

MP1763B
Pulse Pattern Generator
Operation Manual
(GPIB Programming)

Fourth Edition

To ensure that the MP1763B Pulse Pattern Generator is used safely, read the safety information in the MP1763B Pulse Pattern Generator Operation Manual first. Keep this manual with the Pulse Pattern Generator.

Measurement Solutions
ANRITSU CORPORATION

MP1763B Pulse Pattern Generator
Operation Manual
(GPIB Programming)

August 1995 (First Edition)
April 2002 (Fourth Edition)

Copyright © 1995-2002, ANRITSU CORPORATION.
All rights reserved. No part of this manual may be reproduced without the prior written permission of the publisher.
The contents of this manual may be changed without prior notice.
Printed in Japan

'HP Basic' is a registered trademark of the Hewlett-Packard Corporation.

'HP' is a registered trademark of the Hewlett-Packard Company.

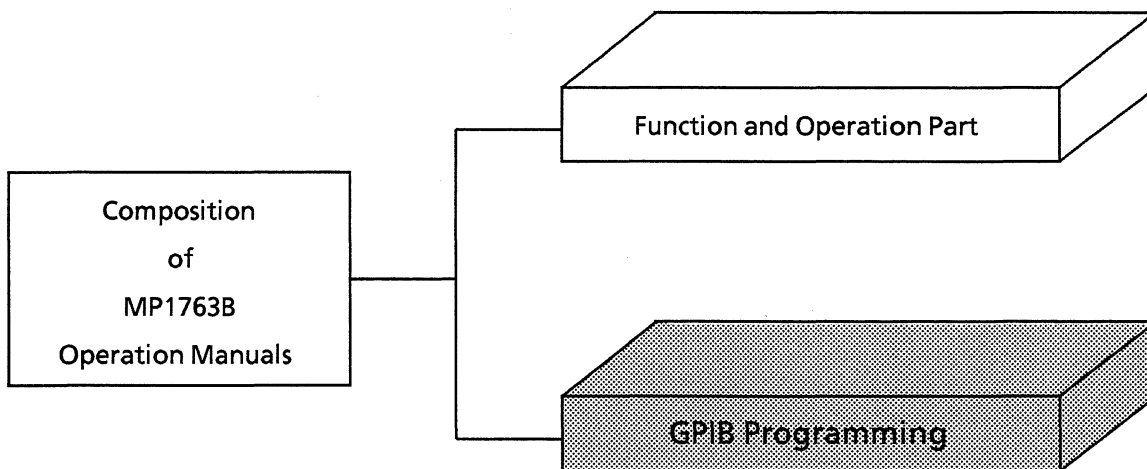
'MS-DOS' is a registered trademark of the Microsoft Corporation.

'Quick Basic' is a registered trademark of the Microsoft Corporation.

(Blank)

Composition of MP1763B Operation Manuals

The MP1763B Pulse Pattern Generator operation manuals are composed of the following two documents. Use them properly according to the usage purpose.



Function and Operation Part

These outline the MP1763B, and describes the preparations before use, the panels, specifications, performances, functions, and operation procedures.

GPIB Programming :

The MP1763B 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	1-1
	1.1	Development of the GPIB Standard	1-3
	1.2	MP1763B GPIB Functions	1-4
	1.2.1	Overviews of GPIB functions	1-4
	1.2.2	Examples of system makeup using GPIB	1-4
SECTION	2	SPECIFICATIONS	2-1
	2.1	Interface Functions	2-3
	2.2	Device Message List	2-4
	2.2.1	IEEE 488.2 common commands and MP1763B supported commands	2-5
	2.2.2	Status messages	2-7
	2.2.3	MP1763B device messages	2-9
SECTION	3	CONNECTING THE BUS AND SETTINGS ADDRESS	3-1
	3.1	Connecting Devices with GPIB Cables	3-3
	3.2	Procedure for Setting the Address and Checking it ..	3-4
	3.2.1	Address setting	3-5
	3.2.2	Connection with MP1764A during the tracking operation	3-6
SECTION	4	INITIAL SETTINGS	4-1
	4.1	Bus Initialization by the IFC Statement	4-4
	4.2	Initialization for Message Exchange by DCL and SDC Bus Commands	4-6
	4.3	Device Initialization by the *RST Command	4-8
	4.4	Device Initialization by the INI Command	4-10
	4.5	Device Status at Power-on	4-11
SECTION	5	LISTENER INPUT FORMAT	5-1
	5.1	Listener Input Program Message Syntax Notation ...	5-4
	5.1.1	Separators, terminators and spaces before headers	5-4

5.1.2	General format for program command messages	5-6
5.1.3	General format for query messages	5-7
5.2	Functional Elements of Program Messages	5-8
5.2.1	<TERMINATED PROGRAM MESSAGE>	5-8
5.2.2	<PROGRAM MESSAGE TERMINATOR>	5-9
5.2.3	<white space>	5-10
5.2.4	<PROGRAM MESSAGE>	5-10
5.2.5	<PROGRAM MESSAGE UNIT SEPARATOR>	5-11
5.2.6	<PROGRAM MESSAGE UNIT>	5-11
5.2.7	<COMMAND MESSAGE UNIT> and <QUERY MESSAGE UNIT>	5-12
5.2.8	<COMMAND PROGRAM HEADER>	5-13
5.2.9	<QUERY PROGRAM HEADER>	5-15
5.2.10	<PROGRAM HEADER SEPARATOR>	5-16
5.2.11	<PROGRAM DATA SEPARATOR>	5-16
5.3	Program Data Format	5-17
5.3.1	<DECIMAL NUMERIC PROGRAM DATA>	5-18
5.3.2	<NON-DECIMAL NUMERIC PROGRAM DATA> ...	5-20
SECTION	6 TALKER OUTPUT FORMAT	6-1
6.1	Syntax Differences Between Formats of Listener Input and Talker Output	6-4
6.2	Functional Elements of Response Message	6-5
6.2.1	<TERMINATED RESPONSE MESSAGE>	6-5
6.2.2	<RESPONSE MESSAGE TERMINATOR>	6-6
6.2.3	<RESPONSE MESSAGE>	6-7
6.2.4	<RESPONSE MESSAGE UNIT SEPARATOR>	6-8
6.2.5	<RESPONSE MESSAGE UNIT>	6-8
6.2.6	<RESPONSE HEADER SEPARATOR>	6-9
6.2.7	<RESPONSE DATA SEPARATOR>	6-9
6.2.8	<RESPONSE HEADER>	6-9
6.2.9	<RESPONSE DATA>	6-11
SECTION	7 COMMON COMMANDS	7-1
7.1	Classification by Function of Common Commands Supported by the MP1763B	7-3
7.2	The Classification of Commands Supported and the Reference	7-4

SECTION	8	STATUS STRUCTURE	8-1
	8.1	IEEE 488.2 Standard Status Model	8-4
	8.2	Status Byte (STB) Register	8-6
	8.2.1	ESB and MAV summary messages	8-6
	8.2.2	Device-dependent summary messages	8-7
	8.2.3	Reading and clearing the STB register	8-8
	8.3	Enabling SRQ	8-10
	8.4	Standard Event Status Register	8-11
	8.4.1	Bit definition	8-11
	8.4.2	Query error details	8-13
	8.4.3	Reading, writing to and clearing the standard event status register	8-14
	8.4.4	Reading, writing to and clearing the standard event status enable register	8-14
	8.5	Extended Event Status Register	8-15
	8.5.1	Bit definition of END event status register	8-16
	8.5.2	Bit definition of ERROR event status register	8-18
	8.5.3	Reading, writing to and clearing the extended event status register	8-20
	8.5.4	Reading, writing to and clearing the extended event status enable register	8-20
	8.6	Queue Model	8-21
	8.7	Techniques for Synchronizing Devices with the Controller	8-23
	8.7.1	Enforcing the sequential execution	8-23
	8.7.2	Wait for a response from the output queue	8-24
	8.7.3	Wait for a service request	8-25
SECTION	9	DETAILS OF DEVICE MESSAGES	9-1
	9.1	Table of Device Messages	9-3
	9.1.1	Table of Device Messages (in the Alphabetic order)	9-3
	9.1.2	Device Messages (Panel correspondence)	9-7
	9.1.3	Detailed Explanation of Device Messages	9-18
SECTION	10	EXAMPLE OF PROGRAMMING	10-1
	10.1	Example of Program creation Using HP9000	10-6

APPENDIX	A	COMPATIBILITY WITH CONVENTIONAL INSTRUMENTS	A-1
	B	PATTERN DMA TRANSFER	B-1
	C	TABLES OF INITIAL VALUES	C-1
	D	TABLE OF TRACKING ITEMS	D-1

SECTION 1

GENERAL

This section outlines the historical development of the GPIB standard and gives a general description of GPIB functions of the MP1763B Pulse Pattern Generator.

TABLE OF CONTENTS

1.1	Development of the GPIB Standard	1-3
1.2	MP1763B GPIB Functions	1-4
1.2.1	Overviews of GPIB functions	1-4
1.2.2	Examples of system makeup using GPIB	1-4

(Blank)

1.1 Development of the GPIB Standard

The MP1763B, when combined with an external controller in a system bus offers automatic measurements. For this purpose it is provided with a GPIB interface bus (IEEE std. 488.2-1987) as a standard feature. The GPIB (General Purpose Interface 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 data 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 identical 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 MP1763B GPIB Functions

The MP1763B has the following GPIB functions.

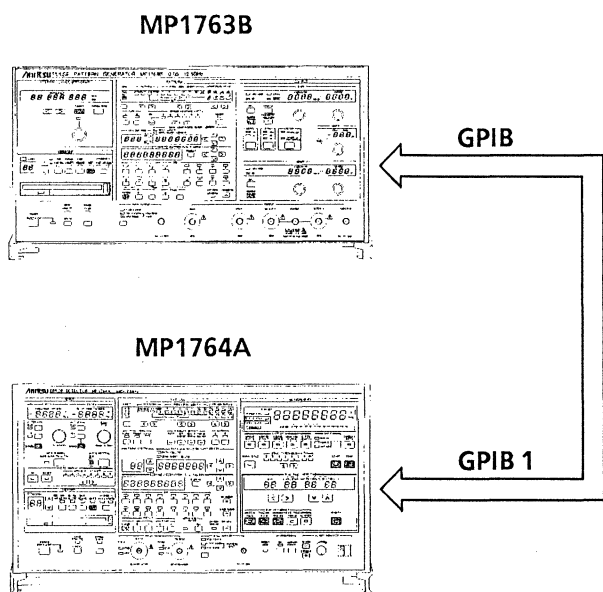
- (1) Apart from the power switch and some LOCAL keys, all functions can be controlled.
- (2) Readout of all setting conditions
- (3) Interrupt function and serial poll operation
- (4) Automatic measuring systems can be constructed by combining the MP1764A with a personal computer and other measuring instruments.

1.2.1 Overviews of GPIB functions

GPIB 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 MP1764A Error Detector by the system controller's settings in tracking operating.

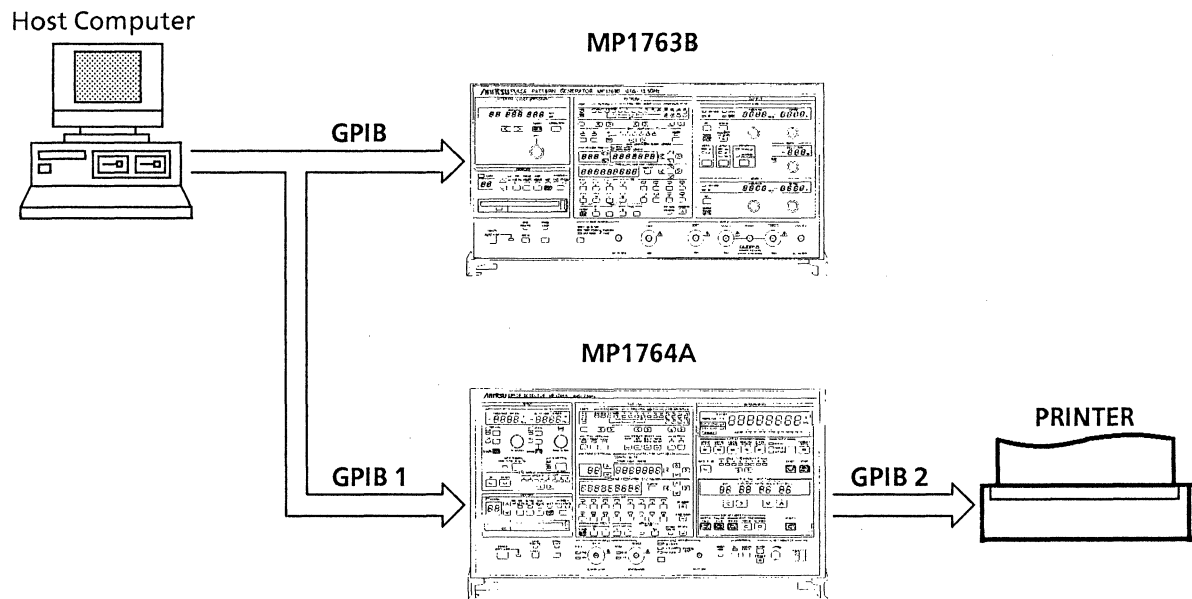
1.2.2 Examples of system makeup using GPIB

(1) Stand-alone type 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 MP1763B or MP1764A can be a master (controller).

(2) Control by the host computer



By means of controlling MP1763B and MP1764A using the host computer via GPIB 1 port, data can be output to the printer via MP1764A GPIB 2 port.

(Blank)

SECTION 2 SPECIFICATIONS

In this section, interface functions of the MP1763B GPIB specifications are explained. For the device message, see SECTION 9.

TABLE OF CONTENTS

2.1	Interface Functions	2-3
2.2	Device Message List	2-4
2.2.1	IEEE 488.2 common commands and MP1763B supported commands	2-5
2.2.2	Status messages	2-7
2.2.3	MP1763B device messages	2-9

(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 instruments used in a GPIB system. The MP1763B GPIB provide the subsets listed in the code columns of the tables below.

GPIB 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. Synchronizes 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.	Devices 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

2.2 Device Message List

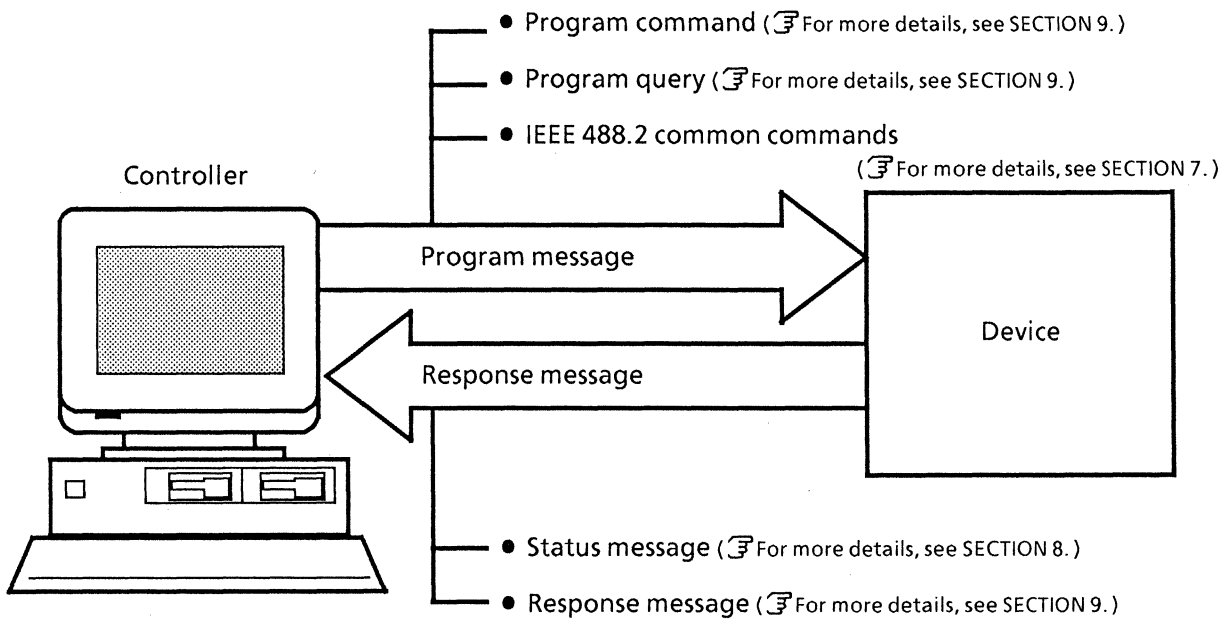
Device messages are message that are transmitted between the controller and the device via the system interface in the bus mode, i.e. when the ATN line is false. There are two types: program messages and response messages.

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 MP1763B and IEEE 488.2 common commands. The latter are common commands used for, in addition to the MP1763B, any measuring instrument conforming to the IEEE 488.2 standard.

Program queries are commands used to elicit 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 MP1763B 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 MP1763B output queue is 256 bytes.


2.2.1 IEEE 488.2 common commands and MP1763A 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 MP1763B are indicated with © symbol in the table.

Mnemonic	Command name	IEEE488.2 Standard	MP1763B 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	©

SECTION 2 SPECIFICATIONS

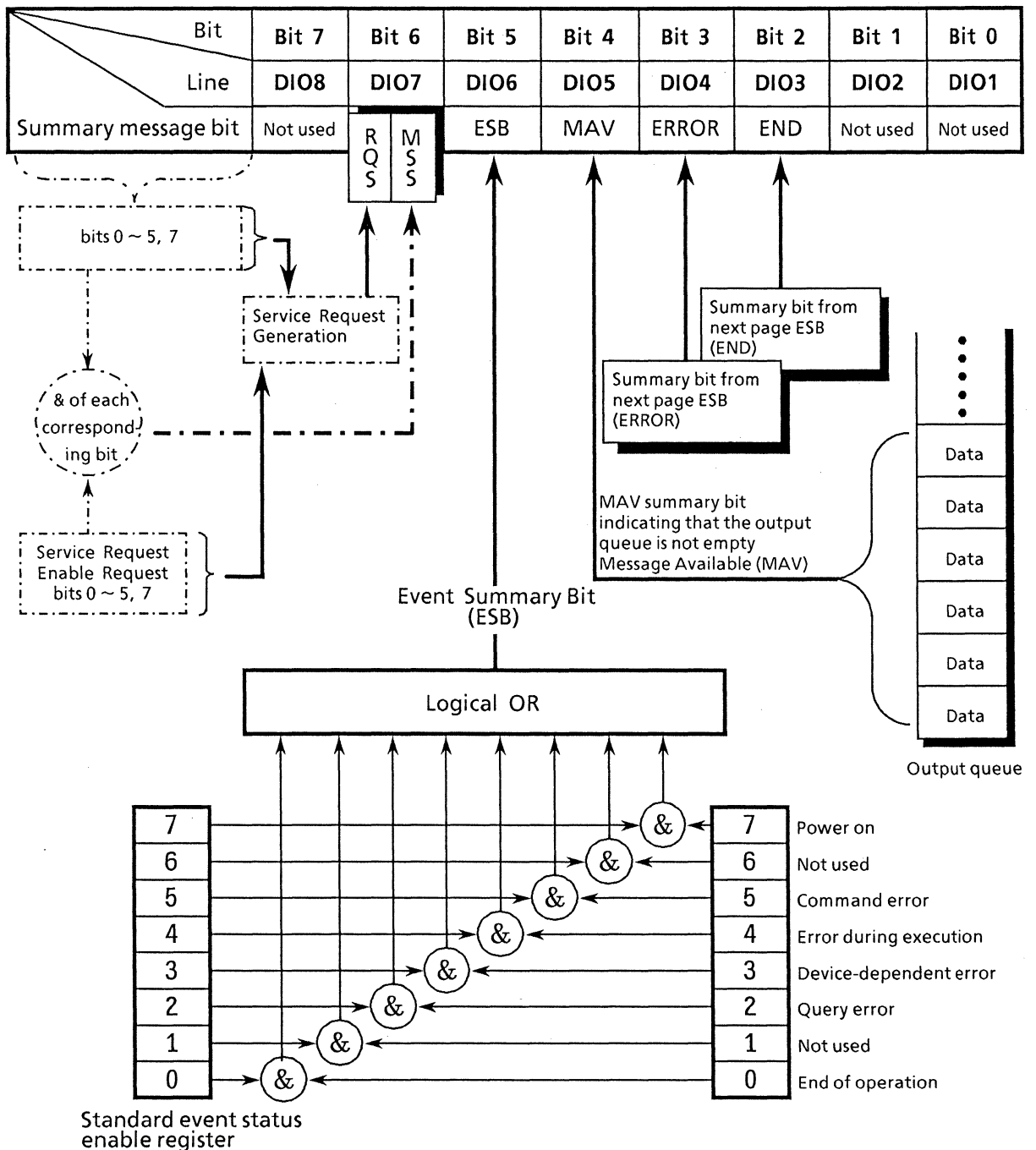
Mnemonic	Command name	IEEE488.2 Standard	MP1763B supported commands
*STB?	Read Status Byte Query	Mandatory	☉
*TRG	Trigger Command	Mandatory if DT1	☉
*TST?	Self Test Query	Mandatory	☉
*WAI	Wait to Continue Command	Mandatory	☉

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

2.2.2 Status messages

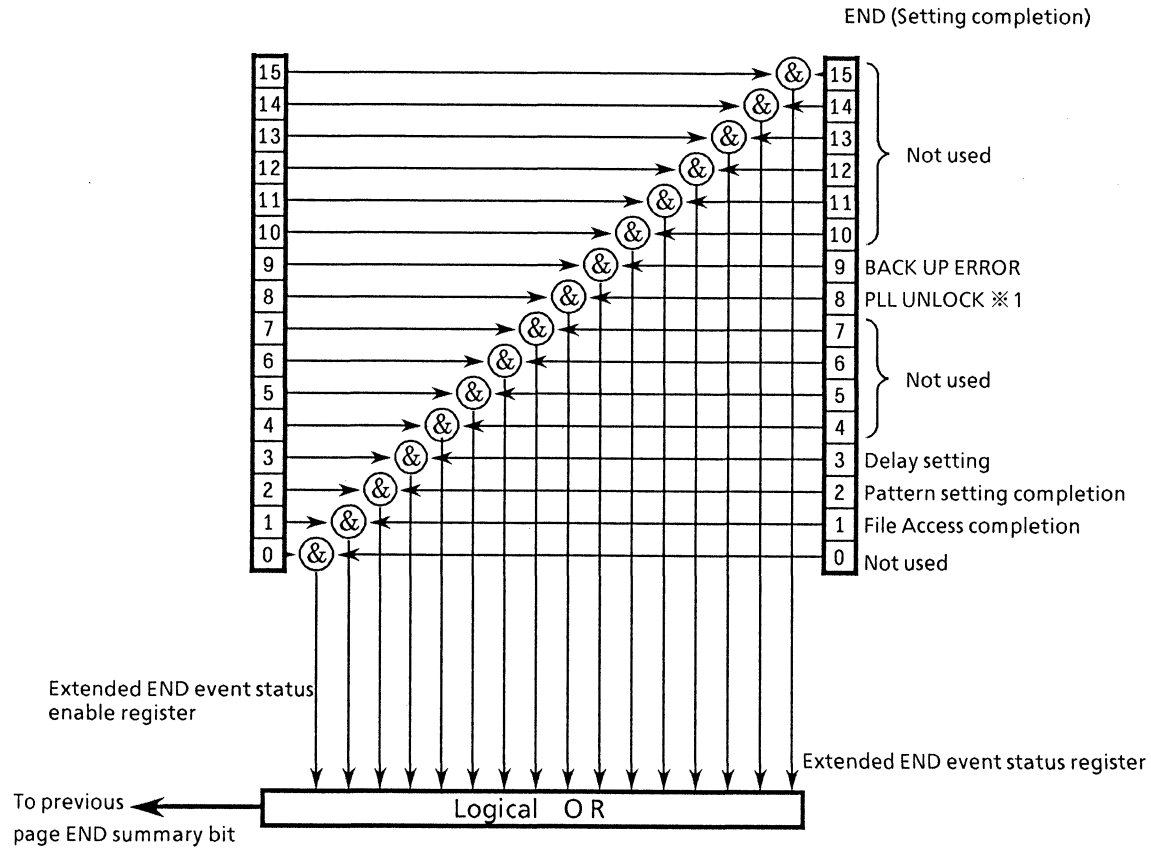
The diagram below shows the structure of service-request summary messages for the status byte register used with the MP1763B.

Status Byte Register Summary Bit Composition

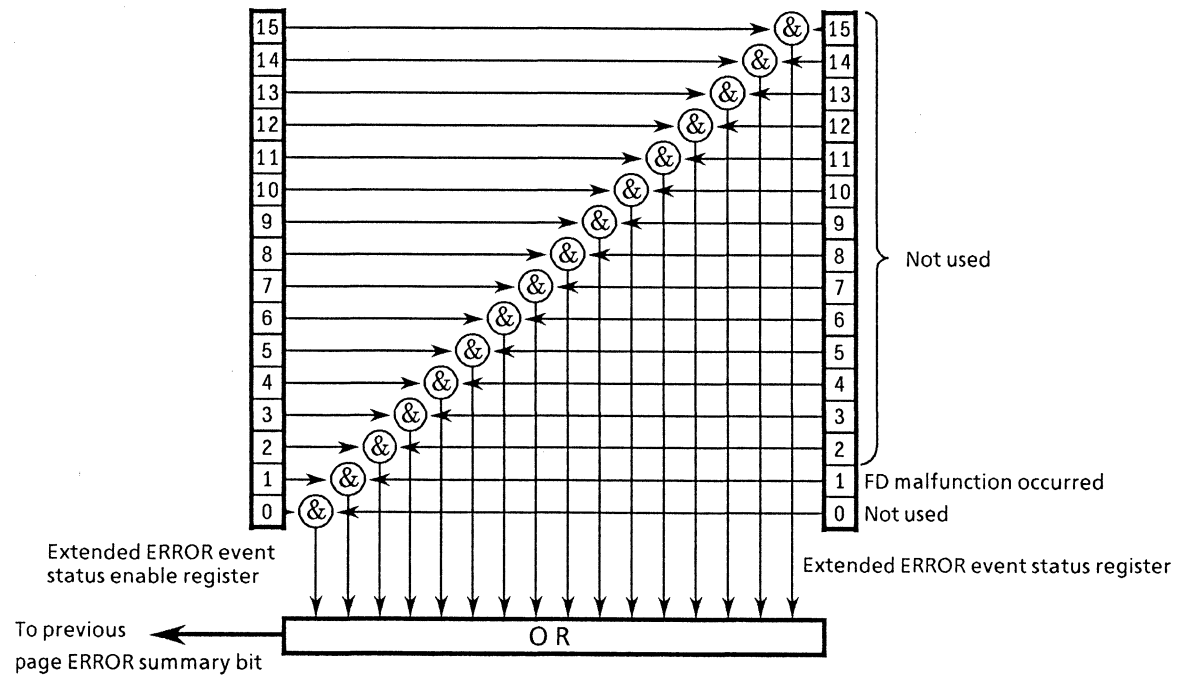


Note : (&) means logical AND operation.

SECTION 2 SPECIFICATIONS



*1) PLL UNLOCK is allowed only when OPTION-01 is installed.



2.2.3 MP1763B device messages

The device messages consist of fixed program commands of the MP1763B queries and response messages. The device messages list and description are shown in Section 9.

SECTION 2 SPECIFICATIONS

(Blank)

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

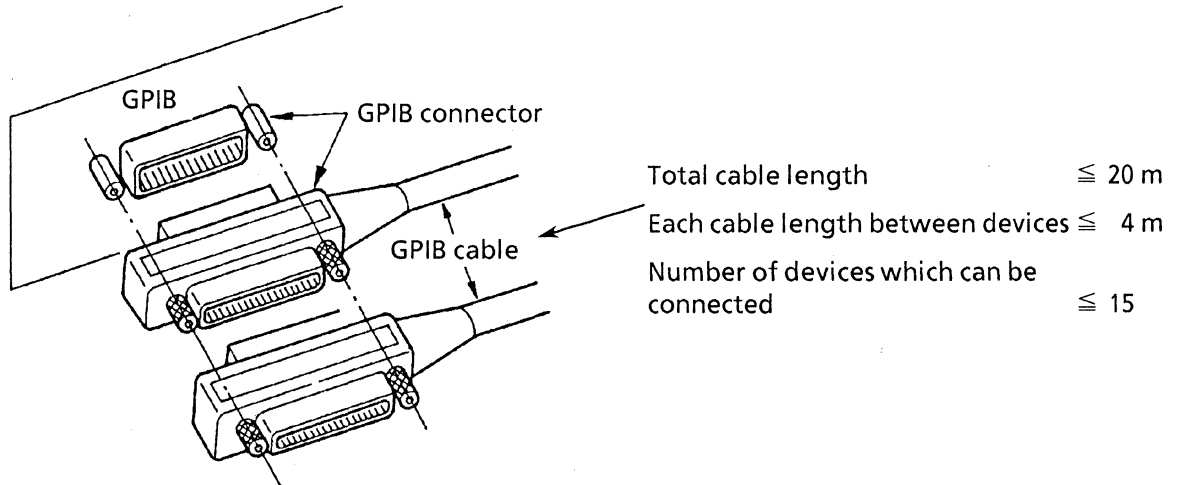
3.1	Connecting Devices with GPIB Cables	3-3
3.2	Procedure for Setting the Address and Checking it	3-4
3.2.1	Address setting	3-5
3.2.2	Connection with MP1764A during the tracking operation	3-6

(Blank)

3.1 Connecting Devices with GPIB Cables

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

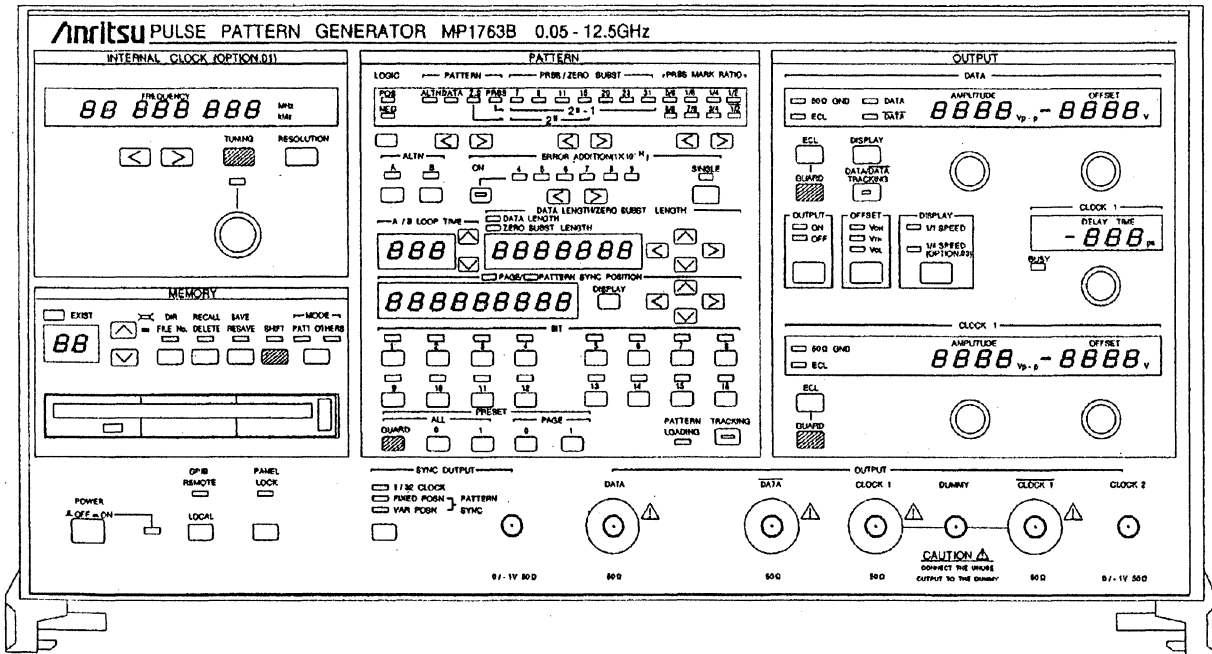
A maximum of 15 devices, including the controller, can be connected to one system. The restrictions indicated at the right of the diagram below should be observed when connecting many devices to one system.



3.2 Procedures for Setting the Address and Checking it

Set the GPIB address for the MP1763B after or before turning on the power. The GPIB address 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 MP1763B 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.

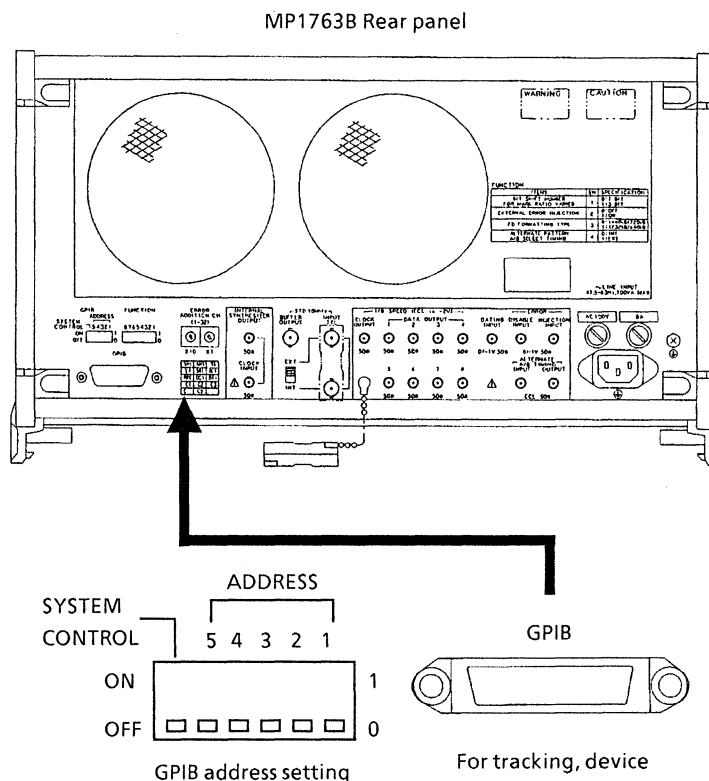
- Note :**
- 1) The system always checks the GPIB "ADDRESS" switch settings when the power is turned on and determines its own address. So, changing the address is always allowed unless the system is in remote state.
 - 2) To control the system as a device from an external controller, set "SYSTEM CONTROL" of the GPIB address switch to OFF(0).



MP1763B Front panel

3.2.1 Address setting

The GPIB address for GPIB port of MP1763B are set with the DIP switch on the rear panel.

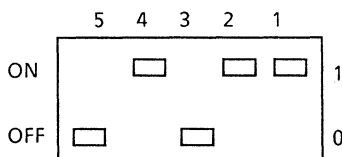


The GPIB address can be set 0 to 30. Five switches are weighed differently: “5”, “4”, “3”, “2” and “1” are respectively weighed to 16, 8, 4, 2 and 1.

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.



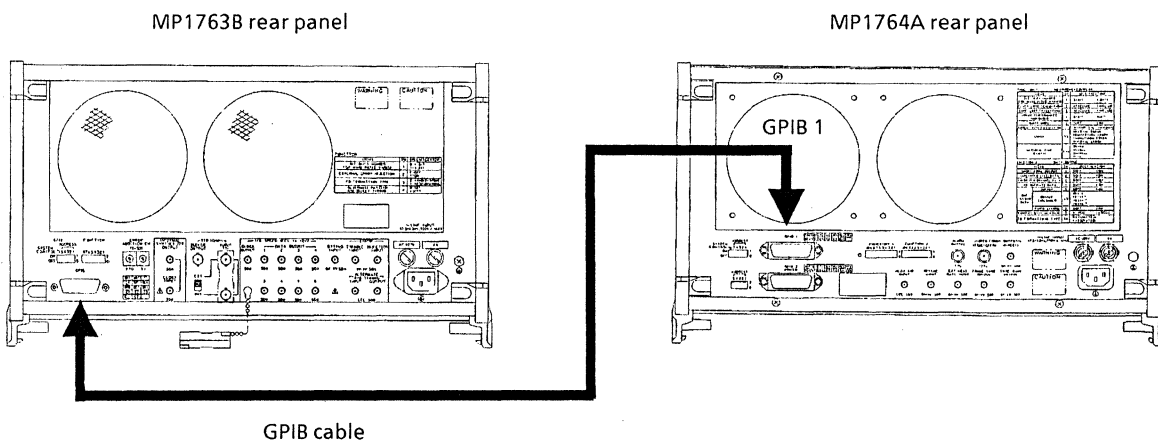
However, address 31, where all the switches are set to ON, is assumed to be address 0.

3.2.2 Connection with MP1764A during the tracking operation

Tracking operation is a function that pattern settings are made to be synchronized with each other between MP1763B and MP1764A. Either MP1763B 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 MP1763B is a Master and controls MP1764A:

When the settings for MP1763B are set to MP1764A via GPIB, the setting and connection are as follows:

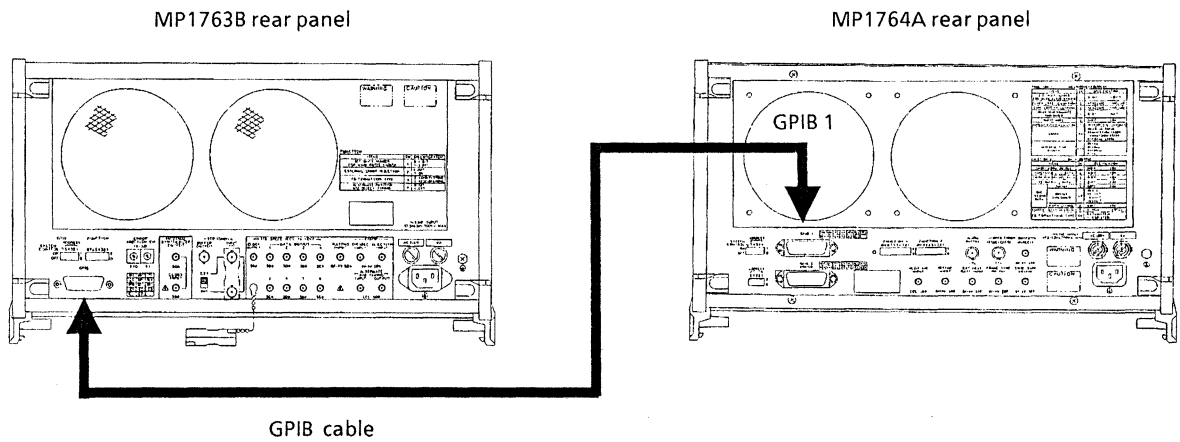


- a) Like the diagram above, the GPIB connector on the MP1763B's rear panel is connected with the GPIB 1 connector on the MP1764A via the GPIB cable (included).
- b) Set "SYSTEM CONTROL" of the GPIB address switch on the MP1763B' rear panel to ON (1).
- c) Set the value of the GPIB 1 address switch on the MP1764A's rear panel to that of MP1763B's GPIB address + 2.
- d) Turn on the MP1763B power again.
- e) Set the TRACKING key on the MP1763B'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 MP1763B via GPIB, the setting and connection are as follows:



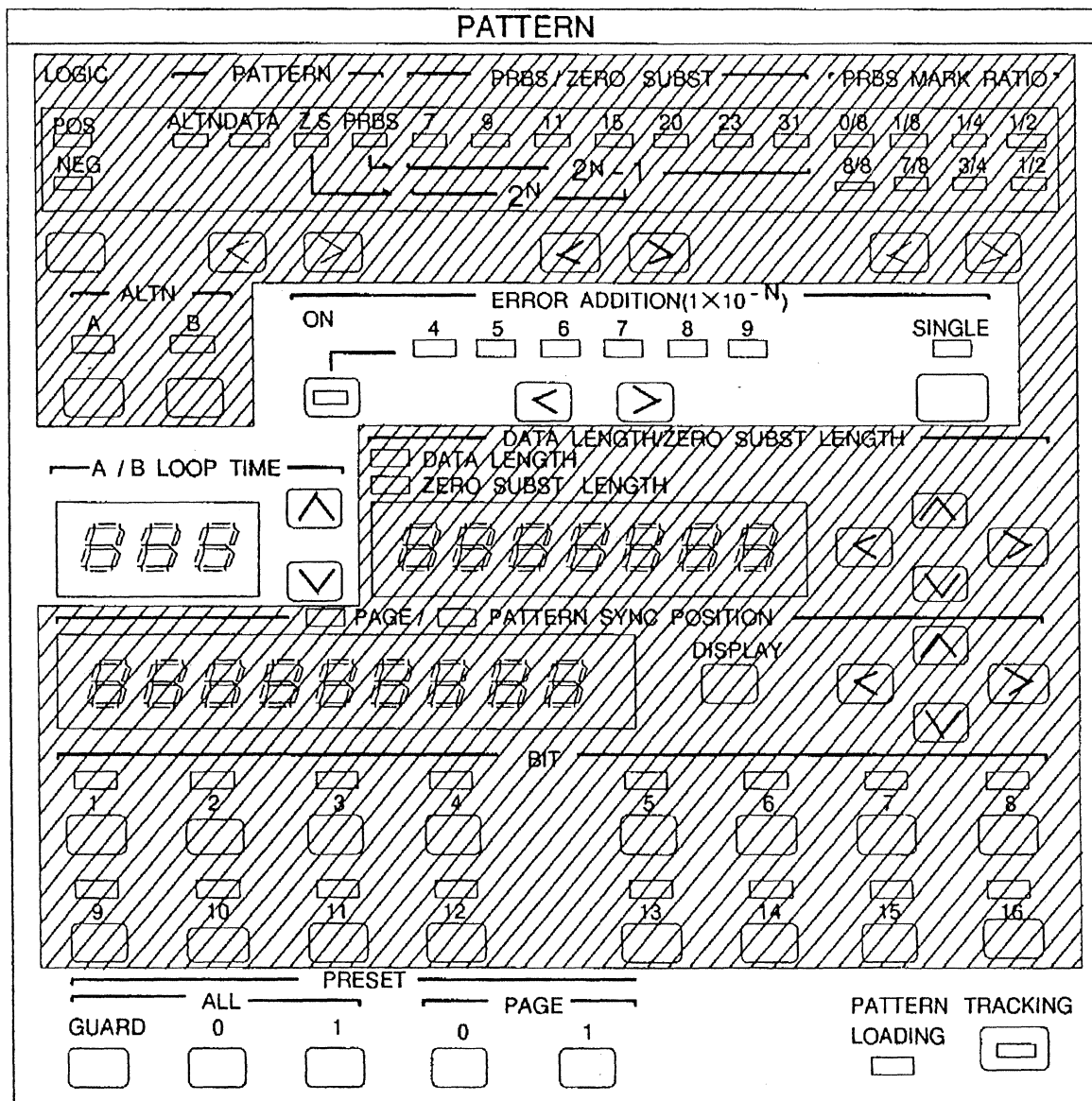
- a) Like the diagram above, the GPIB 1 connector on the MP1764's rear panel is connected with the GPIB connector on the MP1763B via the GPIB cable (included).
- b) Set "SYSTEM CONTROL" of the GPIB 1 address switch on the MP1764A's rear panel to ON (1).
- c) Set the value of the GPIB address switch on the MP1763B'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 MP1763B and MP1764A

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

Pattern-setting area on the MP1763B'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 MP1763B does not coincide with that for MP1764A (such as error analysis data on the MP1764A Error Detector) cannot be pattern-tracked.

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

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

4.1	Bus Initialization by the IFC Statement	4-4
4.2	Initialization for Message Exchange by DCL and SDC Bus Commands	4-6
4.3	Device Initialization by the *RST Command	4-8
4.4	Device Initialization by the INI Command	4-10
4.5	Device Status at Power-on	4-11

(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 initialization. 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 μ s (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 device 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 $100\mu\text{s}$.

② Controller

The controller is put in the idle state (CIDS – Controller Idle State) within $100\mu\text{s}$ 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 @103 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 MP1763B)
- 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 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 @103:"*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 MP1763B for the control of external devices, etc. (Refer to SECTION 8 for items ③, ④ and ⑤. The MP1763B does not support items ⑧ to ⑩.)

The table below shows the initial settings proper to the MP1763B for the functions and status.

Initial Settings

Group	Initial Settings	Notes
Setting States	Initialized to the status at delivery	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 @103:"INI" Initializing only the device assigned address 3 with level 3.

■ Description

The **INI** command is one of the device messages proper to MP1763B; this command is sent as a program message to the device from the controller to reset the device with level 3.

■ 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

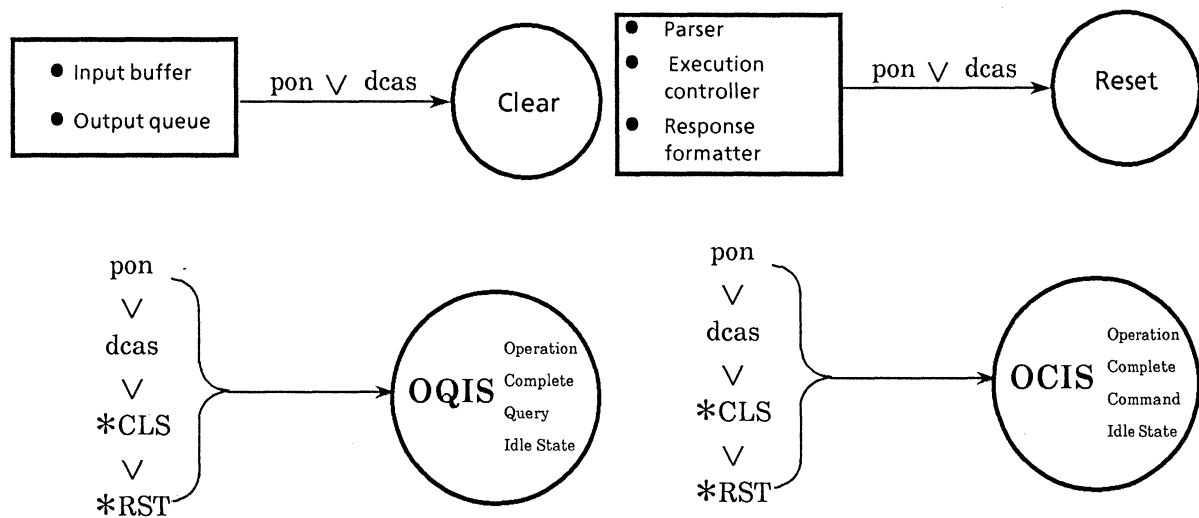
The device's items to be initialized are setting status, clock and date.

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 MP1763B 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 MP1763B 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 MP1763B 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 MP1763B)
- *PSC?
- *PUD? (Not supported by the MP1763B)
- *RDT? (Not supported by the MP1763B)

SECTION 4 OPERATION

■ 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 MP1763B)
- ⑤ Marco-definition defined by *DMC command (not supported by the MP1763B)
- ⑥ Macros enabled by the *EMC command (not supported by the MP1763B)
- ⑦ Addresses received by the *PCB command (not supported by the MP1763B)

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

5.1	Listener Input Program Message Syntax Notation	5-4
5.1.1	Separators, terminators and spaces before headers	5-4
5.1.2	General format for program command messages	5-6
5.1.3	General format for query messages	5-7
5.2	Functional Elements of Program Messages	5-8
5.2.1	<TERMINATED PROGRAM MESSAGE>	5-8
5.2.2	<PROGRAM MESSAGE TERMINATOR>	5-9
5.2.3	<white space>	5-10
5.2.4	<PROGRAM MESSAGE>	5-10
5.2.5	<PROGRAM MESSAGE UNIT SEPARATOR>	5-11
5.2.6	<PROGRAM MESSAGE UNIT>	5-11
5.2.7	<COMMAND MESSAGE UNIT> and <QUERY MESSAGE UNIT>	5-12
5.2.8	<COMMAND PROGRAM HEADER>	5-13
5.2.9	<QUERY PROGRAM HEADER>	5-15
5.2.10	<PROGRAM HEADER SEPARATOR>	5-16
5.2.11	<PROGRAM DATA SEPARATOR>	5-16
5.3	Program Data Format	5-17
5.3.1	<DECIMAL NUMERIC PROGRAM DATA>	5-18
5.3.2	<NON-DECIMAL NUMERIC PROGRAM DATA>	5-20

(Blank)

5.1 Listener Input Program Message Syntax Notation

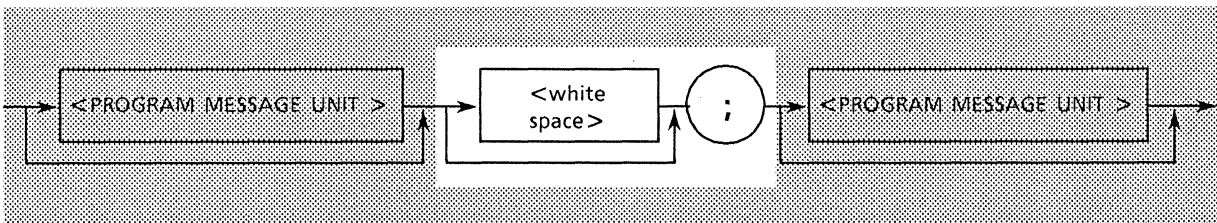
The following explains program message functional element and program data formats (Compound and common commands have been omitted)

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



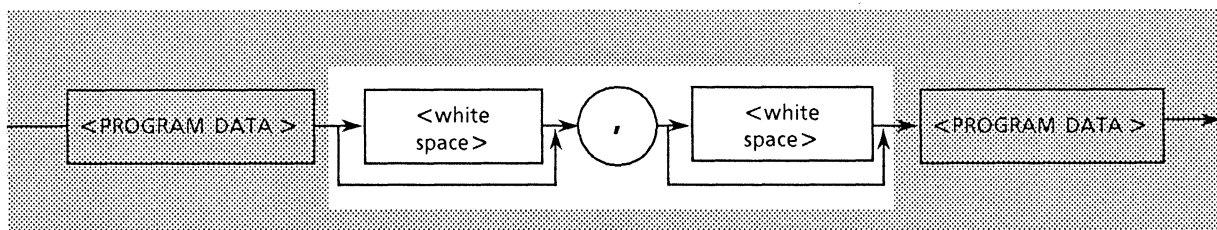
Example 2: 1 space + semicolon

DTM $\emptyset \triangle$; CTM \emptyset

(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, \emptyset

Example 3: Comma + 1 space

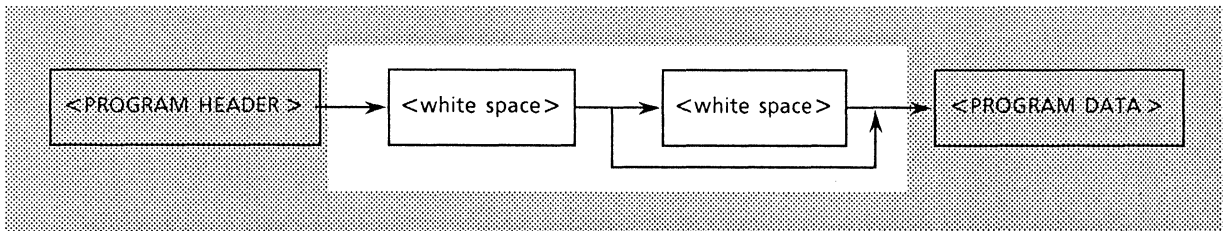
WRT 1, $\triangle \emptyset$

(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



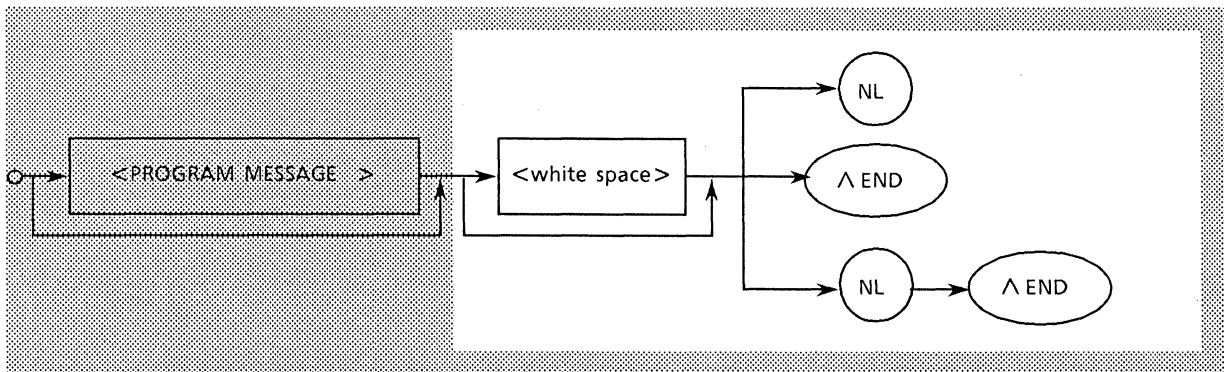
Example 2: 1 space

DTM \triangle \emptyset

(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

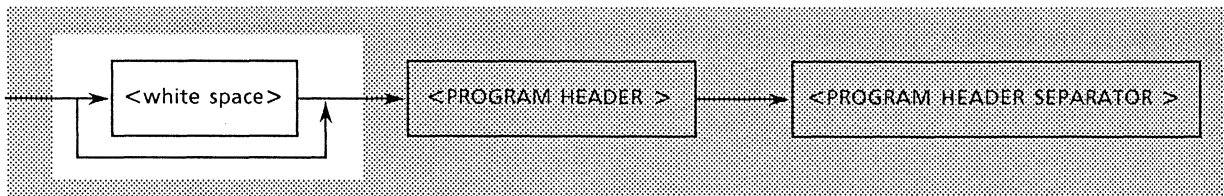
General format:



(5) Spaces before headers

An optional space may be placed before a program header.

General format:

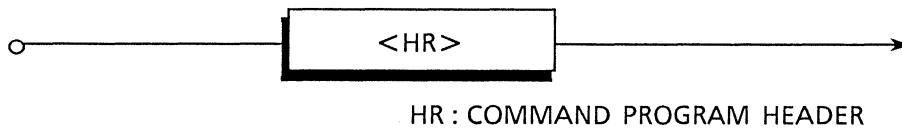


Example: 1 space is placed before the second program header SPF.

DTM \emptyset ; \triangle CTM \emptyset

5.1.2 General format for program command messages

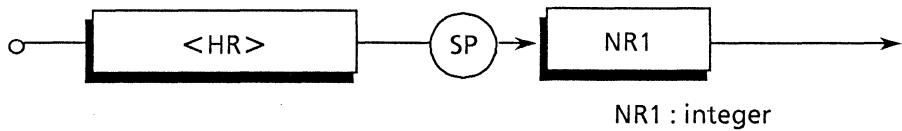
(1) Messages not accompanied by data



Examples:

INI Initializes setting

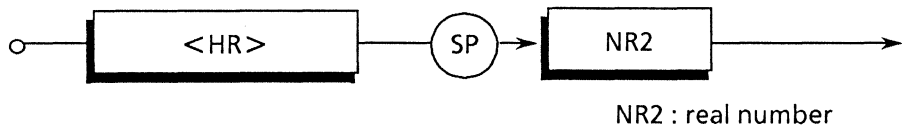
(2) Messages accompanied by integer data



Example:

PTS Δ 0 Sets ALTERNATE pattern
 PTS Δ 1 Sets DATA pattern
 PTS Δ 2 Sets ZERO SUBSTITUTION pattern
 PTS Δ 3 Sets PRBS pattern

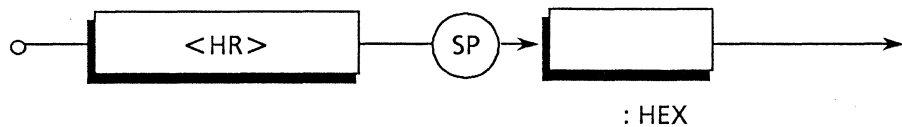
(3) Messages accompanied by real numbers



Example:

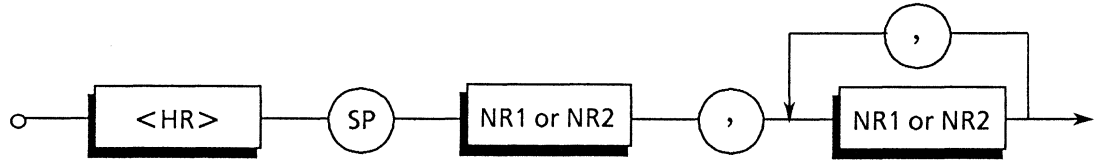
CAP Δ 2.000 Sets the clock output amplitude.

(4) Messages accompanied by HEX (hexadecimal)



Example:

BIT Δ #H FFFF

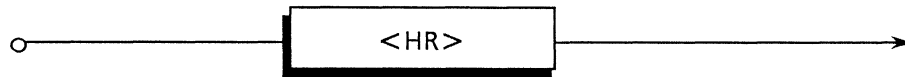
(5) Messages accompanied by multiple program data items

Example:

RTM△99,10,10,14,30,30 Sets internal timer to 14:30:30 10 October 1999.

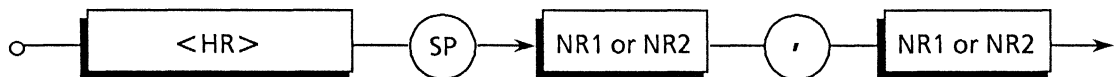
5.1.3 General format for query messages

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 output termination voltage data

(2) Messages accompanied by query data

Example:

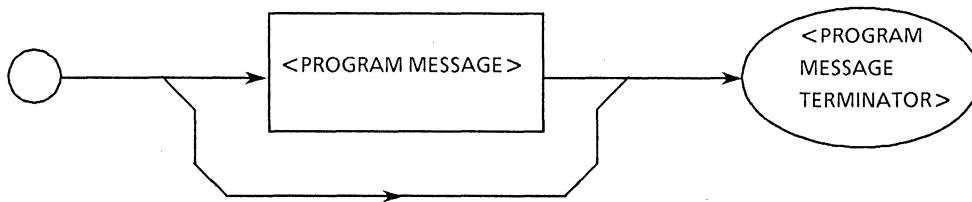
FSH? 1 Requests file information whose file No. is from 51 in the files saving measurement conditions in a floppy disk.

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.

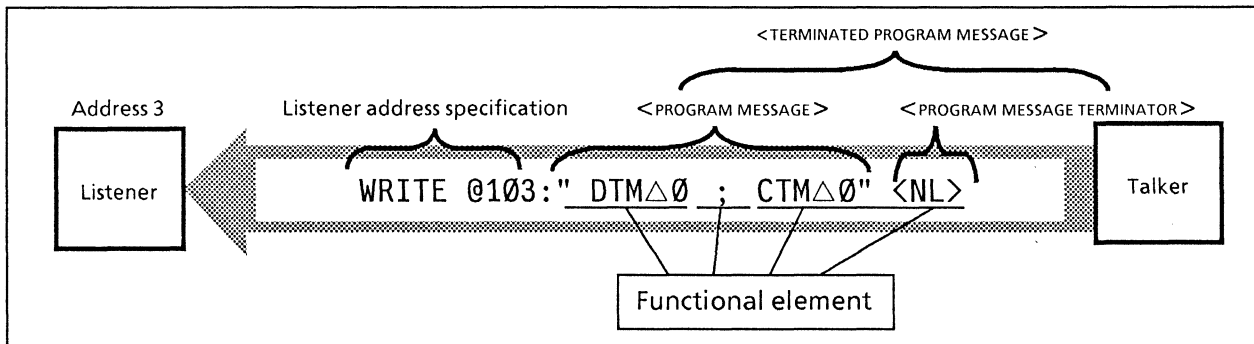
5.2.1 <TERMINATED PROGRAM MESSAGE>

A <TERMINATED PROGRAM MESSAGE> is defined as follows.



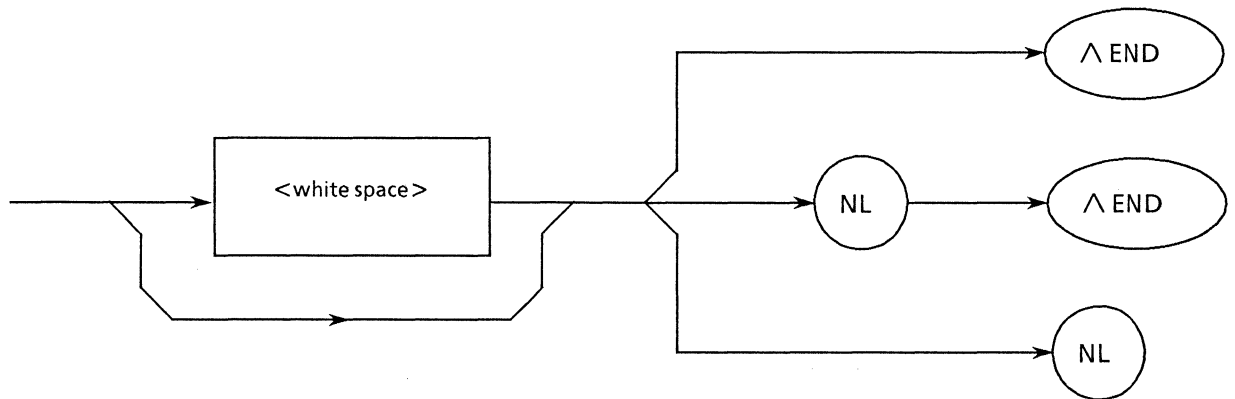
A <TERMINATED PROGRAM MESSAGE> is a data message which has all the functional elements required for transmission from the controller to a listener device. A <PROGRAM MESSAGE TERMINATOR> is attached to the end of a <PROGRAM MESSAGE> to terminate its transmission.

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



5.2.2 <PROGRAM MESSAGE TERMINATOR>

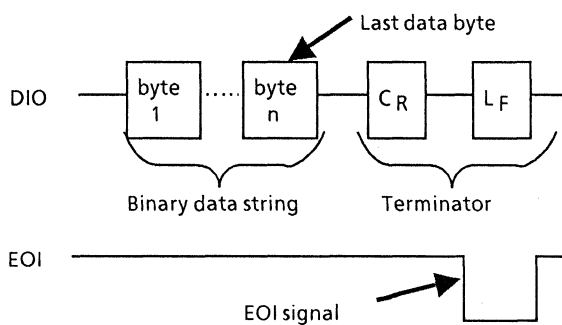
A <PROGRAM MESSAGE TERMINATOR> is defined as follows



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

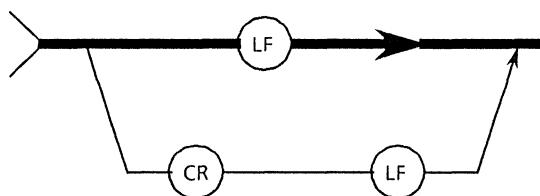
NL: NL is defined as a single ASCII code byte (decimal 10), i.e. the ASCII control code LF (Line Feed) used to return the carriage and bring the print position to the beginning of the next line. It is also called NL (New Line). When a <PROGRAM MESSAGE> is sent by a **WRITE@** statement, there is no need 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.

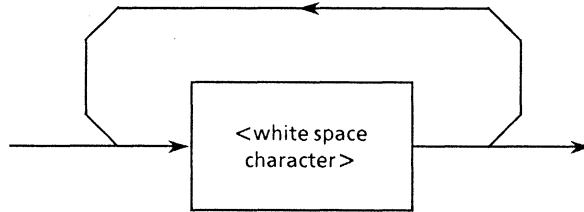


Note :

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.

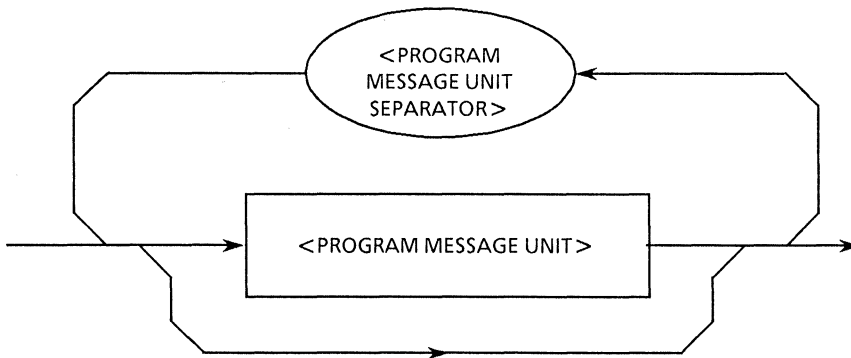


A <white space character> is defined as a single ASCII code byte in the range 00 to 09, 0B to 20 (decimal 0 to 9, 11 to 32).

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>.

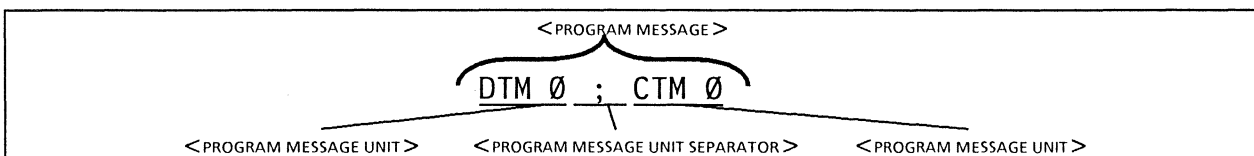
Example 1:

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

DTM 0

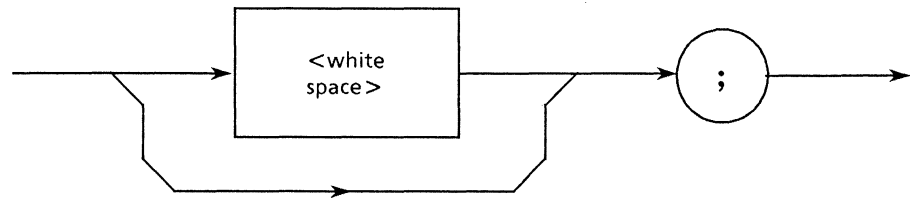
Example 2:

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

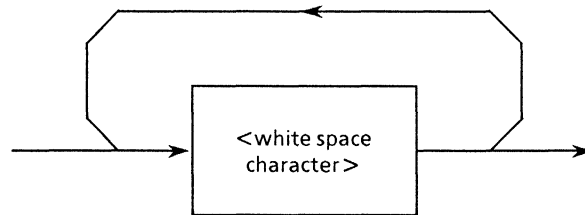


5.2.5 <PROGRAM MESSAGE UNIT SEPARATOR>

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



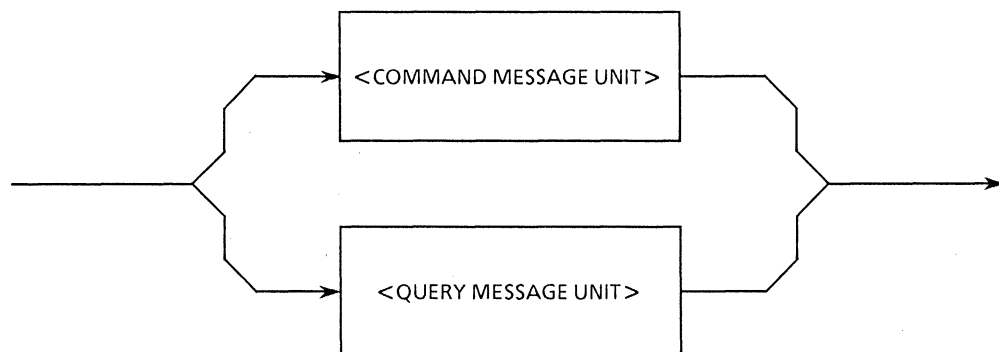
<white space> is defined as follows.



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>

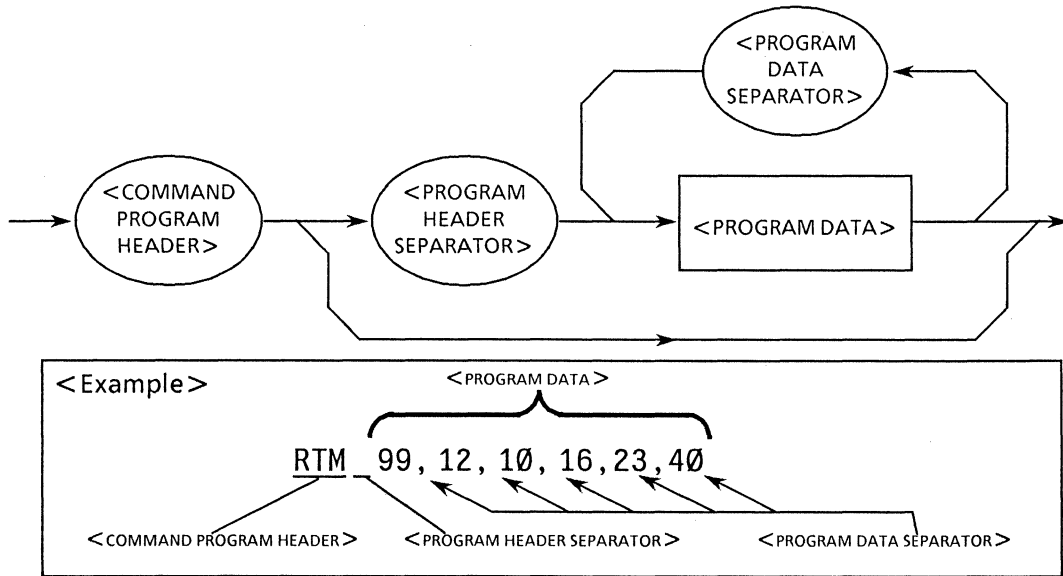
A <PROGRAM MESSAGE UNIT> is defined as follows.



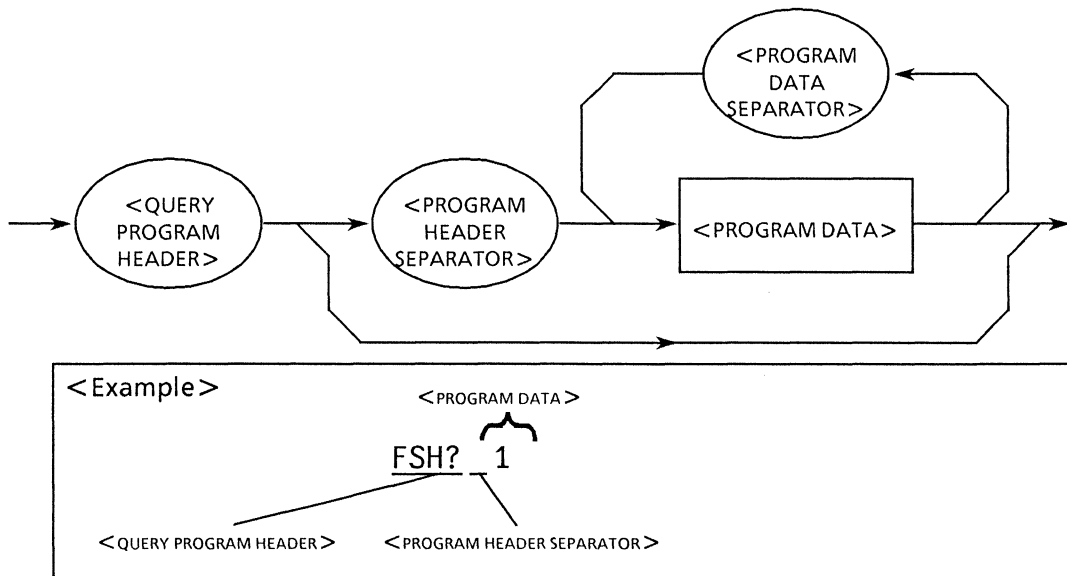
A <PROGRAM MESSAGE UNIT> is either the <COMMAND MESSAGE UNIT> or <QUERY MESSAGE UNIT> received by a device. <COMMAND MESSAGE UNITS> and <QUERY MESSAGE UNITS> are explained in detail on the next page.

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.



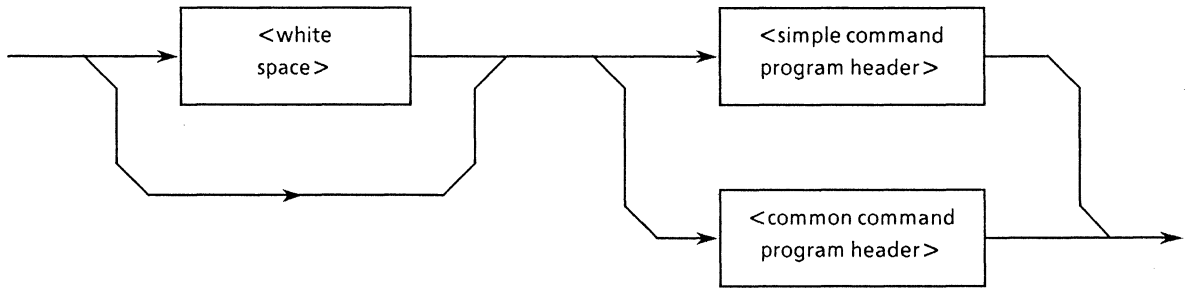
For both <COMMAND MESSAGE UNITS> and <QUERY MESSAGE UNITS>, a space must be inserted between the program header and any program data immediately following it. The application, function and operation of the program data can be seen from the program header. If there is no program data; the application,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

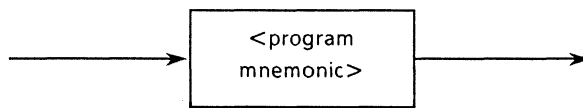
The special feature of the header is that a question mark is always tagged on at the end to indicate that it is a query.

5.2.8 <COMMAND PROGRAM HEADER>

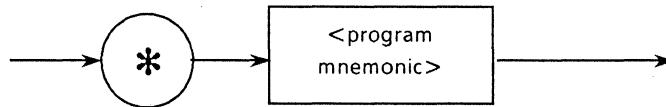
A <COMMAND PROGRAM HEADER> is defined as follows. A <white space> may be placed in front of each header.



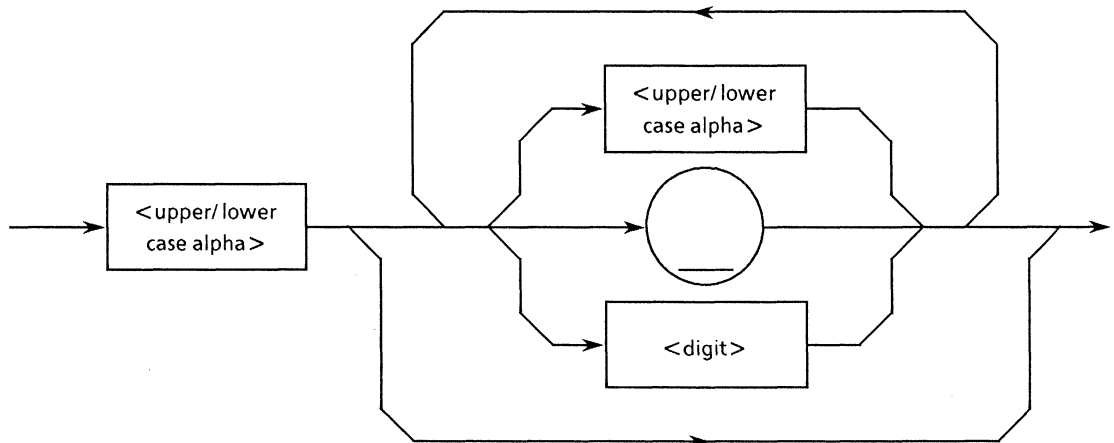
1) A <simple command program header> is defined as follows.



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.

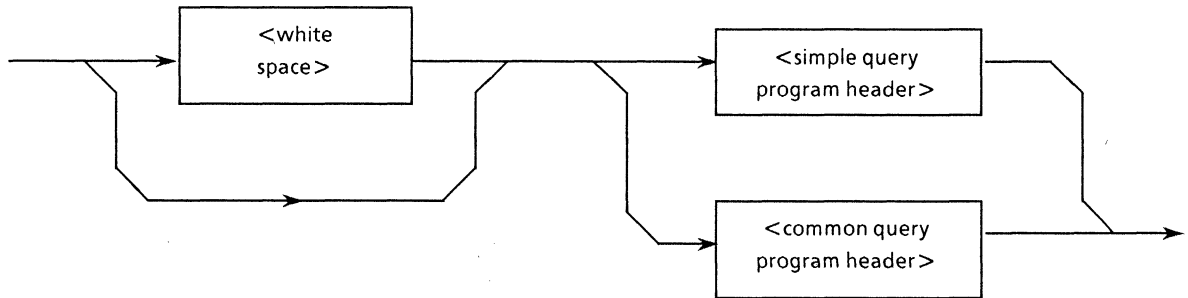
- Example

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.

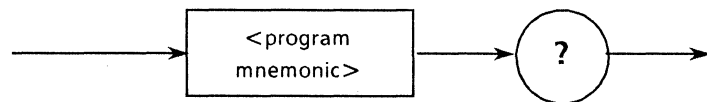
WRITE @103:"*RST": *RST is the common IEEE 488.2 command which executes the above.

5.2.9 <QUERY PROGRAM HEADER>

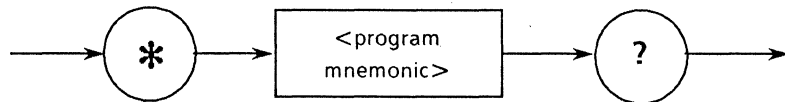
A <QUERY PROGRAM HEADER> is defined as follows. A <white space> is placed before each header.



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



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



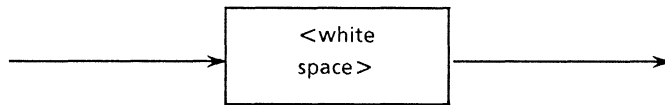
■ <QUERY PROGRAM HEADER>

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.

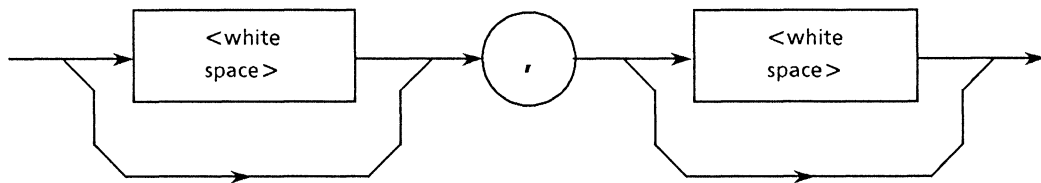


A <PROGRAM HEADER SEPARATOR> is used to separate a <COMMAND PROGRAM HEADER> or <QUERY PROGRAM HEADER> from <PROGRAM DATA>. When there is more than one <white space character> between a program header and program data, the first is interpreted as the separator and the rest are skipped. <white space characters>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

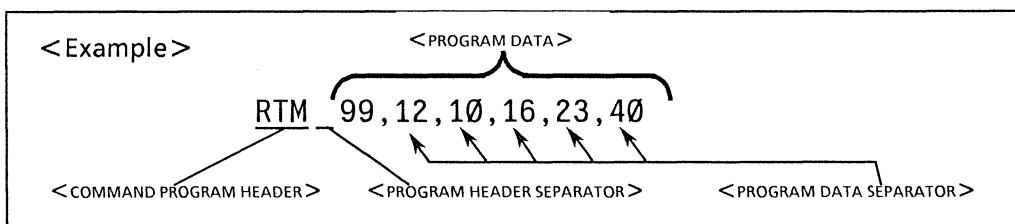
5.2.11 <PROGRAM DATA SEPARATOR>

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.

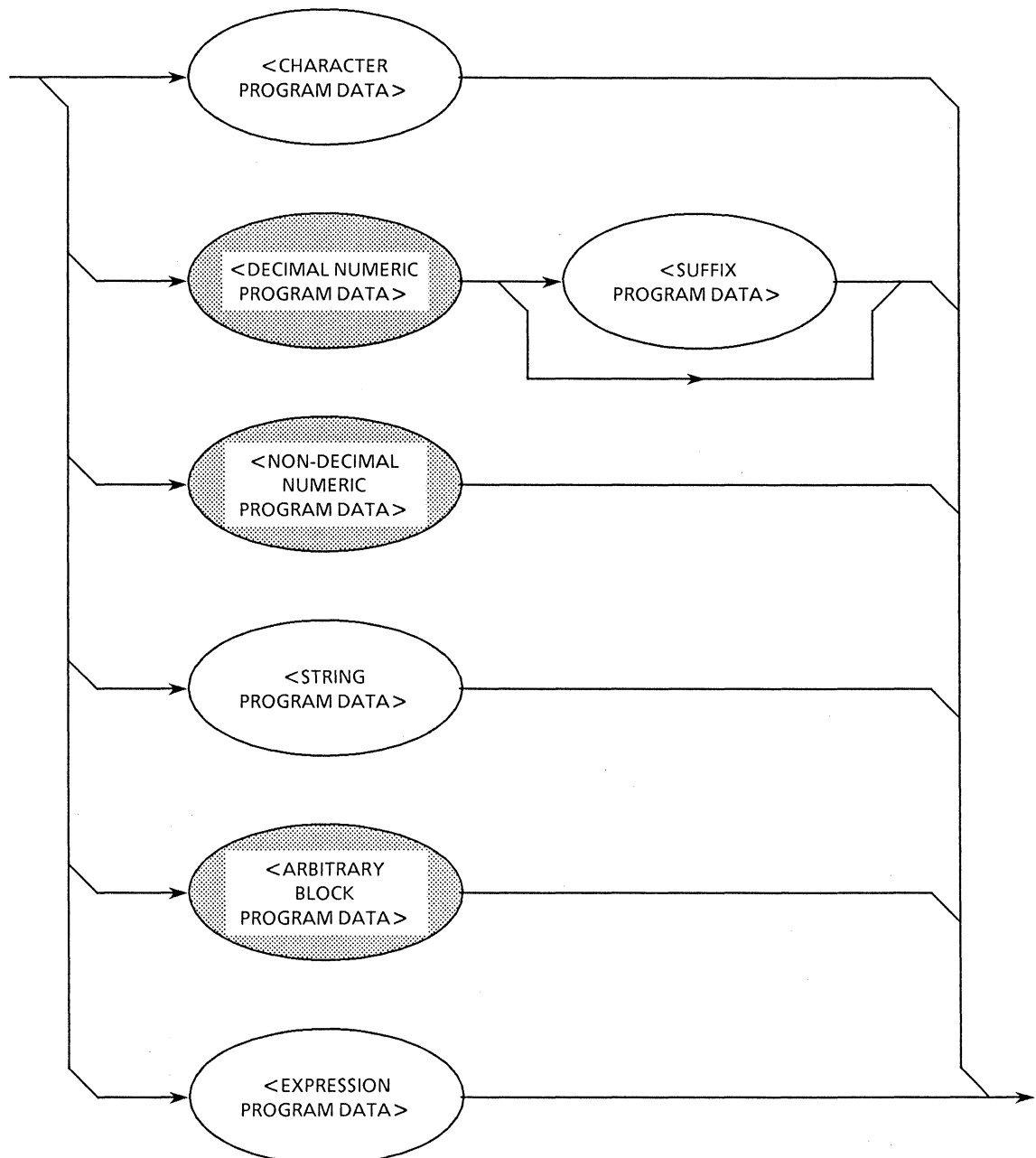
A comma must be used with a <PROGRAM DATA SEPARATOR>, but a white space does not always have to be used. A white space before or after the comma is skipped. They are used to make a program easier to read.



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 MP1763B 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 MP1763B does not use the floating point format.

The program data transmission in the integer or fixed point format used in the MP1763B is described.

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

- Rounding off of numeric elements

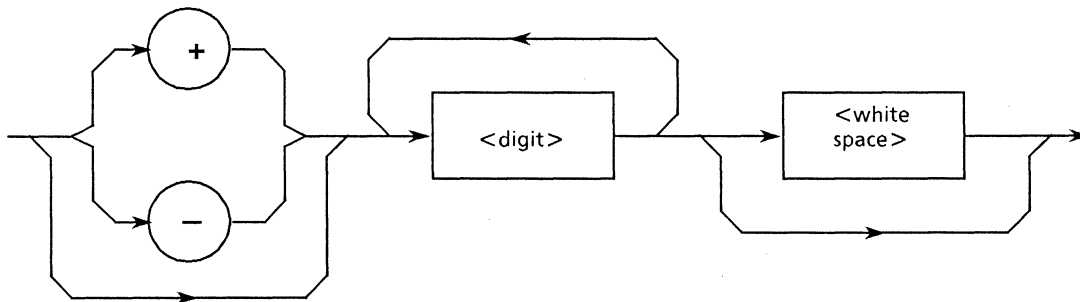
When a device receives <DECIMAL NUMERIC PROGRAM DATA> elements with more digits than it can handle, it ignores the sign and rounds it off to the nearest whole number.

- 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

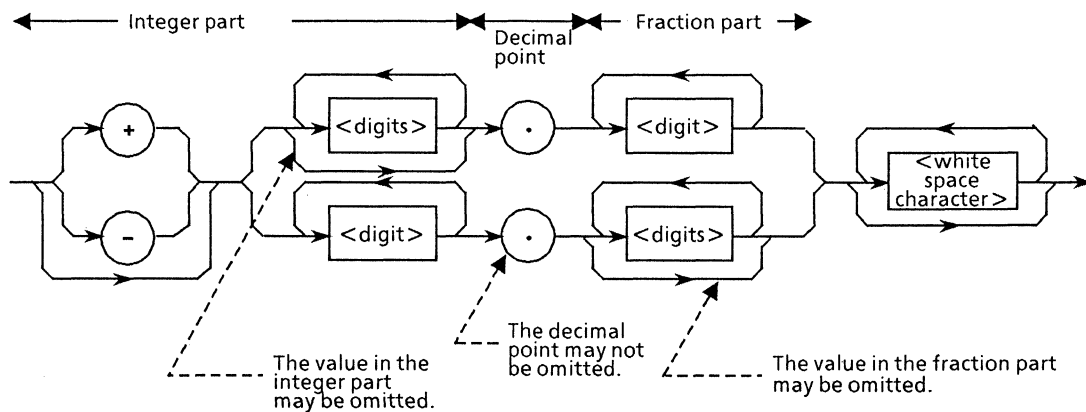
In the diagram below, an integer NR1, i.e. a decimal number which does not contain a floating point or exponential expression, is transmitted.



- 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

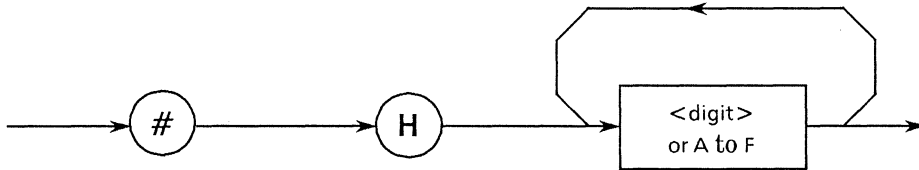
The example below shows the transmission of NR2, a real number with no integer or exponential expressions having digits after the decimal point. The syntax diagram consists of an integer part, the decimal point and a fraction part.



- 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.

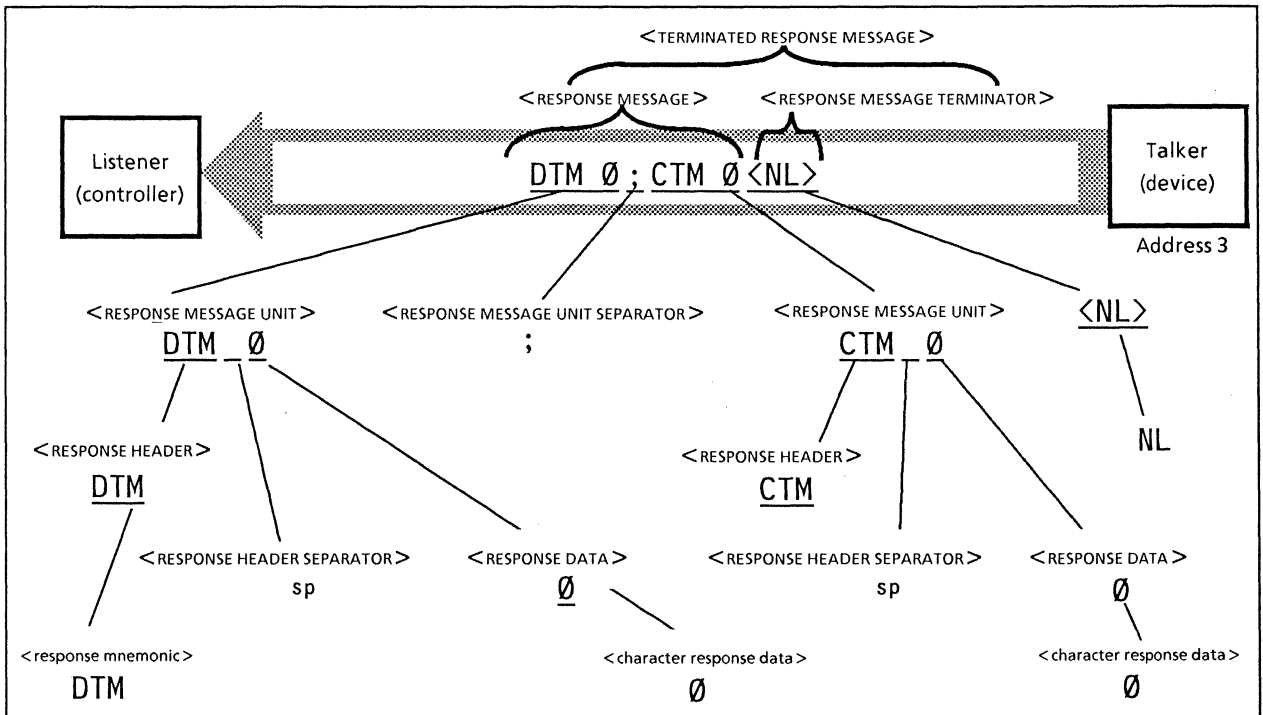
TABLE OF CONTENTS

6.1	Syntax Differences Between Formats of Listener Input and Talker Output	6-4
6.2	Functional Elements of Response Message	6-5
6.2.1	<TERMINATED RESPONSE MESSAGE>	6-5
6.2.2	<RESPONSE MESSAGE TERMINATOR>	6-6
6.2.3	<RESPONSE MESSAGE>	6-7
6.2.4	<RESPONSE MESSAGE UNIT SEPARATOR>	6-8
6.2.5	<RESPONSE MESSAGE UNIT>	6-8
6.2.6	<RESPONSE HEADER SEPARATOR>	6-9
6.2.7	<RESPONSE DATA SEPARATOR>	6-9
6.2.8	<RESPONSE HEADER>	6-9
6.2.9	<RESPONSE DATA>	6-11

(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 output termination voltage query message unit **DTM?** and a clock output termination voltage query message unit **CTM?**.



The program for the above would be as follows:

```

100 WRITE @103:"DTM? "!      Data output termination voltage query message request
110 READ @103:A$!           When the terminator NL is detected, the response message DTM△0 is read into
                             A$.
120 WRITE @103:"CTM? "!      Clock output termination voltage query message request
130 READ @103:B$!           Clock output termination voltage response message CTM△0

```

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 MP1763B.	Upper-case E only
+ sign in NR3 exponent	Can be omitted. Not supported by the MP1763B.	Cannot be omitted
< white space >	Two or more spaces can be placed before or after a separator and before a terminator.	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 separator	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)

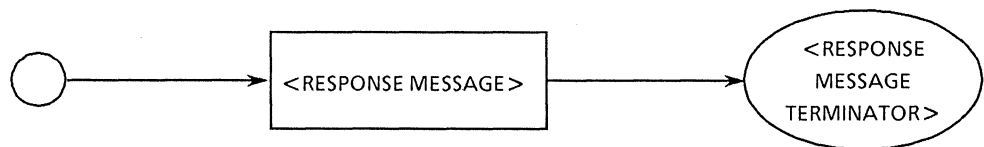
6.2 Functional Elements of Response Message

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>

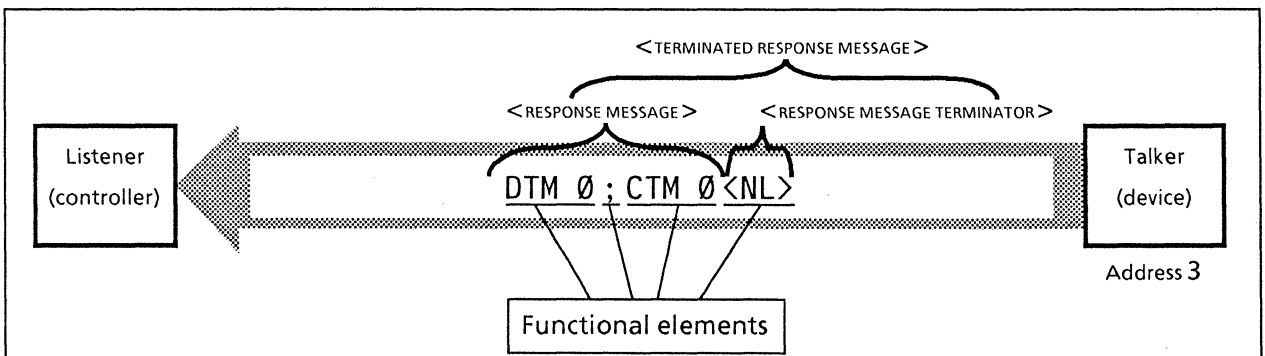
A <TERMINATED RESPONSE MESSAGE> is defined as follows:



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

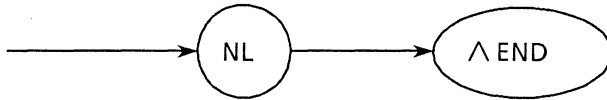
A <RESPONSE MESSAGE TERMINATOR> is attached to the end of a <RESPONSE MESSAGE> to terminate its transmission.

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\$(10)
- For **END (EOI)**: EOI ON

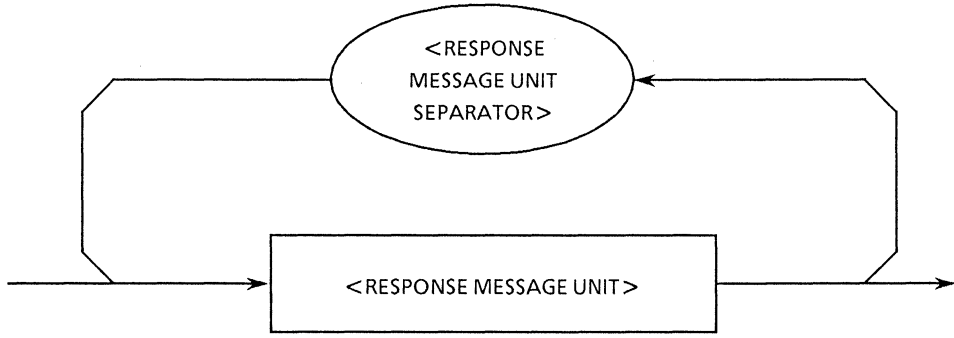
Example: To read the current center frequency setting

```

10 LET ADR=101
20 TERM IS CHR$(10)!      LF (new line) is assigned as the terminator code.
30 EOI ON!                When the last data byte is transmitted, the EOI signal is sent which makes the EOI line
                           true
40 WRITE @ADR:"DTM?!"!    Query to read the data output termination voltage
50 READ @ADR:A$!          EOI signal terminates the reading of response data
60 PRINT A$
70 END
  
```


6.2.3 <RESPONSE MESSAGE>

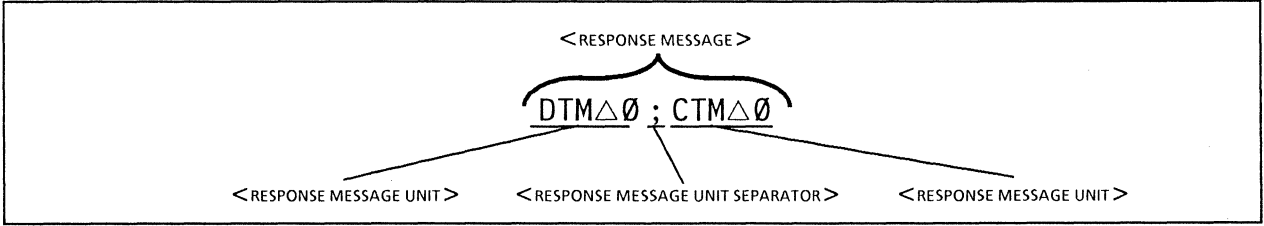
A <RESPONSE MESSAGE> is defined as follows.



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.

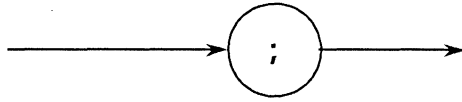
Example:

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



6.2.4 <RESPONSE MESSAGE UNIT SEPARATOR>

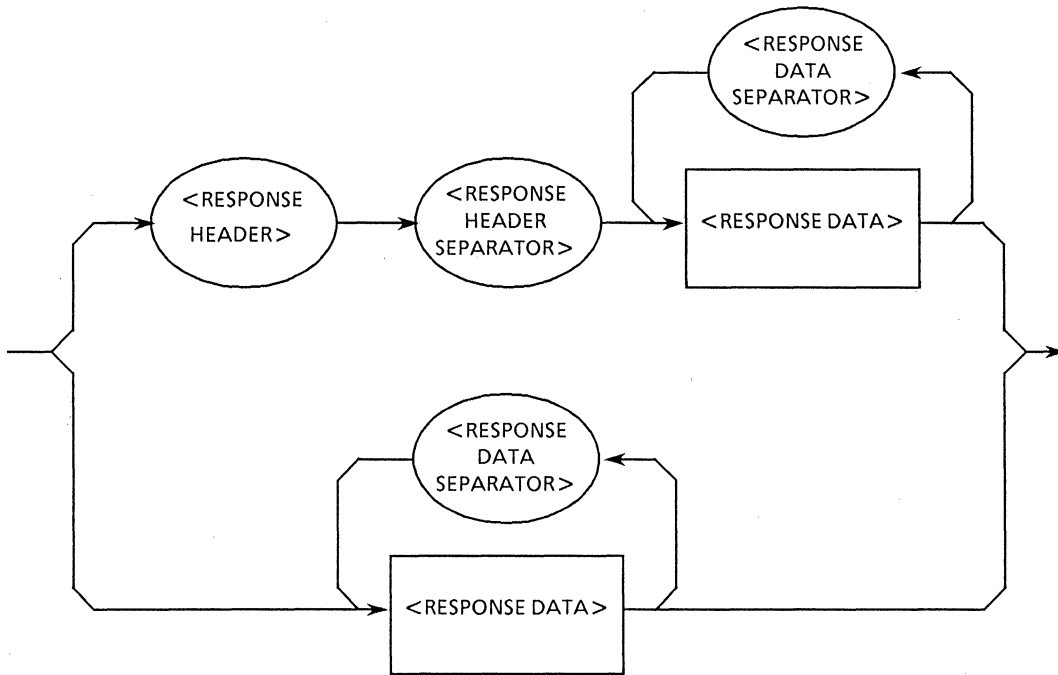
A <RESPONSE MESSAGE UNIT SEPARATOR> is defined as follows.



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>

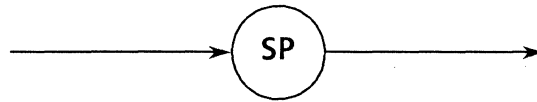
A <RESPONSE MESSAGE UNIT> is defined as follows.



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:

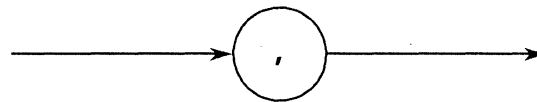


The <RESPONSE HEADER SEPARATOR> is a space after the <RESPONSE HEADER> to separate it from <RESPONSE DATA>. The space, SP, is ASCII code byte 20 (decimal 32).

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>

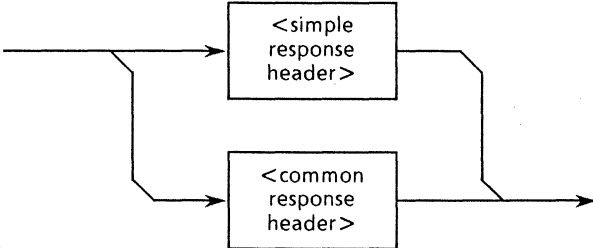
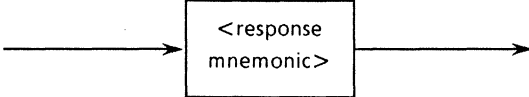
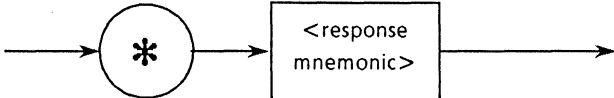
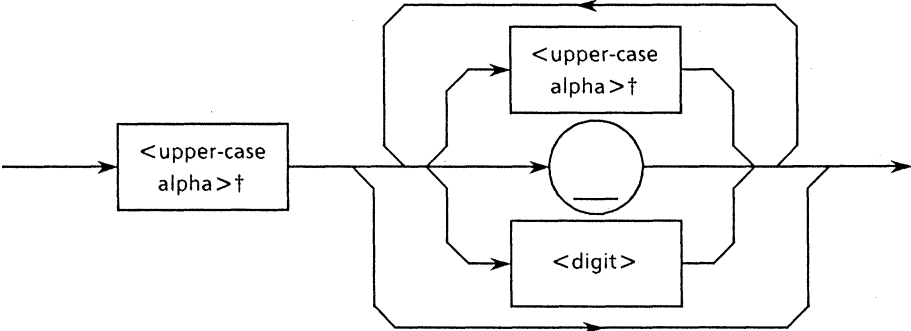
With the exception of the following three points, the format of the <RESPONSE HEADER> is the same as that described for the <COMMAND PROGRAM HEADER> in paragraph 5.2.8.

- 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.

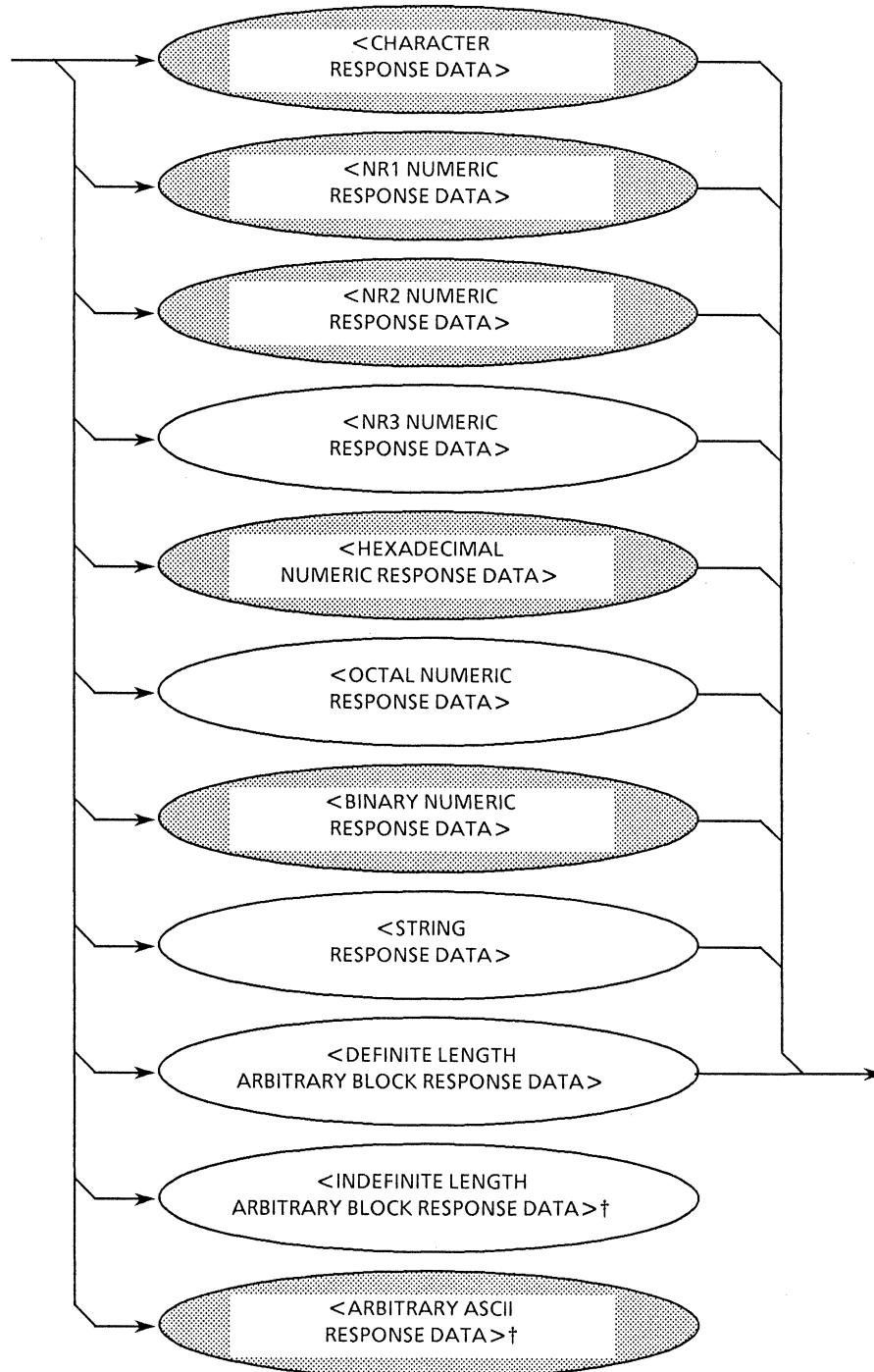
- ☞ For characters used in <response mnemonic>, alphabetic characters are always upper-case characters and other characters are used in the same manner as <response mnemonic>.)

SECTION 6 TALKER OUTPUT FORMAT

Element	Function
RESPONSE HEADER	<p>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.</p>  <p>1) A <simple response header> is defined as follows:</p>  <p>2) A <common response header> is defined as follows:</p>  <p>3) A <response mnemonic> is defined as follows:</p>  <p>†<upper-case alpha> ASCII code bytes 41 to 5A (decimal 65 to 90 = upper-case alphabetic A to Z)</p>

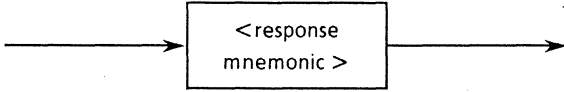
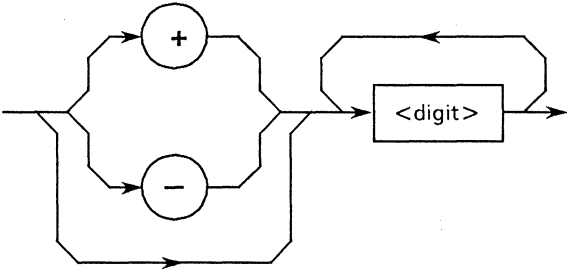
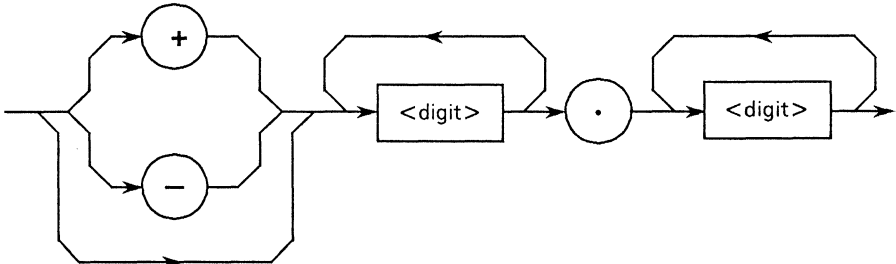
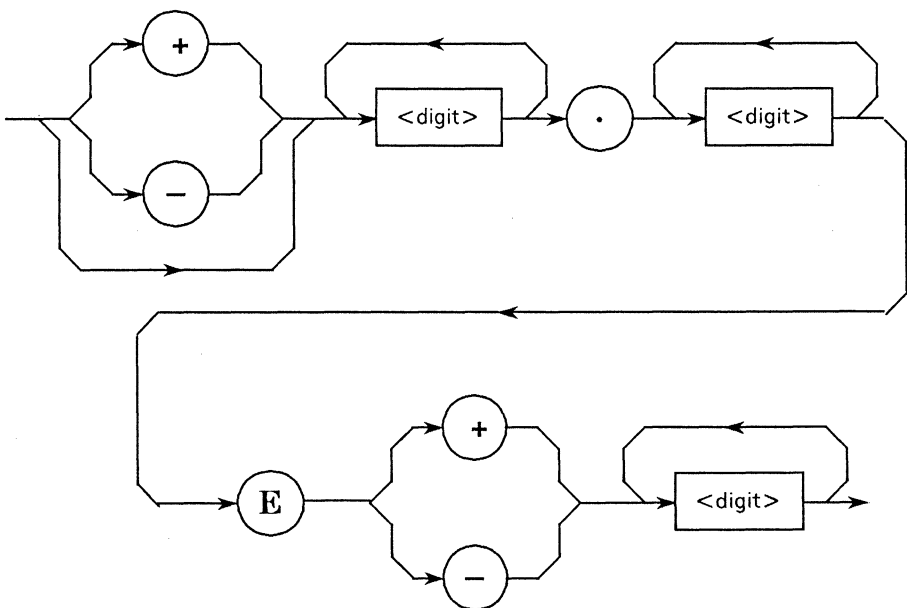
6.2.9 <RESPONSE DATA>

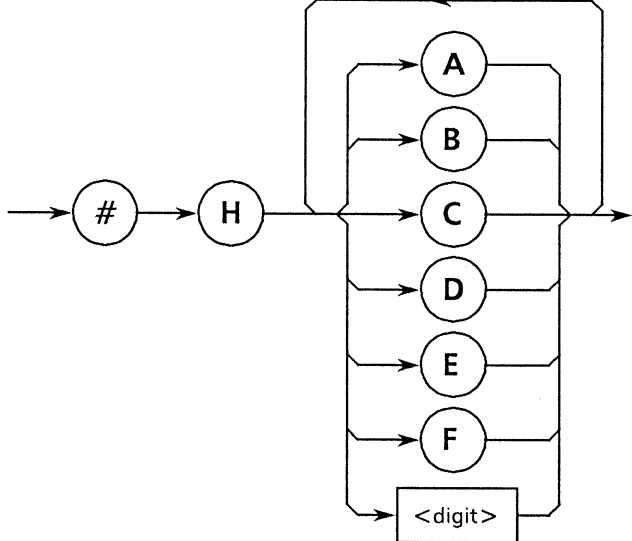
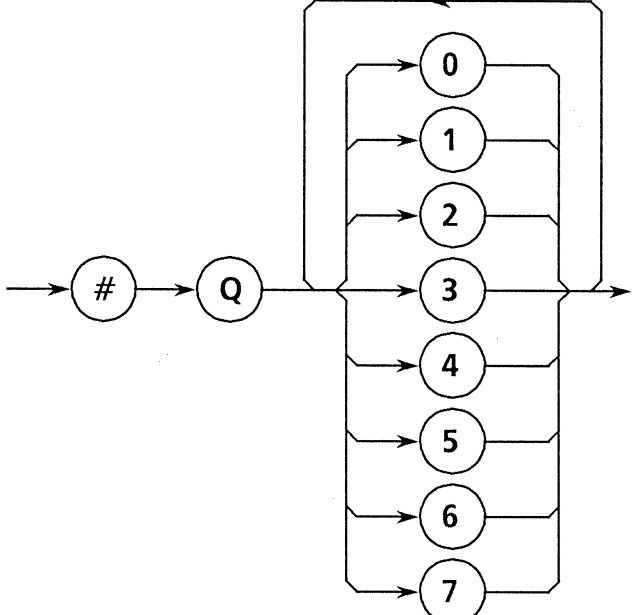
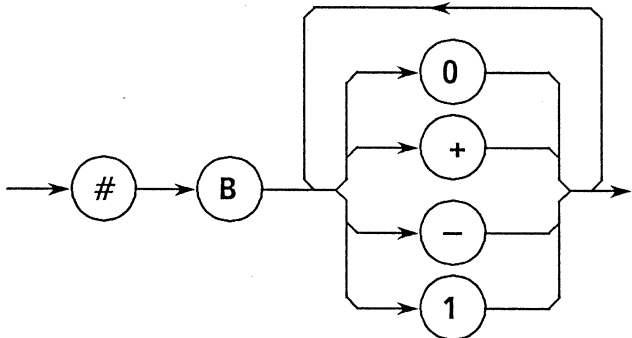
The diagram below shows the 11 types of response data. The MP1763B supports the response data in the shaded ovals below. The type of response data to be returned is determined by the query message.



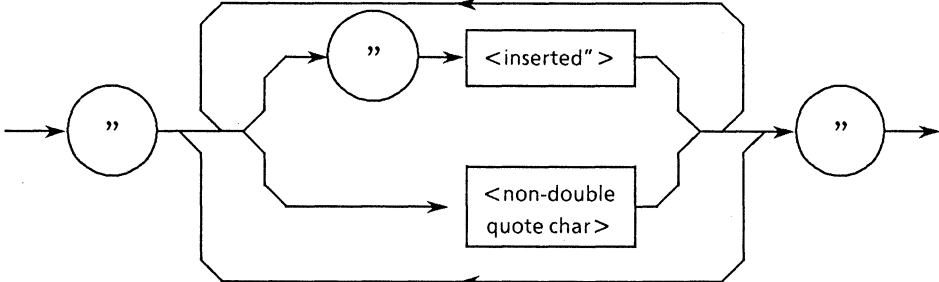
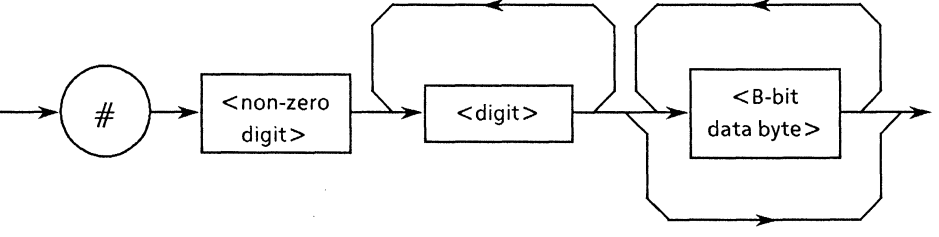
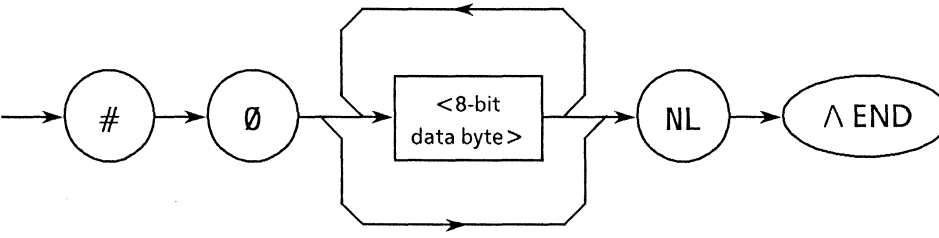
† Both <INDEFINITE LENGTH ARBITRARY BLOCK RESPONSE DATA> and <ARBITRARY ASCII RESPONSE DATA> are terminated by an NL/END in their own last data byte.

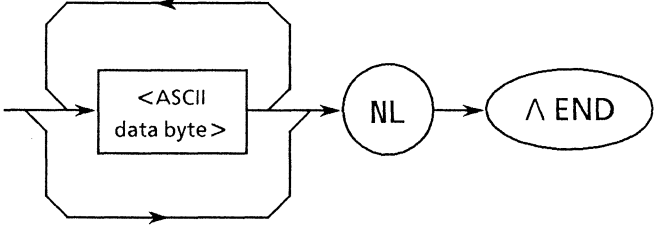
SECTION 6 TALKER OUTPUT FORMAT

Element	Function
<p>(1) CHARACTER RESPONSE DATA</p> <p><Example> AAT2_AUTO AAT2_MANUAL</p>	<p>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.</p> 
<p>(2) NR1 NUMERIC RESPONSE DATA</p> <p><Example> 123 + 123 - 1234</p>	<p>Integer data, i.e. decimal values without a decimal point or exponents.</p> 
<p>(3) NR2 NUMERIC RESPONSE DATA</p> <p><Example> 12.3 + 12.34 - 12.345</p>	<p>Fixed-point data, i.e. decimal values without integers or exponents.</p> 
<p>(4) NR3 NUMERIC RESPONSE DATA</p> <p><Example> 12.3E + 4 + 12.34E - 5 - 12.345 E + 6</p> <ul style="list-style-type: none"> ● 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. 	<p>Floating-point data, i.e. decimal values with exponent digits.</p> 

Element	Function
<p>(5) HEXADECIMAL NUMERIC RESPONSE DATA</p> <p><Example> #HABC123 #H2DC3 #H8301</p>	<p>Hexadecimal numeric data.</p>  <p>The diagram shows an input arrow pointing to a circle containing '#'. An arrow from '#' points to a circle containing 'H'. From 'H', an arrow points to a vertical stack of six circles labeled 'A', 'B', 'C', 'D', 'E', and 'F'. Below these is a rectangular box containing '<digit>'. Arrows from each of these seven elements point to a common output line on the right. A feedback loop arrow at the top points from the output line back to the input of the 'A' circle.</p>
<p>(6) OCTAL NUMERIC RESPONSE DATA</p> <p><Example> #Q37 #Q26703 #Q30562</p>	<p>Octal numeric data.</p>  <p>The diagram shows an input arrow pointing to a circle containing '#'. An arrow from '#' points to a circle containing 'Q'. From 'Q', an arrow points to a vertical stack of eight circles labeled '0', '1', '2', '3', '4', '5', '6', and '7'. Arrows from each of these eight elements point to a common output line on the right. A feedback loop arrow at the top points from the output line back to the input of the '0' circle.</p>
<p>(7) BINARY NUMERIC RESPONSE DATA</p> <p><Example> #B011101 #B1011 #B1011</p>	<p>Binary numeric data.</p>  <p>The diagram shows an input arrow pointing to a circle containing '#'. An arrow from '#' points to a circle containing 'B'. From 'B', an arrow points to a vertical stack of four circles labeled '0', '+', '-', and '1'. Arrows from each of these four elements point to a common output line on the right. A feedback loop arrow at the top points from the output line back to the input of the '0' circle.</p>

SECTION 6 TALKER OUTPUT FORMAT

Element	Function
<p>(8) STRING RESPONSE DATA</p> <p><Example> "This is a text" "Say," "Hello" "."</p>	<p>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, LF's and spaces are available.</p> 
<p>(9) DEFINITE LENGTH ARBITRARY BLOCK RESPONSE DATA</p> <p><Example> Transferring 11256099D in 4 byte length ↓ #1400ABC123</p>	<p>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.</p> 
<p>(10) INDEFINITE LENGTH ARBITRARY BLOCK RESPONSE DATA</p> <p><Example> Transferring -250, -50, 120, ... in undefined length ↓ #0FF06FFCE0078</p>	<p>Undefined-length 8 bit binary block data. So, the first data is preceded with #Ø. The last data is terminated by NL^END.</p> 

Element	Function
<p>(11) ARBITRARY ASCII RESPONSE DATA</p> <p><Example1> <ASCII Byte> <ASCII Byte> NL^END</p> <p><Example2> NL^END</p>	<p>ASCII data bytes (excluding NL characters) sent without separating them; so, the last data is terminated by NL^END.</p>  <pre> graph LR Input(()) --> Data[<ASCII data byte>] Data --> NL((NL)) NL --> End([^END]) </pre> <p>The diagram illustrates the data flow. It starts with an input arrow pointing to a box labeled '<ASCII data byte>'. This box is enclosed in a larger, irregular shape with arrows indicating a loop or continuation. An arrow then points from the box to a circle labeled 'NL'. Finally, an arrow points from the 'NL' circle to an oval labeled '^END'.</p>

(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

7.1	Classification by Function of Common Commands Supported by the MP1763B	7-3
7.2	The Classification of Commands Supported and the Reference	7-4

(Blank)

7.1 Classification by Function of Common Commands Supported by the MP1763B

The table below shows the classification by function of the IEEE 488.2 common commands supported by the MP1763B. 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 MP1763B 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

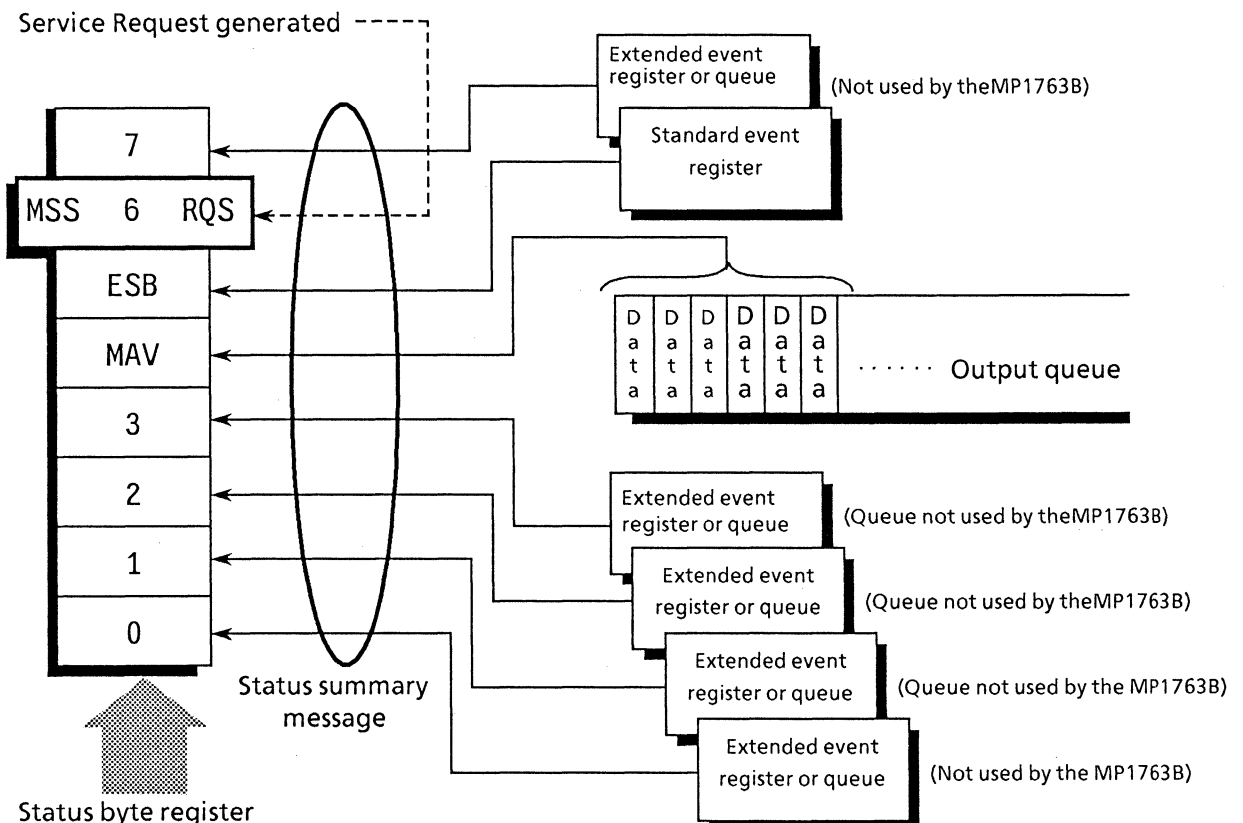
The *CLS common command clears all status data structures (i.e their event registers and queues) except for the output queue and its MAV summary messages. It also clears the summary messages corresponding to these structures.

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:

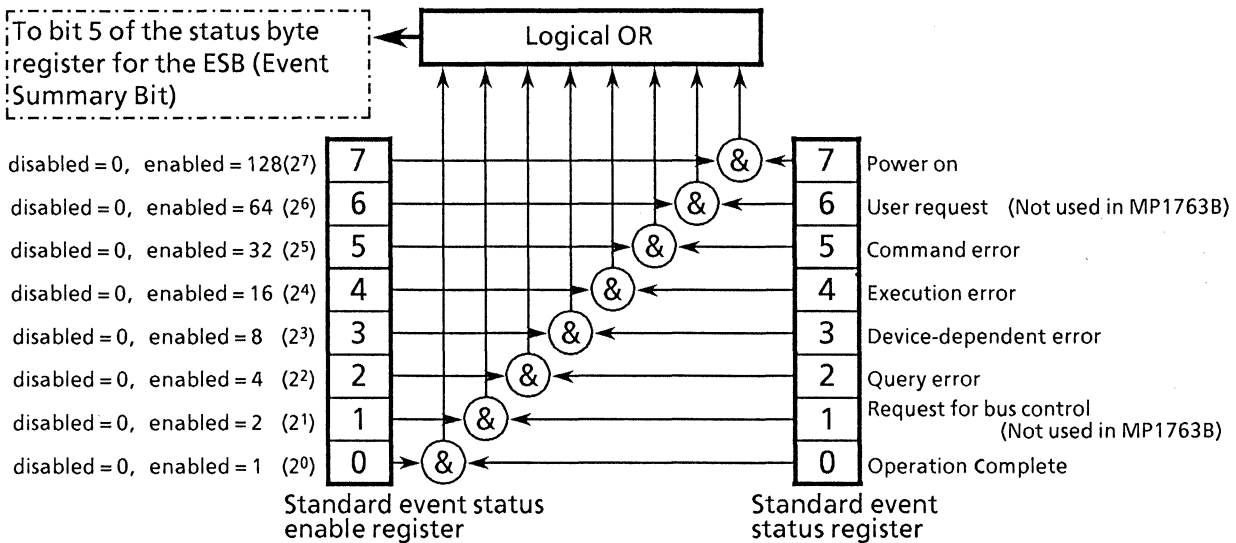
<DECIMAL NUMERIC PROGRAM DATA> = Value rounded to an integer from 0 to 255 (Binary weighted with a base value of 2)

■ **Example**

WRITE @103:"*ESE 20"! 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
```

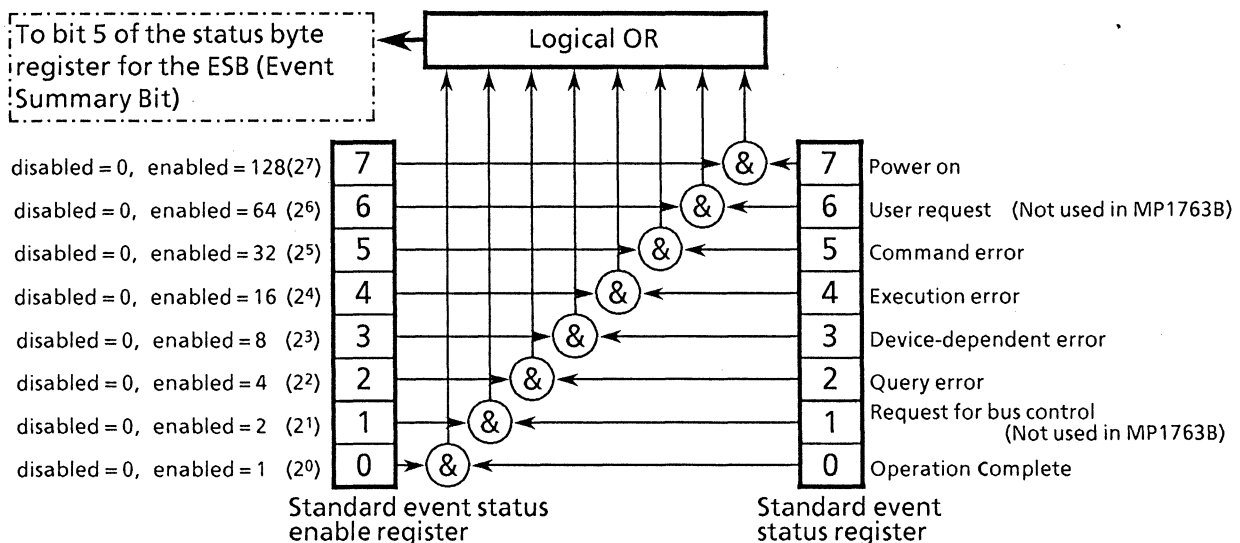
■ Response Message

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 amongst 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

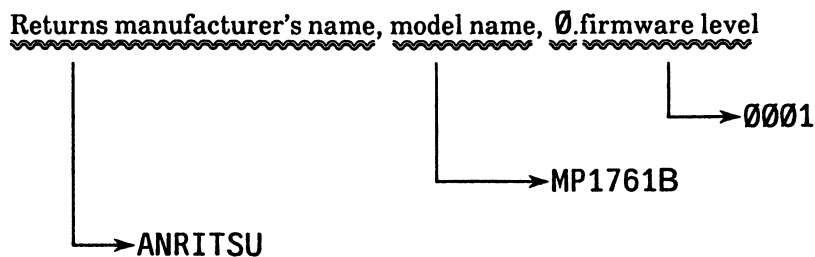
*IDN?

■ Example

```
30 WRITE @103:"*IDN?"  
40 READ @103:IDEN$!
```

Stores names of manufacturer, model, etc.

■ Explanation



If an *IDN? common query is sent to a device when the manufacturer is Anritsu, the model is MP1763B, 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 (MP1761B)
- ③ Field 3 (usually \emptyset)
- 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,MP1761B, \emptyset , $\emptyset\emptyset\emptyset1$

The total length of a response message is ≤ 72 characters

Note

Even if the real model name is MP1763B, a response message is MP1761B.

***OPC Operation Complete Command**

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

■ Syntax

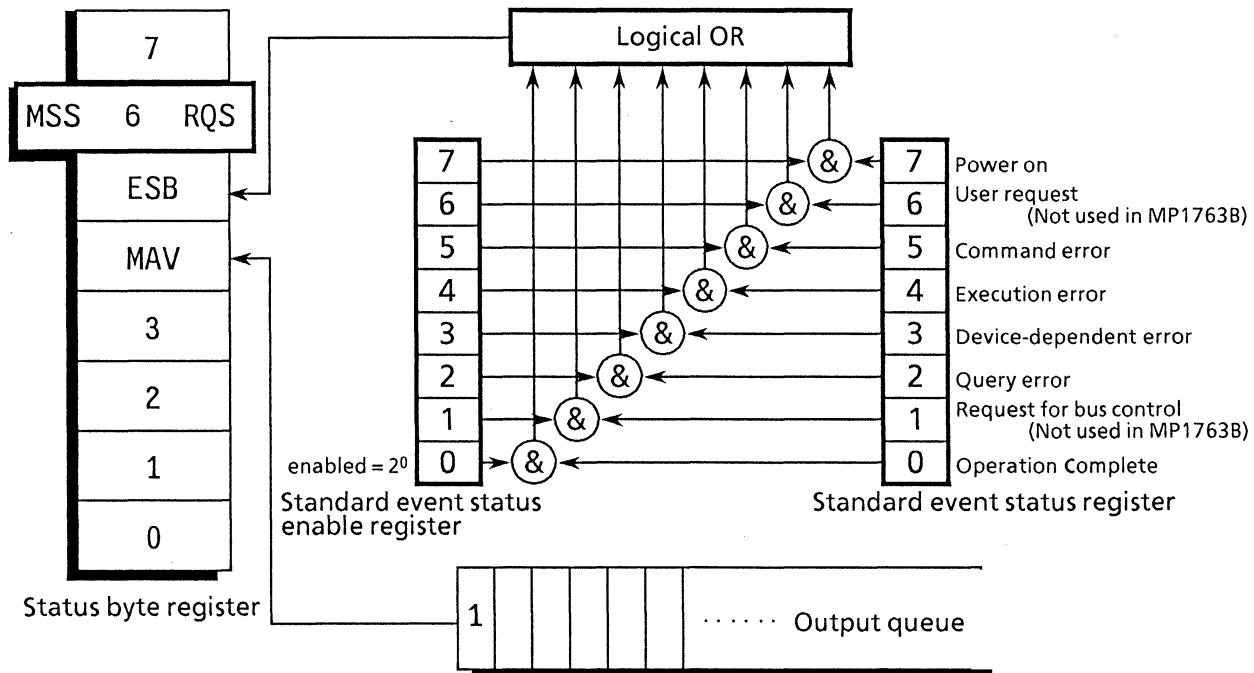
*OPC

■ 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 @103:"*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> = 0 : not cleared
Numbers in range of -32767 to 32767 : cleared

■ Example

```
WRITE @103:"*PSC 0;*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 0** 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**

```
3Ø WRITE @1Ø3:"*PSC?"  
4Ø READ:POWF
```

■ **Explanation**

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

■ **Response message**

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

***RST Reset Command**
(Resets (initializes) device in level 3)

■ Syntax

*RST

■ Example

WRITE @103:"*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>

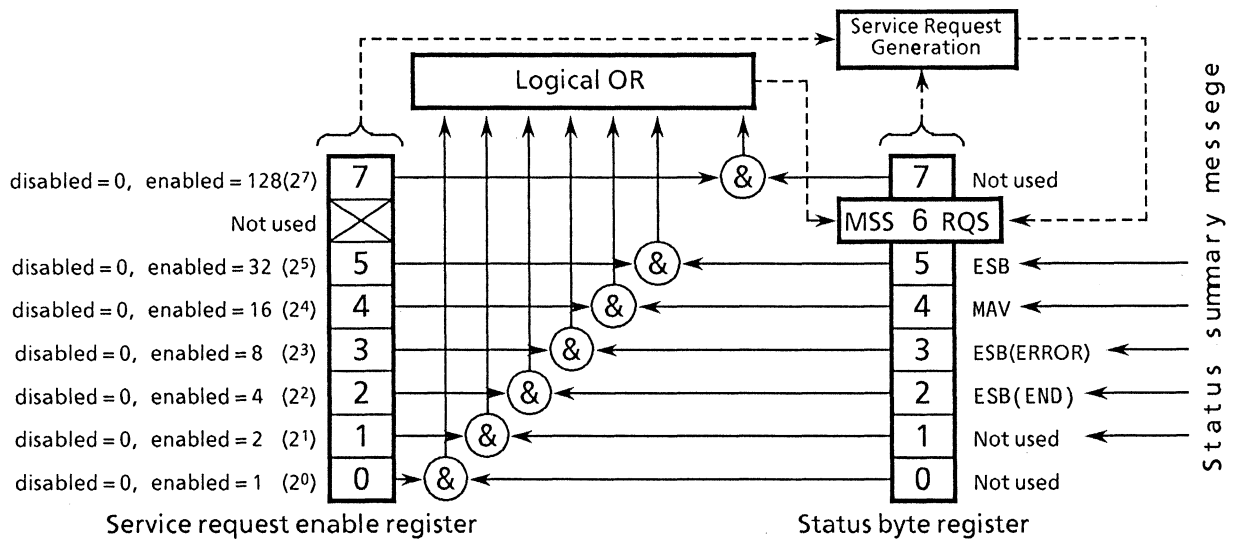
<DECIMAL NUMERIC PROGRAM DATA> = Values rounded to an integer from 0 to 255 (binary weighted with a base value of 2)

■ **Example**

WRITE @103: "*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

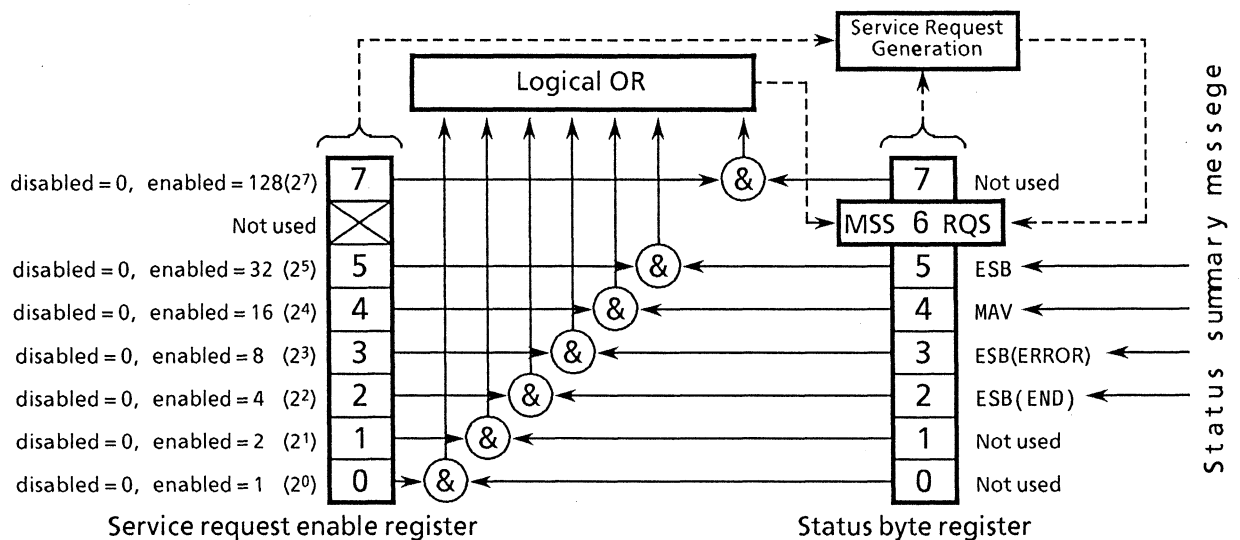
```
30 WRITE @103:"*STB?"
40 READ @103:STBV
50 PRINT STBV
```

■ 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 MP1763B'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 @103:"*TRG"
```

■ Explanation

The *TRG common command has the same function as the IEEE 488.1 GET – Group Execute Trigger-bus command. The MP1763B does not support the *DDT command.

With the MP1763B, *TRG common command has no function.

```
WRITE @103:"*TRG"
```

***TST? Self-test Query**

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

■ Syntax

*TST?

■ Example

```
30 WRITE @103:"*TST?"
40 READ @103:TEST
50 PRINT TEST
```

■ Explanation

The *TST? query executes the self-test of the internal circuit 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. Operator intervention is not required to execute the self-test.

When the power is turned on, the MP1763B 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 = 0 Indicates no errors

NR1 ≠ 0 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 @103:"*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

8.1	IEEE 488.2 Standard Status Model	8-4
8.2	Status Byte (STB) Register	8-6
8.2.1	ESB and MAV summary messages	8-6
8.2.2	Device-dependent summary messages	8-7
8.2.3	Reading and clearing the STB register	8-8
8.3	Enabling SRQ	8-10
8.4	Standard Event Status Register	8-11
8.4.1	Bit definition	8-11
8.4.2	Query error details	8-13
8.4.3	Reading, writing to and clearing the standard event status register	8-14
8.4.4	Reading, writing to and clearing the standard event status enable register ...	8-14
8.5	Extended Event Status Register	8-15
8.5.1	Bit definition of END event status register	8-16
8.5.2	Bit definition of ERROR event status register	8-18
8.5.3	Reading, writing to and clearing the extended event status register	8-20
8.5.4	Reading, writing to and clearing the extended event status enable register ..	8-20
8.6	Queue Model	8-21
8.7	Techniques for Synchronizing Devices with the Controller	8-23
8.7.1	Enforcing the sequential execution	8-23
8.7.2	Wait for a response from the output queue	8-24
8.7.3	Wait for a service request	8-25

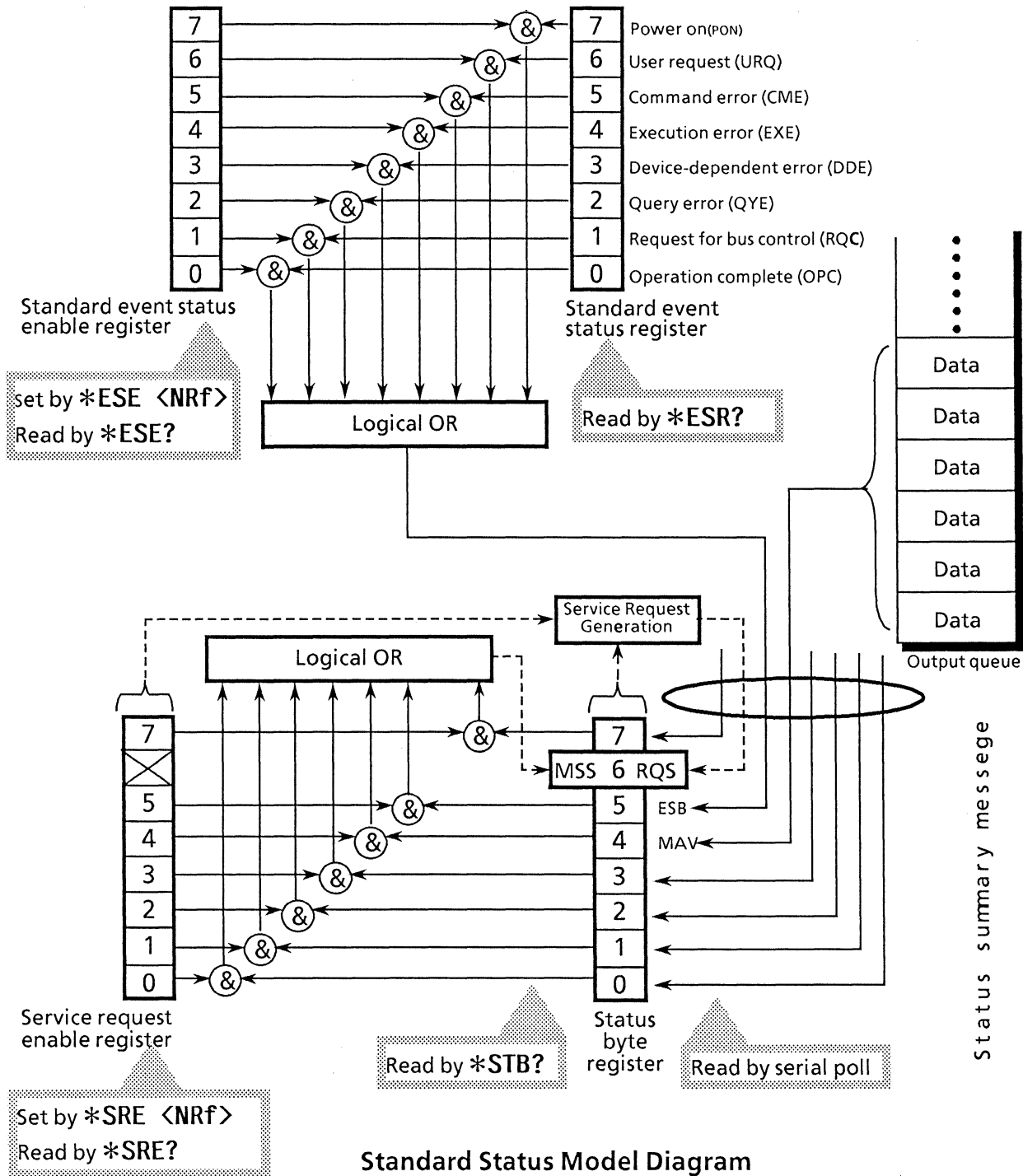
(Blank)

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.

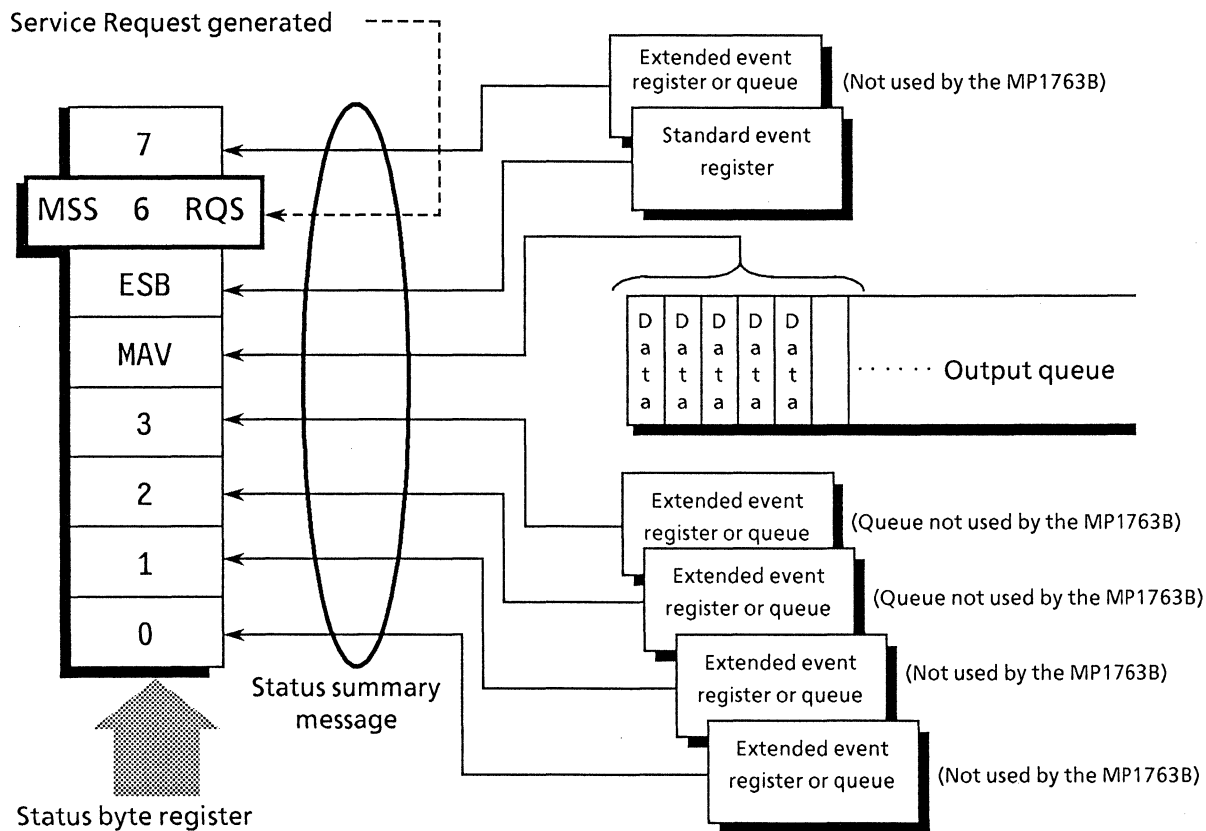
Reading the output queue without first checking MAV will cause all system bus operations to be delayed until the device responds.

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 MP1763B 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.

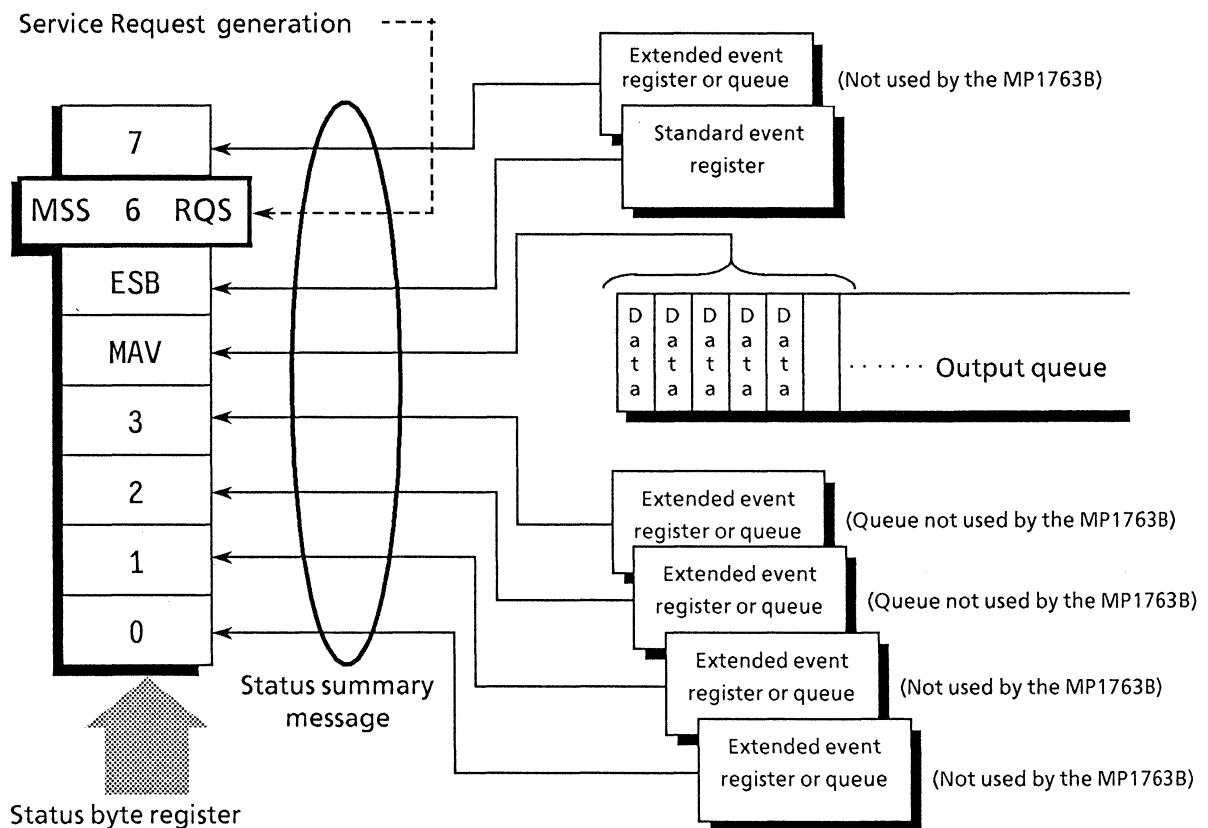
(4) Clearing the STB register by the *CLS common command

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:"DTMΔ0;CTMΔ0"
40 WRITE @103:"*CLS;DTM?"
```

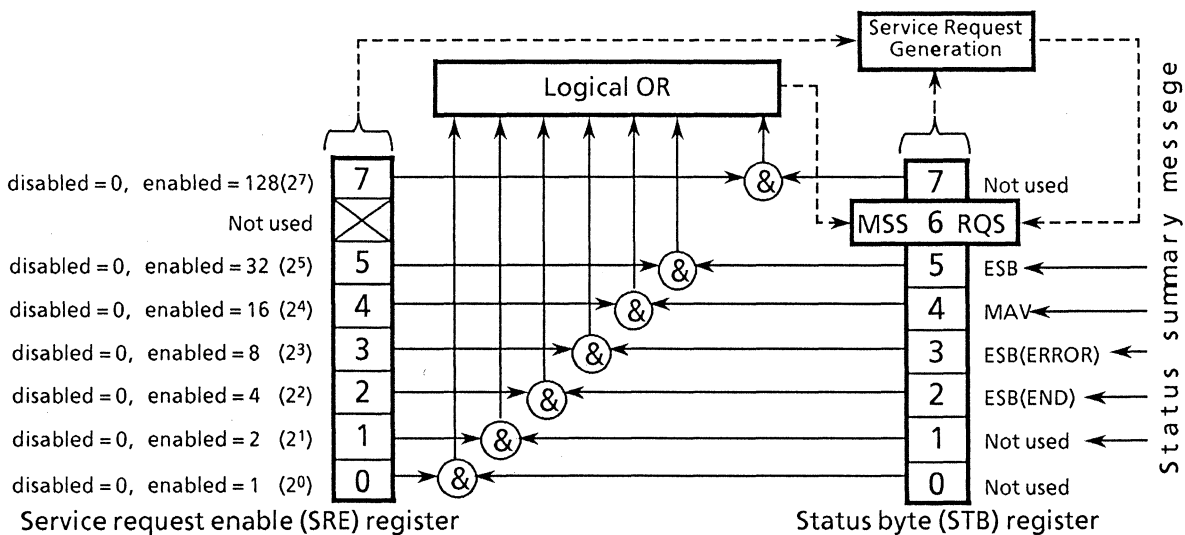
That is to say, sending a *CLS command (after a <PROGRAM MESSAGE TERMINATOR> or before <QUERY MESSAGE UNIT> elements) clears all status bytes. This clears all unread messages in the output queue and sets the MAV message to false. The MSS message is also set to false when a response is made to *STB?. 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.

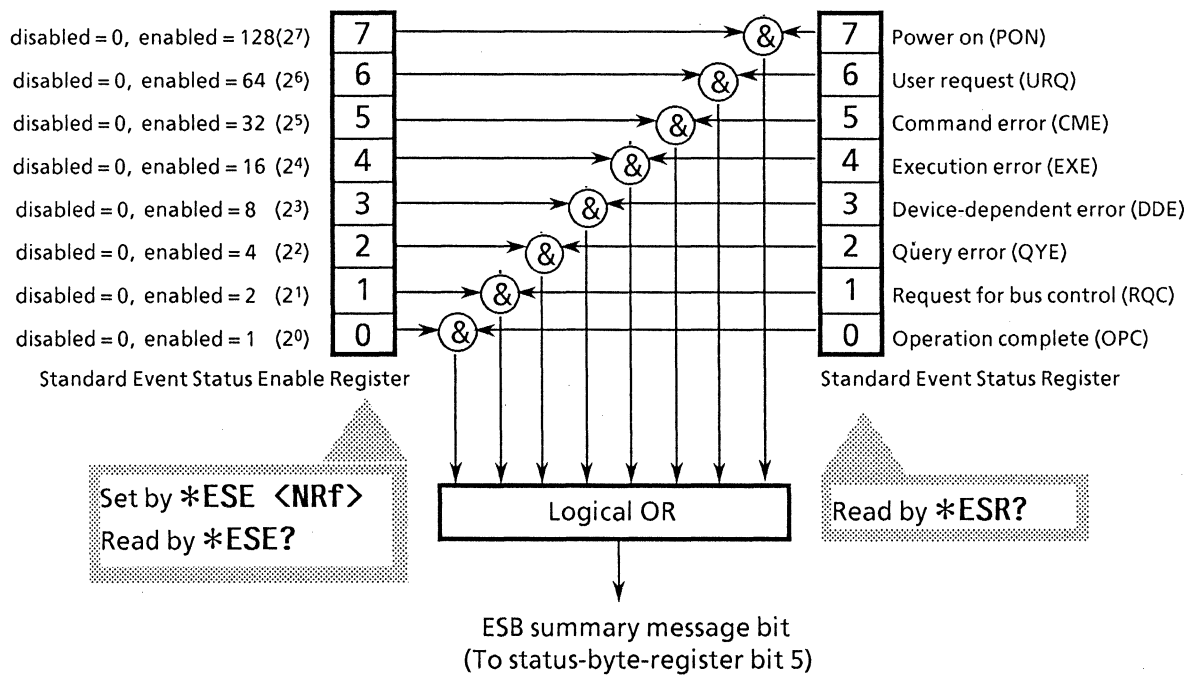
Using the `*SRE` common command, the SRE register is cleared by setting the value of the `<DECIMAL NUMERIC PROGRAM DATA>` element to 0. Clearing the register stops status information from generating rsv local messages, and service requests are no longer generated.

The MP1763B has the `*PSC` command. Therefore, if the PSC flag is true 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.



SECTION 8 STATUS STRUCTURE

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 MP1763B 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 MP1763B 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

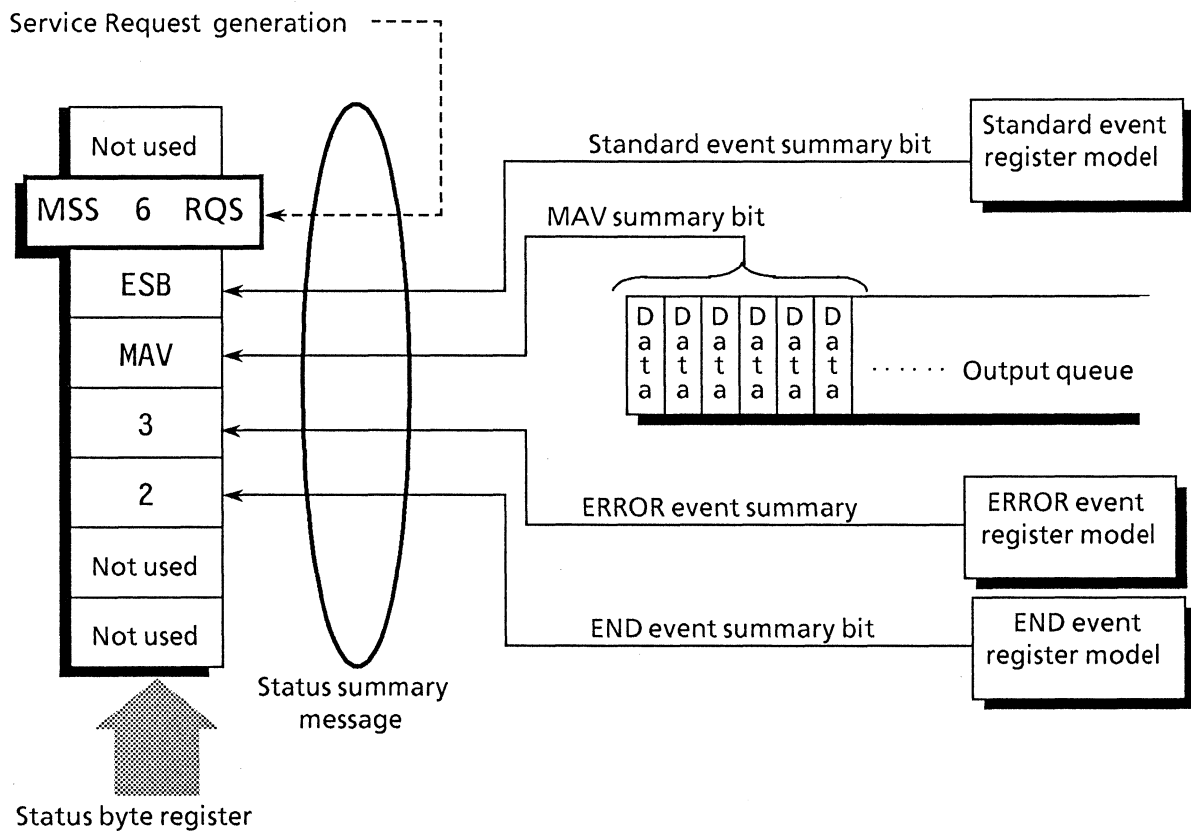
8.5 Extended Event Status Register

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 MP1763B, 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.

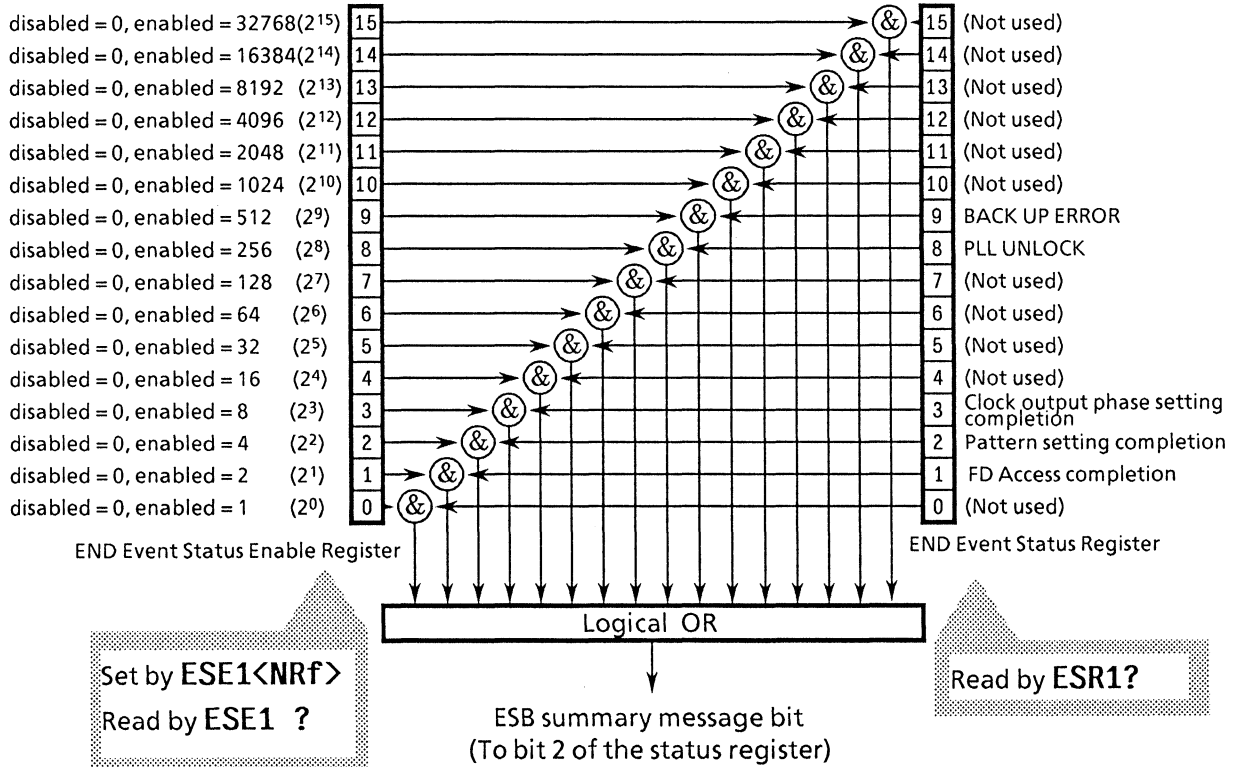
As the queue model is not extended, there is only one type of queue - the output queue.



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

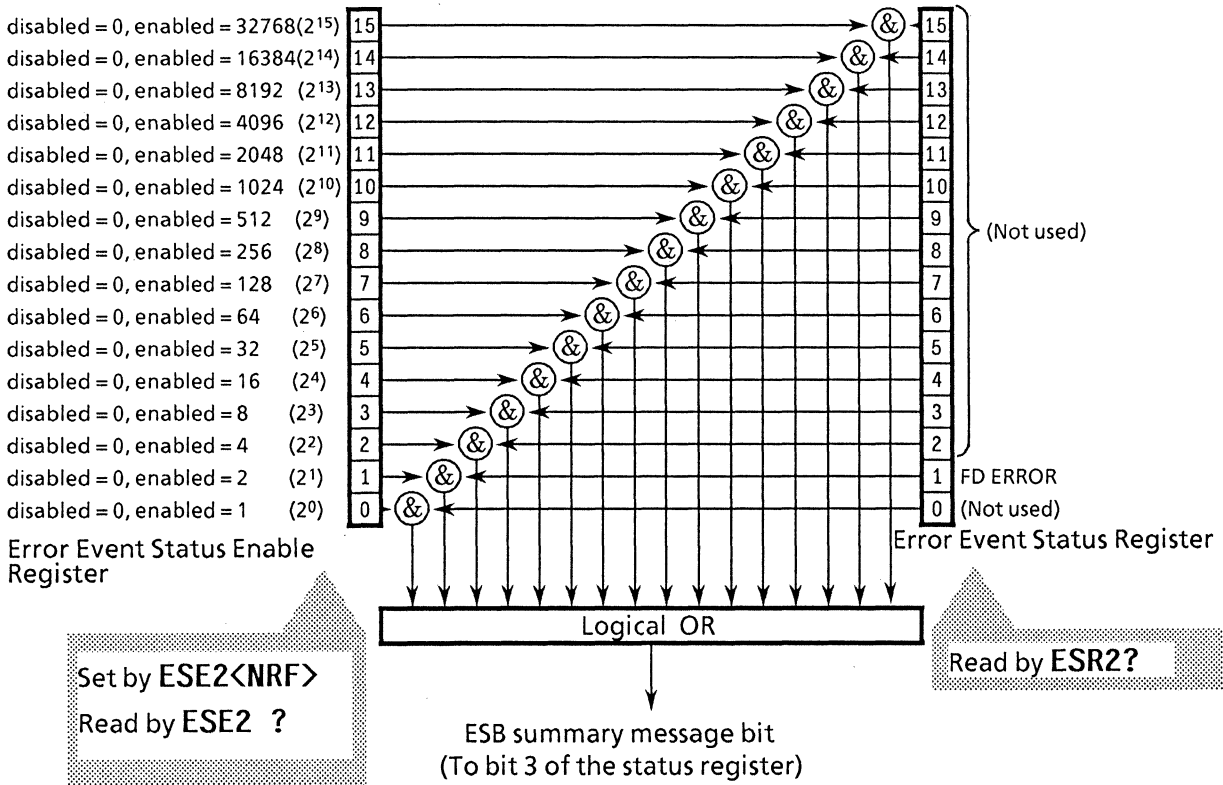
The following describes the operation of the END event status register model, the naming of its event bits and what they mean.



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	(Not used)	(Not used)
9	BACK UP ERROR	Error has been detected from back up data.
8	PLL UNLOCK	Synthesizer PLL unlock has been detected (only when OPTION-01 is installed).
7	(Not used)	(Not used)
6	(Not used)	(Not used)
5	(Not used)	(Not used)
4	(Not used)	(Not used)
3	Clock output phase setting completion	The servo circuit used for setting clock output 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	(Not used)	(Not used)

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 ERROR	FD abnormal status has occurred.
0	(Not used)	(Not used)

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

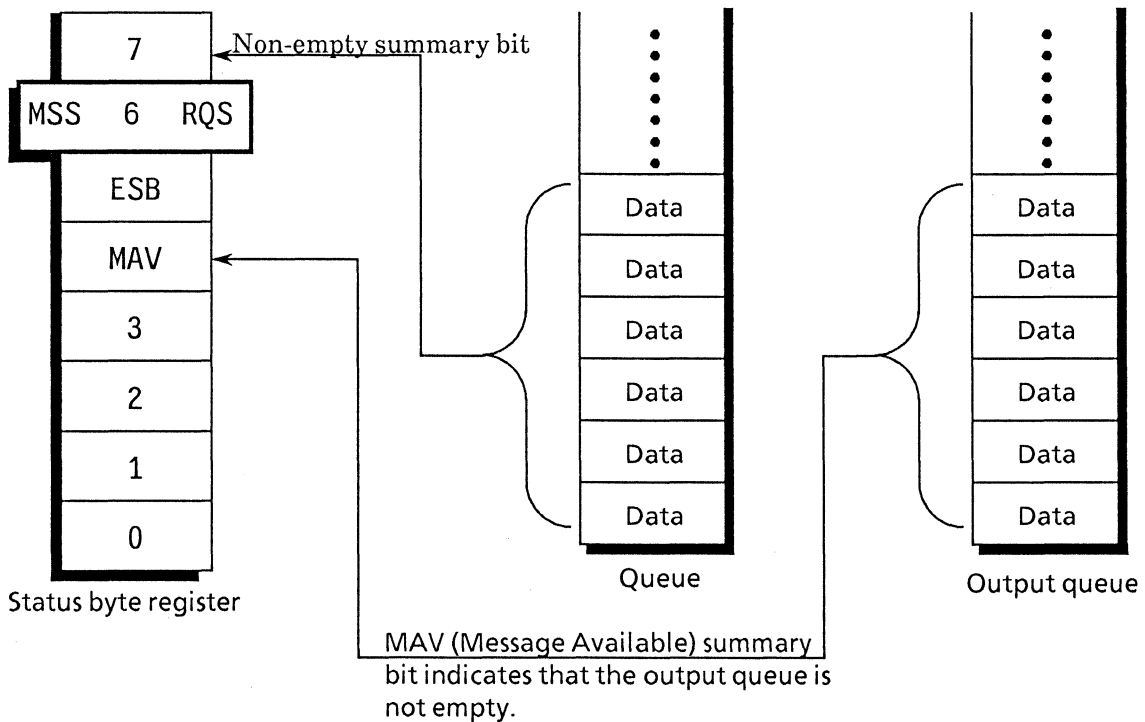
Reading	The register is destructively read by the a query, i.e. it is cleared after being read. The END and ERROR event status registers are read by the ESR1? and ESR2? queries. The read value, <NR1>, is obtained by binary weighting the event bit and converting it to decimal.
Writing	With the exception of clearing, writing operations cannot be performed externally.
Clearing	The register is cleared in the following cases. 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 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 ESE1? and ESE2? 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 ESE1 and ESE2 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 ESE1 and ESE2 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.

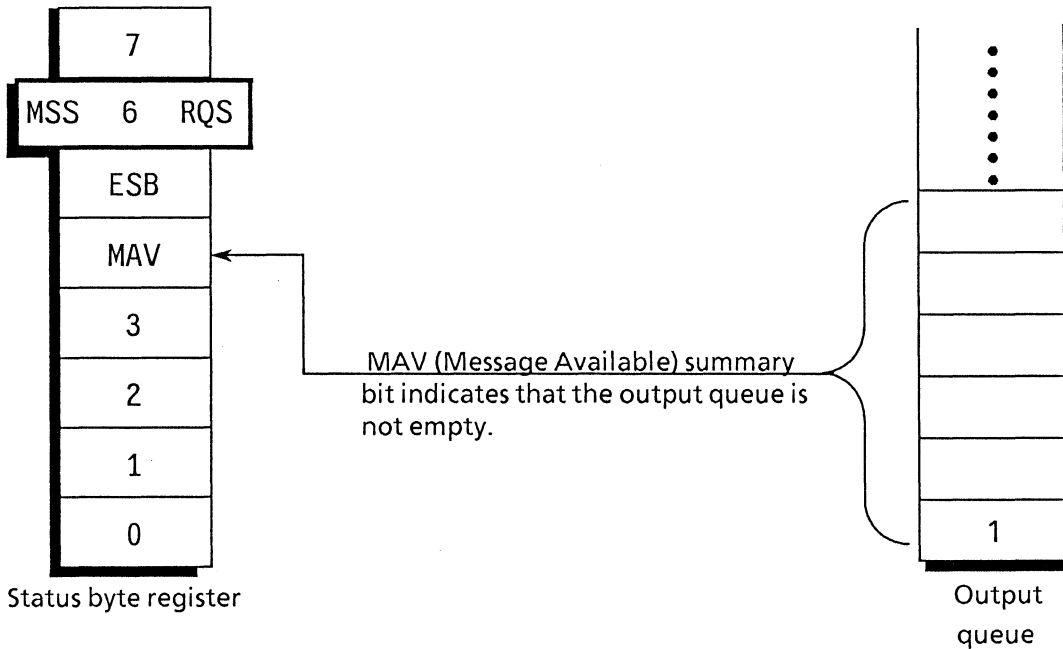
Enforcing the sequential execution is a synchronizing technique used to enforce a command that natively acts as an overlap command to be executed sequentially and not to perform the next process until one process has been completed. In this technique, the ***WAI** command is used.

8.7.2 Wait for a response from the output queue

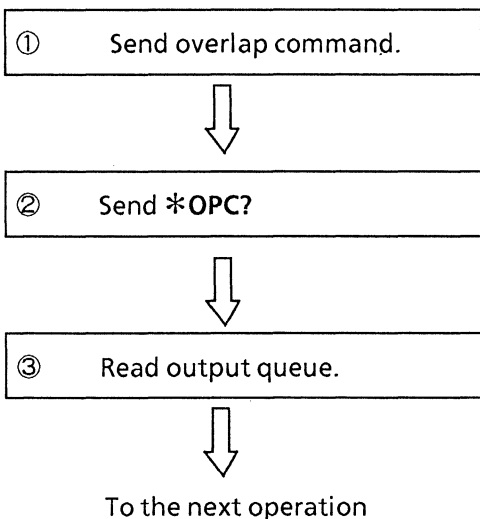
Executing the *OPC? query sets a 1 in the output queue to generate a MAV summary message when a device has completed all of its pending operations.

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>



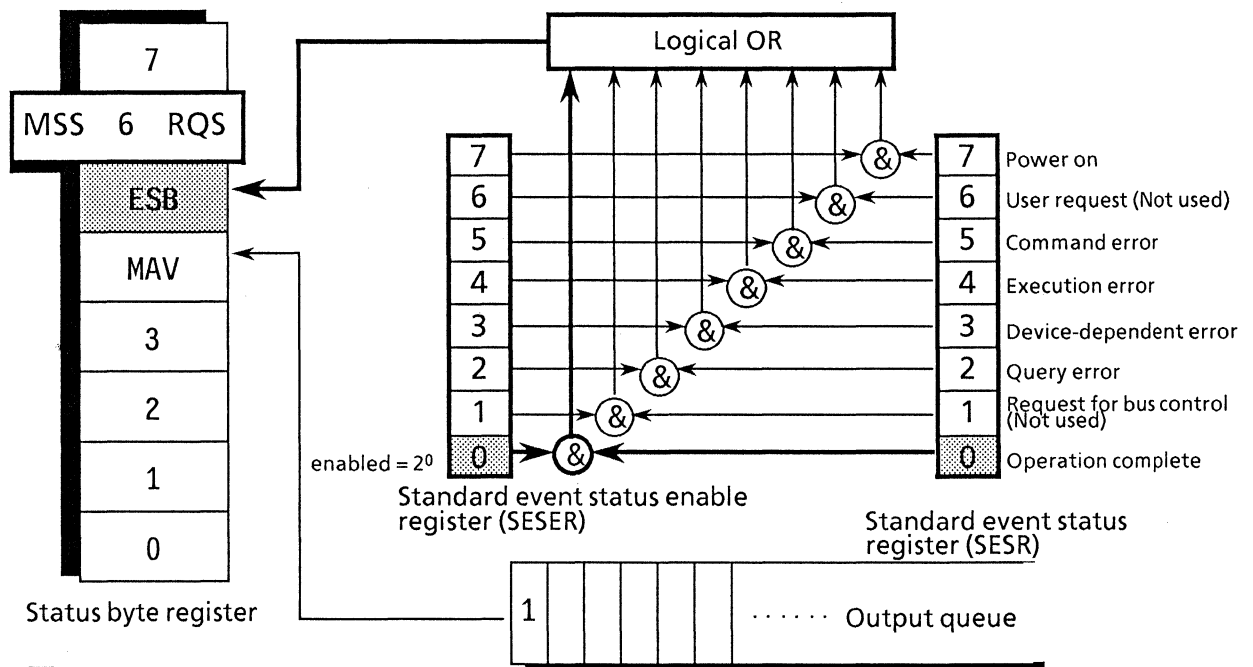
The next *OPC? is used to confirm the completion of the final command. For this reason, it is normal to send *OPC? after executing an overlap command. However, even for sequential commands, if it is necessary to confirm the end of execution of the final command from the execution sequence or execution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

The 1 which is read is ignored, and it goes on 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.

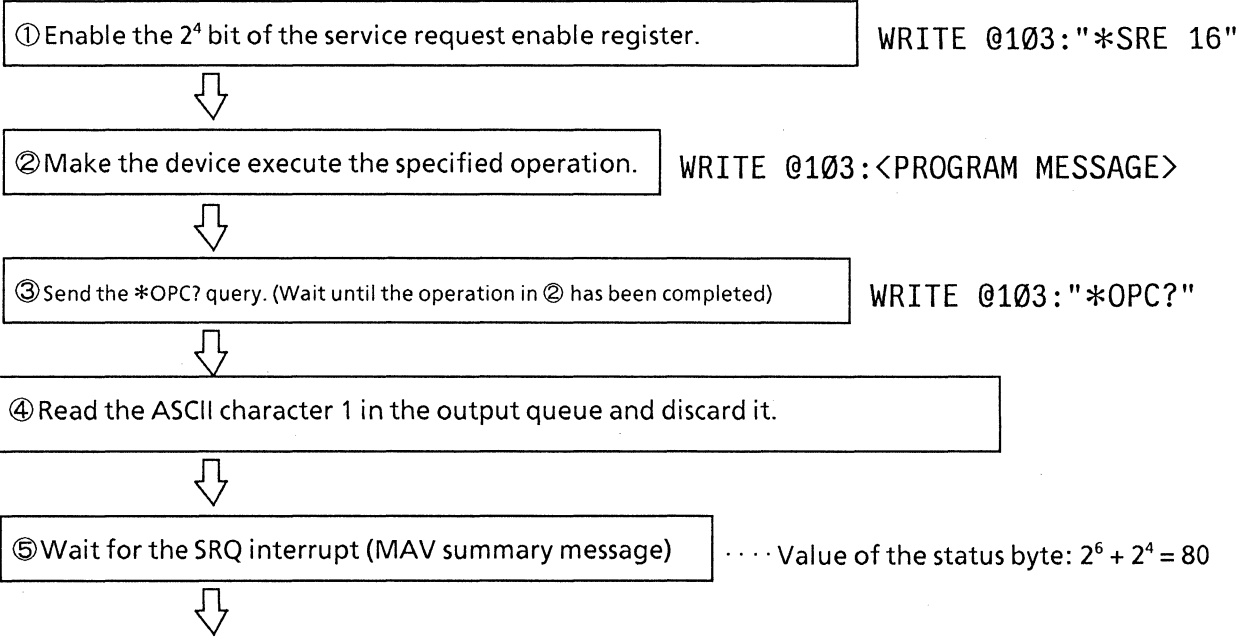


■ <Using the **OPC* command>

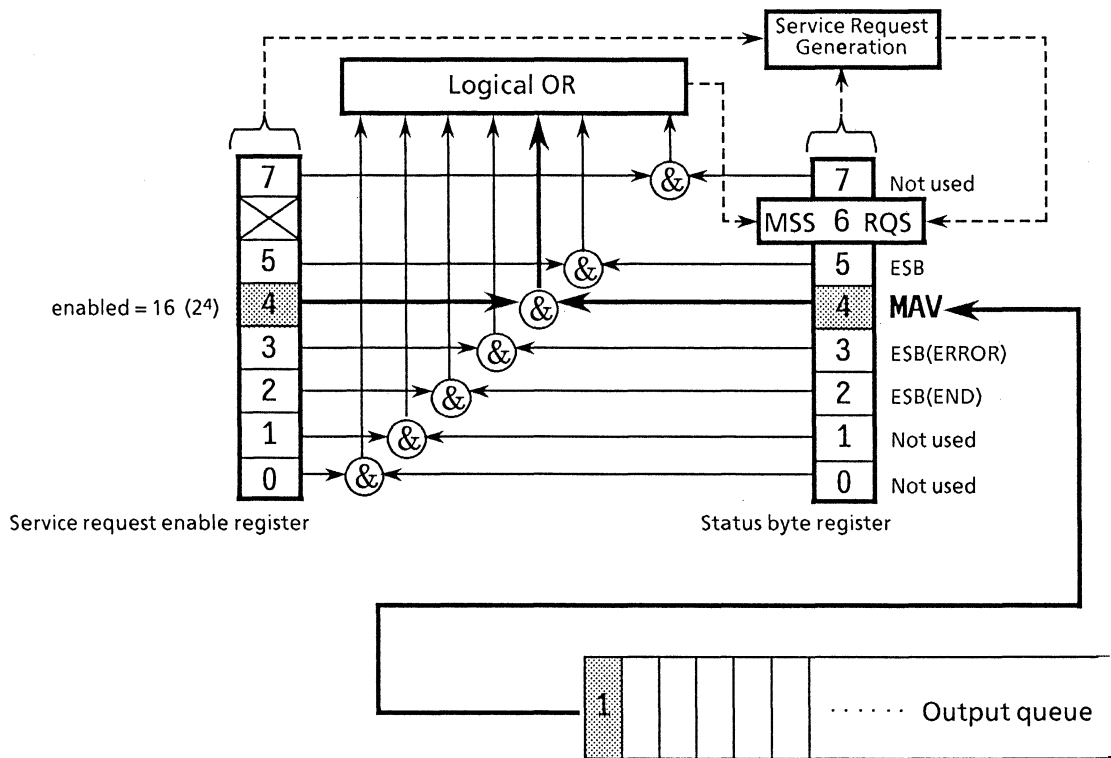
- ① Enable the 2^0 bit of the standard event status enable register. WRITE @103:"*ESE 1"
- ↓
- ② Enable the 2^5 bit of the service request enable register. WRITE @103:"*SRE 32"
- ↓
- ③ Make the device execute the specified operation. WRITE @103:<PROGRAM MESSAGE>
- ↓
- ④ Execute the **OPC* command. (Since it is an overlap command, the next command is also executed) WRITE @103:"*OPC"
- ↓
- ⑤ Wait for an SRQ interrupt. (ESB summary message) ... Value of the status byte: $2^6 + 2^5 = 96$

SECTION 8 STATUS STRUCTURE

■ <Using the *OPC? query>



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

9.1	Table of Device Messages	9-3
9.1.1	Table of Device Messages (in the Alphabetic order)	9-3
9.1.2	Device Message (Panel Correspondence)	9-7
9.1.3	Detailed Explanation of Device Messages	9-18

(Blank)

This section explains each device message by group. Each group is corresponded to the front and rear panel of MP1763B. 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 MP1763B 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 Messages (Alphabetic order)

Function	Control Message		Data Request Message	Device Message Details	
	Header	Numeric Data	Header	Section	Page
Page number	ADR	NR1 format	ADR?	PATTERN	P9-47
Pattern data preset (All page, All bits)	ALL	NR1 format	—	PATTERN	P9-51
Alternate pattern A / B display switch	ALT	NR1 format	ALT?	PATTERN	P9-41
Alternate pattern A / B switch signal selection	APS	NR1 format	APS?	Other	P9-83
Pattern bit	BIT	NR1 format HEX format	BIT?	PATTERN	P9-49
Clock 1 output amplitude	CAP	NR2 format	CAP?	OUTPUT	P9-71
Clock 1 output Delay Time	CDL	NR2 format	CDL?	OUTPUT	P9-70
Clock 1 output termination voltage	CTM	NR1 format	CTM?	OUTPUT	P9-58
Clock 1 output offset	COS	NR2 format	COS?	OUTPUT	P9-72
Data output amplitude	DAP	NR2 format	DAP?	OUTPUT	P9-60
Data / $\overline{\text{Data}}$ display switch	DDS	NR1 format	DDS?	OUTPUT	P9-75
FD data delete	DEL	NR1 format	—	MEMORY	P9-26
Data length	DLN	NR1 format	DLN?	PATTERN	P9-44
Delay status	—	—	DLY?	Other	P9-90
Data output offset	DOS	NR2 format	DOS?	OUTPUT	P9-63
Data output termination voltage	DTM	NR1 format	DTM?	OUTPUT	P9-57
Error insertion	EAD	NR1 format	EAD?	PATTERN	P9-42
Error insertion channel	ECH	NR1 format	ECH?	Other	P9-80
External error insertion	EEI	NR1 format	EEI?	Other	P9-82
FD Format	FDF	—	—	MEMORY	P9-30
File No. / Directory mode switch	FIL	NR1 format	FIL?	MEMORY	P9-24
FD / Error message	—	—	FDE?	MEMORY	P9-35
Memory FD mode	—	—	FMD?	MEMORY	P9-33
Internal clock frequency	FRQ	NR1 format	FRQ?	INTERNAL CLOCK	P9-21
File contents search	—	—	FSH?	MEMORY	P9-31

Table 9-1 Table of Device Messages (Alphabetic order: continued)

Function	Control Message		Data Request Message	Device Message Details	
	Header	Numeric Data	Header	Section	Page
Initialization	INI	—	—	Other	P9-84
Pattern logic	LGC	NR1 format	LGC?	PATTERN	P9-37
Alternate A / B loop times	LPT	NR1 format	LPT?	PATTERN	P9-43
FD access status	—	—	MAC?	MEMORY	P9-34
Memory mode switch	MEM	NR1 format	MEM?	MEMORY	P9-29
PRBS mark ratio	MRK	NR1 format	MRK?	PATTERN	P9-40
Data output amplitude	NAP	NR2 format	NAP?	OUTPUT	P9-62
Data output offset	NOS	NR2 format	NOS?	OUTPUT	P9-68
Offset reference value	OFS	NR1 format	OFS?	OUTPUT	P9-59
Output ON / OFF	OON	NR1 format	OON?	OUTPUT	P9-74
Page number	PAG	NR1 format	PAG?	PATTERN	P9-47
PLL lock status	—	—	PLL?	Other	P9-87
Page number / pattern sync trigger position display switch	PPD	NR1 format	PPD?	PATTERN	P9-55
Pattern sync trigger position	PSP	NR2 format	PSP?	PATTERN	P9-53
Pattern data preset (1 page, All bits)	PST	NR1 format	—	PATTERN	P9-52
ZERO SUBST / PRBS stage	PTN	NR1 format	PTN?	PATTERN	P9-39
Generated pattern selection	PTS	NR1 format	PTS?	PATTERN	P9-38
Power cut, recoverys status	—	—	PWI?	Other	P9-89
FD data recall	RCL	NR1 format	—	MEMORY	P9-25
Pattern data output byte number	RED	NR1 format	—	Other	P9-86
Internal clock resolution switching	RES	NR1 format	RES?	INTERNAL CLOCK	P9-22
FD data resave	RSV	NR1 format	—	MEMORY	P9-28
Internal timer setting	RTM	NR1 format	RTM?	Other	P9-88
FD data save	SAV	NR1 format	—	MEMORY	P9-27
Mark ratio AND bit shift number	SFT	NR1 format	SFT?	Other	P9-81
Sync signal output selection	SOP	NR1 format	SOP?	Other	P9-79

Table 9-1 Table of Device Messages (Alphabetic order: continued)

Function	Control Message		Data Request Message	Device Message Details	
	Header	Numeric Data	Header	Section	Page
1 / 1 SPEED, 1 / 4 SPEED display switch	SPD	NR1 format	SPD?	OUTPUT	P9-77
DATA / $\overline{\text{DATA}}$ tracking	TRK	NR1 format	TRK?	OUTPUT	P9-76
Pattern data input byte number	WRT	NR1 format	—	Other	P9-85
ZERO SUBST length	ZLN	NR1 format	ZLN?	PATTERN	P9-46

9.1.2 Device Messages (Panel correspondence)

Figures 9-2 (1) to (6) and Table 9-2 (1) to (5) show the correspondence of control messages and data arequest messages to the panel keys.

● INTERNAL CLOCK section

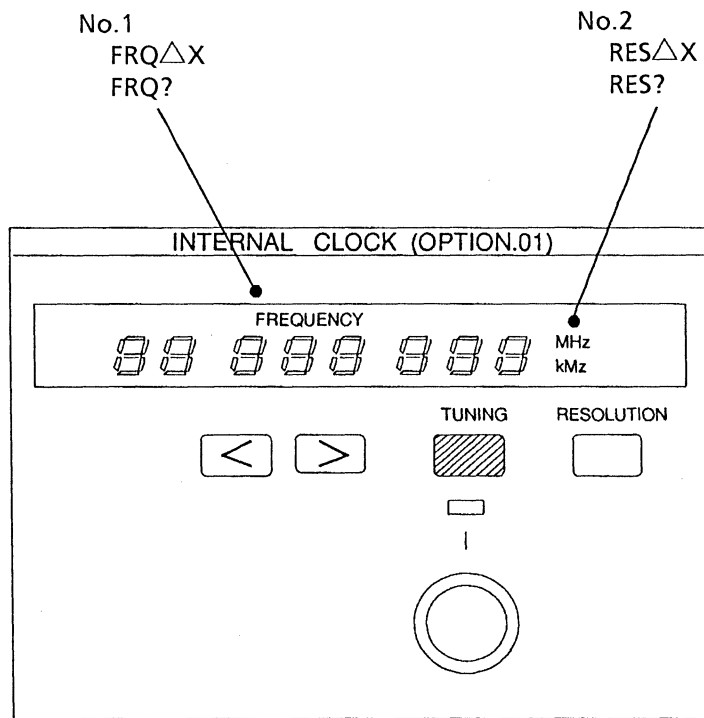
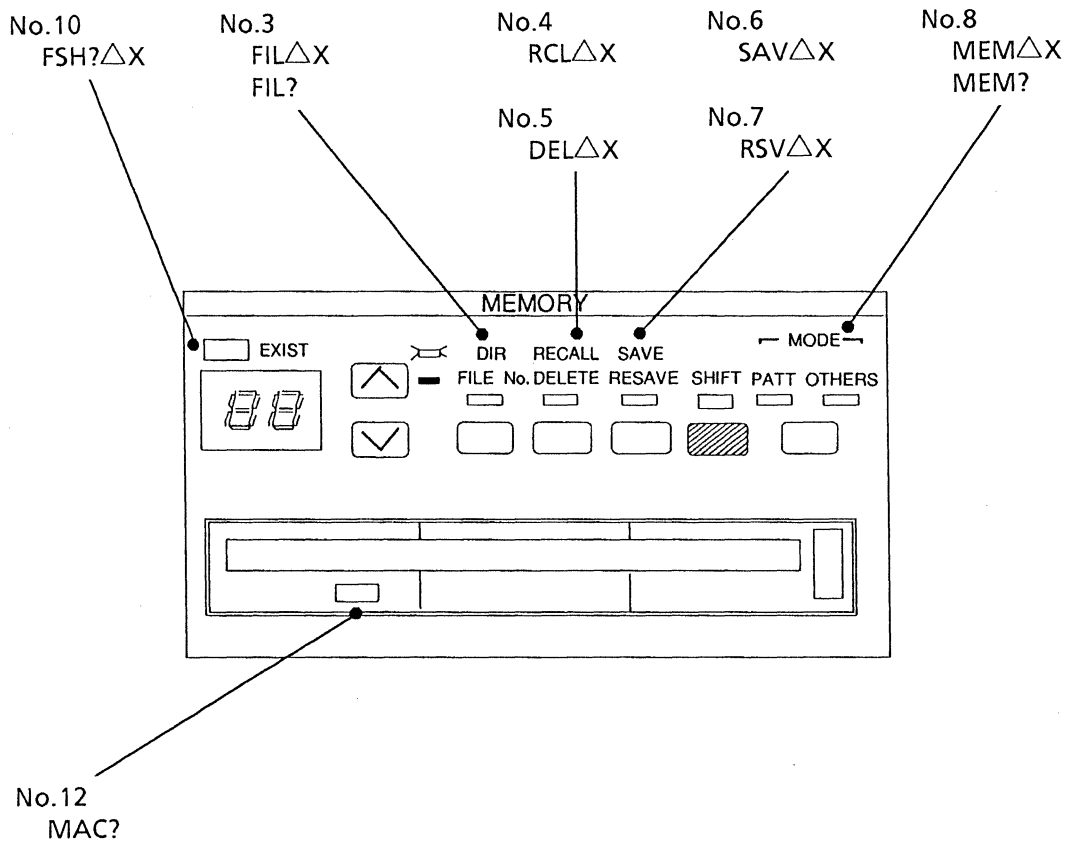


Fig. 9-2-(1) INTERNAL CLOCK Section

Table 9-2-(1) List of Device Messages (INTERNAL CLOCK Section)

Function	Control Message		Data Request Message	Device Message Details	
	Header	Numeric Data	Header	Item No.	Page
• INTERNAL CLOCK section					
Internal clock frequency	FRQ	NR1 format	FRQ?	1	P9-20
Internal clock resolution switching	RES	NR1 format	RES?	2	P9-22

● MEMORY Section



- FD mode : No. 11 FMD?
- FD error message : No. 13 FDE?
- FD format : No. 9 FDF

Fig. 9-2-(2) MEMORY Section

Table 9-2-(2) List of Device Messages (MEMORY Section)

Function	Control Message		Data Request Message	Device Message Details	
	Header	Numeric Data	Header	Item No.	Page
● MEMORY section					
File No. / Directory mode switching	FIL	NR1 format	FIL?	3	P9-24
FD data recall	RCL	NR1 format	—	4	P9-25
FD data delete	DEL	NR1 format	—	5	P9-26
FD data save	SAV	NR1 format	—	6	P9-27
FD data resave	RSV	NR1 format	—	7	P9-28
Memory mode switch	MEM	NR1 format	MEM?	8	P9-29
FD format	FDF	—	—	9	P9-30
File contents search	—	—	FSH?	10	P9-31
Memory FD mode	—	—	FMD?	11	P9-33
FD access status	—	—	MAC?	12	P9-34
FD error message	—	—	FDE?	13	P9-35

● PATTERN Section

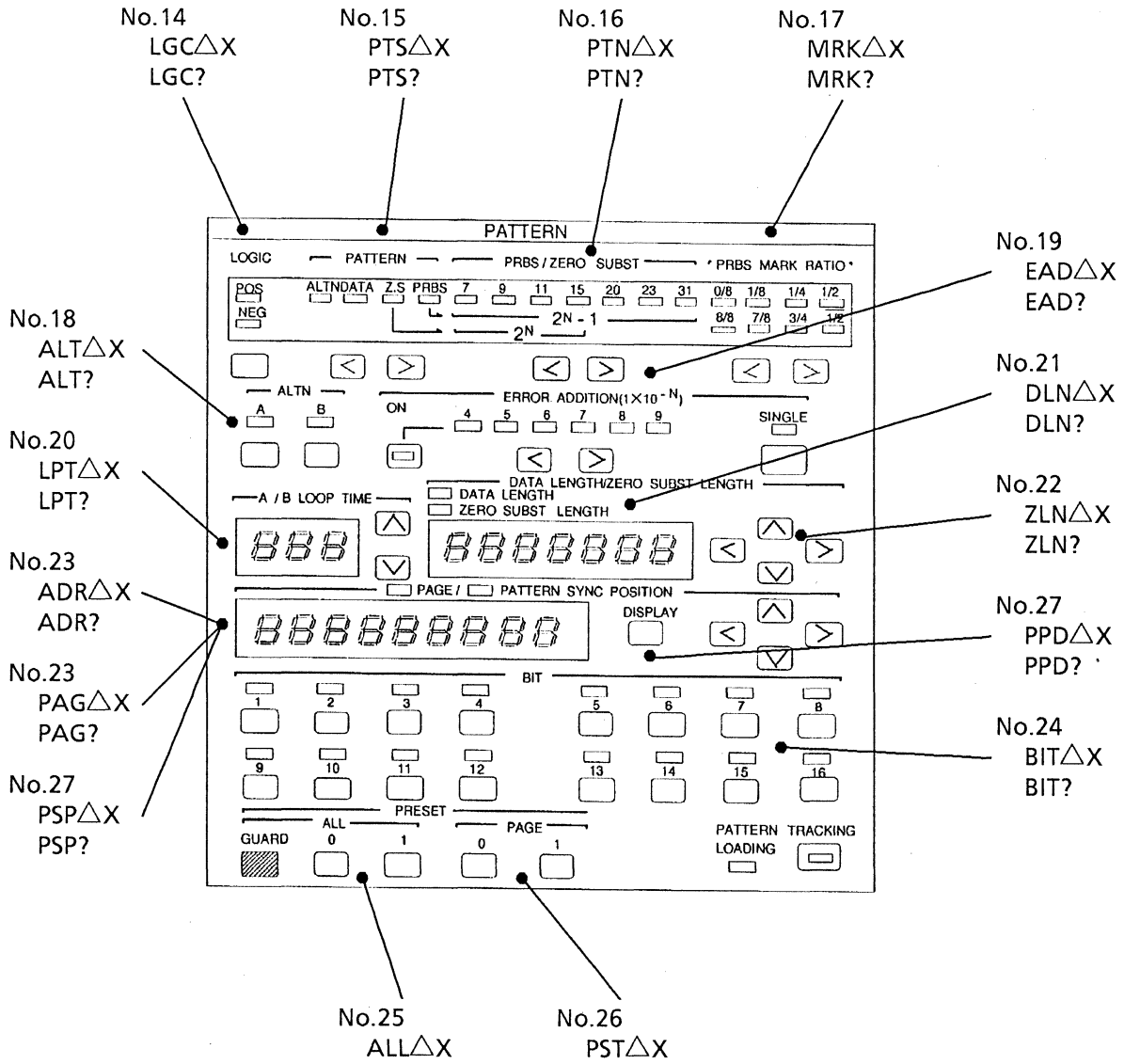


Fig. 9-2-(3) PATTERN Section

Table 9-2-(3) List of Device Messages (PATTERN Section)

Function	Control Message		Data Request Message	Device Message Details	
	Header	Numeric Data	Header	Item No.	Page
● PATTERN section					
Pattern logic	LGC	NR1 format	LGC?	14	P9-37
Generation pattern switch	PTS	NR1 format	PTS?	15	P9-38
ZERO SUBST / PRBS stage	PTN	NR1 format	PTN?	16	P9-39
PRBS mark ratio	MRK	NR1 format	MRK?	17	P9-40
Alternate pattern A / B display switch	ALT	NR1 format	ALT?	18	P9-41
Error insertion	EAD	NR1 format	EAD?	19	P9-42
Alternate A / B loop times	LPT	NR1 format	LPT?	20	P9-43
Data length	DLN	NR1 format	DLN?	21	P9-44
ZERO SUBST length	ZLN	NR1 format	ZLN?	22	P9-46
Page number	PAG ADR	NR1 format NR1 format	PAG? ADR?	23	P9-47
Pattern bit	BIT	NR1 format HEX format	BIT?	24	P9-49
Pattern data preset (All pages, All bits)	ALL	NR1 format	–	25	P9-51
Pattern data preset (1 page, All bits)	PST	NR1 format	–	26	P9-52
Pattern sync trigger position	PSP	NR2 format	PSP?	27	P9-53
Page No. / Pattern sync trigger position display switch	PPD	NR1 format	PPD?	28	P9-55

● OUTPUT Section

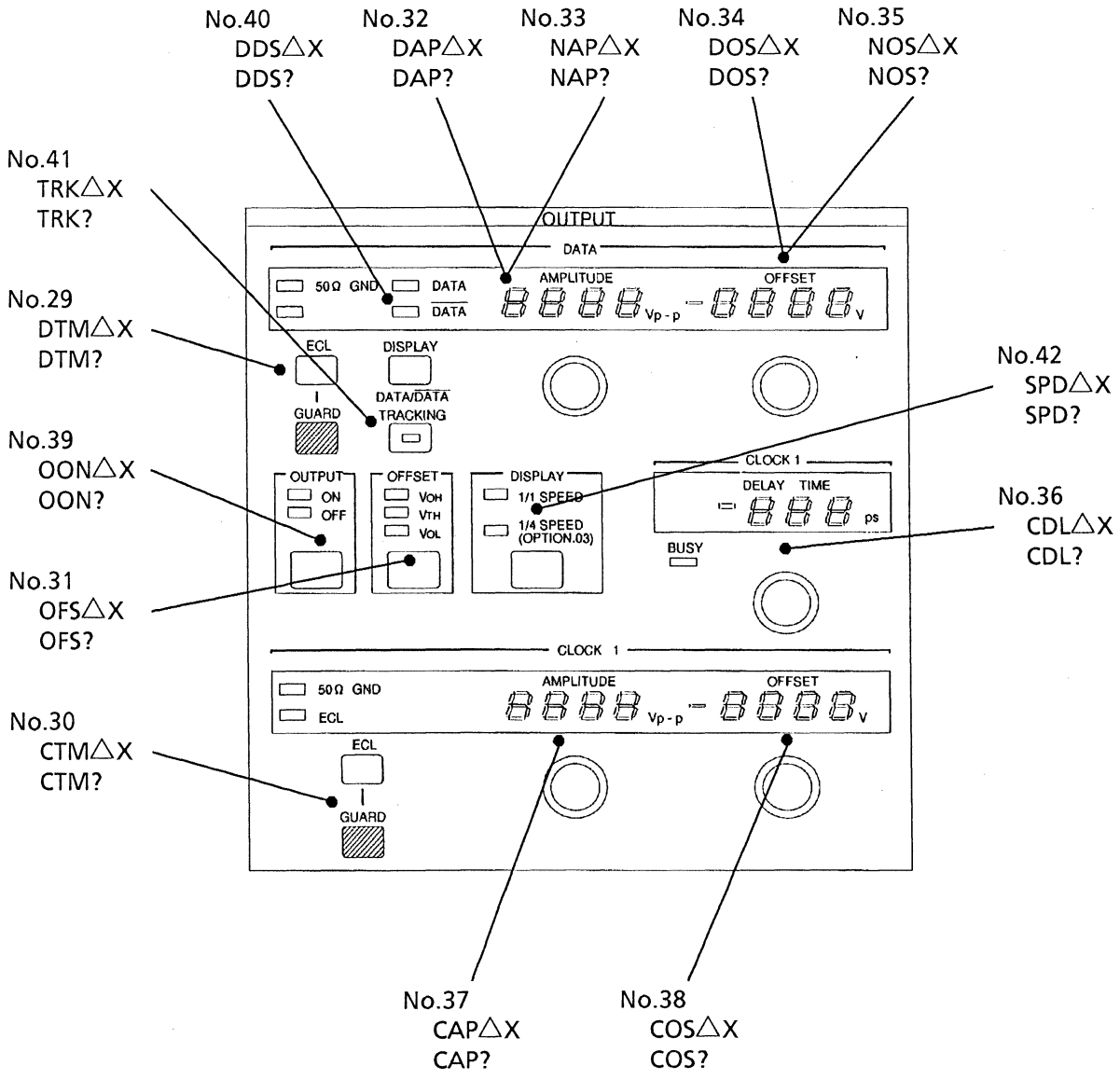


Fig. 9-2-(4) OUTPUT Section

Table 9-2-(4) List of Device Messages (OUTPUT Section)

Function	Control Message		Data Request Message	Device Message Details	
	Header	Numeric Data	Header	Item No.	Page
• OUTPUT section					
Data output termination voltage	DTM	NR1 format	DTM?	29	P9-57
Clock1 output termination voltage	CTM	NR1 format	CTM?	30	P9-58
Offset reference value	OFS	NR1 format	OFS?	31	P9-59
Data output amplitude	DAP	NR2 format	DAP?	32	P9-60
$\overline{\text{Data}}$ output amplitude	NAP	NR2 format	NAP?	33	P9-61
Data output offset	DOS	NR2 format	DOS?	34	P9-62
$\overline{\text{Data}}$ output offset	NOS	NR2 format	NOS?	35	P9-68
Clock1 output delay time	CDL	NR2 format	CDL?	36	P9-70
Clock1 output amplitude	CAP	NR2 format	CAP?	37	P9-71
Clock1 output offset	COS	NR2 format	COS?	38	P9-72
Output ON / OFF	OON	NR1 format	OON?	39	P9-74
DATA / $\overline{\text{DATA}}$ display switch	DDS	NR1 format	DDS?	40	P9-75
DATA / $\overline{\text{DATA}}$ / tracking	TRK	NR1 format	TRK?	41	P9-76
1 / 1 SPEED, 1 / 4 SPEED display switch	SPD	NR1 format	SPD?	42	P9-77

● Other sections

(Front panel)

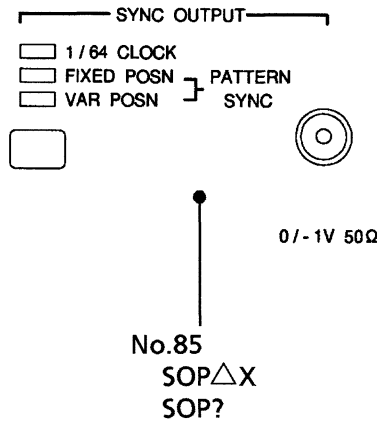


Fig. 9-2-(5) Other Sections (Front Panel)

(Rear panel)

(Function Switch)

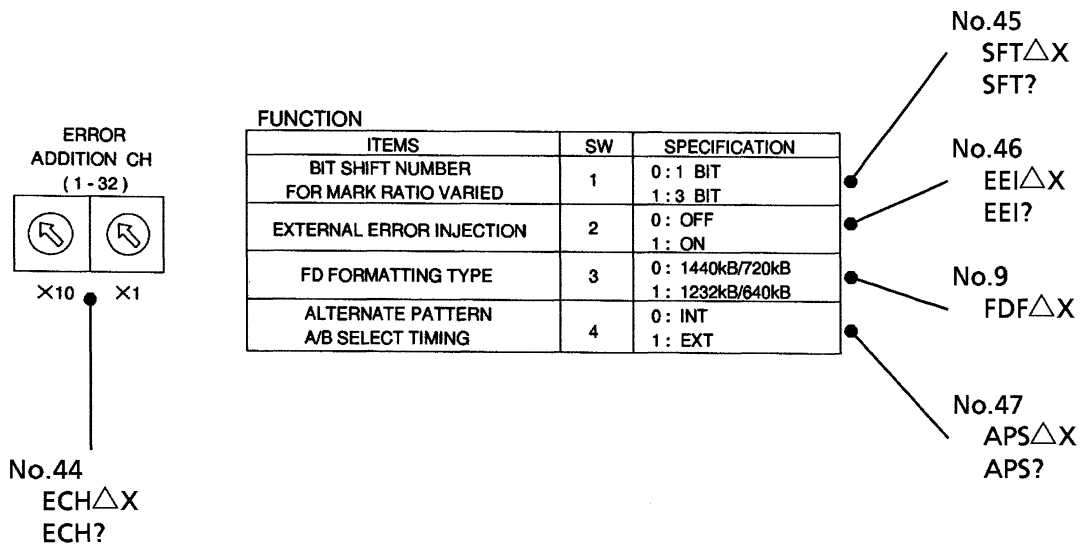


Fig. 9-2-(6) Other Sections (Back Panel / Function Switch)

Note

The back-panel function switches have the following functions when the value set at the REMOTE status and the values of the back-panel switches are different.

- The value set at REMOTE is saved when the power to the PPG is cut in the REMOTE status. (However, when the Initialize command INI or *RST have been sent, the settings are initialized.) Additionally, if the PPG is in the LOCAL status, the setting status of the back-panel function switches takes priority and the setting contents in the REMOTE status are disabled.

Table 9-2-(5) List of Device Messages (Other Sections)

Function	Control Message		Data Request Message	Device Message Details	
	Header	Numeric Data	Header	Item No.	Page
● Front panel Sync signal output selection	SOP	NR1 format	SOP?	43	P9-79
● Back panel Error insertion channel	ECH	NR1 format	ECH?	44	P9-80
● Function switch Mark ratio AND bit shift number	SFT	NR1 format	SFT?	45	P9-81
External error insertion	EEI	NR1 format	EEI?	46	P9-82
Alternate pattern A / B switch signal selection	APS	NR1 format	APS?	47	P9-83
● Other Initialize	INI	—	—	48	P9-84
Pattern data input byte number	WRT	NR1 format	—	49	P9-85
Pattern data output byte number	RED	NR1 format	—	50	P9-86
Internal synthesizer PLL	—	—	PLL?	51	P9-87
Internal timer setting	RTM	NR1 format	RTM?	52	P9-88
Power cut, recovery status	—	—	PWI?	52	P9-89
Delay status	—	—	DLY?	54	P9-90

9.1.3 Detailed Explanation of Device Messages

MP1763B control messages and data request messages are explained in this section.

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

- INTERNAL CLOCK Section

The following pages show each control messages for the INTERNAL CLOCK section.

The Δ in the strings indicates a space.

1) **FRQ** Internal Clock Frequency (**FReQuency**)

■ **Function** Sets internal clock frequency

Header	Program	Query	Response	(Character No.)
FRQ	FRQ△m	FRQ?	FRQ△m	(kHz units: FIX8) (MHz units: FIX5)

■ **Value of m** When internal clock frequency setting resolution in kHz
 Min. value: 50000
 Max. value: 12500000
 Step: 1

The response is as follows:
 FRQ△△△△50000 (Min. value)
 :
 FRQ△12500000 (Max. value)

When internal clock frequency setting resolution in MHz
 Min. value: 50
 Max. value: 12500
 Step: 1

The response is as follows:
 FRQ△△△△50 (Min. value)
 :
 FRQ△12500 (Max. value)

■ **Command Type** Sequential command

■ **Usage Restrictions** The command is invalid under the following setting conditions.
 Program: When Option 01 (internal synthesizer) not installed
 When the FD is being accessed

Query: The command is invalid under the following setting condition
 and ERR (CR / LF) is output
 When Option 01 not installed

1) FRQ

Internal Clock Frequency (FReQuency) continued

■ Usage Example

Program: When resolution is MHz and internal clock frequency is 500 MHz

```
OUTPUT△700;"FRQ△500"
```

When resolution is kHz and internal clock frequency is 500 MHz

```
OUTPUT△700;"FRQ△500000"
```

Query: When frequency set to 1 GHz

```
OUTPUT△700;"FRQ?"
```

```
ENTER△700;BS
```

```
PRINT B$
```

↓

When frequency resolution is MHz units

```
FRQ△△△△1000 (CR / LF)
```

When frequency resolution is kHz units

```
FRQ△△100000 (CR / LF)
```

■ Note

When an external clock is input when Option 01 is installed, the settings shown above can be set, but there is no change in the actual output. In addition, the display value is output in response to the query.

2) RES Internal Clock Resolution Switching (RESolution)

■ **Function** Sets resolution of internal clock frequency

Header	Program	Query	Response (Character No.)
RES	RES△m	RES?	RES△m (FIX 1)

■ **Value of m**
 ∅ : kHz units
 1 : MHz units

■ **Command Type**

■ **Usage Restrictions** The command is invalid under the following setting conditions.

Program: When Option 01 (internal synthesizer) is not installed
 When the FD is being accessed

Query: The command is invalid under the following setting condition
 and ERR (CR / LF) is output
 When Option 01 not installed

■ **Usage Example**

Program: When internal clock frequency resolution is kHz units
 OUTPUT△700;"RES△∅"

Query: When internal clock frequency resolution is MHz units
 OUTPUT△700;"RES?"
 ENTER△700;B\$
 PRINT B \$
 ↓
 RES△1 (CR/LF)

■ **Note** When an external clock is input when Option 01 is installed, the settings shown above can be set, but there is no change in the actual output. In addition, the display value is output in response to the query.

- MEMORY section

The following pages show each control messages for the MEMORY section.

The \triangle in the strings indicates a space.

4) RCL FD data recall (ReCaLI)

- **Function** Sets FD contents at PPG

Header	Program	Query	Response	(Character No.)
RCL	RCL△m	None	None	

- **Value of m** Sets file name in range 0 to 99
 Numeric range: Max. value 99
 Min. Value 0
 Step 1
- **Command Type** Overwrap command
- **Usage Restrictions** The command is invalid under the following setting conditions.
 Program: When the FD is being accessed
- **Usage Example** Program: When setting file contents of file No. 9
 OUTPUT△700;"RCL△9"
- **Note** If the specified file does not exist, an error is returned and the error information is displayed at the MEMORY display.
 This error display is cleared when the settings below are made.

File No. / Directory mode switch	(No. 3)
FD data recall	(No. 4)
FD data delete	(No. 5)
FD data save	(No. 6)
FD data resave	(No. 7)
Memory mode switch	(No. 8)
FD format	(No. 9)

In addition, the FD abnormal occurrence bit of the extended event status register ESR2 (ERROR) bit is set.

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

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

However, in the PATT mode, when there are files where the RR** .PTN and TT** .PTN parts are identical, TT** .PTN takes priority at reading.

5) DEL FD data delete (**DE**lete)

■ **Function** Deletes specified file from FD

Header	Program	Query	Response (Character No.)
DEL	DEL△m	None	None

■ **Value of m** Sets file name in range 0 to 99

Numeric range: Max. value 99
Min. Value 0
Step 1

■ **Command Type** Overwrap command

■ **Usage Restrictions** The command is invalid under the following setting conditions.

Program: When the FD is being accessed

■ **Usage Example** Program: When erasing file contents of file No. 9
OUTPUT△700;"DEL△9"

■ **Note** If the specified file does not exist, an error is returned and the error information is displayed at the MEMORY display.

This error display is cleared when the settings below are made.

File No. / Directory mode switch	(No. 3)
FD data recall	(No. 4)
FD data delete	(No. 5)
FD data save	(No. 6)
FD data resave	(No. 7)
Memory mode switch	(No. 8)
FD format	(No. 9)

In addition, the FD abnormal occurrence bit of the extended event status register ESR2 (ERROR) bit1 is set.

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

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

6) SAV FD data save (**SAVe**)

- **Function** Saves set contents of PPG to FD

Header	Program	Query	Response	(Character No.)
SAV	SAV△m	None	None	

- **Value of m** Sets file name in range 0 to 99
 - Numeric range: Max. value 99
 - Min. Value 0
 - Step 1
- **Command Type** Overwrap command
- **Usage Restrictions** The command is invalid under the following setting conditions.
 - Program: When the FD is being accessed
- **Usage Example** Program: When saving file contents of file No. 9 to PPG
 OUTPUT△700;"SAV△9"
- **Note** If the specified file does not exist, an error is returned and the error information is displayed at the MEMORY display.
 This error display is cleared when the settings below are made.
 - File No. / Directory mode switch (No. 3)
 - FD data recall (No. 4)
 - FD data delete (No. 5)
 - FD data save (No. 6)
 - FD data resave (No. 7)
 - Memory mode switch (No. 8)
 - FD format (No. 9)
 In addition, the FD abnormal occurrence bit of the extended event status register ESR2 (ERROR) bit1 is set.
 The following files are saved according to the memory mode setting status.
 - PATT mode: TT**.*PTN
 - OTHERS mode: TT**.*OTH

7) RSV FD data resave (ReSaVe)

■ **Function** Sets contents of FD to PPG

Header	Program	Query	Response	(Character No.)
RSV	RSV△m	None	None	

■ **Value of m** Sets file name in range 0 to 99

Numeric range: Max. value 99
 Min. Value 0
 Step 1

■ **Command Type** Overwrap command

■ **Usage Restrictions** The command is invalid under the following setting conditions.

Program: When the FD is being accessed

■ **Usage Example** Program: When overwriting file contents of file No. 9 to PPG
 OUTPUT△700;"RSV△9"

■ **Note** If the specified file does not exist, an error is returned and the error information is displayed at the MEMORY display.

This error display is cleared when the settings below are made.

File No. / Directory mode switch (No. 3)
 FD data recall (No. 4)
 FD data delete (No. 5)
 FD data save (No. 6)
 FD data resave (No. 7)
 Memory mode switch (No. 8)
 FD format (No. 9)

In addition, the FD abnormal occurrence bit of the extended event status register ESR2 (ERROR) bit1 is set.

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

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

8) MEM Memory mode switch (MEMemory mode)

- **Function** Performs PATTERN / OTHERS switch setting

Header	Program	Query	Response	(Character No.)
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 under the following setting conditions.
 - Program: When the FD is being accessed
 - Query: None
- **Usage Example**
 - Program: When setting memory mode to PATT
OUTPUT△7ØØ;"MEM△Ø"
 - Query: When setting memory mode to OTHERS
OUTPUT△7ØØ;"MEM?"
ENTER△7ØØ;B\$
PRINT△B\$
↓
MEM△1 (CR/LF) output
- **Note**
 - PATT mode means PATTERN mode.
 - In this case, the memory contents are the target of the PATTERN section contents.
 - In the OTHERS mode, the contents of other sections are the memory target.
 - If an error occurs during file access, etc., the error information is displayed at the MEMORY display.
 - This error display is cleared when the settings below are made.

File No. / Directory mode switch	(No. 3)
FD data recall	(No. 4)
FD data delete	(No. 5)
FD data save	(No. 6)
FD data resave	(No. 7)
Memory mode switch	(No. 8)
FD format	(No. 9)
 - In addition, the FD abnormal occurrence bit of the extended event status register ESR2 (ERROR) bit1 is set.
 - The following files are resaved according to the memory mode setting status.
 - PATT mode: RR** .PTN or TT** .PTN
 - OTHERS mode: TT** .OTH

9) FDF **FD Format (FD Format)**

■ **Function** Formats floppy disk
 Select the format type using FUNCTION SW3 on the rear panel. In addition, the 2DD / 2HD format is determined automatically from the inserted floppy disk.

Header	Program	Query	Response (Character No.)
FDF	FDF	None	None

- **Value of m** None
- **Command Type** Overwrap command
- **Usage Limitations** The command is invalid under the following setting conditions.
 Program: When the FD is being accessed and when the File No. / Directory mode switch is the directory mode.
- **Usage Example** Program: When formatting floppy disk
 OUTPUT△700;"FDF"
- **Note** When an error is generated during file access, the error information is displayed at the MEMORY display.
 This error display is cleared when the settings below are made.
 - File No. / Directory mode switch (No. 3)
 - FD data recall (No. 4)
 - FD data delete (No. 5)
 - FD data save (No. 6)
 - FD data resave (No. 7)
 - Memory mode switch (No. 8)
 - FD format (No. 9)
 In addition, the FD abnormal occurrence bit of the extended event status register ESR2 (ERROR) bit1 is set.

10) FSH?**Search File Contents (File Search)****■ Function**

Outputs information about data saved on the floppy disk

The target file names are as follows:

TT**.*PTN

RR**.*PTN

TT**.*OTH

Header	Program	Query	Response (Character No.)
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 : Selects preceding and succeeding half File No.

0 : Preceding half (No. 0 to No. 49)

1 : Succeeding half (No. 50 to No. 99)

m2 : Unused size

m3 : Used size

m4 : File No.

m5 : Preceding half or succeeding half File No. only target file)

■ Command Type

Sequential command

■ Usage Restrictions

The command is invalid under the following setting conditions.

Query: None

■ Usage Example

Query: When File Nos. 1 to 10 exist on floppy disk (Unused size or Used size are one 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 there are no target files on the floppy disk

OUTPUT△ 700; "FSH?△0"

ENTER△ 700;B\$

PRINT△B\$

↓

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

10) FSH?

■ Note

Search File Contents (File SeaRch) continued

The file contents search is executed according to the memory mode switch (PATT / OTHERS) setting conditions.

(1) When memory mode switch is PATT

The files TT**.*.PTN and RR**.*.PTN are searched.

(2) When memory mode switch is OTHERS

The file TT**.*.OTH is searched.

When a PATT file saved to FD on the MP1763B and a PATT file saved to FD on the MP1764A have the same name, the file name saved on the MP1763B takes priority and is output.

When searching the file contents, the information about the immediately preceding information is output.

In addition, since this equipment does not have a function for detecting an inserted floppy disk, the previous directory information is not updated when the floppy disk is changed.

As a result, when inserting or changing a floppy disk, always switch to the directory mode and update the directory information.

11) FMD?**Memory FD mode (memory Fd MoDe)**■ **Function**

Outputs floppy disk format type

Header	Program	Query	Response	(Character No.)
FMD	None	FMD?	FMD△m	(FIX 1)

■ **Value of m**

0 : 1440 k
 1 : 720 k
 2 : 1232 k
 3 : 640 k

■ **Command Type**

Sequential command

■ **Usage Restrictions**

The command is invalid under the following setting conditions.

Query: None

■ **Usage Example**

Query: When formatting inserted 2DD FD in 1440 kB / 720 kB
 format
 OUTPUT△700;"FMD?"
 ENTER△700;B\$
 PRINT△B\$
 ↓
 FMD△1 (CR/LF) output

■ **Note**

Use FUNCTION SW3 on the back panel to select the MS-DOS format. Since this setting is read at power-on, the setting is not updated until the next power-on.

● **Function SW3 = OFF**

Format Capacity	Sector Length [Byte / Sector]	Sector No. [Sectors / Track]	Track No. [Tracks / Side]
1440 kB	512	18	80
720 kB	512	9	80

● **Function SW3 = ON**

Format Capacity	Sector Length [Byte / Sector]	Sector No. [Sectors / Track]	Track No. [Tracks / Side]
1232 kB	1024	8	77
640 kB	512	8	80

When no floppy disk is inserted when this query is executed, the 1440 kB or 1232 kB information is output in accordance with the currently-set FDD condition (FUNCTION SW3).

13) FDE?**FD Error message (FD Error message)**■ **Function**

Outputs FD error message

Header	Program	Query	Response	(Character No.)
FDE	None	FDE?	FDE△m	(FIX 2)

■ **Value of m**

Error messages

- ∅ : E0 (Error due to different format type)
- 1 : E1 (Write protected at overwrite)
- 2 : E2 (Insufficient space at overwrite)
- 3 : E3 (No specified file at read)
- 4 : E4 (Attempt to save as same file name)
- 5 : E5 (Overwrite error)
- 6 : E6 (Read error)
- 7 : E7 (DMA send error)
- 8 : E8 (Other error)
- 9 : E9 (Hardware error)
- 1∅ : No error

■ **Command Type**

Sequential command

■ **Usage Restrictions**

The command is invalid under the following setting conditions.

Query: None

■ **Usage Example**

Query: When hardware error occurs
 OUTPUT△7∅∅;"FDE?"
 ENTER△7∅∅;B\$
 PRINT△B\$

↓

FDE△9 (CR/LF) output

■ **Note**

This error display is cleared when the settings below are made.

- File No. / Directory mode switch (No. 3)
- FD data recall (No. 4)
- FD data delete (No. 5)
- FD data save (No. 6)
- FD data resave (No. 7)
- Memory mode switch (No. 8)
- FD format (No. 9)

In addition, the FD abnormal occurrence bit of the extended event status register ESR2 (ERROR) bit1 is set.

SECTION 9 DETAILS OF DEVICE MESSAGES

- PATTERN Section

The following pages show each control messages for the PATTERN section.

The Δ in the strings indicates a space.

14) LGC**Pattern Logic (LoGiC mode)**■ **Function**

Sets data logic

The relationship with the logic and the actual data output are different between ALTERNATE / DATA / ZERO SUBST and PRBS. (Refer to the functions and operation manual.)

Header	Program	Query	Response (Character No.)
LGC	LGC△m	LGC?	LGC△m (FIX 1)

■ **Value of m**

∅ : Positive

1 : Negative

■ **Command Type**

Sequential command

■ **Usage Restrictions**

The command is invalid under the following setting conditions.

Program: When the FD is being accessed

Query: None

■ **Usage Example**

Program: When setting pattern logic to positive logic (Positive)
OUTPUT△700;"LGC△∅"

Query: When pattern logic set to negative logic (Negative)
OUTPUT△700;"LGC?"
ENTER△700;B\$
PRINT△B\$

↓

LGC△1 (CR/LF) output

■ **Note**

When the pattern is PRBS mode, the pattern mark ratio is also switched according to the logic when the pattern logic is set.

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

16) PTN

ZERO SUBST / PRBS stage
(zero subst / prbs PaTterN mode)

■ Function

Sets ZERO SUBST / PRBS pattern

Header	Program	Query	Response	(Character No.)
PTN	PTN△m	PTN?	PTN△m	(FIX 1)

■ Value of m

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

■ Command Type

The command is invalid under the following setting conditions.

Program: When the FD is being accessed and when the generation pattern is Alternate or Data

Query: The command is invalid under the following setting conditions and ERR (CR / LF) is output.
When the generation pattern is Alternate or Data

■ Usage Example

Program: When setting generation pattern to PRBS $2^7 - 1$
OUTPUT△700;"PTN△2"Query: When generation pattern logic set to PRBS $2^{31} - 1$
OUTPUT△700;"PTN?"
ENTER△700:B\$
PRINT△B\$

↓

PTN△9 (CR / LF) output

: When generation pattern set to DATA
OUTPUT△700;"PTN?"
ENTER△700:B\$
PRINT△B\$

↓

ERR (CR / LF) output

17) MRK PRBS mark ratio (**MaRK** ration mode)

■ **Function** Sets mark ratio at PRBS pattern

Header	Program	Query	Response (Character No.)
MRK	MRK△m	MRK?	MRK△m (FIX 1)

■ **Value of m**

	When positive logic (Positive)	When negative logic (Negative)
∅	0 / 8	8 / 8
1	1 / 8	7 / 8
2	1 / 4	<u>3 / 4</u>
3	1 / 2	1 / 2

■ **Command Type** Sequential command

■ **Usage Restrictions** The command is invalid under the following setting conditions.

Program: When the FD is being accessed and when the generation pattern is Alternate, Data or Zero Subst

Query: The command is invalid under the following setting conditions and ERR (CR / LF) is output.
When the generation pattern is Alternate, Data or Zero Subst

■ **Usage Example** **Program:** When setting pattern mark ratio to 0 / 8

OUTPUT△700; "MRK△∅"

Query: When pattern mark ratio set to 1 / 8

OUTPUT△700; "MRK?"

ENTER△700; B\$

PRINT△B\$

↓

MRK△1 (CR / LF) output

: When generation pattern set to DATA

OUTPUT△700; "MRK?"

ENTER△700; B\$

PRINT△B\$

↓

ERR (CR / LF) output

18) ALTAlternate pattern A / B display switch (**AL**Ternate a / b)■ **Function**

Sets alternate pattern A / B display switch

Header	Program	Query	Response (Character No.)
ALT	ALT△m	ALT?	ALT△m (FIX 1)

■ **Value of m**

0 : A pattern

1 : B pattern

■ **Command Type**

Sequential command

■ **Usage Restrictions**

The command is invalid under the following setting conditions.

Program: When the FD is being accessed and when the generation pattern is DATA, ZERO SUBST, or PRBS

Query: The command is invalid under the following setting conditions and ERR (CR / LF) is output.
When the generation pattern is DATA, ZERO SUBST, or PRBS

■ **Usage Example**

Program: When switching alternate display to A
OUTPUT△700;"ALT△0"

Query: When alternate display set to A
OUTPUT△700;"ALT?"
ENTER△700;B\$
PRINT△B\$

↓

ALT△1 (CR / LF) output

: When generation pattern set to PRBS
OUTPUT△700;"ALT?"
ENTER△700;B\$
PRINT△B\$

↓

ERR (CR / LF) output

19) EAD Error Insertion (**Error A****D****dition**)

■ **Function** Inserts specified allocation code error at 1 route in 32 routes

Header	Program	Query	Response (Character No.)
EAD	EAD△m	EAD?	EAD△m (FIX 1)

■ **Value of m**

At Internal Error Insertion (EEI△0)	At External Error Insertion (EEI△1)
0 : OFF (No error insertion)	0 : OFF (No error insertion)
1 : 1×10^{-4}	1 : ON (Error insertion)
2 : 1×10^{-5}	
3 : 1×10^{-6}	
4 : 1×10^{-7}	
5 : 1×10^{-8}	
6 : 1×10^{-9}	
7 : SINGLE	

■ **Command Type** Sequential command

■ **Usage Restrictions** The command is invalid under the following setting conditions.

Program: When the FD is being accessed

Query: None

■ **Usage Example** Program: When inserting 1×10^{-5} error
OUTPUT△700;"EAD 2"

Query: When 1×10^{-5} error inserted
OUTPUT△700;"EAD?"
ENTER△700;B\$
PRINT△B\$

↓

EAD△1 (CR/LF) output

■ **Note** When switching between external and internal error insertion, the previous conditions are set.
For example, when external error insertion is set to ON after 1×10^{-8} was set at internal error insertion, the error insertion rate becomes 1×10^{-8} when switching to internal error insertion.

20) LPT**Alternate A / B loop times (Loop Time)**■ **Function**

Sets currently-displayed pattern loop times from A or B pattern

Header	Program	Query	Response	(Character No.)
LPT	LPT△m	LPT?	LPT△m	(FIX 3)

■ **Value of m**

The number of loop times is set in the following range.

Min. value: 1
 Max. value: 127
 Step: 1

The response is as follows:

LPT△△△1 (min. value)
 :
 LPT△127 (max. value)

■ **Command Type**

Sequential command

■ **Usage Restrictions**

The command is invalid under the following setting conditions.

Program: When the FD is being accessed and when a generation pattern other than ALTERNATE is set

Query: The command is invalid under the following setting conditions and ERR (CR / LF) is output.
 When a generation pattern other than ALTERNATE is set

■ **Usage Example**

Program: When setting loop times to 10 times
 OUTPUT△700;"LPT△10"

Query: When loop times is set to 17 times
 OUTPUT△700;"LPT?"
 ENTER△700;B\$
 PRINT△B\$

↓

LPT△△17 (CR / LF) output

: When generated pattern is set to DATA
 OUTPUT△700;"LPT?"
 ENTER△700;B\$
 PRINT△B\$

↓

ERR (CR / LF) output

21) DLN**Data length (Data Length)**■ **Function**

Generation data length is specified when the generation pattern is ALTERNATE or DATA.

Header	Program	Query	Response	(Character No.)
DLN	DLN△m	DLN?	DLN△m	(FIX 7)

■ **Value of m**

The data length is set in the following range.

● **Alternate Pattern**

Max. value: 4194304

Min. value: 128

Step: 128

● **Data Pattern**

Max. value: 8388608

Min value: 2

Step: Divided into following range according to data length

Data length

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**

Sequential Command

■ **Usage Restrictions**

The command is invalid under the following setting conditions.

Program: When the FD is being accessed and when the generation pattern is ZERO SUBST and PRBS

Query: The command is invalid under the following setting conditions and ERR (CR / LF) is output.

When the generation pattern is ZERO SUBST and PRBS

21) DLN

■ Usage Example

Data length (Data LeNgth) continued

Program: When setting data length to 4 bits

OUTPUT△700;"DLN△4"

Query: When data length set to 32 bits

OUTPUT△700;"DLN?"

ENTER△700;B\$

PRINT△B\$

↓

DLN△△△△△△32 (CR/LF) output

: When generation pattern set to PRBS

OUTPUT△700;"DLN?"

ENTER△700;B\$

PRINT△B\$

↓

ERR (CR/LF) output

■ Note

When the data length setting has been set to a non-existent value, the data length is set to the best value as shown below.

Changed to value smaller than and closest to input value

Example) ● Input data length ● Set data length

131075

→→→

131072

22) ZLN ZERO SUBST length (Zero subst LeNgth)

- **Function** Sets zero substitution bit length when generation pattern is ZERO SUBST

Header	Program	Query	Response (Character No.)
ZLN	ZLN△m	ZLN?	ZLN△m (FIX 5)

- **Value of m** The zero substitution bit length is set in the following range.

Max. value: Following range according to ZERO SUBST stage

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

Min. value: 1

Step: 1

- **Command Type** Sequential command

- **Usage Restrictions** The command is invalid under the following setting conditions.

Program: When the FD is being accessed and when the generation pattern is ALTERNATE, DATA and PRBS

Query: The command is invalid under the following setting conditions and ERR (CR / LF) is output.
When the generation pattern is ALTERNATE, DATA and PRBS

- **Usage Example** **Program:** When setting ZERO SUBST length to 1 bit
OUTPUT△700;"ZLN△1"

Query: When ZERO SUBST length set to 127 bits
OUTPUT△700;"ZLN?"
ENTER△700;B\$
PRINT△B\$

↓

DLN△△△127 (CR / LF) output

: When generation pattern set to PRBS
OUTPUT△700;"ZLN?"
ENTER△700;B\$
PRINT△B\$

↓

ERR (CR / LF) output

23) ADR
PAGPage number (ADdRess / PAGe)

- Function Sets page number

Header	Program	Query	Response	(Character No.)
ADR	ADR△m	ADR?	ADR△m	(FIX 9)
PAG	PAG△m	PAG?	PAG△m	(FIX 9)

- Value of m The page number is set in the following range.

Max. value: 134217728

Min. value: 1

Step: 1

- Command Type Sequential Command

- Usage Restrictions The command is invalid under the following setting conditions.

Program: When the FD is being accessed and when the display is the pattern sync position

Query: The command is invalid under the following setting conditions and ERR (CR / LF) is output.
When the display is the pattern sync position

- Usage Example

Program: When setting page to 1 page
OUTPUT△700;"PAG△1"

Query: When page number is set to 16000 pages
OUTPUT△700;"PAG?"
ENTER△700;B\$
PRINT△B\$

↓

PAG△△△△△16000 (CR / LF) output

: When display is pattern sync position
OUTPUT△700;"PAG?"
ENTER△700;B\$
PRINT△B\$

↓

ERR (CR / LF) output

23) ADR PAG

■ Note

Page number (ADdRes / PAGe) continued

There are two page number setting commands: ADR and PAG, but their function is the same.

However, the maximum settable page number varies according to the set generation pattern and the value set for the data length.

In addition, the maximum value of m above cannot exceed 1342177278 when a value exceeding the maximum settable page number is input, the value is changed to the maximum page number.

Example: At data length 32, displayed page number 1 and maximum page number 2, when $\text{PAG}\triangle 3$ is input, the displayed page number becomes 2.

The maximum page number cannot be a value exceeding the data length / 16; if there is a remainder, page number is set to the result of the division + 1.

For equipment not having the DISPLAY key near the center of the pattern section, there is no ERR output for a query.

In addition, the program command is valid except when the FD is being accessed.

24) BIT**Pattern bit (pattern BIT)**

- **Function** Sets bit pattern

Header	Program	Query	Response (Number of characters)
BIT	<ul style="list-style-type: none"> • NR1 format BIT△m • HEX format BIT△#Hm 	BIT?	<p>The bit contents from the set page number to the max. 8 page part, and up to the max. pattern set page, are output in the following format.</p> <p>PAG△*****: BIT△#H****, #H****, . . . , #H****</p> <p style="text-align: center;">} _____ }</p> <p>Displayed as HEX together with bit pattern output header page. Max. 8 page part (FIX 4 each)</p>

- **Value of m** The bit pattern is set in the following range.
- | | |
|--|---|
| <ul style="list-style-type: none"> • NR1 format <li style="padding-left: 20px;">Max. value: 65535 <li style="padding-left: 20px;">Min. value: 0 <li style="padding-left: 20px;">Step: 1 | <ul style="list-style-type: none"> • HEX format <li style="padding-left: 20px;">Max. value: FFFF <li style="padding-left: 20px;">Min. value: 0 <li style="padding-left: 20px;">Step: 1 |
|--|---|

- **Command Type** Sequential command

- **Usage Restrictions** The command is invalid under the following setting conditions.
- Program: When the FD is being accessed and when the generation pattern is ZERO SUBST and PRBS

Query: None

24) BIT

Pattern bit (pattern BIT) continued

■ Usage Example

Program: When setting bit pattern from currently set page to page 3
 OUTPUT△700;"BIT△10,20,30"
 OUTPUT△700;"BIT△#HFFFF,#H1000,#H2000"

The bit pattern can be set for continuous pages by separating the data with a comma (,).

When setting the page number and the bit pattern from that page number to page 4

OUTPUT△700;"PAG△10;BIT△10,20,30,40"
 OUTPUT△700;"PAG△10;BIT△#HFFFF,#H1000,
 #H2000,#H3000"

Query: When display page is 1 and maximum obtainable page number is 10 and reading data from page 1 to page 8
 OUTPUT△700;"BIT?"

ENTER△700;B\$
 PRINT△B\$

↓

PAG△*****1;BIT△#H0000,#H0000,
 #H0000,#H0000,#H0000,
 #H0000,#H0000,#H0000

(CR/LF) output

■ Note

The pattern bits for continuous pages (max. 8 pages) can be set by separating the data with commas (,) for both NR1 and HEX formats.

Bit 1 and bit 16 of the bit display can be set and respond as the LSB and MSB, respectively.

For example, when 32768 is set, the MSB (bit 16) becomes 1.

25) ALLPattern data preset (All pages, All bits) (**preset ALL**)■ **Function**

Sets all bits of all pages of pattern data to 0s or 1s

Header	Program	Query	Response	(Character No.)
ALL	ALL△m	None	None	

■ **Value of m**

∅ : All pages clear

1 : All pages set

■ **Command Type**

Sequential command

■ **Usage Restrictions**

The command is invalid under the following setting conditions.

Program: When the FD is being accessed and when the generation pattern is ZERO SUBST and PRBS

■ **Usage Example**

Program: When clearing all pages when generation pattern is DATA
OUTPUT△7∅∅;"ALL△∅"

Clears data of all pages

■ **Note**

When the generation pattern is ALTERNATE pattern, the A or B pattern is preset according to the A / B display switch (No. 18) condition.

For example, when the A pattern is displayed, only the A pattern is preset when this command is executed.

26) PST Pattern data preset (1 page, All bits) (PreST)

- **Function** Sets all bits of page 1 of pattern data to 0s or 1s

Header	Program	Query	Response (Character No.)
PST	PST△m	None	None

- **Value of m**
 - ∅ : Page 1 clear
 - 1 : Page 1 set
- **Command Type** Sequential command
- **Usage Restrictions** The command is invalid under the following setting conditions.
 - Program: When the FD is being accessed and when the generation pattern is ZERO SUBST and PRBS
- **Usage Example** Program: When clearing page 1 when generation pattern is DATA OUTPUT△700; "PST△∅"
 - Clears data for page 1 of set pages
- **Note**
 - When the generation pattern is ALTERNATE pattern, the A or B pattern is preset according to the A / B display switch (No. 18) condition.
 - For example, when the A pattern is displayed, only the A pattern is preset when this command is executed.

27) PSP**Pattern sync trigger position (Pattern Sync Position)**■ **Function**

Sets trigger sync position of pattern sync (variable)

Header	Program	Query	Response	(Character No.)
PSP	PSP△m	PSP?	PSP△m	(FIX 9)

■ **Value of m**

The pattern sync position is set in the following range.

Max. value: Differs according to generation pattern

PRBS7, ZERO SUBST. 7 : 8

PRBS9, ZERO SUBST. 11 : 32

PRBS11, ZERO SUBST. 11 : 128

PRBS15, ZERO SUBST. 15 : 2048

PRBS20 : 65536

PRBS23 : 524288

PRBS31 : 134217728

DATA : (bit length / 16) + alpha

If the result of the division is not an integer, alpha = 1.

If it is an integer, alpha = 0.

Min. value: 1

Step: 1

The response is as follows:

PSP△△△△△△△△△△1 (Min. value)

:

PSP△134217728 (Max. value)

■ **Command Type**

Sequential command

■ **Usage Restrictions**

The command is invalid under the following setting conditions.

Program: When the FD is being accessed and when the display is page

Query: The command is invalid under the following setting conditions and ERR (CR / LF) is output.

When the display is page

27) PSP

Pattern sync trigger position (Pattern Sync Position)
continued

■ Usage Example

Program: When setting pattern sync position to page 3

```
OUTPUT△700;"PSP△3"
```

Query: When pattern sync position set to page 189

```
OUTPUT△700;"PSP?"
```

```
ENTER△700;B$
```

```
PRINT△B$
```

↓

```
PSP△△△△△△△189 (CR/LF) output
```

: When display is page

```
OUTPUT△700;"PSP?"
```

```
ENTER△700;B$
```

```
PRINT△B$
```

↓

```
ERR (CR/LF) output
```

28) PPD**Page number / Pattern sync trigger position display switch
(Page / Pattern sync position Display)**■ **Function**

Switches page number and pattern sync trigger position display

Header	Program	Query	Response	(Character No.)
PPD	PPD△m	PPD?	PPD△m	(FIX 1)

■ **Value of m**

0 : Page number display

1 : Pattern sync position display

■ **Command Type**

Sequential command

■ **Usage Restrictions**

The command is invalid under the following setting conditions.

Program: When the FD is being accessed and when the display is page

Query: None

■ **Usage Examples**Program: When displaying pattern sync trigger position
OUTPUT△700;"PPD△1"Query: When page number is displayed
OUTPUT△700;"PPD?"
ENTER△700;B\$#
PRINT△B\$

↓

PPD△0 (CR/LF) output

SECTION 9 DETAILS OF DEVICE MESSAGES

- **OUTPUT Section**

The following pages show each control message for the OUTPUT section.

The Δ in the strings indicates a space.

29) DTM Data output termination voltage (**Data Termination**)

- **Function** Sets data output termination voltage

Header	Program	Query	Response	(Character No.)
DTM	DTM△m	DTM?	DTM△m	(FIX 1)

- **Value of m**
 - 0 : GND
 - 1 : -2 V (ECL)
- **Command Type** Sequential command
- **Usage Restrictions** The command is invalid under the following setting conditions.
 - Program: When the FD is being accessed
 - Query: None
- **Usage Example**
 - Program: When setting data output termination voltage to GND
OUTPUT△700;"DTM△0"
 - Query: When data output termination voltage set to -2 V
OUTPUT△700;"DTM?"
ENTER△700;B\$
PRINT△B\$
↓
DTM△1 (CR/LF) output

30) CTM Clock1 output termination voltage (Clock TerMination)

- **Function** Sets clock output termination voltage

Header	Program	Query	Response (Character No.)
CTM	CTM△m	CTM?	CTM△m (FIX 1)

- **Value of m**
 - 0 : GND
 - 1 : -2 V (ECL)
- **Command Type** Sequential command
- **Usage Restrictions** The command is invalid under the following setting conditions.
 - Program: When the FD is being accessed
 - Query: None
- **Usage Example**
 - Program: When setting clock output termination voltage to GND
 OUTPUT△700;"CTM△0"
 - Query: When clock output termination voltage set to -2 V
 OUTPUT△700;"CTM?"
 ENTER△700;B\$
 PRINT△B\$
 ↓
 CTM△1 (CR/LF) output

31) OFS**Offset reference value (OFSset)**■ **Function**Sets DATA / $\overline{\text{DATA}}$ / CLOCK output offset reference value

Header	Program	Query	Response (Character No.)
OFS	OFS△m	OFS?	OFS△m (FIX 1)

■ **Value of m**

0 : Offset reference value VOH

1 : Offset reference value VTH

2 : Offset reference value VOL

■ **Command Type**

Sequential command

■ **Usage Restrictions**

The command is invalid under the following setting conditions.

Program: When the FD is being accessed

Query: None

■ **Usage Example**

Program: When setting offset reference voltage to VOL
 OUTPUT△700;"OFS△2"

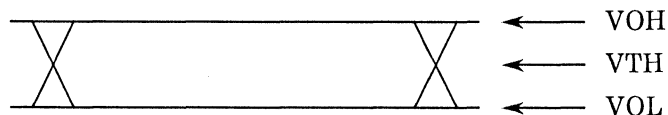
Query: When offset reference voltage set to VTH
 OUTPUT△700;"OFS?"
 ENTER△700;B\$
 PRINT△B\$

↓

OFS△1 (CR/LF) output

■ **Note**

The outline of the offset reference voltage is shown below.



When setting the offset reference voltage, the display of each output offset voltage is updated to the best value at that time.

32) DAPData output amplitude (**Data Amplitude**)

- **Function** Sets DATA/ $\overline{\text{DATA}}$ output amplitude

Header	Program	Query	Response	(Character No.)
DAP	DAP Δ m	DAP?	DAP Δ m	(FIX 5)

- **Value of m**
- Numeric range: Max. value: 2.000
 Min. value: 0.250
 Step: 0.002

However, when Option 03 (1/4 output) is installed, when the display is 1/4 SPEED, 1/4 output amplitude setting can be performed at the following numeric range.

Numeric range: Max. value: 2.000
 Min. value: 0.500
 Step: 0.002

- **Command Type** Sequential command

- **Usage Restrictions** The command is invalid under the following setting conditions.

Program: When the FD is being accessed. When Option 03 is not installed, 1/4 output amplitude setting is disabled.

Query: None

- **Usage Example**
- Program: When setting data amplitude to 1.2 V
 OUTPUT Δ 700;"DAP Δ 1.2"

Query: When data amplitude set to 0.5 V
 OUTPUT Δ 700;"DAP?"
 ENTER Δ 700;B\$
 PRINT Δ B\$

↓

DAP Δ 0.500 (CR/LF) output

32) DAP

■ **Note**

Data output amplitude (**Data Amplitude**) continued

DATA / $\overline{\text{DATA}}$ tracking set to off, only data output amplitude is set.

33) NAP

$\overline{\text{DATA}}$ output amplitude (iNverted AmPlitude)

■ **Function** Sets $\overline{\text{DATA}}$ output amplitude

Header	Program	Query	Response (Character No.)
NAP	NAP△m	NAP?	NAP△m (FIX 5)

■ **Value of m** The $\overline{\text{DATA}}$ output amplitude setting range is set from 0.25 V to 2.0 V.

Numeric Range Max. value: 2.000
 Min. value: 0.250
 Step: 0.002

■ **Command Type** Sequential command

■ **Usage Restrictions** The command is invalid under the following setting conditions.

Program: At $\overline{\text{DATA}}$ / $\overline{\text{DATA}}$ tracking on
 When the display is 1 / 4 SPEED when Option 03 is installed
 When the FD is being accessed

Query: The command is invalid under the following setting conditions and ERR (CR / LF) is output.
 At $\overline{\text{DATA}}$ / $\overline{\text{DATA}}$ tracking on
 When the display is 1 / 4 SPEED when Option 03 is installed

■ **Usage Example** Program: When setting $\overline{\text{DATA}}$ amplitude to 1.2 V
 OUTPUT△700;"NAP△1.2"

Query: When $\overline{\text{DATA}}$ amplitude set to 0.5 V
 OUTPUT△700;"NAP?"
 ENTER△700;B\$
 PRINT△B\$
 ↓
 NAP△0.500 (CR / LF) output

34) DOS**Data output offset (Data OffSet)**■ **Function**Sets $\overline{\text{DATA}}$ / DATA output offset

Header	Program	Query	Response	(Character No.)
DOS	DOS Δ m	DOS?	DOS Δ m	(FIX 6)

■ **Value of m**

The DATA output offset setting range differs according to the setting of the offset reference value. When the offset reference value is VOH, the value is set from -2.0 V to $+2.0\text{ V}$.

Numeric Range Max. value: 2.000
 Min. value: -2.000
 Step: 0.001

When the offset reference value is VTH, the value is set from -3.0 V to $+1.875\text{ V}$.

Numeric Range Max. value: 1.875
 Min. value: -3.000
 Step: 0.001

When the offset reference value is VOL, the value is set from -4.0 V to $+1.75\text{ V}$.

Numeric Range Max. value: 1.750
 Min. value: -4.000
 Step: 0.001

When Option 03 is installed and the display is 1 / 4 SPEED, the 1 / 4 DATA output offset reference value is set in the following range.

When the offset reference value is VOH, the value is set from -1.5 V to $+1.5\text{ V}$.

Numeric Range Max. value: 1.500
 Min. value: -1.500
 Step: 0.001

When the offset reference value is VTH, the value is set from -2.5 V to $+1.25\text{ V}$.

Numeric Range Max. value: 1.250
 Min. value: -2.500
 Step: 0.001

When the offset reference value is VOL, the value is set from -3.5 V to $+1.0\text{ V}$.

Numeric Range Max. value: 1.000
 Min. value: -3.500
 Step: 0.001

34) DOSData output offset (Data OffSet) continued■ **Command Type**

Sequential command

■ **Usage Restrictions**

The command is invalid under the following setting conditions.

Program: When the FD is being accessed

Query: None

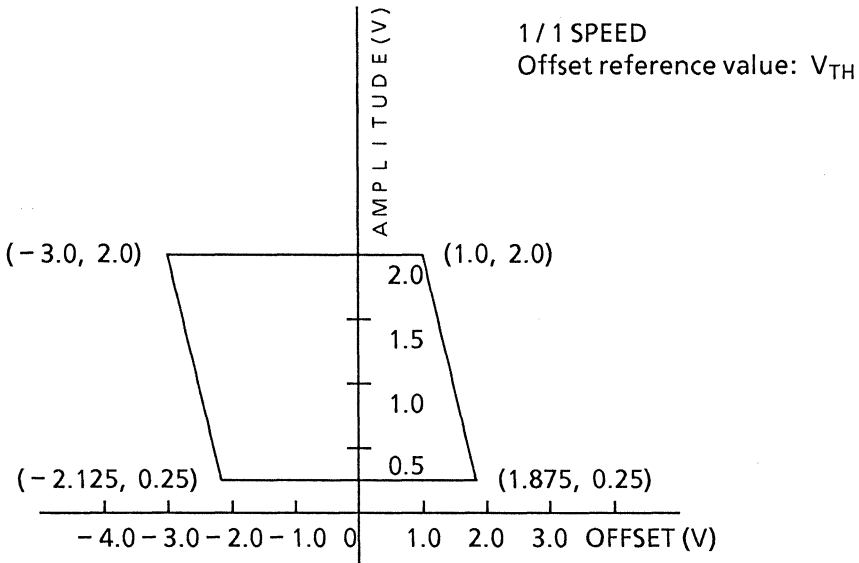
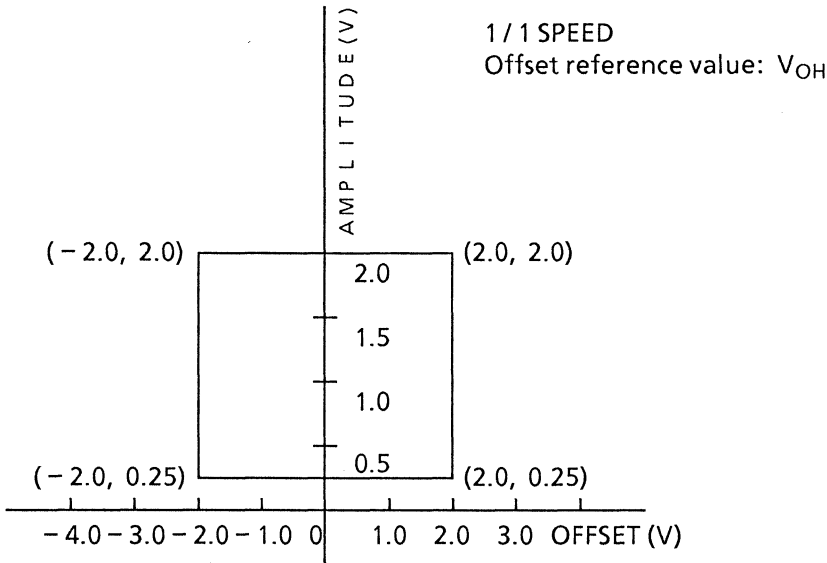
■ **Usage Example**Program: When setting offset to -1.2 V
OUTPUT Δ 700;"DOS Δ -1.2"Query: When offset set to 0.5 V
OUTPUT700;"DOS?"
ENTER Δ 700;B\$
PRINT Δ B\$

↓

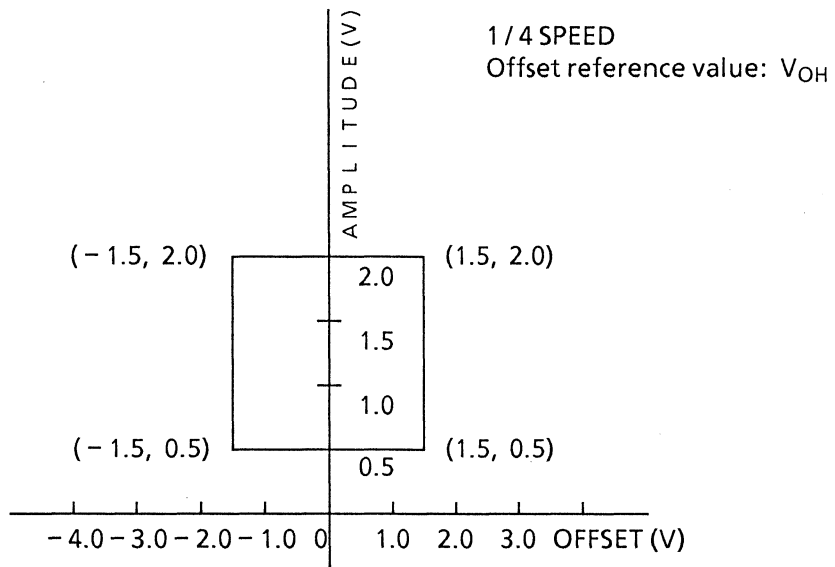
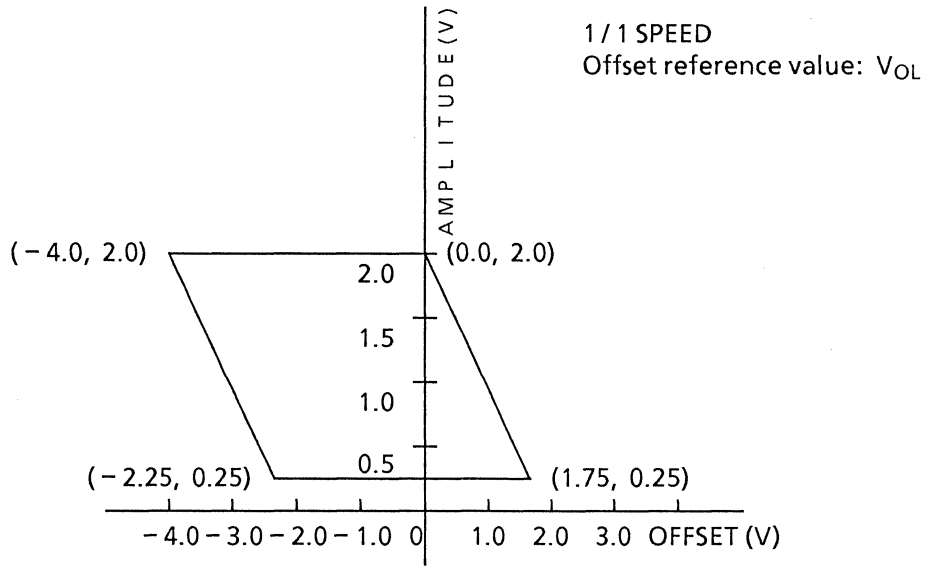
DOS Δ Δ 0.500 (CR/LF) output■ **Note**The only data output off set is set at DATA/DATA tracking off.

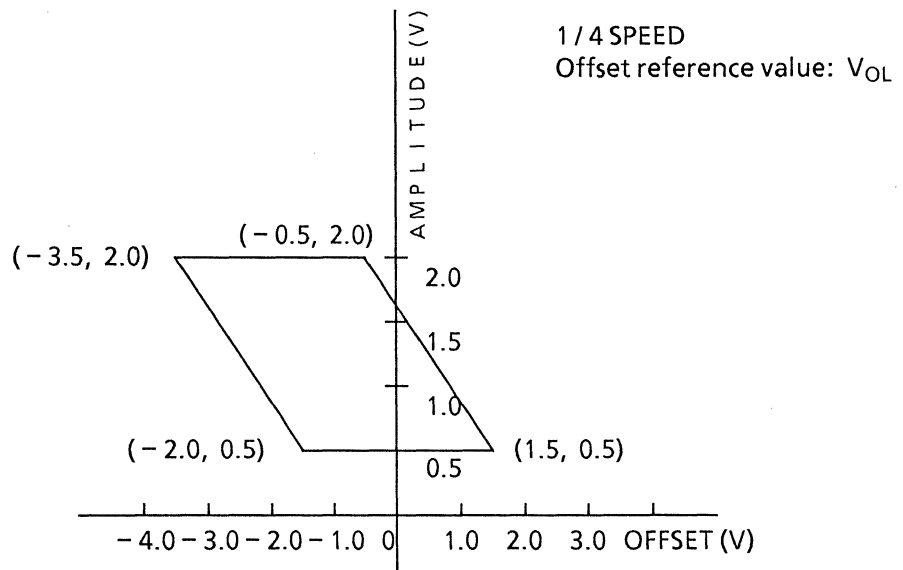
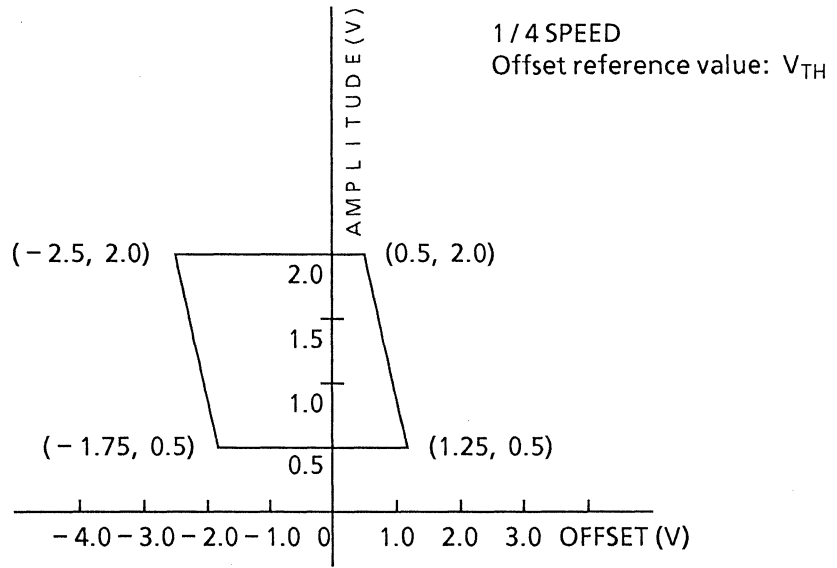
When Option 03 is installed and the display is 1 / 4 SPEED, the 1 / 4 output offset is set.

The settable offset range differs according to the set output amplitude as shown on the next page.



SECTION 9 DETAILS OF DEVICE MESSAGES





35) NOS

$\overline{\text{DATA}}$ output offset (iNverted data OffSet)

■ **Function** Sets $\overline{\text{DATA}}$ output offset

Header	Program	Query	Response	(Character No.)
NOS	NOS Δ m	NOS?	NOS Δ m	(FIX 6)

■ **Value of m** The $\overline{\text{DATA}}$ output offset setting range differs according to the setting of the offset reference value. When the offset reference value is VOH, the value is set from -2.0 V to +2.0 V.

Numeric Range Max. value: 2.000
 Min. value: -2.000
 Step: 0.001

When the offset reference value is VTH, the value is set from -3.0 V to +1.875 V.

Numeric Range Max. value: 1.875
 Min. value: -3.000
 Step: 0.001

When the offset reference value is VOL, the value is set from -4.0 V to +1.75 V.

Numeric Range Max. value: 1.75
 Min. value: -4.000
 Step: 0.001

■ **Command Type**

■ **Usage Restrictions** The command is invalid under the following setting conditions.

Program: At $\overline{\text{DATA}}$ / $\overline{\text{DATA}}$ tracking on
 When the display is 1 / 4 SPEED when Option 03 is installed
 When the FD is being accessed

Query: The command is invalid under the following setting conditions and ERR (CR / LF) is output.
 At $\overline{\text{DATA}}$ / $\overline{\text{DATA}}$ tracking on
 When the display is 1 / 4 SPEED when Option 03 is installed

35) NOS

■ Usage Example

 $\overline{\text{DATA}}$ output offset (iNverted data OffSet) continued

Program: When setting $\overline{\text{DATA}}$ offset to 1.2 V
 OUTPUT Δ 700;"NOS Δ 1.2"

Query: When $\overline{\text{DATA}}$ amplitude set to 0.5 V
 OUTPUT Δ 700;"NOS?"
 ENTER Δ 700;B\$
 PRINT Δ B\$

↓

NOS Δ 0.500 (CR / LF) output

: When Option 02 installed
 OUTPUT Δ 700;"NOS?"
 ENTER Δ 700;B\$
 PRINT Δ B\$

↓

ERR (CR / LF) output

■ Note

The offset settable range is the same as the 1 / 1 SPEED data offset setting (DOS).

36) CDL

Clock1 output delay time (Clock1 DeLay)

■ **Function** Sets delay time between Clock1 and DATA / $\overline{\text{DATA}}$

Header	Program	Query	Response	(Character No.)
CDL	CDL△m	CDL?	CDL△m	(FIX 5)

■ **Value of m** The Clock1 output phase setting range is 500 ps to -500 ps.

Numeric range Max. value: 500 ps
 Min. value: -500 ps
 Step: 1

■ **Command Type** Sequential command

■ **Usage Restrictions** The command is invalid under the following setting conditions.

Program: When the display is 1 / 4 SPEED when Option 03 is installed
 When the FD is being accessed

Query: The command is invalid under the following setting conditions and ERR (CR / LF) is output.
 When the display is 1 / 4 SPEED when Option 03 is installed

■ **Usage Example** Program: When setting clock delay time to -100 ps

OUTPUT△700;"CDL△-100"

Query: When clock delay time set to 100 ps

OUTPUT△700;"CDL"
 ENTER△700;B\$
 PRINT△B\$



CDL△△△100 (CR / LF) output

37) CAP**Clock1 output amplitude (Clock1 Amplitude)**■ **Function**

Sets Clock1 output amplitude

Header	Program	Query	Response (Character No.)
CAP	CAP△m	CAP?	CAP△m (FIX 5)

■ **Value of m**

The Clock1 output amplitude setting range is set in the range 0.25 V to 2.0 V

Numeric range Max. value: 2.000
 Min. value: 0.250
 Step: 0.002

When Option 03 is installed and the display is set to 1 / 4 SPEED, the output amplitude range is 0.5 V to 2.0 V.

Numeric range Max. value: 2.000
 Min. value: 0.500
 Step: 0.002

■ **Command Type**

Sequential command

■ **Usage Restrictions**

The command is invalid under the following setting conditions.

Program: When the FD is being accessed

Query: None

■ **Usage Example**

Program: When setting Clock1 output amplitude to 1.5 V
 OUTPUT△700;"CAP△1.5"

Query: When Clock1 output amplitude set to 0.25 V
 OUTPUT△700;"CAP?"
 ENTER△700;B\$
 PRINT△B\$

↓

CAP△0.250 (CR / LF) output

38) COS**Clock1 output offset (Clock1 OffSet)**■ **Function**

Sets Clock1 output amplitude

Header	Program	Query	Response	(Character No.)
COS	COS Δ m	COS?	COS Δ m	(FIX 6)

■ **Value of m**

The Clock1 output offset setting range differs according to the setting of the offset reference value. When the offset reference value is VOH, the value is set from -2.0 V to $+2.0$ V.

Numeric Range Max. value: 2.000
 Min. value: -2.000
 Step: 0.001

When the offset reference value is VTH, the value is set from -3.0 V to $+1.875$ V.

Numeric Range Max. value: 1.875
 Min. value: -3.000
 Step: 0.001

When the offset reference value is VOL, the value is set from -4.0 V to $+1.75$ V.

Numeric Range Max. value: 1.750
 Min. value: -4.000
 Step: 0.001

When Option 03 is installed and the display is 1/4 SPEED, the 1/4 DATA output offset reference value is set in the following range.

When the offset reference value is VOH, the value is set from -1.5 V to $+1.5$ V.

Numeric Range Max. value: 1.500
 Min. value: -1.500
 Step: 0.001

When the offset reference value is VTH, the value is set from -2.5 V to $+1.25$ V.

Numeric Range Max. value: 1.250
 Min. value: -2.500
 Step: 0.001

When the offset reference value is VOL, the value is set from -3.5 V to $+1.0$ V.

Numeric Range Max. value: 1.000
 Min. value: -3.500
 Step: 0.001

■ **Command Type**

Sequential command

38) COS**Clock1 output offset (Clock1 OffSet) continued**

■ Usage Restrictions

The command is invalid under the following setting conditions.

Program: When the FD is being accessed

Query: None

■ Usage Example

Program: When setting Clock1 output offset to 1.5 V
 OUTPUT△700;"COS△1.5"

Query: When Clock1 output offset set to -0.25 V
 OUTPUT△700;"COS?"
 ENTER△700;B\$
 PRINT△B\$

↓

COS△-0.250 (CR / LF) output

■ Note

The CLOCK1 output offset settable range is the same as the data output offset and varies according to the set CLOCK1 output amplitude.

39) OON Output on / off (**Output ON / off**)

■ **Function** Sets DATA / $\overline{\text{DATA}}$, CLOCK1 / $\overline{\text{CLOCK1}}$, and 1 / 4 output to 0 V

Header	Program	Query	Response	(Character No.)
OON	OON△m	OON?	OON△m	(FIX 1)

■ **Value of m** ∅ : Output off
 1 : Output on

■ **Command Type** Sequential Command

■ **Usage Restrictions** The command is invalid under the following setting conditions.

Program: When the FD is being accessed

Query: None

■ **Usage Example** Program: When setting output to off
 OUTPUT△700;"OON△0"

Query: When output on
 OUTPUT△700;"OON?"
 ENTER△700;B\$
 PRINT△B\$

↓

OON△1 (CR/LF) output

40) DDS**Data / $\overline{\text{Data}}$ display switch (Data / $\overline{\text{data}}$ Display Select)****■ Function**Selects DATA / $\overline{\text{DATA}}$ display setting value

Header	Program	Query	Response	(Character No.)
DDS	DDS△m	DDS?	DDS△m	(FIX 1)

■ Value of m

∅ : DATA setting value display

1 : $\overline{\text{DATA}}$ setting value display**■ Command Type**

Sequential command

■ Usage Restrictions

The command is invalid under the following setting conditions.

Program: At DATA / $\overline{\text{DATA}}$ tracking on

When the display is 1 / 4 SPEED when Option 03 is installed

When the FD is being accessed

Query: The command is invalid under the following setting conditions and ERR (CR / LF) is output.

At DATA / $\overline{\text{DATA}}$ tracking on

When the display is 1 / 4 SPEED when Option 03 is installed

■ Usage Example

Program: When displaying DATA setting value

OUTPUT△700;"DDS△∅"

Query: When DATA setting value displayed

OUTPUT△700;"DDS?"

ENTER△700;B\$

PRINT△B\$

↓

DDS△1 (CR / LF) output

41) TRKDATA / $\overline{\text{DATA}}$ tracking (**data / $\overline{\text{data}}$ TRacking**)■ **Function**Sets DATA / $\overline{\text{DATA}}$ tracking on / off

Header	Program	Query	Response	(Character No.)
TRK	TRK△m	TRK?	TRK△m	(FIX 1)

■ **Value of m**

∅ : Tracking off

1 : Tracking on

■ **Command Type**

Sequential command

■ **Usage Restrictions**

The command is invalid under the following setting conditions.

Program: When the display is 1 / 4 SPEED when Option 03 is installed
 When the FD is being accessed

Query: The command is invalid under the following setting
 conditions and ERR (CR / LF) is output.
 When the display is 1 / 4 SPEED when Option 03 is installed

■ **Usage Example**

Program: When setting tracking off
 OUTPUT△700;"TRK△0"

Query: When tracking on
 OUTPUT△700;"TRK?"
 ENTER△700;B\$
 PRINT△B\$

↓

TRK△1 (CR / LF) output

42) SPD

1 / 1 SPEED / 1 / 4 SPEED display switch
(1 / 1SPeED / 1 / 4speed display select)

- **Function** Selects 1 / 1SPEED / 1 / 4SPEED setting value display

Header	Program	Query	Response (Character No.)
SPD	SPD△m	SPD?	SPD△m (FIX 1)

- **Value of m** 0 : 1 / 1SPEED display
 1 : 1 / 4SPEED display
- **Command Type** Sequential command
- **Usage Restrictions** The command is invalid under the following setting conditions.
- Program: When Option 03 is not installed
 When the FD is being accessed
- Query: The command is invalid under the following setting
 conditions and ERR (CR / LF) is output.
 When Option 03 is not installed
- **Usage Example** Program: When displaying 1 / 1SPEED
 OUTPUT△700;"SPD△0"
- Query: When 1 / 4SPEED displayed
 OUTPUT△700;"SPD?"
 ENTER△700;B\$
 PRINT△B\$
- ↓
- SPD△ 1 (CR / LF) output
- : When Option 03 not installed
 OUTPUT△700;"SPD?"
 ENTER△700;B\$
 PRINT△B\$
- ↓
- ERR (CR / LF) output
- **Note** This command is valid when Option 03 is installed.

- Other Sections

The following pages show each control message for the other sections.

The \triangle in the strings indicates a space.

- Note

Please note that when the setting of the function switch on the back panel is Remote, the contents set by the command take priority of the setting of the function switch, but when the setting is Local, the back-panel function switch setting contents are returned.

43) SOP**Sync signal output selection (Sync OutPut)**

- **Function** Controls sync signal output

Header	Program	Query	Response (Character No.)
SOP	SOP△m	SOP?	SOP△m (FIX 1)

- **Value of m**
 - ∅ : 1 / 64 CLOCK
 - 1 : PATTERN SYNC (FIXED)
 - 2 : PATTERN SYNC (VARIABLE)
- **Command Type** Sequential command
- **Usage Restrictions** The command is invalid under the following setting conditions.
 - Program: When the FD is being accessed
 - Query: None
- **Usage Example**
 - Program: When setting sync signal output to 1 / 64 CLOCK
OUTPUT△700; "SOP△∅"
 - Query: When sync signal output set to PATTERN SYNC (FIXED)
OUTPUT△700; "SOP?"
ENTER△700; B\$
PRINT△B\$
↓
SOP△1 (CR / LF) output

44) ECH Error insertion channel (Error addition Channel)

- **Function** Selects error insertion channel

Header	Program	Query	Response (Character No.)
ECH	ECH△m	ECH?	ECH△m (FIX 2)

- **Value of m** The error insertion channel can be set in the following range of channels 1 to 32.

(ch1) Min. value: 1

(ch32) Max. value: 32

The response is as follows:

(ch1) ECH△△1

:

(ch32) ECH△32

- **Command Type** Sequential command

- **Usage Restrictions** The command is invalid under the following setting conditions.

Program: When the FD is being accessed

Query: None

- **Usage Example** Program: When inserting error in ch3
OUTPUT△700;"ECH△3"

Query: When error inserted in ch8
OUTPUT△700;"ECH?"
ENTER△700;B\$
PRINT△B\$

↓

ECH△△8 (CR/LF)

45) SFT**Mark ratio and bit shift value
(mark ratio and bit ShiFT)**■ **Function**

Sets PRBS mark ratio AND bit shift value

Header	Program	Query	Response	(Character No.)
SFT	SFT△m	SFT?	SFT△m	(FIX 1)

■ **Value of m**

∅ : 1 bit shift

1 : 3 bit shift

■ **Command Type**

Sequential command

■ **Usage Restrictions**

The command is invalid under the following setting conditions.

Program: When the FD is being accessed and when the generation pattern is ALTERNATE, DATA and ZERO SUBST

Query: The command is invalid under the following setting conditions and ERR is output.

When the generation pattern is ALTERNATE, DATA and ZERO SUBST

■ **Usage Example**

Program: When setting mark ratio AND bit shift value to 1 bit
OUTPUT△700;"SFT△0"

Query: When mark ratio AND bit shift set to 1 bit
OUTPUT△700;"SFT?"
ENTER△700;B\$
PRINT△B\$

↓

SFT△1 (CR/LF) output

: When generation pattern set to ALTERNATE, DATA or ZERO SUBST

OUTPUT△700;"SFT?"
ENTER△700;B\$
PRINT△B\$

↓

ERR (CR/LF) output

46) EEI External error insertion (**External Error Injection**)

- **Function** Switches error insertion method between external insertion and internal insertion

Header	Program	Query	Response (Character No.)
EEI	EEI△m	EEI?	EEI△m (FIX 1)

- **Value of m**
 - ∅ : Internal error insertion
 - 1 : External error insertion
- **Command Type** Sequential command
- **Usage Restrictions** The command is invalid under the following setting conditions.
 - Program: When the FD is being accessed
 - Query: None
- **Usage Example**
 - Program: When setting external error insertion to on
OUTPUT△700;"EEI△1"
 - Query: When external error insertion set to off
OUTPUT△700;"EEI?"
ENTER△700;B\$
PRINT△B\$
↓
EEI△∅ (CR/LF) output

47) APS**Alternate pattern A / B switch signal selection
(Alternate Pattern A / B Select timing)**■ **Function**

Selects whether to use internally-generated output or external input signal for alternate pattern A / B switch signal

Header	Program	Query	Response (Character No.)
APS	APS△m	APS?	APS△m (FIX 1)

■ **Value of m**

∅ : Internal error insertion

1 : External error insertion

■ **Command Type**

Sequential command

■ **Usage Restrictions**

The command is invalid under the following setting conditions.

Program: When the FD is being accessed

Query: None

■ **Usage Example**

Program: When setting use of external input signal
OUTPUT△700;"APS△1"

Query: When using internally-generated signal
OUTPUT△700;"APS?"
ENTER△700;B\$
PRINT△B\$

↓

APS△∅ (CR / LF) output

48) INI Initialize (INItialize)

- **Function** Forcibly initializes to settings at factory shipment

Header	Program	Query	Response (Character No.)
INI	INI	None	None

- **Command Type** Sequential command
- **Usage Restrictions** When the FD is being accessed
- **Usage Example** Program: When initializing to settings at factory shipment
OUTPUT△700;"INI"
- **Note** This command executes the same as the operation when the LOCAL key is kept pressed at power-on.

49) WRT**Pattern data input byte number (pattern data WRiTe)****■ Function**

Sets pattern data DMA send byte number and start address

Header	Program	Query	Response (Character No.)
WRT	WRT△m1,m2	None	None

■ Value of m**m1** : Pattern send byte number

Numeric range: Max. value 1048376

Min. value 1

Step 1

m2 : Pattern input header address

Numeric range: Max. value 524288

Min. value 0

Step 1

The above maximum values are halved when the generation pattern setting is ALTERNATE pattern.

■ Command Type

Sequential command

■ Usage Restrictions

The command is invalid under the following setting conditions.

Program: When the FD is being accessed

When the generation pattern is ZERO SUBST or PRBS

When the pattern send byte number + the pattern header address × 2 is greater than 1048376.

■ Usage Example

Program: When the generation pattern is DATA and setting data from page 1 to page 10.

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 for pages 1 to 10 is set

■ Note

This equipment defines the byte number required for the DMA sending pattern data and the input header address, switches the DMA mode, and defines the storage address to the internal RAM area, from each value of the NR part.

The relationship between the pattern header address and the actually-set page is

$$(\text{pattern header address} + 1) = \text{actual page number}$$

In addition, the DMA mode is released after sending of the pattern data is completed.

For sending the pattern data DMA, refer to Sending Pattern Data DMA in the appendix.

50) RED?

Pattern data output byte number
(**pattern data REaD ?**)

■ **Function**

Sets byte number and start address read when sending pattern data
DMA

Header	Program	Query	Response	(Character No.)
RED	None	RED?△m1, m2	Data Pattern String	(according to m1)

■ **Value of m**

m1 : Pattern send byte number
 Numeric range: Max. value 1048376
 Min. value 1
 Step 1

m2 : Pattern output header address
 Numeric range: Max. value 524288
 Min. value 0
 Step 1

The above maximum values are halved when the generation pattern setting is ALTERNATE pattern.

■ **Command Type**

Sequential command

■ **Usage Restrictions**

The command is invalid under the following setting conditions and ERR (CR/ LF) is output.

Query: When the generation pattern is ZERO SUBST or PRBS
 When the pattern send byte number + the pattern header address × 2 is greater than 1048376.

■ **Usage Example**

Query: When the generation pattern is DATA and setting data from page 1 to page 10.
 DIM△B(9)
 OUTPUT△700; "RED?△20, 0"
 ENTER△700△USING△"W"; B\$(*)
 PRINT△B(*)

Data for pages 1 to 10 is printed

■ **Note**

This equipment defines the byte number required for the DMA sending pattern data and the input header address, switches the DMA mode, and defines the storage address to the internal RAM area, from each value of the NR part.

The relationship between the pattern header address and the actually-set page is

(pattern header address + 1) = actual page number

In addition, the DMA mode is released after sending of the pattern data is completed.

For sending the pattern data DMA, refer to Sending Pattern Data DMA in the appendix.

51) PLL?**Phase locked loop condition (PLL unlock?)**■ **Function**

Checks whether internal synthesizer PLL locked or unlocked

Header	Program	Query	Response (Character No.)
None	None	PLL?	PLL△m (FIX 1)

■ **Value of m**

∅ : Locked

1 : Unlocked

■ **Command Type**

Sequential command

■ **Usage Restrictions**

The command is invalid under the following setting conditions and ERR (CR/LF) is output.

Query: When Option 01 not installed

■ **Usage Example**

Query: When internal synthesizer unlocked

OUTPUT△700;"PLL?"

ENTER△700:B\$

PRINT△B\$

↓

PLL△1 (CR/LF) output

: When Option 01 not installed

OUTPUT△700;"PLL?"

ENTER△700:B\$

PRINT△B\$

↓

ERR (CR/LF) output

52) RTM Set internal timer (Real Time setting)

■ **Function** Sets internal timer

Header	Program	Query	Response (Character No.)
RTM	RTM△m1 , m2 , m3 , m4 , m5 , m6	RTM?	RTM△m1 , m2 , m3m4 , m5 , m6 (FIX 2 each)

■ **Value of m** The setting step is 1 for each if m1 to m6

Function	Setting Range
m1 : Year	0~99 (Gregorian)
m2 : Month	1~12
m3 : Day	1~31 (Leap Year; 1~29)
m4 : Hour	0~23
m5 : Minute	0~59
m6 : Second	0~59

■ **Command Type** Sequential command

■ **Usage Restrictions** The command is invalid under the following setting conditions.

Program: When the FD is being accessed

Query: None

■ **Usage Example**

When setting time to 8:23:45 28 May 1967

OUTPUT△700; "RTM△67 , 5 , 28 , 8 , 23 , 45"

The timer starts after the setting is completed.

Query: When internal timer is 11:30:00 23 April 1994

OUTPUT△700; "RTM?"

ENTER△700:B\$

PRINT△B\$

↓

RTM△94 , △4 , 23 , 11 , 30 , △0 (CR/LF) output

53) PWI**Power cut, Circuit recovery status (Power fail Interval)****■ Function**

Output power cut time, power recovery time and cut interval

Header	Program	Query	Response (Character No.)
None	None	PWI?	PWF△m1 , m2 , m3 , m4 , m5 , m6 , PWR△m1 , m2 , m3 , m4 , m5 , m6 , PWI△△△△m7 , m4 , m5 , m6 (CR / LF)

■ Value of m

PWF displays the power cut time in the year, month, day, hour, minute, and second format.

PWR displays the circuit recovery time in the year, month, day, hour, minute, and second format.

PWI displays the cut interval in the day, hour, minute and second format.

Function	Setting Range
m1 : Year	0~99 (Gregorian)
m2 : Month	1~12
m3 : Day	1~31 (Leap Year: 1~29)
m4 : Hour	0~23
m5 : Minute	0~59
m6 : Second	0~59
m7 : Day	0~99999

■ Command Type

Sequential command

■ Usage Restrictions

The command is invalid under the following setting conditions and ERR (CR / LF) is output.

Query: After device clear is received and after initialize

■ Use Example

```
Query: OUTPUT△700;"PWI?"
      ENTER△700;B$
      PRINT△B$

      PWF△95,04,14,00,00,00,
      PWR△95,04,15,12,00,00,
      PWI△△△△△△△△1,12,00,00 (CR / LF) output
```

: After device clear received or after initialize

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

↓

ERR (CR / LF) output

■ Note

The response to the query in a single character string without a line feed.

54) DLY

Delay status (DeLaY unlock?)

■ **Function**

Check whether servo circuit of clock delay circuit in READY or BUSY status

Header	Program	Query	Response	(Character No.)
None	None	DLY?	DLY△m	(FIX 1)

■ **Value of m**

∅ : READY status
1 : BUSY status

■ **Command Type**

Sequential command

■ **Usage Restrictions**

The command is invalid under the following setting conditions and ERR (CR / LF) is output.

Query: When the display is 1 / 4SPEED when Option 03 is installed

■ **Usage Example**

Query: When servo circuit of clock delay circuit READY
 OUTPUT△700:"DLY?"
 ENTER△700;B\$
 PRINT△B\$
 ↓
 DLY△∅(CR / LF) output

: When display 1 / 4SPEED
 OUTPUT△700:"DLY?"
 ENTER△700;B\$
 PRINT△B\$
 ↓
 ERR (CR / LF) output

SECTION 10
EXAMPLE OF PROGRAM CREATION

TABLE OF CONTENTS

10.1	Example of Program creation Using HP9000	10-6
------	--	------

(Blank)

This section describes examples of how to create MP1763B GPIB programs.

The sample programs which appear in this section were written for a PC-compatible computer with GPIB interface card of National Instruments (N.I).

The program 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) Frequency setting 1 (point setting)
- (2) Frequency setting 2 (Fixed range setting)
- (3) Generation pattern setting
- (4) Output signal setting
- (5) Reading file information from floppy disk
- (6) Floppy disk operation
- (7) Status byte checking
- (8) DMA transfer for pattern data

Table 10-1 shows the preparations for sample program execution.

Table 10-1 Preparation for Sample Program Execution (1 / 2)

Controller	Preparation for program execution
DEC pc	<ul style="list-style-type: none"> ● Set the GPIB address of MP1763B as "1". ● Set IBCONF as follows. <ul style="list-style-type: none"> ① <Board Characteristics> Board : GPIB 0 (Defines board as "GPIB0") 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 0AH 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 02c0h 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	<p>② <Device Characteristics></p> <p>Device : PPG (Defines device name as "PPG")</p> <table data-bbox="746 539 1401 931"> <tr> <td>Primary GPIB Address</td> <td>1</td> </tr> <tr> <td>Secondary GPIB Address</td> <td>NONE</td> </tr> <tr> <td>Timeout setting</td> <td>1000 sec</td> </tr> <tr> <td>Serial Poll Timeout</td> <td>1 sec</td> </tr> <tr> <td>Terminate Read on EOS</td> <td>Yes</td> </tr> <tr> <td>Set EOI with EOS on Writes</td> <td>Yes</td> </tr> <tr> <td>Type of compare on EOS</td> <td>7-Bit</td> </tr> <tr> <td>EOS byte</td> <td>0Ah</td> </tr> <tr> <td>Send EOI at end of Write</td> <td>Yes</td> </tr> <tr> <td>Enable Repeat Addressing</td> <td>No</td> </tr> </table> <p>③ Devices ② is connected to the GPIB0 of device ① using the GPIB Device Map.</p> <ul data-bbox="560 1104 1203 1133" style="list-style-type: none"> • Connects MP1763B and DECpc with GPIB cables. 	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	0Ah	Send EOI at end of Write	Yes	Enable Repeat Addressing	No
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	0Ah																				
Send EOI at end of Write	Yes																				
Enable Repeat Addressing	No																				

10.1 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 four types of functions.

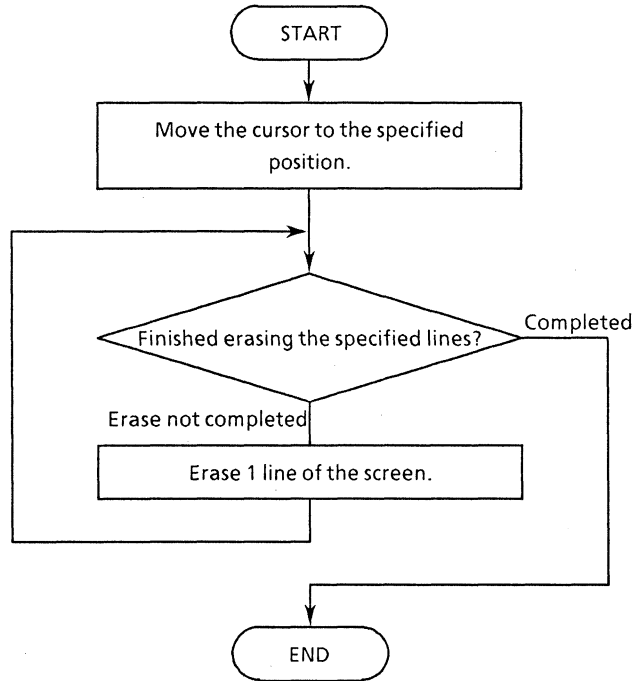
Table 10-2 Table of COMMON.BAS Functions

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	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
1.3	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
1.4	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



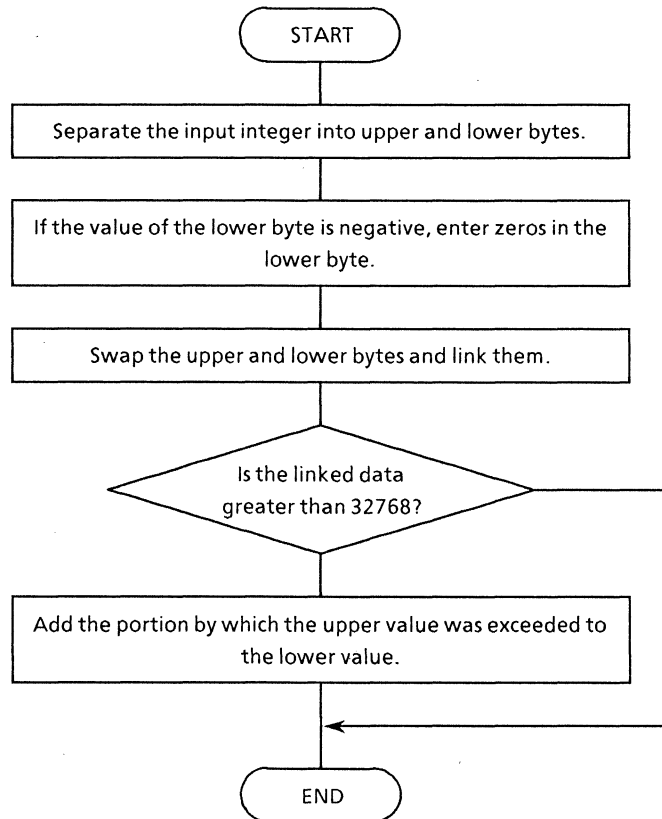
- Program list

```

' ---- 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) FUNCTION Exchange (i%): Swaps 16-bit integer data in units of byte.

- Flowchart



- Program list

```

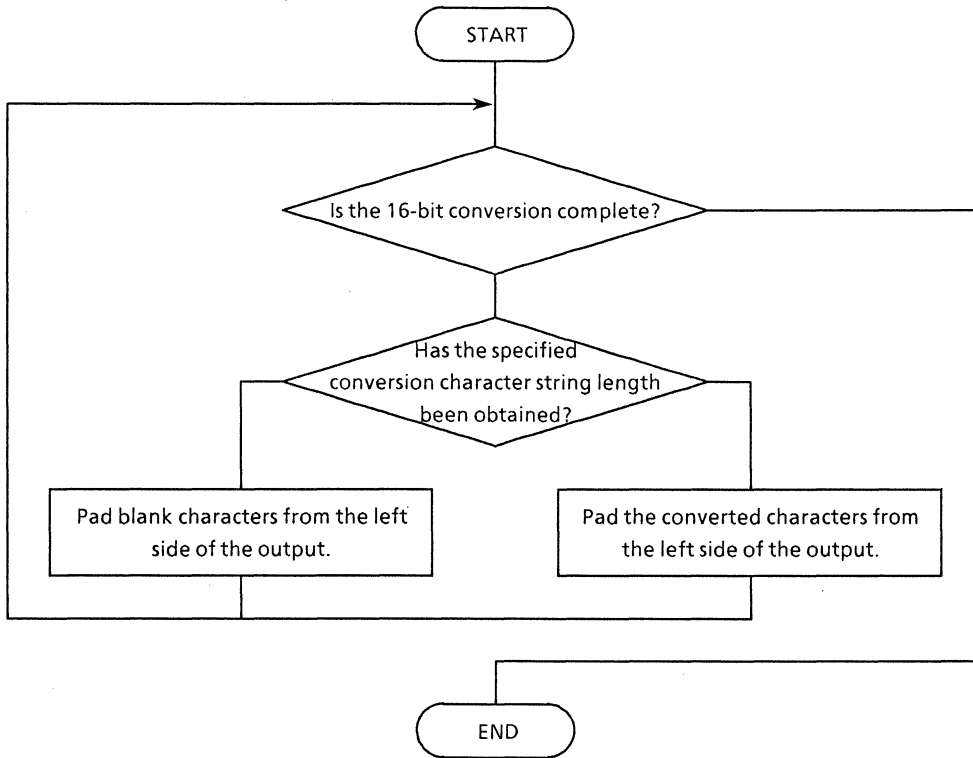
' ---- Exchange 16-bits pattern data ----
' In  i%:16bits pattern data (used integer)
'
' Procedure for swap of low byte and high byte .
' This program is bit manipulation of integer valu. Why this program used
' real value because one is overflow detect on bit manipulation of integer
' value, another one is internal manipulation by real value although input
' parameter is integer. And integer declare 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

```

(1.3) itob\$(l%, v%): Converts integers into binary character strings.

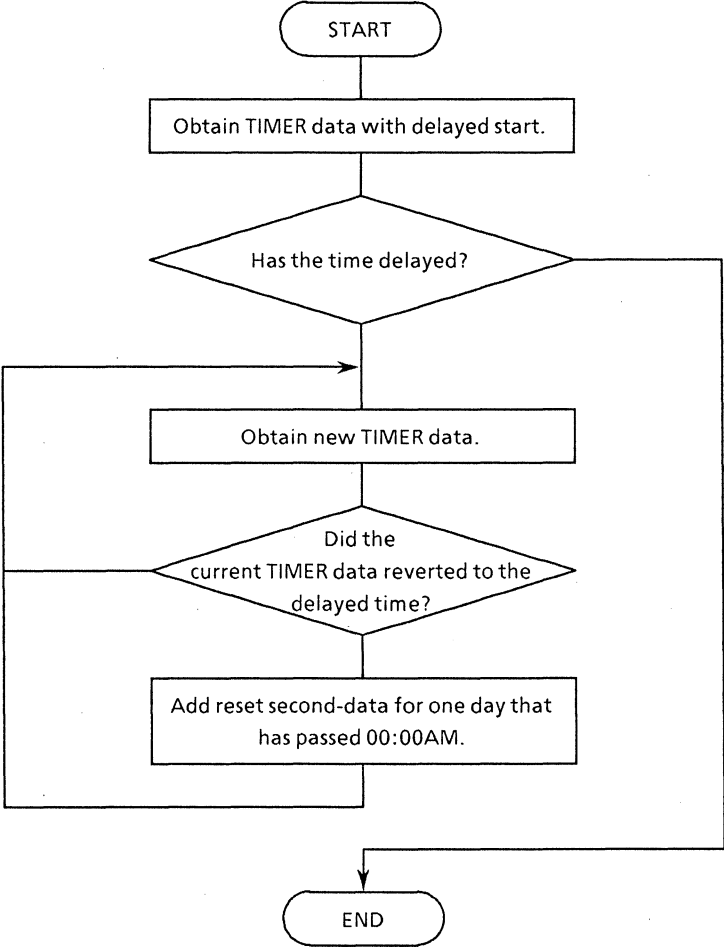
- Flowchart



- Program list

(1.4) waidly (tim!): Creates the wait time

- Flowchart



- Program list

<Explanation of ACS_GPIB.BAS>

ACS_PGIB.BAS consists of the following 10 types of functions.

Table 10-3 ACS_GPIB.BAS Functions

Module number	Function	Processing
2.1	SUB wrtcmd1 (w\$)	Send commands to PPG. w\$: Command character string to be sent (input)
2.2	FUNC readcmd1\$ ()	Reads messages from PPG. readcmd1\$: Message character string (returned value)
2.3	SUB dmawrt (w% (), i%)	Transfers in DMA to PPG. w% () : Integer array of pattern data to be transferred i% : Number of elements of integer array
2.4	SUB EndPoll ()	Performs polling of the END bit of the MSS status register.
2.5	SUB SRQPoll ()	Performs polling of the ERROR bit and SRQ bit of the MSS status register.
2.6	SUB StatusMask (sre%, ese%, ese1%, ese2%)	Sets the mask pattern for the status, event, and expansion registers. sre% : Mask pattern for status register ese% : Mask pattern for standard event register ese1% : Mask pattern for expansion event register 1 ese2% : Mask pattern for expansion event register 2
2.7	SUB StatusDisp (stb%, esr%, esr1%, esr2%)	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 esr1% : Pattern of expansion event register 1 setting status esr2% : Pattern of expansion event register 2 setting status
2.8	FUNC gpinit% ()	Executes GPIB initialization and returns the initialization as function values. 0 (False) : Error in setting. Initialization failed. 1 (True) : PPG completed initialization.
2.9	SUB trap ()	Processes system errors.
2.10	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%, GPIBØ%, PPG% ..... ②
```

Item ① loads the NI-488 function definition using the GPIB driver of National Instruments.

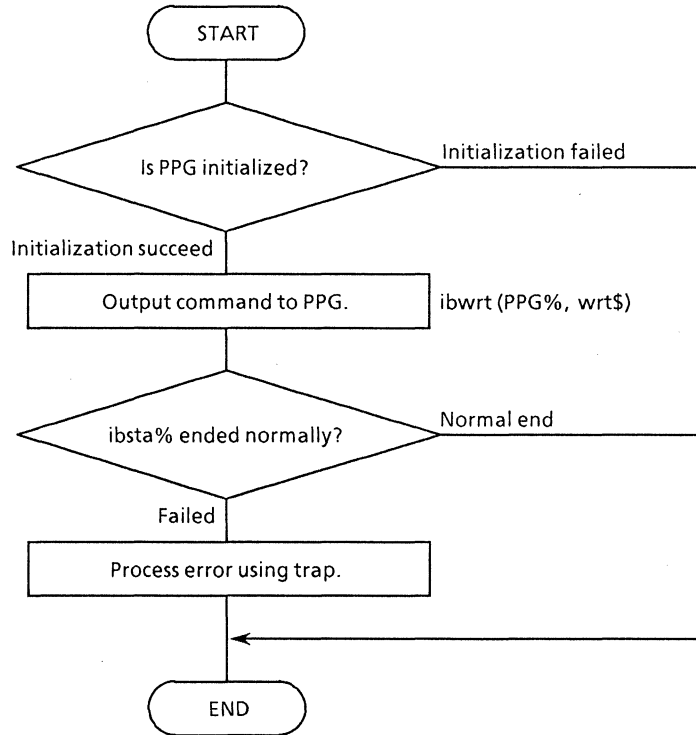
In actual use, specify a directory including 'qbdecl.bas'.

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

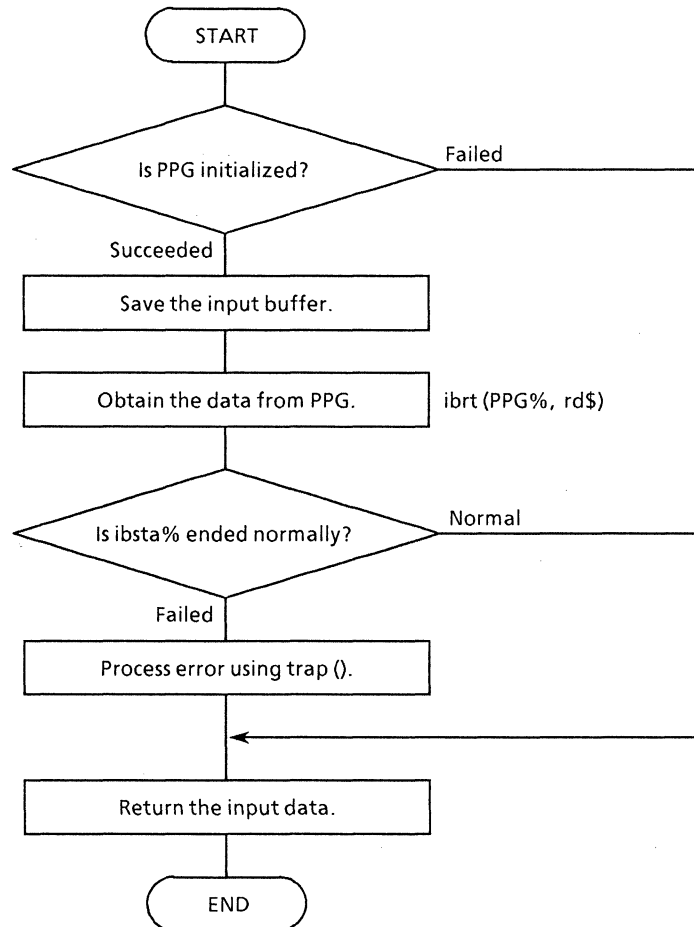


- Program list

(2.2) FUNCTION readcmd1\$ ():

Obtains data in response to the command sent from PPG separately.

- Flowchart



- Program list

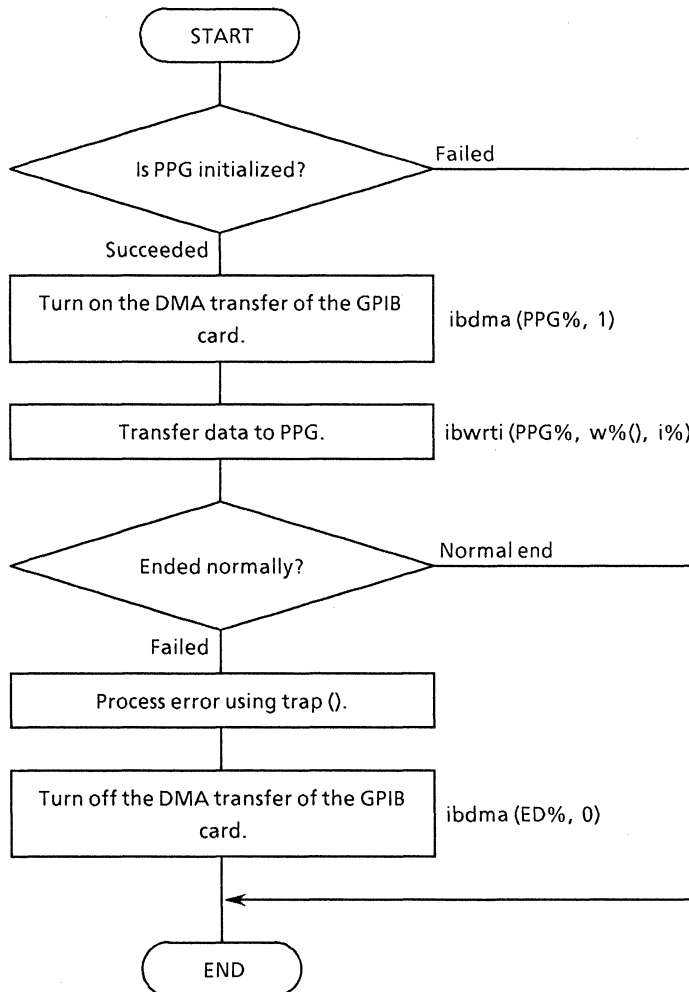
```

' ---- Procedure for data read from PPG ----
'
FUNCTION readcmd1$
  IF DEV% = 1 OR DEV% = 3 THEN
    r$ = SPACE$(256)
    CALL IBRD(PPG%, r$)
    IF IBSTA% < 0 THEN CALL trap
    ' Read data from PPG%
  END IF
  readcmd1$ = r$
END FUNCTION

```

(2.3) SUB dmawrt (w%, i%): Transfers the PPG data in DMA transfer.

- Flowchart



- Program list

```

' --- Procedure for DMA transfer ----
' in   w%():Transmit data pattern of integer array
'     i% :length count for integer array
'
SUB dmawrt (w%(), i%)
  IF DEV% = 1 OR DEV% = 3 THEN
    CALL IBDMA(PPG%, 1)           ' DMA enable

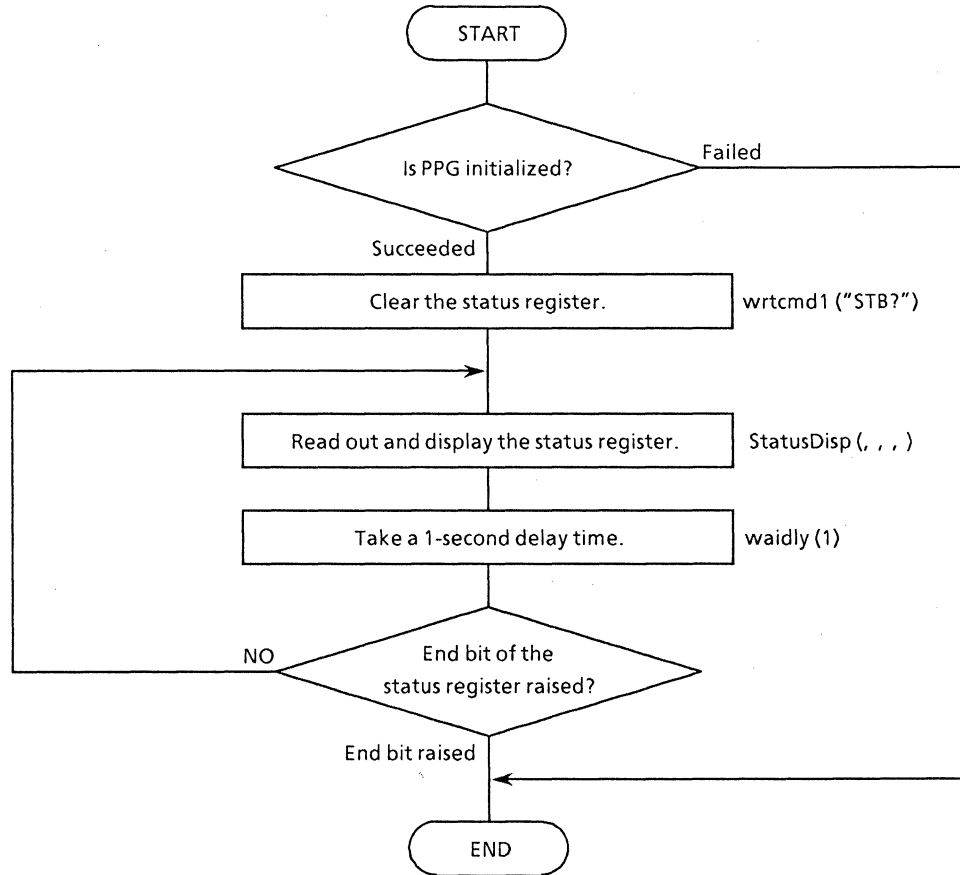
    i% = i% * 2 + 1              ' make up to a byte count
    CALL IBWRTI(PPG%, w%(), i%)
    IF IBSTA% < 0 THEN CALL trap ' call trap if illegal end

    CALL IBDMA(PPG%, 0)         ' DMA disable
  END IF
END SUB

```

(2.4) SUB EndPoll (): Waits until the status end bit is set.

- Flowchart



- Program list

```

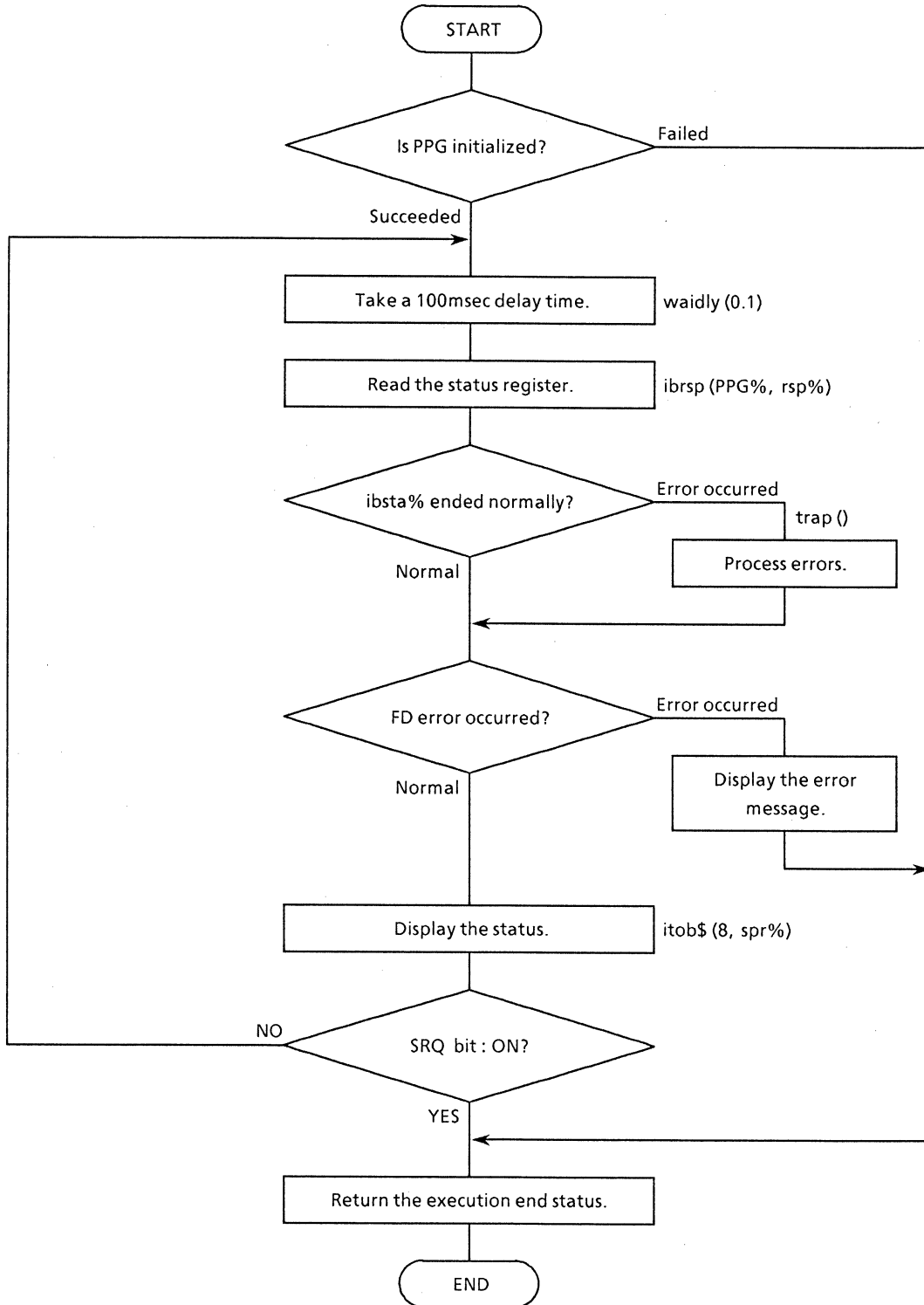
' ---- Procedure for judgement of Measurement end ----
,
SUB EndPoll
    IF DEV% = 1 OR DEV% = 3 THEN
        CALL wrtcmd1("*STB?")           ' reset event flag
        RD$ = LEFT$(readcmd1$, IBCNT% - 1)

        DO
            CALL StatusDisp(reg%, dmy%, dmy2%, dmy3%)

            waidly (1)
        LOOP UNTIL reg% AND &H4
    END IF
END SUB
    
```

(2.5) FUNCTION SRQPoll (): Judges SRQ and error bits.

- Flowchart



- Program list

```
' ---- Procedure for Seli11 poll with SRQ bit ----
'
FUNCTION SRQPoll%
  IF DEV% = 1 OR DEV% = 3 THEN
    exe% = 1
    DO
      waidly (.1)

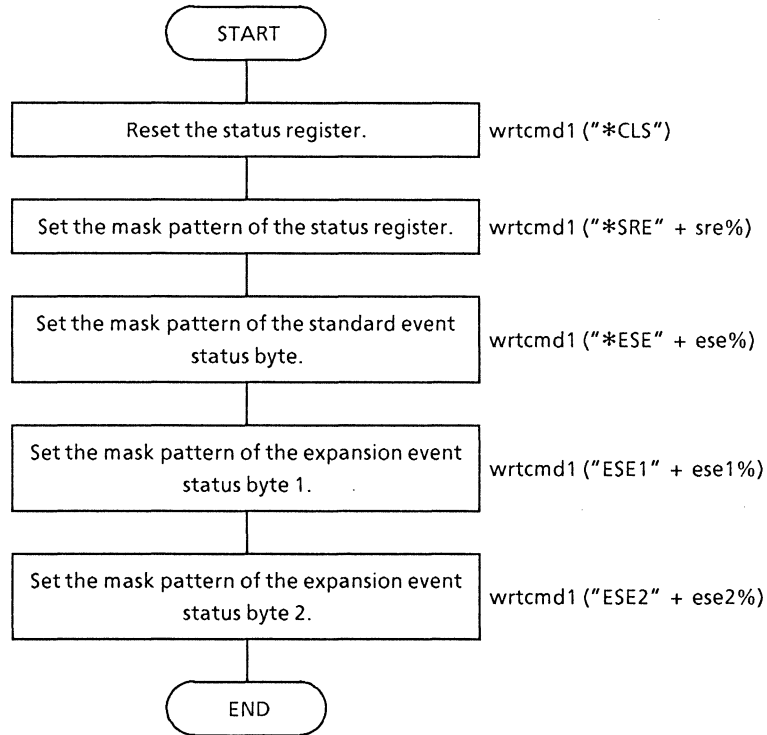
      CALL IBRSP(PPG%, SPR%)
      IF IBSTA < 0 THEN CALL trap
      srq = SPR% AND &H40
      esr1 = SPR% AND &H4
      esr2 = SPR% AND &H8

      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 esr1 = &H4
    END IF
  '
  SRQPoll% = exe%
END FUNCTION
```

(2.6) SUB StatusMask (sre%, ese%, ese1%, ese2%): Sets status registers.

- Flowchart



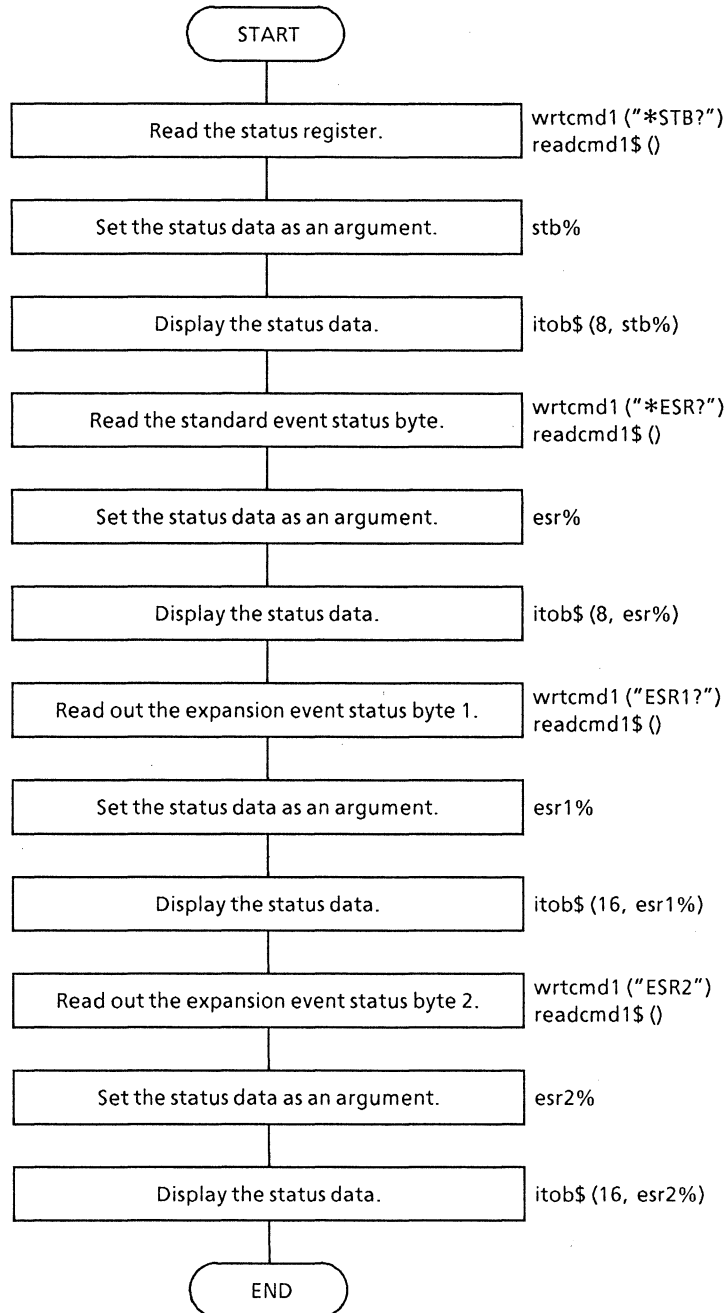
- Program list

```

' ---- 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-1 mask pattern
'      s3%:Extend event status enable register-2 mask pattern
'
SUB StatusMask (s0%, s1%, s2%, s3%)
    wrtcmd1 ("*CLS")
    wrtcmd1 ("*SRE " + STR$(s0%))
    wrtcmd1 ("*ESE " + STR$(s1%))
    wrtcmd1 ("ESE1 " + STR$(s2%))
    wrtcmd1 ("ESE2 " + STR$(s3%))
END SUB
  
```

(2.7) SUB StatusDisp (stb%, esr%, esr1%, esr2%):
Reads out and displays the status register.

- Flowchart



SECTION 10 EXAMPLE OF PROGRAM CREATION

• Program list

```
' ---- Procedure for status byte display ----
' out   stb% :Status byte
'       esr% :Normal event status byte
'       esr1%:Extend event-1 status byte
'       esr2%:Extend event-2 status byte
'
SUB StatusDisp (stb%, esr%, esr1%, esr2%)
  CALL wrtcmd1("*STB?")
  RD$ = LEFT$(readcmd1$, IBCNT% - 1)
  stb% = VAL(RD$)
  sta$ = itob$(8, VAL(RD$))
  LOCATE 1, 60
  PRINT "*STB:"; sta$

  CALL wrtcmd1("*ESR?")
  RD$ = LEFT$(readcmd1$, IBCNT% - 1)
  esr% = VAL(RD$)
  sta$ = itob$(8, VAL(RD$))

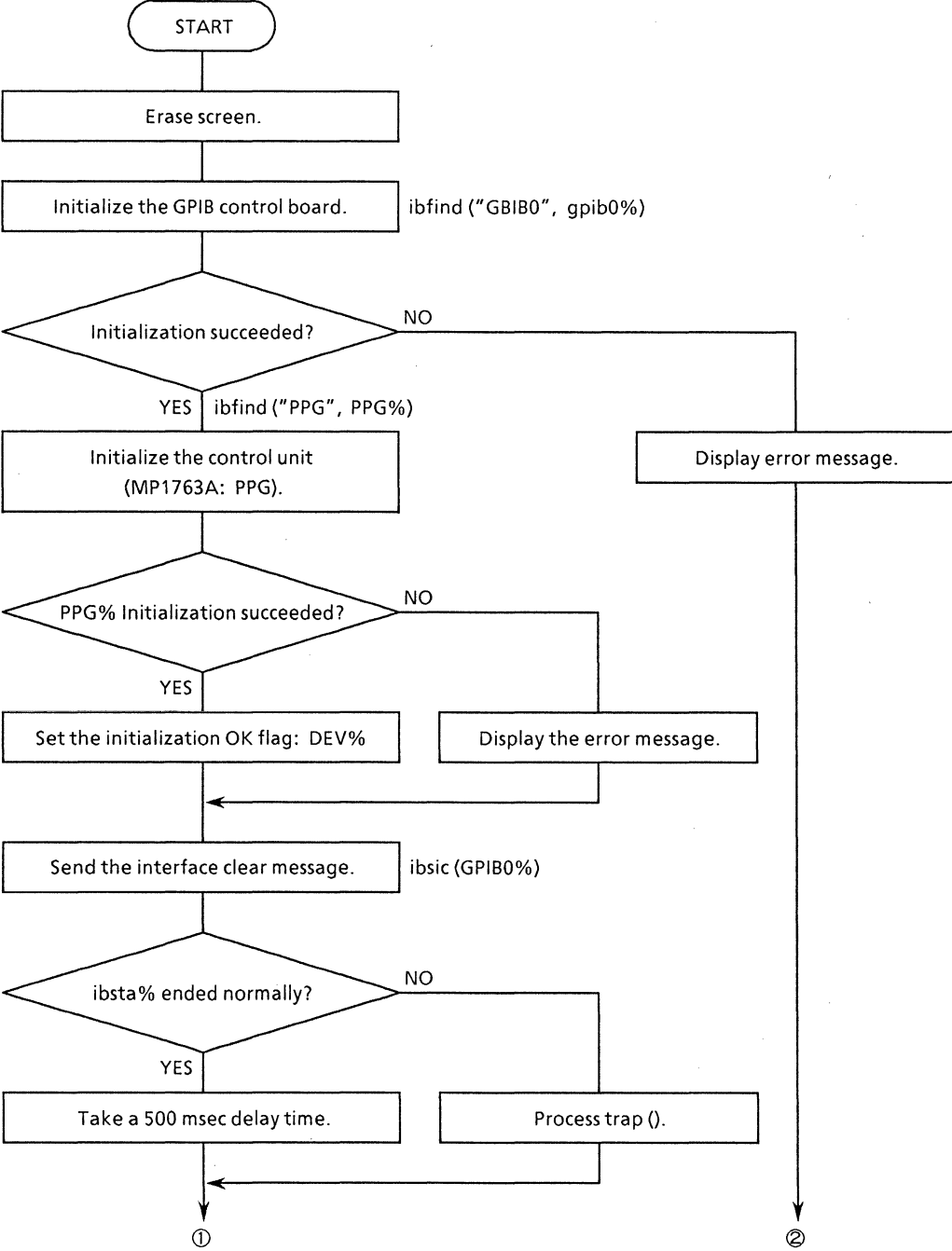
  LOCATE 2, 60
  PRINT "*ESR:"; sta$

  CALL wrtcmd1("ESR1?")
  RD$ = LEFT$(readcmd1$, IBCNT% - 1)
  esr1% = VAL(MID$(RD$, 6, 5))
  sta$ = itob$(16, VAL(MID$(RD$, 6, 5)))
  LOCATE 3, 60
  PRINT "ESR1:"; sta$

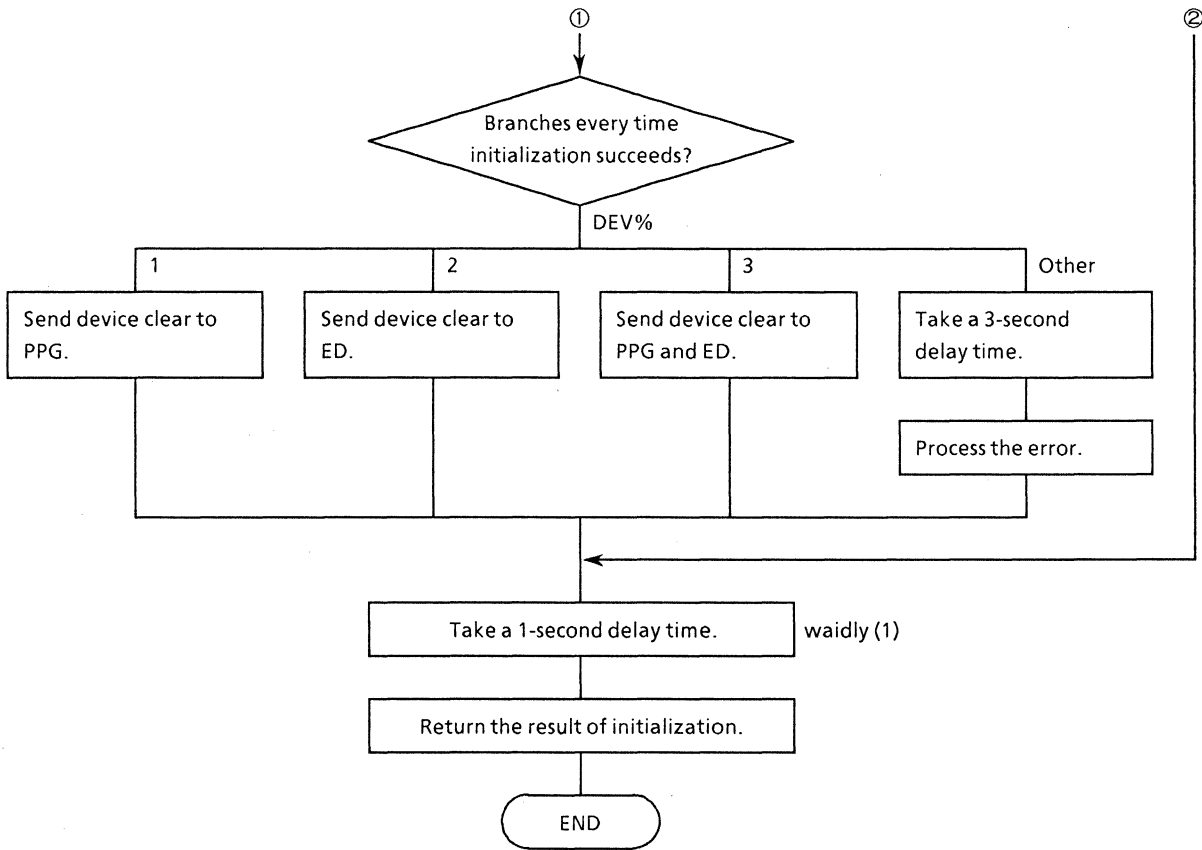
  CALL wrtcmd1("ESR2?")
  RD$ = LEFT$(readcmd1$, IBCNT% - 1)
  esr2% = VAL(MID$(RD$, 6, 5))
  sta$ = itob$(16, VAL(MID$(RD$, 6, 5)))
  LOCATE 4, 60
  PRINT "ESR2:"; sta$
END SUB
```


(2.8) FUNCTION gpinit (): Initializes the GPIB control environment.

- Flowchart



SECTION 10 EXAMPLE OF PROGRAM CREATION



- Program list

```

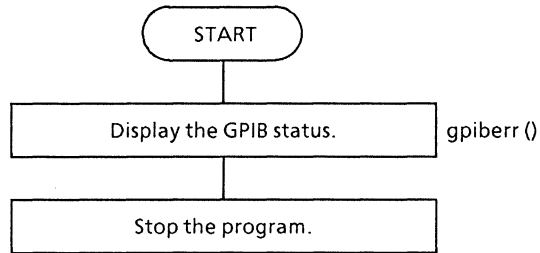
' ---- Procedure for initialize equipments and interface board ----
'
FUNCTION gpinit%
  CLS
  CALL IBFIND("GPIB0", GPIB0%)      'Open DEVICE (GPIB0)
  IF GPIB0% < 0 THEN
    PRINT "Configuration fail!!"
    PRINT "You need verify are hardware condition, and try
    again."
    ret% = 0
  ELSE
    CALL IBFIND("PPG", PPG%)        'Open DEVICE (PPG)
    IF PPG% < 0 THEN
      PRINT "Lost PPG address!!"
      PRINT "If you use a PPG, then verify configuration
      and environment."
      DEV% = 0
    ELSE
      DEV% = 1
    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

```

(2.9) SUB trap (msg\$): Processes errors.

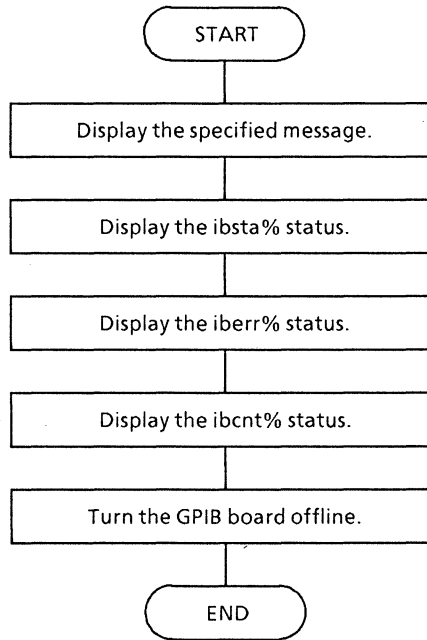
- Flowchart



- Program list

(2.10) SUB gpiberr (msg\$): Displays the STATIC: GPIB status.

- Flowchart



SECTION 10 EXAMPLE OF PROGRAM CREATION

- Program list

<Program start>

The procedures used to process the above common functions and to start the sample programs (1) to (8) 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.

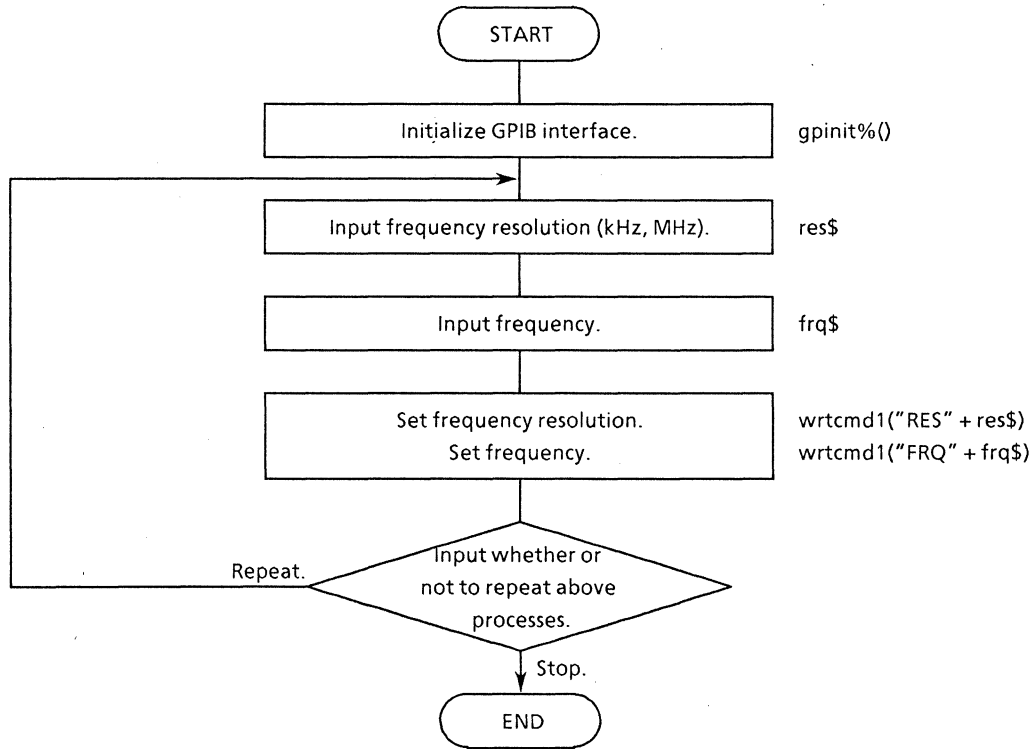
(Procedure 4): Open Run from the menu bar, and select "Set Main Module", then make the sample program loaded in procedure 3 the main module.

(Procedure 5): Open Run from the menu bar and execute "Start".

(☞ Refer to the Quick Basic Instruction Manual for details.)

(1) Frequency Setting 1 (Point Setting)

This program sets the frequency and frequency resolution of the internal synthesizer.



- Program list

```

REM $INCLUDE: 'c:\wat-gpib\qbasic\qbdecl.bas'

COMMON SHARED DEV%, GPIB0%, PPG%

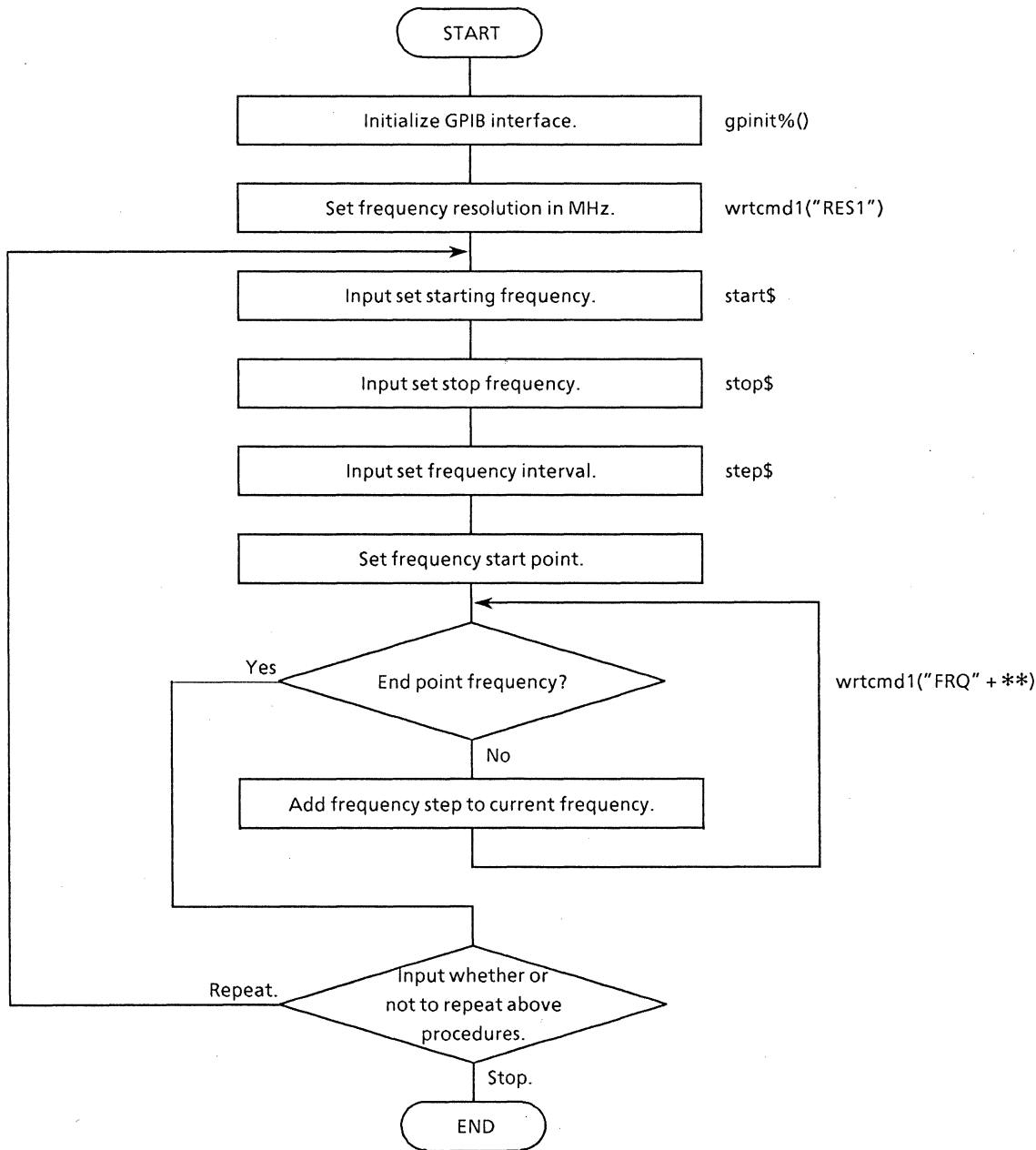
DECLARE SUB waidly (tim!)
DECLARE SUB wrtcmdl (w$)
DECLARE FUNCTION gpinit% ()

'*****
'*      MP1761B / MP1763B  FREQUENCY SAMPLE SOFT_1      *
'*****
|
|-----|
|                   MAIN ROUTINE                       |
|-----|
|
CLS
IF gpinit% <> 0 THEN      'setup interface
|
|   DO
|       CLS
|       LOCATE 3, 1
|       GOSUB setfrql
|       GOSUB dset
|       PRINT ""
|       INPUT " SET NEXT DATA [ Y or N ] ?", loop$
|
|   LOOP UNTIL loop$ = "n" OR loop$ = "N"
END IF
STOP
|
|-----|
|                   SUB ROUTINE                       |
|-----|
|
setfrql: '----- INPUT DATA CONDITIONS
|
PRINT ""
PRINT " ***** MP1761B / MP1763B FREQUENCY_1 SAMPLE PROGRAM *****"
PRINT ""
|
INPUT " FREQUENCY RESOLUTION [ KHz = 0, MHz = 1 ]?", res$
PRINT ""
|   IF res$ = "0" THEN
|       INPUT " FREQUENCY DATA [ KHz : 50000 to 12500000 ] ?", frq$
|   ELSE
|       INPUT " FREQUENCY DATA [ MHz : 50 to 12500 ] ?", frq$
|   END IF
RETURN
|
|
dset: '----- OUTPUT DATA CONDITIONS
|
CALL wrtcmdl("RES " + res$)      ' Set Resolution mode
CALL wrtcmdl("FRQ " + frq$)     ' Set Frequency
RETURN
|
END

```

(2) Frequency Setting 2 (Fixed Range Setting)

This program sets the frequency between two specified points in the specified interval.



- Program list

```

REM $INCLUDE: 'c:\vat-gpib\qbasic\qbdecl.bas'

COMMON SHARED DEV%, GPIB0%, PPG%

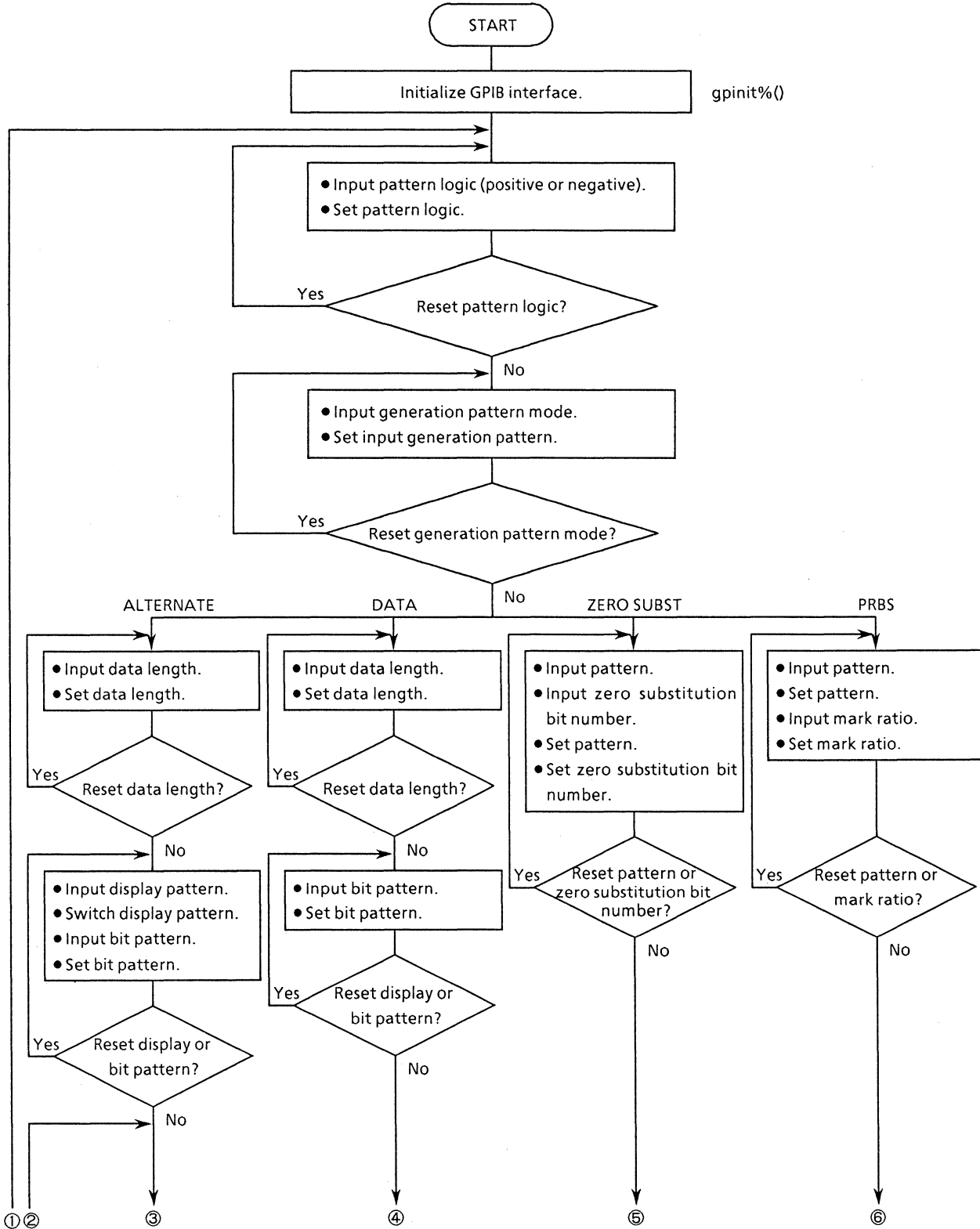
DECLARE SUB waidly (tim!)
DECLARE SUB wrtcmdl (w$)
DECLARE FUNCTION gpinit% ()

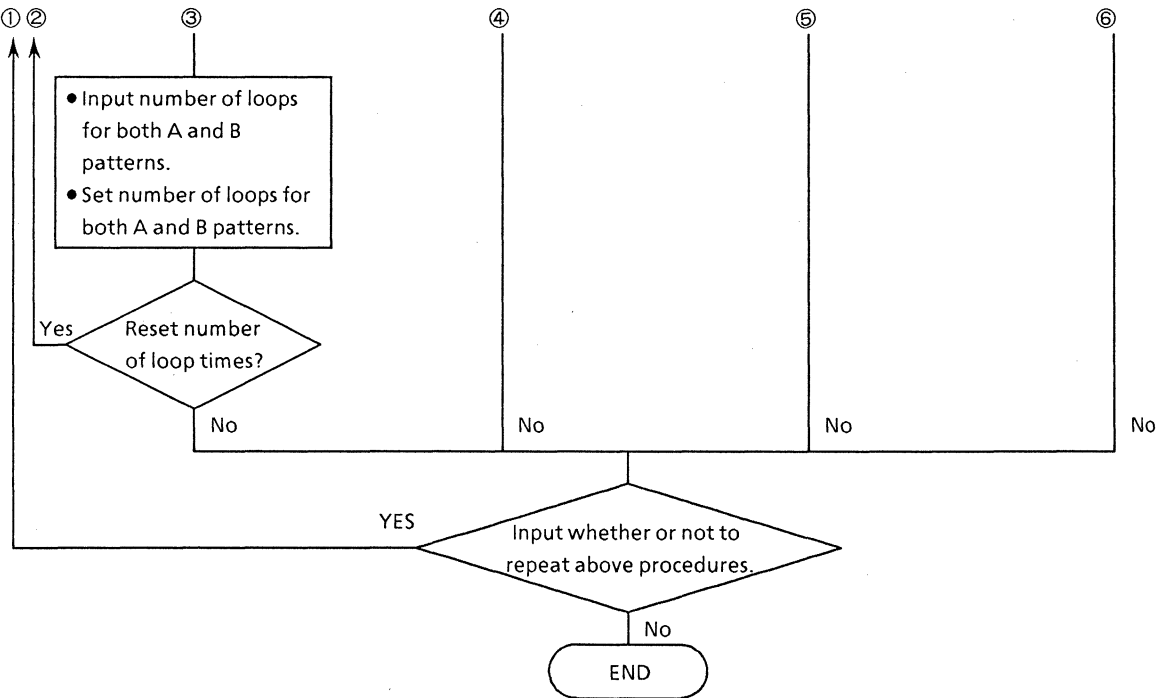
'*****
'*      MP1761B / MP1763B  FREQUENCY SAMPLE SOFT      *
'*****
|
|-----|
|              MAIN ROUTINE              |
|-----|
CLS
IF gpinit% <> 0 THEN      'setup interface
|
CALL wrtcmdl("RES 1")
|
  DO
    CLS
    LOCATE 3, 1
    GOSUB setfrq2
    GOSUB dset
    |
    PRINT ""
    INPUT " SET NEXT DATA [ Y or N ] ?", loop$
    |
    LOOP UNTIL loop$ = "N" OR loop$ = "n"
  END IF
STOP
|
|-----|
|              SUB ROUTINE              |
|-----|
|
setfrq2: '----- INPUT DATA CONDITIONS
|
PRINT ""
PRINT " ***** MP1761B / MP1763B FREQUENCY_2 SAMPLE PROGRAM *****"
PRINT ""
|
INPUT " START FREQUENCY [ 50 to 12500 : MHz ]? ", startf$
PRINT ""
INPUT " STOP  FREQUENCY [ 50 to 12500 : MHz ]? ", stopf$
PRINT ""
INPUT " FREQUENCY STEP  [ MHz ]? ", stepf$
|
RETURN
|
dset:    '----- OUTPUT DATA CONDITIONS
|
FOR I = VAL(startf$) TO VAL(stopf$) STEP VAL(stepf$)
      CALL wrtcmdl("FRQ " + STR$(I))      ' Set Frequency
      CALL waidly(.1)
NEXT I
RETURN
|
END

```

(3) Generation Pattern Setting

This program performs the settings related to the generation pattern. First, the pattern logic and generation pattern are selected and then each setting is performed for the selected pattern.





SECTION 10 EXAMPLE OF PROGRAM CREATION

• Program list

```

DECLARE SUB ClearDisp (p%, l%)
REM $INCLUDE: 'c:\mat-gpib\qbasic\qbdecl.bas'

COMMON SHARED DEV%, GPIB0%, PPG%

DECLARE SUB waidly (tim!)
DECLARE SUB wrtcmdl (w$)
DECLARE FUNCTION gpinit% ()

'*****
'*
'*      MP1761B / MP1763B PATTERN SET SAMPLE SOFT      *
'*
'******
'
'-----
'
'              MAIN ROUTINE
'-----
'
IF gpinit% <> 0 THEN      'setup interface
'
  DO
    GOSUB pattern
    PRINT ""
    INPUT " SET NEXT DATA [ YES = ENTER, NO = 1 ] ?", loop$
    LOOP UNTIL loop$ = "1"
  END IF
  STOP
'
'-----
'
'              SUB ROUTINE
'-----
'
pattern: '----- INPUT DATA CONDITIONS
'
L = 4
CLS
PRINT ""
PRINT " ***** MP1761B / MP1763B PATTERN SET SAMPLE PROGRAM *****"
'
DO
  LOCATE L, 1
  INPUT " LOGIC MODE [ POSITIVE = 0, NEGATIVE = 1 ]?", lgc$
  wrtcmdl ("LGC " + lgc$)
  INPUT " Do you wish to change LOGIC (Y OR N) ?", yes$
  LOOP UNTIL yes$ = "N" OR yes$ = "n"

CALL ClearDisp(5, 1)
L = L + 2

DO
  DO
    LOCATE L, 1
    INPUT " PATTERN MODE [ ALTN=0, DATA=1, Z.S.=2, PRBS=3 ] ?", pat$
    LOOP UNTIL pat$ = "0" OR pat$ = "1" OR pat$ = "2" OR pat$ = "3"
    wrtcmdl ("PTS " + pat$)
  '
  INPUT " Do you wish to change PATTERN (Y OR N) ?", yes$
  LOOP UNTIL yes$ = "N" OR yes$ = "n"

```

```

CALL ClearDisp(7, 1)
L = L + 2

IF pat$ = "3" OR pat$ = "2" THEN          '   PRBS OR Z.S. PATTERN
DO
    LOCATE L, 1
    IF pat$ = "3" THEN
        PRINT " PRBS MODE [PN7 =2, PN9 =3, PN11=5, PN15=6,"
        INPUT "                PN20=7, PN23=8, PN31=9          ]?", ptn$
        wrtcmdl ("PTN " + ptn$)
        '
        PRINT " MARK RATIO (POSITIVE) [0/8=0,1/8=1,1/4=2,1/2=3]"
        INPUT "                (NEGATIVE) [8/8=0,7/8=1,3/4=2,1/2=3]?",
mrk$
        wrtcmdl ("MRK " + mrk$)
        '
        INPUT " Do you wish to change PRBS MODE & MARK RATIO (Y OR
N) ?", y$
    END IF
    IF pat$ = "2" THEN
        INPUT " Z.S. MODE [2^7=2, 2^9=3, 2^11=5, 2^15=6]?", ptn$
        INPUT " ZERO SUBSTITUTION LENGTH [ 1 to 32767 ] ?", zln$
        wrtcmdl ("PTN " + ptn$)
        wrtcmdl ("ZLN " + zln$)
        '
        INPUT " Do you wish to change Z.S MODE & Z.S. LENGTH (Y OR
N) ?", y$
    END IF
    LOOP UNTIL y$ = "N" OR y$ = "n"
ELSE          '   DATA OR ALTN PATTERN
    IF pat$ = "0" THEN GOSUB setaltn
    IF pat$ = "1" THEN GOSUB setdata
END IF
RETURN
'
setaltn: '----- SET ALTERNATE A/B PATTERN -----
'===== SET DATA LENGTH =====
DO
DO
    LOCATE L, 1
    INPUT " DATA LENGTH [ 128 to 4194304 ] ?", dln
    LOOP UNTIL dln >= 128 AND dln <= 4194304
    wrtcmdl ("DLN " + STR$(dln))
    '
    INPUT " Do you wish to change DATA LENGTH (Y OR N) ?", yes$
    LOOP UNTIL yes$ = "n" OR yes$ = "N"

CALL ClearDisp(9, 1)
L = L + 1
'
'===== SET ALTN PATTERN =====
DO
    LOCATE L + 2, 1
    FOR cl = 1 TO 6: PRINT SPACE$(78): NEXT cl
    LOCATE L + 1, 1
    INPUT " CHOSE ALTERNATE [ A = 0, B = 1 ] ?", alt$
    wrtcmdl ("ALT " + alt$)
    GOSUB setbit
    '
    INPUT " Do you wish to change A/B pattern & bit pattern (y or n) ?", yes$
    LOOP UNTIL yes$ = "n" OR yes$ = "N"
L = L + 2
'

```

SECTION 10 EXAMPLE OF PROGRAM CREATION

```

DO
  LOCATE L + 1, 1: FOR cl = 1 TO 2
  PRINT SPACE$(78)
  NEXT cl
  '
  LOCATE L, 1
  INPUT " A PATTERN LOOP TIME [ 1 to 127 ] ?", lpt$
  wrtcmdl ("ALT 0")
  wrtcmdl ("LPT " + lpt$)
  INPUT " B PATTERN LOOP TIME [ 1 to 127 ] ?", lpt$
  wrtcmdl ("ALT 1" + ";LPT " + lpt$)
  '
  INPUT " Do you wish to change LOOP TIME (Y OR N) ?", yes$
  LOOP UNTIL yes$ = "n" OR yes$ = "N"
  L = L + 3
  '
RETURN
'
setdata: '----- SET DATA PATTERN -----
DO
  DO
    LOCATE L, 1
    INPUT " DATA LENGTH [ 2 to 8388608 ] ?", dln
    LOOP UNTIL dln >= 2 AND dln <= 8388608
    wrtcmdl ("DLN " + STR$(dln))
    '
    INPUT " Do you wish to change DATA LENGTH (Y OR N) ?", yes$
    LOOP UNTIL yes$ = "n" OR yes$ = "N"
    LOCATE L + 1, 1: PRINT SPACE$(78)
    L = L + 2
    '===== SET DATA PATTEN =====
  DO
    LOCATE L + 1, 1: PRINT SPACE$(78)
    LOCATE L + 1, 1
    GOSUB setbit
    '
    INPUT " Do you wish to change BIT PATTERN (Y OR N) ?", yes$
    LOCATE L + 2, 1: FOR cl = 1 TO 5
    PRINT SPACE$(78)
    NEXT cl
  LOOP UNTIL yes$ = "N" OR yes$ = "n"
  '
RETURN
'

```



```

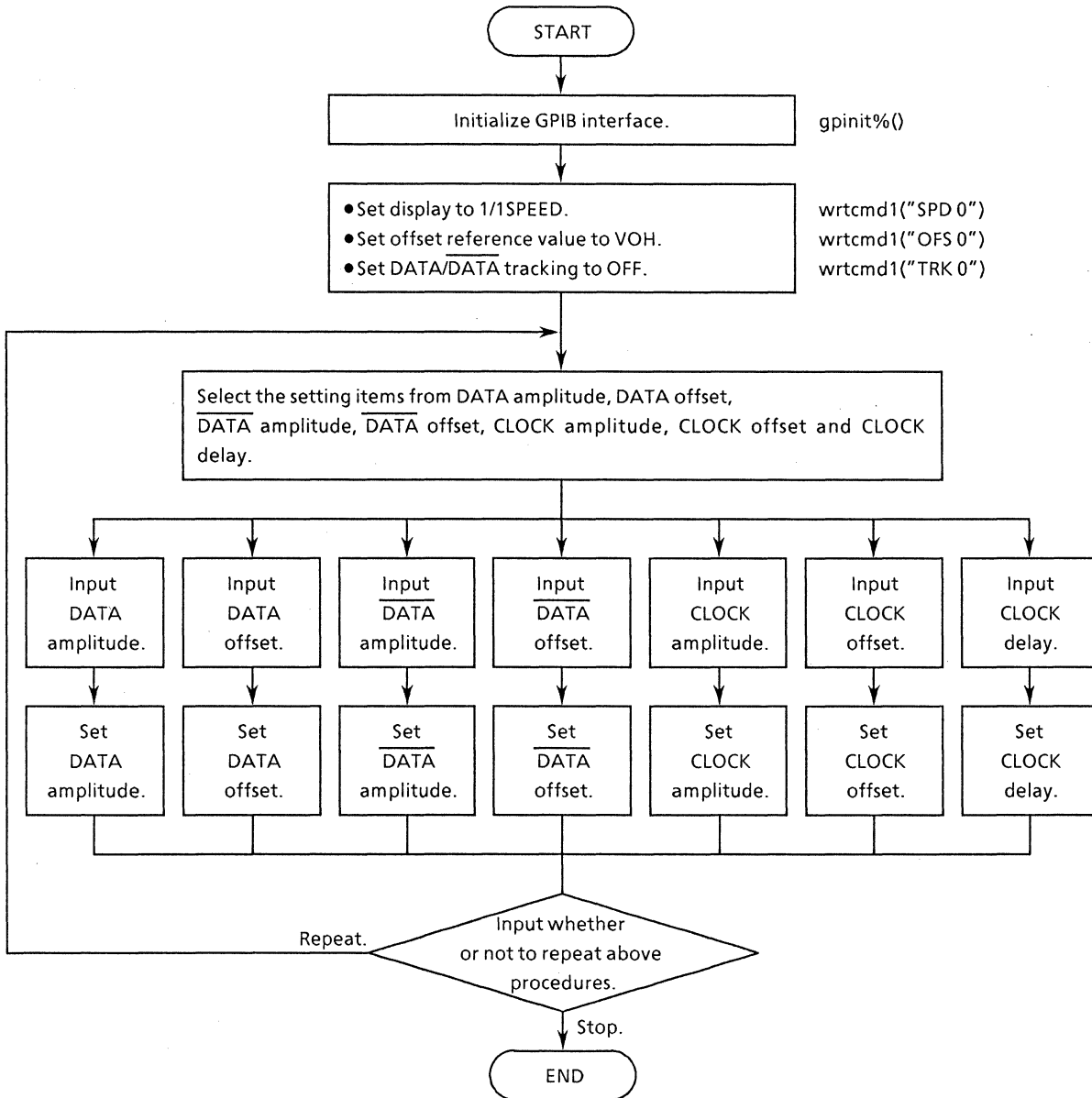
setbit: '----- SET BIT PATTERN -----'
INPUT " TOP PAGE TO SET ?:", page
'
j = 8
PRINT " You are able to choice data format of Hexadecimal or Decimal."
PRINT " Default data format is Hexadecimal."
'
BIT$ = ""
FOR k = 0 TO j - 1
  LOCATE L + 4, 1
  PRINT " Do you set bit-pattern of" + STR$(page + k) + " PAGE ? ";
  INPUT "(y or n)", yes$
  IF yes$ = "n" OR yse$ = "N" THEN
    EXIT FOR
  END IF
  '
  IF k <> 0 THEN BIT$ = BIT$ + ","
  '
  INPUT " Which do you choice format? (Hexadecimal / Decimal)", ftype$
  LOCATE L + 2, 1:
  FOR c1 = 1 TO 4
    PRINT SPACE$(78)
  NEXT c1
  LOCATE L + 2, 1
  PRINT SPACE$(78)
  IF ftype$ = "d" OR ftype$ = "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 ""
'
LOCATE L + 1, 1
FOR c1 = 1 TO 5: PRINT SPACE$(78): NEXT c1
LOCATE L + 2, 1
PRINT " PAG " + STR$(page) + ";BIT " + BIT$
wrtcmdl ("PAG " + STR$(page) + ";BIT " + BIT$)
RETURN

END

```

(4) Output signal setting

This program sets the CLOCK signal delay, and the DATA / $\overline{\text{DATA}}$ / CLOCK signal amplitude and offset.



- Program list

```

REM $INCLUDE: 'c:\vat-gpib\qbasic\qbdecl.bas'

COMMON SHARED DEV%, GPIB0%, PPG%

DECLARE SUB waidly (tim!)
DECLARE SUB wrtcmdl (w$)
DECLARE FUNCTION gpinit% ()

'*****
'*
'*      MP1761B / MP1763B  OUTPUT  SAMPLE SOFT      *
'*      ----- 1/1 SPEED, VOH, TRACKING OFF ----- *
'*
'*
'*****
|
|-----|
|
|              MAIN ROUTINE
|-----|
|
CLS
IF gpinit% <> 0 THEN      'setup interface
|
CALL wrtcmdl("SPD 0")    ' 1/1 SPEED
CALL wrtcmdl("OFS 0")    ' OFFSET VOH
CALL wrtcmdl("TRK 0")    ' DATA / NDATA TRACKING OFF
|
DO
    CLS
    LOCATE 3, 1
    GOSUB setout
    GOSUB dset
    |
    PRINT ""
    INPUT " SET NEXT DATA [ Y or N ] ?", loop$
    |
    LOOP UNTIL loop$ = "N" OR loop$ = "n"
END IF
STOP

```

SECTION 10 EXAMPLE OF PROGRAM CREATION

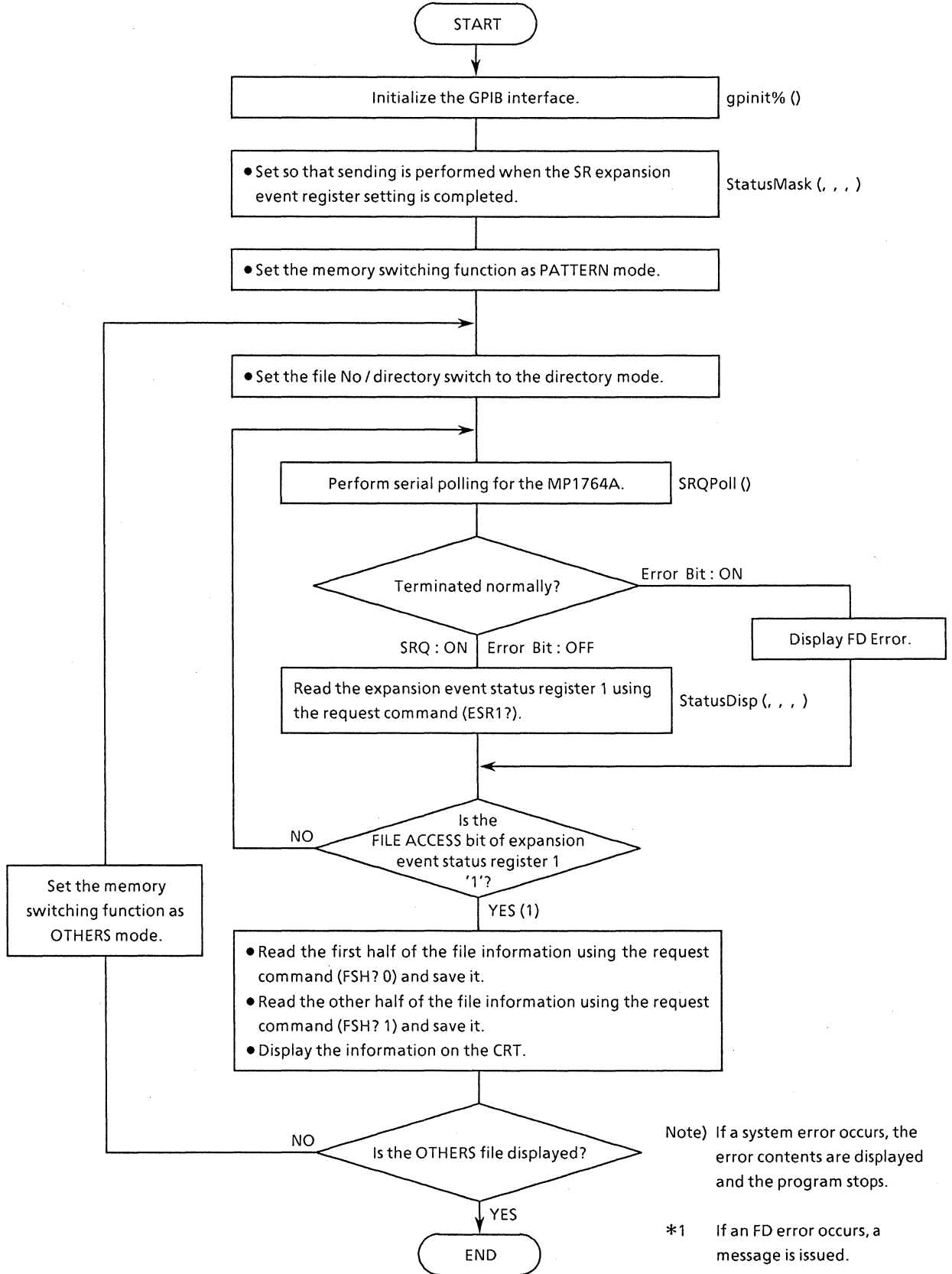
```

'
'-----
'                SUB ROUTINE
'-----
'
setout: '----- OUTPUT SIGNAL CONDITIONS
'
PRINT ""
PRINT " ***** MP1761B / MP1763B OUTPUT SIGNAL SET SAMPLE PROGRAM *****"
PRINT ""
'
PRINT " SETTING ITEM [ DATA AMPLITUDE = 1, DATA OFFSET = 2"
PRINT "                   NDATA AMPLITUDE = 3, NDATA OFFSET = 4"
PRINT "                   CLOCK AMPLITUDE = 5, CLOCK OFFSET = 6"
INPUT "                   CLOCK DELAY = 7 ] ? ", ITM
PRINT ""
'
RETURN
'
'
dset:   '----- SET OUTPUT CONDITIONS
'
IF ITM = 1 OR ITM = 2 THEN CALL wrtcmdl("DDS 0")
IF ITM = 3 OR ITM = 4 THEN CALL wrtcmdl("DDS 1")
'
IF ITM = 1 THEN
    INPUT " INPUT DATA AMPLITUDE [ 2.000 to 0.25 ] ", DAP$
    CALL wrtcmdl("DAP " + DAP$)
END IF
IF ITM = 2 THEN
    INPUT " INPUT DATA OFFSET [ 2.000 to -2.000 ] ", DOS$
    CALL wrtcmdl("DOS " + DOS$)
END IF
IF ITM = 3 THEN
    INPUT " INPUT NDATA AMPLITUDE [ 2.000 to 0.25 ] ", NAP$
    CALL wrtcmdl("NAP " + NAP$)
END IF
IF ITM = 4 THEN
    INPUT " INPUT NDATA OFFSET [ 2.000 to -2.000 ] ", NOS$
    CALL wrtcmdl("NOS " + NOS$)
END IF
IF ITM = 5 THEN
    INPUT " INPUT CLOCK AMPLITUDE [ 2.000 to 0.25 ] ", CAP$
    CALL wrtcmdl("CAP " + CAP$)
END IF
IF ITM = 6 THEN
    INPUT " INPUT CLOCK OFFSET [ 2.000 to -2.000 ] ", COS$
    CALL wrtcmdl("COS " + COS$)
END IF
IF ITM = 7 THEN
    INPUT " INPUT CLOCK DELAY [ 500 to -500 ] ", CDL$
    CALL wrtcmdl("CDL " + CDL$)
END IF
'
RETURN
'
END

```

(5) Reading of floppy disk file information

This program reads file directory information stored on a floppy disk and displays it on the CRT.



SECTION 10 EXAMPLE OF PROGRAM CREATION

• Program list

```

REM $INCLUDE: 'c:\vat-gpib\qbasic\qbdecl.bas'

COMMON SHARED DEV%, GPIB0%, PPG%

DECLARE SUB wrtcmdl (w$)
DECLARE SUB ErrPoll ()
DECLARE SUB StatusDisp (stb%, esr%, esr2%, esr3%)
DECLARE SUB StatusMask (s0%, s1%, s2%, s3%)
DECLARE FUNCTION itob$ (l%, V%)
DECLARE FUNCTION SRQPoll% ()
DECLARE FUNCTION gpinit% ()
DECLARE FUNCTION readcmdl$ ()

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 =====
        wrtcmdl ("MEM " + STR$(i))

        '==== Set FILE DIR mode =====
        wrtcmdl ("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 =====
wrtcmdl ("FDE?")
rd1$ = LEFT$(readcmdl$, 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
wrtcmdl ("FSH? 0")
rd1$ = LEFT$(readcmdl$, IBCNT% - 1)
wrtcmdl ("FSH? 1")
rd2$ = LEFT$(readcmdl$, 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

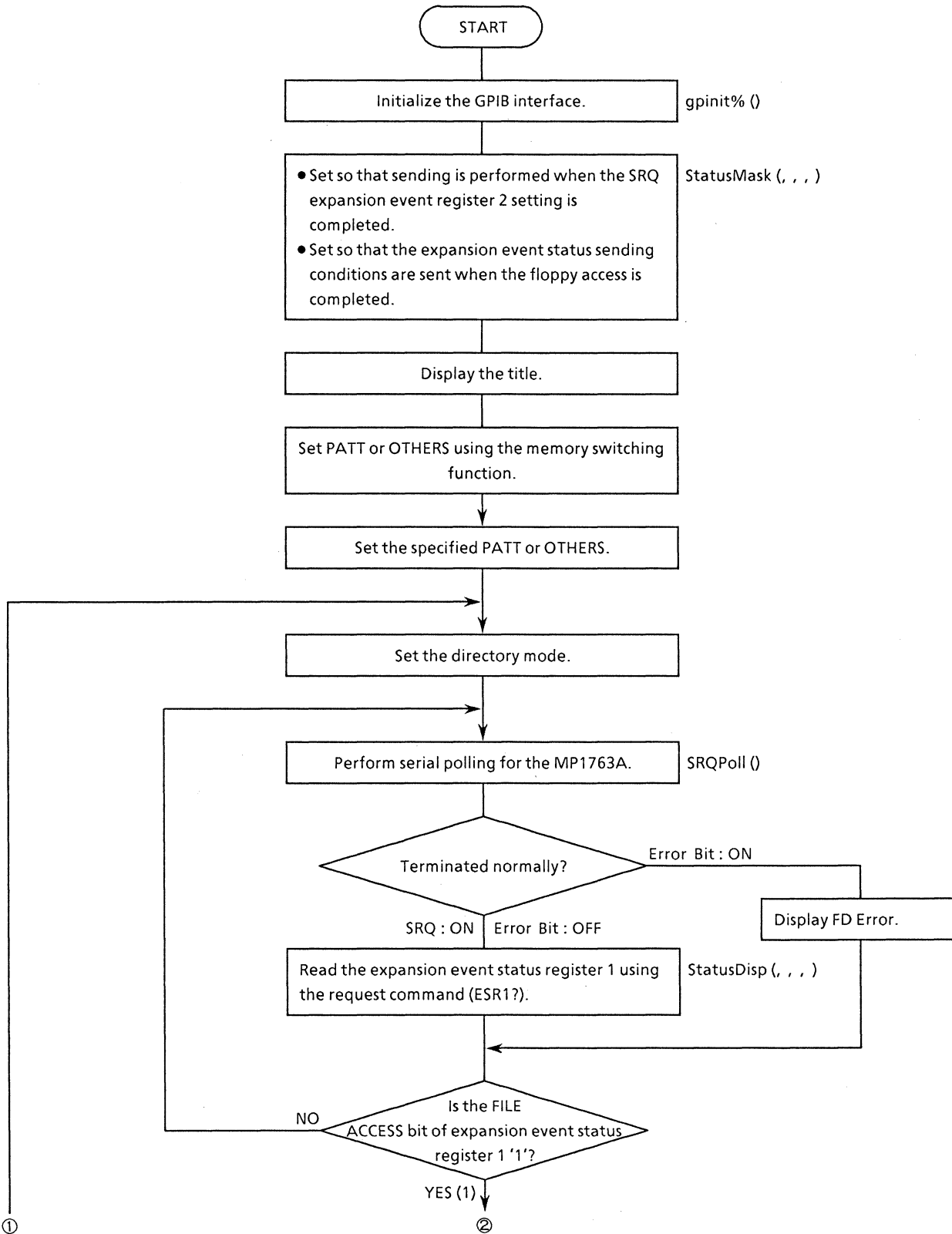
'print file no
PRINT "File name  :";MID$(rd1$,24)+", " + MID$(rd2$,24)
NEXT i
PRINT
INPUT "End of FD analize. Press 'Enter' to fin. "; f$
END IF

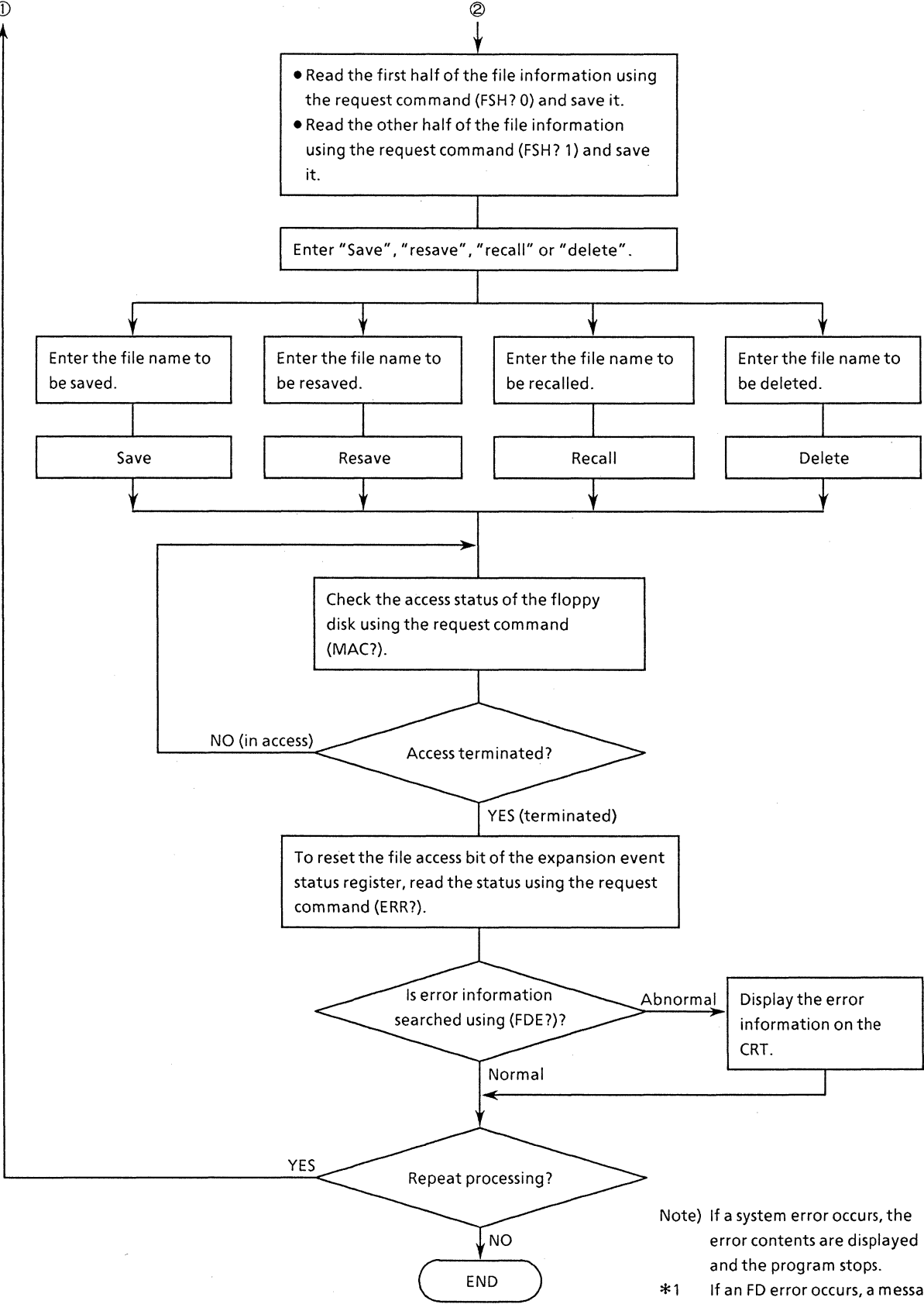
STOP

```

(6) 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.

SECTION 10 EXAMPLE OF PROGRAM CREATION

• Program list

```

DECLARE SUB ClearDisp (p%, l%)
REM $INCLUDE: 'c:\vat-gpib\qbasic\qbdecl.bas'

COMMON SHARED DEV%, GPIB0%, PPG%, ED%

DECLARE SUB waidly (tim!)
DECLARE SUB wrtcmdl (w$)
DECLARE SUB ErrPoll ()
DECLARE SUB StatusDisp (stb%, esr%, esr2%, esr3%)
DECLARE SUB StatusMask (s0%, s1%, s2%, s3%)
DECLARE FUNCTION itob$ (l%, v%)
DECLARE FUNCTION SRQPoll% ()
DECLARE FUNCTION gpinit% ()
DECLARE FUNCTION readcmdl$ ()

IF gpinit% <> 0 THEN                                'Setup interface

    '==== Set MSS status byte register ====
    CALL StatusMask(&HC, &H0, &H2, &H2)

    DO
        CLS
        PRINT "*** MP1761B/MP1763B 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"
        wrtcmdl ("MEM " + mem$)

        '==== Set FILE DIR mode =====
        wrtcmdl ("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$
    wrtcmdl ("SAV " + NO$)
    '
CASE 1
    INPUT "Enter file number for RESAVE:"; NO$
    wrtcmdl ("RSV " + NO$)
    '
CASE 2
    INPUT "Enter file number for RECALL:"; NO$
    wrtcmdl ("RCL " + NO$)
    '
CASE 3
    INPUT "Enter file number for DELEAT:"; NO$
    wrtcmdl ("DEL " + NO$)
    '
END SELECT
GOSUB Faccess
GOSUB Fderr
CALL ClearDisp(17, 1)

'=====  

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: '=====  

'
DO
    CALL StatusDisp(dmy%, dmy1%, dmy2%, dmy3%)
    waidly (1)
    wrtcmdl ("MAC?")
    RD$ = LEFT$(readcmdl$, IBCNT% - 1)
    LOOP UNTIL MID$(RD$, 1, 5) = "MAC 0"
    '
RETURN
'

```

SECTION 10 EXAMPLE OF PROGRAM CREATION

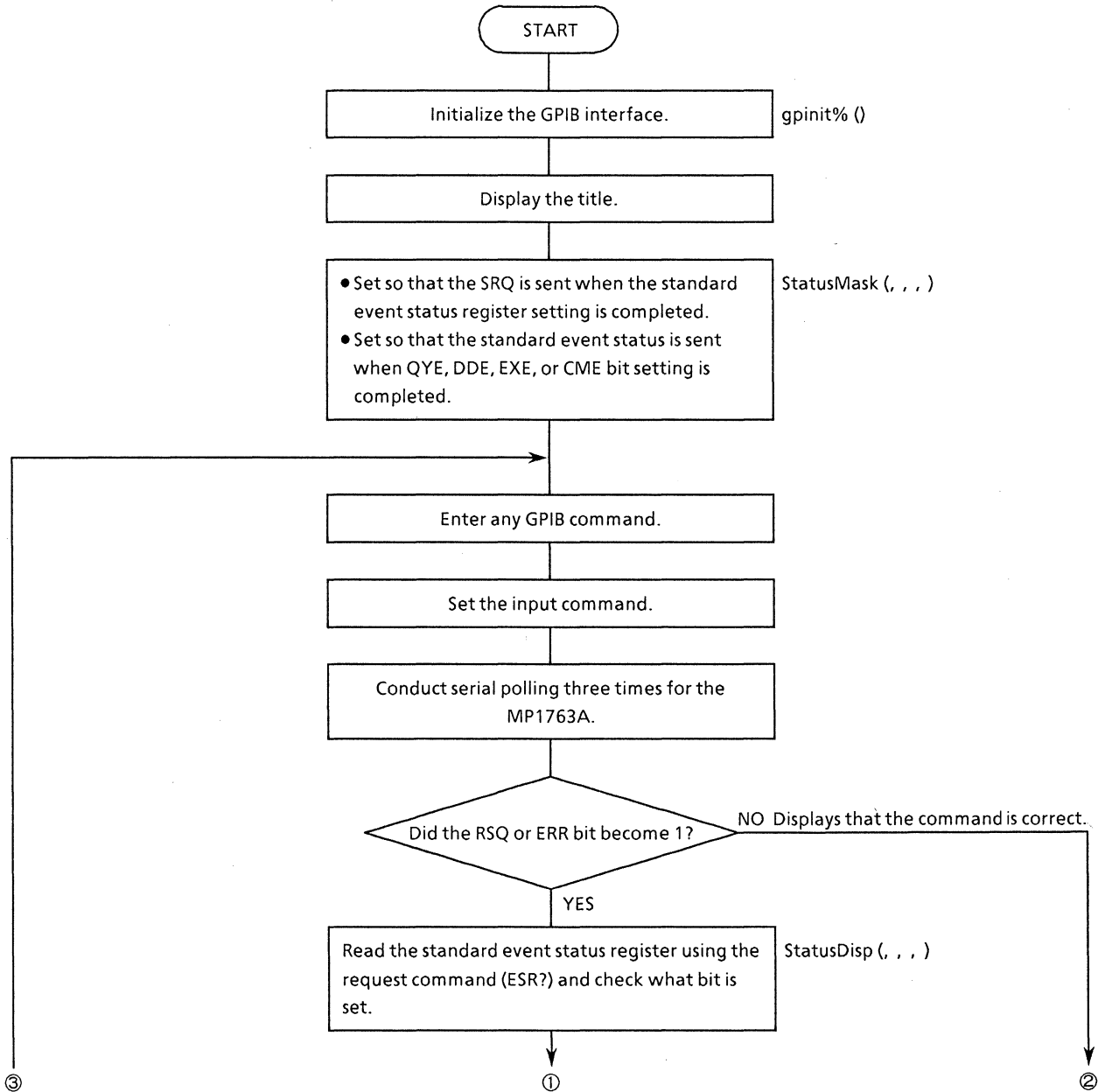
```

Fderr: '===== FD error message =====
      '
      CALL wrtcmdl("FDE?")
      RD$ = LEFT$(readcmdl$, 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

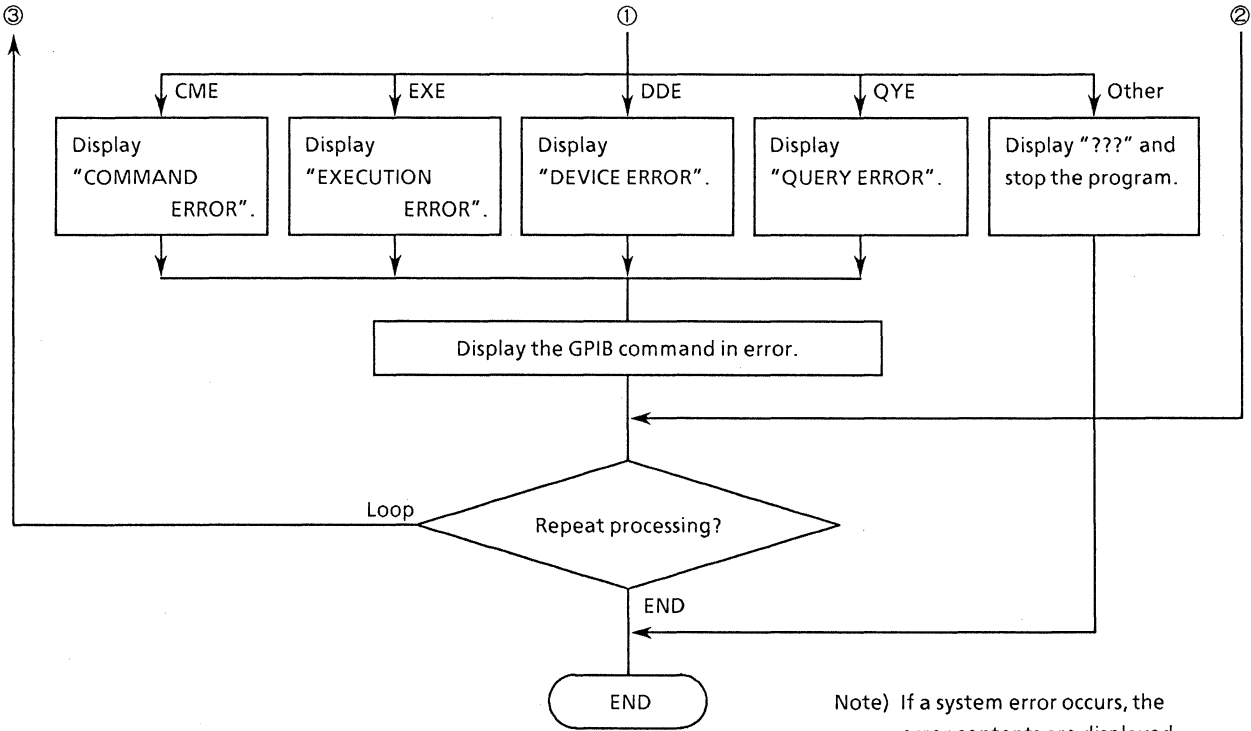
```

(7) Standard status byte (4 types) checking

This program checks the standard status bytes (QYE, DDE, EXE, and CME bits).



SECTION 10 EXAMPLE OF PROGRAM CREATION



Note) If a system error occurs, the error contents are displayed and the program stops.

- Program list

```

REM $INCLUDE: 'c:\vat-gpib\qbasic\qbdecl.bas'

COMMON SHARED DEV%, GPIB0%, PPG%

DECLARE SUB waidly (tim!)
DECLARE SUB wrtcmdl (WRT$)
DECLARE SUB trap ()
DECLARE SUB ClearDisp (p%, l%)
DECLARE SUB StatusDisp (stb%, esr%, esr2%, esr3%)
DECLARE SUB StatusMask (s0%, s1%, s2%, s3%)
DECLARE FUNCTION itob$ (l%, v%)
DECLARE FUNCTION gpinit% ()
DECLARE FUNCTION readcmdl$ ()

CLS
IF gpinit% <> 0 THEN                                'Setup interface

PRINT "*** MP1761B/MP1763B 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$)
    CALL wrtcmdl(com$)
    LOCATE 5, 1
    PRINT "Please enter some GPIB command(s):"
    "

    sta% = 0
    FOR i = 0 TO 2
        CALL IBRSP(PPG%, 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

```

SECTION 10 EXAMPLE OF PROGRAM CREATION

```

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%)

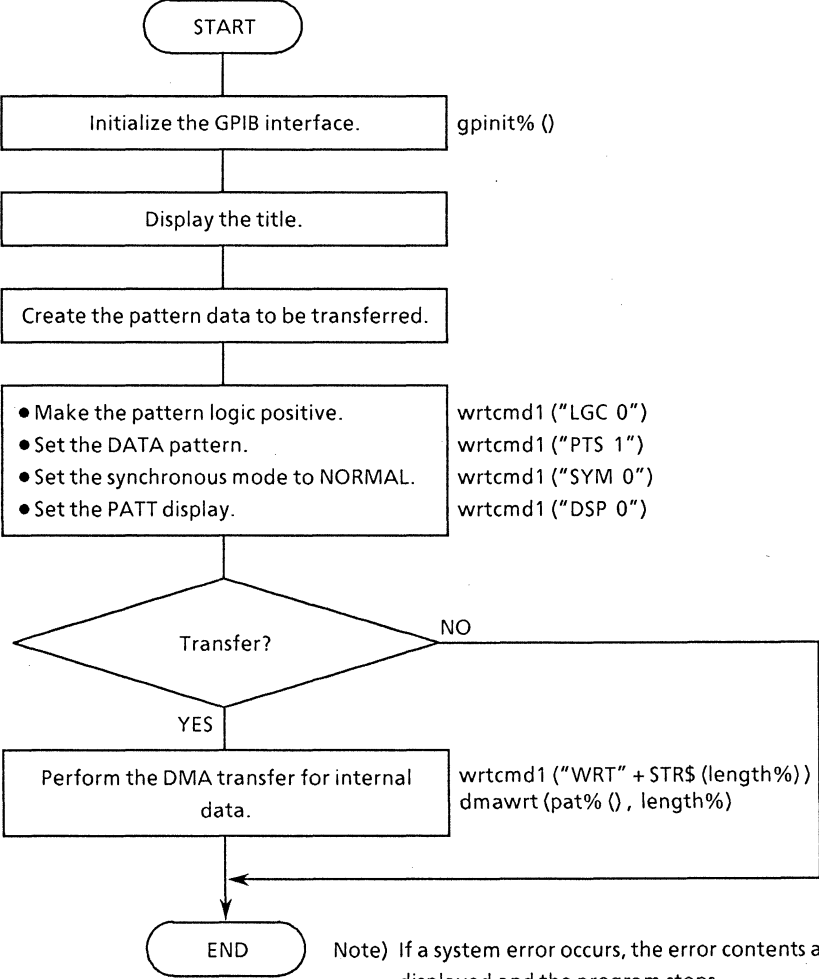
    CALL ClearDisp(8, 6): LOCATE 8, 1
    IF (reg% AND &H2) OR (reg% AND &H40) THEN PRINT
        " ??? ": STOP
    IF reg% AND &H4 THEN PRINT "* QUERY ERROR *"
    IF reg% AND &H8 THEN PRINT "* DEVICE ERROR *"
    IF reg% AND &H10 THEN PRINT "* EXECUTION ERROR *"
    IF reg% AND &H20 THEN PRINT "* COMMAND ERROR *"
ELSE
    LOCATE 7, 1
    PRINT "Command succed execution.                "
    CALL ClearDisp(9, 6)
    CALL ClearDisp(2, 3)
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

```


(8) Pattern data DMA transfer processing

This program performs the DMA transfer for pattern data.



SECTION 10 EXAMPLE OF PROGRAM CREATION

● Program list

```

REM $INCLUDE: 'c:\vat-gpib\qbasic\qbdecl.bas'

COMMON SHARED DEV%, gpib0%, PPG%

DECLARE SUB StatusMask (s0%, s1%, s2%, s3%)
DECLARE SUB StatusDisp (stb%, esr%, esr2%, esr3%)
DECLARE SUB wrtcmdl (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 "*** MP1761B/MP1763B 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 wrtcmdl("LGC 0")          'Pattern logic          : positive
  CALL wrtcmdl("PTS 1")         'Pattern              : data
  CALL wrtcmdl("DLN " + STR$(Dlength% * 16)): DATA Length

  CALL StatusDisp(dmy%, dmy1%, dmy2%, dmy3%)

  INPUT "Do you wish transmit PATTERN data? [Yes/No]:"; a$
  IF a$ = "y" OR a$ = "Y" THEN
    CALL wrtcmdl("WRT " + STR$(Dlength% * 2) + ",0")

    ' CALL ibconfig(gpib0%, 20, 1)
    ' CALL gpiberr("'ibconfig' execute status")

    CALL dmawrt(pat%(), Dlength%)
  END IF
END IF

STOP

```

APPENDIX A COMPATIBILITY WITH CONVENTIONAL INSTRUMENTS

When the mainframe uses programs generated by conventional instruments (MP1701B / MP1608A / MP1650A), some additional editing is required for programming.

(1) Status common command structure

MP1763B supports some of the common commands that conform to 488.2, but these commands are expressed without * in MP1701B.

Command	MP1701B	MP1763B
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) Other GPIB commands

Appendix table A-1 shows the correspondence between GPIB and MP1701B commands.

Re-edit programming by referring to the GPIB instruction manuals for each instrument.

- : Commands common with MP1701B
- × : Commands not common with MP1701B

Table A-1 Table of Device Messages

Function	Control message		Data request message	Device message details		Compati- bility with MP1701B
	Header part	Numeric data part	Header part	No.	Page	
● INTERNAL CLOCK section						
Internal clock frequency	FRQ	NR1 format	FRQ?	1	P9-20	○
Internal clock resolution switching	RES	NR1 format	RES?	2	P9-22	○
● MEMORY section						
File No./directory mode switching	FIL	NR1 format	FIL?	3	P9-24	○
FD data recall	RCL	NR1 format	—	4	P9-25	○
FD data delete	DEL	NR1 format	—	5	P9-26	×
FD data save	SAV	NR1 format	—	6	P9-27	○
FD data resave	RSV	NR1 format	—	7	P9-28	○
Memory mode switch	MEM	NR1 format	MEM?	8	P9-29	○
FD format	FDF	—	—	9	P9-30	×
File content search	—	—	FSH?	10	P9-31	○
Memory FD mode	—	—	FMD?	11	P9-33	×
FD access status	—	—	MAC?	12	P9-34	○
FD error message	—	—	FDE?	13	P9-35	×
● PATTERN section						
Pattern logic	LGC	NR1 format	LGC?	14	P9-37	○
Generation pattern selection	PTS	NR1 format	PTS?	15	P9-38	○
ZERO SUBST / PRBS stage	PTN	NR1 format	PTN?	16	P9-39	○
PRBS mark ratio	MRK	NR1 format	MRK?	17	P9-40	○
Alternate pattern A / B display switch	ALT	NR1 format	ALT?	18	P9-41	×
Error insertion	EAD	NR1 format	EAD?	19	P9-42	×
Alternate A / B loop times	LPT	NR1 format	LPT?	20	P9-43	×
Data length	DLN	NR1 format	DLN?	21	P9-44	×
ZERO SUBST length	ZLN	NR1 format	ZLN?	22	P9-46	×
Page number Pattern synchronous trigger position	PAG ADR	NR1 format NR1 format	PAG? ADR?	23	P9-47	×
Pattern bit	BIT	NR1 format HEX format	BIT?	24	P9-49	○

Table A-1 Table of Device Messages (contd.)

Function	Control message		Data request message	Device message details		Compati- bility with MP1701B
	Header part	Numeric data part	Header part	No.	Page	
● PATTERN section (contd.)						
Pattern data preset (all pages, all bits)	ALL	NR1 format	—	25	P9-51	○
Pattern data preset (1 page, all bits)	PST	NR1 format	—	26	P9-52	○
Pattern sync tigger position	PSP	NR1 format	PSP?	27	P9-53	×
Page number / pattern sync trigger position display switch	PPD	NR1 format	PPD?	28	P9-55	×
● OUTPUT section						
Data output termination voltage	DTM	NR1 format	DTM?	29	P9-57	○
Clock1 output termination voltage	CTM	NR1 format	CTM?	30	P9-58	○
Offset reference value	OFS	NR1 format	OFS?	31	P9-59	○
Data output amplitude	DAP	NR2 format	DAP?	32	P9-60	○
$\overline{\text{Data}}$ output amplitude	NAP	NR2 format	NAP?	33	P9-62	○
Data output offset	DOS	NR2 format	DOS?	34	P9-63	○
$\overline{\text{Data}}$ output offset	NOS	NR2 format	NOS?	35	P9-68	○
Clock1 output delay time	CDL	NR2 format	CDL?	36	P9-70	○
Clock1 output amplitude	CAP	NR2 format	CAP?	37	P9-71	○
Clock1 output offset	COS	NR2 format	COS?	38	P9-72	○
Output ON / OFF	OON	NR1 format	OOS?	39	P9-74	×
DATA / $\overline{\text{DATA}}$ display switch	DDS	NR1 format	DDS?	40	P9-75	×
DATA / $\overline{\text{DATA}}$ / tracking	TRK	NR1 format	TRK?	41	P9-76	○
1 / 1 SPEED, 1 / 4 SPEED display switch	SPD	NR1 format	SPD?	42	P9-77	○
● Front panel						
Sync signal output selection	SOP	NR1 format	SOP?	43	P9-79	×
● Back panel						
Error insertion channel	ECH	NR1 format	ECH?	44	P9-80	○
● Function switch						
Mark ratio AND bit shift No.	SFT	NR1 format	SFT?	45	P9-81	○
External error insertion	EEI	NR1 format	EEI?	46	P9-82	×
Alternate pattern A / B switch signal selection	APS	NR1 format	APS?	47	P9-83	×

Table A-1 Table of Device Messages (contd.)

Function	Control message		Data request message	Device message details		Compatibility with MP1701B
	Header part	Numeric data part	Header part	No	Page	
• Other						
Initialize	INI	—	—	48	P9-84	○
Pattern data input byte number	WRT	NR1 format	—	49	P9-85	○
Pattern data output byte number	RED	NR1 format	—	50	P9-86	○
Internal synthesizer PLL	—	—	PLL?	51	P9-87	○
Internal timer setting	RTM	NR1 format	RTM?	52	P9-88	○
Power cut, recovery status	—	—	PWI?	53	P9-89	○
Delay status	—	—	DLY?	54	P9-90	○

APPENDIX B PATTERN DMA TRANSFER

DMA, Direct Memory Access, transfers large amounts of data at high speed.

This mainframe has 2 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.

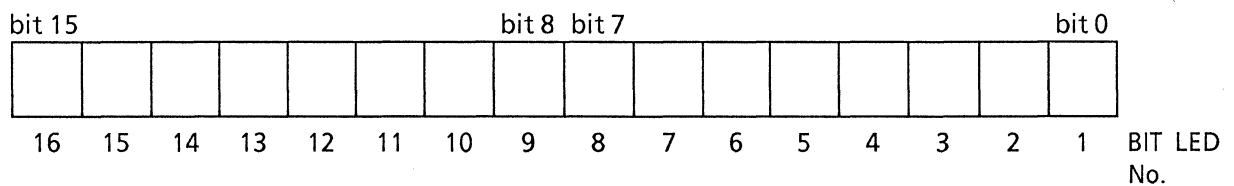
This command is used to notify the following information to the mainframe using WRT.

- 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.



When an odd byte is transferred, only the upper bits (bit 15 to bit 8) are specified.

APPENDIX B

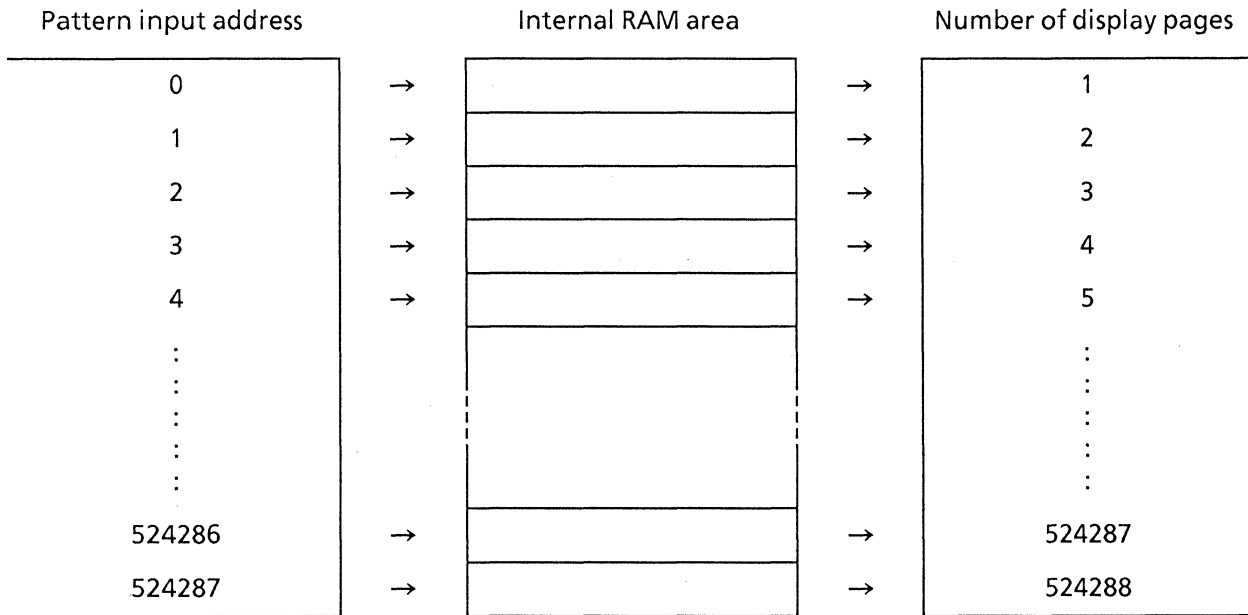
2) In which address in the mainframe internal RAM area, is the transferred pattern data stored?

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.



(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.

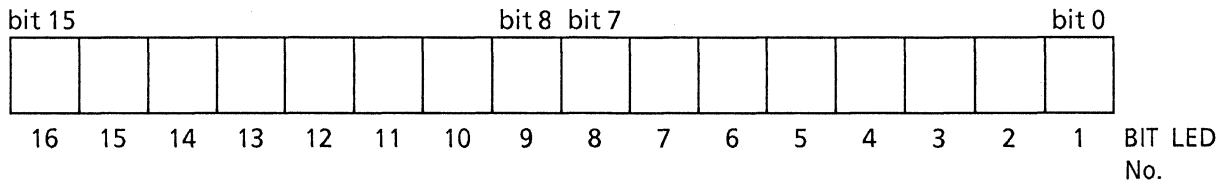
This command is used to notify the following information to the mainframe using WRT.

- 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.



When an odd byte is transferred, only the upper bits (bit 15 to bit 8) are specified.

APPENDIX B

2) In which address in the mainframe internal RAM area, is the transferred pattern data stored?

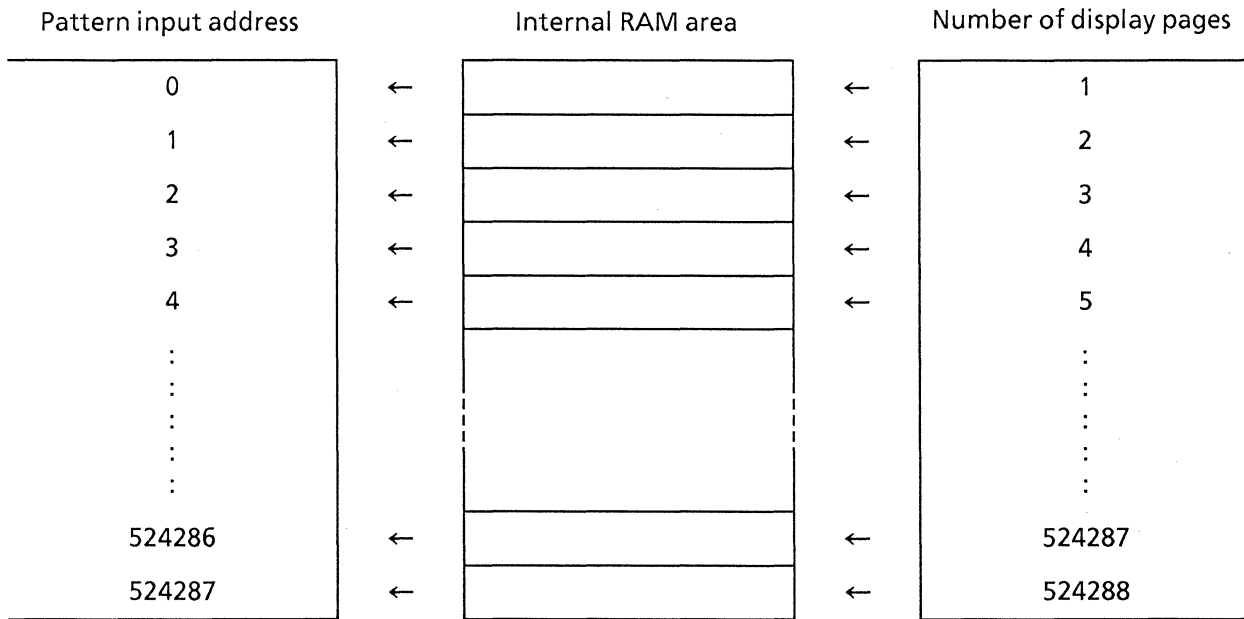
The RAM address of the mainframe is:

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.



APPENDIX C TABLES OF INITIAL VALUES

Appendix tables C-1 to C-5 shows initial values of MP1763B at the time of factory shipment.

Table C-1 Table of INTERNAL CLOCK section Initial Values

Function	Header part	Initial value	Device message details	
			Item No.	Page
● INTERNAL CLOCK section				
Internal clock frequency	FRQ	12.5 GHz	1	P9-20
Internal clock resolution switch	RES	MHz unit	2	P9-22

Table C-2 Table of MEMORY Section Initial values

Function	Header part	Initial value	Device message details	
			Item No.	Page
• MEMORY section				
File No. / Directory mode switching	FIL	File No.	3	P9-24
FD data recall	RCL	—	4	P9-25
FD data delete	DEL	—	5	P9-26
FD data save	SAV	—	6	P9-27
FD data resave	RSV	—	7	P9-28
Memory mode switch	MEM	PATT	8	P9-29
FD format	FDF	—	9	P9-30
File contents search	—	No data	10	P9-31
Memory / FD mode	—	1440 K	11	P9-33
FD access status	—	Non access	12	P9-34
FD error message	—	No error	13	P9-35

Table C-3 Table of PATTERN Section Initial Values

Function	Header part	Initial value	Device message details	
			Item No.	Page
● PATTERN section				
Pattern logic	LGC	Positive	14	P9-37
Generation pattern selection	PTS	PRBS	15	P9-38
ZERO SUBST / PRBS stage	PTN	PRBS : $2^{15} - 1$ Z. S. : 2^7	16	P9-39
PRBS mark ratio	MRK	1 / 2	17	P9-40
Alternate / pattern A / B display switch	ALT	A pattern	18	P9-41
Error insertion	EAD	Non external error insertion	19	P9-42
Alternate A / B loop times	LPT	1 time both A and B pattern	20	P9-43
Data length	DLN	Alternate : 128 bits Data : 2 bits	21	P9-44
ZERO SUBST length	ZLN	1 bit	22	P9-46
Page No. / pattern sync trigger position	PAG ADR	1 page	23	P9-47
Pattern bit	BIT	All bits 0	24	P9-49
Pattern sync trigger position	PSP	1 page	27	P9-53
Page No. / pattern sync trigger position display switch	PPD	Page No. display	28	P9-55

Table C-4 Table of OUTPUT Section Initial Values

Function	Header part	Initial value	Device Message Details	
			Item No.	Page
• OUTPUT section				
Data output termination voltage	DTM	GND	29	P9-57
Clock1 output termination voltage	CTM	GND	30	P9-58
Offset reference value	OFS	VOH	31	P9-59
Data output amplitude	DAP	1.0 Vp-p	32	P9-60
$\overline{\text{Data}}$ output amplitude	NAP	1.0 Vp-p	33	P9-62
Data output offset	DOS	0.0 V	34	P9-63
$\overline{\text{Data}}$ output offset	NOS	0.0 V	35	P9-68
Clock1 output delay time	CDL	0 ps	36	P9-70
Clock1 output amplitude	CAP	1.0 Vp-p	37	P9-71
Clock1 output offset	COS	0.0 V	38	P9-72
Output ON / OFF	OON	Output OFF	39	P9-74
DATA / $\overline{\text{DATA}}$ display switch	DDS	Data display	40	P9-75
DATA / $\overline{\text{DATA}}$ / tracking	TRK	Tracking OFF	41	P9-76
1 / 1 SPEED, 1 / 4 SPEED display switch	SPD	1 / 1 speed	42	P9-77

Table C-5 Table of Other Section Initial Values

Function	Header part	Initial value	Device message details	
			Item No.	Page
● Front panel Sync signal output selection	SOP	1 / 64 CLOCK	43	P9-79
● Back panel Error insertion channel	ECH	CH. 1	44	P9-80
● Function switch Mark ratio AND bit shift No.	SFT	1 bit shift	45	P9-81
External error insertion	EEI	External error insertion OFF	46	P9-82
Alternate pattern A / B switch signal selection	APS	Switch signal internal generation	47	P9-83
● Other Internal timer setting	RTM	00:00:00: 1 January 1995	52	P9-88

(Blank)

APPENDIX D TABLE OF TRACKING ITEMS

In this Appendix, MP1763B tracking items are explained.

Tracking denotes a function to send the MP1764A setting conditions to MP1763B 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 : Generation 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 : Generation pattern (Data) 3 : Data length 4 : Page setting 5 : Pattern bit (DMA transfer)
Zero subst pattern	1 : LOGIC 2 : Generation pattern 3 : Zero subst step 4 : Zero subst length 5 : Page setting
PRBS pattern	1 : LOGIC 2 : Generation pattern 3 : PRBS step 4 : PRBS mark ratio 5 : Page setting

(Blank)