FILES    >> "
820         PRINT " << PATT File count >>       "&Fil_pat$(1)[21,22]
830         K=24
840         Max_fil1=LEN(Fil_pat$(1))
850         Max_fil2=LEN(Fil_pat$(2))
860         IF Fil_pat$(1)[K,Max_fil1+1]<>"--" AND Fil_pat$(2)[K,Max_fil2+1]<>"-
-" THEN
870             PRINT " << File name         >>        "&Fil_pat$(1)[K,Max_fil1+1]&"
,"&Fil_pat$(2)[K,Max_fil2+1]
880         ELSE
890             IF Fil_pat$(1)[K,Max_fil1+1]<>"--" THEN
900                 PRINT " << File name          >>      "&Fil_pat$(1)[K,Max_fil1+
1]
910             ELSE
920                 PRINT " << File name          >>      "&Fil_pat$(2)[K,Max_fil2+
1]
930             END IF
940         END IF
950     ELSE
960         PRINT " << OTHERS FILES     >>"
970         PRINT " << OTHS File count >>       "&Fil_oths$(1)[21,22]
980         K=24
990         Max_fil1=LEN(Fil_oths$(1))
1000        Max_fil2=LEN(Fil_oths$(2))
1010        IF Fil_oths$(1)[K,Max_fil1+1]<>"--" AND Fil_oths$(2)[K,Max_fil2+1]<>
"--" THEN
1020            PRINT " << File name         >>        "&Fil_oths$(1)[K,Max_fil1+1]&
","&Fil_oths$(2)[K,Max_fil2+1]
1030        ELSE
1040            IF Fil_oths$(1)[K,Max_fil1+1]<>"--" THEN
1050                PRINT " << File name          >>      "&Fil_oths$(1)[K,Max_fil1
+1]
1060            ELSE
1070                PRINT " << File name          >>      "&Fil_oths$(2)[K,Max_fil2
+1]
1080            END IF
1090        END IF
1100    END IF
1110 ELSE
1120    SELECT Fde$[6,6]
1130        CASE "0"
1140            PRINT " << E0:Media error                   >> "
1150        CASE "1"
1160            PRINT " << E1:Write protection error        >> "
1170        CASE "2"
1180            PRINT " << E2:File full                     >> "
1190        CASE "3"
```

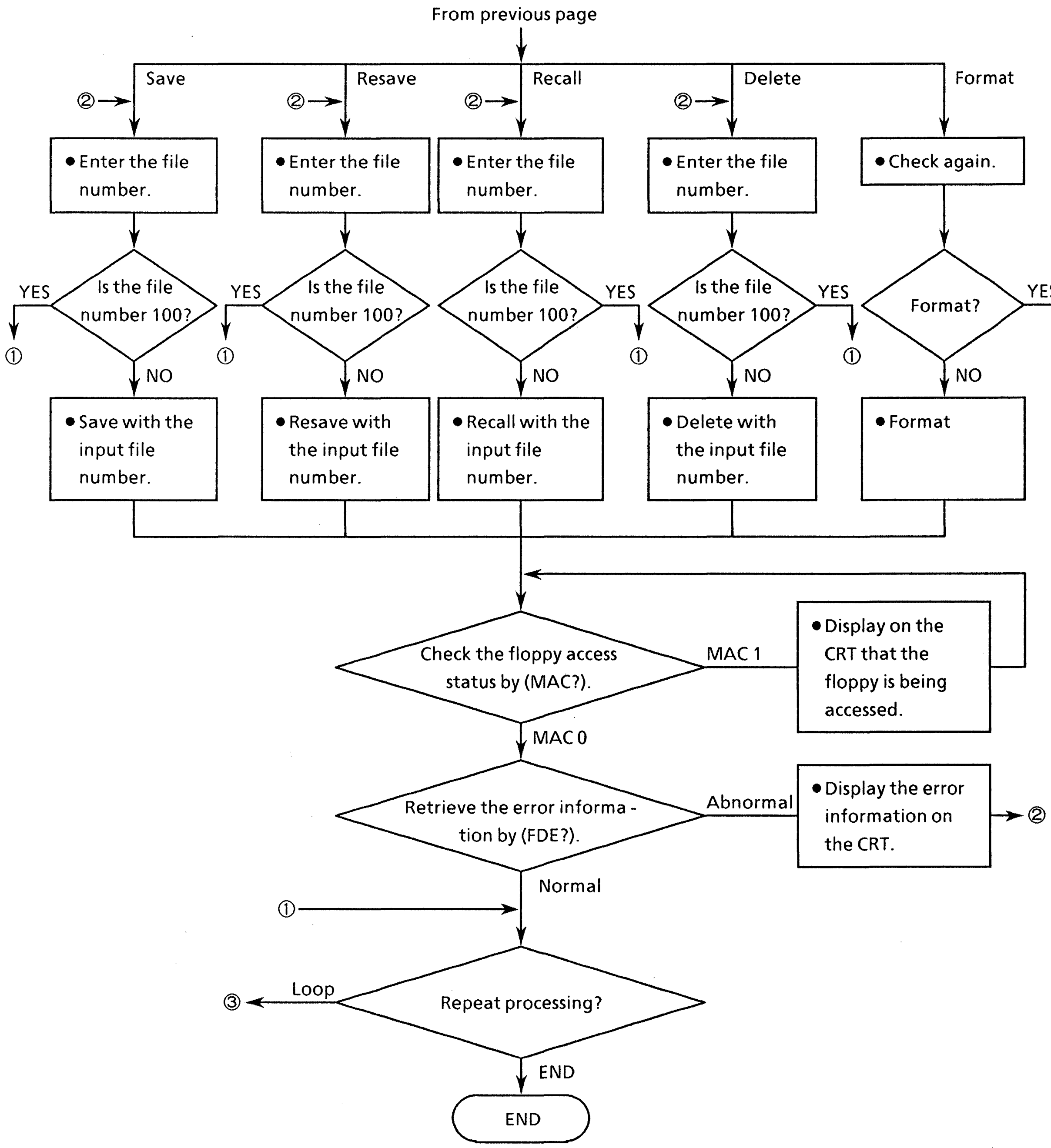
```
1200            PRINT " << E3:File not found              >> "
1210        CASE "4"
1220            PRINT " << E4:File already exists error >> "
1230        CASE "5"
1240            PRINT " << E5:Write error                >> "
1250        CASE "6"
1260            PRINT " << E6:Read error                 >> "
1270        CASE "7"
1280            PRINT " << E7:File type , File error     >> "
1290        CASE "8"
1300            PRINT " << E8:FD error                   >> "
1310        CASE "9"
1320            PRINT " << E9:Hardware error             >> "
1330     END SELECT
1340 END IF
1350!
1360 RETURN
1370!
1380 END
```

(10) Floppy disk operation

This program executes save, recall, resave, delete, and format operations for the floppy disk.

The floppy disk access status is checked by either serial polling or the request command (MAC?).




From previous page
Save
Resave
Recall
Delete
Format
②
• Enter the file number.
• Check again.
Is the file number 100?
Format?
YES
NO
①
• Save with the input file number.
• Resave with the input file number.
• Recall with the input file number.
• Delete with the input file number.
• Format
Check the floppy access status by (MAC?).
MAC 1
• Display on the CRT that the floppy is being accessed.
MAC 0
Retrieve the error informa - tion by (FDE?).
Abnormal
• Display the error information on the CRT.
Normal
Repeat processing?
Loop
③
END

- Program list

```
10 !*********************************************************************
20 !*                                                                   *
30 !*   MP1762A/MP1764A FLOPPY DISK OPERATION SAMPLE PROGRAM            *
40 !*                                                  ED_FD2           *
50 !*********************************************************************
60 !
70 Add=701                              !MP1762A/MP1764A GPIB ADDRESS
80 CLEAR Add                            !DEVICE CLEAR
90 !
100 DIM Fil_pat$(2)[255],Fde$[255]
110!
120 LOOP
130     !
140     CLEAR SCREEN
150     PRINT "** MP1762A/MP1764A FD OPERATION SAMPLE PROGRAM ** "
160     !
170     GOSUB Status_set                !STATUS RESISITOR SET
180     GOSUB Fd_ope                    !FD OPERATION SET
190     GOSUB Fd_fil                    !FD FILE OPERATION
200     !
210     !
220     INPUT "   Try again?[Yes:0, No:1]",Loop$
230 EXIT IF Loop$="1"
240 END LOOP
250 !
260 STOP
270 !
280 !***************** STATUS RESISTOR SET ************************
290 Status_set:!
300 !
310 OUTPUT Add;"*SRE 4"                 !ESR2 ENABLE
320 OUTPUT Add;"ESE2 2"                 !FD ACCESS END
330 !
340 RETURN
350 !
360 !***************** FD OPERATION SET ****************************
370 Fd_ope:!
380 !
390 OUTPUT Add;"*STB?"
400 ENTER Add;Stb$
410 OUTPUT Add;"ESR2?"
420 ENTER Add;Esr2$
430 !
440 LOOP
450     INPUT " Memory mode select [PATT:0, OTHERS:1]",Mmod$
460 EXIT IF Mmod$="0" OR Mmod$="1"
470     PRINT " Wrong chosen number!! Please select a correct number "
480 END LOOP
490 OUTPUT Add;"MEM "&Mmod$
500 !
510 OUTPUT Add;"FIL 1"
520 !
530 LOOP
540     A=SPOLL(Add)                !SERIAL POLLING
550 EXIT IF BIT(A,2)=1              !ESR2 ENABLE
560     WAIT .1
570 END LOOP
580 GOSUB Fd_err                    !FD DIR INFORMATION
590 OUTPUT Add;"ESR2?"
600 ENTER Add;Esr2$
610 !
```

```
620 RETURN
630 !
640 !********************** FD FILE OPERATION *************************
650 Fd_fil: !
660   !
670   LOOP
680       INPUT " Select [ Save:0, Resave:1, Recall:2, Delete:3, Format:4 ]",Ope
$
690   EXIT IF Ope$="0" OR Ope$="1" OR Ope$="2" OR Ope$="3" OR Ope$="4"
700       CLEAR SCREEN
710       PRINT "Wrong chosen number!! Please select a correct number"
720   END LOOP
730   !
740   SELECT Ope$
750       CASE "0"
760           GOSUB Dsave
770       CASE "1"
780           GOSUB Dresave
790       CASE "2"
800           GOSUB Drecall
810       CASE "3"
820           GOSUB Ddelete
830       CASE "4"
840           GOSUB Dformat
850    END SELECT
860    !
870    RETURN
880    !
890  !********************** DATA SAVE OPERATION *********************
900 Dsave: !
910   !
920   LOOP
930       !
940       Num$=""
950       !
960       INPUT "** DATA SAVE ** FILE NUMBER [0 to 99],[Exit:100] ",Num$
970   EXIT IF Num$="100"
980       OUTPUT Add;"SAV "&Num$
990       GOSUB Access
1000  EXIT IF Fde$="FDE 10"
1010  END LOOP
1020  CLEAR SCREEN
1030  !
1040  RETURN
1050  !
1060 !********************** DATA RESAVE OPERATION *******************
1070 Dresave:!
1080  !
1090  LOOP
1100      !
1110      Num$=""
1120      !
1130      INPUT "** DATA RESAVE ** FILE NUMBER [0 to 99],[Exit:100] ",Num$
1140  EXIT IF Num$="100"
1150      OUTPUT Add;"RSV "&Num$
1160      GOSUB Access
1170  EXIT IF Fde$="FDE 10"
1180  END LOOP
1190  CLEAR SCREEN
1200  !
1210  RETURN
1220  !
1230 !********************** DATA RECALL OPERATION *******************
1240 Drecall:!
1250  !
1260  LOOP
```

```
1270        !
1280        Num$=""
1290        !
1300        INPUT "** DATA RECALL ** FILE NUMBER [0 to 99],[Exit:100] ",Num$
1310    EXIT IF Num$="100"
1320        OUTPUT Add;"RCL "&Num$
1330        GOSUB Access
1340    EXIT IF Fde$="FDE 10"
1350    END LOOP
1360    CLEAR SCREEN
1370    !
1380    RETURN
1390    !
1400 !********************** DATA DELETE OPERATION *******************
1410 Ddelete:!
1420    !
1430    LOOP
1440        !
1450        Num$=""
1460        !
1470        INPUT "** FILE DELETE ** FILE NUMBER [0 to 99],[Exit:100] ",Num$
1480    EXIT IF Num$="100"
1490        OUTPUT Add;"DEL "&Num$
1500        GOSUB Access
1510    EXIT IF Fde$="FDE 10"
1520    END LOOP
1530    CLEAR SCREEN
1540    !
1550    RETURN
1560    !
1570    !************************ FD FORMAT OPERATION *******************
1580 Dformat:!
1590     INPUT " Format disk [Yes:0, No:1]",Fmt$
1600     IF Fmt$="0" THEN
1610         OUTPUT Add;"FIL 0"
1620         OUTPUT Add;"FDF"
1630         GOSUB Access
1640     END IF
1650     CLEAR SCREEN
1660     !
1670     RETURN
1680     !
1690    !************************ FD ERROR CHECK ***********************
1700 Fd_err:!
1710     !
1720     OUTPUT Add;"FDE?"
1730     ENTER Add;Fde$
1740     IF Fde$<>"FDE 10" THEN
1750        SELECT Fde$[6,6]
1760            CASE "0"
1770                PRINT " << E0:Media error                >> "
1780            CASE "1"
1790                PRINT " << E1:Write protection error     >> "
1800            CASE "2"
1810                PRINT " << E2:File full                  >> "
1820            CASE "3"
1830                PRINT " << E3:File not found             >> "
1840            CASE "4"
1850                PRINT " << E4:File already exists error  >> "
1860            CASE "5"
1870                PRINT " << E5:Write error                >> "
1880            CASE "6"
1890                PRINT " << E6:Read error                 >> "
1900            CASE "7"
1910                PRINT " << E7:File type , File error     >> "
1920            CASE "8"
```

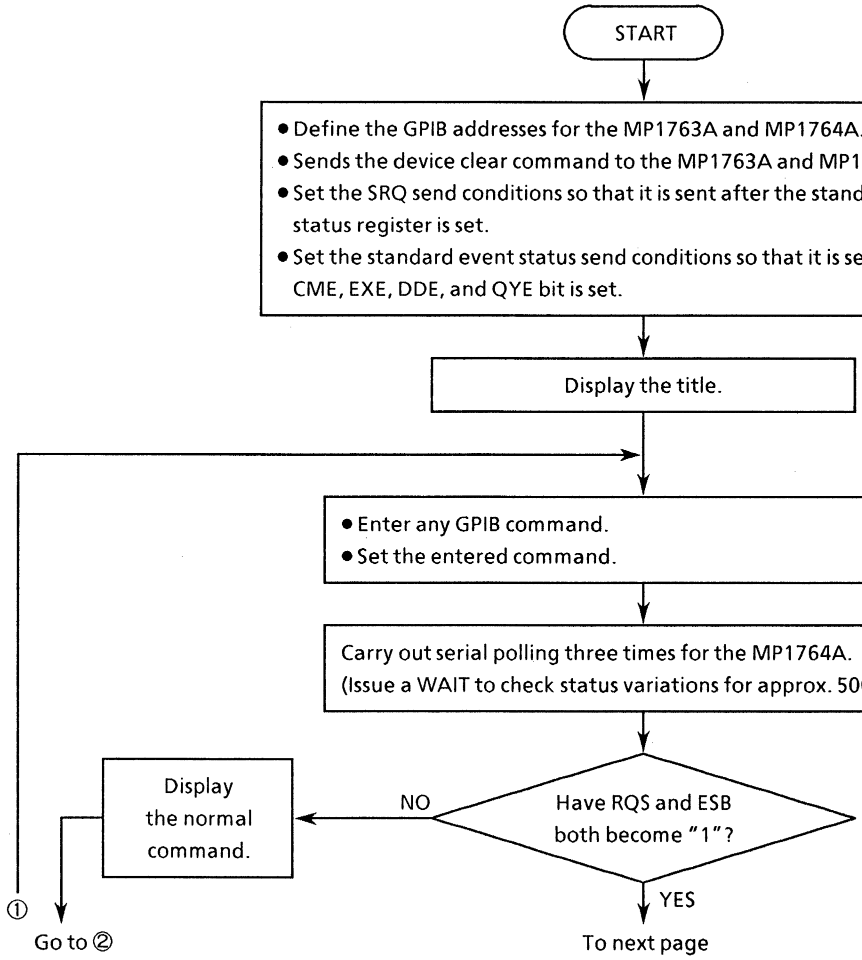
```
1930                PRINT " << E8:FD error                        >> "
1940            CASE "9"
1950                PRINT " << E9:Hardware error                  >> "
1960        END SELECT
1970     ELSE
1980        PRINT " << Operation complete !! >> "
1990     END IF
2000    !
2010     RETURN
2020    !
2030    !************************ FD ACCESS CHECK ************************
2040 Access: !
2050    !
2060    LOOP
2070        OUTPUT Add;"MAC?"
2080        ENTER Add;Mac$
2090    EXIT IF Mac$="MAC 0"
2100        PRINT " FD ACCESS "
2110        WAIT .5
2120        CLEAR SCREEN
2130    END LOOP
2140    !
2150    GOSUB Fd_err
2160    !
2170    RETURN
2180    !
2190    END
```

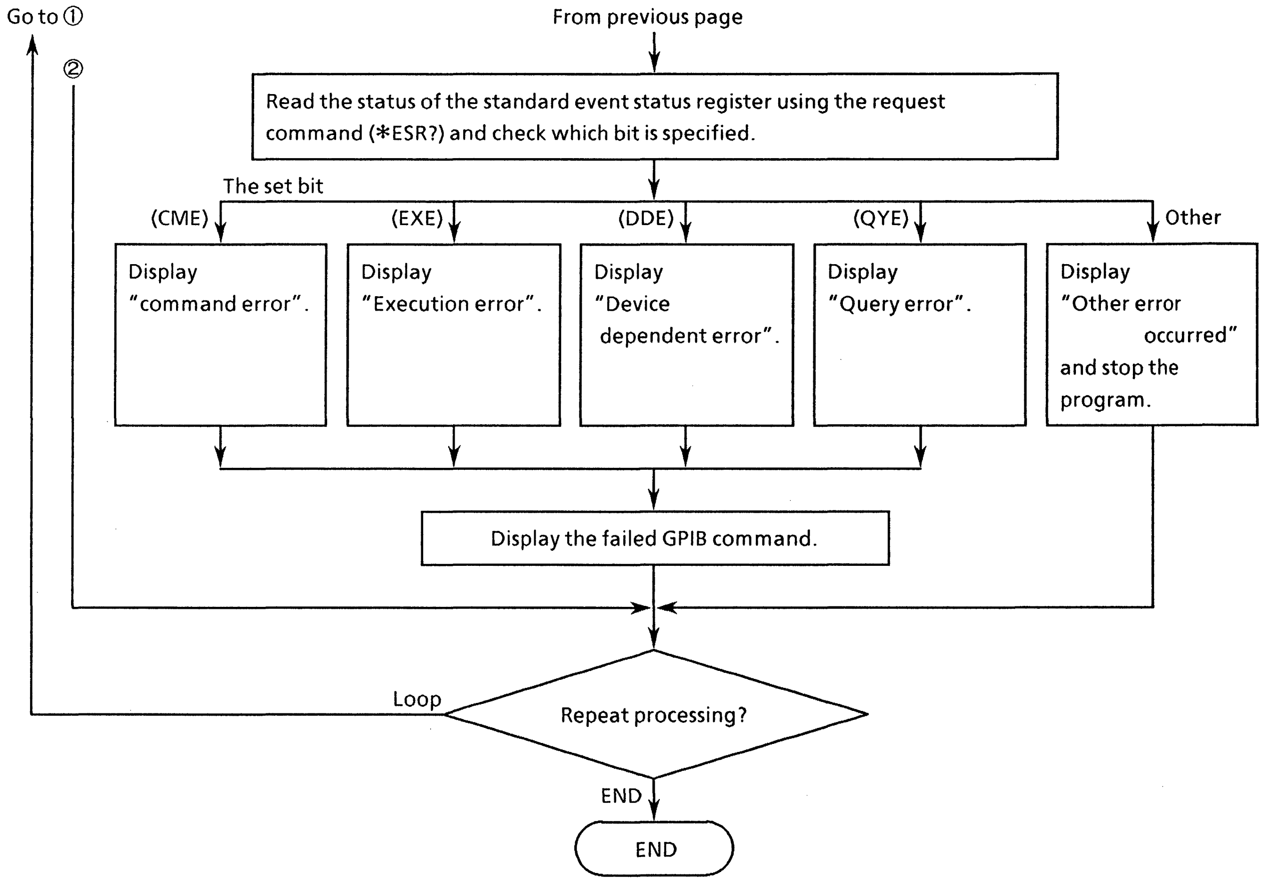
(11) Status byte checking

This program checks the CME, EXE, DDE, and QYE bit of the standard event status byte and displays whether the input GPIB command is correct. The data is displayed on the CRT.

If abnormal, the meaning of the error is displayed.

Also, the GPIB status byte is checked using serial polling, and the status of the standard event status register is checked by the data request command.




Go to ①
②
From previous page
Read the status of the standard event status register using the request command (*ESR?) and check which bit is specified.
The set bit
(CME)
(EXE)
(DDE)
(QYE)
Other
Display "command error".
Display "Execution error".
Display "Device dependent error".
Display "Query error".
Display "Other error occurred" and stop the program.
Display the failed GPIB command.
Loop
Repeat processing?
END
END

- Program list

```
10 !*********************************************************************
20 !*                                                                   *
30 !*     MP1762A/MP1764A STANDARD EVENT STATUS REGISTOR CHECK          *
40 !*                       SAMPLE PROGRAM                  ED_ESR      *
50 !*********************************************************************
60 !
70 Add=701                                !MP1762A/MP1764A GPIB ADDRESS
80 CLEAR Add                              !DEVICE CLEAR
90 !
100 OUTPUT Add;"*SRE 32"                  !SRQ ON ESR bit
110 OUTPUT Add;"*ESE 60"                  !ESR ON CME,EXE,DDE,QYE
120 OUTPUT Add;"ESE2 0"                   !DISABLE ESR2
130 OUTPUT Add;"ESE3 0"                   !DISABLE ESR3
140 !
150 PRINT " **           MP1762A/MP1764A            ** "
160 PRINT " ** STANDARD EVENT STATUS REGISTOR CHECK ** "
170 !
180 LOOP
190     !
200     INPUT " Input any GPIB command ? ",Cmd$
210     OUTPUT Add;Cmd$
220     !
230     GOSUB S_poll
240     !
250     INPUT " Next command set ? [Yes:0, No:1] ",Loop$
260     !
270 EXIT IF Loop$="1"
280 !
290 END LOOP
300 !
310 STOP
320 !
330 !************************** POLLING **************************
340 S_poll:!
350  !
360  Byt=0
370  !
380  FOR I=0 TO 2
390      !
400      A=SPOLL(Add)
410      IF BIT(A,6)=1 AND BIT(A,5)=1 THEN
420         Byt=A
430      END IF
440      WAIT .5
450  NEXT I
460  !
470  IF BIT(Byt,6)=1 AND BIT(Byt,5)=1 THEN
480      GOSUB Err
490  ELSE
500      PRINT " GPIB command is OK!! "
510      PRINT
520  END IF
530  !
540  RETURN
550  !
560  !************************ ESR CHECK ******************************
570 Err: !
580  !
590  OUTPUT Add;"*ESR?"
600  ENTER Add;Esr
610  IF BIT(Esr,2)=1 THEN PRINT " << Query error >> "
```

```
620  IF BIT(Esr,3)=1 THEN PRINT " << Device dependent error >> "
630  IF BIT(Esr,4)=1 THEN PRINT " << Execution error >> "
640  IF BIT(Esr,5)=1 THEN PRINT " << Command error >> "
650  !
660  PRINT " Input command = "&Cmd$
670  PRINT
680  !
690  IF BIT(Esr,0)=1 OR BIT(Esr,1)=1 OR BIT(Esr,6)=1 OR BIT(Esr,7)=1 THEN
700      PRINT " Other error occured !! "
710  END IF
720  !
730  RETURN
740  !
750  END
```

(12) DMA transfer for pattern data

This program transfers pattern data to and from an HP9000 series computer, which is used as a controller, by DMA.

The output transferred using DMA to the MP1764A is then retransmitted using DMA.

The results of actual execution are shown below, and the relationship between each array variable data and the values to be set to these array variables (decimal notation, binary notation, hexadecimal notation) are shown in Table 12-1. The relationship differs depending on the controller used.

Table 12-1 Relationship between array variable and setting value

Array variable	Setting value (decimal)	Binary numbers and BIT LED No.																Hexa-decimal number	Page reference number
		16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1		
Dta (0)	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1H	1
Dta (1)	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	2H	2
Dta (2)	4	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	4H	3
Dta (3)	8	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	8H	4
Dta (4)	16	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	10H	5
Dta (5)	32	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	20H	6
Dta (6)	64	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	40H	7
Dta (7)	128	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	80H	8
Dta (8)	256	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	100H	9
Dta (9)	512	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	200H	10
Dta (10)	1024	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	400H	11
Dta (11)	2048	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	800H	12
Dta (12)	4096	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1000H	13
Dta (13)	8192	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2000H	14
Dta (14)	16384	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000H	15
Dta (15)	32767	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	7FFFH	16
Dta (16)	−32768	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8000H	17
Dta (17)	−16384	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	C000H	18
Dta (18)	−8192	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	E000H	19
Dta (19)	−4096	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	F000H	20
Dta (20)	−2048	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	F800H	21
Dta (21)	−1024	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	FC00H	22
Dta (22)	−512	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	FE00H	23
Dta (23)	−256	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	FF00H	24
Dta (24)	−128	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	FF80H	25
Dta (25)	−64	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	FFC0H	26
Dta (26)	−32	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	FFE0H	27
Dta (27)	−16	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	FFF0H	28
Dta (28)	−8	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	FFF8H	29
Dta (29)	−4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	FFFCH	30
Dta (30)	−2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	FFFEH	31
Dta (31)	−1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	FFFFH	32

START

- Set the maximum number of element in array variable "Dta" and "Red" to 32.
- Set the maximum number of storage characters in variable "Bit$" to 20 characters.
- Define the GPIB addresses for the MP1763A and MP1764A.
- Sends the device clear command to the MP1763A and MP1764A.

Display the title.

- Set the DATA mode.
- Set the page to 1.
- Set the data length to 512 (for 32 pages).
- Set the sync mode to "Normal".

Set the pattern data to be transferred to the array variable "Dta".

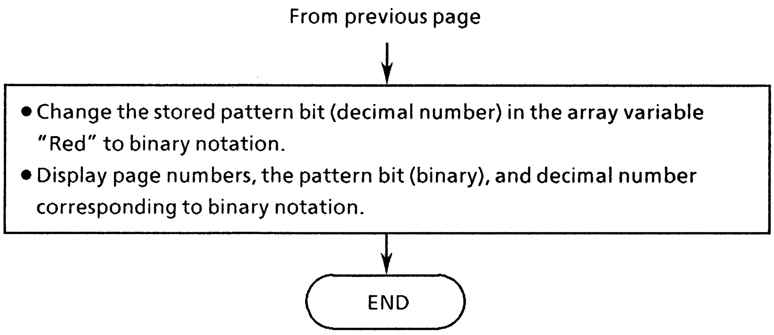
- Set the pattern byte data to be transferred to the internal RAM to 64 (for transferring 32 pages), and the pattern input header address to 0 (for transferring from page 1).

Transfer the pattern data of the array variable "Dta" to the internal RAM in DMA transfer. *1

- Set the pattern byte data from the internal RAM to 64 bytes (for transferring 32 pages), and the pattern output header address to 0 (for transferring from page 1).

Read the pattern data from the internal RAM in the RAM transfer, and store it into the array variable "Red". *2

To next page



*1 Pattern data transfer < OUTPUT Add USING "W"; Dta(*)>

W : The integer of the 2's complement of 16 bits is output.
Since the GPIB interface board is 8 bits I / O, the upper bytes are the head and 2 bytes are sent first.

* : All of the specified array Dta is output.

*2 Pattern data transfer < ENTER Add USING "W"; Red(*)>

W : The integer of the 2's complement of 16 bits is output.
Since the GPIB interface board is 8 bits I / O, the upper bytes are the head and 2 bytes are sent first.

* : Data is stored in all of the specified arrays.

- Program list

```
10 !*****************************************************************
20 !*                                                               *
30 !*    MP1762A/MP1764A PROGRAMMABLE PATTERN DATA DMA TRANSFER     *
40 !*                    SAMPLE PROGRAM                    ED_DMA   *
50 !*****************************************************************
60 !
70 DIM Red(31)
80 DIM Dta(31)
90 DIM Bit$[20]
100!
110 Add=701                                  !MP1762A/MP1764A GPIB ADDRESS
120 CLEAR Add                                !DEVICE CLEAR
130 CLEAR SCREEN
140 !
150 PRINT " **          MP1762A/MP1764A              ** "
160 PRINT " **     PATTERN DATA DMA TRANSFER         ** "
170 !
180 OUTPUT Add;"PTS 1"
190 OUTPUT Add;"PAG 1"
200 !
210 OUTPUT Add;"SYM 0"
220 !
230 OUTPUT Add;"DLN 512"
240 !
250 !*********************** DATA SET *************************
260 Dta(0)=1
270 Dta(1)=2
280 Dta(2)=4
290 Dta(3)=8
300 Dta(4)=16
310 Dta(5)=32
320 Dta(6)=64
330 Dta(7)=128
340 Dta(8)=256
350 Dta(9)=512
360 Dta(10)=1024
370 Dta(11)=2048
380 Dta(12)=4096
390 Dta(13)=8192
400 Dta(14)=16384
410 Dta(15)=32767
420 Dta(16)=-32768
430 Dta(17)=-16384
440 Dta(18)=-8192
450 Dta(19)=-4096
460 Dta(20)=-2048
470 Dta(21)=-1024
480 Dta(22)=-512
490 Dta(23)=-256
500 Dta(24)=-128
510 Dta(25)=-64
520 Dta(26)=-32
530 Dta(27)=-16
540 Dta(28)=-8
550 Dta(29)=-4
560 Dta(30)=-2
570 Dta(31)=-1
580 !
590 !
600 OUTPUT Add;"WRT 64,0"
610 !
```

```
620 OUTPUT Add USING "W";Dta(*)
630 !
640 OUTPUT Add;"RED? 64,0"
650 !
660 ENTER Add USING "W";Red(*)
670 !
680 FOR I=1 TO 32
690     Bit$=IVAL$(Red(I-1),2)
700     !
710     IMAGE "PATTERN BIT PAGE=",AA,XXX,AAAAAAAAAAAAAAAA,XX,DDDDDD
720     PRINT USING 710;VAL$(I);Bit$;Red(I-1)
730     !
740 NEXT I
750 !
760 END
```

Execution result

```
PATTERN BIT PAGE=1    0000000000000001        1
PATTERN BIT PAGE=2    0000000000000010        2
PATTERN BIT PAGE=3    0000000000000100        4
PATTERN BIT PAGE=4    0000000000001000        8
PATTERN BIT PAGE=5    0000000000010000       16
PATTERN BIT PAGE=6    0000000000100000       32
PATTERN BIT PAGE=7    0000000001000000       64
PATTERN BIT PAGE=8    0000000010000000      128
PATTERN BIT PAGE=9    0000000100000000      256
PATTERN BIT PAGE=10   0000001000000000      512
PATTERN BIT PAGE=11   0000010000000000     1024
PATTERN BIT PAGE=12   0000100000000000     2048
PATTERN BIT PAGE=13   0001000000000000     4096
PATTERN BIT PAGE=14   0010000000000000     8192
PATTERN BIT PAGE=15   0100000000000000    16384
PATTERN BIT PAGE=16   0111111111111111    32767
PATTERN BIT PAGE=17   1000000000000000   -32768
PATTERN BIT PAGE=18   1100000000000000   -16384
PATTERN BIT PAGE=19   1110000000000000    -8192
PATTERN BIT PAGE=20   1111000000000000    -4096
PATTERN BIT PAGE=21   1111100000000000    -2048
PATTERN BIT PAGE=22   1111110000000000    -1024
PATTERN BIT PAGE=23   1111111000000000     -512
PATTERN BIT PAGE=24   1111111100000000     -256
PATTERN BIT PAGE=25   1111111110000000     -128
PATTERN BIT PAGE=26   1111111111000000      -64
PATTERN BIT PAGE=27   1111111111100000      -32
PATTERN BIT PAGE=28   1111111111110000      -16
PATTERN BIT PAGE=29   1111111111111000       -8
PATTERN BIT PAGE=30   1111111111111100       -4
PATTERN BIT PAGE=31   1111111111111110       -2
PATTERN BIT PAGE=32   1111111111111111       -1
```

(13) DMA transfer for BLOCK WINDOW

This program transfers the BLOCK WINDOW pattern to an HP9000 series computer, which is used as a controller, by DMA.

The BLOCK WINDOW pattern data transferred to the MP1764A is output using the request command (MGB?).

The execution results are shown below and the relationship between each array "Dta" and numeric value to be set (decimal, hexadecimal) is shown in Fig. 13-1. This relationship differs depending on the controller used.

The corresponding page number in the table assumes the header address of the DMA transfer is 0. (See Appendix 2.)

Table 13-1 Relationship between Array variable and setting value

Array variable	Setting value in decimal notation	The corresponding page number when the header is 1. 1 8	9 16	17 24	25 32	Setting value in hexa-decimal notation
Dta (0)	1	00000000	00000000	11000000	00000000	1H
Dta (1)	2	00000000	00000000	00110000	00000000	2H
Dta (2)	4	00000000	00000000	00001100	00000000	4H
Dta (3)	8	00000000	00000000	00000011	00000000	8H
Dta (4)	16	00000000	00000000	00000000	11000000	10H
Dta (5)	32	00000000	00000000	00000000	00110000	20H
Dta (6)	64	00000000	00000000	00000000	00001100	40H
Dta (7)	128	00000000	00000000	00000000	00000011	80H
Dta (8)	256	11000000	00000000	00000000	00000000	100H
Dta (9)	512	00110000	00000000	00000000	00000000	200H
Dta (10)	1024	00001100	00000000	00000000	00000000	400H
Dta (11)	2048	00000011	00000000	00000000	00000000	800H
Dta (12)	4096	00000000	11000000	00000000	00000000	1000H
Dta (13)	8192	00000000	00110000	00000000	00000000	2000H
Dta (14)	16384	00000000	00001100	00000000	00000000	4000H
Dta (15)	32767	11111111	11111100	11111111	11111111	7FFFH
Dta (16)	−32768	00000000	00000011	00000000	00000000	8000H
Dta (17)	−16384	00000000	00001111	00000000	00000000	C000H
Dta (18)	−8192	00000000	00111111	00000000	00000000	E000H
Dta (19)	−4096	00000000	11111111	00000000	00000000	F000H
Dta (20)	−2048	00000011	11111111	00000000	00000000	F800H
Dta (21)	−1024	00001111	11111111	00000000	00000000	FC00H
Dta (22)	−512	00111111	11111111	00000000	00000000	FE00H
Dta (23)	−256	11111111	11111111	00000000	00000000	FF00H
Dta (24)	−128	11111111	11111111	00000000	00000011	FF80H
Dta (25)	−64	11111111	11111111	00000000	00001111	FFC0H
Dta (26)	−32	11111111	11111111	00000000	00111111	FFE0H
Dta (27)	−16	11111111	11111111	00000000	11111111	FFF0H
Dta (28)	−8	11111111	11111111	00000011	11111111	FFF8H
Dta (29)	−4	11111111	11111111	00001111	11111111	FFFCH
Dta (30)	−2	11111111	11111111	00111111	11111111	FFFEH
Dta (31)	−1	11111111	11111111	11111111	11111111	FFFFH

START

↓

- Set the maximum number of element in array variable "Dta" to 32.
- Set the maximum number of element in array variable Bit$ to 4, and the maximum characters for storage to 255 charactors.
- Set the maximum characters of variable Page$ to 255 charactors.
- Define the GPIB addresses for the MP1763A and MP1764A.
- Sends the device clear command to the MP1763A and MP1764A.

↓

Display the title.

↓

- Set the DATA mode.
- Set the page to 1.
- Set the sync mode to "Normal".
- Set the data length to 512.

↓

The BLOCK WINDOW pattern to be transferred is specified as array variable "Dta".

↓

- Repeat the following processing until the maximum number of elements of the array variable "Dta" is reached.
  (1) Set the BLOCK WINDOW pattern of the internal RAM to 2 bytes, and the pattern input header address to 0.
  (2) Transfer the BLOCK WINDOW pattern data of the array variable "Dta" to internal RAM.
  (3) Read in 32 pages of the BLOCK WINDOW data (bit$) using the request command (MGB?).
  (4) Convert the read BLOCK WINDOW pattern data into page units and display it on the CRT.

↓

END

- Program list

```
10 !**********************************************************************
20 !*                                                                    *
30 !*    MP1762A/MP1764A BLOCK WINDOW PATTERN DATA DMA TRANSFER          *
40 !*                       SAMPLE PROGRAM                  ED_DMA2      *
50 !**********************************************************************
60 !
70 !
80 DIM Dta(31)
90 DIM Bit$(3)[255]
100 DIM Page$[32]
110!
120 Add=701                                  !MP1762A/MP1764A GPIB ADDRESS
130 CLEAR Add                                !DEVICE CLEAR
140 CLEAR SCREEN
150 !
160 PRINT " **            MP1762A/MP1764A                **  "
170 PRINT " **   BLOCK WINDOW DATA DMA TRANSFER          **  "
180 !
190 OUTPUT Add;"PTS 1"
200 OUTPUT Add;"PAG 1"
210 !
220 OUTPUT Add;"SYM 0"
230 !
240 OUTPUT Add;"DLN 512"
250 !
260 !*********************** DATA SET ************************
270 Dta(0)=1
280 Dta(1)=2
290 Dta(2)=4
300 Dta(3)=8
310 Dta(4)=16
320 Dta(5)=32
330 Dta(6)=64
340 Dta(7)=128
350 Dta(8)=256
360 Dta(9)=512
370 Dta(10)=1024
380 Dta(11)=2048
390 Dta(12)=4096
400 Dta(13)=8192
410 Dta(14)=16384
420 Dta(15)=32767
430 Dta(16)=-32768
440 Dta(17)=-16384
450 Dta(18)=-8192
460 Dta(19)=-4096
470 Dta(20)=-2048
480 Dta(21)=-1024
490 Dta(22)=-512
500 Dta(23)=-256
510 Dta(24)=-128
520 Dta(25)=-64
530 Dta(26)=-32
540 Dta(27)=-16
550 Dta(28)=-8
560 Dta(29)=-4
570 Dta(30)=-2
580 Dta(31)=-1
590 !
600 !
```

```
610 FOR I=1 TO 32
620      OUTPUT Add;"MWT 2,0"
630      !
640      OUTPUT Add USING "W";Dta(I-1)
650      !
660      GOSUB Page_ana
670      IMAGE "BLOCK WINDOW PAGE = ",XXX,AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA,XX,DDD
DDD
680      PRINT USING 670;Page$;Dta(I-1)
690      !
700 NEXT I
710 !
720 STOP
730 !
740 !********************* BLOCK WINDOW PAGE ANALYSIS ************************
750 Page_ana:!
760  !
770  FOR J=1 TO 4
780       !
790       OUTPUT Add;"PAG "&VAL$((J-1)*8+1)
800       OUTPUT Add;"MGB?"
810       ENTER Add;Bit$(J-1)
820       FOR K=1 TO 8
830            !
840            IF Bit$(J-1)[21+7*(K-1),24+7*(K-1)]="FFFF" THEN
850                 Page$[K+(J-1)*8,K+(J-1)*8]="1"
860            ELSE
870                 Page$[K+(J-1)*8,K+(J-1)*8]="0"
880            END IF
890       NEXT K
900 NEXT J
910 !
920 RETURN
930 !
940 END
```

Execution result

```
BLOCK WINDOW PAGE =     00000000000000001100000000000000          1
BLOCK WINDOW PAGE =     00000000000000000011000000000000          2
BLOCK WINDOW PAGE =     00000000000000000000110000000000          4
BLOCK WINDOW PAGE =     00000000000000000000001100000000          8
BLOCK WINDOW PAGE =     00000000000000000000000011000000         16
BLOCK WINDOW PAGE =     00000000000000000000000000110000         32
BLOCK WINDOW PAGE =     00000000000000000000000000001100         64
BLOCK WINDOW PAGE =     00000000000000000000000000000011        128
BLOCK WINDOW PAGE =     11000000000000000000000000000000        256
BLOCK WINDOW PAGE =     00110000000000000000000000000000        512
BLOCK WINDOW PAGE =     00001100000000000000000000000000       1024
BLOCK WINDOW PAGE =     00000011000000000000000000000000       2048
BLOCK WINDOW PAGE =     00000000110000000000000000000000       4096
BLOCK WINDOW PAGE =     00000000001100000000000000000000       8192
BLOCK WINDOW PAGE =     00000000000011000000000000000000      16384
BLOCK WINDOW PAGE =     11111111111111001111111111111111      32767
BLOCK WINDOW PAGE =     00000000000000110000000000000000     -32768
BLOCK WINDOW PAGE =     00000000000011110000000000000000     -16384
BLOCK WINDOW PAGE =     00000000001111110000000000000000      -8192
BLOCK WINDOW PAGE =     00000000111111110000000000000000      -4096
BLOCK WINDOW PAGE =     00000011111111110000000000000000      -2048
BLOCK WINDOW PAGE =     00001111111111110000000000000000      -1024
BLOCK WINDOW PAGE =     00111111111111110000000000000000       -512
BLOCK WINDOW PAGE =     11111111111111110000000000000000       -256
BLOCK WINDOW PAGE =     11111111111111110000000000000011       -128
BLOCK WINDOW PAGE =     11111111111111110000000000001111        -64
BLOCK WINDOW PAGE =     11111111111111110000000000111111        -32
BLOCK WINDOW PAGE =     11111111111111110000000011111111        -16
BLOCK WINDOW PAGE =     11111111111111110000001111111111         -8
BLOCK WINDOW PAGE =     11111111111111110000111111111111         -4
BLOCK WINDOW PAGE =     11111111111111110011111111111111         -2
BLOCK WINDOW PAGE =     11111111111111111111111111111111         -1
```

10.2 Example of Program creation Using DECpc

<Explanation of common section of the program>

The following sample programs are created using Microsoft Quick Basic Ver 4.50 and the GPIB interface card of National Instrument. (☞ Refer to the instruction manuals of Quick Basic and GPIB driver for details.)

The necessary common functions in the sample program are summarized in the two programs below.

- COMMON.BAS
- ACS_GPIB.BAS

These two programs must be prepared when the sample programs are executed.

Also, only the necessary functions may be prepared.

The two kinds of common functions are described the following pages.

<COMMON.BAS>

COMMON.BAS consists of seven types of functions.

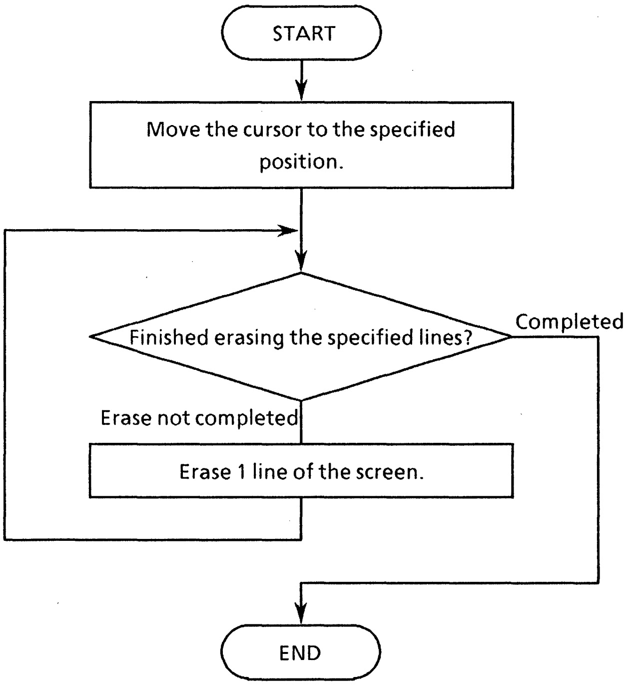
Table 10-2 Table of COMMON.BAS Functions

Module No.	Function	Processing
1.1	SUB ClearDisp (p%, l%)	Erases screen in units of line. p% : Start line number for erase l% : Number of lines to be erased
1.2	SUB Connect (ttl$)	Displays the connection diagram ttl$: Character string of title which is displayed together with connection diagram.
1.3	FUNC Exchange% (i%)	The upper and lower bytes of data having a bit pattern of a single precision integer are exchanged in byte units. i% : Bit pattern data
1.4	FUNC itob$ (l%, v%)	Single precision integer is converted into a binary character string of bit length which is specified by LSB. However, output character length is fixed at 16 characters. l% : Binary character string length v% : Conversion data
1.5	SUB SelItem (dist$())	Performs key enter processing for selected measurement item. dist$ () : Argument character array (output)
1.6	SUB Displ ()	Displays the data which has been read by the command set by SelItem ().
1.7	SUB waidly (tim)	Waits for the specified period of time. tim : Specified time (seconds) (input)

Each functions and its flowchart are shown on the following pages.

(1.1) SUB ClearDisp (p%, 1%): Erases screen.

- Flowchart

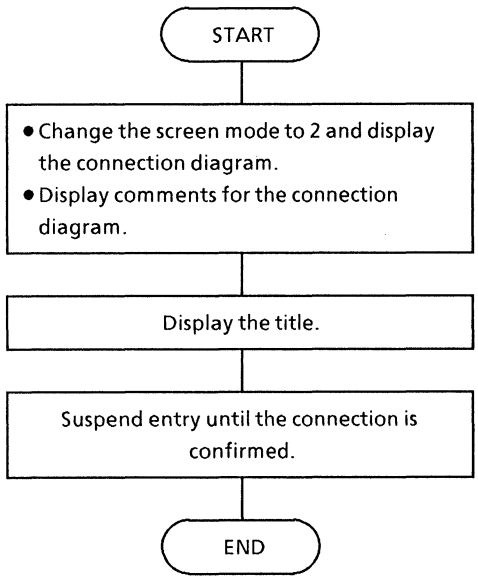


- Program list

```
' ---- Procedure for Clear display ----
' in    p%:Location line number
'       l%:clear line count
'
SUB ClearDisp (p%, l%)
    LOCATE p%, 1
    FOR i% = 0 TO (l% - 1)
        PRINT "
          "
    NEXT i%
END SUB
```

(1.2) SUB Connect (tt1$): Displays the title and connection diagram.

- Flowchart

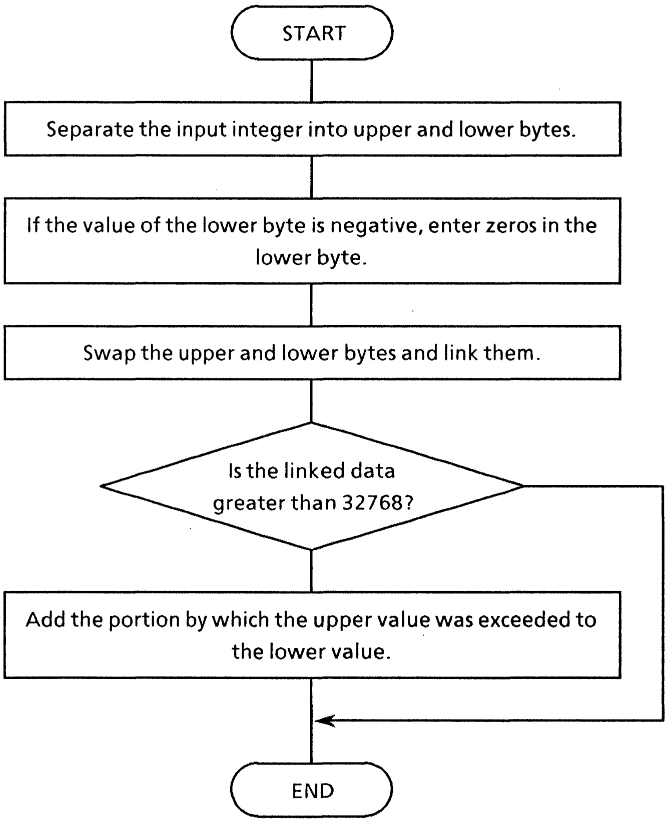


- Program list

```
' ---- Connection layout ----
' in    ttl$:Title message for method
'
' This is drawing connection line layout for MP1762A with other PPG.
'
SUB Connect (ttl$)
    CLS
    SCREEN 2
    WINDOW (-600, -500)-(600, 500)

    LINE (-50, -50)-(600, 500), 14, B
    LINE (10, 100)-STEP(235, 200), , B
    LINE (300, 100)-STEP(235, 200), , B
    '
    CIRCLE (130, 130), 8
    LINE (130, 125)-STEP(0, -55)
    LINE (130, 70)-STEP(220, 0)
    LINE (350, 70)-STEP(0, 55)
    CIRCLE (350, 130), 8
    '
    CIRCLE (190, 130), 8
    LINE (190, 125)-STEP(0, -100)
    LINE (190, 25)-STEP(220, 0)
    LINE (410, 25)-STEP(0, 100)
    CIRCLE (410, 130), 8
    '
    LOCATE 2, 45: PRINT "<<  CONNECTION Layout  >>"'
    LOCATE 4, 42: PRINT "MP1761A/MP1763A     MP1762A/MP1764A"
    LOCATE 9, 45: PRINT "DATA CLOCK1"'
    LOCATE 9, 62: PRINT "DATA CLOCK"
    '
    LOCATE 1, 1: PRINT ttl$
    LOCATE 22, 1: PRINT "You must confirm connection line."
    LOCATE 23, 1: INPUT "Aer You ready to start? Press 'Enter' to continue.", a
    '
    LOCATE 22, 1: PRINT "                                                     "
    LOCATE 23, 1: PRINT "                                                     "
END SUB
```

(1.3) FUNCTION Exchange (i%): Swaps 16-bit integer data in units of byte.

- Flowchart

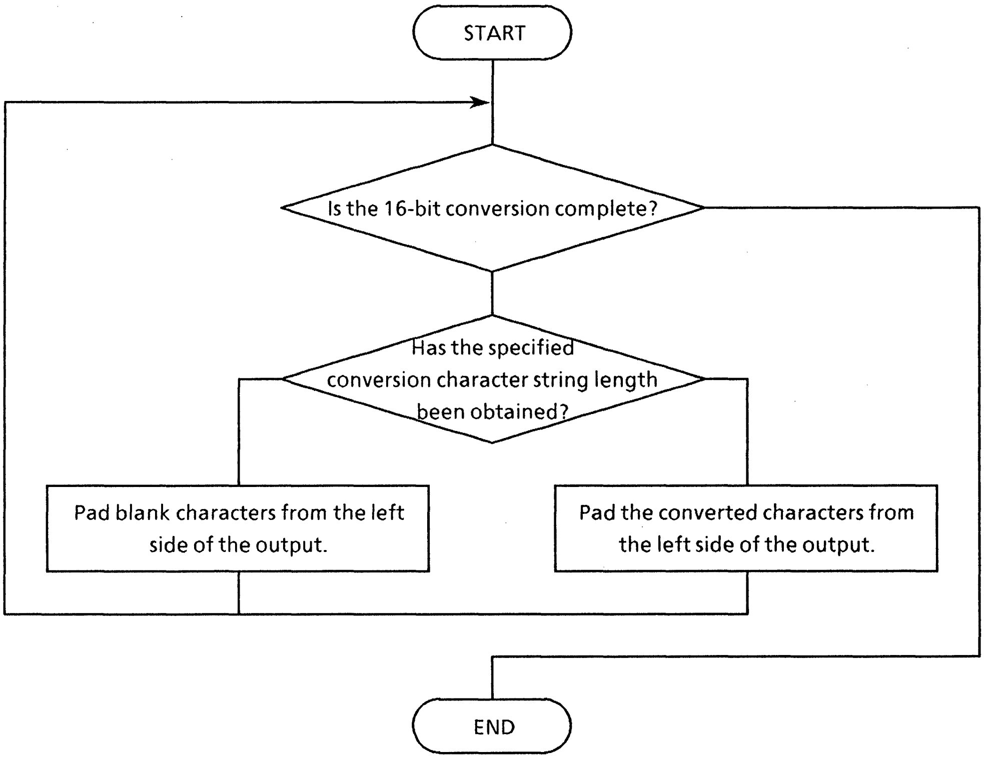


- Program list

```
' ---- Exchange 16-bits pattern data ----
' In   i%:16bits pattern data (used integer)
'
' Procedure for swap of low byte and high byte .
' This program is bit manipulation of integer value. Why this program used
' real value because one is overflow detect on bit manipulation of integer
' value, another one is internal manipulation by real value although input
' parameter is integer. And integer declear value is same operation.
'
FUNCTION Exchange% (i%)
    h = i% AND &HFF
    l = i% AND &HFF00
    IF h < 0 THEN
        h = 0
    END IF

    a = INT(h * 256) + ((l ¥ 256) AND &HFF)
    IF a >= 32768 THEN
        b = a - 32768
        a = -32768 + b
    END IF
    Exchange% = a
END FUNCTION
```

(1.4) itob$ (l%, v%): Converts integers into binary character strings.

- Flowchart

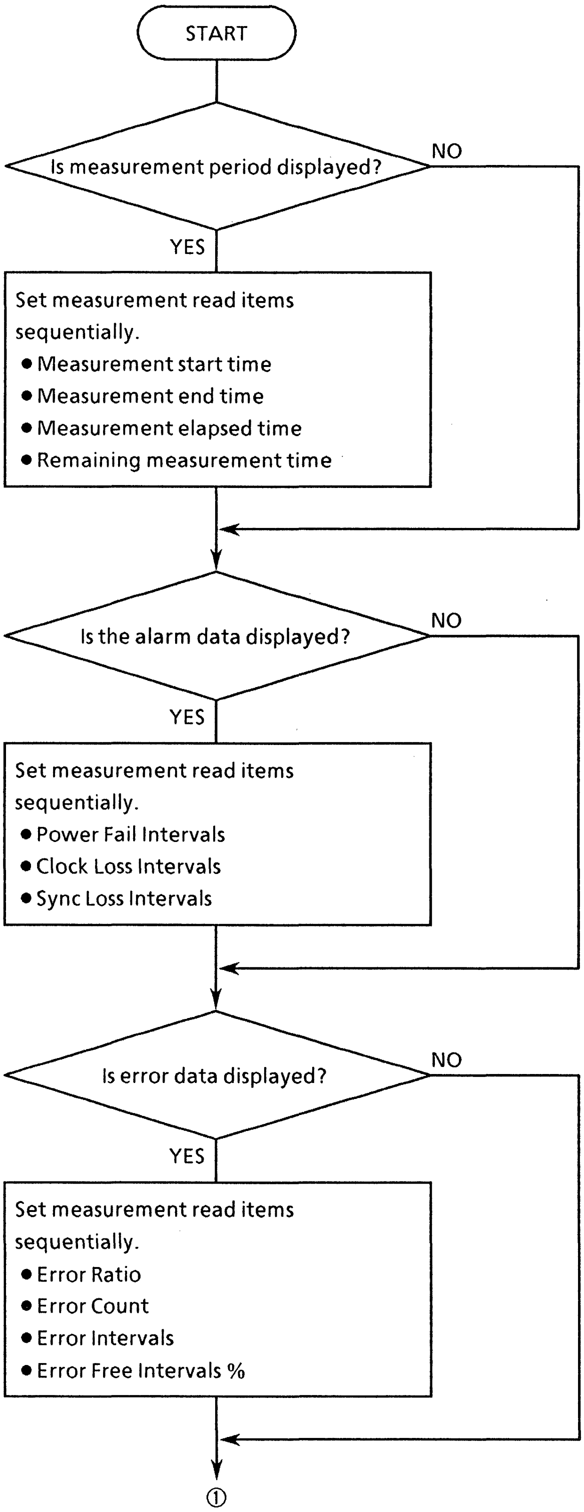


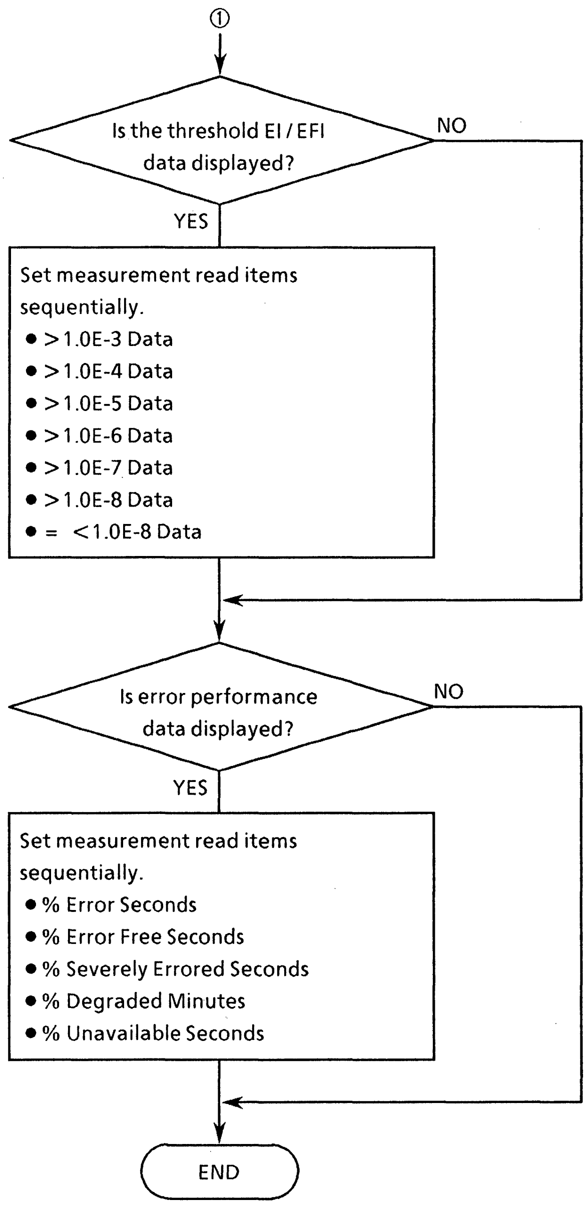
- Program list

```
' ---- Interger convert to binary strings ----
' in    l%:convert binary length
'       v%:convert value
'
' This convert binary strings is always possession 16-character field.
'
FUNCTION itob$ (l%, v%)
b$ = ""
FOR i% = 1 TO 16
    IF i% <= l% THEN
        IF v% AND &H1 THEN
            b$ = "1" + b$
        ELSE
            b$ = "0" + b$
        END IF
        v% = INT(v% / 2)
    ELSE
        b$ = " " + b$
    END IF
NEXT i%
itob$ = b$
END FUNCTION
```

(1.5) SelItem (dst$): Read outs measurement results, and sets command arguments.

- Flowchart



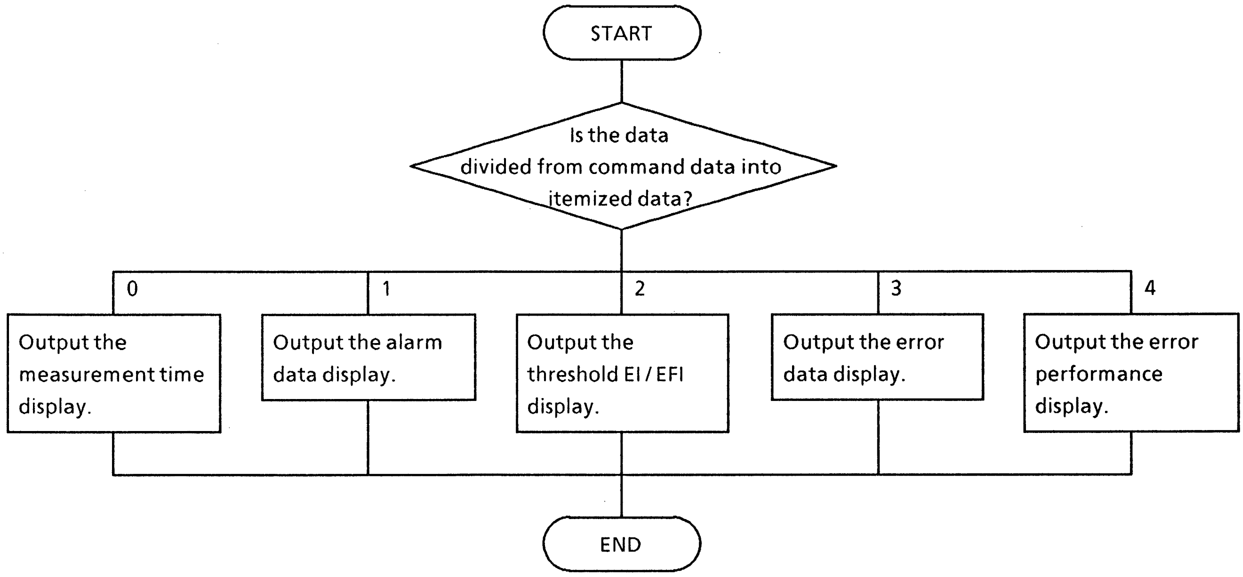

①
Is the threshold EI / EFI data displayed?
NO
YES
Set measurement read items sequentially.
● >1.0E-3 Data
● >1.0E-4 Data
● >1.0E-5 Data
● >1.0E-6 Data
● >1.0E-7 Data
● >1.0E-8 Data
● = <1.0E-8 Data
Is error performance data displayed?
NO
YES
Set measurement read items sequentially.
● % Error Seconds
● % Error Free Seconds
● % Severely Errored Seconds
● % Degraded Minutes
● % Unavailable Seconds
END

- Program list

```
SUB SelItem (dst$())
LOCATE 16, 1                  '======== Select measurement item(s) ========
PRINT "Choose item for measure time.                "
LOCATE 17, 1
INPUT "Do you wish output measure TIME data?    [Yes/No]:"; a$
IF a$ = "y" OR a$ = "Y" THEN
    dst$(0, 0) = "0,1"
    dst$(0, 1) = "0,2"
    dst$(0, 2) = "0,3"
    dst$(0, 3) = "0,4"
END IF

CALL ClearDisp(16, 2)
LOCATE 16, 1
PRINT "Choose item for alarm data                   "
LOCATE 17, 1
INPUT "Do you wish output ALARM data?           [Yes/No]:"; a$
IF a$ = "y" OR a$ = "Y" THEN
    dst$(1, 0) = "1,1"
    dst$(1, 1) = "1,2"
    dst$(1, 2) = "1,3"
END IF

CALL ClearDisp(16, 2)
LOCATE 16, 1
PRINT "Choose item for ERROR measurement data. "
LOCATE 17, 1
INPUT "Do you wish output ERROR data?           [Yes/No]:"; a$
IF a$ = "y" OR a$ = "Y" THEN
    dst$(2, 0) = "2,1"
    dst$(2, 1) = "2,2"
    dst$(2, 2) = "2,3"
    dst$(2, 3) = "2,4"
END IF

CALL ClearDisp(16, 2)
LOCATE 16, 1
PRINT "Choose item for THRESHOLD EI/EFI data.  "
LOCATE 17, 1
INPUT "Do you wish output THR. EI/EFI data ?    [Yes/No]:"; a$
IF a$ = "y" OR a$ = "Y" THEN
    dst$(3, 0) = "3,1"
    dst$(3, 1) = "3,2"
    dst$(3, 2) = "3,3"
    dst$(3, 3) = "3,4"
    dst$(3, 4) = "3,5"
    dst$(3, 5) = "3,6"
    dst$(3, 6) = "3,7"
END IF

CALL ClearDisp(16, 2)
LOCATE 16, 1
PRINT "Choose item for ERROR PERFORMANCE data. "
LOCATE 17, 1
INPUT "Do you wish output EP data data ?        [Yes/No]:"; a$
IF a$ = "y" OR a$ = "Y" THEN
    dst$(4, 0) = "4,1"
    dst$(4, 1) = "4,2"
    dst$(4, 2) = "4,3"
    dst$(4, 3) = "4,4"
    dst$(4, 4) = "4,5"
END IF

 PRINT " -- End select measurement item -- "
 END SUB
```

(1.6) Disp1 (cmd$, dt$): Displays the read measurement results by item.

- Flowchart



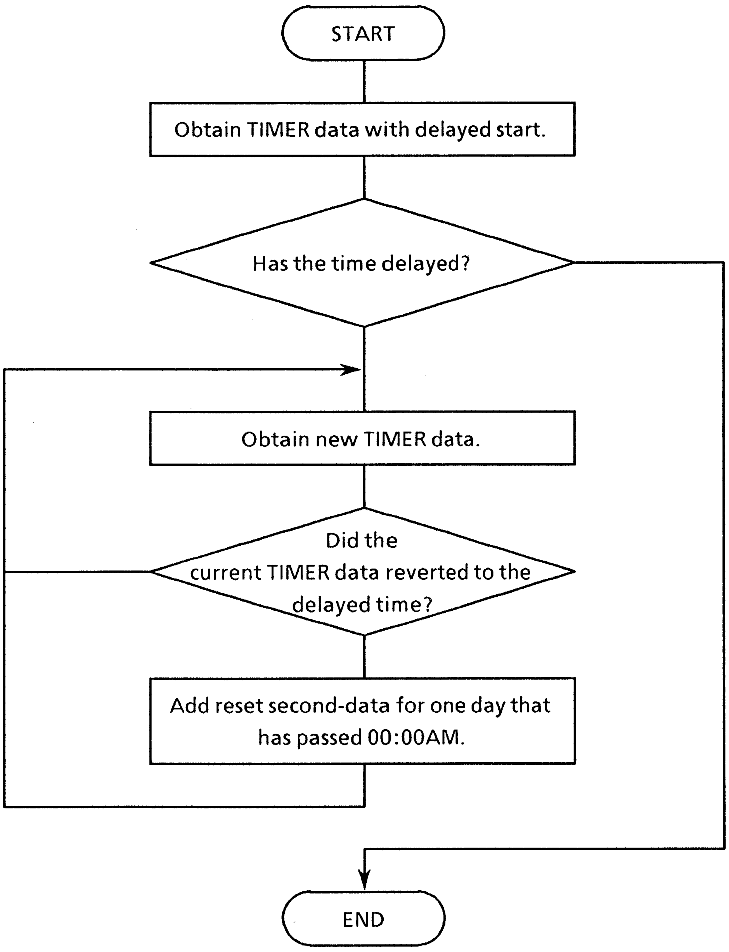
- Program list

```
' ---- Procedure for Measure result display ----
' in    cmd$:parameter strings of query
'       dt$ :response message from MP1762A/MP1764A(ED)
'
SUB Displ (CMD$, DT$)
    unit$ = ""
    SELECT CASE VAL(MID$(CMD$, 1, 1))
    CASE 0                              ' Time data
        SELECT CASE VAL(MID$(CMD$, 3, 1))
        CASE 1
            ttl$ = "Start time                  :"
        CASE 2
            ttl$ = "Stop time                   :"
        CASE 3
            ttl$ = "Elapsed time                :"
        CASE 4
            ttl$ = "Remain time                 :"
        END SELECT
    CASE 1                              ' Alarm data
        SELECT CASE VAL(MID$(CMD$, 3, 1))
        CASE 1
            ttl$ = "Power Fail Intervals        :"
        CASE 2
            ttl$ = "Clock Loss Intervals        :"
        CASE 3
            ttl$ = "Sync Loss Intervals         :"
        END SELECT
    CASE 2                              ' Error data
        SELECT CASE VAL(MID$(CMD$, 3, 1))
        CASE 1
            ttl$ = "Error Ratio                 :"
        CASE 2
            ttl$ = "Error Count                 :"
        CASE 3
            ttl$ = "Error Intervals             :"
        CASE 4
            ttl$ = "Error Free Intervals%       :"
            unit$ = "%"
        END SELECT
    CASE 3                              ' Threshold EI/EFI data
        unit$ = "%"
        SELECT CASE VAL(MID$(CMD$, 3, 1))
        CASE 1
            ttl$ = "Threshold EI/EFI  >1.0E-3:"
        CASE 2
            ttl$ = "Threshold EI/EFI  >1.0E-4:"
        CASE 3
            ttl$ = "Threshold EI/EFI  >1.0E-5:"
        CASE 4
            ttl$ = "Threshold EI/EFI  >1.0E-6:"
        CASE 5
            ttl$ = "Threshold EI/EFI  >1.0E-7:"
        CASE 6
            ttl$ = "Threshold EI/EFI  >1.0E-8:"
        CASE 7
            ttl$ = "Threshold EI/EFI =<1.0E-8:"
        END SELECT
    CASE 4                              ' Error Performance data
        unit$ = "%"
        SELECT CASE VAL(MID$(CMD$, 3, 1))
        CASE 1
            ttl$ = "Error Performance %ES       :"
        CASE 2
```

```
            ttl$ = "Error Performance %EFS    :"
        CASE 3
            ttl$ = "Error Performance %SES    :"
        CASE 4
            ttl$ = "Error Performance %DM     :"
        CASE 5
            ttl$ = "Error Performance %US     :"
        END SELECT
    END SELECT
    IF DT$ = "ERR" THEN
        DT$ = " No data "
    END IF
    PRINT ttl$ + DT$ + unit$ + "   "
END SUB
```

(1.7) waidly (tim!): Creates the wait time

- Flowchart



- Program list

```
' ---- Make a timing delay ----
'in    tim:wait time length (unit is seconds)
'
SUB waidly (tim)
    stm = TIMER
    etm = TIMER
    WHILE etm - stm < tim
        etm = TIMER
        IF etm < stm THEN etm = etm + 86400
    WEND
END SUB
```

<Explanation of ACS_GPIB.BAS>

ACS_PGIB.BAS consists of the following 15 types of functions.

Table 10-3 ACS_GPIB.BAS Functions (1/2)

Module number	Function	Processing
2.1	SUB wrtcmd1 (w$)	Send commands to PPG. w$: Command character string to be sent (input)
2.2	SUB wrtcmd2 (w$)	Send commands to ED. w$: Command character string to be sent (input)
2.3	FUNC readcmd2$ ()	Reads messages from ED. readcmd2$: Message character string (returned value)
2.4	SUB dmawrt (w% ( ), i%)	Transfers in DMA to ED. w% ( ): Integer array of pattern data to be transferred i% : Number of elements of integer array
2.5	SUB EndPoll ( )	Performs polling of the END bit of the MSS status register.
2.6	SUB SRQPoll ( )	Performs polling of the ERROR bit and SRQ bit of the MSS status register.
2.7	SUB StatusMask (sre%, ese%, ese2%, ese3%)	Sets the mask pattern for the status, event, and expansion registers. sre% : Mask pattern for status register ese% : Mask pattern for standard event register ese2%: Mask pattern for expansion event register 2 ese3%: Mask pattern for expansion event register 3
2.8	SUB StatusDisp (stb%, esr%, esr2%, esr3%)	Displays the setting status of the status, event, and expansion registers. The read data is specified as an argument and sent to the calling side. stb% : Pattern of status register setting status esr% : Pattern of standard event register setting status esr2% : Pattern of expansion event register 2 setting status esr3% : Pattern of expansion event register3 setting status
2.9	SUB MeasStart ( )	Starts measurement.
2.10	SUB MeasStop ( )	Stops or terminates measurement.
2.11	SUB ChecClk ( )	Judges the CLOCK LOSS and holds the processing until recovery.

Table 10-3 ACS_GPIB.BAS Functions (2 / 2)

Module number	Function	Processing
2.12	FUNC AutoSrc% ( )	Executes the Auto Search operation and returns the results as function values. 0 (False) : GPIB initialization failed due to wrong setting. 1 (True) : Either one or both ED and PPG completed initialization.
2.13	FUNC gpinit% ( )	Executes GPIB initialization and returns the initialization as function values. 0 (False) : Error in setting. Initialization failed. 1 (True) : ED or PPG, or both completed initialization.
2.14	SUB trap ( )	Processes system errors.
2.15	SUB gpiberr ( )	Processes internal errors included in the GPIB sample program provided by National Instruments, displays status information.

Flowcharts of each function and program lists are described in the following pages.

The following must be entered at the header of the module:

```
REM $INCLUDE:'C:¥at-gbib¥qbasic¥qbdecl.bas' ........ ①
COMMON SHARED DEV%,GPIB0%,PPG%,ED% .................... ②
```

Item ① loads the NI-488 function definition using the GPIB driver of National Instruments.

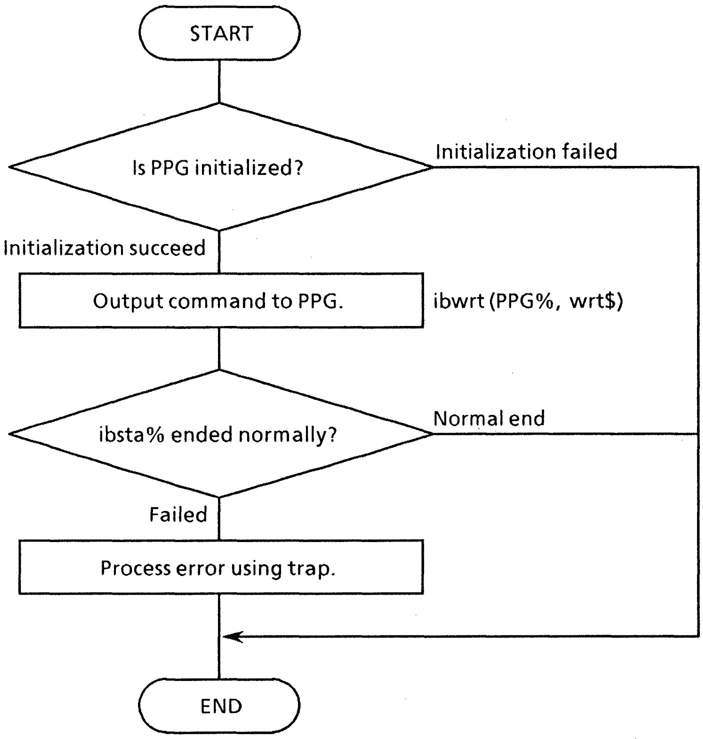
In actual use, specify a directory including 'qbdecl.bas'.

Item ② is a Quick Basic statement which defines the common variables between multiple modules.

■ **Note** : For item ①, note that the GPIB varies with the environment used.

(2.1) SUB wrtcmd1 (w$): Sends commands to PPG.

- Flowchart

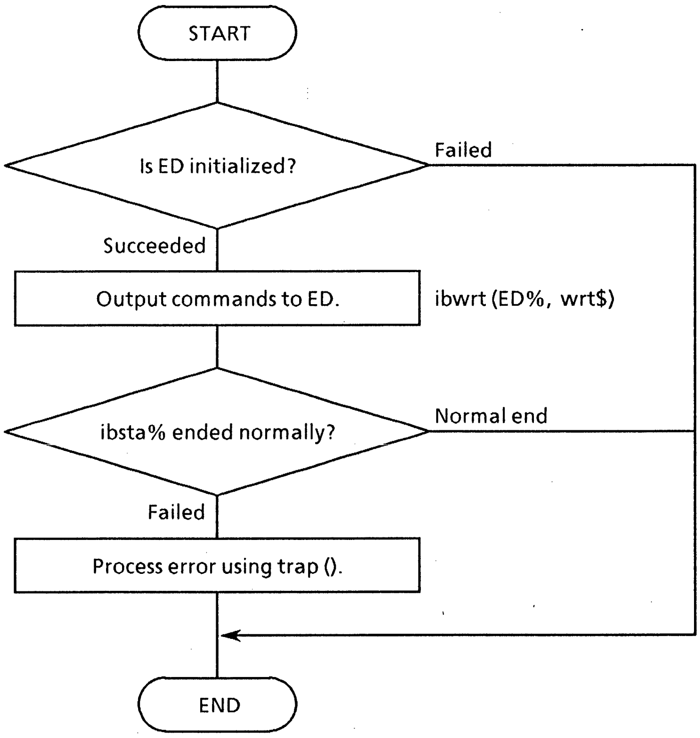


- Program list

```
' ---- Procedure for command write to PPG ----
'
SUB wrtcmd1 (WRT$)
    IF DEV% = 1 OR DEV% = 3 THEN
        WRT$ = WRT$ + CHR$(13) + CHR$(10)
        CALL IBWRT(PPG%, WRT$)              'write command(ppg)
        IF IBSTA% < 0 THEN CALL trap        'call trap if illegal end
    END IF
END SUB
```

(2.2) SUB wrtcmd2 (w$): Sends commands to ED.

- Flowchart

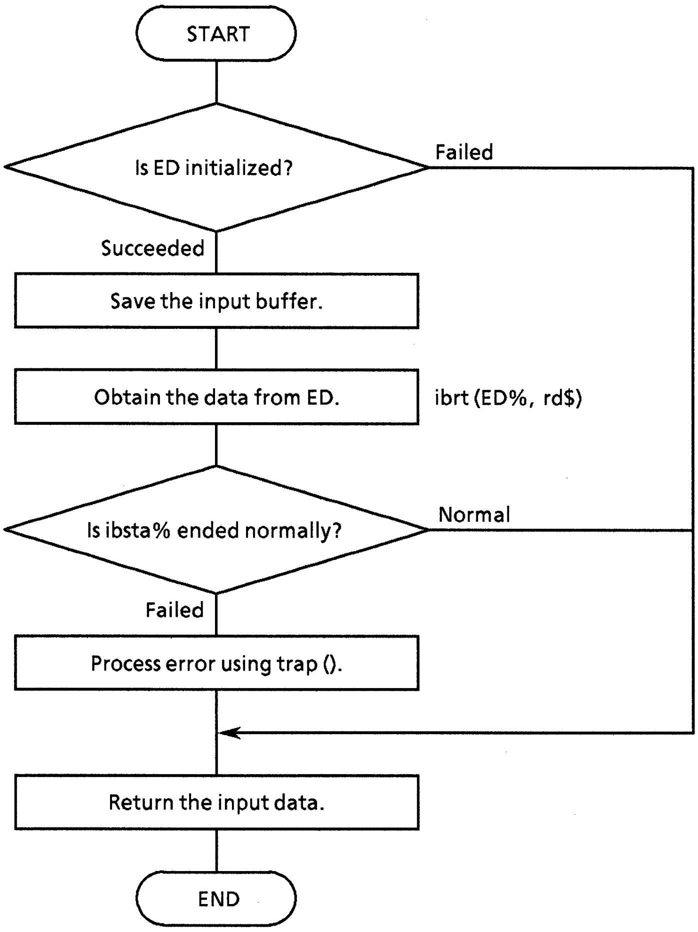


- Program list

```
' ---- Procedure for command write to ED ----
'
SUB wrtcmd2 (WRT$)
    IF DEV% = 2 OR DEV% = 3 THEN
        WRT$ = WRT$ + CHR$(13) + CHR$(10)
        CALL IBWRT(ED%, WRT$)              ' write command to ED
        IF IBSTA% < 0 THEN CALL trap       ' call trap if illegal end
    END IF
END SUB
```

(2.3) FUNCTION readcmd2$ (): Obtains data in response to the command sent from ED separately.

- Flowchart

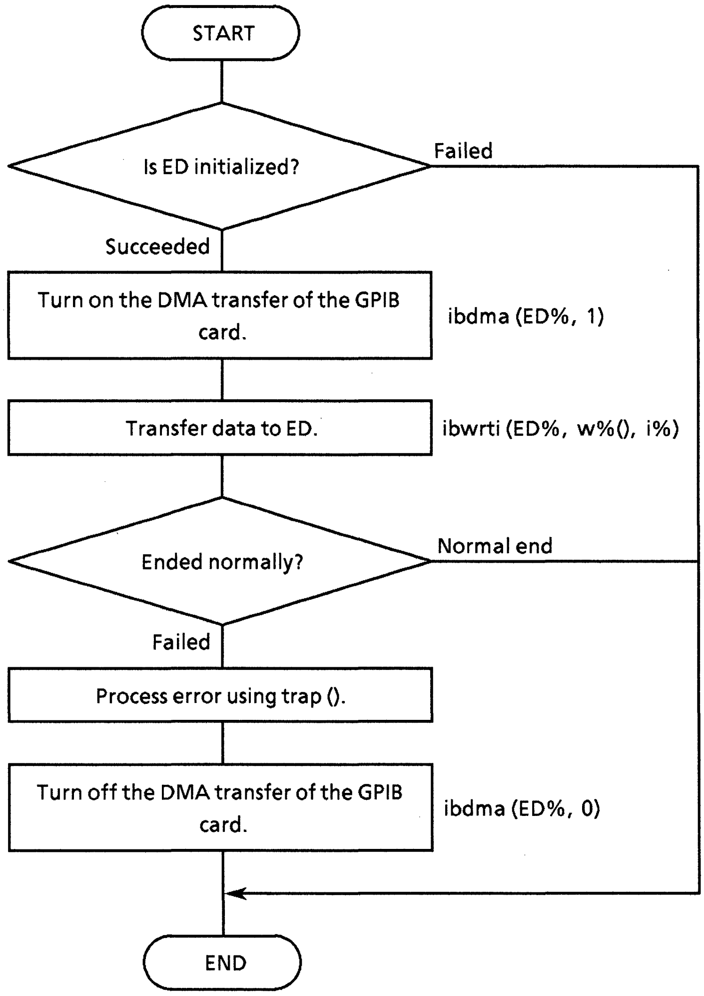


- Program list

```
' ---- Procedure for data read from ED ----
'
FUNCTION readcmd2$
    IF DEV% = 2 OR DEV% = 3 THEN
        r$ = SPACE$(256)
        CALL IBRD(ED%, r$)                ' Read data from ED%
        IF IBSTA% < 0 THEN CALL trap
'
        readcmd2$ = r$
    END IF
END FUNCTION
```

(2.4) SUB dmawrt (w%, i%): Transfers the ED data in DMA transfer.

- Flowchart

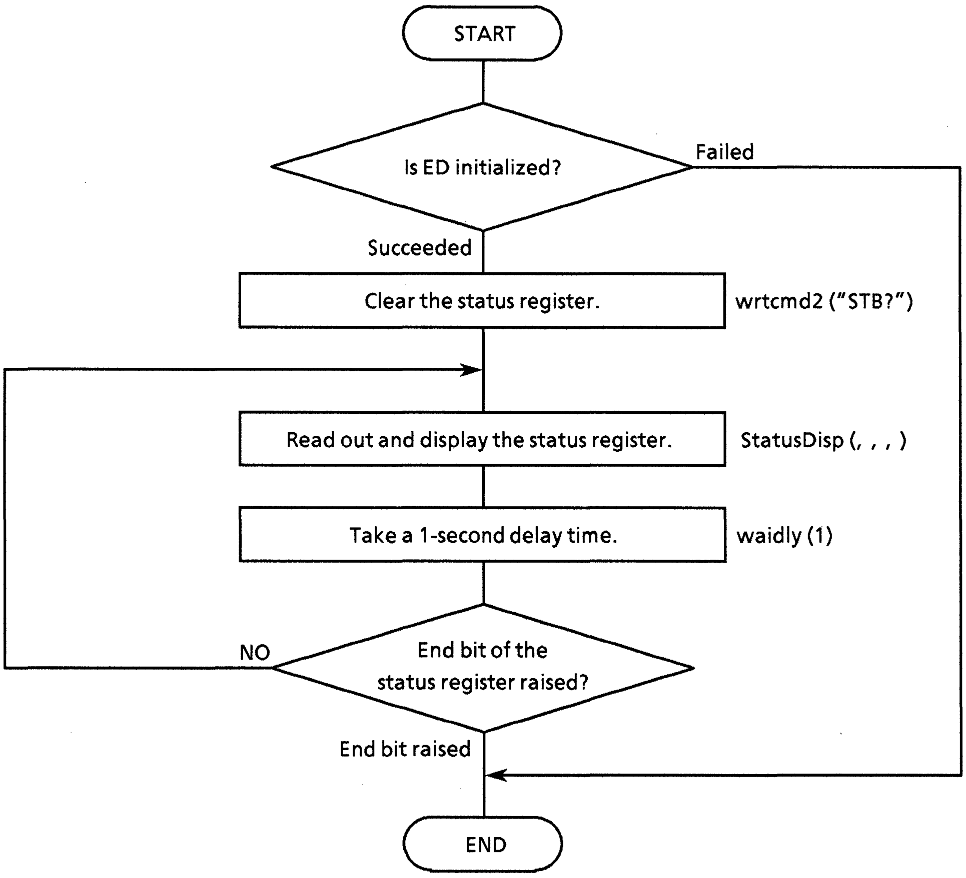


- Program list

```
' --- Procedure for DMA transfer ----
' in    w%():Transmit data pattern of integer arry
'       i   :length count for integer arry
'
SUB dmawrt (w%(), i%)
    IF DEV% = 2 OR DEV% = 3 THEN
        CALL IBDMA(ED%, 1)          ' DMA enable

        i% = i% * 2 + 1             ' make up to a byte count
        CALL IBWRTI(ED%, w%(), i%)
        IF IBSTA% < 0 THEN CALL trap ' call trap if illegal end

        CALL IBDMA(ED%, 0)          ' DMA disable
    END IF
END SUB
```

(2.5) SUB EndPoll (): Waits until the status end bit is set.

- Flowchart

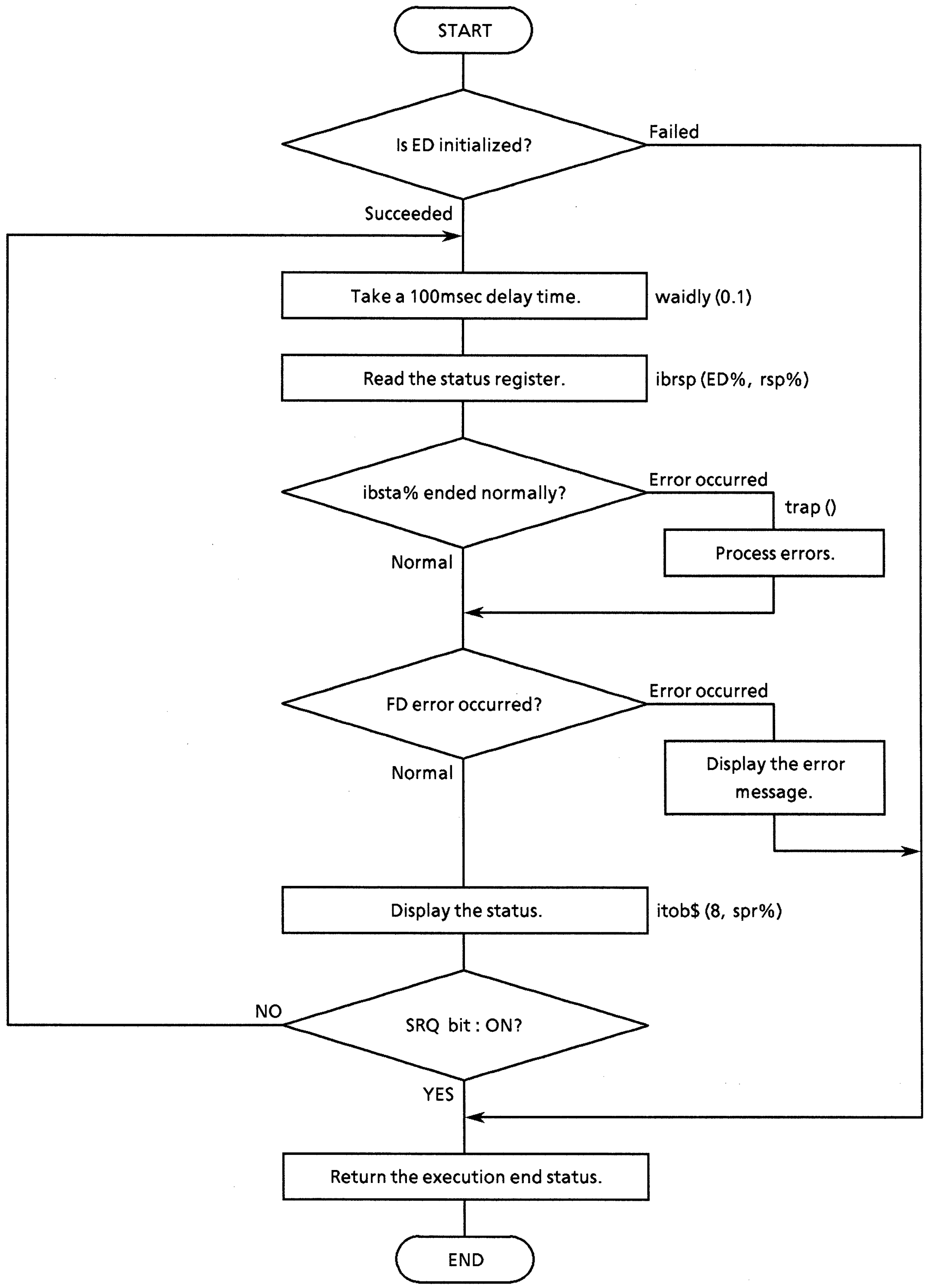


- Program list

```
' ---- Procedure for judgement of Measurement end ----
'
SUB EndPoll
    IF DEV% = 2 OR DEV% = 3 THEN
        CALL wrtcmd2("*STB?")           ' reset event flag
        RD$ = LEFT$(readcmd2$, IBCNT% - 1)

        DO
            CALL StatusDisp(reg%, dmy%, dmy2%, dmy3%)

            waidly (1)
        LOOP UNTIL reg% AND &H4
    END IF
END SUB
```

(2.6) FUNCTION SRQPoll (): Judges SRQ and error bits.

- Flowchart

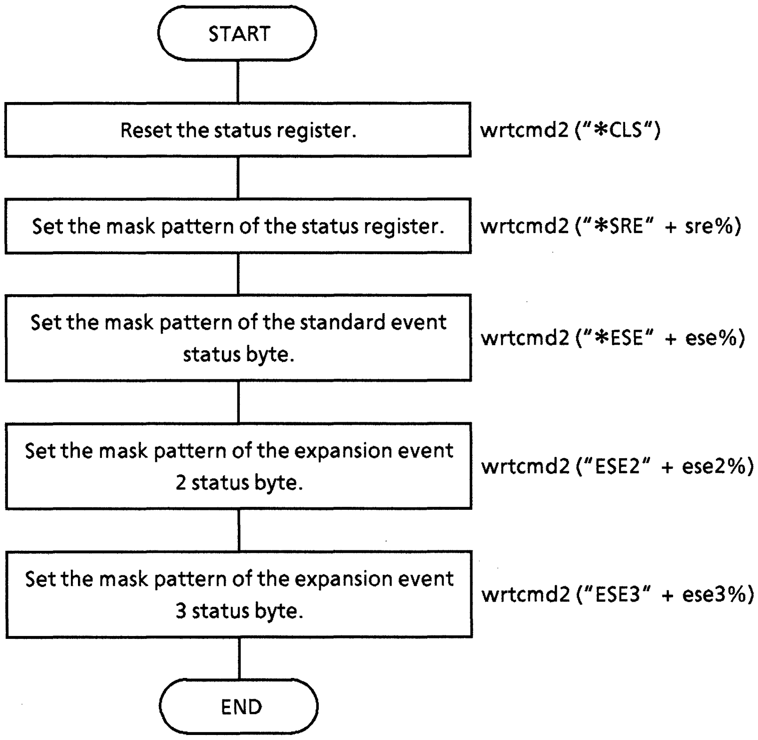


- Program list

```
' ---- Procedure for Seliall poll with SRQ bit ----
'
FUNCTION SRQPoll%
    IF DEV% = 2 OR DEV% = 3 THEN
        exe% = 1
        DO
            waidly (.1)

            CALL IBRSP(ED%, SPR%)
            IF IBSTA < 0 THEN CALL trap
            srq = SPR% AND &H40
            esrl = SPR% AND &H4
            esr2 = SPR% AND &H8

            IF esr2 = &H8 THEN          ' Output warning message, if error detect
                LOCATE 12, 35
                PRINT "FD error detect!!"
                exe% = 0
                EXIT DO
            END IF

            sta$ = itob$(8, SPR%)
            LOCATE 1, 60
            PRINT "*STB:"; sta$
        LOOP UNTIL srq = &H40 AND ersl = &H0
    END IF
    '
    SRQPoll% = exe%
END FUNCTION
```

(2.7) SUB StatusMask (sre%, ese%, ese2%, ese3%): Sets status registers.

- Flowchart

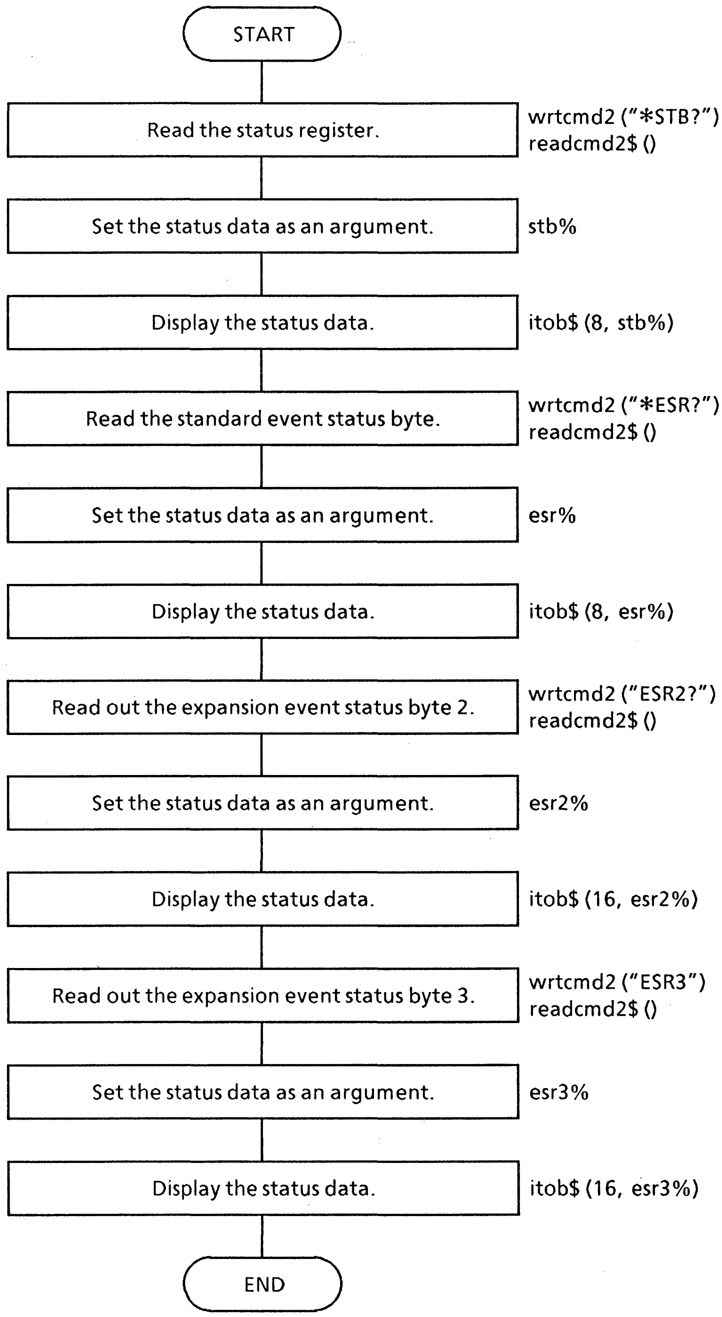


- Program list

```
' ---- Procedure for set status mask pattern ----
' in     s0%:status byte enable register mask pattern
'        s1%:normal event status enable register mask pattern
'        s2%:Extend event status enable register-2 mask pattern
'        s3%:Extend event status enable register-3 mask pattern
'
SUB StatusMask (s0%, s1%, s2%, s3%)
    wrtcmd2 ("*CLS")
    wrtcmd2 ("*SRE " + STR$(s0%))
    wrtcmd2 ("*ESE " + STR$(s1%))
    wrtcmd2 ("ESE2 " + STR$(s2%))
    wrtcmd2 ("ESE3 " + STR$(s3%))
END SUB
```

(2.8) SUB StatusDisp (stb%, esr%, esr2%, esr3%): Reads out and displays the status register.

- Flowchart

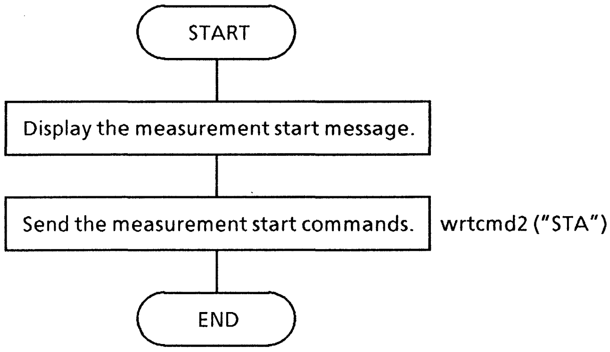


- Program list

```
' ---- Procedure for status byte display ----
' out   stb% :Status byte
'       esr% :Normal event status byte
'       esr2%:Extend event-2 status byte
'       esr3%:Extend event-3 status byte
'
SUB StatusDisp (stb%, esr%, esr2%, esr3%)
    CALL wrtcmd2("*STB?")
    RD$ = LEFT$(readcmd2$, IBCNT% - 1)
    stb% = VAL(RD$)
    sta$ = itob$(8, VAL(RD$))
    LOCATE 1, 60
    PRINT "*STB:"; sta$

    CALL wrtcmd2("*ESR?")
    RD$ = LEFT$(readcmd2$, IBCNT% - 1)
    esr% = VAL(RD$)
    sta$ = itob$(8, VAL(RD$))
    LOCATE 2, 60
    PRINT "*ESR:"; sta$

    CALL wrtcmd2("ESR2?")
    RD$ = LEFT$(readcmd2$, IBCNT% - 1)
    esr2% = VAL(MID$(RD$, 6, 5))
    sta$ = itob$(16, VAL(MID$(RD$, 6, 5)))
    LOCATE 3, 60
    PRINT "ESR2:"; sta$

    CALL wrtcmd2("ESR3?")
    RD$ = LEFT$(readcmd2$, IBCNT% - 1)
    esr3% = VAL(MID$(RD$, 6, 5))
    sta$ = itob$(16, VAL(MID$(RD$, 6, 5)))
    LOCATE 4, 60
    PRINT "ESR3:"; sta$
END SUB
```

(2.9) SUB MeasStart ()

- Flowchart

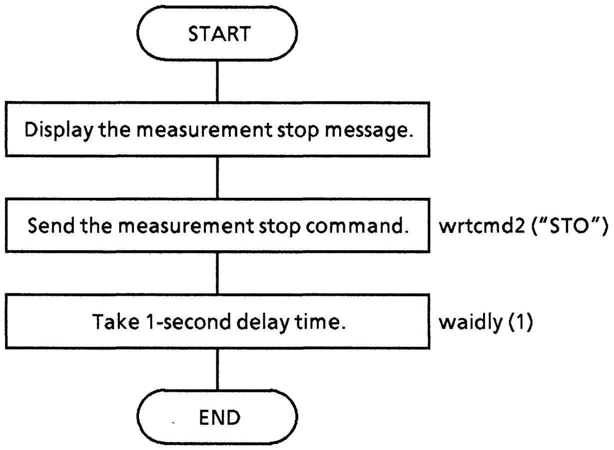


- Program list

```
' ---- Procedure for Measurement start ----
'
SUB MeasStart
    CLS
    LOCATE 1, 1
    PRINT "***** Measure START *****
    CALL wrtcmd2("STA")
END SUB
```

(2.10) SUB MeasStop ()

- Flowchart

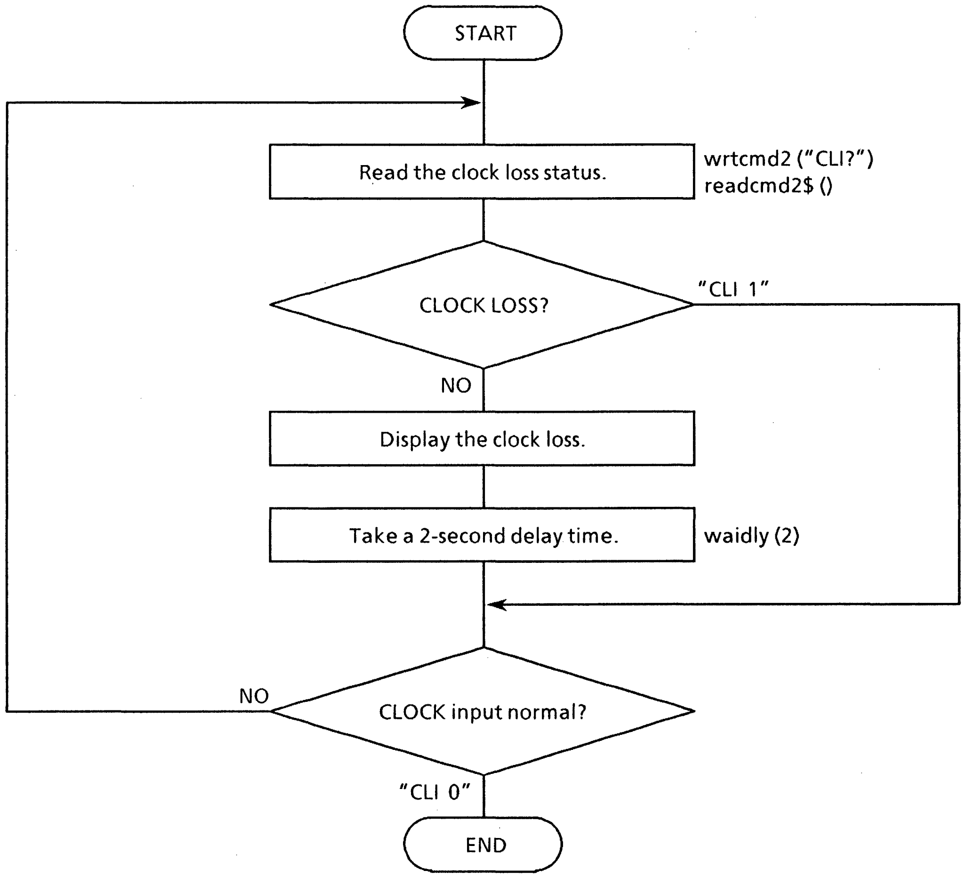


- Program list

```
' ---- Procedure for Measurement stop ----
'
SUB MeasStop
    LOCATE 1, 1
    PRINT "***** Measure STOP *****
    CALL wrtcmd2("STO")
    waidly (1)
END SUB
```

(2.11) SUB ChecClk (): Checks the clock loss status.

- Flowchart

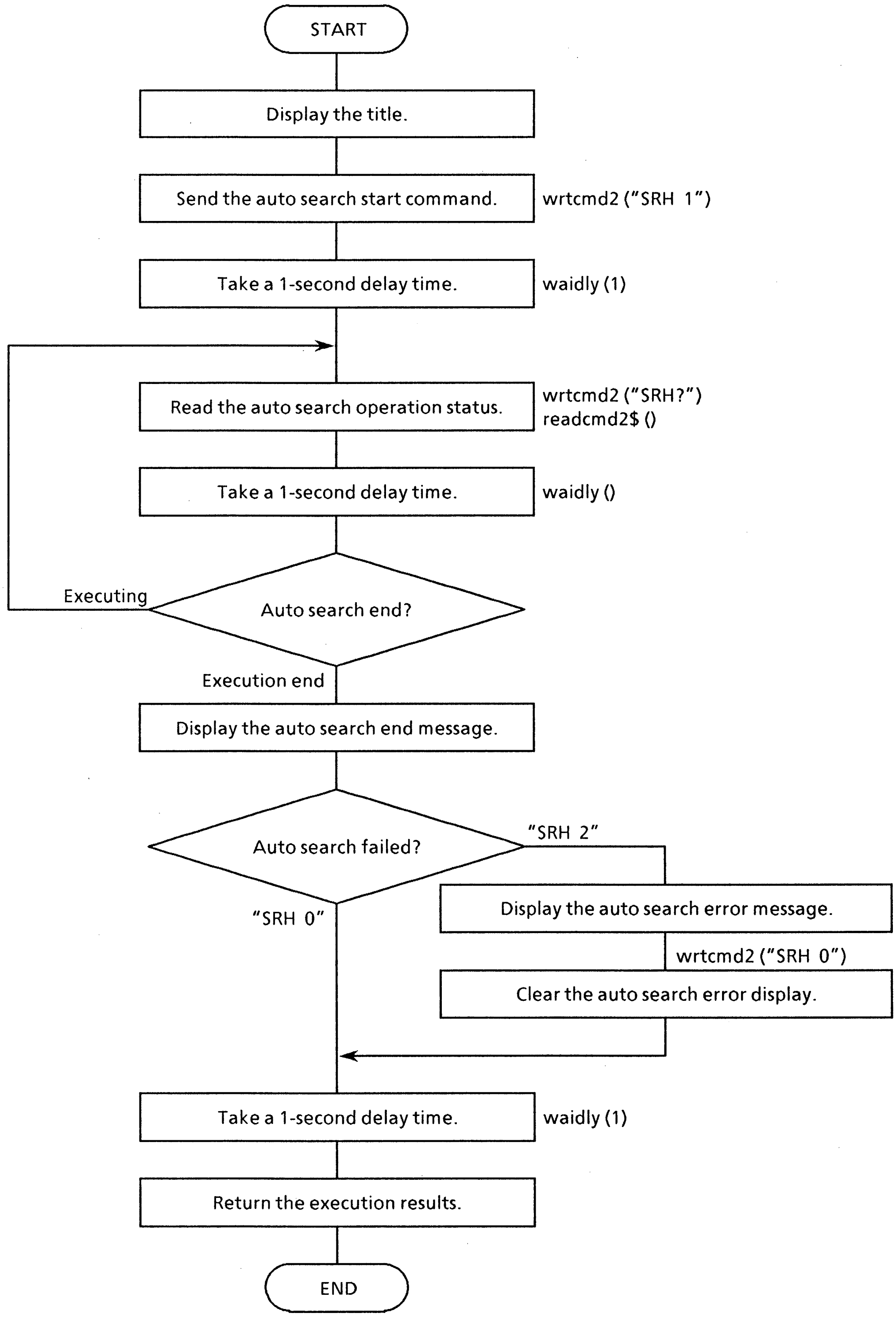


- Program list

```
' ---- Procedure for clock status ----
' This program is loop until to clock detect.
'
SUB ChecClk
    DO
        CALL wrtcmd2("CLI?")
        RD$ = LEFT$(readcmd2$, IBCNT% - 1)
        IF MID$(RD$, 1, 5) = "CLI 1" THEN
            LOCATE 4, 1
            PRINT "** CLOCK LOSS **"
            waidly (2)
        END IF
    LOOP UNTIL MID$(RD$, 1, 5) = "CLI 0"
    LOCATE 4, 1
    PRINT "                "
END SUB
```

(2.12) FUNCTION AutoSrc% ()

- Flowchart

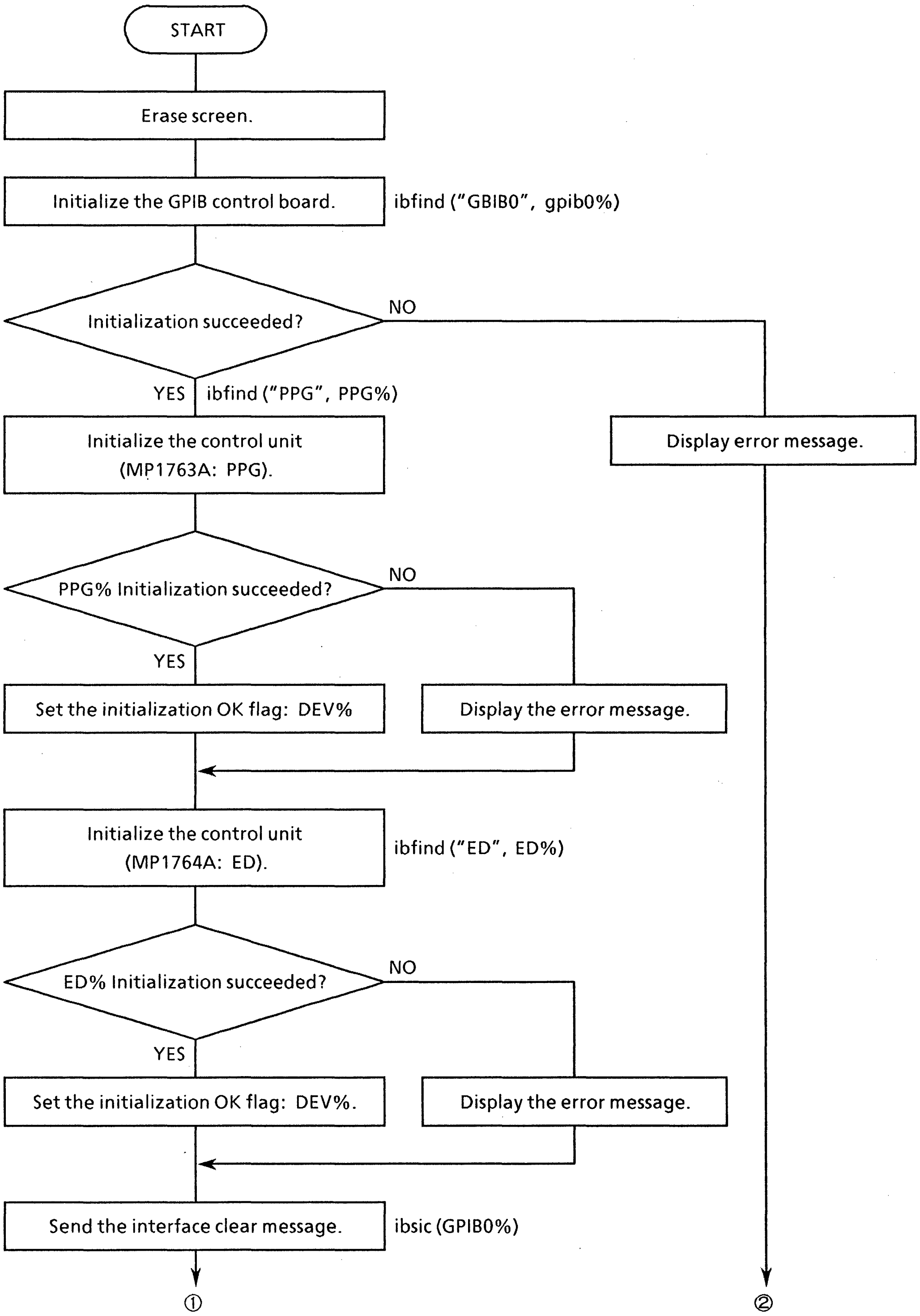


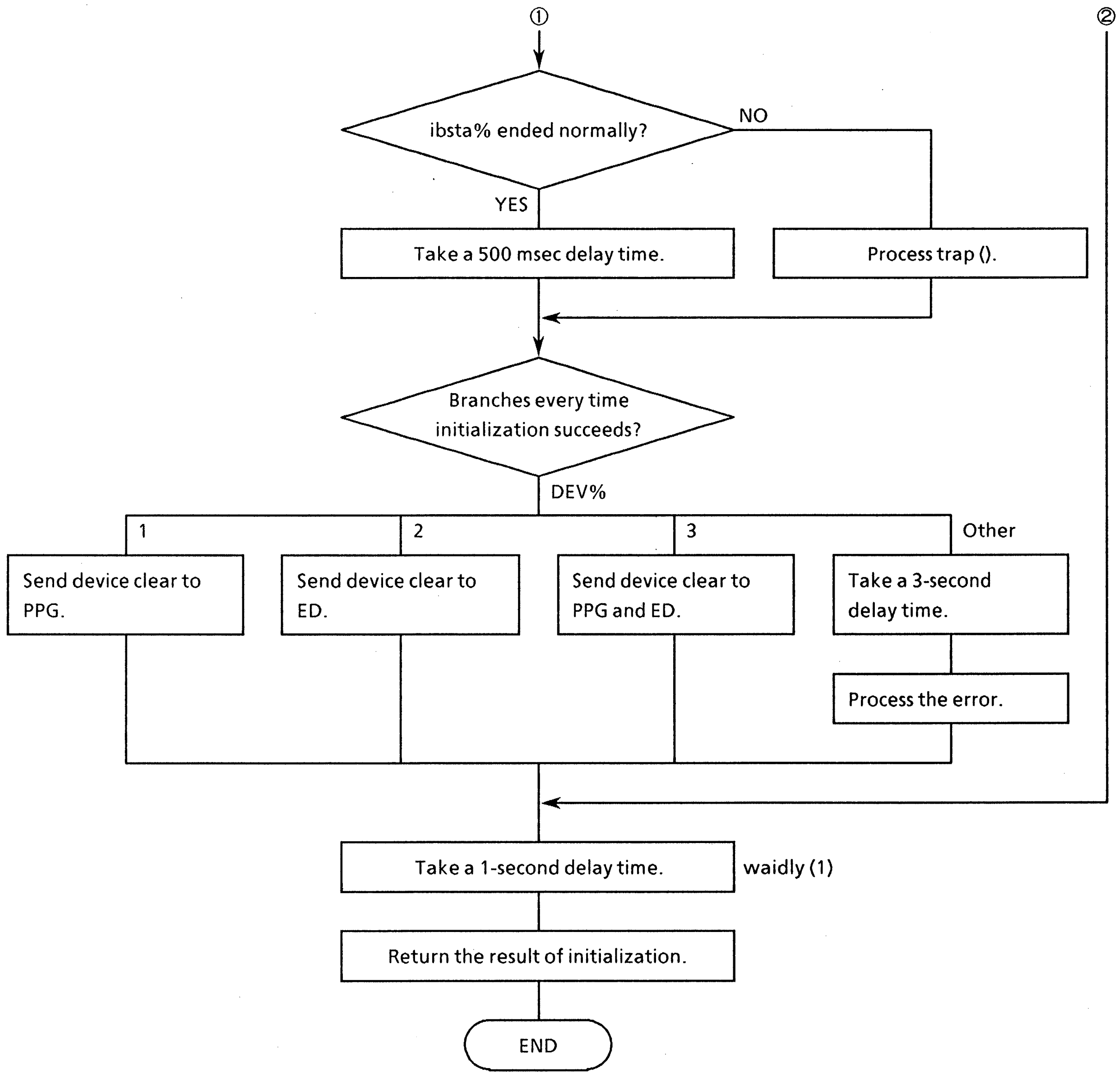
- Program list

```
' ---- Procedure for Auto Search ----
' out   AutoSrc%:Auto Search execution status
'               0(false):illegal termination
'               1(true) :normal end
'
FUNCTION AutoSrc%
    LOCATE 4, 1
    PRINT "*** Auto Search START ***"

    '========  Auto search ON =========
    CALL wrtcmd2("SRH 1")
    waidly (1)

    '===========  Polling  ============
    DO
        CALL wrtcmd2("SRH?")
        RD$ = LEFT$(readcmd2$, IBCNT% - 1)
        waidly (1)
    LOOP UNTIL MID$(RD$, 1, 5) = "SRH 0" OR MID$(RD$, 1, 5) = "SRH 2"
    rsl% = 1                         ' Auto Search success

    LOCATE 4, 1
    PRINT "*** Finish Auto Search ***"

    '======== Fail Auto Search ========
    IF MID$(RD$, 1, 5) = "SRH 2" THEN
        PRINT "<<< Failed on AUTO SEARCH ! >>>"
        CALL wrtcmd2("SRH 0")
        rsl% = 0                      ' Auto Search fail
    END IF
    waidly (1)

    AutoSrc% = rsl%
END FUNCTION
```

(2.13) FUNCTION gpinit (): Initializes the GPIB control environment.

- Flowchart



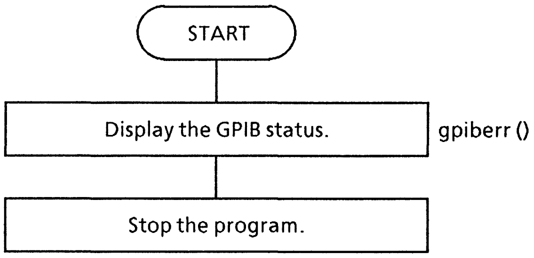

①
②
ibsta% ended normally?
NO
YES
Take a 500 msec delay time.
Process trap ().
Branches every time initialization succeeds?
DEV%
1
2
3
Other
Send device clear to PPG.
Send device clear to ED.
Send device clear to PPG and ED.
Take a 3-second delay time.
Process the error.
Take a 1-second delay time.
waidly (1)
Return the result of initialization.
END

- Program list

```
' ---- Procedure for initialize equipments and interface board ----
'
FUNCTION gpinit%
    CLS
    CALL IBFIND("GPIB0", GPIB0%)        'Open DEVice (GPIB0)
    IF GPIB0% < 0 THEN
        PRINT "Configration fail!!"
        PRINT "You need verify are hardware condition, and try again."
        ret% = 0
    ELSE
        CALL IBFIND("PPG", PPG%)        'Open DEVice (PPG)
        IF PPG% < 0 THEN
            PRINT "Lost PPG address!!"
            PRINT "If you use a PPG, then verify are configration and environmen
t."
            DEV% = 0
        ELSE
            DEV% = 1
        END IF
        '
        CALL IBFIND("ED", ED%)          'Open DEViec (ED)
        IF ED% < 0 THEN
            PRINT "Lost ED address!!"
            PRINT "If you use a PPG, then verify are configration and environmen
t."
        ELSE
            IF DEV% = 0 THEN
                DEV% = 2
            ELSE
                DEV% = 3
            END IF
        END IF
        '
        CALL IBSIC(GPIB0%)              'Interface clear
        IF IBSTA% < 0 THEN CALL trap
        '
        CALL waidly(.5)                 '500ms wait
        '
        SELECT CASE DEV%
        CASE 1
            CALL IBCLR(PPG%)            'DEVice clear (PPG)
        CASE 2
            CALL IBCLR(ED%)             'DEVice clear (ED)
        CASE 3
            CALL IBCLR(PPG%)            'DEVice clear (PPG)
            CALL IBCLR(ED%)             'DEVice clear (ED)
        CASE ELSE
            waidly (3)
            CALL trap
        END SELECT
        ret% = 1
    END IF

    waidly (1)
    CLS
    gpinit% = ret%                          ' set Execution status
END FUNCTION
```

(2.14) SUB trap (msg$): Processes errors.

- Flowchart

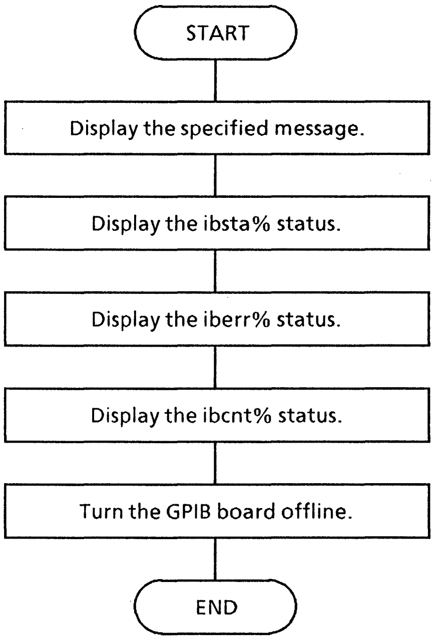


- Program list

```
' ---- Procedure for illegal process trap ----
' This subroutine, call on illegal execution or fatal error detect.
' And, you will get are status condition by presented NI-488 function.
'
SUB trap
    CALL gpiberr("Program trap condition.")  ' call NI subroutine
    STOP
END SUB
```

(2.15) SUB gpiberr (msg$): Displays the STATIC: GPIB status.

- Flowchart



- Program list

```
'===========================================================================
'                         Subroutine GPIBERR
'  This subroutine will notify you that a NI-488 function failed by printing
'  an error message.  The status variable IBSTA% will also be printed
'  in hexadecimal along with the mnemonic meaning of the bit position.
'  The status variable IBERR% will be printed in decimal along with the
'  mnemonic meaning of the decimal value.  The status variable IBCNT% will
'  be printed in decimal.
'
'  The NI-488 function IBONL is called to disable the hardware and software.
'
'  The STOP command will terminate this program.
'===========================================================================
'
SUB gpiberr (msg$) STATIC

   PRINT msg$

   PRINT "ibsta = &H"; HEX$(IBSTA%); " <";
   IF IBSTA% AND EERR THEN PRINT " ERR";
   IF IBSTA% AND TIMO THEN PRINT " TIMO";
   IF IBSTA% AND EEND THEN PRINT " END";
   IF IBSTA% AND SRQI THEN PRINT " SRQI";
   IF IBSTA% AND RQS THEN PRINT " RQS";
   IF IBSTA% AND SPOLL THEN PRINT " SPOLL";
   IF IBSTA% AND EEVENT THEN PRINT " EVENT";
   IF IBSTA% AND CMPL THEN PRINT " CMPL";
   IF IBSTA% AND LOK THEN PRINT " LOK";
   IF IBSTA% AND RREM THEN PRINT " REM";
   IF IBSTA% AND CIC THEN PRINT " CIC";
   IF IBSTA% AND AATN THEN PRINT " ATN";
   IF IBSTA% AND TACS THEN PRINT " TACS";
   IF IBSTA% AND LACS THEN PRINT " LACS";
   IF IBSTA% AND DTAS THEN PRINT " DTAS";
   IF IBSTA% AND DCAS THEN PRINT " DCAS";
   PRINT " >"

   PRINT "iberr = "; IBERR%;
   IF IBERR% = EDVR THEN PRINT " EDVR <DOS Error>"
   IF IBERR% = ECIC THEN PRINT " ECIC <Not CIC>"
   IF IBERR% = ENOL THEN PRINT " ENOL <No Listener>"
   IF IBERR% = EADR THEN PRINT " EADR <Address error>"
   IF IBERR% = EARG THEN PRINT " EARG <Invalid argument>"
   IF IBERR% = ESAC THEN PRINT " ESAC <Not Sys Ctrlr>"
   IF IBERR% = EABO THEN PRINT " EABO <Op. aborted>"
   IF IBERR% = ENEB THEN PRINT " ENEB <No GPIB board>"
   IF IBERR% = EOIP THEN PRINT " EOIP <Async I/O in prg>"
   IF IBERR% = ECAP THEN PRINT " ECAP <No capability>"
   IF IBERR% = EFSO THEN PRINT " EFSO <File sys. error>"
   IF IBERR% = EBUS THEN PRINT " EBUS <Command error>"
   IF IBERR% = ESTB THEN PRINT " ESTB <Status byte lost>"
   IF IBERR% = ESRQ THEN PRINT " ESRQ <SRQ stuck on>"
   IF IBERR% = ETAB THEN PRINT " ETAB <Table Overflow>"

   PRINT "ibcnt = "; IBCNT%

   Call the IBONL function to disable the hardware and software.

   CALL IBONL(dvm%, 0)
END SUB
```

<Program start>

The procedures used to process the above common functions and to start the sample programs (1) to (13) are described below.

(Procedure 1) : Open File from the menu bar and select "Load File. . . . ".
Next, load the common function file name COMMON.BAS.

(Procedure 2) : Load ACS__GPIB.BAS in the same way as in procedure 1.

(Procedure 3) : Load the sample program in the same way as in procedure 1.

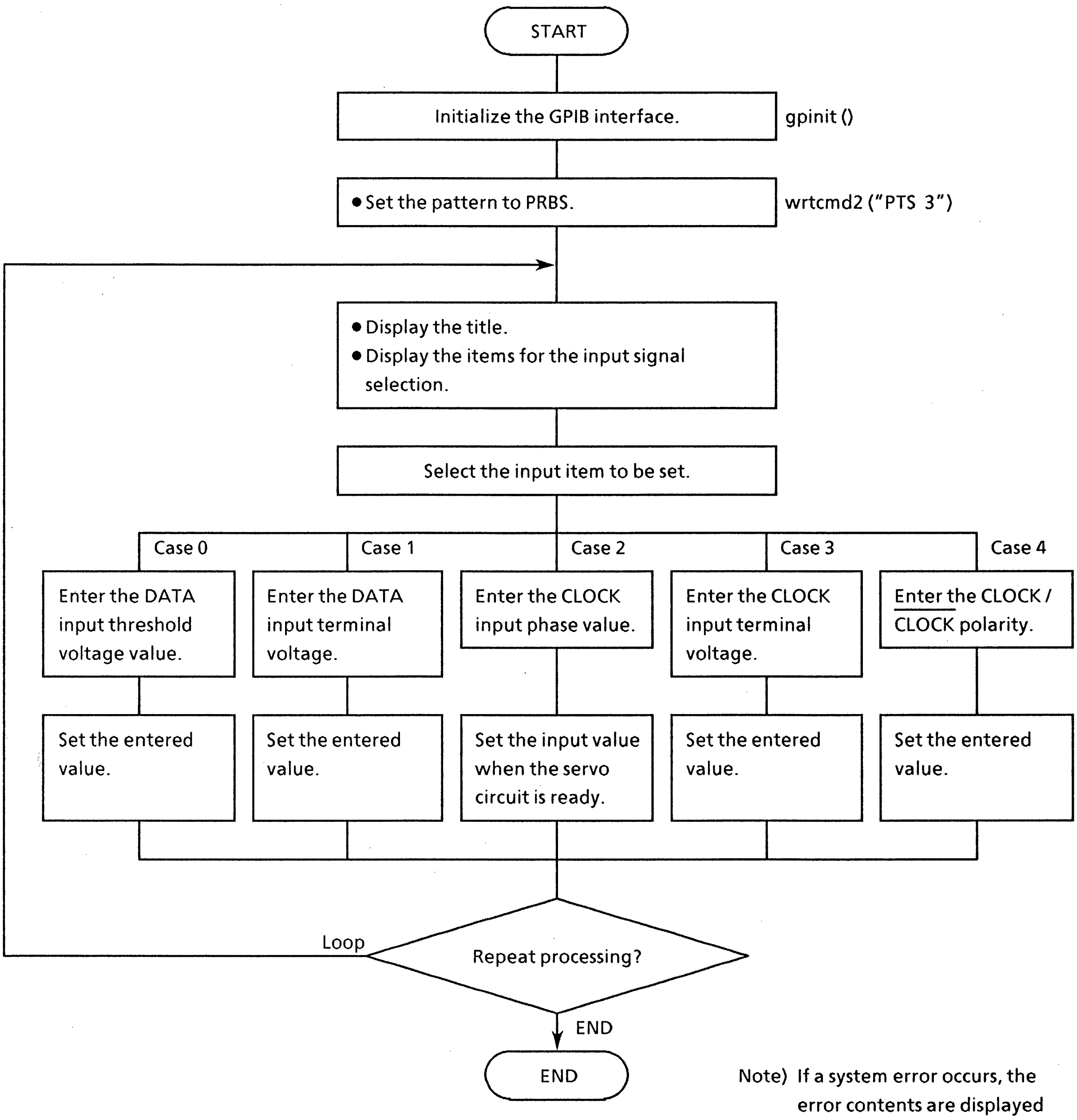
(Procedure 4) : Open Run from the menu bar, and select "Set Main Module", then make the sample program loaded in procedure 3 the main module.

(Procedure 5) : Open Run from the menu bar and execute "Start".

(Refer to the Quick Basic Instruction Manual for details.)

(1) Input signal setting

This program controls the input signals of the GPIB.



Note) If a system error occurs, the error contents are displayed and the program stops.

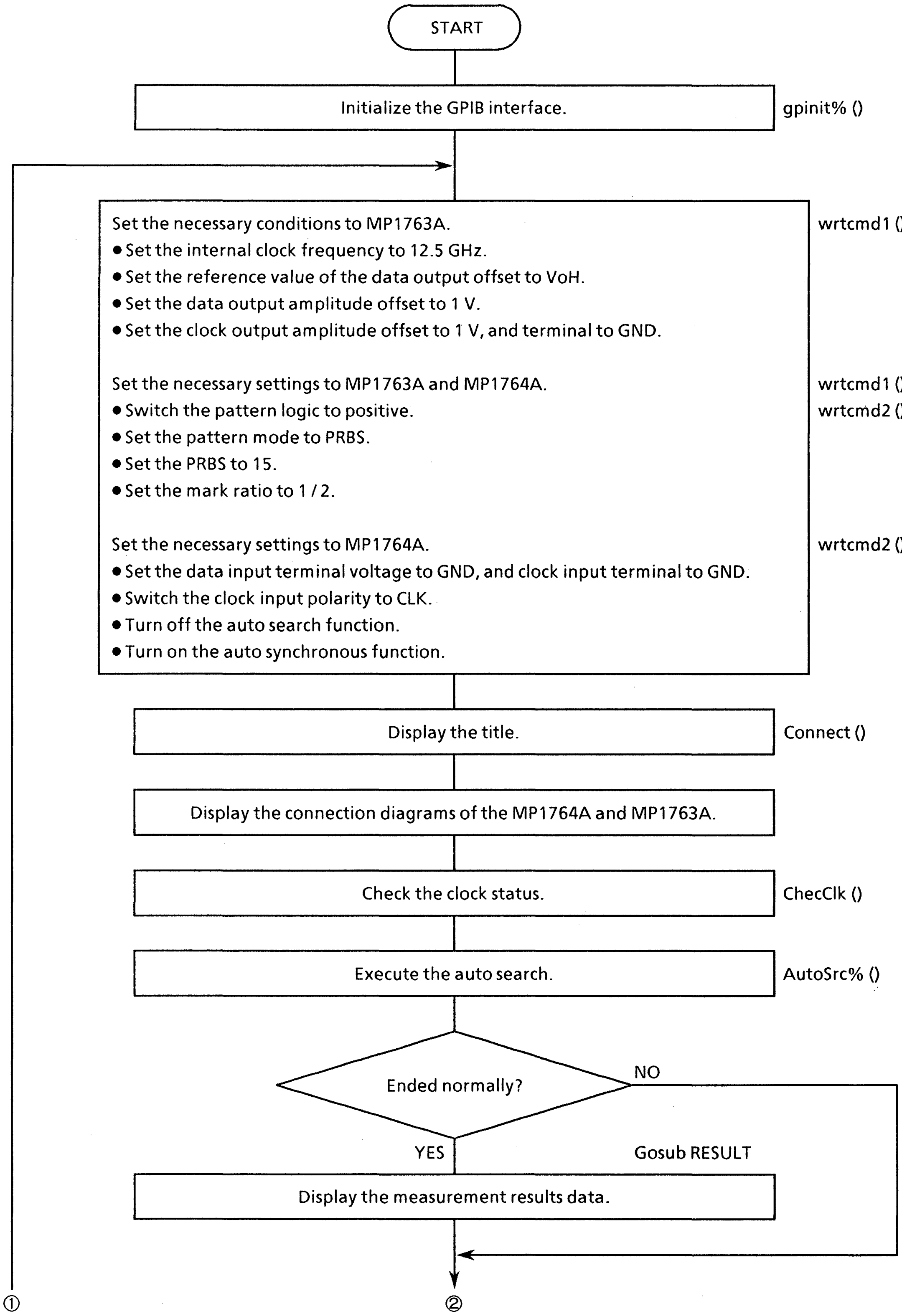
- Program list

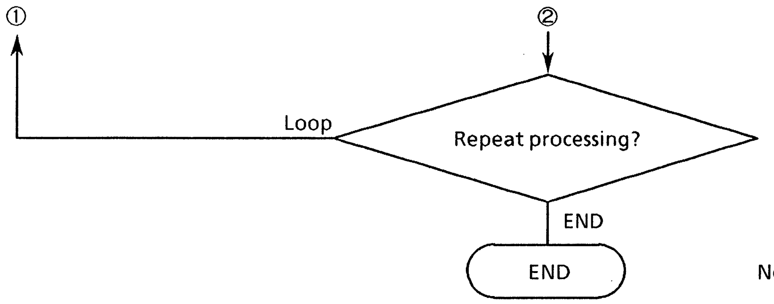
```
REM $INCLUDE: 'c:¥at-gpib¥qbasic¥qbdecl.bas'

COMMON SHARED DEV%, GPIB0%, PPG%, ED%

DECLARE SUB waidly (tim!)
DECLARE SUB wrtcmd2 (w$)
DECLARE FUNCTION gpinit% ()
DECLARE FUNCTION readcmd2$ ()

CLS
IF gpinit% <> 0 THEN            ' Initialize GPIB environment
'
    CALL wrtcmd2("PTS 3")
    DO
        DO
            PRINT "** MP1762/MP1764A INPUT SIGNAL SAMPLE PROGRAM ** "
            PRINT
            PRINT " INPUT SIGNAL * DATA THRESHOLD      = [0] "
            PRINT "              * DATA TERMINATION    = [1] "
            PRINT "              * CLOCK PHASE ADJUST  = [2] "
            PRINT "              * CLOCK TERMINATION   = [3] "
            PRINT "              * CLOCK POLARITY      = [4] "
            PRINT
            INPUT "Choose function [0 to 4]:"; sel%

            IF sel% < 0 OR sel% > 4 THEN
                CLS
                PRINT "Wrong chosen number!!"
                PRINT "Please, enter correct number."
            END IF
        LOOP UNTIL sel% >= 0 AND sel% <= 4
        '
        SELECT CASE sel%
        CASE 0
            PRINT "Please, type number for the DATA THRESHOLD."
            INPUT "Possible data range is -3.000 to +1.875V, STEP 0.001V."; dth$
            CALL wrtcmd2("DTH " + dth$)
            '
        CASE 1
            INPUT "Choose DATA TERMINATION.[GND:0, -2V:1] "; dtm$
            CALL wrtcmd2("DTM " + dtm$)
            '
        CASE 2
            PRINT "Please, type number for the CLOCK PHASE ADJUST."
            INPUT "Possible data range is -500 to 500ps, STEP by 1ps"; cpa$
            '
            DO
                CALL wrtcmd2("DLY?")
                RD$ = readcmd2$
                IF MID$(RD$, 1, 5) = "DLY 0" THEN
                    EXIT DO
                ELSE
                    ' Wait status read timing delay for clock phase adjust
                    CALL waidly(1)
                END IF
            LOOP
            '
            WRT$ = "CPA " + cpa$: CALL wrtcmd2(WRT$)
            '
        CASE 3
            INPUT "Choose CLOCK TERMINATION.[GND:0, -2V:1] "; ctm$
            CALL wrtcmd2("CTM " + ctm$)
            '
        CASE 4
            INPUT "Choose CLOCK POLALITY.[CLK:0, NCLK:1] "; cpl$
            CALL wrtcmd2("CPL " + cpl$)
```

```
               '
          END SELECT
          '
          INPUT "Do you set are another data? [Yes:0, No:1]:"; loop$
     LOOP UNTIL loop$ = "1"
END IF
'
STOP
```

(2) Auto threshold search (auto search) setting

This program executes the auto search operation.





Note) If a system error occurs, the error contents are displayed and the program stops.

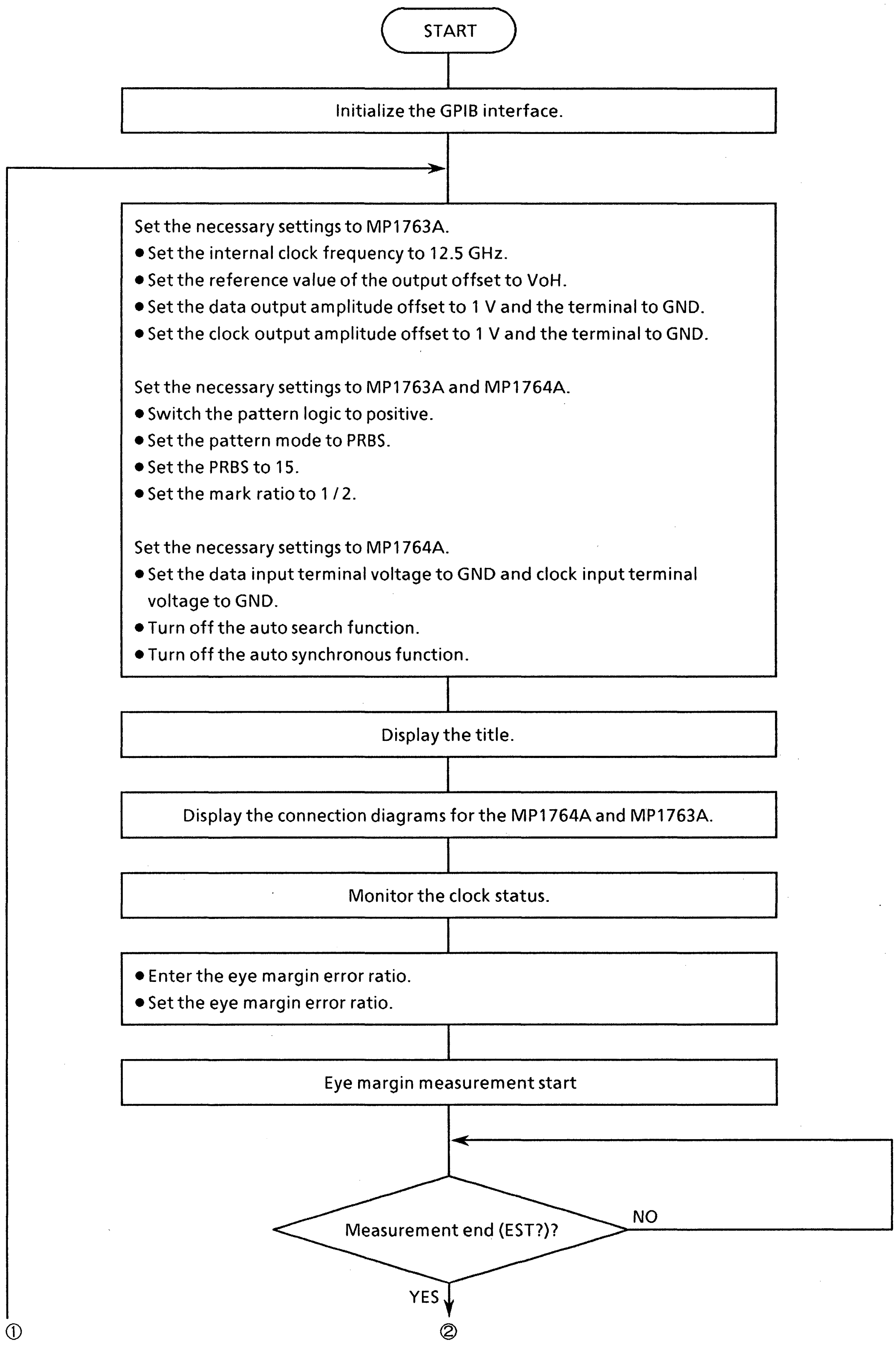
- Program list

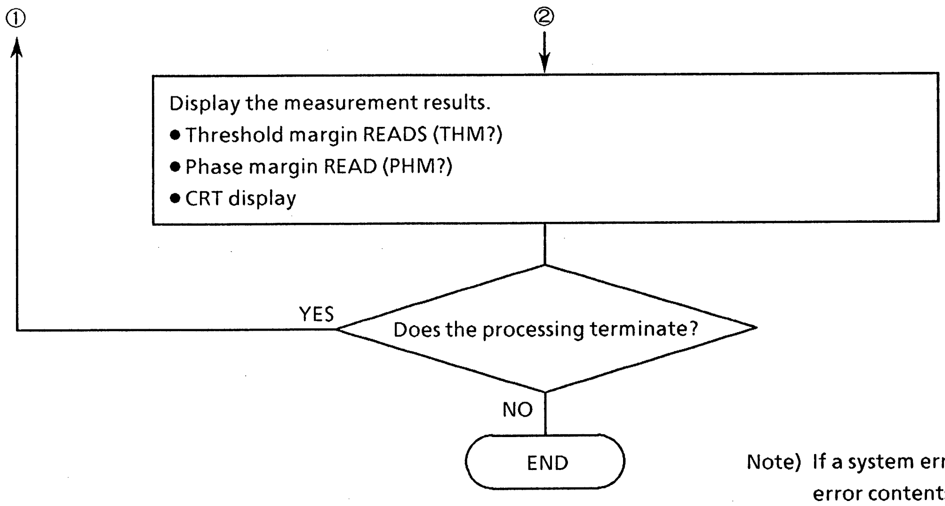
```
DECLARE SUB wrtcmd1 (WRT$)
REM $INCLUDE: 'c:¥at-gpib¥qbasic¥qbdecl.bas'

COMMON SHARED DEV%, GPIB0%, PPG%, ED%
'
DECLARE SUB ChecClk ()
DECLARE SUB Connect (ttl$)
DECLARE SUB wrtcmd2 (w$)
DECLARE FUNCTION gpinit% ()
DECLARE FUNCTION AutoSrc% ()
DECLARE FUNCTION readcmd2$ ()
'
CLS
IF gpinit% <> 0 THEN                        'Setup interface
    DO
        ' Setup to PPG
        CALL wrtcmd1("CLK 1;RES 1;FRQ 12500")  'FREQUENCY
        CALL wrtcmd1("OFS 0")                  'Offset
        CALL wrtcmd1("DAP 1;DOS 1;DTM 0")      'Data
        CALL wrtcmd1("CDL 100;CAP 1;COS 1")    'Clock
        ' Setup to ED
        CALL wrtcmd2("DTM 0;CTM 0;CPL 0")      'Input
        CALL wrtcmd2("SRH 0;SYN 1")
        ' Setup to PPG/ED
        CALL wrtcmd1("LGC 0")                  'Pattern Logic      :POSITIVE
        CALL wrtcmd2("LGC 0")
        CALL wrtcmd1("PTS 3")                  'Pattern            :PRBS
        CALL wrtcmd2("PTS 3")
        CALL wrtcmd1("PTN 6")                  'PRBS               :PN15
        CALL wrtcmd2("PTN 6")
        CALL wrtcmd1("MRK 3")                  'Mark ratio         :1/2
        CALL wrtcmd2("MRK 3")
        '

        CALL Connect("*** AUTO SEARCH SAMPLE PROGRAM ***")

        CALL ChecClk                        'Check Clock loss

        IF AutoSrc% <> 0 THEN
            GOSUB RESULT                    'Display Result
        END IF
        '
        PRINT
        INPUT "Try again [Yes/No]"; loop$
    LOOP UNTIL loop$ = "n" OR loop$ = "N"
END IF
STOP

        '
RESULT: '-----  Display Result  -----
        '
    CALL wrtcmd2("DTH?")
    RD$ = LEFT$(readcmd2$, IBCNT% - 1)
    PRINT "DATA THRESHOLD       = " + MID$(RD$, 5, 6) + " V"

    CALL wrtcmd2("DTM?")
    RD$ = LEFT$(readcmd2$, IBCNT% - 1)
    IF MID$(RD$, 1, 5) = "DTM 0" THEN
        RD$ = "   GND"
    ELSE
        RD$ = "   -2V"
    END IF
    PRINT "DATA TERMINATION     = " + RD$
```

```
CALL wrtcmd2("CPA?")
RD$ = LEFT$(readcmd2$, IBCNT% - 1)
PRINT "CLOCK PHASE ADJUST   = " + MID$(RD$, 5, 4) + " ps"

CALL wrtcmd2("CTM?")
RD$ = LEFT$(readcmd2$, IBCNT% - 1)
IF MID$(RD$, 1, 5) = "CTM 0" THEN
    RD$ = "   GND"
ELSE
    RD$ = "   -2V"
END IF
PRINT "CLOCK TERMINATION    = " + RD$

CALL wrtcmd2("CPL?")
RD$ = LEFT$(readcmd2$, IBCNT% - 1)
IF MID$(RD$, 1, 5) = "CPL 0" THEN
    RD$ = "   CLK"
ELSE
    RD$ = " N CLK"
END IF
PRINT "CLOCK POLARITY       = " + RD$

RETURN
'
```

(3) Eye margin measurement

This program measures the eye margins.





Note) If a system error occurs, the error contents are displayed and the program stops.

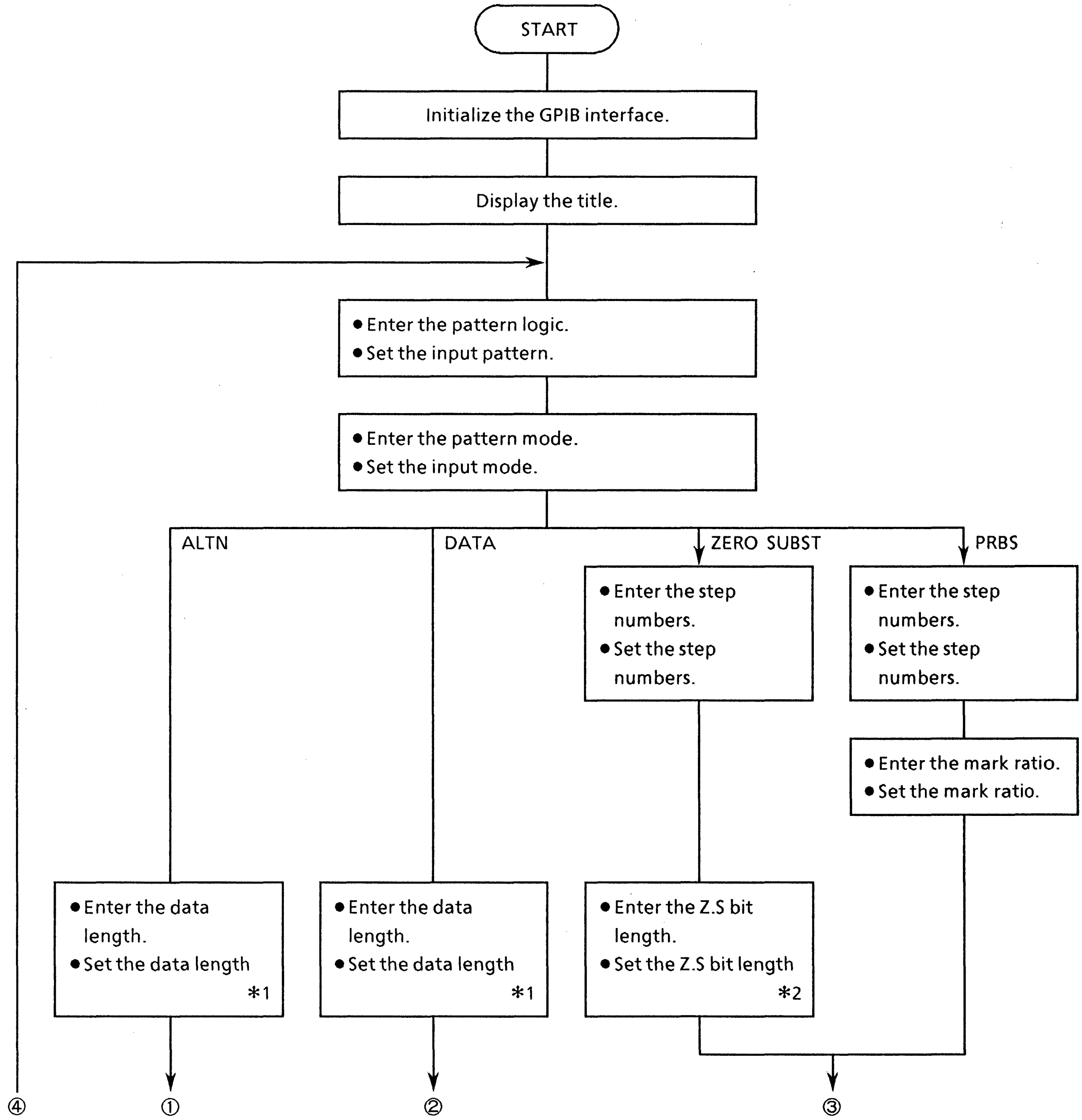
- Program list

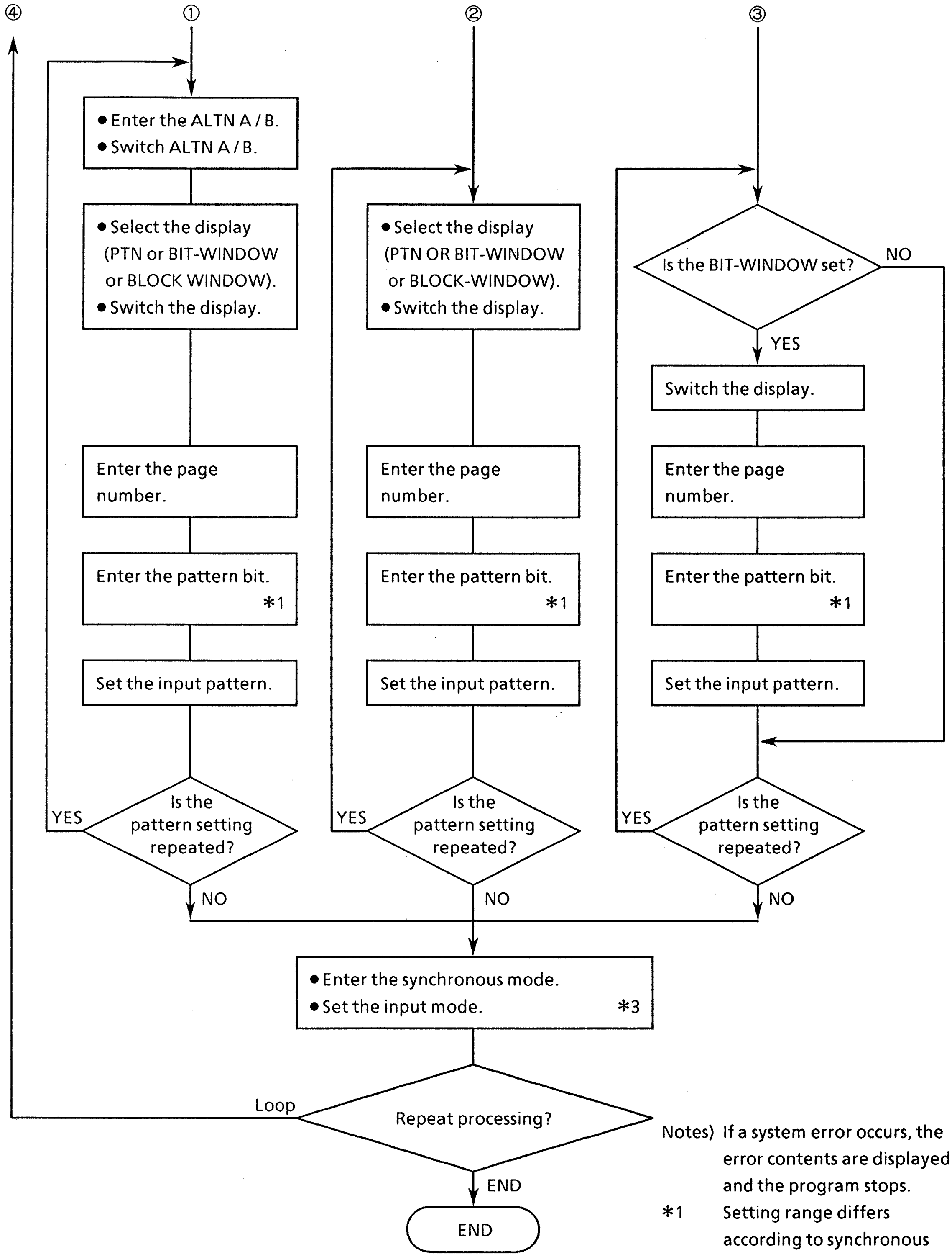
```
DECLARE SUB wrtcmd1 (WRT$)
REM $INCLUDE: 'c:¥at-gpib¥qbasic¥qbdecl.bas'

COMMON SHARED DEV%, GPIB0%, PPG%, ED%

DECLARE SUB ChecClk ()
DECLARE SUB ClearDisp (p%, l%)
DECLARE SUB waidly (tim!)
DECLARE SUB Connect (ttl$)
DECLARE SUB wrtcmd2 (w$)
DECLARE FUNCTION gpinit% ()
DECLARE FUNCTION readcmd2$ ()
'
IF gpinit% <> 0 THEN                    'Setup interface
    DO
        '==============  Eye margin start  ===============
        ' Setup to PPG
        CALL wrtcmd1("CLK 1;RES 1;FRQ 12500")  'FREQUENCY
        CALL wrtcmd1("OFS 0")                  'Offset
        CALL wrtcmd1("DAP 1;DOS 1;DTM 0")      'Data
        CALL wrtcmd1("CDL 100;CAP 1;COS 1")    'Clock
        ' Setup to ED
        CALL wrtcmd2("DTM 0;CTM 0;CPL 0")      'Input
        CALL wrtcmd2("SRH 0;SYN 1")
        ' Setup to PPG/ED
        CALL wrtcmd1("LGC 0")                  'Pattern Logic      :POSITIVE
        CALL wrtcmd2("LGC 0")
        CALL wrtcmd1("PTS 3")                  'Pattern            :PRBS
        CALL wrtcmd2("PTS 3")
        CALL wrtcmd1("PTN 6")                  'PRBS               :PN15
        CALL wrtcmd2("PTN 6")
        CALL wrtcmd1("MRK 3")                  'Mark ratio         :1/2
        CALL wrtcmd2("MRK 3")
        CALL wrtcmd1("SFT 0")                  'AND bit shift      :1bit
        CALL wrtcmd2("SFT 0")
        '

        CALL Connect("*** EYE MARGIN SAMPLE PROGRAM ***")

        DO                              ' ====  Set error ratio   ====
            LOCATE 20, 1
            PRINT "INPUT ERROR RATIO [ 1.0E-2:0 , 1.0E-3:1 , 1.0E-4:2 , 1.0E-5:3
 ]"
            PRINT "                  [ 1.0E-6:4 , 1.0E-7:5 , 1.0E-8:6 , 1.0E-9:7
 ]"
            PRINT
            INPUT "Choose error ratio:"; ratio%
            IF ratio% < 0 OR ratio% > 7 THEN
                LOCATE 18, 1
                PRINT "Wrong chosen number!! Please, enter a correct number."
                LOCATE 23, 1
                PRINT "                                    "
            END IF
        LOOP UNTIL ratio% >= 0 AND ratio% <= 7
        CALL wrtcmd2("EME 1;EYT " + STR$(ratio%))

        CALL ChecClk                    ' ====  test clock loss  ====

        CALL ClearDisp(18, 6)

        CALL wrtcmd2("EST 1")           '====  Eye margin start ====
        LOCATE 4, 1
        PRINT "*** Eye Margin Start ***    "
```

```
        '=========  Polling end of Measurement  ==========
        DO
            CALL wrtcmd2("EST?")
            RD$ = LEFT$(readcmd2$, IBCNT% - 1)
            IF MID$(RD$, 1, 5) = "EST 0" THEN
                LOCATE 4, 1
                PRINT "*** Eye Margin finish ***   "
                waidly (1)

                '============== Display result  =================
                CALL wrtcmd2("THM?")
                RD1$ = LEFT$(readcmd2$, IBCNT% - 1)
                CALL wrtcmd2("PHM?")
                RD2$ = LEFT$(readcmd2$, IBCNT% - 1)
                PRINT "THRESHOLD MARGIN : " + MID$(RD1$, 5, 6) + " Vp-p"
                PRINT "PHASE MARGIN     :  " + MID$(RD2$, 5, 6) + " psp-p"
                EXIT DO
            ELSE
                IF MID$(RD$, 1, 5) = "EST 1" THEN
                    waidly (1)
                ELSE
                    'if execute fail then restart measure
                    LOCATE 4, 1
                    PRINT "*** Eye Margin Execution fail ***    "
                    CALL wrtcmd2("EST 0")
                    EXIT DO
                END IF
            END IF
        LOOP

        PRINT
        INPUT "Do you wish try again [Yes/No]"; loop$
    LOOP UNTIL loop$ = "n" OR loop$ = "N"

    CALL wrtcmd2("EME 0")        ' ====  Eye margin OFF  ====
END IF
'
STOP
```

(4) Measurement pattern, BIT WINDOW, and BLOCK WINDOW setting

This program sets the measurement pattern, BIT WINDOW, and BLOCK WINDOW.





Notes) If a system error occurs, the error contents are displayed and the program stops.

*1 Setting range differs according to synchronous mode.

*2 Setting range differs according to number of steps of PRBS or ZERO SUBST.

*3 Selection item differs according to pattern mode.

```
REM $INCLUDE: 'c:¥at-gpib¥qbasic¥qbdecl.bas'

COMMON SHARED DEV%, GPIB0%, PPG%, ED%

DECLARE SUB wrtcmd2 (w$)
DECLARE FUNCTION gpinit% ()
'
CLS
IF gpinit% <> 0 THEN                                    'Setup GPIB interface
    DO
        CLS

        PRINT "** MP1762A/MP1764A PATTERN SAMPLE PROGRAM **"
        PRINT

        '======== Set Pattern Logic =========
        DO
            INPUT "Choose LOGIC polarity [ Positive:0 , Negative:1 ] "; lg%
            IF lg% <> 0 AND lg% <> 1 THEN
                CLS
                PRINT "Wrong Chosen number!! Please select a correct LOGIC polar
ity."
            END IF
        LOOP UNTIL lg% = 0 OR lg% = 1
        CALL wrtcmd2("LGC " + STR$(lg%))

        '======== Set Pattern mode ==========
        DO
            INPUT "Choose Measure PATTERN [ ALTERNATE:0 , DATA:1 , ZERO Subst.:2
 , PRBS:3 ] "; ptn%
            IF ptn% < 0 OR ptn% > 3 THEN
                CLS
                PRINT "Wrong Chosen number!! Please select a correct new PATTERN
."
            END IF
        LOOP UNTIL ptn% >= 0 AND ptn% <= 3
        CALL wrtcmd2("PTS " + STR$(ptn%))

        SELECT CASE ptn%
        CASE 0
            '======== Set Altn pattern ========
            GOSUB SetAltn

            '========  Set Sync mode   ========
            DO
                INPUT "Choose SYNC MODE [ NOMAL:0 , FRAME:1 ] "; sync%
                IF sync% <> 0 AND sync% <> 1 THEN
                    CLS
                    PRINT "Wrong Chosen number!! Please select a correct pattern
 sync."
                END IF
            LOOP UNTIL sync% = 0 OR sync% = 1
            wrtcmd2 ("SYM " + STR$(sync%))

        CASE 1
            '======== Set data pattern ========
            GOSUB SetData

            '========  Set Sync mode   ========
            DO
                INPUT "Choose SYNC MODE [ NOMAL:0 , FRAME:1 , QUICK:2 ] "; sync%
                IF sync% < 0 OR sync% > 2 THEN
                    CLS
```

```
                    PRINT "Wrong Chosen number!! Please select a correct pattern
sync."
                END IF

            LOOP UNTIL sync% >= 0 AND sync% <= 2
            wrtcmd2 ("SYM " + STR$(sync%))

        CASE 2
            '===== Set zero subst pattern =====
            GOSUB SetZero

            '========  Set Sync mode   ========
            DO
                INPUT "Choose SYNC MODE [ NOMAL:0 , FRAME:1 , QUICK:2 ] "; sync%
                IF sync% < 0 OR sync% > 2 THEN
                    CLS
                    PRINT "Wrong Chosen number!! Please select a correct pattern
sync."
                END IF

            LOOP UNTIL sync% >= 0 AND sync% <= 2
            wrtcmd2 ("SYM " + STR$(sync%))

        CASE 3
            '======== Set prbs pattern ========
            GOSUB SetPrbs
            '
        END SELECT

        '============ continue ? ===============
        INPUT "Do you wish to set other pattern set? [Yes/No]"; loop$
    LOOP UNTIL loop$ = "n" OR loop$ = "N"
END IF
'
STOP
'
        '
SetAltn: '------------  Set Altn Pattern -------------
        '
    '============  Data length  ===============
    DO
        INPUT "Set DATA LENGTH [128 to 4194304] "; length
        IF length < 128 OR length > 4194304 THEN
            CLS
            PRINT "Wrong input data length!! Please to set a correct number."
        END IF
    LOOP UNTIL length >= 128 AND length <= 4194304
    wrtcmd2 ("DLN " + STR$(length))
    '

    '============  Set pattern  ===============
    DO
        INPUT "Choose ALTERNATE A or B:"; alt$
        IF alt$ = "a" OR alt$ = "A" THEN
            wrtcmd2 ("ALT 0")
        ELSE
            IF alt$ = "b" OR alt$ = "B" THEN
                wrtcmd2 ("ALT 1")
            END IF
        END IF

        DO
            INPUT "Choose DISPLAY type[ PATTERN:0 , BIT-WINDOW:1 , BLOCK-WINDOW:
```

```
2 ] "; disp%
            IF disp% < 0 OR disp% > 2 THEN
                CLS
                PRINT "Wrong Chosen number!! Please select a correct correct DIS
PLAY type."
            END IF

        LOOP UNTIL disp% >= 0 AND disp% <= 2
        wrtcmd2 ("DSP " + STR$(disp%))

        GOSUB SetBit

        INPUT "Do you wish to continue to set another pattern? [Yes/No] "; loop$
    LOOP UNTIL loop$ = "n" OR loop$ = "N"
RETURN
'
     '
SetData: '------------- Set DATA Pattern ---------------
     '
    '========== Set data length ============
    DO
        INPUT "Set DATA LENGTH [2 to 8388608] "; length
        IF length < 2 OR length > 8388608 THEN
            CLS
            PRINT "Wrong length number!! Please set correct number."
        END IF
    LOOP UNTIL length >= 2 AND length <= 8388608
    wrtcmd2 ("DLN " + STR$(length))

    DO
        '============ Set display =============
        DO
            INPUT "Choose DISPLAY type[ PATTERN:0 , BIT-WINDOW:1 , BLOCK-WINDOW:
2 ] "; disp%
            IF disp% < 0 OR disp% > 2 THEN
                CLS
                PRINT "Wrong Chosen number!! Please select a correct correct DIS
PLAY type."
            END IF
        LOOP UNTIL disp% >= 0 AND disp% <= 2
        wrtcmd2 ("DSP " + STR$(disp%))

        '============= Set pattern ==============
        GOSUB SetBit

        INPUT "Do you wish continue another pattern set? [Yes/No] "; loop$
    LOOP UNTIL loop$ = "n" OR loop$ = "N"
RETURN
'
     '
SetZero: '--------- Set ZERO SUBST PATTERN ------------
     '
    '========== Set zero subst =============
    DO
        PRINT "ZERO SUBSTITUTION [2^7-1:0 , 2^9-1:1 , 2^11-1:2 , 2^15-1:3 ]"
        PRINT
        INPUT "Choose ZERO Substitution lenght:"; dan%
        IF dan% < 0 OR dan% > 3 THEN
            CLS
            PRINT "Wrong Chosen number!! Please put things right length."
        END IF
    LOOP UNTIL dan% >= 0 AND dan% <= 3
```

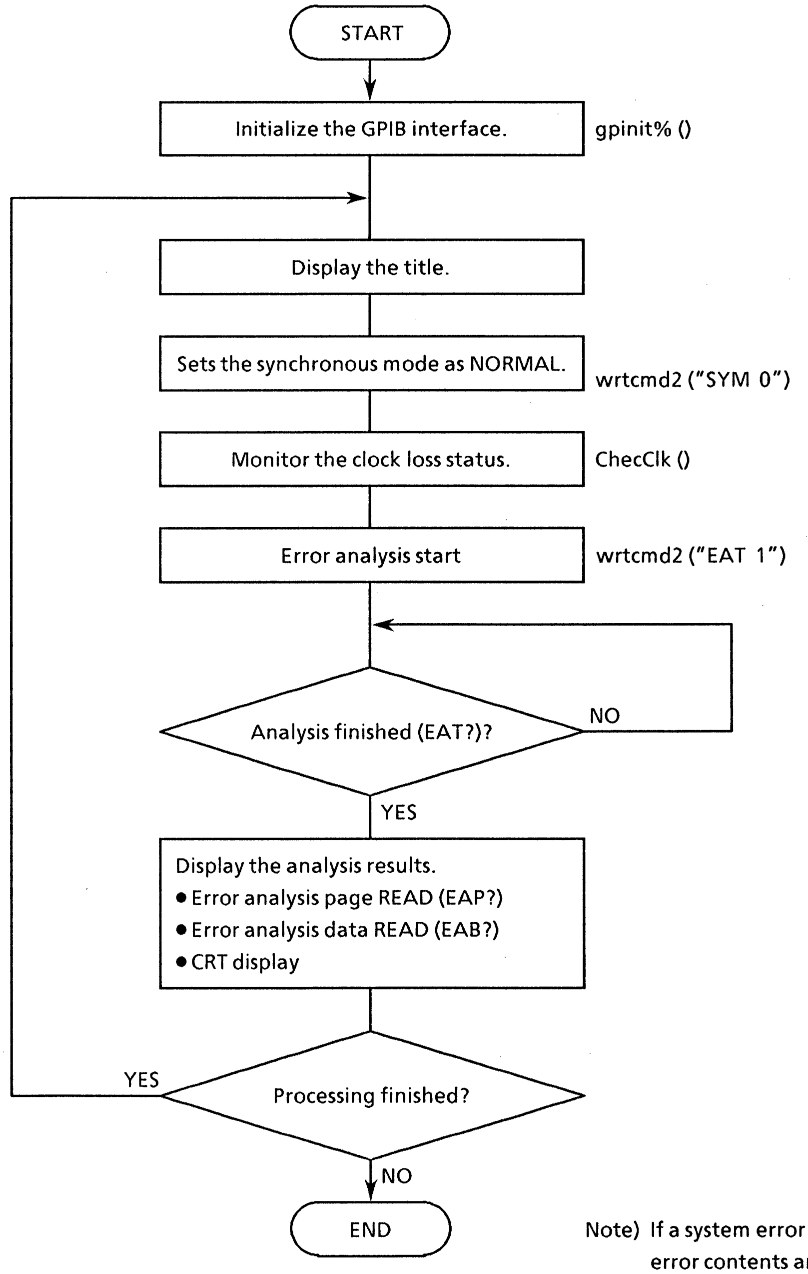
```
    DO
        '========= Set zero-sub length ===========
        SELECT CASE dan%
        CASE 0
            wrtcmd2 ("PTN 2")
            DO
                INPUT "Set ZERO Substitution BIT LENGTH [1 to 127]"; length%
                IF length% < 1 OR length > 127 THEN
                    CLS
                    PRINT "Wrong input for out of range limit!! Please enter a c
orrect bit length."
                END IF
            LOOP UNTIL length% >= 1 AND length% <= 127

        CASE 1
            wrtcmd2 ("PTN 3")
            DO
                INPUT "Set ZERO Substitution BIT LENGTH [1 to 511]"; length%
                IF length% < 1 OR length > 511 THEN
                    CLS
                    PRINT "Wrong input for out of range limit!! Please enter a c
orrect bit length."
                END IF
            LOOP UNTIL length% >= 1 AND length% <= 511

        CASE 2
            wrtcmd2 ("PTN 5")
            DO
                INPUT "Set ZERO Substitution BIT LENGTH [1 to 2047]"; length%
                IF length% < 1 OR length > 2047 THEN
                    CLS
                    PRINT "Wrong input for out of range limit!! Please enter a c
orrect bit length."
                END IF
            LOOP UNTIL length% >= 1 AND length% <= 2047

        CASE 3
            wrtcmd2 ("PTN 6")
            DO
                INPUT "Set ZERO Substitution BIT LENGTH [1 to 32767]"; length%
                IF length% < 1 OR length > 32767 THEN
                    CLS
                    PRINT "Wrong input for out of range limit!! Please enter a c
orrect bit length."
                END IF
            LOOP UNTIL length% >= 1 AND length% <= 32767

        END SELECT
        wrtcmd2 ("ZLN " + STR$(length%))

        '===========  Chanel mask  =============
        INPUT "Do you set Mask Chanel ? [Yes/No] "; ans$
        IF ans$ = "y" OR ans$ = "Y" THEN
            wrtcmd2 ("DSP 1")
            disp% = 1
            GOSUB SetBit
        ELSE
            wrtcmd2 ("DSP 0")
            disp% = 0
        END IF

        INPUT "Do you wish continue another pattern set? [Yes/No] "; loop$
    LOOP UNTIL loop$ = "n" OR loop$ = "N"
```

```
RETURN
'
         '
SetPrbs: '------------  Set PRBS PATTERN  --------------
         '
    '============   Set prbs   ===============
    DO
        PRINT
        PRINT "SELECT PRBS [ 2^7 :0 , 2^9 :1 , 2^11:2 , 2^15:3 ]"
        PRINT "             [ 2^20:4 , 2^23:5 , 2^31:6         ]"
        PRINT
        INPUT "Choose PRBS lenght:"; dan%
        IF dan% < 0 OR dan% > 6 THEN
            CLS
            PRINT "Wrong Chosen number!! Please put things right length."
        END IF
    LOOP UNTIL dan% >= 0 AND dan% <= 6

    SELECT CASE dan%
    CASE 0
        wrtcmd2 ("PTN 2")              ' PRBS 2^7
    CASE 1
        wrtcmd2 ("PTN 3")              ' PRBS 2^9
    CASE 2
        wrtcmd2 ("PTN 5")              ' PRBS 2^11
    CASE 3
        wrtcmd2 ("PTN 6")              ' PRBS 2^15
    CASE 4
        wrtcmd2 ("PTN 7")              ' PRBS 2^20
    CASE 5
        wrtcmd2 ("PTN 8")              ' PRBS 2^23
    CASE 6
        wrtcmd2 ("PTN 9")              ' PRBS 2^31
    END SELECT

    DO
        '============ Set mark ratio ============
        PRINT
        PRINT "MARK RATIO (Positive)[ 0/8:0 , 1/8:1 , 1/4:2 , 1/2:3 ]"
        PRINT "           (Negative)[ 8/8:0 , 7/8:1 , 3/4:2 , 1/2:3 ]"
        PRINT
        INPUT "Choose MARK RATIO:"; m$
        wrtcmd2 ("MRK " + m$)

        '============  Chanel mask  ============
        INPUT "Do you set Mask Chanel? [Yes/No] "; ans$
        IF ans$ = "y" OR ans$ = "Y" THEN
            wrtcmd2 ("DSP 1")
            disp% = 1
            GOSUB SetBit
        ELSE
            wrtcmd2 ("DSP 0")
            disp% = 0
        END IF

        INPUT "Do you wish continue another pattern set? [Yes/No] "; loop$
    LOOP UNTIL loop$ = "n" OR loop$ = "N"
RETURN
'
'
        '
SetBit: '--------  Set Page and Bit Pattern  -----------
        '
```

```
    SELECT CASE disp%
    CASE 0
        '==========    Set page    ==========
        INPUT "to set top PAGE:"; page

        '========= Set pattern bit =========
        j = 8
        GOSUB Inbit
        PRINT "PAG " + STR$(page) + ";BIT " + BIT$
        wrtcmd2 ("PAG " + STR$(page) + ";BIT " + BIT$)

    CASE 1
        '==========    Set page    =========
        INPUT "Set BIT-WINDOW PAGE ( 1 or 2 ):"; page

        '=========   Set mask bit   =========
        j = 2
        GOSUB Inbit
        wrtcmd2 ("MSK " + STR$(page) + ";CHM " + BIT$)

    CASE 2
        '==========    Set page    =========
        INPUT "Set top PAGE:"; page

        '=========   Set mask bit   =========
        j = 8
        GOSUB Inbit
        wrtcmd2 ("PAG " + STR$(page) + ";MGB " + BIT$)
    END SELECT
RETURN

        '
Inbit:  '
        '
    PRINT "You are able to choice data format of Hexadecimal or Decimal."
    PRINT "Default data format is Hexadecimal."

    BIT$ = ""
    FOR k = 0 TO j - 1
        PRINT "< Do you set bit-pattern of " + STR$(page + k) + " PAGE ? [Yes/No
] > ";
        INPUT ; a$
        IF a$ = "n" OR a$ = "N" THEN
            EXIT FOR
        END IF
        '
        IF k <> 0 THEN BIT$ = BIT$ + ","
        '
        PRINT " "
        INPUT "Which do you choice format? [ Hex/Dec ]"; type$
        IF type$ = "d" OR type$ = "D" THEN
            PRINT "Enter " + STR$(page + k) + " PAGE pattern [0 to 65535]:";
            INPUT ; b$
        ELSE
            PRINT "Enter " + STR$(page + k) + " PAGE pattern [0 to FFFF]:";
            INPUT ; b$
            b$ = "#H" + b$
        END IF
        BIT$ = BIT$ + b$
        '
    NEXT k
    PRINT " "
    '
RETURN
```

(5) Error analysis

This program executes the error analysis function.

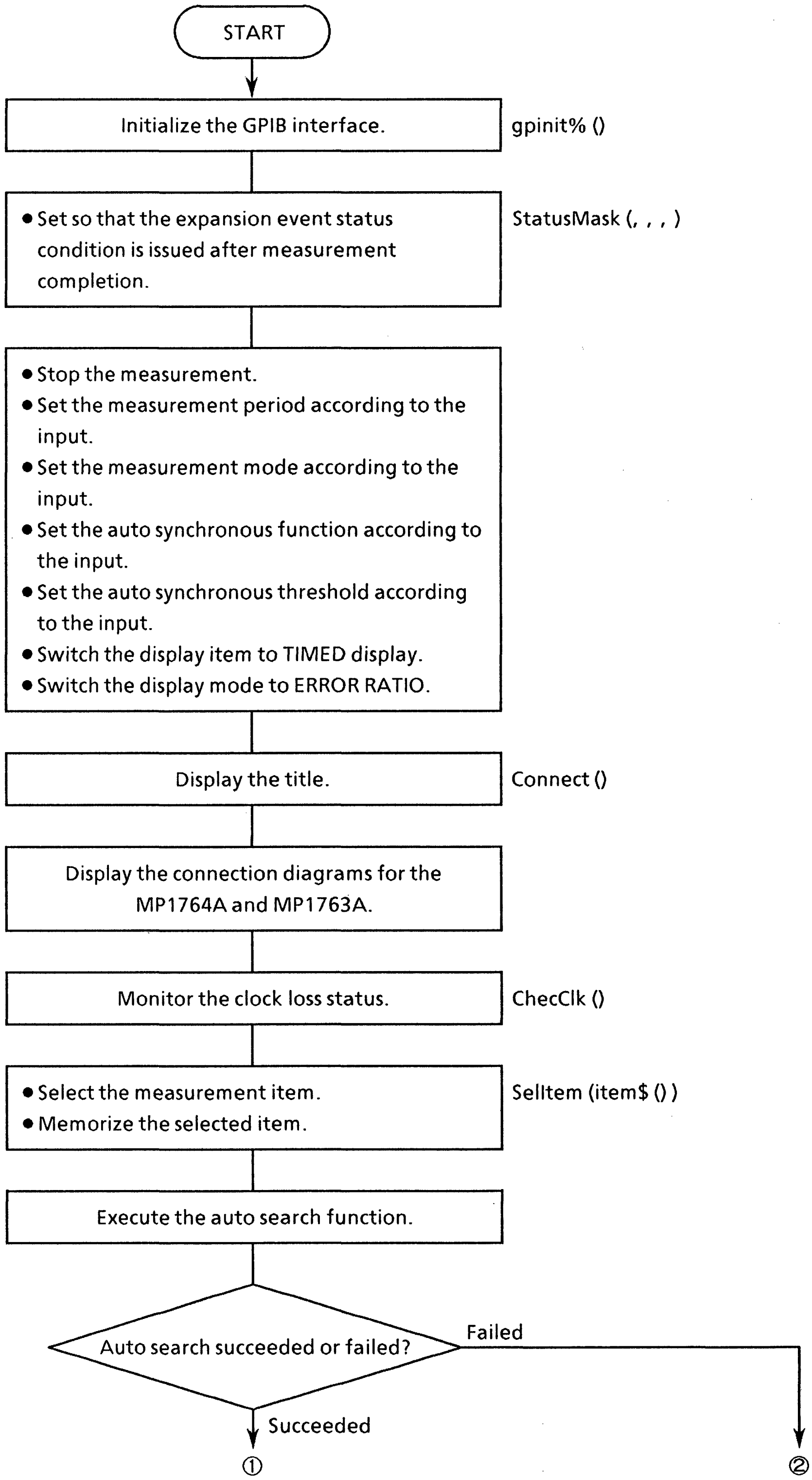


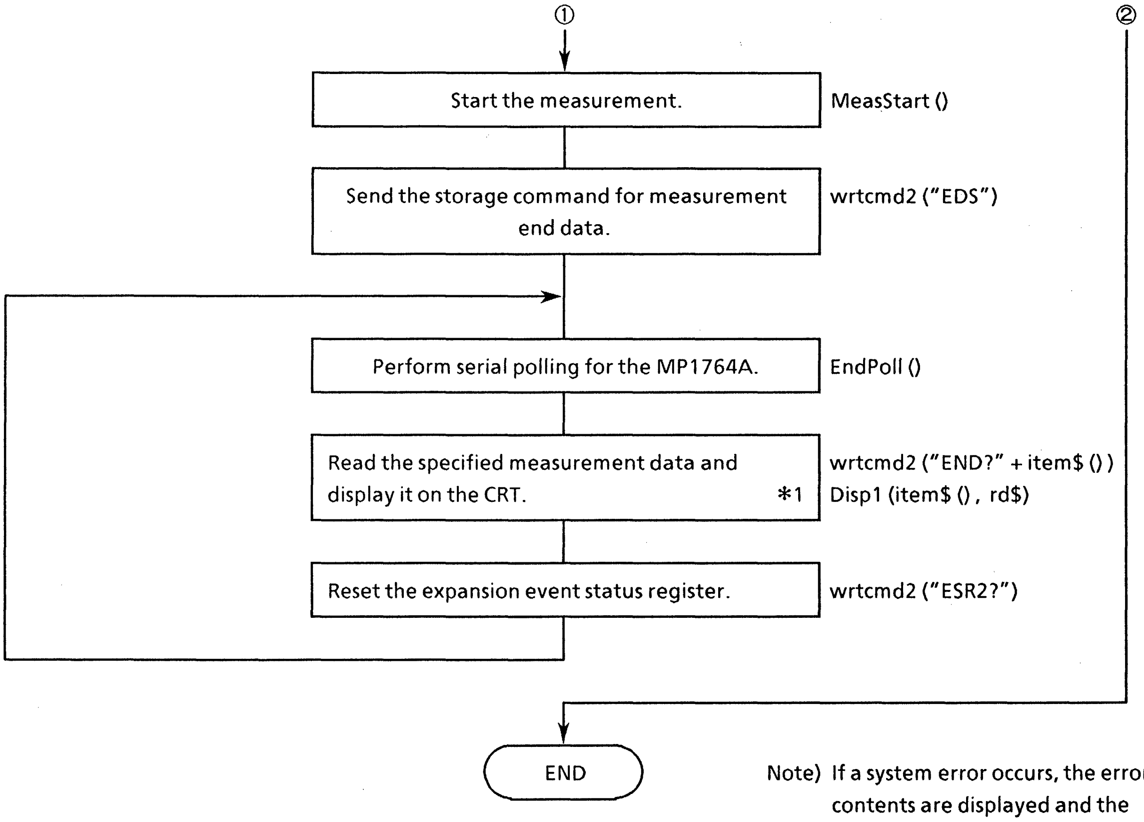
Note) If a system error occurs, the error contents are displayed and the program stops.

```
REM $INCLUDE: 'c:¥at-gpib¥qbasic¥qbdecl.bas'

COMMON SHARED DEV%, GPIB0%, PPG%, ED%

DECLARE SUB ChecClk ()
DECLARE SUB waidly (tim!)
DECLARE SUB wrtcmd2 (w$)
DECLARE FUNCTION gpinit% ()
DECLARE FUNCTION readcmd2$ ()

IF gpinit% <> 0 THEN                'Setup interface
    DO
        CLS
        PRINT "** MP1762A/MP1764A ERROR ANALYSIS SAMPLE PROGRAM ** "
        PRINT
        CALL wrtcmd2("SYM 0")        ' Sync mode set to normal

        CALL ChecClk                 ' Test Clock loss

        '============ Error analysis start ==============
        CALL wrtcmd2("EAT 1")

        '=========== Polling end of analysis ============
        DO
            CALL wrtcmd2("EAT?")
            RD$ = readcmd2$
            waidly (1)
        LOOP UNTIL MID$(RD$, 1, 5) = "EAT 2"
        CALL wrtcmd2("EAT 0")

        '============== Display result =================
        PRINT "Error Analysis data"
        FOR j = 1 TO 16
            CALL wrtcmd2("EAP " + STR$(j))
            CALL wrtcmd2("EAB?")
            RD$ = LEFT$(readcmd2$, IBCNT% - 1)

            PRINT "page:" + MID$(RD$, 5, 9) + ", Data:" + MID$(RD$, 17, 16)
        NEXT j
        PRINT

        INPUT "Do you want to try again? [Yes/No]:"; loop$
    LOOP UNTIL loop$ = "n" OR loop$ = "N"
END IF

STOP
```

(6) Measurement results display 1 (Displays the measurement results by serial polling)

This program displays the measurement results on the CRT.





Note) If a system error occurs, the error contents are displayed and the program stops.

*1 Only the data of the specified item is read.

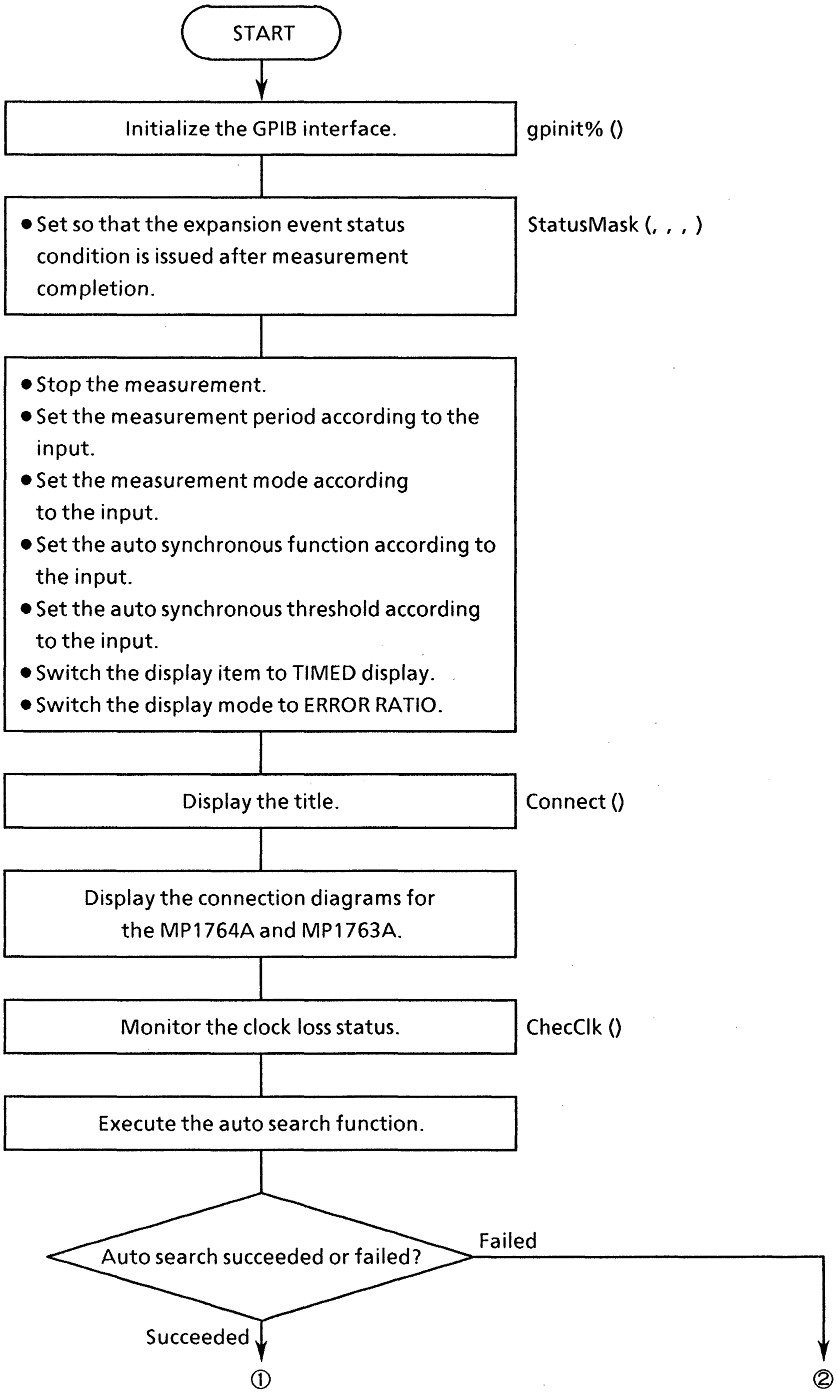
- Program list

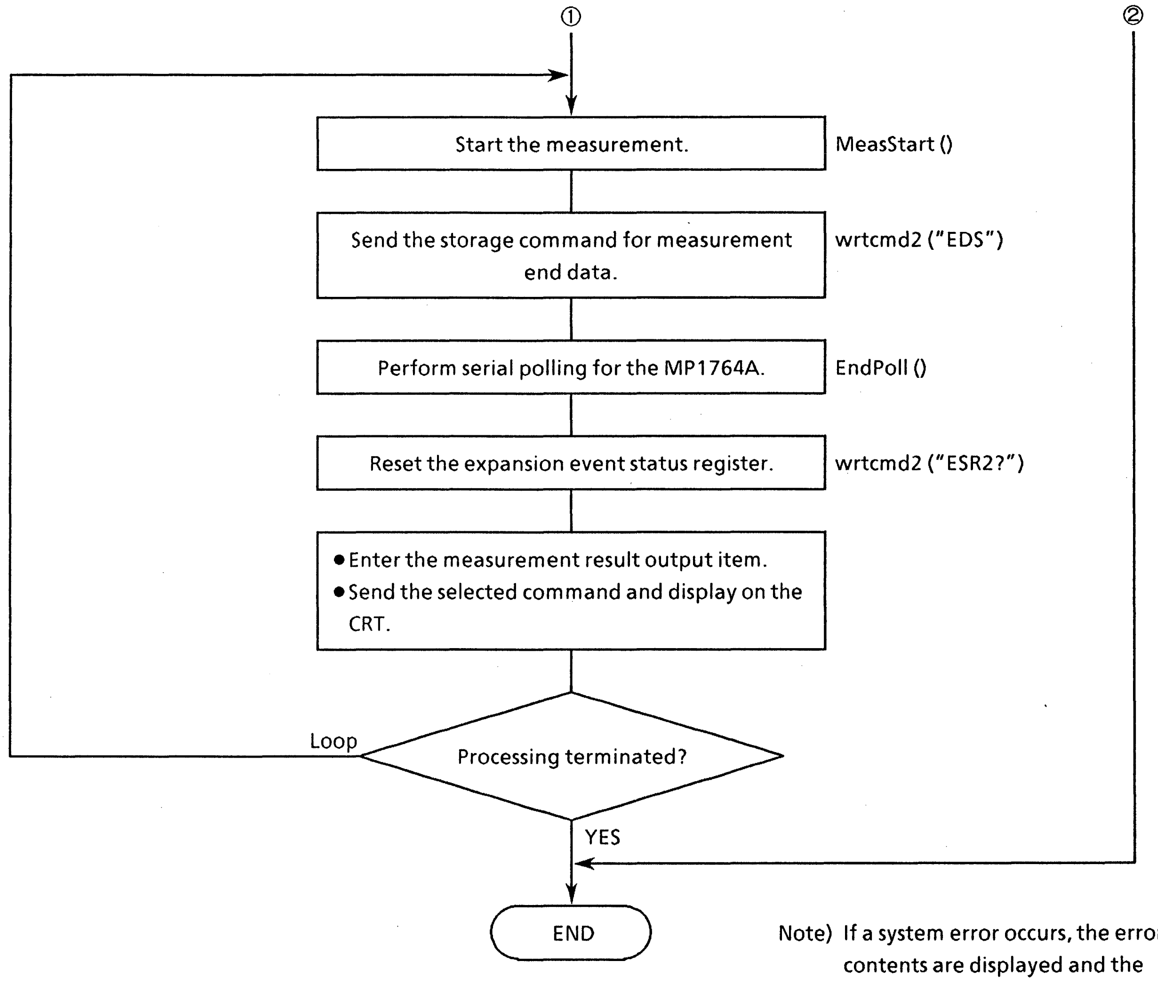
```
REM $INCLUDE: 'c:¥at-gpib¥qbasic¥qbdecl.bas'

COMMON SHARED DEV%, GPIB0%, PPG%, ED%

DECLARE SUB waidly (tim!)
DECLARE SUB ClearDisp (p%, l%)
DECLARE SUB Displ (CMD$, RD$)
DECLARE SUB ChecClk ()
DECLARE SUB MeasStart ()
DECLARE SUB MeasStop ()
DECLARE SUB EndPoll ()
DECLARE SUB SelItem (item$())
DECLARE SUB wrtcmd2 (w$)
DECLARE SUB Connect (ttl$)
DECLARE SUB StatusMask (s0%, s1%, s2%, s3%)
DECLARE FUNCTION gpinit% ()
DECLARE FUNCTION AutoSrc% ()
DECLARE FUNCTION readcmd2$ ()

DIM item$(5, 7)                              'Command string

CLS
IF gpinit% <> 0 THEN                         'Setup interface
    '======= Set event status enable register =======
    CALL StatusMask(&H0, &H0, &H1, &H0)

    '===========   Set mode   ============
    CALL MeasStop
    DO
        LOCATE 17, 1
        INPUT "MEAS.MODE? [Repeat:0, Single:1, Untimed:2]", mmode%
        IF mmode% < 0 OR mmode% > 2 THEN
            LOCATE 16, 1
            PRINT "Wrong chosen number!! Please select corrct threshold."
            CALL ClearDisp(17, 4)
        END IF
    LOOP UNTIL mmode% >= 0 AND mmode% <= 8
    CALL wrtcmd2("MOD " + STR$(mmode%))

    CALL ClearDisp(16, 2)
    LOCATE 17, 1
    IF mmode% <> 2 THEN
        INPUT "MEAS.TIME? [DAY,HOUR,MINUTE,SECOND]", prd1%, prd2%, prd3%, prd4%
    END IF
    CALL wrtcmd2("PRD " + STR$(prd1%) + "," + STR$(prd2%) + "," + STR$(prd3%) +
"," + STR$(prd4%))

    CALL ClearDisp(16, 2)
    DO
        LOCATE 17, 1
        INPUT "AUTO SYNC CONDITION? [OFF:0, ON:1]", async%
        IF async% < 0 OR async% > 1 THEN
            LOCATE 16, 1
            PRINT "Wrong chosen number!! Please select corrct threshold."
            CALL ClearDisp(17, 4)
        END IF
    LOOP UNTIL async% >= 0 AND async% <= 1
    CALL wrtcmd2("SYN " + STR$(async%))

    CALL ClearDisp(16, 2)
    DO
        LOCATE 17, 1
        PRINT "AUTO SYNC Threshold Ratio"
        PRINT "[ 1E-2:0 , 1E-3:1 , 1E-4:2 , 1E-5:3 , 1E-6:4 ] "
        PRINT "[ 1E-7:5 , 1E-8:6 ,        , INT :8          ] "
        INPUT "Choose Auto Sync Threshold:"; sye%
```

```
        IF sye% < 0 OR sye% > 8 THEN
            LOCATE 16, 1
            PRINT "Wrong chosen number!! Please select corrct threshold."
            CALL ClearDisp(17, 4)
        END IF
    LOOP UNTIL sye% >= 0 AND sye% <= 8
    CALL ClearDisp(16, 5)
    CALL wrtcmd2("SYE " + STR$(sye%))

    CALL wrtcmd2("DMS 0")          'Error Display mode      : ERROR RATIO
    CALL wrtcmd2("TIM 3")          'Real time display mode : TIMED

    '==========  Draw layout box  ========
    CALL Connect("** MEASUREMENT SAMPLE PROGRAM **")

    '==========  Check clock  ============
    CALL ChecClk

    '======== Select display item ========
    CALL SelItem(item$())                'item: command string (output)
                                         'sort: command number (output)

    '========  Auto Search ON  ===========
    IF AutoSrc% = 1 THEN

        '========  Measurement start =========
        'CLS
        SCREEN 0
        CALL MeasStart
        CALL EndPoll                                    'Reset SRQ
        LOCATE 24, 48
        PRINT "** Report Measure execution. **"
        LOCATE 24, 48
        PRINT "** Push ESC key then stop.    **"

        '========== Display result ===========
        DO
            CALL wrtcmd2("MSR?")
            RD$ = LEFT$(readcmd2$, IBCNT% - 1)
            IF RD$ = "MSR 0" THEN
                CALL wrtcmd2("STA")
            END IF

            CALL wrtcmd2("EDS")                         'Data store
            CALL EndPoll                                'Polling END bit

            LOCATE 1, 1
            FOR i = 0 TO 4
                FOR j = 0 TO 6
                    IF item$(i, j) <> "" THEN
                        CALL wrtcmd2("END? " + item$(i, j))  'Write command
                        RD$ = LEFT$(readcmd2$, IBCNT% - 1)
                        IF RD$ <> "ERR" THEN
                            CALL Displ(item$(i, j), RD$)        'Display data
                        END IF
                    END IF
                NEXT j
            NEXT i
            CALL wrtcmd2("ESR2?")
            RD$ = readcmd2$                                  'Reset ESR1
        LOOP UNTIL INKEY$ = CHR$(&H1B)                      'ESC(&H1B)
    END IF
END IF
STOP
```

(7) Measurement results display 2 (Displays the measurement results using request command)

This program displays the measurement results on the CRT.





Note) If a system error occurs, the error contents are displayed and the program stops.

*1 Only the data of the specified item is read.

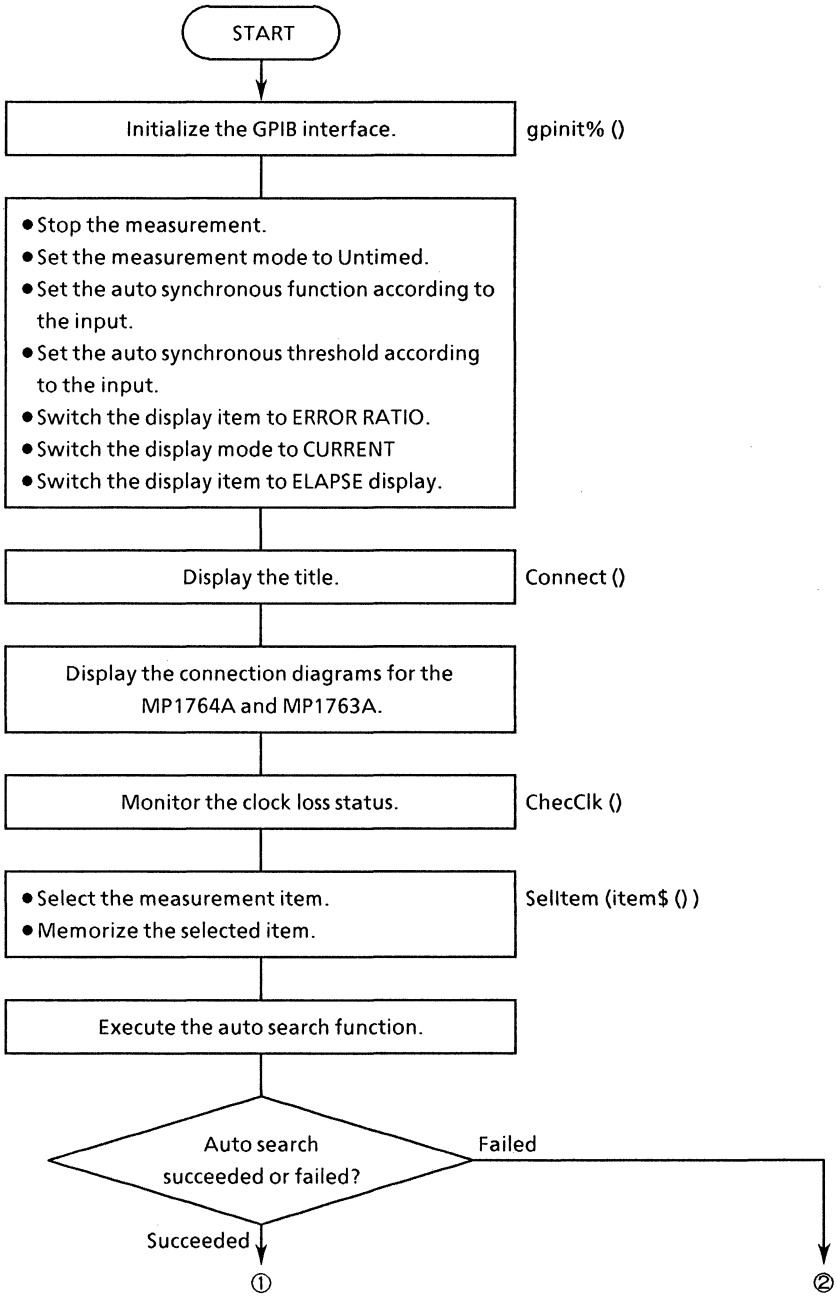
```
REM $INCLUDE: 'c:¥at-gpib¥qbasic¥qbdecl.bas'

COMMON SHARED DEV%, GPIB0%, PPG%, ED%

DECLARE SUB StatusMask (s0%, s1%, s2%, s3%)
DECLARE SUB MeasStop ()
DECLARE SUB wrtcmd2 (w$)
DECLARE SUB EndPoll ()
DECLARE SUB ClearDisp (p%, l%)
DECLARE SUB Connect (ttl$)
DECLARE SUB ChecClk ()
DECLARE SUB MeasStart ()
DECLARE SUB waidly (tim!)
DECLARE FUNCTION gpinit% ()
DECLARE FUNCTION AutoSrc% ()
DECLARE FUNCTION readcmd2$ ()

CLS
IF gpinit% <> 0 THEN                'Setup interface
    '======= Set event status enable register =======
    CALL StatusMask(&H0, &H0, &H1, &H0)

    '===========   Set mode  ============
    CALL MeasStop
    DO
        LOCATE 17, 1
        INPUT "MEAS.MODE? [Repeat:0, Single:1, Untime:2]", mmode%
        IF mmode% < 0 OR mmode% > 2 THEN
            LOCATE 16, 1
            PRINT "Wrong chosen number!! Please select corrct MEAS.MODE."
            CALL ClearDisp(17, 4)
        END IF
    LOOP UNTIL mmode >= 0 AND mmode <= 2
    CALL ClearDisp(17, 4)
    CALL wrtcmd2("MOD " + STR$(mmode%))

    IF mmode% <> 2 THEN
        LOCATE 17, 1
        INPUT "MEAS.TIME?[DAY,HOUR,MIN,SEC]", prd1%, prd2%, prd3%, prd4%
    END IF
    CALL ClearDisp(17, 4)
    CALL wrtcmd2("PRD " + STR$(prd1%) + "," + STR$(prd2%) + "," + STR$(prd3%) +
"," + STR$(prd4%))

    DO
        LOCATE 17, 1
        INPUT "AUTO SYNC CONDITION? [OFF:0, ON:1]", async%
        IF async% < 0 OR async% > 1 THEN
            LOCATE 16, 1
            PRINT "Wrong chosen number!! Please select corrct AUTO SYNC CONDITIO
N"
            CALL ClearDisp(17, 4)
        END IF
    LOOP UNTIL async% >= 0 AND async% <= 1
    CALL ClearDisp(17, 4)
    CALL wrtcmd2("SYN " + STR$(async%))

    DO
        LOCATE 17, 1
        PRINT "AUTO SYNC Threshold Ratio"
        PRINT "[ 1E-2:0 , 1E-3:1 , 1E-4:2 , 1E-5:3 , 1E-6:4 ] "
        PRINT "[ 1E-7:5 , 1E-8:6 , 1E-9:7 , INT :8          ] "
        INPUT "Choose Auto Sync Threshold:"; sye%
        IF sye% < 0 OR sye% > 8 THEN
            LOCATE 16, 1
            PRINT "Wrong chosen number!! Please select corrct threshold "
```

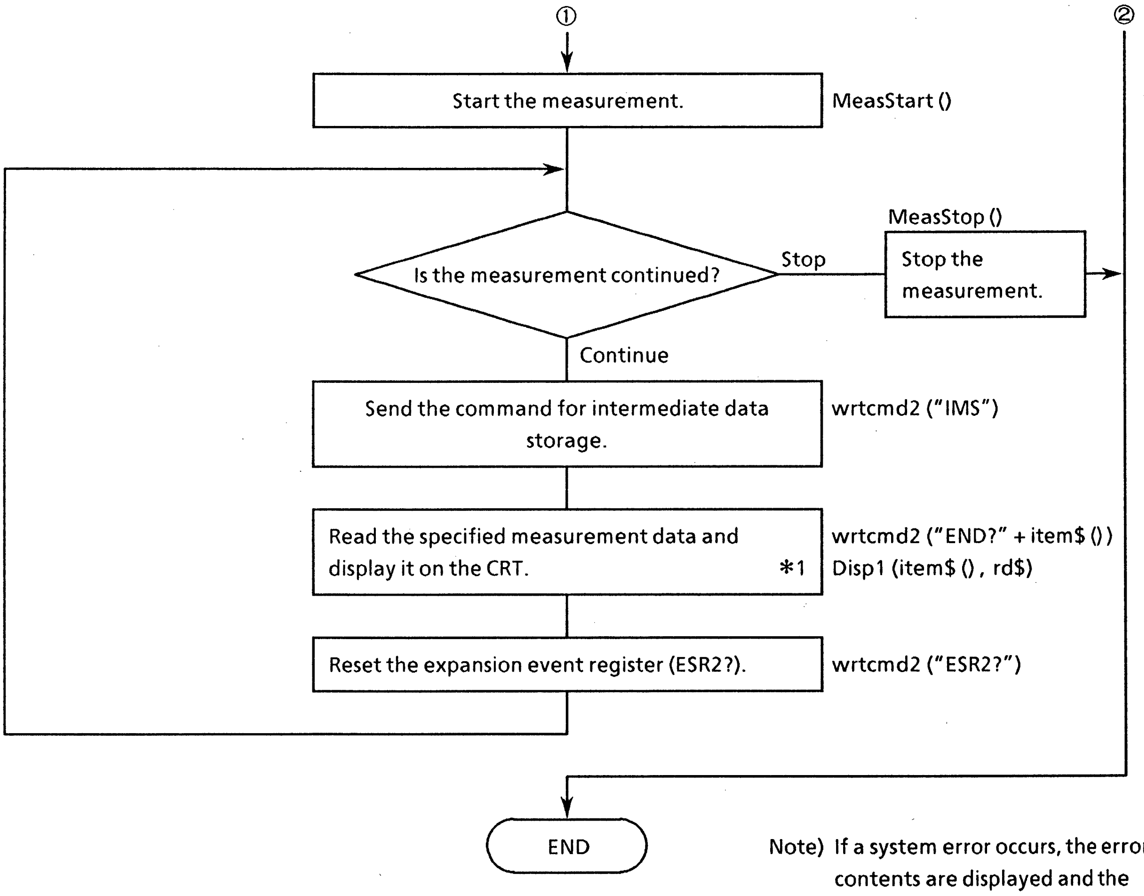
```
        CALL ClearDisp(17, 4)
    END IF
LOOP UNTIL sye% >= 0 AND sye% <= 8
CALL ClearDisp(16, 5)
CALL wrtcmd2("SYE " + STR$(sye%))

CALL wrtcmd2("DMS 0")        'Error Display mode       : ERROR RATIO
CALL wrtcmd2("TIM 3")        'Real time display mode : TIMED
CALL wrtcmd2("CUR 1")        'Current data             : ON

'==========  Draw layout box  ========
CALL Connect("** MEASUREMENT SAMPLE PROGRAM **")

'==========  Check clock  ============
CALL ChecClk

IF AutoSrc% = 1 THEN    '
    SCREEN 0
    DO
        '========  Measurement start =========
        CALL MeasStart

        '========= Polling END bit ===========
            CALL EndPoll

        '=========   Reset ESR2   ============
        CALL wrtcmd2("ESR2?")
        RD$ = LEFT$(readcmd2$, IBCNT% - 1)

        PRINT "          << Print measure data >> "
        PRINT "          Push any key then END !!        "

        '==========  Serect request ==========
        DO


            CALL wrtcmd2("ER?")
            RD$ = LEFT$(readcmd2$, IBCNT% - 1)
            LOCATE 10, 10
            PRINT "Error Ratio    ->  "; MID$(RD$, 5)

            CALL wrtcmd2("EC?")
            RD$ = LEFT$(readcmd2$, IBCNT% - 1)
            LOCATE 11, 10
            PRINT "Error Count    ->  "; MID$(RD$, 5)

            CALL wrtcmd2("EI?")
            RD$ = LEFT$(readcmd2$, IBCNT% - 1)
            LOCATE 12, 10
            PRINT "EI             ->  "; MID$(RD$, 5)

            CALL wrtcmd2("EFI?")
            RD$ = LEFT$(readcmd2$, IBCNT% - 1)
            LOCATE 13, 10
            PRINT "%EFI           -> " + MID$(RD$, 5) + "   "

            CALL wrtcmd2("FRQ?")
            RD$ = LEFT$(readcmd2$, IBCNT% - 1)
            LOCATE 14, 10
            PRINT "Clock Cycle    -> "; MID$(RD$, 5)

            CALL wrtcmd2("MSR?")
            RD$ = LEFT$(readcmd2$, IBCNT% - 1)
```

```
                IF RD$ = "MSR 0" THEN
                    CALL wrtcmd2("STA")
                END IF

                CALL waidly(1)

            LOOP UNTIL INKEY$ <> ""

            INPUT "Do you want measure try again? [Yes/No]:"; loop$
        LOOP UNTIL loop$ = "n" OR loop$ = "N"
    END IF
END IF

STOP
```

(8) Intermediate measurement data display

This program displays the intermediate measurement data on the CRT.





Note) If a system error occurs, the error contents are displayed and the program stops.

*1 Only the data of the specified item is read.

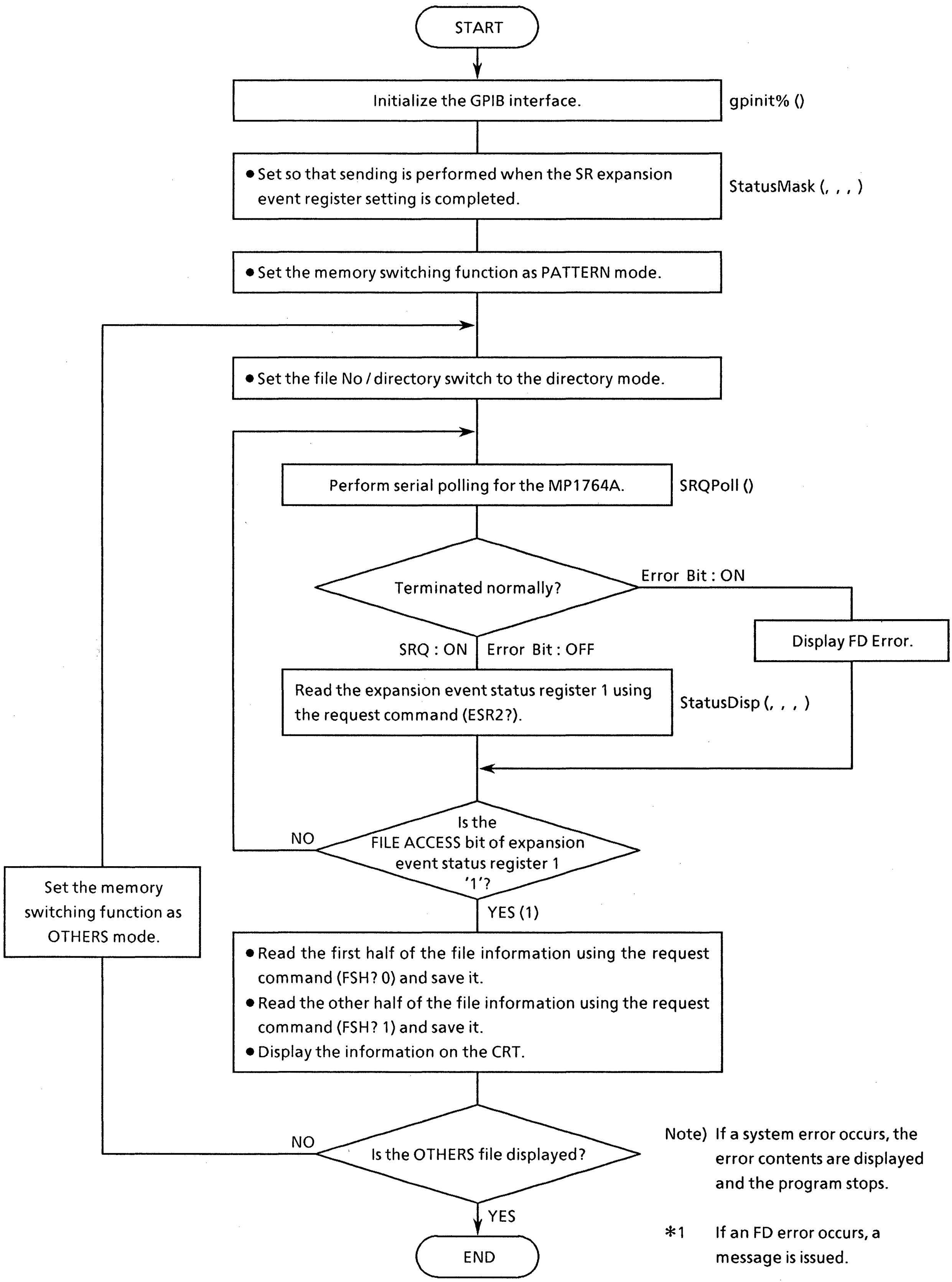
- Program list

```
REM $INCLUDE: 'c:¥at-gpib¥qbasic¥qbdecl.bas'

COMMON SHARED DEV%, GPIB0%, PPG%, ED%

DECLARE SUB waidly (tim!)
DECLARE SUB ClearDisp (p%, l%)
DECLARE SUB Displ (CMD$, RD$)
DECLARE SUB ChecClk ()
DECLARE SUB MeasStart ()
DECLARE SUB MeasStop ()
DECLARE SUB EndPoll ()
DECLARE SUB SelItem (item$())
DECLARE SUB wrtcmd2 (w$)
DECLARE SUB Connect (ttl$)
DECLARE SUB StatusMask (s0%, s1%, s2%, s3%)
DECLARE FUNCTION gpinit% ()
DECLARE FUNCTION AutoSrc% ()
DECLARE FUNCTION readcmd2$ ()

DIM item$(5, 7)                              'Command string

CLS
IF gpinit% <> 0 THEN                         'Setup interface
    '======= Set event status enable register =======
    CALL StatusMask(&H0, &H0, &H1, &H0)

    '===========  Set mode  ============
    CALL MeasStop
    CALL wrtcmd2("MOD 2")         'Meas mode              :Untimed

    CALL ClearDisp(16, 2)
    DO
        LOCATE 17, 1
        INPUT "AUTO SYNC CONDITION? [OFF:0, ON:1]", async%
        IF async% < 0 OR async% > 1 THEN
            LOCATE 16, 1
            PRINT "Wrong chosen number!! Please select corrct threshold."
            CALL ClearDisp(17, 4)
        END IF
    LOOP UNTIL async% >= 0 AND async% <= 1
    CALL wrtcmd2("SYN " + STR$(async%))

    CALL ClearDisp(16, 2)
    DO
        IF async% = 1 THEN
            LOCATE 17, 1
            PRINT "AUTO SYNC Threshold Ratio"
            PRINT "[ 1E-2:0 , 1E-3:1 , 1E-4:2 , 1E-5:3 , 1E-6:4 ] "
            PRINT "[ 1E-7:5 , 1E-8:6 ,        , INT :8         ] "
            INPUT "Choose Auto Sync Threshold:"; sye%
            IF sye% < 0 OR sye% > 8 THEN
                LOCATE 16, 1
                PRINT "Wrong chosen number!! Please select corrct threshold."
                CALL ClearDisp(17, 4)
            END IF
        END IF
    LOOP UNTIL sye% >= 0 AND sye% <= 8
    CALL ClearDisp(16, 5)
    CALL wrtcmd2("SYE " + STR$(sye%))

    CALL wrtcmd2("DMS 0")         'Error Display mode     : ERROR RATIO
    CALL wrtcmd2("CUR 1")         'Current data           : ON
    CALL wrtcmd2("TIM 4")         'Real time display mode : ELAPSED

    '==========  Draw layout box  ========
```

```
    CALL Connect("** MEASUREMENT SAMPLE PROGRAM **")

    '==========  Check clock  ============
    CALL ChecClk

    '======== Select display item ========
    CALL SelItem(item$())                'item: command string (output)
                                         'sort: command number (output)

    '========  Auto Search ON  ===========
    IF AutoSrc% = 1 THEN

        '========  Measurement start =========
        CLS
        SCREEN 0
        CALL MeasStart

        DO
            '========  wait trigger      =========

            LOCATE 23, 48
            PRINT "** Report Measure execution. **"
            LOCATE 23, 48
            PRINT "** Push ESC key then stop.    **"

            CALL wrtcmd2("IMS")                         'Data store
            CALL waidly(1)

            '========== Display result ===========

            LOCATE 1, 1
            FOR i = 0 TO 4
                FOR j = 0 TO 6
                    IF item$(i, j) <> "" THEN
                        CALL wrtcmd2("IMD? " + item$(i, j))  'Write command
                        RD$ = LEFT$(readcmd2$, IBCNT% - 1)
                        CALL Displ(item$(i, j), RD$)         'Display data
                    END IF
                NEXT j
            NEXT i
            CALL wrtcmd2("ESR2?")
            RD$ = readcmd2$                                  'Reset ESR1
        LOOP UNTIL INKEY$ = CHR$(&H1B)                       'ESC(&H1B)
    END IF
END IF
STOP
```

(9) Reading of floppy disk file information

This program reads file directory information stored on a floppy disk and displays it on the CRT.



Note) If a system error occurs, the error contents are displayed and the program stops.

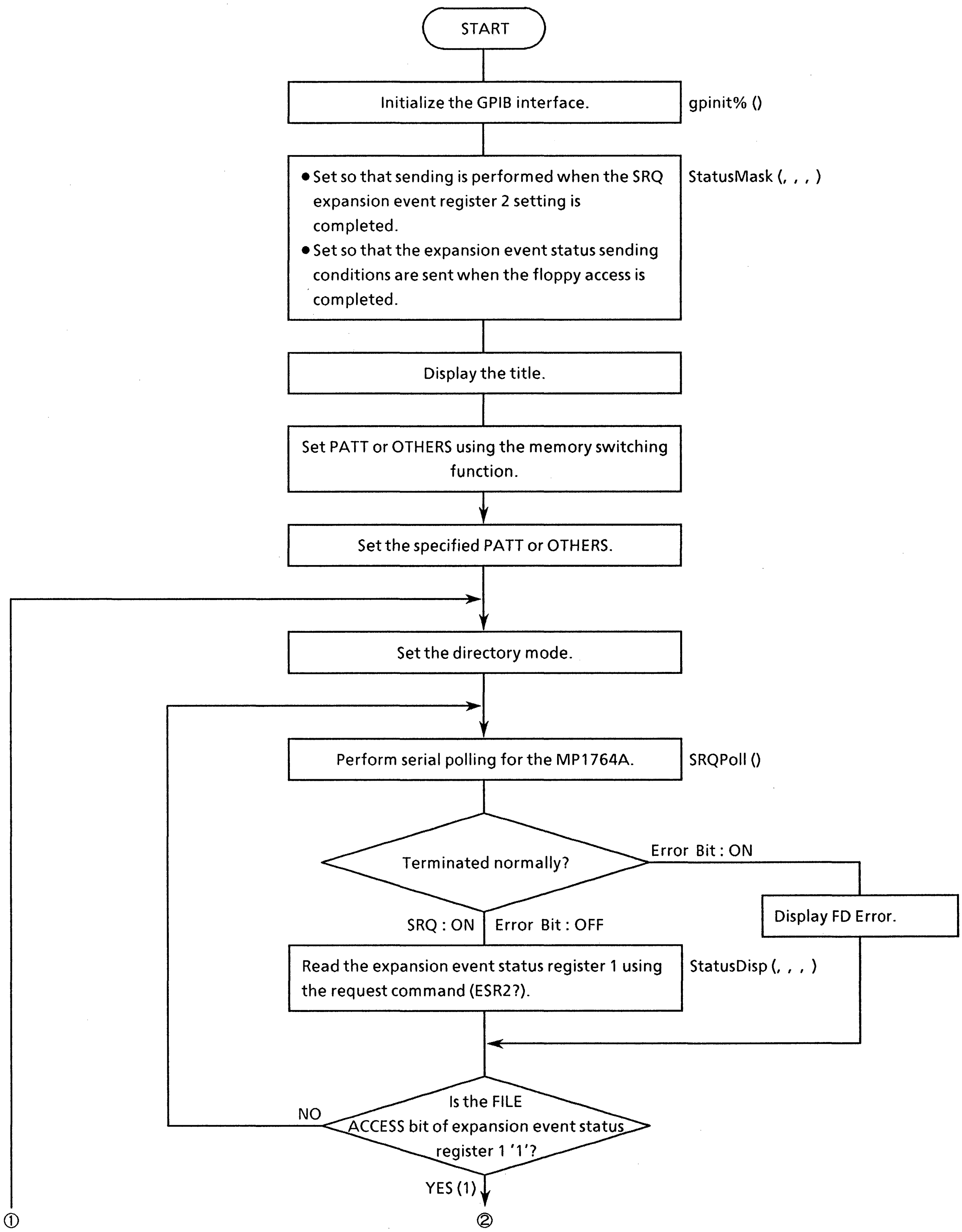
*1 If an FD error occurs, a message is issued.

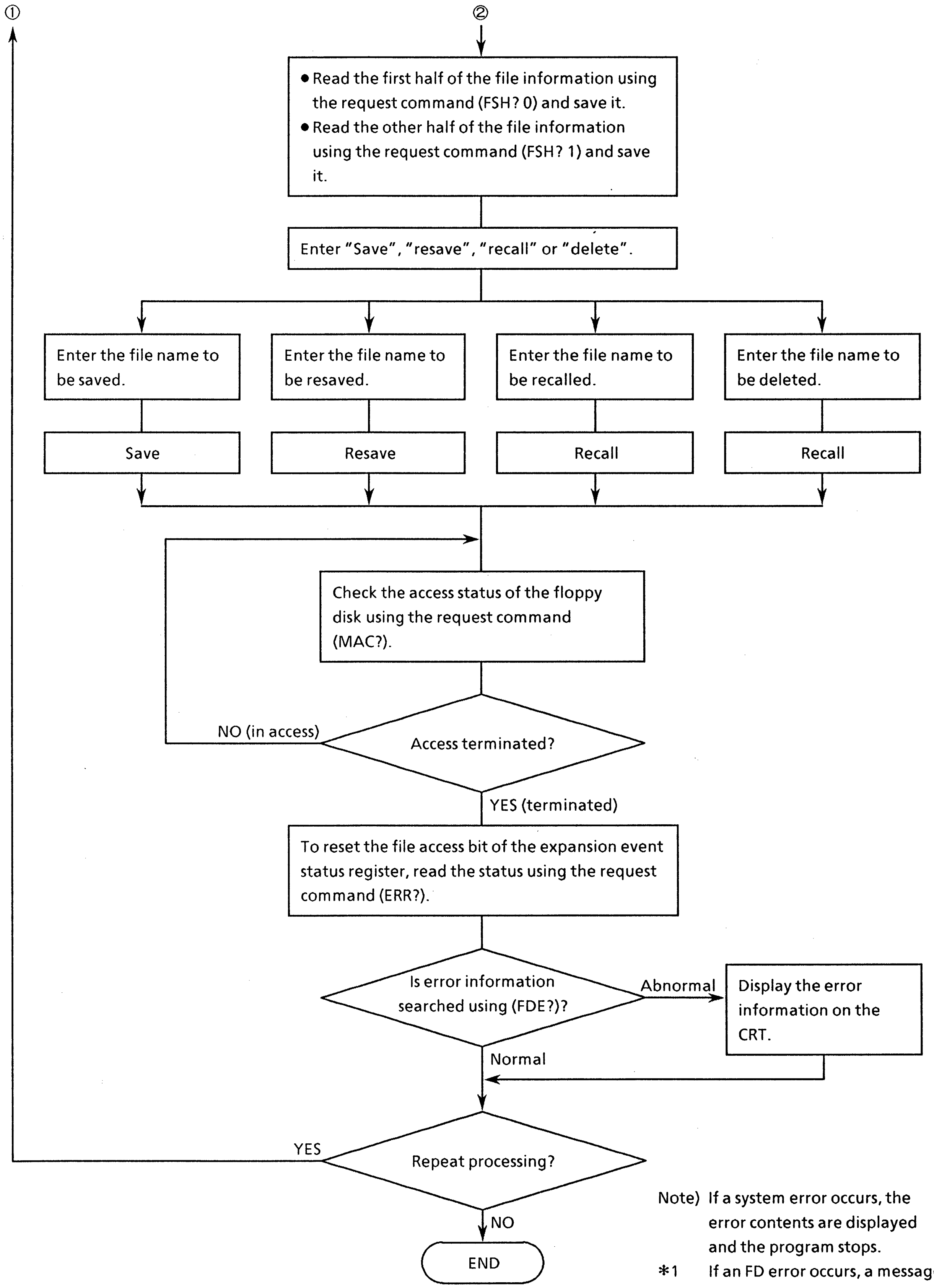
```
REM $INCLUDE: 'c:¥at-gpib¥qbasic¥qbdecl.bas'

COMMON SHARED DEV%, GPIB0%, PPG%, ED%

DECLARE SUB wrtcmd2 (w$)
DECLARE SUB ErrPoll ()
DECLARE SUB StatusDisp (stb%, esr%, esr2%, esr3%)
DECLARE SUB StatusMask (s0%, s1%, s2%, s3%)
DECLARE FUNCTION itob$ (1%, V%)
DECLARE FUNCTION SRQPoll% ()
DECLARE FUNCTION gpinit% ()
DECLARE FUNCTION readcmd2$ ()

IF gpinit% <> 0 THEN                              'Setup interface
    '====== Set MSS status byte register =======
    CALL StatusMask(&H4, &H0, &H2, &H2)

    FOR i = 0 TO 1
        '======== Set memory mode Pattern/Others =========
        wrtcmd2 ("MEM " + STR$(i))

        '========== Set FILE DIR mode ===========
        wrtcmd2 ("FIL 1")

        '========= Polling FILE ACCESS bit =========
        DO
            IF SRQPoll% <> 0 THEN
                CALL StatusDisp(dmy%, dmy1%, reg%, dmy3%)
            ELSE
                LOCATE 12, 35
                PRINT "FD error detect!!"
                EXIT DO
            END IF
         LOOP UNTIL reg% AND &H2

        '=========== Read FD infomation ============
         wrtcmd2 ("FDE?")
         rd1$ = LEFT$(readcmd2$, IBCNT% - 1)
         IF rd1$ <> "FDE 10" THEN
             LOCATE 1, 1
             SELECT CASE VAL(MID$(rd1$, 5, 2))
                 CASE 0
                     PRINT "<<E0:Media error                  >>"
                 CASE 1
                     PRINT "<<E1:Write protection error      >>"
                 CASE 2
                     PRINT "<<E2:File full                    >>"
                 CASE 3
                     PRINT "<<E3:File not found               >>"
                 CASE 4
                     PRINT "<<E4:File already exists error   >>"
                 CASE 5
                     PRINT "<<E5:Write error                  >>"
                 CASE 6
                     PRINT "<<E6:Read error                   >>"
                 CASE 7
                     PRINT "<<E7:File type , File error      >>"
                 CASE 8
                     PRINT "<<E8:FD error                     >>"
                 CASE 9
                     PRINT "<<E9:Hardware error               >>"
             END SELECT
             LOCATE 23, 1
             INPUT "End of FD analize. Press 'Enter' to fin."; f$
```

```
                STOP
            END IF
            wrtcmd2 ("FSH? 0")
            rd1$ = LEFT$(readcmd2$, IBCNT% - 1)
            wrtcmd2 ("FSH? 1")
            rd2$ = LEFT$(readcmd2$, IBCNT% - 1)

            '=============  Output CRT  ================
            IF i = 0 THEN
                LOCATE 4, 1
                PRINT "Pattern directory data."
            ELSE
                LOCATE 11, 1
                PRINT "Others directory data."
            END IF
            PRINT "Unused size : " + MID$(rd1$, 5, 7)       'print unused size
            PRINT "Used size   : " + MID$(rd1$, 13, 7)      'print used size
            PRINT "File count  : "; VAL(MID$(rd1$, 21, 22)) 'print file num
                                                            'print file no
            PRINT "File name   : "; MID$(rd1$, 24) + "," + MID$(rd2$, 24)
        NEXT i
        '
        PRINT
        INPUT "End of FD analize. Press 'Enter' to fin."; f$
    END IF

    STOP
```

(10) FD operation (data save, resave, and recall)

This program saves and resaves data to a floppy disk and recalls data from a floppy disk.





Note) If a system error occurs, the error contents are displayed and the program stops.

*1 If an FD error occurs, a message is issued.

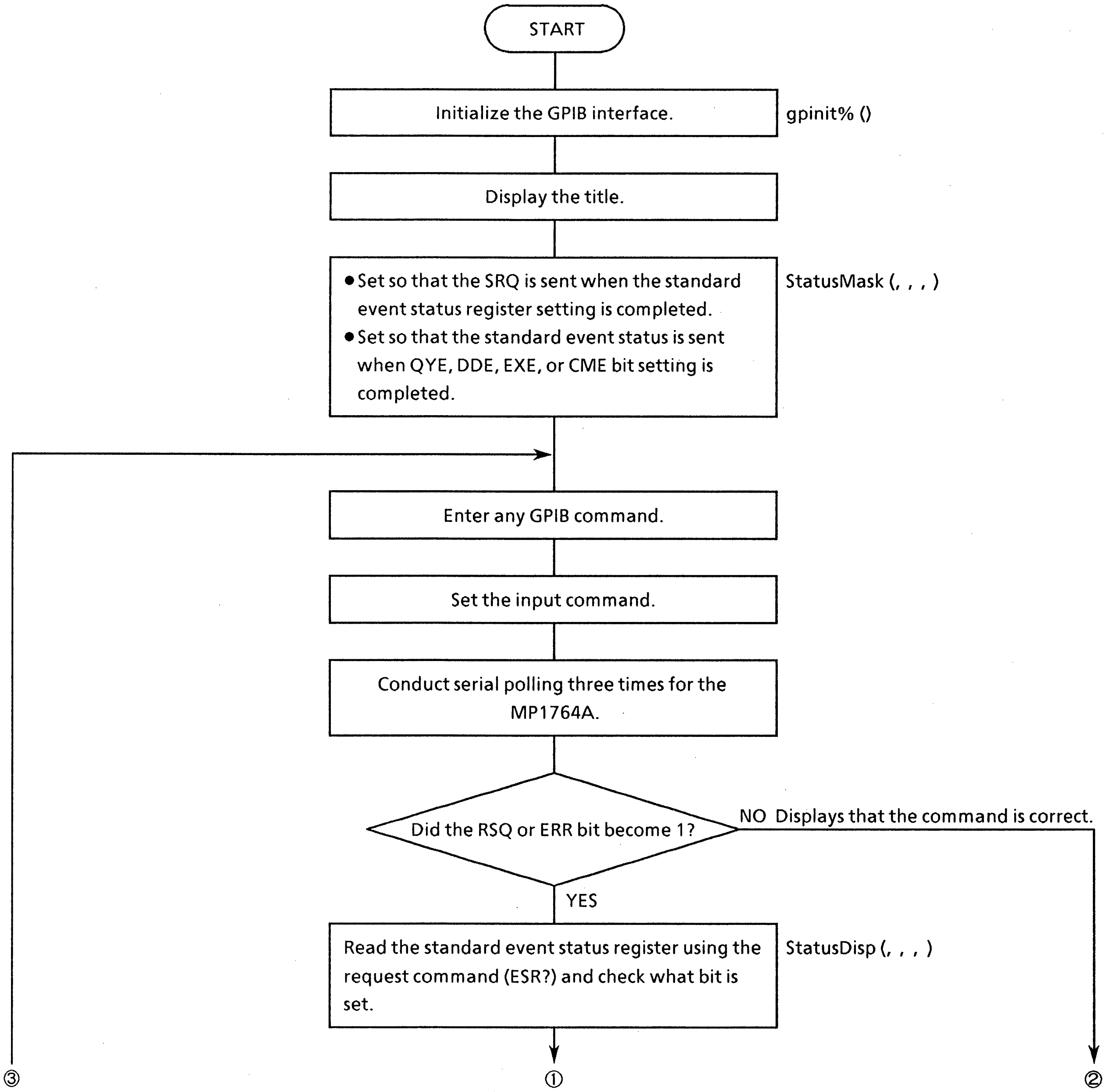
```
DECLARE SUB ClearDisp (p%, 1%)
REM $INCLUDE: 'c:¥at-gpib¥qbasic¥qbdecl.bas'

COMMON SHARED DEV%, GPIB0%, PPG%, ED%

DECLARE SUB waidly (tim!)
DECLARE SUB wrtcmd2 (w$)
DECLARE SUB ErrPoll ()
DECLARE SUB StatusDisp (stb%, esr%, esr2%, esr3%)
DECLARE SUB StatusMask (s0%, s1%, s2%, s3%)
DECLARE FUNCTION itob$ (1%, V%)
DECLARE FUNCTION SRQPoll% ()
DECLARE FUNCTION gpinit% ()
DECLARE FUNCTION readcmd2$ ()

IF gpinit% <> 0 THEN                              'Setup interface

    '====== Set MSS status byte register =======
    CALL StatusMask(&HC, &H0, &H2, &H2)

    DO
        CLS
        PRINT "** MP1762A/MP1764A FD OPERATION PROGRAM ** "
        '
        '===========  Select PTN/OTHERS  ==============
        DO
            LOCATE 17, 1
            INPUT "Memory mode select [ PATTERN:0 , OTHERS:1 ]"; mem$
            IF mem$ <> "0" AND mem$ <> "1" THEN
                LOCATE 16, 1
                PRINT "Wrong chosen number!! Please select a correct number"
            END IF
            CALL ClearDisp(16, 2)
        LOOP UNTIL mem$ = "0" OR mem$ = "1"
        wrtcmd2 ("MEM " + mem$)


        '========== Set FILE DIR mode ===========
        wrtcmd2 ("FIL 1")

        '========= Polling FILE ACCESS bit =========
        DO
            IF SRQPoll% <> 0 THEN
                CALL StatusDisp(dmy%, dmy1%, reg%, dmy3%)
            ELSE
                GOSUB Fderr
                GOTO jump
            END IF
        LOOP UNTIL reg% AND &H2

        '==== Save, Resave, Recall or Delete? =====
        DO
            LOCATE 17, 1
            INPUT "Choose function [ SAVE:0 , RESAVE:1 , RECALL:2 , DELEAT:3 ]";
 op%
            IF op% < 0 OR op% > 3 THEN
                LOCATE 16, 1
                PRINT "Wrong chosen number!! Please select correct function."
            END IF
        LOOP UNTIL op% >= 0 AND op% <= 3
        CALL ClearDisp(16, 2)

        LOCATE 17, 1
        SELECT CASE op%
```

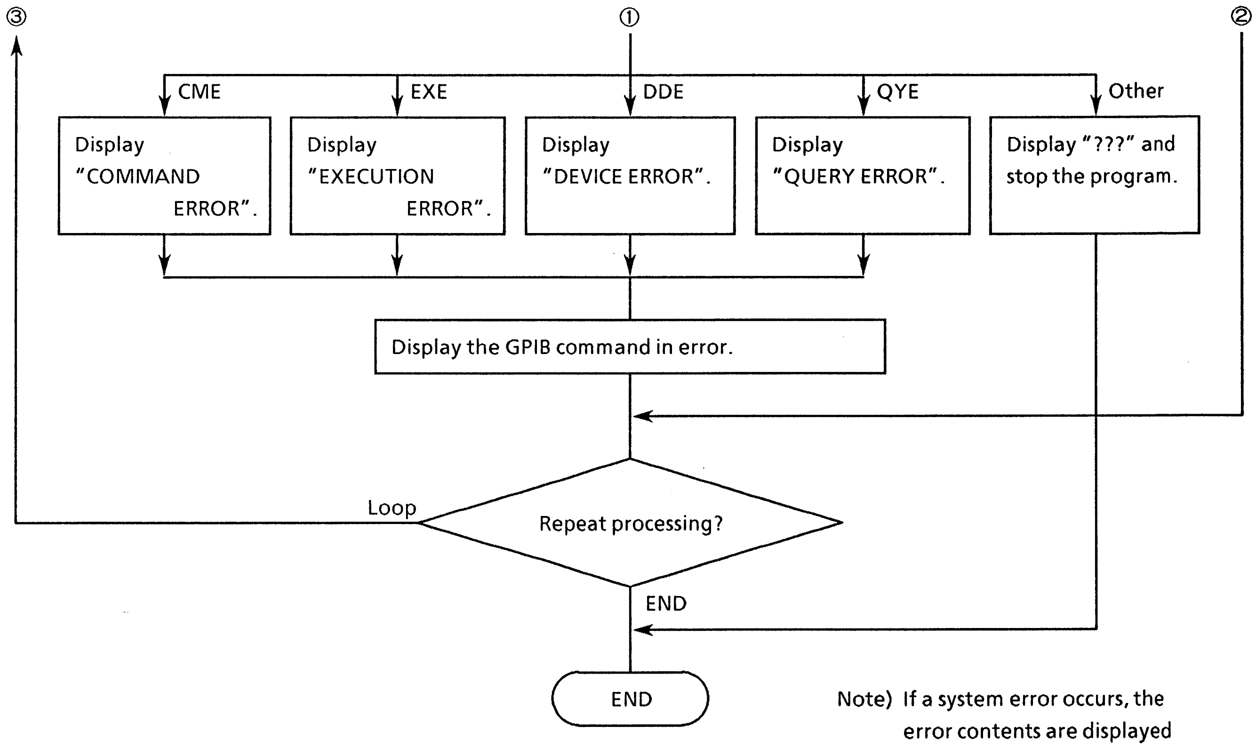
```
        CASE 0
            INPUT "Enter file number for SAVE:"; NO$
            wrtcmd2 ("SAV " + NO$)
            '
        CASE 1
            INPUT "Enter file number for RESAVE:"; NO$
            wrtcmd2 ("RSV " + NO$)
            '
        CASE 2
            INPUT "Enter file number for RECALL:"; NO$
            wrtcmd2 ("RCL " + NO$)
            '
        CASE 3
            INPUT "Enter file number for DELEAT:"; NO$
            wrtcmd2 ("DEL " + NO$)
            '
        END SELECT
        GOSUB Faccess
        GOSUB Fderr
        CALL ClearDisp(17, 1)

        '======= Reset EventStatusRegister1 ========
jump:   CALL StatusDisp(dmy%, dmy1%, dmy2%, dmy3%)
        '
        LOCATE 17, 1
        INPUT "Do you more test another function? [Yes/No]"; loop$
    LOOP UNTIL loop$ = "n" OR loop$ = "N"
END IF
'
STOP
'
Faccess: '============ FD access end ?  =============
        '
        DO
            CALL StatusDisp(dmy%, dmy1%, dmy2%, dmy3%)
            waidly (1)
            wrtcmd2 ("MAC?")
            RD$ = LEFT$(readcmd2$, IBCNT% - 1)
        LOOP UNTIL MID$(RD$, 1, 5) = "MAC 0"
        '

RETURN
'
Fderr: '=========== FD error message ===============
        '
        CALL wrtcmd2("FDE?")
        RD$ = LEFT$(readcmd2$, IBCNT% - 1)
        LOCATE 10, 1
        IF RD$ <> "FDE 10" THEN
            PRINT "FD error occuerd!! "
            SELECT CASE MID$(RD$, 6, 1)
                CASE "0"
                    PRINT "E0:Media error                          "
                CASE "1"
                    PRINT "E1:Write protection error               "
                CASE "2"
                    PRINT "E2:File full                            "
                CASE "3"
                    PRINT "E3:File not found                       "
                CASE "4"
                    PRINT "E4:File already exists error            "
                CASE "5"
                    PRINT "E5:Write error                          "
```

```
            CASE "6"
                PRINT "E6:Read error                                  "
            CASE "7"
                PRINT "E7:File type , File error                      "
            CASE "8"
                PRINT "E8:FD error                                    "
            CASE "9"
                PRINT "E9:Hardware error                              "
        END SELECT
    ELSE
        PRINT "<<  ** FD operation complete!! **   >>"
        PRINT "<<  ** Accept file number is " + NO$ + ". **  >>"
    END IF
    '
RETURN
'
```

(11) Standard status byte (4 types) checking

This program checks the standard status bytes (QYE, DDE, EXE, and CME bits).





Note) If a system error occurs, the error contents are displayed and the program stops.

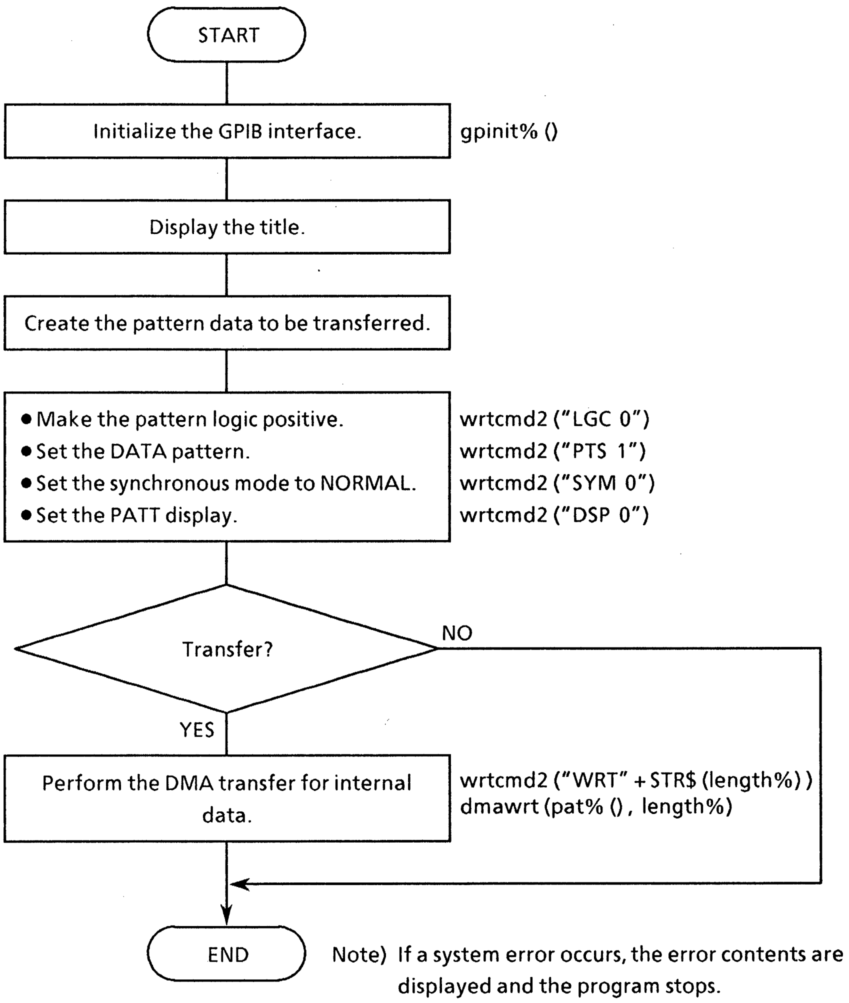
- Program list

```
REM $INCLUDE: 'c:¥at-gpib¥qbasic¥qbdecl.bas'

COMMON SHARED DEV%, GPIB0%, PPG%, ED%

DECLARE SUB waidly (tim!)
DECLARE SUB wrtcmd2 (WRT$)
DECLARE SUB trap ()
DECLARE SUB ClearDisp (p%, 1%)
DECLARE SUB StatusDisp (stb%, esr%, esr2%, esr3%)
DECLARE SUB StatusMask (s0%, s1%, s2%, s3%)
DECLARE FUNCTION itob$ (1%, V%)
DECLARE FUNCTION gpinit% ()
DECLARE FUNCTION readcmd2$ ()

CLS
IF gpinit% <> 0 THEN                              'Setup interface

    PRINT "** MP1762A/MP1764A STANDARD STATUS REGISTER CHECK **"
    PRINT

    '====== Set MSS status byte register =======
    CALL StatusMask(&H3C, &H7E, &H77F, &H3)

    DO
        CALL ClearDisp(5, 15)
        LOCATE 5, 1
        INPUT "Please enter some GPIB command(s):"; com$
        length% = LEN(com$)
        CALL wrtcmd2(com$)
        LOCATE 5, 1
        PRINT "Please enter some GPIB command(s):                          "

        sta% = 0
        FOR i = 0 TO 2
            CALL IBRSP(ED%, SPR%)
            IF IBSTA < 0 THEN CALL trap
            sta% = sta% OR SPR%

            sta$ = itob$(8, SPR%)
            LOCATE 1, 60
            PRINT "*SRE:"; sta$

            waidly (.1)
        NEXT i

        IF (sta% AND &H20) THEN
            LOCATE 7, 1
            PRINT "Execution command(s) fail of ''                            "
            IF length% > 0 THEN
                LOCATE 7, 1
                PRINT "Execution command(s) fail of '"; LEFT$(com$, length%); "'
"
            END IF

            CALL StatusDisp(dmy%, reg%, dmy2%, dmy3%)

            CALL ClearDisp(8, 6): LOCATE 8, 1
            IF (reg% AND &H2) OR (reg% AND &H40) THEN PRINT " ??? ": STOP
            IF reg% AND &H4 THEN PRINT "* QUERY ERROR *"
            IF reg% AND &H8 THEN PRINT "* DEVICE ERROR *"
            IF reg% AND &H10 THEN PRINT "* EXECUTION ERROR *"
            IF reg% AND &H20 THEN PRINT "* COMMAND ERROR *"
        ELSE
```

```
            LOCATE 7, 1
            PRINT "Command succed execution.                                   "
            CALL ClearDisp(9, 6)
        END IF

        LOCATE 15, 36: PRINT "                                   "
        LOCATE 15, 1
        INPUT "Do you test other command? [Yes/No] "; loop$
    LOOP UNTIL loop$ = "n" OR loop$ = "N"
END IF
'
STOP
```

(12) Pattern data DMA transfer processing

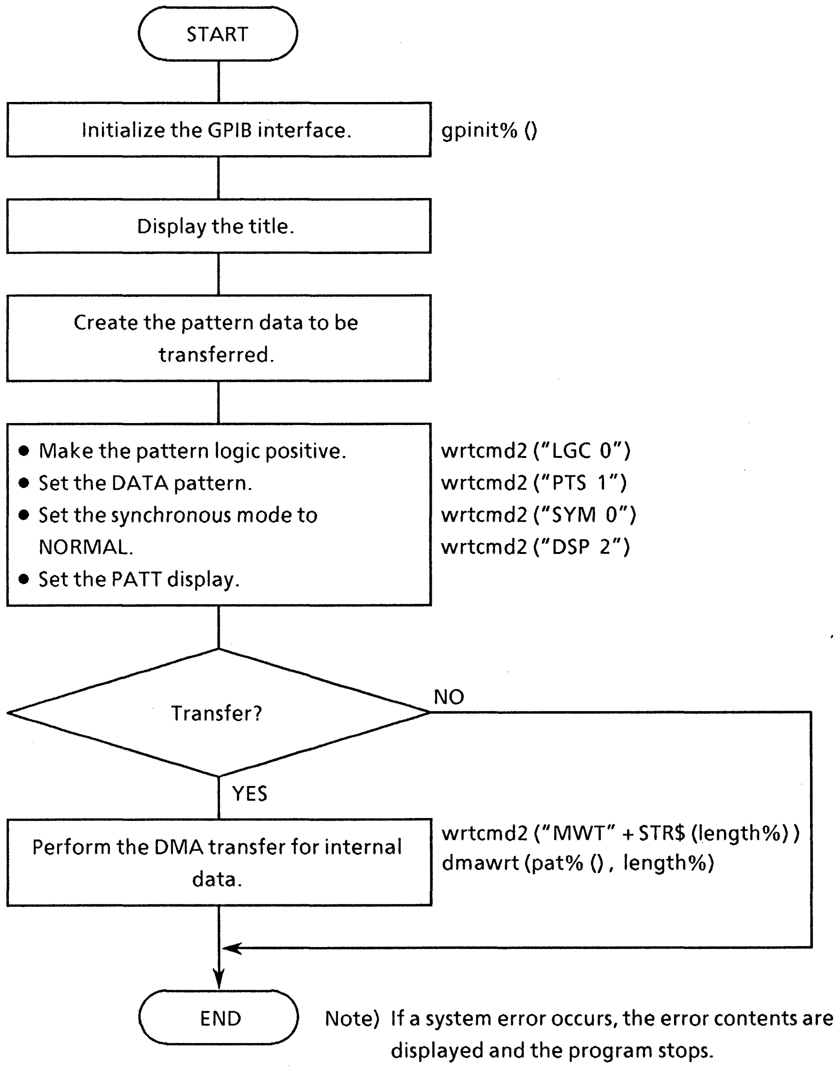
This program performs the DMA transfer for pattern data.



```
REM $INCLUDE: 'c:¥at-gpib¥qbasic¥qbdecl.bas'

COMMON SHARED DEV%, gpib0%, PPG%, ED%

DECLARE SUB StatusMask (s0%, s1%, s2%, s3%)
DECLARE SUB StatusDisp (stb%, esr%, esr2%, esr3%)
DECLARE SUB wrtcmd2 (w$)
DECLARE SUB dmawrt (w%(), i%)
DECLARE SUB gpiberr (msg$)
DECLARE FUNCTION gpinit% ()
DECLARE FUNCTION Exchange% (i%)

DIM pat%(302)

IF gpinit% <> 0 THEN            'Setup interface
    CLS
    PRINT "** MP1762A/MP1764A DMA(pattern data) SAMPLE PROGRAM ** "
    PRINT

    CALL StatusMask(&H0, &H0, &H0, &H0)
    '
    '===========   Table   ============
    ' Test pattern set and swap data.
    ' if you use ibconfig() swap function,
    ' then don't call Exchange() function. Because, its same operation.
    '
    Dlength% = 300                                  ':Max Page
    j% = 0
    FOR i% = 0 TO Dlength% - 1
        pat%(i%) = Exchange(j%)
        j% = j% + 1
    NEXT i%
    pat%(i%) = &HA

    '============  initial  ================
    CALL wrtcmd2("LGC 0")       'Pattern logic         : positive
    CALL wrtcmd2("PTS 1")       'Pattern               : data
    CALL wrtcmd2("SYM 0")       'Sync mode             : normal
    CALL wrtcmd2("DSP 0")       'Display               : pattern
    CALL wrtcmd2("DLN " + STR$(Dlength% * 16)): DATA Length

    CALL StatusDisp(dmy%, dmy1%, dmy2%, dmy3%)

    INPUT "Do you wish transmit are PATTERN data? [Yes/No]:"; a$
    IF a$ = "y" OR a$ = "Y" THEN
        CALL wrtcmd2("WRT " + STR$(Dlength% * 2) + ",0")

       ' CALL ibconfig(gpib0%, 20, 1)
       ' CALL gpiberr("'ibconfig' execute status")

        CALL dmawrt(pat%(), Dlength%)
    END IF
END IF
'
STOP
```

(13) DMA transfer for BLOCK WINDOW data

This program performs the DMA transfer for BLOCK WINDOW data.



- Program list

```
' ---- Note ----
' If you are use to control command of 'MWT' then you must set a data length
' to multiplication by integral number of 32-bit.
'
REM $INCLUDE: 'c:¥at-gpib¥qbasic¥qbdecl.bas'

COMMON SHARED DEV%, GPIB0%, PPG%, ED%

DECLARE SUB StatusMask (s0%, s1%, s2%, s3%)
DECLARE SUB StatusDisp (stb%, esr%, esr2%, esr3%)
DECLARE SUB wrtcmd2 (WRT$)
DECLARE SUB dmawrt (w%(), i%)
DECLARE FUNCTION gpinit% ()

DIM pat%(310)

IF gpinit% <> 0 THEN              'Setup interface
    CLS
    PRINT "** MP1762A/MP1764A DMA(block window data) SAMPLE PROGRAM ** "
    PRINT

    CALL StatusMask(&H0, &H0, &H0, &H0)
    '
    '===========   Table   ============
    ' Test pattern set and swap data.
    ' if you use ibconfig() swap function,
    ' then don't call Exchange() function. Because, its same operation.
    '
    Dlength% = 300
    FOR i% = 0 TO Dlength% - 1
        pat%(i%) = &H5555
    NEXT i%
    pat%(i%) = &HA                'Append termination code (LF)

    '============  initial  ================
    CALL wrtcmd2("LGC 0")         'Pattern logic : positive
    CALL wrtcmd2("PTS 1")         'Pattern       : data
    CALL wrtcmd2("SYM 0")         'Sync mode     : normal
    CALL wrtcmd2("DSP 2")         'Display       : pattern

    CALL StatusDisp(dmy%, dmy1%, dmy2%, dmy3%)

    INPUT "Do you wish transmit are BLOCK WINDOW data? [Yes/No]:"; a$
    IF a$ = "y" OR a$ = "Y" THEN
        CALL wrtcmd2("MWT " + STR$(Dlength% * 2) + ",0")

        CALL dmawrt(pat%(), Dlength%)
    END IF
END IF
'
STOP
```

(Blank)

APPENDIX A COMPATIBILITY WITH CONVENTIONAL INSTRUMENTS

When the mainframe uses programs generated by conventional instruments (MP1702A / MP1609A / MP1651A), some additional editing is required for programming.

(1) Status common command structure

MP1742A supports some of the common commands that conform to 488.2, but these commands are expressed without * in MP1702A.

Command	MP1702A	MP1764A
Service request	STB?	*STB?
Service request enable register	SRQ	*SRE
Standard event status register	ESR?	*ESR?
Standard event status enable register	ESE	*ESE
Expansion event status register	EER?	ESR2? ESR3?
Expansion event status enable register	EES	ESE2? ESE3?

See " Section 8: STATUS STRUCTURE " for other common commands and statuses.

(2) Measurement data calculation and read out

Read commands for MP1764A measurement data are output in response form for data request messages.

Also, data saved in the internal buffer is output as measurement results data.

Re-edit programming for measurement data calculation.

(3) Other GPIB commands

Appendix table A-1 shows a the correspondence between GPIB and MP1702A commands.

Re-edit programming by referring to the GPIB instruction manuals for each instrument.

○ : Commands common with MP1702A
× : Commands not common with MP1702A

Table A-1 Table of Device Messages

Function	Control message		Data request message	Device message details		Compatibility with MP1702A
	Header part	Numeric data part	Header part	No.	Page	
● INPUT section						
Data input threshold voltage	DTH	NR2 format	DTH?	1	P9-26	○
Eye margin measurement result (threshold)	–	–	THM?	2	P9-27	×
Clock input phase	CPA	NR1 format	CPA?	3	P9-28	○
Eye margin measurement result (phase)	–	–	PHM?	4	P9-29	×
Data input termination voltage	DTM	NR1 format	DTM?	5	P9-30	○
Clock input termination voltage	CTM	NR1 format	CTM?	6	P9-31	○
Delay status	–	–	DLY?	7	P9-32	○
Automatic phase threshold search	SRH	NR1 format	SRH?	8	P9-33	○
Clock input polarity	CPL	NR1 format	CPL?	9	P9-34	○
Eye margin measurement display switching	EME	NR1 format	EME?	10	P9-35	×
Eye margin measurement start	EST	NR1 format	EST?	11	P9-36	×
Eye margin measurement (Error ratio selection)	EYT	NR1 format	EYT?	12	P9-37	×
● MEMORY section						
Memory FD mode	–	–	FMD?	13	P9-39	×
File content search	–	–	FSH?	14	P9-40	×
File No./directory mode switching	FIL	NR1 format	FIL?	15	P9-42	×
Floppy data recall	RCL	NR1 format	–	16	P9-43	○
Floppy data delete	DEL	NR1 format	–	17	P9-44	×
Floppy data save	SAV	NR1 format	–	18	P9-45	○
Floppy data resave	RSV	NR1 format	–	19	P9-46	○
Memory mode switching	MEM	NR1 format	MEM?	20	P9-47	○
Floppy access status	–	–	MAC?	21	P9-49	○
FD error message	–	–	FDE?	22	P9-50	×
FD format	FDF	–	–	23	P9-51	×
● PATTERN section						
Pattern logic	LGC	NR1 format	LGC?	24	P9-53	○

Table A-1 Table of Device Messages (contd.)

Function	Control message		Data request message	Device message details		Compati-bility with MP1702A
	Header part	Numeric data part	Header part	No.	Page	
● PATTERN section (contd.)						
Measurement pattern	PTS	NR1 format	PTS?	25	P9-54	×
Number of ZERO SUBST and PRBS steps	PTN	NR1 format	PTN?	26	P9-55	○
PRBS mark ratio	MRK	NR1 format	MRK?	27	P9-56	○
Synchronous mode	SYM	NR1 format	SYM?	28	P9-57	×
Display selection	DSP	NR1 format	DSP?	29	P9-58	×
Alternate pattern A / B switching	ALT	NR1 format	ALT?	30	P9-59	×
Frame bit length	FLN	NR1 format	FLN?	31	P9-60	○
Data length	DLN	NR1 format	DLN?	32	P9-61	×
ZERO SUBST length	ZLN	NR1 format	ZLN?	33	P9-63	×
Number of pages. Pattern synchronous trigger position	PAG ADR	NR1 format NR1 format	PAG? ADR?	34	P9-64	×
Pattern bit	BIT	NR1 format HEX format	BIT?	35	P9-65	○
Bit window pattern	CHM	NR1 format HEX format	CHM?	36	P9-67	×
Bit window page	MSK	NR1 format	MSK?	37	P9-69	×
Block window pattern	MGB	NR1 format HEX format	MGB?	38	P9-70	×
Error analysis data	–	–	EAB?	39	P9-72	×
Error analysis page	EAP	NR1 format	EAP?	40	P9-74	×
Bit window ON/OFF	MSE	NR1 format	MSE?	41	P9-75	×
Block window ON/OFF	MGE	NR1 format	MGE?	42	P9-76	×
Error analysis trigger	EAT	NR1 format	EAT?	43	P9-77	×
Number of bytes of pattern data input	WRT	NR1 format	–	44	P9-78	×
Number of bytes of pattern data output	–	–	RED?	45	P9-80	×
Number of bytes of block window data input	MWT	NR1 format	–	46	P9-81	×
Number of bytes of block window data output	–	–	MRD?	47	P9-83	×
Pattern data preset (all pages, all bits)	ALL	NR1 format	–	48	P9-85	○

Table A-1 Table of Device Messages (contd.)

Function	Control message: Header part	Control message: Numeric data part	Data request message: Header part	Device message details: No	Device message details: Page	Compatibility with MP1702A
● PATTERN section (contd.)						
Pattern data preset (1 page, all bits)	PST	NR1 format	–	49	P9-86	○
Block window preset (all pages, all bits)	MAL	NR1 format	–	50	P9-87	×
Block window preset (1 page, all bits)	MPS	NR1 format	–	51	P9-88	×
Bit window preset (all pages, all bits)	HAL	NR1 format	–	52	P9-89	×
Bit window preset (1 page, all bits)	HPS	NR1 format	–	53	P9-90	×
PATTERN SYNC POSITION	PSP	NR1 format	PSP?	54	P9-91	
PAGE / PATTERN SYNC POSITION switch	PPD	NR1 format	PPD?	55	P9-92	
● MEASUREMENT section						
Clock off status	–	–	CLI?	56	P9-94	○
Synchronous loss status	–	–	SLI?	57	P9-95	○
Error detection status	–	–	ERS?	58	P9-96	○
Measurement result display mode	DMS	NR1 format	DMS?	59	P9-97	○
Intermediate result display function	CUR	NR1 format	CUR?	60	P9-98	○
Measurement mode	MOD	NR1 format	MOD?	61	P9-99	○
Measurement start and restart	STA	–	–	62	P9-100	○
Measurement stop	STO	–	–	63	P9-101	○
Measurement status	–	–	MSR?	64	P9-102	○
Automatic synchronous function	SYN	NR1 format	SYN?	65	P9-103	○
Automatic synchronous threshold	SYE	NR1 format	SYE?	66	P9-104	×
Real time and measurement period display switching	TIM	NR1 format	TIM?	67	P9-105	○
Internal timer setting	RTM	NR1 format	RTM?	68	P9-106	○
Measurement period setting	PRD	NR1 format	PRD?	69	P9-107	○
Error ratio measurement result	–	–	ER?	70	P9-108	○
Number of errors of measurement result	–	–	EC?	71	P9-109	○
Number of EIs of measurement result	–	–	EI?	72	P9-110	○
%EFI measurement result	–	–	EFI?	73	P9-111	○
Clock frequency data	–	–	FRQ?	74	P9-112	○
1-second data measurement result	–	–	OSD?	75	P9-113	×
Alarm measurement result	–	–	AMD?	76	P9-114	×

Table A-1 Table of Device Messages (contd.)

Function	Control message		Data request message	Device message details		Compatibility with MP1702A
	Header part	Numeric data part	Header part	No.	Page	
● MEASUREMENT section (contd.)						
Stores measurement end data in buffer	EDS	–	–	77	P9-116	×
Clears measurement end data from buffer	EDC	–	–	78	P9-117	×
Measurement end data output	–	–	END?	79	P9-118	×
Stores intermediate measurement data in buffer	IMS	–	–	80	P9-121	×
Clears intermediate measurement data from buffer	IMC	–	–	81	P9-122	×
Intermediate measurement data output	–	–	IMD?	82	P9-123	×
● Other section (Front panel)						
Printer ON / OFF	PRN	NR1 format	PRN?	83	P9-127	○
Manual print	PSA	NR1 format	–	84	P9-128	○
Alarm monitor (Alarm detection)	ALM	NR1 format	ALM?	85	P9-129	○
Alarm monitor (Error detection)	MON	NR1 format	MON?	86	P9-130	○
Synchronous signal output	SOP	NR1 format	SOP?	87	P9-131	×
● Other section (Rear panel GPIB)						
GPIB 2 address	GPA	NR1 format	GPA?	88	P9-132	×
● Other section (rear panel FUNCTION switch)						
Number of shifts of the mark ratio AND bit	SFT	NR1 format	SFT?	89	P9-133	○
Clock off processing	CLS	NR1 format	CLS?	90	P9-134	○
Synchronous loss processing	SLS	NR1 format	SLS?	91	P9-135	○
Error performance data threshold selection	ETH	NR1 format	ETH?	92	P9-136	○
BURST measurement mode	BST	NR1 format	BST?	93	P9-137	×
Intermediate data calculation during measurement	CAL	NR1 format	CAL?	94	P9-138	○
Error detection mode	ETY	NR1 format	ETY?	95	P9-139	○
EI, EFI interval period	EIT	NR1 format	EIT?	96	P9-140	×

Table A-1 Table of Device Messages (contd.)

Function	Control message		Data request message	Device message details		Compatibility with MP1702A
	Header part	Numeric data part	Header part	No.	Page	
● OTHER section (Rear panel FUNCTION switch) (contd.)						
Data print format	FMT	NR1 format	FMT?	97	P9-141	○
Threshold EI, EFI data print	THR	NR1 format	THR?	98	P9-142	○
Error performance data print	EPF	NR1 format	EPF?	99	P9-143	○
Intermediate data print	ITM	NR1 format	ITM?	100	P9-144	○
1-second data print	OSC	NR1 format	OSC?	101	P9-145	○
1-second data print threshold	DOT	NR1 format	DOT?	102	P9-146	○
Paper saving function	PSV	NR1 format	PSV?	103	P9-147	○
Measurement interval time	ITV	NR1 format	ITV?	104	P9-148	×
Memory FD mode	–	–	FMD?	13	P9-39	×

APPENDIX B PATTERN DMA TRANSFER

DMA, Direct Memory Access, transfers large amounts of data at high speed.

This mainframe has 4 types of DMA transfer commands. These are explained below.

(1) Pattern data DMA transfer command (WRT)

When the measurement pattern is ALTERNATE or DATA, this command transfers the contents of the program pattern DMA transfer in advance.

This command is used to notify the following information to the mainframe using WRT.

1 : How many data bytes are transferred?
2 : In which address in the mainframe internal RAM area, is the transferred pattern data stored?

1) How many data bytes are transferred?

The mainframe construction is 16 bit pattern. Therefore, a display of 1 page (16 bits) of the BIT display unit is normally transferred in 2 bytes.

The correspondence between pattern data and bit is as follows.

bit 15							bit 8	bit 7							bit 0	
16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT LED No.

When an odd byte is transferred, only the upper bits (bit 15 to bit 8) are specified.

2) In which address in the mainframe internal RAM area, is the transferred pattern data stored?

The RAM address of the mainframe is:

When DATA: 1 to 524287
When ALTERNATE: 0 to 262143

ALTERNATE has pattern A and pattern B so the internal RAM area is divided in half. (A or B pattern is selected according to the switch status of the A / B display switching).

The following shows the relationship between the address and the actual numbers of pages.

Pattern input address		Internal RAM area		Number of display pages
0	→		→	1
1	→		→	2
2	→		→	3
3	→		→	4
4	→		→	5
:				:
:				:
:				:
:				:
:				:
524286	→		→	524287
524287	→		→	524288

(2) Pattern data DMA transfer command (RED?)

When the measurement pattern is ALTERNATE or DATA, this command transfers the contents of the program pattern DMA transfer in advance.

This command is used to notify the following information to the mainframe using WRT.

1 : How many data bytes are transferred?
2 : In which address in the mainframe internal RAM area, is the transferred pattern data stored?

1) How many data bytes are transferred?

The mainframe construction is 16 bit pattern. Therefore, a display of 1 page (16 bits) of the BIT display unit is normally transferred in 2 bytes.

The correspondence between pattern data and bit is as follows.

bit 15							bit 8	bit 7							bit 0	
16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT LED No.

When an odd byte is transferred, only the upper bits (bit 15 to bit 8) are specified.

2) In which address in the mainframe internal RAM area, is the transferred pattern data stored?

The RAM address of the mainframe is:

When DATA: 0 to 524287
When ALTERNATE: 0 to 262143

ALTERNATE has pattern A and pattern B, so the internal RAM area is divided in half. (A or B pattern is selected according to the switch status of the A / B display switching).

The following shows the relationship between the address and the actual numbers of pages.

Pattern input address		Internal RAM area		Number of display pages
0	←		←	1
1	←		←	2
2	←		←	3
3	←		←	4
4	←		←	5
:				:
:				:
:				:
:				:
:				:
524286	←		←	524287
524287	←		←	524288

(3) BLOCK WINDOW data DMA transfer command (MWT)

For BLOCK WINDOW data, this command transfers the contents of the BLOCK WINDOW pattern DMA transfer in advance.

This command is used to notify the following information to the mainframe using MWT.

1 : How many data bytes are transferred?
2 : In which address in the mainframe internal RAM area, is the transferred pattern data stored?

1) How many data bytes are transferred?

The BLOCK WINDOW data is in 32-bit units, so 32 bits (2 pages) of the BIT display data is saved in the internal RAM area.

Therefore, in contrast with (1) and (2) previously, the specified 1 bit expresses 32 bits of mainframe BLOCK WINDOW DATA.

The correspondence between pattern data and bit is as follows.

The above are the page values when the header address is assumed to be zero (0). So add 32 every time the above page address increases by 1.

bit 15							bit 8	bit 7							bit 0	
15	13	11	9	7	5	3	1	31	29	27	25	23	21	19	17	PAGE No.
16	14	12	10	8	6	4	2	32	30	28	26	24	22	20	18	

When an odd byte is transferred, only the upper bits (bit 15 to bit 8) are specified.

2) In which address in the mainframe internal RAM area, is the transferred pattern data stored?

The RAM address of the mainframe is:

When DATA: 0 to 16383
When ALTERNATE: 0 to 8192

ALTERNATE has pattern A and pattern B, so the internal RAM area is divided in half. (A or B pattern is selected according to the switch status of the A / B display switching).

The following shows the relationship between the address and the actual numbers of pages.

Pattern output address		Internal RAM area		Number of display pages
0	→		→	1 to 32 pages
1	→		→	33 to 64 pages
2	→		→	65 to 96 pages
3	→		→	97 to 128 pages
4	→		→	129 to 160 pages
⋮				⋮
16382	→		→	524225 to 524256 pages
16383	→		→	524257 to 524288 pages

(4) DMA transfer command for BLOCK WINDOW data (MRD?)

For BLOCK WINDOW data, this command transfers the contents of the BLOCK WINDOW pattern DMA transfer of in advance.

This command is used to notify the following information to the mainframe using MRD.

1 : How many data bytes are transferred?
2 : In which address in the mainframe internal RAM area, is the transferred pattern data stored?

1) How many data bytes are transferred?

The BLOCK WINDOW data is in 32-bit units, so 32 bits (2 pages) data of the BIT display data is saved in the internal RAM area.

Therefore, in contrast with (1) and (2) previously, the specified 1 bit expresses 32 bits of mainframe BLOCK WINDOW data.

The correspondence between pattern data and bit is as follows.

bit 15							bit 8	bit 7							bit 0	
15	13	11	9	7	5	3	1	31	29	27	25	23	21	19	17	PAGE No.
16	14	12	10	8	6	4	2	32	30	28	26	24	22	20	18	

The above are the page values when the header address is assumed to be zero (0). So add 32 every time the above page address increases by 1.

When an odd byte is transferred, only the upper bit (bit 15 to bit 8) are specified.

2) In which address in the mainframe internal RAM area, is the transferred pattern data stored?

The RAM address of the mainframe is:

When DATA: 0 to 16383
When ALTERNATE: 0 to 8192

ALTERNATE has pattern A and pattern B, so the internal RAM area is divided in half. (A or B pattern is selected according to the switch status of the A / B display switching).

The following shows the relationship between the address and the actual numbers of pages.

Pattern output address	Internal RAM area	Number of display pages
0 →	→	1 to 32 pages
1 →	→	33 to 64 pages
2 →	→	65 to 96 pages
3 →	→	97 to 128 pages
4 →	→	129 to 160 pages
:		:
16382 →	→	524225 to 524256 pages
16383 →	→	524257 to 524288 pages

APPENDIX C TABLES OF INITIAL VALUES

Appendix tables C-1 to C-5 shows initial values of MP1764A at the time of factory shipment.

Table C-1 Table of INPUT section Initial Values

Function	Header part	Initial value	Device message details	
			Item No.	Page
● INPUT section				
Data input threshold voltage	DTH	−0.500V	1	P9-26
Eye margin measurement result (threshold)	THM	No measurement result	2	P9-27
Clock input phase	CPA	0ps	3	P9-28
Eye margin measurement result (phase)	PHM	No measurement result	4	P9-29
Data input termination voltage	DTM	GND	5	P9-30
Clock input termination voltage	CTM	GND	6	P9-31
Delay status	DLY	READY status	7	P9-32
Automatic phase threshold search	SRH	OFF	8	P9-33
Clock input polarity	CPL	CLOCK	9	P9-34
Eye margin measurement display switching	EME	Input threshold / input phase	10	P9-35
Eye margin measurement start	EST	Eye margin measurement stop	11	P9-36
Eye margin measurement (Error ratio selection)	EYT	1.0E-2	12	P9-37

Table C-2 Table of MEMORY Section Initial values

Function	Header part	Initial value	Device message details	
			Item No.	Page
● MEMORY section				
Memory FD mode	FMD	1440 K	13	P9-39
File contents search	FSH	No data	14	P9-40
File No. / directory mode switching	FIL	File No.	15	P9-42
Floppy data recall	RCL	–	16	P9-43
Floppy data delete	DEL	–	17	P9-44
Floppy data save	SAV	–	18	P9-45
Floppy data resave	RSV	–	19	P9-46
Memory mode switching	MEM	PATT	20	P9-47
Floppy access status	MAC	Non access	21	P9-49
FD error message	FDE	No error	22	P9-50
FD format	FDF	–	23	P9-51

Table C-3 Table of PATTERN Section Initial Values

Function	Header part	Initial value	Device message details Item No.	Device message details Page
● PATTERN section				
Pattern logic	LGC	Positive	24	P9-53
Measurement pattern	PTS	PRBS	25	P9-54
Number of ZERO SUBST and PRBS steps	PTN	PRBS : $2^{15}-1$ Zero : 2^{15}	26	P9-55
PRBS mark ratio	MRK	1 / 2	27	P9-56
Synchronous mode	SYM	Alternate : Normal Data : Normal Zero sunbst: Normal PRBS : None	28	P9-57
Display selection	DSP	Pattern	29	P9-58
Alternate pattern A / B switching	ALT	A	30	P9-59
Frame bit length	FLN	32 bits	31	P9-60
Data length	DLN	Alternate : 128 bits Data : 2bits	32	P9-61
ZERO SUBST length	ZLN	1 bit	33	P9-63
Number of pages. Pattern synchronous trigger position	PAG ADR	1 page	34	P9-64
Pattern bit	BIT	All bits 0	35	P9-65
Bit window pattern	CHM	All bits 0	36	P9-67
Bit window page	MSK	1 page	37	P9-69
Block window pattern	MGB	All bits 0	38	P9-70
Error analysis data *1	EAB	No measurement data	39	P9-72
Error analysis page *1	EAP	1 page	40	P9-74
Bit window ON / OFF	MSE	OFF	41	P9-75
Block window ON / OFF	MGE	OFF	42	P9-76
Error analysis trigger *1	EAT	OFF	43	P9-77
PATTERN SYNC POSITION	PSP		54	P9-91
PAGE / PATTERN SYNC POSITION switch	PPD		55	P9-92

Table C-4 Tableof MEASUREMENT Section Initial Values

Function	Header part	Initial value	Device message details	
			Item No.	Page
● MEASUREMENT section				
Measurement result display mode	DMS	ERROR RATIO	59	P9-97
Intermediate result display function	CUR	OFF	60	P9-98
Measurement mode	MOD	Repeat	61	P9-99
Measurement status	MSR	Measurement stop	64	P9-102
Automatic synchronous function	SYN	OFF	65	P9-103
Automatic synchronous threshold	SYE	10^{-2}	66	P9-104
Real time and measurement period display switching	TIM	Period	67	P9-105
Internal timer setting	RTM	Current time	68	P9-106
Measurement period setting	PRD	1 second	69	P9-107

Table C-5 Table of Other Section Initial Values

Function	Header part	Initial value	Device message details Item No.	Device message details Page
● Other section (Front panel)				
Printer ON / OFF	PRN	OFF	83	P9-127
Manual print	PSA	Stop	84	P9-128
Alarm monitor (Alarm detection)	ALM	OFF	85	P9-129
Alarm monitor (Error detection)	MON	OFF	86	P9-130
Synchronous signal output	SOP	1 / 32 CLOCK	87	P9-131
● Other section (Rear panel GPIB)				
GPIB 2 address	GPA	0	88	P9-132
● Other section (Rear panel FUNCTION switch)				
Number of shifts of mark ratio AND bit	SFT	1 bit shift	89	P9-133
Clock off processing	CLS	Exclude	90	P9-134
Synchronous loss processing	SLS	Exclude	91	P9-135
Error performance data threshold selection	ETH	1.0E-3	92	P9-136
BURST measurement mode	BST	OFF	93	P9-137
Intermediate data calculation function during measurement	CAL	Cumulative calculation data	94	P9-138
Error detection mode	ETY	Total errors	95	P9-139
EI, EFI interval period	EIT	1 msec	96	P9-140
Data print format	FMT	Standard format	97	P9-141
Threshold EI, EFI data print	THR	No print	98	P9-142
Error performance data print	EPF	No print	99	P9-143
Intermediate data print	ITM	No print	100	P9-144
1-second data print	OSC	No print	101	P9-145
1-second data print threshold	DOT	Error > 0	102	P9-146
Paper saving function	PSV	OFF	103	P9-147
Measurement interval period	ITV	100 msec	104	P9-148
Memory FD mode	FMD	1440 K / 720 K	13	P9-39

(Blank)

APPENDIX D   TABLE OF TRACKING ITEMS

In this Appendix, MP1764A tracking items are explained.
Tracking denotes a function to send the MP1764A setting conditions to MP1763A through GPIB.
See "Section 3 Bus Connections and address setting" for connections.
The trackings are differ according to the measurement patterns.

Table D-1 Table of Tracking Items

Measurement pattern	Tracking Items
Alternate pattern	1 : LOGIC 2 : Measurement pattern (Alternate) 3 : A / B display switching 4 : Page setting 5 : Pattern bit (DMA transfer, Both A and B transferred)
Data pattern	1 : LOGIC 2 : Measurement pattern (Data) 3 : Data length 4 : Page setting 5 : Pattern bit (DMA transfer)
Zero subst pattern	1 : LOGIC 2 : Measurement pattern 3 : Zero subst step 4 : Zero subst length 5 : Page setting
PRBS pattern	1 : LOGIC 2 : Measurement pattern 3 : PRBS step 4 : PRBS mark ratio 5 : Page setting

(Blank)