

**GP-IB
OPERATION MANUAL
PULSE PATTERN GENERATOR
MP1652A**

ANRITSU CORPORATION

MANUAL CHANGE FOR THE MP1652A

Substitute MP1652A and MP1653A whenever MP1650A and MP1651A appear in this manual, as follows.

MP1650A → MP1652A

MP1651A → MP1653A

GP-IB
OPERATION MANUAL
PULSE PATTERN GENERATOR
MP1701B/MP1755A/MP1608A/MP1650A

MAY 1992
VER. I

ANRITSU CORPORATION

MAR.
1996

CERTIFICATION

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

WARRANTY

All parts of this product are warranted by Anritsu Corporation of Japan against defects in material or workmanship for a period of one year from the date of delivery.

In the event of a defect occurring during the warranty period, Anritsu Corporation will repair or replace this product within a reasonable period of time after notification, free-of-charge, provided that: it is returned to Anritsu; has not been misused; has not been damaged by an act of God; and that the user has followed the instructions in the operation manual.

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

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

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

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

Copyright © 1992 by ANRITSU CORPORATION.
Printed in Japan.
All rights reserved. No part of this manual may be reproduced in any form without written permission of ANRITSU CORPORATION.

WARNING

**NO OPERATOR SERVICEABLE PARTS INSIDE .
REFER SERVICING TO QUALIFIED PERSONNEL .**

CAUTION

**FOR CONTINUED FIRE PROTECTION REPLACE
ONLY WITH SPECIFIED TYPE AND RATED FUSE .**

TABLE OF CONTENTS

SECTION	1	GENERAL	1-1
SECTION	2	GP-IB ADDRESS SETTING	2-1
SECTION	3	GP-IB CABLE CONNECTION	3-1
SECTION	4	INTERFACE FUNCTION	4-1
SECTION	5	GP-IB COMMAND FORMAT	5-1
	5.1	COMMAND FORMAT	5-1
	5.2	Explanation of Symbols and Description	5-1
	5.3	Header Field (HR)	5-2
	5.4	Numeric Data Field (NR)	5-2
	5.4.1	NR1 format (integer type)	5-2
	5.4.2	NR2 format (real number type)	5-3
	5.4.3	Hexadecimal format (HEX type)	5-4
	5.4.4	Binary format (BIN type)	5-5
	5.5	Command Separator	5-6
	5.5.1	SR1 format (semicolon ;)	5-6
	5.5.2	SR1 format (comma ,)	5-6
	5.5.3	SR2 format (line feed LF)	5-7
	5.5.4	SR3 format (END message)	5-7
	5.6	Definition of Space	5-7
SECTION	6	EXECUTING PROGRAMS USED BY OTHER DEVICES	6-1
SECTION	7	GP-IB CONTROL COMMANDS	7-1
	7.1	GP-IB Control Command List	7-1
	7.2	GP-IB Control Command List in Alphabetic Order	7-9
	7.3	Correspondence with Common Commands	7-13
SECTION	8	DEVICE CLEAR FUNCTION	8-1

SECTION	9	DEVICE TRIGGER FUNCTION	9-1
SECTION	10	MESSAGE LENGTH	10-1
SECTION	11	CONTROL MESSAGE AND DATA REQUEST MESSAGE ...	11-1
	11.1	INTERNAL CLOCK Section	11-1
		① Operation clock	11-2
		② Internal clock frequency setting resolution (<u>Resolution mode</u>)	11-3
		③ Internal clock frequency (<u>Frequency</u>)	11-4
	11.2	PATTERN Section	11-7
		④ Output pattern logic (<u>Logic mode</u>)	11-8
		⑤ Output pattern (<u>Pattern mode</u>)	11-9
		⑥ Output pattern mark ratio (<u>Mark ratio mode</u>)	11-10
		⑦ Code error addition (<u>Error addition mode</u>)	11-11
		⑧ Number of words (<u>Number of words</u>)	11-12
		⑨ Word length (<u>Word length</u>)	11-14
		⑩ Data length (<u>Data length</u>)	11-15
		⑪ Number of page (<u>Page</u>)	11-17
		⑫ Pattern bit (<u>Pattern bit</u>)	11-19
		⑬ Number of pattern data input bytes (<u>Pattern data write</u>)	11-22
		⑭ Number of pattern data output bytes (<u>Pattern data read</u>)	11-24
		⑮ Preset (all bits on all pages) (<u>Preset all 0 or 1</u>)	11-26
		⑯ Preset (all bits on 1 page) (<u>Preset page 0 or 1</u>)	11-27
	11.3	OUTPUT Section	11-28
		⑰ DATA, $\overline{\text{DATA}}$, CLOCK output offset reference value (<u>Offset mode</u>)	11-29
		⑱ Front panel/rear panel output switching (<u>Speed</u>)	11-30

⑲	DATA output amplitude (<u>D</u> ata <u>a</u> mplitude)	11-31
⑳	DATA output offset (<u>D</u> ata <u>o</u> ffset)	11-32
㉑	DATA/DATA tracking (<u>D</u> ata/ <u>D</u> ata <u>t</u> racking)	11-41
㉒	DATA output amplitude (<u>I</u> nverted data <u>a</u> mplitude)	11-42
㉓	DATA output amplitude (<u>I</u> nverted data <u>o</u> ffset)	11-44
㉔	Delay time between CLOCK outputs (<u>C</u> lock <u>d</u> elay) ...	11-46
㉕	CLOCK output amplitude (<u>C</u> lock <u>a</u> mplitude)	11-48
㉖	CLOCK output offset (<u>C</u> lock <u>o</u> ffset)	11-49
11.4	MEMORY Section	11-52
㉗	Memory function switching (Memory mode ptn/other)	11-53
㉘	File No./directory mode switching (<u>F</u> ile no./directory mode)	11-54
㉙	Data recall (<u>D</u> ata <u>r</u> ecall)	11-56
㉚	Data save (<u>D</u> ata <u>s</u> ave)	11-57
㉛	Data resave (<u>D</u> ata <u>r</u> esave)	11-58
11.5	Rear Panel Section	11-59
㉜	Number of external pattern input channel (1/8 Speed <u>ch</u> select)	11-60
㉝	Error additional channel (<u>E</u> rror addition <u>ch</u> annel)	11-61
㉞	Number of mark ratio AND bit shifts (Mark ratio bit <u>sh</u> ift)	11-62
11.6	Other Section	11-63
㉟	Service request enable register (<u>S</u> ervice <u>r</u> equ <u>e</u> st enable register)	11-64
㊱	Status byte register (<u>S</u> tatus <u>b</u> yte register?)	11-66
㊲	Standard event status enable register (<u>E</u> vent <u>s</u> tatus <u>e</u> nable register [Standard])	11-68
㊳	Standard event status register (<u>E</u> vent <u>s</u> tatus <u>b</u> yte <u>r</u> egister? [Standard])	11-70

③⑨	Extended event status enable register (Event status enable register [Extension])	11-72
④①	Extended event status register (Event status byte register? [Extension])	11-74
④②	INITIALIZE (Initialize)	11-76
④③	Internal timer setting (Real time setting)	11-77
④④	Power failure, power recovery (Power failure interval?)	11-78
④⑤	PLL lock status (PLL unlock?)	11-80
④⑥	Delay status (Delay unlock?)	11-81
④⑦	Floppy disk access status	11-82

SECTION	12 GP-IB STATUS BYTE	12-1
	12.1 Status Byte Configuration	12-1
	12.2 Description of Each Register and Status Byte	12-3
	12.2.1 SRQ enable register	12-3
	12.2.2 SRQ status byte	12-5
	12.2.3 Standard event status enable register	12-8
	12.2.4 Standard event status byte	12-10
	12.2.5 Extended event status enable register	12-13
	12.2.6 Standard event status byte	12-10

SECTION	13 PATTERN DATA TRANSFER BY DMA	13-1
	13.1 DMA	13-1
	13.2 Commands for Number of Pattern Data Input Bytes and Number of Pattern Data Output Bytes	13-1
	13.2.1 Number of bytes of pattern data to be transferred	13-2
	13.2.2 Start address of the MP1701B/MP1755A/MP1608A/MP1650A internal RAM to store transferred pattern data and that to output pattern data to be transferred	13-3
	13.3 Editing Program using DMA	13-4

SECTION	14 SAMPLE PROGRAMS	14-1
	14.1 Sample Program List	14-1
	14.2 Controller Used	14-2
	14.3 Preparation for Program Execution	14-3
	14.4 Sample Program Using HP9000 as the Controller	14-4
	(1) Frequency setting	14-4
	(2) Frequency setting	14-8
	(3) Pattern setting	14-11
	(4) Output signal setting	14-15
	(5) Floppy disk file information read (Floppy disk access status check → serial polling)	14-17
	(6) Floppy disk file information read (Floppy disk access status check → status byte register service request)	14-21
	(7) Data save, resave, and recall	14-25
	(8) Standard event status bytes check (COMMAND ERROR bit check → serial polling)	14-31
	(9) Standard event status byte check (COMMAND ERROR bit check → status byte register service request)	14-35
	(10) Internal timer setting	14-39
	(11) Power failure and power recovery status check	14-42
	(12) Number of external pattern input channel setting	14-44
	(13) Error addition channel setting	14-46
	(14) Number of mark ratio AND bit shifts setting	14-51
	(15) Pattern data transfer by DMA	14-54
	(16) Pattern data transfer by DMA	14-59

14.5	Sample Programs Using J3100 as the	
	Controller	14-63
(1)	Frequency setting	14-64
(2)	Frequency setting	14-67
(3)	Pattern setting	14-71
(4)	Output signal setting	14-76
(5)	Floppy disk file information read (Floppy disk access status check → serial polling)	14-79
(6)	Floppy disk file information read (Floppy disk access status check → status byte register service request)	14-83
(7)	Data save, resave, and recall	14-87
(8)	Standard event status bytes check (COMMAND ERROR bit check → serial polling)	14-93
(9)	Standard event status byte check (COMMAND ERROR bit check → status byte register service request)	14-98
(10)	Internal timer setting	14-103
(11)	Power failure and power recovery status check	14-106
(12)	Number of external pattern input channel setting	14-109
(13)	Error addition channel setting	14-113
(14)	Number of mark ratio AND bit shifts setting ..	14-117

SECTION 1 GENERAL

The MP1701B/MP1755A/MP1608A/MP1650A has an IEEE Std. 488.2 GP-IB (General Purpose Interface Bus).

Control commands that are equivalent to the common commands (Common Command: header with *) defined by IEEE Std. 488.2 are provided.

For the relationship between the IEEE common commands and the MP1701B/MP1755A/MP1608A/MP1650A commands, see paragraph 7.3.

Also, the standard event status byte uses only the necessary functions.

In addition to the programmable pattern setting and the set state outputting via GP-IB, and the output interface parameter setting and the set status outputting; the set pattern and set parameter data can also be stored to and read from a 3.5-inch floppy disk.

The MP1701B/MP1755A/MP1608A/MP1650A also has a DMA function for high-speed transfer of large volumes of pattern data.

SECTION 2

GP-IB ADDRESS SETTING

With systems that use GP-IB, a unique address must be set for each device connected to the bus.

The MP1701B/MP1755A/MP1608A/MP1650A GP-IB address is set by the GP-IB ADDRESS switches on the rear panel.

Addresses from 0 to 30 can be set by the five switches.

The five address switches (named 1 to 5) have different weights of 1, 2, 4, 8, and 16, respectively,

For example to set address 11 (= 1 + 2 + 8), set the switches as follows:

Switch	1	2	3	4	5
Weight	1	2	4	8	16
Setting	1	1	0	1	0

When all the switches are set to 1 ($1 + 2 + 4 + 8 + 16 = 31$), the address is assumed to be 0.

Note: The MP1701B/MP1755A/MP1608A/MP1650A always checks the address switch status, so address switch change is always recognized.

SECTION 3

GP-IB CABLE CONNECTION

GP-IB cable is connected to the GP-IB connector on the rear panel by fastening the screws on the cable connector.

One GP-IB system can contain up to 15 devices.

However, the following restriction applies to the cable length.

(Total cable length) $\leq 2\text{m} * (\text{number of devices})$

The maximum total length of all cables is 20m.

The MP1701B/MP1755A/MP1608A/MP1650A GP-IB connector is shown on the next page.

● Pin connections

Pin No.	Signal name	Pin No.	Signal name
1	DIO1	13	DIO5
2	DIO2	14	DIO6
3	DIO3	15	DIO7
4	DIO4	16	DIO8
5	EOI	17	REN
6	DAV	18	Signal ground
7	NRFD	19	Signal ground
8	NDAC	20	Signal ground
9	IFC	21	Signal ground
10	SRQ	22	Signal ground
11	ATN	23	Signal ground
12	Shield	24	Signal ground

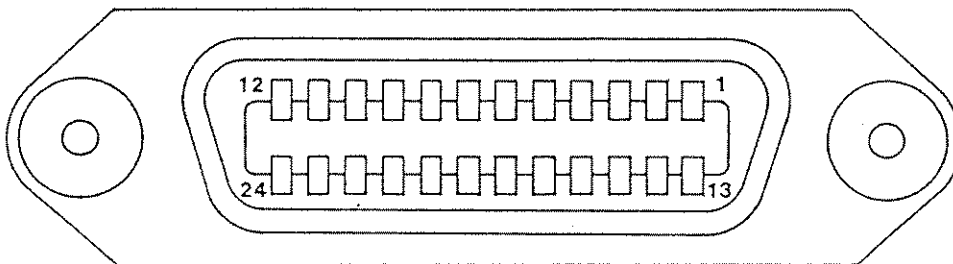


Fig. 3-1 GP-IB Connector

SECTION 4 INTERFACE FUNCTION

The GP-IB interface functions of the MP1701B/MP1755A/MP1608A/MP1650A are listed in Table 4-1.

Table 4-1 GP-IB Interface Functions

Function	Symbol	Explanation
Source handshake	SH1	All source handshake functions provided
Acceptor handshake	AH1	All acceptor handshake functions provided
Talker	T6	Basic talker functions provided Serial polling function provided Talk only function not provided Talker designation clear function by listener designation provided
Extended talker	TE0	Extended talker function not provided
Listener	L3	Basic listener functions provided Listen only mode function provided Listener designation clear function by talker designation provided
Extender listener	LE0	Extended listener function not provided
Service request	SR1	All service request functions provided
Remote/local	RL1	All remote local functions provided
Parallel polling	PP0	Parallel polling function not provided
Device clear	DC1	All device clear functions provided
Device trigger	DT0	Device trigger functions not provided
Controller	C0	System controller function not provided

SECTION 5 GP-IB COMMAND FORMAT

5.1 COMMAND FORMAT

The MP1701B/MP1755A/MP1608A/MP1650A GP-IB command format consists of a header field that indicates the command type, and a numeric data field that indicates spaces and values.

Use at least one space after the header field to separate it from the numeric data field.

In the data request message format, the header field, which shows the command type, is followed by [?].

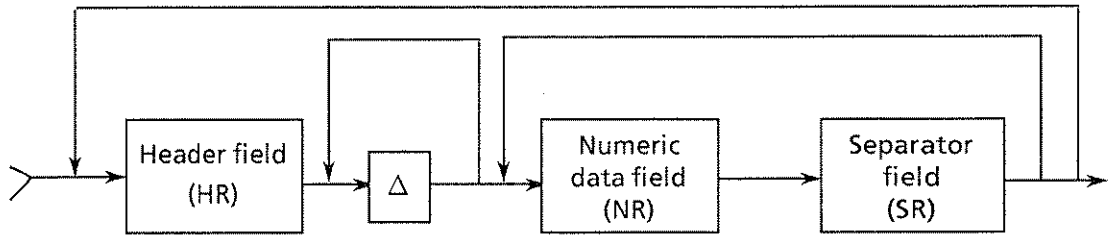

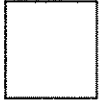





Fig. 5-1 Basic Command Format

5.2 Explanation of Symbols and Description

The method using the figure shown in Fig. 5-1 and the format figures shown in paragraph 5.3 are described below.

- 
: Indicates data element.
- 
: Indicates the data element of one character.
- 
: Indicates connection of data elements.
- 
: Indicates that data elements in this area can be repeated.
- 
: Indicates that this area can be omitted.

5.3 Header Field (HR)

The header field has an HR2 format consisting of three alphabetic characters (A to Z, a to z).

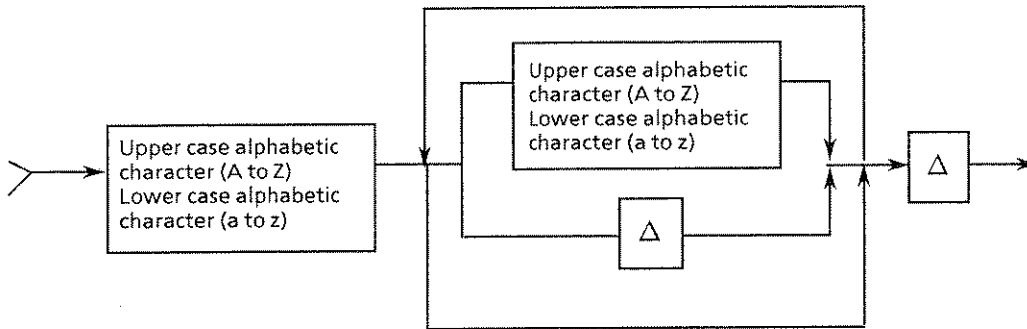


Fig. 5-2 HR2 Format

5.4 Numeric Data Field (NR)

The numeric data field has four formats depending on the command: NR1 format (integer type), NR2 format (real number type), Hexadecimal format (HEX type), Binary format (BIN type).

5.4.1 NR1 format (integer type)

NR1 format is an integer type numeric representation format. It consists of a one character sign (+, -) followed by numerics (0 to 9). The + sign can be omitted, or replaced by a space. Leading spaces can also be inserted to align the number of digits of data, etc.

Examples:

- General representation +1234, - 567, +0
- + sign omitted +1234 → 1234
- + sign replaced +1234 → Δ1234
- Spaces inserted +1234 → ΔΔ+1234
 -567 → ΔΔΔ-567

(Δ : Space)

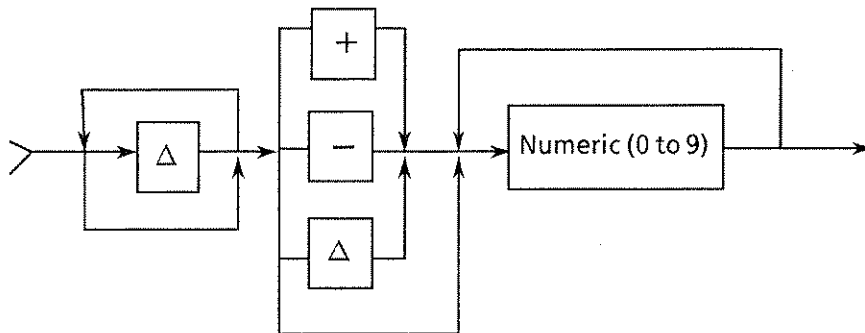


Fig. 5-3 NR1 Format

5.4.2 NR2 format (real number type)

NR2 format is real number type numeric representation.

It consists of a character sign (+, -) followed by numerics (0 to 9), decimal point (.), and numerics (0 to 9).

The + sign can be omitted or replaced and leading spaces can be inserted to align the number of digits of data, the same as NR1 format. When the decimal point is preceded, or followed by 0, the 0 can be omitted.

Examples:

- General representation +1.23, -45.6, -0.12,
 +34.0, +0.0
- + sign omitted +1.23 → 1.23
- + sign replaced +1.23 → Δ1.23
- Space insertion +1.23 → ΔΔ+1.23
 -45.6 → Δ-45.6
- High-order 0 omitted -0.12 → -.12
- Low-order 0 omitted +34.0 → +34
- +0.0 omitted +0.0 → +0. or +.0

(Δ : Space)

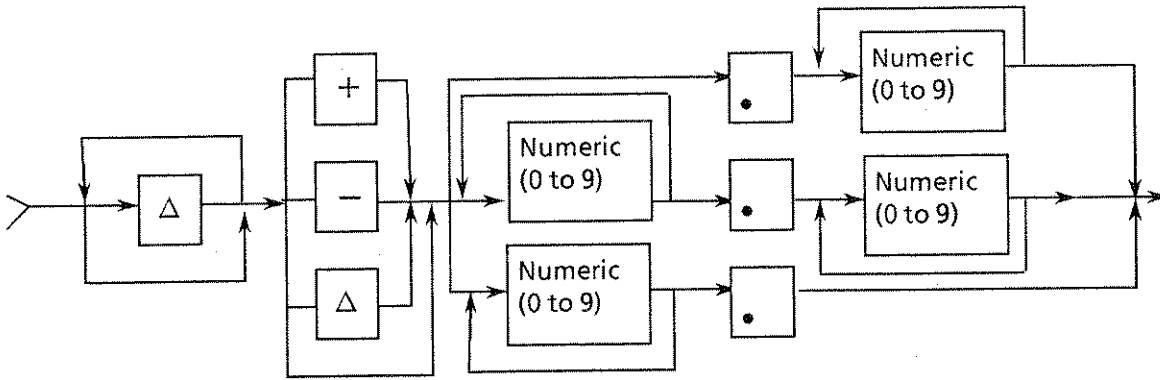


Fig. 5-4 NR2 Format

5.4.3 Hexadecimal format (HEX type)

Hexadecimal format consists of the identifier #H followed by numerics (0 to 9) or alphabetic characters (A to F). In hexadecimal format, a space can be inserted between the identifier and data. High-order 0 can be omitted.

Examples:

- General representation #H1234, #H00FF, #H0000
- Space insertion #H1234 → #HΔ1234
 #H00AF → #HΔΔΔ00AF
- 0 omission #H00FF → #HFF
 #H0000 → #H0

(Δ : Space)

5.5 Command Separator

The command separator has the following four formats:

SR1 format (semicolon ;) SR1 format (comma ,) SR2 format (line feed LF), SR3 format (END message).

5.5.1 SR1 format (semicolon ;)

SR1 format (semicolon ;) is used to separate the commands from each other when commands are sent consecutively.

Example:

ABC123;DEF45.6 is decoded as two commands "ABC123" and "DEF45.6".



Fig. 5-7 SR1 (Semicolon ;) Format

5.5.2 SR1 format (comma ,)

The SR1 format (comma ,) is used to separate data when data are sent after a header filed.

Example:

ABC123,45.6 is decoded as the two data "123" and "45.6" following the header "ABC".



Fig. 5-8 SR1 (Comma ,) Format

5.5.3 SR2 format (line feed LF)

The SR2 format is used as the symbol that indicates the end of the command. The line feed (LF) code can be used as the SR2 format code. The carriage return (CR) code can also be inserted before the line feed code.

When the MP1701B/MP1608A is designed as a listener, the receiving operation is performed until this SR2 format code or the SR3 format code described next is received. When the SR2 format or SR3 format code is received, the receiving operation ends and the make up of the received command is decoded.

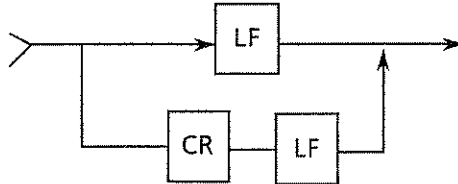


Fig. 5-9 SR2 Format

5.5.4 SR3 format (END message)

The SR3 format is used as the symbol that indicates the end of a command, the same as SR2 format.

An END message (EOI line used) can be used as the SR3 format code. (SR3 format and SR2 format can be used together.)

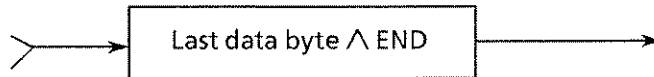


Fig. 5-10 SR3 Format

5.6 Definition of Space

The MP1701B/MP1755A/MP1608A/MP1650A treats the ASCII codes 00H to 09H and 0BH to 20H as blank space.

SECTION 6

EXECUTING PROGRAMS USED BY OTHER DEVICES

When executing programs for other devices (MP1601A/MP1604A) with the MP1701B/MP1755A/MP1608A/MP1650A, the program must be re-edited with respect to the four items described below.

1. Leave at least one space after the header (HR) field.

The header field and numeric data field (or identifier) are identified by this space.

Example:

```
OUTPUT 700;"CLK0"  
      ↓  
OUTPUT 700;"CLKΔ0"  
      ↑  
      |  
      |_____ Space inserted here.
```

2. Discriminate between use of a separator (SR) field comma (,) and a semicolon (;).

The semicolon (;) is used to separate commands and the comma (,) is used to separate data.

Examples:

- Command separation

```
OUTPUT 700;"CLKΔ0,PTNΔ0"  
      ↓  
OUTPUT 700;"CLKΔ0;PTNΔ0"  
      ↑  
      |  
      |_____ Changed to semicolon(;) here.
```

- Data separation

```
OUTPUT 700;"BITΔ10;20"  
      ↓  
OUTPUT 700;"BITΔ10,20"  
      ↑  
      |  
      |_____ Changed to comma(,) here.
```

3. When data request messages are received consecutively, only the information for the data request message received last is output.

When data request messages are sent consecutively to the MP1701B/MP1755A/MP1608A/MP1650A as shown in the example below, the MP1701B/MP1755A/MP1608A/MP1650A outputs only the data in response to the data request message received last.

Examples:

- When one line of the program contains multiple data request messages

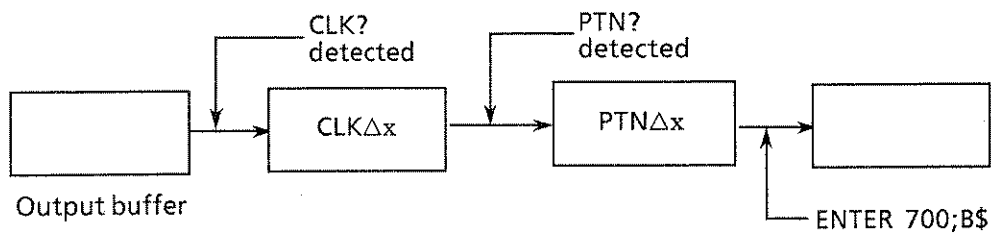
```
OUTPUT 700;"CLK?;PTN?"
```

```
ENTER 700;B$
```

```
PRINT B$ ← Outputs PTNΔx.
```

(x: Status at that time)

When the program above is executed, the output buffer operates as shown below.



Change to the following format:

```
OUTPUT 700;"CLK?"
```

```
ENTER 700;B$
```

```
PRINT B$ ← Outputs CLKΔx.
```

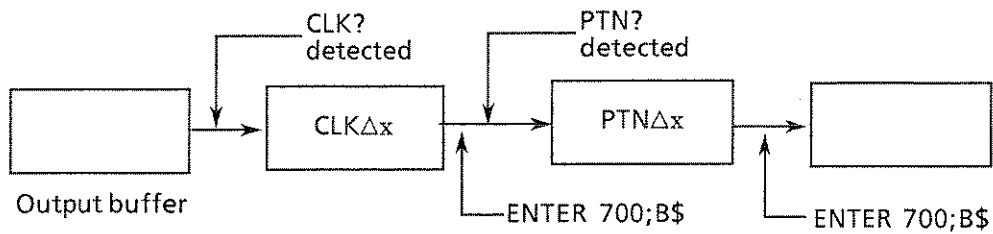
```
!
```

```
OUTPUT 700;"PTN?"
```

```
ENTER 700; B$
```

```
PRINT B$ ← Outputs PTNΔx.
```

(x: Status at that time)



- When multiple data request messages were sent

OUTPUT 700;"CLK"

OUTPUT 700;"PTN?"

ENTER 700;B\$

PRINT B\$ ← Outputs PTNΔx.

(x: Status at that time)

↓

Change to the following format:

OUTPUT 700;"CLK?"

ENTER 700;B\$

PRINT B\$ ← Outputs CLKΔx.

!

OUTPUT 700;"PTN?"

ENTER 700; B\$

PRINT B\$ ← Outputs PTNΔx.

(x: Status at that time)

4. The GP-IB status byte basic format is the same as the conventional one but the functions have been increased.

The MP1701B/MP1755A/MP1608A/MP1650A GP-IB status byte is described in SECTION 12. A sample program is shown in SECTION 14.

SECTION 7

GP-IB CONTROL COMMANDS

7.1 GP-IB Control Command List

The control command and data request commands are listed in Table 7-1. The commands that can be used differ with the current set operation mode.

There may be restrictions by the conditions already set in the command.

Table 7-1 GP-IB Commands (1/4)

Function	Control message			Data request message	No.
	Header field	Space	Numeric data field	Header field	
Operation clock *	CLK		NR1 format	CLK?	①
Internal clock frequency setting resolution *	RES		NR1 format	RES?	②
Internal clock frequency *	FRQ		NR1 format	FRQ?	③
Output pattern logic	LGC		NR1 format	LGC	④
Output pattern	PTN		NR1 format	PTN?	⑤
Output pattern mark ratio	MRK		NR1 format	MRK?	⑥
Code error addition	EAD		NR1 format	EAD?	⑦
Numbr of words	WNB		NR1 format	WNB?	⑧
Word length	WLN		NR1 format	WLN?	⑨
Data length	DLN		NR1 format	DLN?	⑩
Number of page	ADR		NR1 format	ADR?	⑪
	PAG			PAG?	
Pattern bit	BIT		NR1 format	BIT?	⑫
			Hexadecimal format		

* : Not effective for the MP1755A.

Table 7-1 GP-IB Commands (2/4)

Function	Control message			Data request message	No.
	Header field	Space	Numeric data field	Header field	
Number of pattern data input bytes	WRT		NR1 format	———	⑬
Number of pattern data output bytes	RED		NR1 format	———	⑭
Preset (all bits on all pages)	ALL		NR1 format	———	⑮
Preset (all bits on 1 page)	PST		NR1 format	———	⑯
DATA, $\overline{\text{DATA}}$, CLOCK output offset reference value	OFS		NR1 format	OFS?	⑰
Front panel/rear panel output switching	SPD		NR1 format	SPD?	⑱
DATA output amplitude	DAP		NR2 format	DAP?	⑲
DATA output offset	DOS		NR2 format	DOS?	⑳
$\overline{\text{DATA}}/\text{DATA}$ tracking	TRK		NR1 format	TRK?	㉑
$\overline{\text{DATA}}$ output amplitude	NAP		NR2 format	NAP?	㉒
$\overline{\text{DATA}}$ output offset	NOS		NR2 format	NOS?	㉓
Delay time between CLOCK outputs	CDL		NR2 format	CDL?	㉔
CLOCK output amplitude	CAP		NR2 format	CAP?	㉕

Table 7-1 GP-IB Commands (3/4)

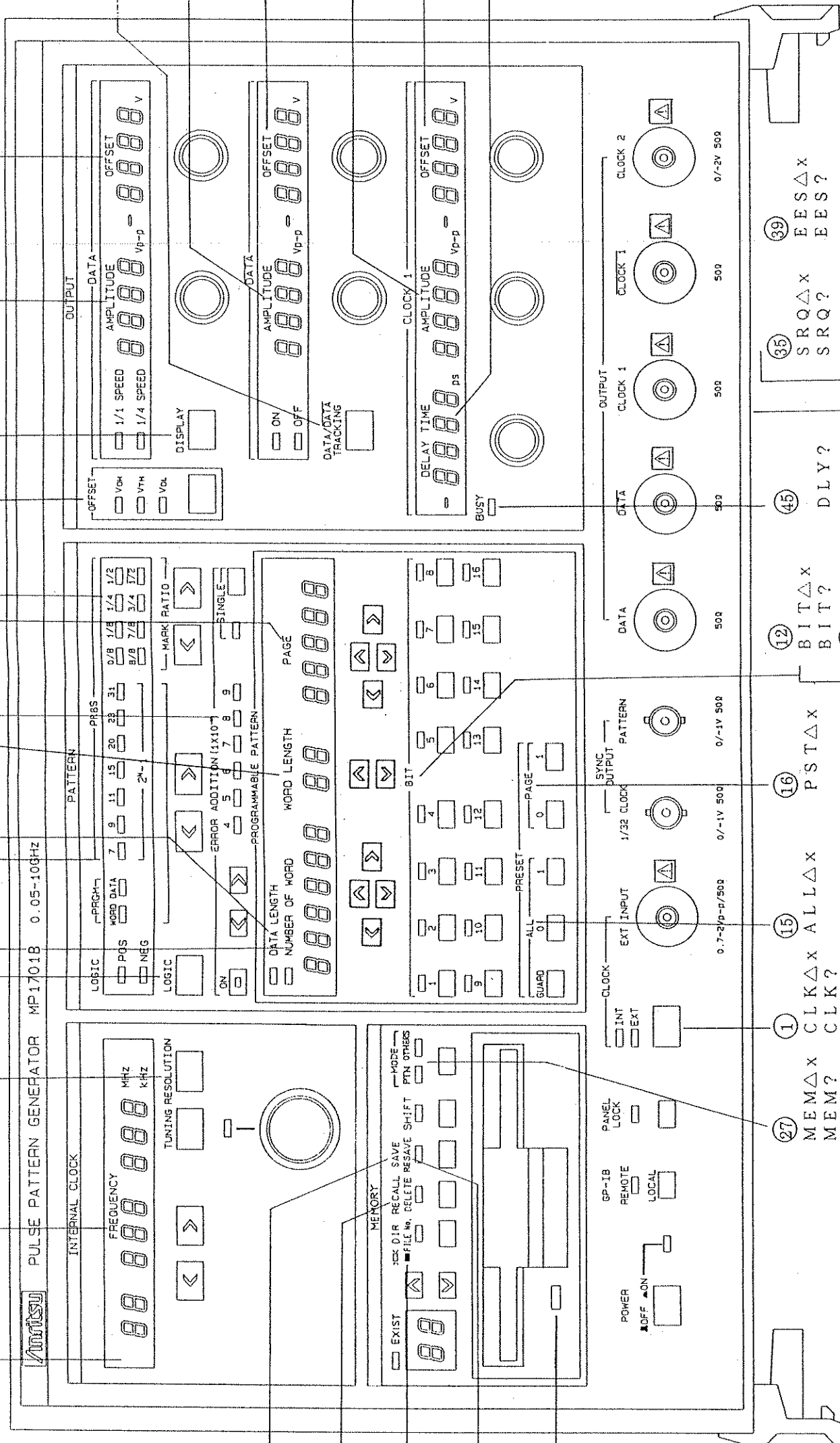
Function	Control message			Data request message	No.
	Header field	Space	Numeric data field	Header field	
CLOCK output offset	COS		NR2 format	COS?	②6
Memory function switching	MEM		NR1 format	MEM?	②7
File No./directory mode switching	FIL		NR1 format	FIL?	②8
Data recall	RCL		NR1 format	———	②9
Data save	SAV		NR1 format	———	③0
Data resave	RSV		NR1 format	———	③1
Number of external pattern input channel	SCH		Binary format	SCH?	③2
Error addition channel	ECH		NR1 format	ECH?	③3
Number of mark ratio AND bit shifts	SFT		NR1 format	SFT?	③4
Service request enable register	SRQ		NR1 format	SRQ?	③5
Status byte register	———		———	STB?	③6
Standard event status enable register	ESE		NR1 format	ESE?	③7
Standard event status register	———		———	ESR?	③8

Table 7-1 GP-IB Commands (4/4)

Function	Control message			Data request message	No.
	Header field	Space	Numeric data field	Header field	
Extended event status enable register	EES		NR1 format	EES?	③⑨
Extended event status register	_____		_____	EER?	④⑩
Initialize	INI		_____	_____	④①
Internal timer setting	RTM		NR1 format	RTM?	④②
Power failure, power recovery status	_____		_____	PWI?	④③
PLL lock status	_____		_____	PLL?	④④
Delay status	_____		_____	DLY?	④⑤
Floppy access status	_____		_____	MAC?	④⑥

LGCΔx PTNΔx WLNΔx
 LGC? PTN? WLN?
 FFSΔx
 CLK? Data request message
 Δ Space
 X Numeric data field

② FRQΔx RESΔx
 FRQ? RES?
 ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪ ⑫ ⑬ ⑭ ⑮ ⑯ ⑰ ⑱ ⑲ ⑳ ㉑ ㉒ ㉓ ㉔ ㉕ ㉖ ㉗ ㉘ ㉙ ㉚ ㉛ ㉜ ㉝ ㉞ ㉟ ㊱ ㊲ ㊳ ㊴ ㊵ ㊶ ㊷ ㊸ ㊹ ㊺ ㊻ ㊼ ㊽ ㊾ ㊿



① MEMΔx CLKΔx ALLΔx
 MEM? CLK?
 ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪ ⑫ ⑬ ⑭ ⑮ ⑯ ⑰ ⑱ ⑲ ⑳ ㉑ ㉒ ㉓ ㉔ ㉕ ㉖ ㉗ ㉘ ㉙ ㉚ ㉛ ㉜ ㉝ ㉞ ㉟ ㊱ ㊲ ㊳ ㊴ ㊵ ㊶ ㊷ ㊸ ㊹ ㊺ ㊻ ㊼ ㊽ ㊾ ㊿

① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪ ⑫ ⑬ ⑭ ⑮ ⑯ ⑰ ⑱ ⑲ ⑳ ㉑ ㉒ ㉓ ㉔ ㉕ ㉖ ㉗ ㉘ ㉙ ㉚ ㉛ ㉜ ㉝ ㉞ ㉟ ㊱ ㊲ ㊳ ㊴ ㊵ ㊶ ㊷ ㊸ ㊹ ㊺ ㊻ ㊼ ㊽ ㊾ ㊿

① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪ ⑫ ⑬ ⑭ ⑮ ⑯ ⑰ ⑱ ⑲ ⑳ ㉑ ㉒ ㉓ ㉔ ㉕ ㉖ ㉗ ㉘ ㉙ ㉚ ㉛ ㉜ ㉝ ㉞ ㉟ ㊱ ㊲ ㊳ ㊴ ㊵ ㊶ ㊷ ㊸ ㊹ ㊺ ㊻ ㊼ ㊽ ㊾ ㊿

① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪ ⑫ ⑬ ⑭ ⑮ ⑯ ⑰ ⑱ ⑲ ⑳ ㉑ ㉒ ㉓ ㉔ ㉕ ㉖ ㉗ ㉘ ㉙ ㉚ ㉛ ㉜ ㉝ ㉞ ㉟ ㊱ ㊲ ㊳ ㊴ ㊵ ㊶ ㊷ ㊸ ㊹ ㊺ ㊻ ㊼ ㊽ ㊾ ㊿

① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪ ⑫ ⑬ ⑭ ⑮ ⑯ ⑰ ⑱ ⑲ ⑳ ㉑ ㉒ ㉓ ㉔ ㉕ ㉖ ㉗ ㉘ ㉙ ㉚ ㉛ ㉜ ㉝ ㉞ ㉟ ㊱ ㊲ ㊳ ㊴ ㊵ ㊶ ㊷ ㊸ ㊹ ㊺ ㊻ ㊼ ㊽ ㊾ ㊿

Fig. 7-1 Control Message and Data Request Message (Front Panel)

7-5/(7-6 blank)

Notes
 ① : No. Control message
 CLKΔx

⑪ ADDRΔx
 ADDR?

④ ⑤ ⑥ ⑦

⑧ ⑨ ⑩ ⑪

⑫ ⑬ ⑭ ⑮

⑯ ⑰ ⑱ ⑲

⑳ ㉑ ㉒ ㉓

㉔ ㉕ ㉖ ㉗

㉘ ㉙ ㉚ ㉛

㉜ ㉝ ㉞ ㉟

㊱ ㊲ ㊳ ㊴

㊵ ㊶ ㊷ ㊸

㊹ ㊺ ㊻ ㊼

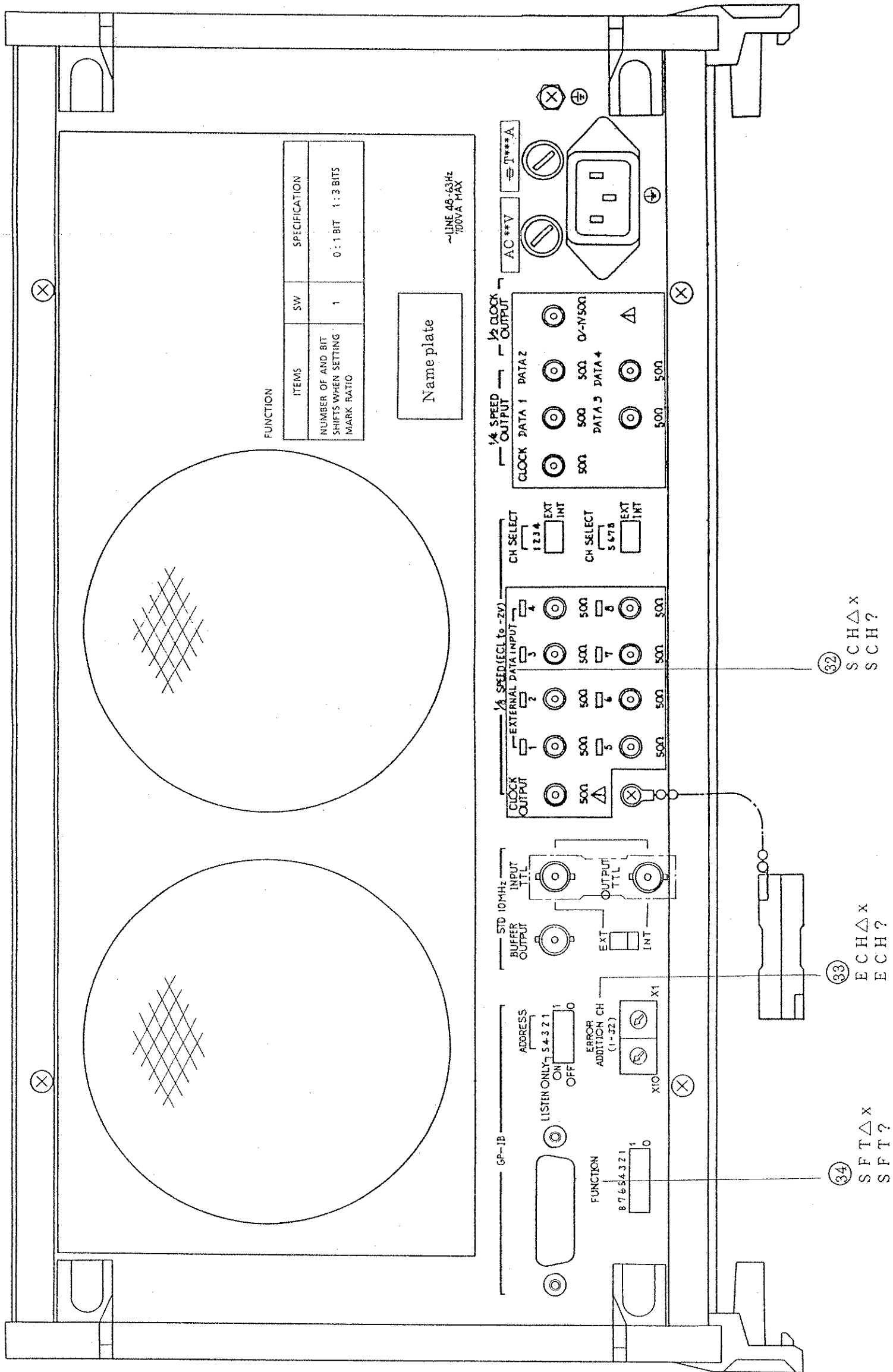


Fig. 7-2 Control Message and Data Request Message (Rear Panel)

7.2 GP-IB Control Command List in Alphabetic Order

The MP1701B/MP1755A/MP1608A/MP1650A GP-IB control commands given in paragraph 7.1 are listed in alphabetic order in Table 7-2.

Table 7-2 GP-IB Commands in Alphabetic Order(1/4)

Function	Control message			Data request message	No.
	Header field	Space	Numeric data field	Header field	
Number of page	ADR		NR1 format	ADR?	⑪
Preset (all bits on all pages)	ALL		NR1 format	———	⑮
Pattern bit	BIT		NR1 format	BIT?	⑫
			Hexadecimal format		
CLOCK output amplitude	CAP		NR2 format	CAP?	⑳
CLOCK output delay time	CDL		NR2 format	CDL?	㉔
Operation clock *	CLK		NR1 format	CLK?	①
CLOCK output offset	COS		NR2 format	COS?	㉖
DATA output amplitude	DAP		NR2 format	DAP?	⑲
Data length	DLN		NR1 format	DLN?	⑩
Delay status	———		———	DLY?	④⑤
DATA output format	DOS		NR2 format	DOS?	㉒
Code error addition	EAD		NR1 format	EAD?	⑦
Error addition channel	ECH		NR1 format	ECH?	③③

* : Not effective for the MP1755A.

Table 7-2 GP-IB Commands in Alphabetic Order (2/4)

Function	Control message			Data request message	No.
	Header field	Space	Numeric data field	Header field	
Extended event status register	_____		_____	EER?	④⑩
Extended event status enable register	EES		NR1 format	EES?	③⑨
Standard event status enable register	ESE		NR1 format	ESE?	③⑦
Standard event status register	_____		_____	ESR?	③⑧
File No./directory mode switching	FIL		NR1 format	FIL?	②⑧
Internal clock frequency *	FRQ		NR1 format	FRQ?	③
Initialize	INI		_____	_____	④①
Output pattern logic	LGC		NR1 format	LGC?	④
Floppy disk access status	_____		_____	MAC?	④⑥
Memory function switching	MEM		NR1 format	MEM?	②⑦
Output pattern mark ratio	MRK		NR1 format	MRK?	⑥
DATA output amplitude	NAP		NR2 format	NAP?	②②
DATA output offset	NOS		NR2 format	NOS?	②③

* : Not effective for the MP1755A.

Table 7-2 GP-IB Commands in Alphabetic Order (3/4)

Function	Control message			Data request message	No.
	Header field	Space	Numeric data field	Header field	
DATA, $\overline{\text{DATA}}$, CLOCK output offset reference value	OFS		NR1 format	OFS?	⑰
Number of page	PAG		NR1 format	PAG?	⑪
PLL lock status	———		———	PLL?	④④
Preset (all bits on 1 page)	PST		NR1 format	———	⑰⑥
Output pattern	PTN		NR1 format	PTN?	⑤
Power failure, power recovery status	———		———	PWI?	④③
Data recall	RCL		NR1 format	———	②⑨
Number of pattern data output bytes	RED		NR1 format	———	⑭④
Internal clock frequency setting resolution *	RES		NR1 format	RES?	②
Data resave	RSV		NR1 format	———	③①
Internal time setting	RTM		NR1 format	RTM?	④②
Data save	SAV		NR1 format	———	③⑦
Number of external pattern input channel	SCH		Binary format	SCH	③②
Number of mark ratio AND bit shifts	SFT		NR1 format	SFT?	③④

* : Not effective for the MP1755A.

Table 7-2 GP-IB Commands in Alphabetic Order (4/4)

Function	Control message			Data request message	No.
	Header field	Space	Numeric data field	Header field	
Front panel/rear panel output switching	SPD		NR1 format	SPD?	⑱
Service request enable register	SRQ		NR1 format	SRQ?	⑳
Status byte register	—		—	STB?	㉑
DATA/ $\overline{\text{DATA}}$ tracking	TRK		NR1 format	TRK?	㉒
Word length	WLN		NR1 format	WLN?	⑨
Number of words	WNB		NR1 format	WNB?	⑧
Number of pattern data input bytes	WRT		NR1 format	—	⑬

7.3 Correspondence with Common Commands

The relationship between the common commands defined by IEEE 488.2 and the MP1701B/MP1755A/MP1608A/MP1650A commands is shown in Table 7-3.

The common commands of the table below are essential commands defined by IEEE 488.2. For a detailed description of the common commands, refer to IEEE 488.2.

Table 7-3 Correspondence with Common Commands

Common Command	Corresponding Command	No.	Item on IEEE 488.2 standard
*IDN?	_____	_____	10.14
*RST	_____	_____	10.32
*TST?	_____	_____	10.38
*OPC	_____	_____	10.18
*OPC?	_____	_____	10.19
*WAI	_____	_____	10.39
*CLS	_____	_____	10.3
*ESE	ESE	③⑦	10.10
*ESE?	ESE?	③⑦	10.11
*ESR?	ESR?	③⑧	10.12
*PSC	_____	_____	10.25
*PSC?	_____	_____	10.26
*SRE	SRQ	③⑤	10.34
*SRE?	SRQ?	③⑤	10.35
*STB?	STB?	③⑥	10.36

SECTION 8

DEVICE CLEAR FUNCTION

When the MP1701B/MP1755A/MP1608A/MP1650A receives the device clear command, it returns to its state immediately after the power is turned on. Lamp check is not performed.

The rear panel functions return to the state set by the dip switches.

(When the MP1701B/MP1755A/MP1608A/MP1650A is returned from the remote state to the local state, the rear panel functions return to the state set by the dip switches.)

Moreover, if the floppy disk is being accessed, access is interrupted.

SECTION 9 DEVICE TRIGGER FUNCTION

The MP1701B/MP1755A/MP1608A/MP1650A does not have a device trigger function.

SECTION 10 MESSAGE LENGTH

The effective message length of the data acceptable to the MP1701B/MP1755A/MP1608A/MP1650A is up to 256 bytes (256 characters).

It can be assumed that data exceeding this number is not input.

That is, limit the number of characters of a GP-IB command sent by one line to 256 characters or less, including the last CR/LF.

CLKΔ0 ; PTNΔ0 ; ; MEMΔ0 (CR/LF)

256 characters or less

SECTION 11

CONTROL MESSAGE AND DATA REQUEST MESSAGE

11.1 INTERNAL CLOCK Section

The INTERNAL CLOCK section GP-IB commands are described in the following pages.

The Δ in the test means a space.

Note: The MP1755A does not have an internal clock.
So, the INTERNAL CLOCK section GP-IB commands are not effective for the MP1755A.

① Operation clock (Clock mode)

Set value	Control message	Data request message	Output message
External clock (EXT)	CLKΔ0	CLK?	CLKΔ0
Internal clock (INT)	CLKΔ1		CLKΔ1

Restrictions

Control message Invalid in the following case:
 When floppy disk is being accessed

Data request message No restrictions

Examples

Control message OUTPUT 700; "CLKΔ0"
 Selects external clock (EXT) as the operation clock.

Data request message When operation clock is external (EXT)
 OUTPUT 700; "CLK?"
 ENTER 700; B\$
 PRINT B\$
 ↓
 Outputs CLKΔ0 (CR/LF).

Note: Those commands are not effective for the MP1755A.

② Internal clock frequency setting resolution (Resolution mode)

Set value	Control message	Data request message	Output message
kHz units	RESΔ0	RES?	RESΔ0
MHz units	RESΔ1		RESΔ1

Restrictions

Control message Invalid in the following cases:
 When operation clock is external clock (EXT)
 When floppy disk is being accessed

Data request message Invalid in the following case and ERR is output:
 When operation clock is external clock (EXT)

Examples

Control message When operation clock is internal clock (INT)
 OUTPUT 700; "RESΔ0"
 Sets the internal clock frequency setting resolution to kHz.

Data request message When the internal clock frequency setting resolution is kHz
 OUTPUT 700; "RES?"
 ENTER 700;B\$
 PRINT B\$
 ↓
 Outputs RESΔ0 (CR/LF).

 When operation clock is external clock (EXT)
 OUTPUT 700; "RES?"
 ENTER 700;B\$
 PRINT B\$
 ↓
 Outputs ERR (CR/LF).

- Notes:** 1. The number of settable digits and display digits of the internal clock frequency differs depending on whether the setting is in kHz or MHz units.
 2. Those commands are not effective for the MP1755A.

③ Internal clock frequency (Frequency)

MP1701B: When the internal clock frequency setting resolution is in kHz units.			
Set value	Control message	Data request message	Output message
5 0 0 0 0 kHz	FRQΔ5 0 0 0 0	FRQ?	FRQΔΔΔΔ 5 0 0 0 0
⋮	⋮		⋮
1 0 0 0 0 0 0 0 kHz	FRQΔ1 0 0 0 0 0 0 0		FRQΔ1 0 0 0 0 0 0 0

MP1608A: When the internal clock frequency setting resolution is in kHz units.			
Set value	Control message	Data request message	Output message
5 0 0 0 0 kHz	FRQΔ5 0 0 0 0	FRQ?	FRQΔΔΔΔ 5 0 0 0 0
⋮	⋮		⋮
5 0 0 0 0 0 0 kHz	FRQΔ5 0 0 0 0 0 0		FRQΔΔ5 0 0 0 0 0 0

MP1650A: When the internal clock frequency setting resolution is in kHz units.			
Set value	Control message	Data request message	Output message
3 0 0 0 0 kHz	FRQΔ3 0 0 0 0	FRQ?	FRQΔΔΔΔ 3 0 0 0 0
⋮	⋮		⋮
3 0 0 0 0 0 0 kHz	FRQΔ3 0 0 0 0 0 0		FRQΔΔ3 0 0 0 0 0 0

MP1701B: When the internal clock frequency setting resolution is in MHz units.			
Set value	Control message	Data request message	Output message
5 0 MHz	FRQΔ5 0	FRQ?	FRQΔ Δ Δ Δ 5 0
⋮	⋮		⋮
1 0 0 0 0 MHz	FRQΔ1 0 0 0		FRQΔ 1 0 0 0 0

MP1608A: When the internal clock frequency setting resolution is in MHz units.

Set value	Control message	Data request message	Output message
50 MHz	FRQΔ50	FRQ?	FRQΔΔΔΔ50
⋮	⋮		⋮
5000 MHz	FRQΔ5000		FRQΔΔ5000

MP1650A: When the internal clock frequency setting resolution is in MHz units.

Set value	Control message	Data request message	Output message
30 MHz	FRQΔ30	FRQ?	FRQΔΔΔΔ30
⋮	⋮		⋮
3000 MHz	FRQΔ3000		FRQΔΔ3000

Restrictions

Control message	Invalid in the following cases: When operation clock is external clock (EXT) When floppy disk is being accessed
Data request message	Invalid in the following case and ERR is output: When operation clock is external clock (EXT)

Examples

Control message When operation clock is internal clock (INT) and internal clock frequency setting resolution is set in kHz units.

OUTPUT 700; "FRQΔ50000"

 Sets the internal clock frequency to 50000 kHz.

Data request message When internal clock frequency is 50000 kHz

OUTPUT 700; "FRQ?"

ENTER 700;B\$

PRINT B\$

↓

 Outputs FRQΔΔΔΔ50000 (CR/LF).

When operation clock is external clock (EXT)

OUTPUT 700; "FRQ?"

ENTER 700;B\$

PRINT B\$

↓

 Outputs ERR (CR/LF).

Notes: 1. The number of settable digits and display digits of the internal clock frequency differs depending on whether the setting is kHz or MHz units.

2. Those commands are not effective for the MP1755A.

11.2 PATTERN Section

The PATTERN section GP-IB commands are described on the following pages.

The Δ in the text means a space.

④ Output pattern logic (Logic mode)

Set value	Control message	Data request message	Output message
Positive logic (POSITIVE)	LGCΔ0	LGC?	LGCΔ0
Negative logic (NEGATIVE)	LGCΔ1		LGCΔ1

Restrictions

Control message Invalid in the following cases:
 When floppy disk is being accessed

Data request message No restrictions

Examples

Control message OUTPUT 700; "LGCΔ0"
 Sets the Output pattern logic to positive logic (POSITIVE).

Data request message When output pattern logic is positive logic (POSITIVE)
 OUTPUT 700; "LGC?"
 ENTER 700;B\$
 PRINT B\$
 ↓
 Outputs LGCΔ0 (CR/LF).

Note: When output pattern is in PRBS mode, if the output pattern logic is set, the output pattern mark ratio is switched automatically.

⑤ Output pattern (Pattern mode)

Set value	Control message	Data request message	Output message
PRGM. WORD	PTNΔ0	PTN?	PTNΔ0
PRGM. DATA	PTNΔ1		PTNΔ1
PRBS 27-1	PTNΔ2		PTNΔ2
PRBS 29-1	PTNΔ3		PTNΔ3
PRBS 211-1	PTNΔ5		PTNΔ5
PRBS 215-1	PTNΔ6		PTNΔ6
PRBS 220-1	PTNΔ7		PTNΔ7
PRBS 223-1	PTNΔ8		PTNΔ8
PRBS 231-1	PTNΔ9		PTNΔ9

Restrictions

Control message Invalid in the following cases:
 When PTNΔ4 is set
 When floppy disk is being accessed

Data request message No restrictions

Examples

Control message OUTPUT 700; "PTNΔ0"
 Sets the word pattern to PRGM.WORD pattern.

Data request message When output pattern is PRGM.WORD
 OUTPUT 700; "PTN?"
 ENTER 700;B\$
 PRINT B\$
 ↓
 Outputs PTNΔ0 (CR/LF).

Note: PRGM: Programmable pattern, PRBS: Pseudorandom pattern

When output pattern is set in the PRBS mode, the previously set output pattern mark ratio display and its settings are reset.

⑦ Code error addition (Error addition mode)

Set value	Control message	Data request message	Output message
OFF	EADΔ0	EAD?	EADΔ0
1×10^{-4}	EADΔ1		EADΔ1
1×10^{-5}	EADΔ2		EADΔ2
1×10^{-6}	EADΔ3		EADΔ3
1×10^{-7}	EADΔ4		EADΔ4
1×10^{-8}	EADΔ5		EADΔ5
1×10^{-9}	EADΔ6		EADΔ6
SINGLE	EADΔ7		EADΔ7

Restrictions

Control message Invalid in the following case:
 When floppy disk is being accessed

Data request message No restrictions

Examples

Control message OUTPUT 700; "EADΔ0"
 Turns OFF code error addition.

Data request message When code error addition is OFF
 OUTPUT 700; "EAD?"
 ENTER 700;B\$
 PRINT B\$

↓

Outputs EADΔ0 (CR/LF).

Note: SINGLE addition adds 1 pulse error.

⑧ Number of words (Number of words)

Set value	Control message	Data request message	Output message
1	WNBΔ 1		WNBΔΔΔΔΔ1
⋮	⋮	WNB?	⋮
3 2 7 6 8	WNBΔ 32768		WNBΔ32768

However, there are some restrictions on the step value.

Restrictions

- Control message Invalid in the following cases:
 When output pattern is other than WORD pattern
 When floppy disk is being accessed
- Data request message Invalid in the following case and ERR is output:
 When output pattern is other than WORD pattern

Examples

- Control message When output pattern is WORD pattern
 OUTPUT 700; "WNBΔ1"
 Sets the number of words to 1.
- Data request message When number of words is 1
 OUTPUT 700; "WNB?"
 ENTER 700;B\$
 PRINT B\$
 ↓
 Outputs WNBΔΔΔΔΔ1 (CR/LF).
 When output pattern is other than WORD pattern
 OUTPUT 700; "WNB?"
 ENTER 700;B\$
 PRINT B\$
 ↓
 Outputs ERR (CR/LF).

Note: The number of words step value that can be set differs depending on the word length set value.

(See Table 11-1.)

When the input number of words can not be set, the input value is compared to the currently displayed value and the input value is changed to an optimum value.

Note (Cont.):

Example: For word length 2

When number of words 2100 is input, it is changed to 2048 and displayed.

Next, when number of words 2100 is input again, it is changed to 2112 and displayed.

Table 11-1 Numeric Relationship between Word Length and Number of Words

Word length	Number of words			
	Step width	Range	Step width	Range
2	1 step	1 to 2048	64 step	2112 to 32768
3	1 step	1 to 1365	128 step	1408 to 32768
4	1 step	1 to 1024	32 step	1056 to 32768
5	1 step	1 to 819	128 step	896 to 32768
6	1 step	1 to 682	64 step	704 to 32768
7	1 step	1 to 585	128 step	640 to 32768
8	1 step	1 to 512	16 step	528 to 32768
9	1 step	1 to 455	128 step	512 to 32768
10	1 step	1 to 409	64 step	448 to 32768
11	1 step	1 to 372	128 step	384 to 32768
12	1 step	1 to 341	32 step	352 to 32768
13	1 step	1 to 315	128 step	384 to 32768
14	1 step	1 to 292	64 step	320 to 32768
15	1 step	1 to 273	128 step	384 to 32768
16	1 step	1 to 256	8 step	264 to 32768

⑨ Word length (Word length)

Set value	Control message	Data request message	Output message
2	WLNΔ 2		WLN ΔΔ 2
⋮	⋮	WLN?	⋮
16	WLNΔ16		WLN Δ16

The set state is every step in the table above.

Restrictions

- Control message Invalid in the following cases:
 When output pattern is other than WORD pattern
 When floppy disk is being accessed
- Data request message Invalid in the following case and ERR is output:
 When output pattern is other than WORD pattern

Examples

- Control message When output pattern is WORD pattern
 OUTPUT 700; "WLNΔ2"
 Sets the word length to 2.
- Data request message When word length is 2
 OUTPUT 700; "WLN?"
 ENTER 700;B\$
 PRINT B\$
 ↓
 Outputs WLNΔΔ2 (CR/LF).
 When output pattern is other than WORD pattern
 OUTPUT 700; "WLN?"
 ENTER 700;B\$
 PRINT B\$
 ↓
 Outputs ERR (CR/LF).

Note: Depending on the word length setting, if the number of words at that time is not suitable, it is changed automatically to the corresponding optimum value. (See Note in ⑧ command.)

⑩ Data length (Data length)

Set value	Control message	Data request message	Output message
2	DLN△ 2		DLN△△△△△△2
⋮	⋮	DLN?	⋮
5 2 4 2 8 8	DLN△524288		DLN△524288

However, there are some restrictions on the step value in the table above.

Restrictions

- | | |
|----------------------|---|
| Control message | Invalid in the following cases:
When output pattern is other than DATA pattern
When floppy disk is being accessed |
| Data request message | Invalid in the following case and ERR is output:
When output pattern is other than DATA pattern |

Examples

- | | |
|----------------------|--|
| Control message | When output pattern is DATA pattern
OUTPUT 700; "DLN△2"
Sets the data length to 2. |
| Data request message | When data length is 2
OUTPUT 700; "DLN?"
ENTER 700;B\$
PRINT B\$
↓
Outputs DLN△△△△△△2 (CR/LF).
When output pattern is other than DATA pattern
OUTPUT 700; "DLN?"
ENTER 700;B\$
PRINT B\$
↓
Outputs ERR (CR/LF). |

Note: There are two data-length step values: 1 step and 128 steps. (See Table 11-2.)

When the input data length can not be set, the input value is compared to the currently displayed value and the input value is changed to a optimum value.

Example

When data length 4100 is input, it is changed to 4096 and displayed

When data length 4100 is input again, it is changed to 4224 and displayed.

Table 11-2 Numerical Relationship of Data Length

Data length	Step width	
2 to 4096	1 step	$524288/128 = 4096$
4224 to 524288	128 steps	$128N (N = 33 \text{ to } 4096)$

⑪ Number of page (Page)

Set value	Control message	Data request message	Output message
1 ⋮ 3 2 7 6 8	PAGΔ 1 ⋮ PAGΔ32768	PAG?	PAG ΔΔΔΔΔ 1 ⋮ PAGΔ3 2 7 6 8
1 ⋮ 3 2 7 6 8	ADRΔ 1 ⋮ ADRΔ32768	ADR?	ADR ΔΔΔΔΔ 1 ⋮ ADRΔ3 2 7 6 8

Restrictions

Control message

Invalid in the following cases:
When output pattern is PRBS mode
When floppy disk is being accessed

Data request message

Invalid in the following case and ERR is output:
When output pattern is PRBS mode

Examples

Control message

When output pattern is PRGM mode
OUTPUT 700; "PAGΔ1"
Sets the number of page to 1.
OUTPUT 700; "ADRΔ1"
Sets the number of page to 1.

Data request message

When number of page is 1
OUTPUT 700; "PAG?"
ENTER 700;B\$
PRINT B\$
↓
Outputs PAGΔΔΔΔΔ1 (CR/LF).
OUTPUT 700; "ADR?"
ENTER 700;B\$
PRINT B\$
↓
Outputs ADRΔΔΔΔΔ1 (CR/LF).

When output pattern is PRBS mode
 OUTPUT 700; "PAG?"
 ENTER 700;B\$
 PRINT B\$



Outputs ERR (CR/LF).

Note: There are two kinds of page number set commands: PAG and ADR.

They have the same function. The maximum number of page that can be set depends on the number of words, word length, and data length.

When a number of page exceeds a maximum number of page, page number are held to that maximum number set.

Example

When data length = 32 and displayed number of page = 1, the maximum number of page is 2.

If PAGΔ3 is input, the displayed number of page is changed to page 2.

Table 11-3 Relationship between Numer of Page and WORD/DATA Pattern

Output pattern	Page variation range										
WORD	1 to set number of words, 1 step width										
DATA	<p>1 to value not exceeding the value of $(\text{data length} \div 16)$, 1 step width</p> <p>(When remainder is 0, up to value of quotient, and when there is a remainder, up to value of quotient + 1, 1 step width)</p> <table style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th><u>Data length</u></th> <th><u>Number of page</u></th> </tr> </thead> <tbody> <tr> <td>2 to 16</td> <td>1</td> </tr> <tr> <td>17 to 32</td> <td>2</td> </tr> <tr> <td>33 to 48</td> <td>3</td> </tr> <tr> <td>49 to</td> <td></td> </tr> </tbody> </table>	<u>Data length</u>	<u>Number of page</u>	2 to 16	1	17 to 32	2	33 to 48	3	49 to	
<u>Data length</u>	<u>Number of page</u>										
2 to 16	1										
17 to 32	2										
33 to 48	3										
49 to											

⑫ **Pattern bit (Pattern bit)**

Set value	Control message		Data request message
0	BITΔ0	BITΔ0#H0000	BIT?
:	:	:	
65535	BITΔ65535	BITΔ#HFFFF	
Output message image			
<p>The current number of page set and the contents of the bits up to 8 pages including the maximum pattern set page from that page are output in the following format:</p> <p>PAG Δ****;BITΔ#H****,#H****,#H****,#H****,#H****,#H****,#H****,#H****</p>			

Restrictions

- Control message Invalid in the following cases:
 - When output pattern is PRBS mode
 - When floppy disk is being accessed
- Data request message Invalid in the following case and ERR is output:
 - When output pattern is PRBS mode

Examples

- Control message When output pattern is in PRGM mode and bit pattern of 3 pages from the currently set page is set.
 - OUTPUT 700; "BITΔ10,20,30"
 - OUTPUT 700; "BITΔ#HFFFF,#H1000,#H2000"
 - Bit pattern of consecutive pages can be set by separating the data from each other with a comma (,).
 - When setting number of page and setting pattern bit of 4 pages from the current page.
 - OUTPUT 700; "PAGΔ10;BITΔ10,20,30,40"
 - OUTPUT 700; "PAGΔ10;BITΔ#HFFFF,#H1000,#H2000,#H3000"
 - OUTPUT 700; "ADRΔ10;BITΔ10,20,30,40"
 - OUTPUT 700; "ADRΔ10;BITΔ#HFFFF,#H1000,#H2000,#H3000"

Data request message When number of displayed page is 1 and maximum number of page is 29,
data will appear in this format.

```
OUTPUT 700; "BIT?"
```

```
FOR I = 1 to 4
```

```
  ENTER 700;B$
```

```
  PRINT B$
```

```
NEXT I "
```

↓

```
PAGΔΔΔΔΔ1;BITΔ#H0000,#H0000,#H0000,#H0000,#H0000,#H0000,#H0000,#H0000
```

```
PAGΔΔΔΔΔ9;BITΔ#H0000,#H0000,#H0000,#H0000,#H0000,#H0000,#H0000,#H0000
```

```
PAGΔΔΔΔΔ17;BITΔ#H0000,#H0000,#H0000,#H0000,#H0000,#H0000,#H0000,#H0000
```

```
PAGΔΔΔΔΔ25;BITΔ#H0000,#H0000,#H0000,#H0000,#H0000
```

When output pattern is in PRBS mode

```
OUTPUT 700; "BIT?"
```

```
ENTER 700;B$
```

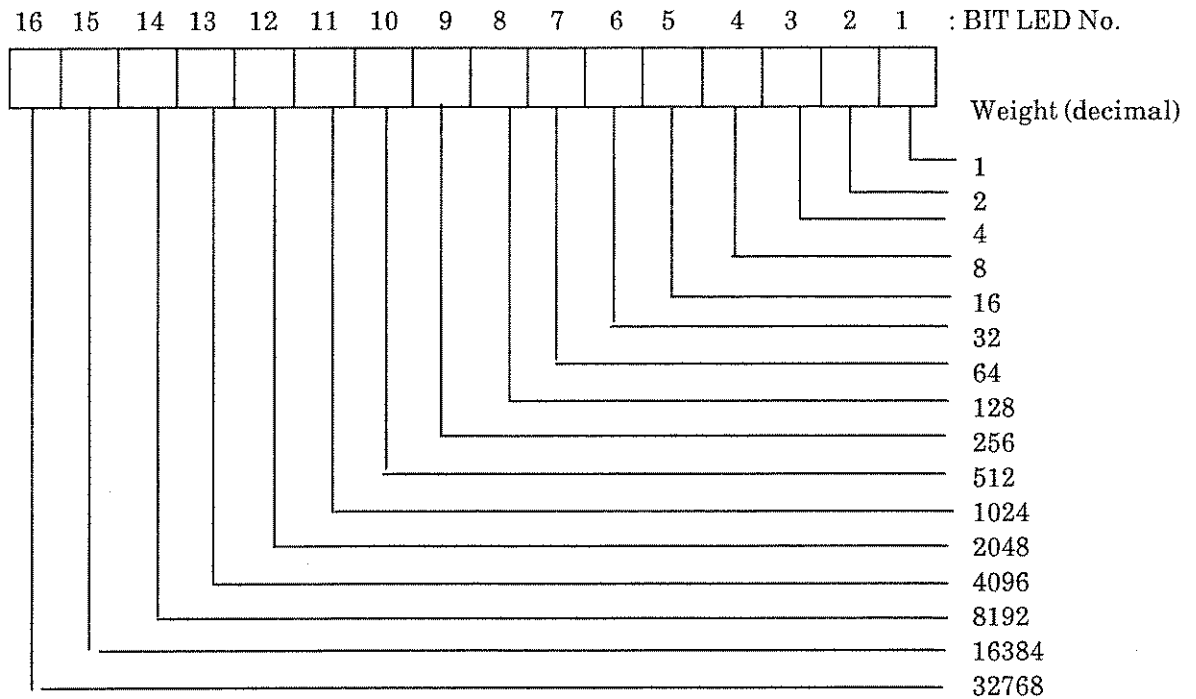
```
PRINT B$
```

↓

Outputs ERR (CR/LF).

Set value and bit weight

The relationship between actual pattern bit setting and data values set are shown below.



- BitΔ1 : Only BIT LED
No. 1 is set.
- BitΔ16 : Only BIT LED
No. 5 is set.
- BitΔ10 : Only BIT LED's
Nos. 2 and 4 are set.
- BitΔ#H000F : Only BIT LED's
Nos. 1 to 4 are set. (Decimal: 15)
- BitΔ#H00F0 : Only BIT LED's
Nos. 5 to 8 are set. (Decimal: 240)
- BitΔ#H1000 : Only BIT LED
No. 13 is set. (Decimal: 4096)
- BitΔ#HA000 : Only BIT LED's
Nos. 14 and 16 are set. (Decimal 40960)

Note: The pattern bit is set at the page which is already set from the value of the NR field.

Pattern bit of consecutive pages can be set by separating the data of the NR field from each other by a comma (.). However, when the output pattern is a WORD pattern, the settable pattern bit are restricted from 1 to a number of set word-length bits; when the output pattern is a DATA pattern, they are restricted to up to number of bits of quotient [data length/16; when remainder is 0, up to all bits.]

⑬ Number of pattern data input bytes (Pattern data write)

Set value		Control message	Data request message
Number of pattern transfer bytes	Pattern input start address		
1 ⋮ 65536	0 ⋮ 32767	WRTΔ1, 0 ⋮ WRTΔ65536,32767	None
Perform in "WRTΔ number of pattern transfer bytes, pattern input start address" format			

Restrictions

Control message

Invalid in the following cases:

When (the number of pattern transfer bytes + pattern input start address × 2) > 65536

When output pattern is in PRBS mode

When floppy disk is being accessed

Examples

Control message

When output pattern is in PRGM mode

DIM B (9)

READ B (*)

DATA 1,2,4,8,16,32,64,128,256,512

OUTPUT 700; "WRTΔ20,0"

OUTPUT 700 USING "W";B(*)

↓

Sets the pattern data corresponding to the DATA statement for the number of page 1 to 10.

Note: The MP1701B/MP1755A/MP1608A/MP1650A defines the necessary number of bytes of pattern data to be DMA-transferred and the input start address from each value of the NR field, and defines DMA mode switching and the internal RAM area storage address.

The relationship between pattern input start address and actual number of page is:

$$(\text{pattern input start address} + 1) = \text{actual number of page}$$

The DMA transfer mode is released at the end of pattern data transfer.

However, the following is performed as exception processing:

- When the pattern data is not all sent after the number of pattern input bytes is set, the DMA mode is released and bit 0 (command error bit) of the GP-IB status byte is set to 1 by time-out function.

The time-out is 60 seconds.

- When a separate control command is transferred after a number of pattern input bytes is set, that command is assumed to be the pattern data set value.

Pattern data DMA transfer is also described in SECTION 13.

⑭ Number of pattern data output bytes (Pattern data read)

Set value		Control message	Data request message
Number of pattern transfer bytes	Pattern input start address		
1 ⋮ ⋮ 65536	0 ⋮ ⋮ 32767	REDΔ1, 0 ⋮ REDΔ65536,32767	None
Perform in "REDΔ number of pattern transfer bytes, pattern input start address" format			

Restrictions

Control message

Invalid in the following cases:

When (number of pattern transfer bytes + pattern output start address × 2) > 65536

When output pattern is in PRBS mode

When floppy disk is being accessed

Examples

Control message

When output pattern is in PRGM mode

DIM B (9)

OUTPUT 700; "REDΔ20,0"

ENTER 700 USING "W";B(*)

PRINT B (*)

↓

Outputs the pattern data corresponding to number of pages 1 to 10.

Note: The MP1701B/MP1755A/MP1608A/MP1650A defines the necessary number of bytes of pattern data to be DMA-transferred and the output start address from each value of the NR field, and defines DMA mode switching and the internal RAM area storage addresses.

The relationship between pattern input start address and actual number of page is:

$$(\text{pattern output start address} + 1) = \text{actual number of page}$$

The DMA transfer mode is released at the end of pattern data transfer.

However, the following is performed as exception processing:

- When the pattern data is not received at all after number of pattern output bytes is set, the DMA mode is released and bit 0 (command error bit) of the GP-IB status byte is set to 1 by time-out function.

The time-out is 60 seconds.

Pattern data DMA transfer is also described in SECTION 13.

⑮ **Preset (all bits on all pages) (Preset all 0 or 1)**

Set value	Control message	Data request message	Output message
All bits on all pages, clear	ALLΔ0	None	_____
All bits on all pages, set	ALLΔ1		

Restrictions

Control message Invalid in the following cases:
 When output pattern is in PRBS mode
 When floppy disk is being accessed

Examples

Control message When output pattern is in PRGM mode
 OUTPUT 700; "ALLΔ0"
 Sets all bits on all pages to 0.

Note: All bits on all pages have 512k bits of pattern data.

⑩ Preset (all bits on 1 page) (Preset page 0 or 1)

Set value	Control message	Data request message	Output message
All bits on 1 page, clear	PSTΔ0	None	_____
All bits on 1 page, set	PARΔ1		

Restrictions

Control message

Invalid in the following cases:
 When output pattern is in PRBS mode
 When floppy disk is being accessed

Examples

Control message

When output pattern is PRGM mode
 OUTPUT 700; "PSTΔ0"
 Sets all bits of the displayed page to 0.

Note: All bits on 1 page indicate all the bits of the displayed page (1 page).

11.3 OUTPUT Section

The OUTPUT section GP-IB commands are described on the following pages.

The Δ in the text means a space.

⑰ DATA, $\overline{\text{DATA}}$, CLOCK output offset reference value (Offset mode)

Set value	Control message	Data request message	Output message
Offset reference value VOH	OFS Δ 0	OFS?	OFS Δ 0
Offset reference value VTH	OFS Δ 1		OFS Δ 1
Offset reference value VOL	OFS Δ 2		OFS Δ 2

Restrictions

Control message Invalid in the following case:
 When floppy disk is being accessed

Data request message No restrictions

Examples

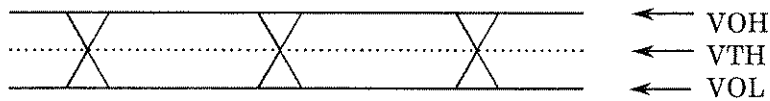
Control message OUTPUT 700; "OFS Δ 0"
 Sets the DATA, $\overline{\text{DATA}}$, CLOCK output offset reference value to VOH.

Data request message When DATA, $\overline{\text{DATA}}$, CLOCK output offset reference value is VOH
 OUTPUT 700, "OFS?"
 ENTER 700; B\$
 PRINT B\$



Output OFS Δ 0 (CR/LF).

Note: The output offset reference values are shown below.



When the offset reference value is set, the DATA, $\overline{\text{DATA}}$, and CLOCK output offset voltage display changes to the optimum value at that time.

⑱ Front panel/rear panel output switching (Speed)

Set value	Control message	Data request message	Output message
1/1 SPEED	SPDΔ0	SPD?	SPDΔ0
1/4 SPEED	SPDΔ1		SPDΔ1

Restrictions

Control message Invalid in the following case:
 When floppy disk is being accessed

Data request message No restrictions

Examples

Control message OUTPUT 700; "SPDΔ0"
 Sets front panel/rear panel output switching to 1/1 SPEED
 (front panel connector output).

Data request message When front panel/rear panel output switching is 1/1 SPEED (front
 panel connector output)
 OUTPUT 700; "SPD?"
 ENTER 700;B\$
 PRINT B\$
 ↓
 Outputs SPDΔ0 (CR/LF).

Note: When front panel/rear panel output switching is set to 1/4 SPEED (rear panel connector output), the setting display of the DATA group including the DATA/DATA TRACKING function and CLOCK1 group is turned OFF.

①9 DATA output amplitude (Data amplitude)

Front panel/rear panel output switching: 1/1 SPEED			
Set value	Control message	Data request message	Output message
0.50 V _{P-P}	DAPΔ0.5	DAP?	DAPΔ0.500
:	:		:
2.00 V _{P-P}	DAPΔ2.00		DAPΔ2.000
Step value is in 0.01 V _{P-P} units.			

Front panel/rear panel output switching: 1/4 SPEED			
Set value	Control message	Data request message	Output message
0.50 V _{P-P}	DAPΔ0.5	DAP?	DAPΔ0.500
:	:		:
1.00 V _{P-P}	DAPΔ1.00		DAPΔ1.000
Step value is in 0.01 V _{P-P} units.			

Restrictions

Control message	Invalid in the following case: When floppy disk is being accessed
Data request message	No restrictions

Examples

Control message	OUTPUT 700; "DAPΔ0.5" Sets the DATA output amplitude to 0.50 V _{P-P} .
Data request message	When DATA output amplitude is 0.50 V _{P-P} . OUTPUT 700; "DAP?" ENTER 700;B\$ PRINT B\$ ↓ Outputs DAPΔ0.500 (CR/LF).

Note: The maximum DATA output amplitude differs with the front panel/rear panel output setting.

The third digit after the decimal point of the defined DATA output amplitude is rounded off.

20 DATA output offset (Data offset)

Front panel/rear panel output switching: 1/1 SPEED,
DATA/DATA/CLOCK output offset reference value: VOH

Set value	Control message	Data request message	Output message
- 2.000 V	DOSΔ - 2.0	DOS?	DOSΔ - 2.000
:	:		:
+ 2.000 V	DOSΔ 2.0		DOSΔΔ 2.000

Step value is in 0.005 V units.

Front panel/rear panel output switching: 1/1 SPEED,
DATA/DATA/CLOCK output offset reference value: VTH

Set value	Control message	Data request message	Output message
- 3.000 V	DOSΔ - 3.0	DOS?	DOSΔ - 3.000
:	:		:
+ 1.750 V	DOSΔ 1.75		DAPΔΔ 1.750

Step value is in 0.005 V units.

Front panel/rear panel output switching: 1/1 SPEED,
DATA/DATA/CLOCK output offset reference value: VOL

Set value	Control message	Data request message	Output message
- 4.000 V	DOSΔ - 4.0	DOS?	DOSΔ - 4.000
:	:		:
+ 1.500 V	DOSΔ 1.5		DOSΔΔ 1.500

Step value is in 0.005 V units.

Front panel/rear panel output switching: 1/4 SPEED,
DATA/DATA/CLOCK output offset reference value: VOH

Set value	Control message	Data request message	Output message
- 1.500 V	DOSΔ - 1.5	DOS?	DOSΔ - 1.500
:	:		:
+ 1.500 V	DOSΔ 1.5		DOSΔΔ 1.500

Step value is in 0.005 V units.

Front panel/rear panel output switching: 1/4 SPEED, DATA/DATA/CLOCK output offset reference value: VTH			
Set value	Control message	Data request message	Output message
- 2.000 V	DOSΔ - 2.0	DOS?	DOSΔ - 2.000
:	:		:
+ 1.250 V	DOSΔ 1.25		DOSΔΔ 1.250
Step value is in 0.005 V units.			

Front panel/rear panel output switching: 1/4 SPEED, DATA/DATA/CLOCK output offset reference value: VOL			
Set value	Control message	Data request message	Output message
- 2.500 V	DOSΔ - 2.5	DOS?	DOSΔ - 2.500
:	:		:
+ 1.000 V	DOSΔ 1.0		DOSΔΔ 1.000
Step value is in 0.005 V units.			

Restrictions

Control message	Invalid in the following case: When floppy disk is being accessed
Data request message	No restrictions

Examples

Control message	When front panel/rear panel output switching is 1/1 SPEED and DATA/DATA/CLOCK output offset reference value is VOH OUTPUT 700; "DOSΔ-2.0" Sets the DATA output offset to - 2.0 V.
Data request message	When DATA output offset is - 2.0 V OUTPUT 700; "DOS?" ENTER 700;B\$ PRINT B\$ ↓ Outputs DOSΔ - 2.000 (CR/LF).

Note: When the third digit after the decimal point of the defined DATA output offset is not an effective value (other than in 0.005 V units), it is changed to the effective value shown below.

1mV, 2mV→0mV; 3mV, 4mV, 6mV, 7mV→5mV; 8mV, 9mV→10mV

The DATA output offset setting range differs with the DATA output amplitude set value as shown in Figs. 11-1 to 11-6 on the following pages.

- 1/1 SPEED

Offset reference value: VOH

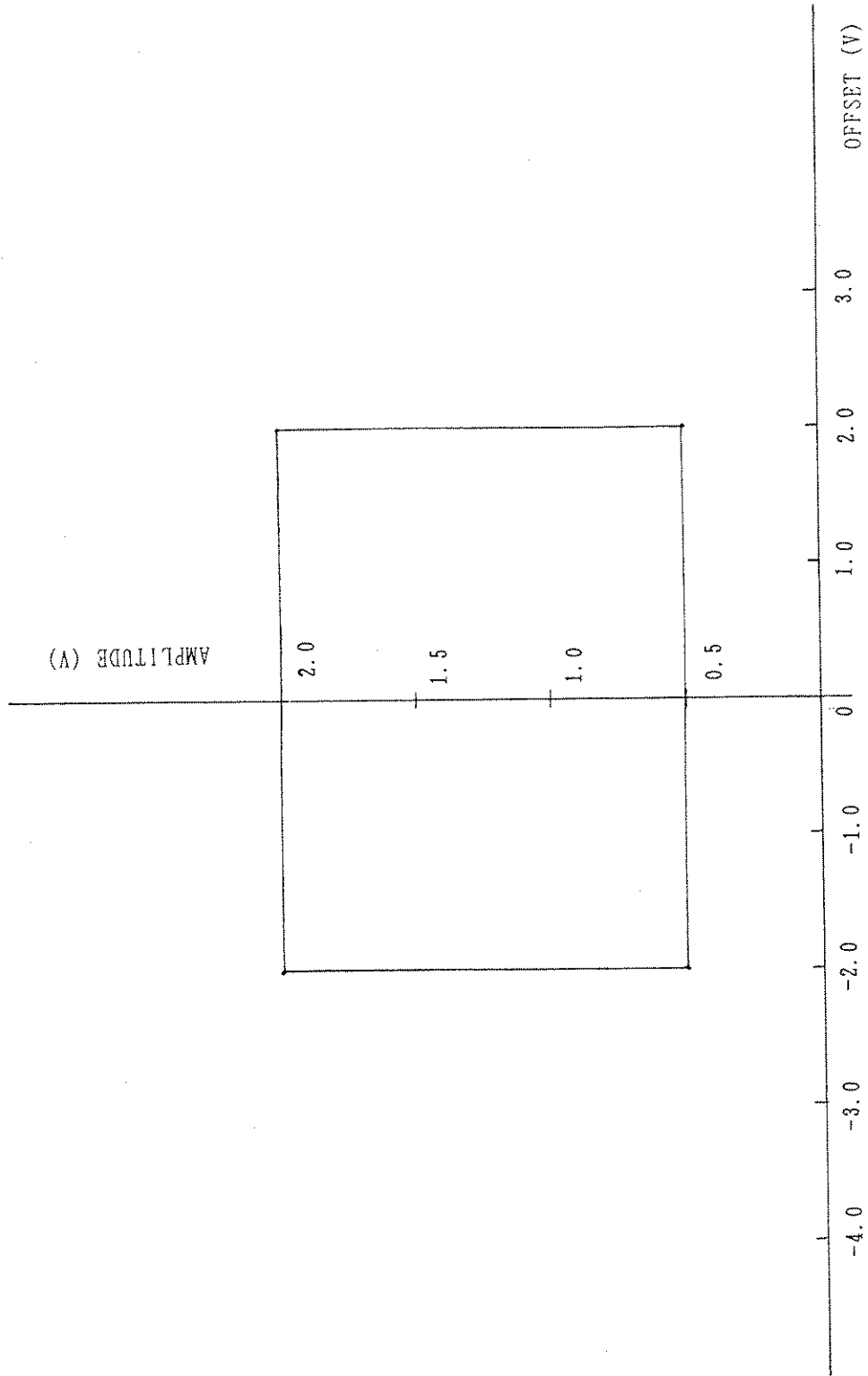


Fig. 11-1-1 Setting Range of Each Output Amplitude and Offset Voltage for Offset Reference Value

- 1/1 SPEED

Offset reference value: VTH

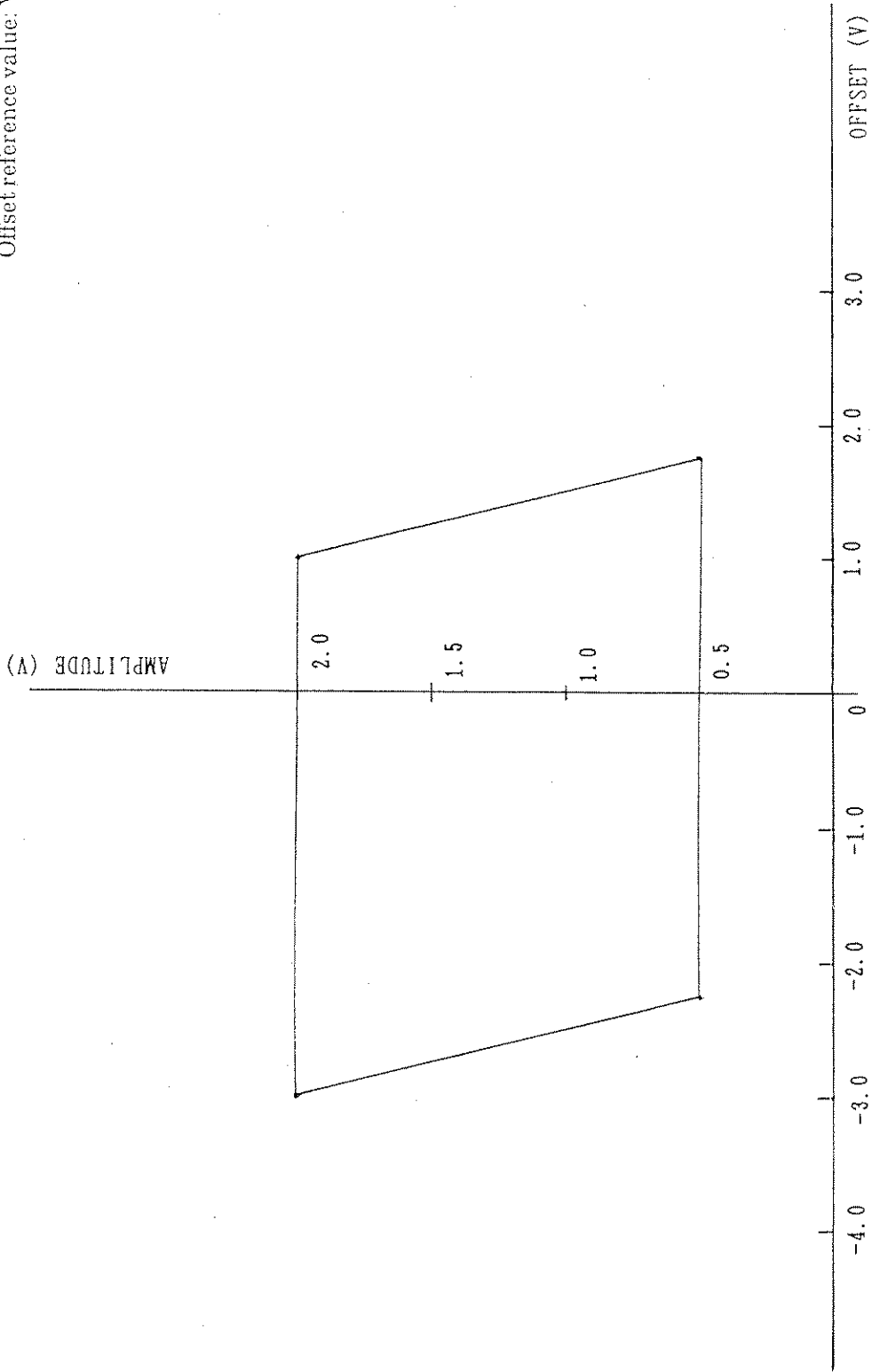


Fig. 11-2 Setting Range of Each Output Amplitude and Offset Voltage for Offset Reference Value

- 1/1 SPEED

Offset reference value: VOL

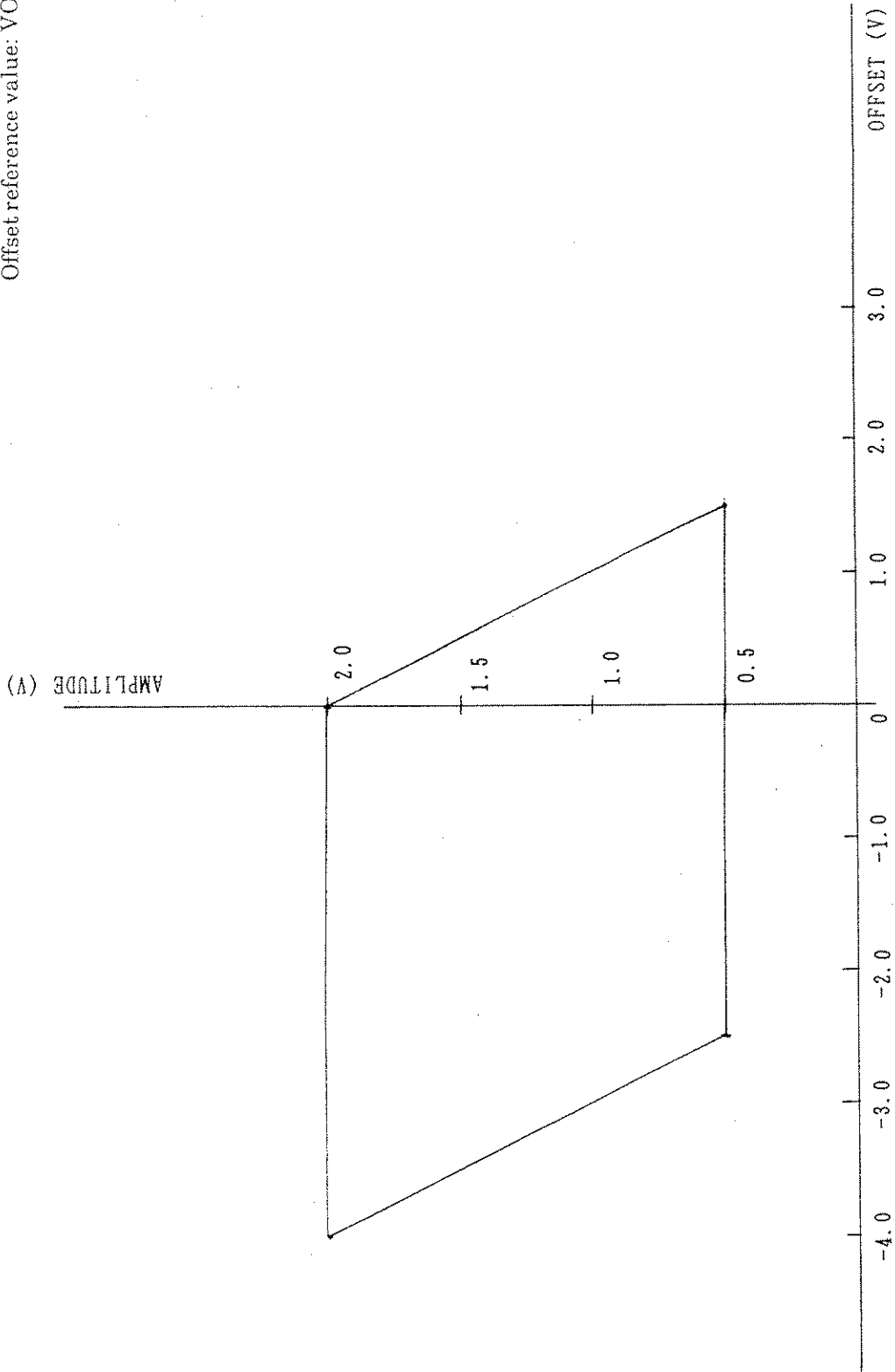


Fig. 11-3 Setting Range of Each Output Amplitude and Offset Voltage for Offset Reference Value

- 1/4 SPEED

Offset reference value: VOH

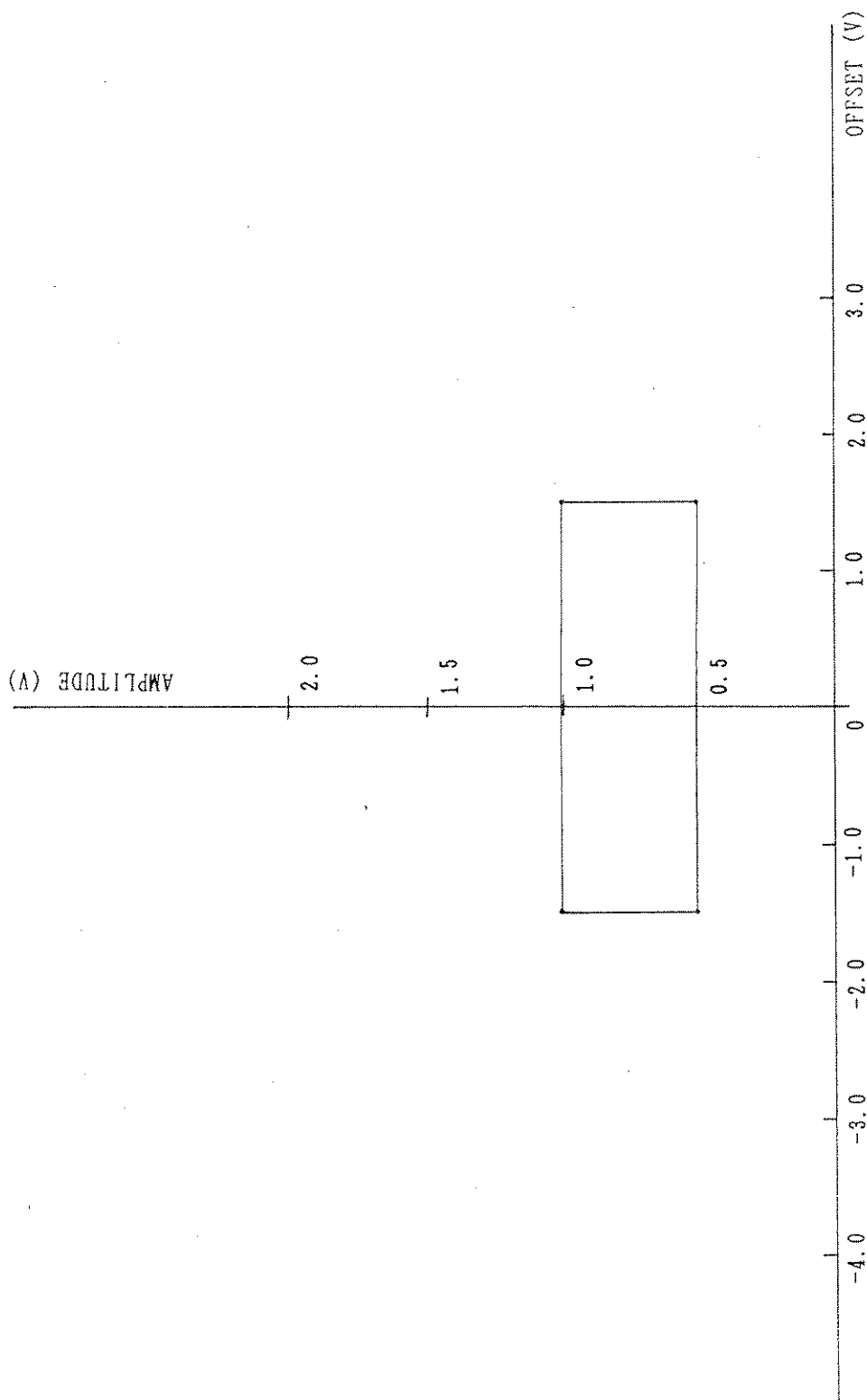


Fig. 11-4 Setting Range of Each Output Amplitude and Offset Voltage for Offset Reference Value

- 1/4 SPEED

Offset reference value: VTH

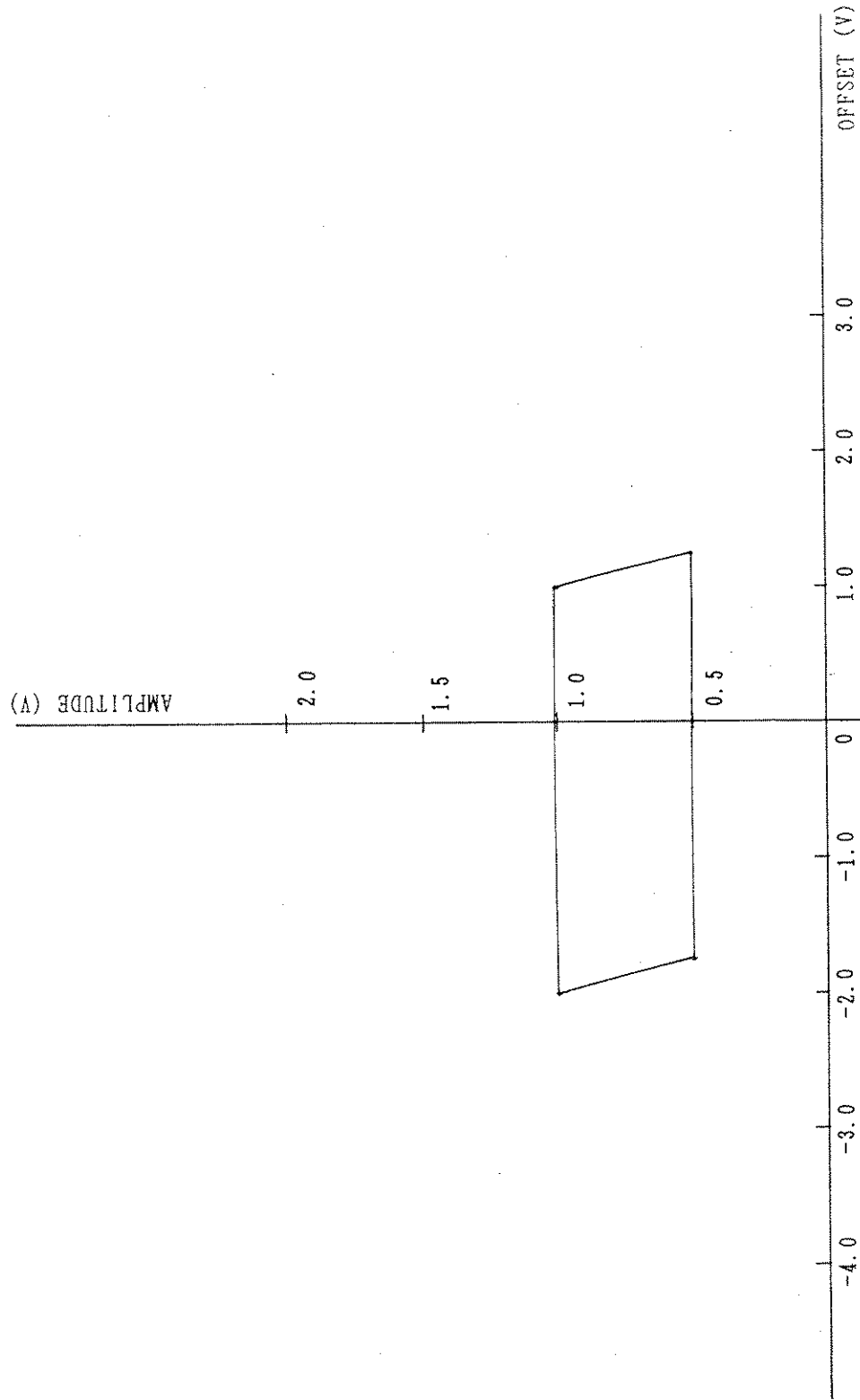


Fig. 11-5 Setting Range of Each Output Amplitude and Offset Voltage for Offset Reference Value

- 1/4 SPEED

Offset reference value: VOL

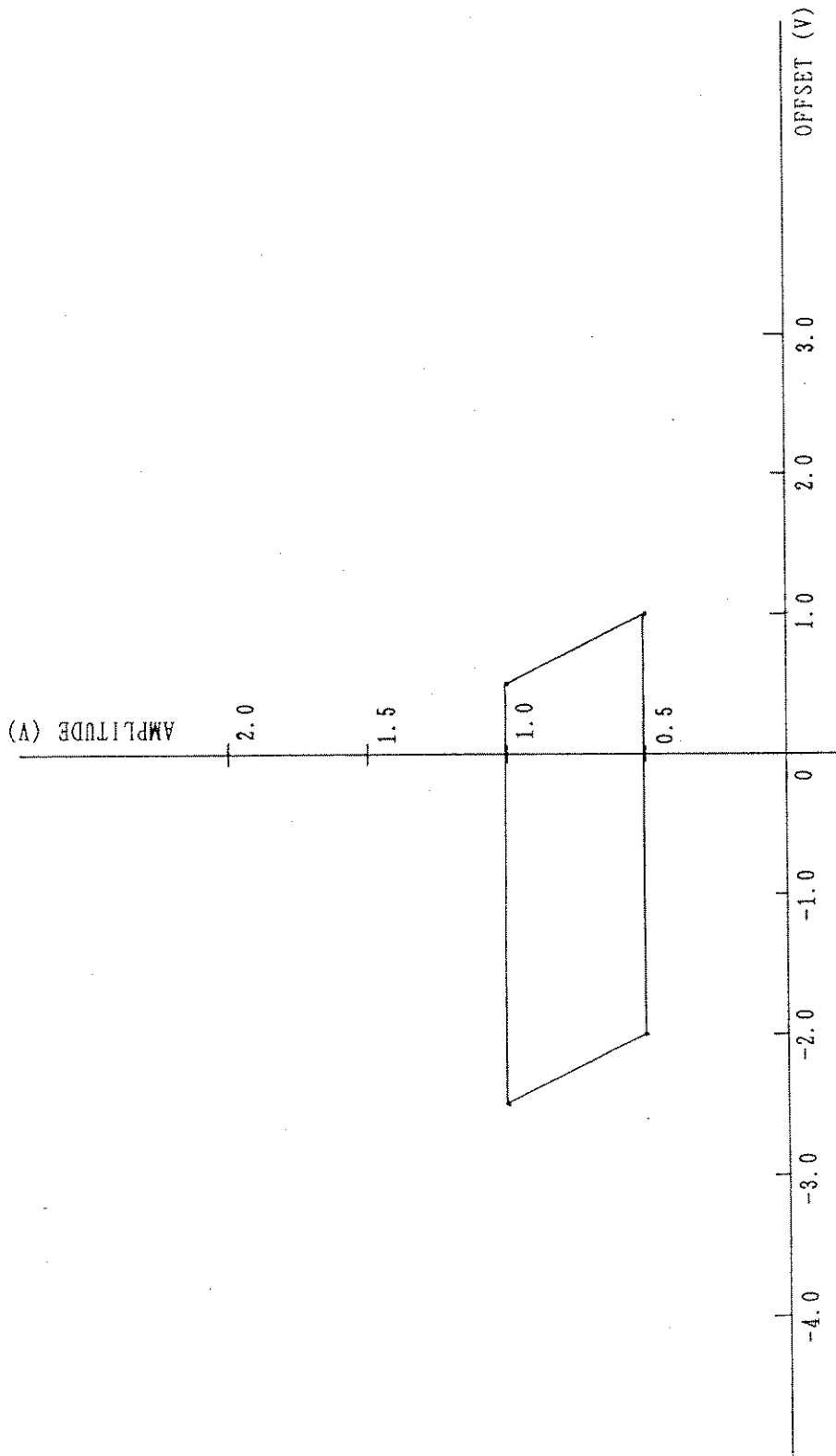


Fig. 11-6 Setting Range of Each Output Amplitude and Offset Voltage for Offset Reference Value

②1 DATA/DATA tracking (Data/Data tracking)

Set value	Control message	Data request message	Output message
OFF (no tracking)	TRKΔ0	TRK?	TRKΔ0
ON (tracking)	TRKΔ1		TRKΔ1

Restrictions

- Control message Invalid in the following cases:
 When front panel/rear panel output switching is 1/4 SPEED (rear panel connector output)
 When floppy disk is being accessed
- Data request message Invalid in the following case and ERR is output:
 When front panel/rear panel output switching is 1/4 SPEED (rear panel connector output)

Examples

- Control message When front panel/rear panel output switching is 1/1 SPEED (front panel connector output)
 OUTPUT 700; "TRKΔ0"
 Switches DATA/DATA output setting to tracking OFF (no tracking).
- Data request message When DATA/DATA output setting is tracking OFF (no tracking)
 OUTPUT 700; "TRK?"
 ENTER 700;B\$
 PRINT B\$
 ↓
 Outputs TRKΔ0 (CR/LF).
- When front panel/rear panel output switching is 1/4 SPEED (rear panel connector output)
 OUTPUT 700; "TRK?"
 ENTER 700;B\$
 PRINT B\$
 ↓
 Outputs ERR (CR/LF).

Note: The tracking OFF function sets the DATA and DATA outputs independently.
 The tracking ON function sets the DATA set values automatically to the DATA set values.

②② DATA output amplitude (Inverted data amplitude)

Front panel/rear panel output switching: 1/1 SPEED			
Set value	Control message	Data request message	Output message
0.50 V _{P-P}	NAPΔ0.5	NAP?	NAPΔ0.500
⋮	⋮		⋮
2.00 V _{P-P}	NAPΔ2.00		NAPΔ2.000
Step value is in 0.01 V _{P-P} units.			

Restrictions

- Control message Invalid in the following cases:
 - When tracking function is ON
 - When front panel/rear panel output switching is 1/4 SPEED (rear panel connector output)
 - When floppy disk is being accessed
- Data request message Invalid in the following cases and ERR is output:
 - When tracking function is ON
 - When front panel/rear panel output switching is 1/4 SPEED (rear panel connector output)

Examples

- Control message When tracking function is OFF and front panel/rear panel output switching is 1/1 SPEED (front panel connector output)
 - OUTPUT 700; "NAPΔ0.5"
 - Sets the DATA output amplitude to 0.50 V_{P-P}.

Data request message

When DATA output amplitude is 0.50 V_{P-P}.

OUTPUT 700; "NAP?"

ENTER 700;B\$

PRINT B\$

↓

Outputs NAPΔ0.500 (CR/LF).

When tracking function is ON or front panel/rear panel output switching is 1/4 SPEED (rear panel connector output)

OUTPUT 700; "NAP?"

ENTER 700;B\$

PRINT B\$

↓

Outputs ERR (CR/LF).

Note: The third digit after the decimal point of the defined DATA output amplitude is rounded off.

23 DATA output offset (Inverted data offset)

Front panel/rear panel output switching: 1/1 SPEED DATA/DATA/CLOCK output offset reference value: VOH			
Set value	Control message	Data request message	Output message
-2.000 V	NOSΔ-2.0	NOS?	NOSΔ-2.000
:	:		:
+2.000 V	NOSΔ2.0		NOSΔΔ2.000
Step value is in 0.005 V units.			

Front panel/rear panel output switching: 1/1 SPEED DATA/DATA/CLOCK output offset reference value: VTH			
Set value	Control message	Data request message	Output message
-3.000 V	NOSΔ-3.0	NOS?	NOSΔ-3.000
:	:		:
+1.750 V	NOSΔ1.75		NOSΔΔ1.750
Step value is in 0.005 V units.			

Front panel/rear panel output switching: 1/1 SPEED DATA/DATA/CLOCK output offset reference value: VOL			
Set value	Control message	Data request message	Output message
-4.000 V	NOSΔ-4.0	NOS?	NOSΔ-4.000
:	:		:
+1.500 V	NOSΔ1.5		NOSΔΔ1.500
Step value is in 0.005 V units.			

Restrictions

Control message	Invalid in the following cases: When tracking function is ON When front panel/rear panel output switching is 1/4 SPEED (rear panel connector output) When floppy disk is being accessed
Data request message	Invalid in the following cases and ERR is output: When tracking function is ON When front panel/rear panel output switching is 1/4 SPEED (rear panel connector output)

Examples

Control message	When tracking function is OFF, front panel/rear panel output switching is 1/1 SPEED, and DATA/DATA/CLOCK output offset reference value is VOH OUTPUT 700; "NAP Δ - 2.0" Sets the DATA output offset to -2.0 V.
Data request message	When DATA output offset is -2.0 V OUTPUT 700; "NOS?" ENTER 700; B\$ PRINT B\$ ↓ Outputs NOS Δ - 2.000 (CR/LF). When tracking function is ON, or front panel/rear panel output switching is 1/4 SPEED (rear panel connector output) OUTPUT 700; "NOS?" ENTER 700; B\$ PRINT B\$ ↓ Outputs ERR (CR/LF).

Note: When the third digit after the decimal point of the defined DATA output offset is not an effective value (other than in 0.005V units), it is changed to the effective value shown below.

1mV, 2mV → 0mV; 3mV, 4mV, 6mV, 7mV → 5mV; 8mV, 9mV → 10mV

The DATA output offset setting range differs with the DATA output amplitude set value as shown in Figs. 11-1 to 11-6.

②④ Delay time between CLOCK outputs (Clock delay)

MP1701B, MP1608A, MP1755A			
Set value	Control message	Data request message	Output message
+ 500 ps	CDLΔ500	CDL?	CDLΔΔΔ500
⋮	⋮		⋮
- 500 ps	CDLΔ - 500		CDLΔΔ - 500
Step value is in 1 ps unit.			

MP1605A			
Set value	Control message	Data request message	Output message
+ 1000 ps	CDLΔ1000	CDL?	CDLΔΔ1000
⋮	⋮		⋮
- 1 000 ps	CDLΔ - 1000		CDLΔ - 1000
Step value is in 2 ps units.			

Restrictions

Control message

Invalid in the following cases:

When front panel/rear panel output switching is 1/4 SPEED (rear panel connector output)

When floppy disk is being accessed

Data request message

Invalid in the following case and ERR is output:

When front panel/rear panel output switching is 1/4 SPEED (rear panel connector output)

Examples

Control message

When front panel/rear panel output switching is 1/1 SPEED (front panel connector output)

```
OUTPUT 700; "CDLΔ500"
```

Sets the CLOCK output delay time to + 500 ps.

Data request message

When CLOCK output delay time is

+ 500 ps

```
OUTPUT 700; "CDL?"
```

```
ENTER 700;B$
```

```
PRINT B$
```

↓

Outputs CDLΔΔΔ500 (CR/LF).

When front panel/rear panel output switching is 1/4 SPEED (rear panel connector output)

```
OUTPUT 700; "CDL?"
```

```
ENTER 700;B$
```

```
PRINT B$
```

↓

Outputs ERR (CR/LF).

②5 **CLOCK output amplitude (Clock amplitude)**

Front panel/rear panel output switching: 1/1 SPEED			
Set value	Control message	Data request message	Output message
0.50 V _{P-P}	CAPΔ 0.5	CAP?	CAPΔ0.500
⋮	⋮		⋮
2.00 V _{P-P}	CAPΔ2.00		CAPΔ2.000
Step value is in 0.01 V _{p-p} units			

Front panel/rear panel output switching: 1/4 SPEED			
Set value	Control message	Data request message	Output message
0.50 V _{P-P}	CAPΔ 0.5	CAP?	CAPΔ0.500
⋮	⋮		⋮
1.00 V _{P-P}	CAPΔ1.00		CAPΔ1.000
Step value is in 0.01 V _{p-p} units			

Restrictions

- Control message Invalid in the following case:
 When floppy disk is being accessed
- Data request message No restrictions

Examples

- Control message OUTPUT 700; "CAPΔ0.5"
 Sets the CLOCK output amplitude to 0.50 V_{P-P}.
- Data request message When CLOCK output amplitude is 0.50 V_{P-P}
 OUTPUT 700; "CAP?"
 ENTER 700;B\$
 PRINT B\$
 ↓
 Outputs CAPΔ0.500 (CR/LF).

Note: The maximum CLOCK output amplitude differs with the front panel/rear panel output setting.
The third digit after the decimal point of the defined CLOCK output amplitude is rounded off.

26) **CLOCK output offset (Clock offset)**

Front panel/rear panel output switching: 1/1 SPEED DATA/DATA/CLOCK output offset reference value: VOH			
Set value	Control message	Data request message	Output message
-2.000 V	COS Δ -2.0	COS?	COS Δ -2.000
:	:		:
+2.000 V	COS Δ 2.0		COS $\Delta\Delta$ 2.000
Step value is in 0.005 V units.			

Front panel/rear panel output switching: 1/1 SPEED DATA/DATA/CLOCK output offset reference value: VTH			
Set value	Control message	Data request message	Output message
-3.000 V	COS Δ -3.0	COS?	COS Δ -3.000
:	:		:
+1.750 V	COS Δ 1.75		COS $\Delta\Delta$ 1.750
Step value is in 0.005 V units.			

Front panel/rear panel output switching: 1/1 SPEED DATA/DATA/CLOCK output offset reference value: VOL			
Set value	Control message	Data request message	Output message
-4.000 V	COS Δ -4.0	COS?	COS Δ -4.000
:	:		:
+1.500 V	COS Δ 1.5		COS $\Delta\Delta$ 1.500
Step value is in 0.005 V units.			

Front panel/rear panel output switching: 1/4 SPEED DATA/DATA/CLOCK output offset reference value: VOH			
Set value	Control message	Data request message	Output message
-1.500 V	COS Δ -1.5	COS?	COS Δ -1.500
:	:		:
+1.500 V	COS Δ 1.5		COS $\Delta\Delta$ 1.500
Step value is in 0.005 V units.			

Front panel/rear panel output switching: 1/4 SPEED DATA/DATA/CLOCK output offset reference value: VTH			
Set value	Control message	Data request message	Output message
-2.000 V	COSΔ - 2.0	COS?	COSΔ - 2.000
⋮	⋮		⋮
+1.250 V	COSΔ1.25		COSΔΔ1.250
Step value is in 0.005 V units.			

Front panel/rear panel output switching: 1/4 SPEED DATA/DATA/CLOCK output offset reference value: VOL			
Set value	Control message	Data request message	Output message
-2.500 V	COSΔ - 2.5	COS?	COSΔ - 2.500
⋮	⋮		⋮
+1.000 V	COSΔ1.0		COSΔΔ1.000
Step value is in 0.005 V units.			

Restrictions

- Control message Invalid in the following case:
When floppy disk is being accessed
- Data request message No restrictions

Examples

- Control message When front panel/rear panel output switching is 1/1 SPEED and DATA/DATA/CLOCK output offset reference value is VOH
OUTPUT 700; "COSΔ - 2.0"
Sets the CLOCK output offset to -2.0 V.
- Data request message When CLOCK output offset is -2.0 V
OUTPUT 700; "COS?"
ENTER 700;B\$
PRINT B\$
↓
Outputs COSΔ - 2.000 (CR/LF).

Note: When the third digit after the decimal point of the defined CLOCK output offset is not an effective value (other than in 0.005 V units), it is changed to the effective value shown below.

1mV, 2mV→0mV; 3mV, 4mV, 6mV, 7mV→5mV; 8mV, 9mV→10mV

The CLOCK output offset setting range differs with the CLOCK output amplitude set value as shown in Figs. 11-1 to 11-6.

11.4 MEMORY Section

The MEMORY section GP-IB commands are described in the following pages. The Δ in the text means a space.

②⑦ Memory function switching (Memory mode ptn/other)

Set value	Control message	Data request message	Output message
PTN mode	MEMΔ0	MEM?	MEMΔ0
OTH mode	MEMΔ1		MEMΔ1

Restrictions

Control message	Invalid in the following cases: When MEMORY display is in error display mode When floppy disk is being accessed
Data request message	No restrictions

Examples

Control message	OUTPUT 700; "MEMΔ0" Switches the memory function to the PTN mode.
-----------------	--

Data request message	When memory function is in PTN mode OUTPUT 700; "MEM?" ENTER 700;B\$ PRINT B\$ ↓ Outputs MEMΔ (CR/LF).
----------------------	---

Note: "PTN mode" means the PATTERN mode. That is, only pattern data is the objective of the memory function.

"OTH mode" means the OTHERS mode. That is, other than pattern data is the objective of the memory function.

28 File No./directory mode switching (File no./directory mode)

Set value	Control message	Data request message	Output message
FILE No.	FILΔ0	FIL?	FILΔ0
DIR	FILΔ1		FILΔ1

Restrictions

Control message Invalid in the following cases:
 When MEMORY display is in error display mode
 When floppy disk is being accessed

Data request message No restrictions

Examples

Control message OUTPUT 700; "FILΔ0"
 Switches the memory function to the FILE No. mode.

 OUTPUT 700; "FILΔ1"
 Switches the memory function to the DIR mode.
 At that time, the inserted floppy disk is accessed and the
 directory information is stored.

Data request message When memory function is in FILE No. mode
 OUTPUT 700; "FIL?"
 ENTER 700;B\$
 PRINT B\$
 ↓
 Outputs FILΔ0(CR/LF).

 When memory system is in DIR mode
 OUTPUT 700; "FIL?"
 LOOP
 ENTER 700;B\$
 PRINT B\$
 EXIT IF B\$="FILΔ1"
 END LOOP
 Outputs the directory information and FILΔ1 (CR/LF) in the
 format shown on the next page.

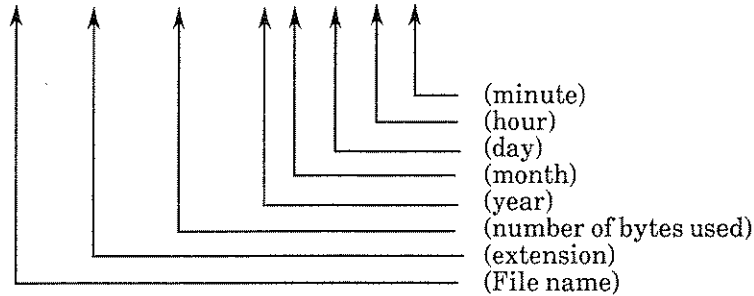
Note: The MP1701B/MP1755A/MP1608A/MP1650A does not have floppy disk insertion detection function.

Therefore, after inserting a floppy disk, set the directory mode to store the inserted floppy disk directory information to the RAM area of the MP1701B/MP1608A.

Directory information and FILΔ1 (CR/LF) output format

- Existing directory information (output by the amount of number of files)

***** , *** , ***** , **--** , **:** (CR/LF)



* marks above indicate the file name, extension (file type), number of bytes used, date, and time.

Each item has fixed length as follows:

File name:	8 digits
Extension (file type):	3 digits
Number of bytes used:	8 digits
Date:	2 digits each
Time:	2 digits each

- Setting state

FILΔ1 (CR/LF + EOI)

Example: An actual output example is shown below.

```
T01ΔΔΔΔΔ,PTN, ΔΔΔ65640, 89-03-27,13:45
T99ΔΔΔΔΔ,OTH,ΔΔΔΔΔ108, 89-03-27,09:01
FILΔ1
```


29 Data recall (Data recall)

File name	Control message	Data request message	Output message
0 ⋮ 99	RCLΔ0 ⋮ RCLΔ99	None	_____

Restrictions

Control message Invalid in the following case:
 When floppy disk is being accessed

Examples

Control message OUTPUT 700; "RCLΔ0"
 Reads, displays, and sets the data of the file name on floppy disk
 by the set memory function.

Note: When the specified file does not exist, an error is generated.
This error display is cleared when a data recall, data save, or data resave command is set.

③ Data save (Data save)

File name	Control message	Data request message	Output message
0	SAV△0	None	_____
⋮	⋮		
99	SAV△99		

Restrictions

Control message Invalid in the following cases:
 In directory (DIR) mode
 When floppy disk is being accessed

Examples

Control message When memory system is FILE No. mode
 OUTPUT 700; "SAV△0"
 Stores the data using the specified file name under the
 conditions of the set memory function.

Note: When the specified file name exists already, an error is generated.

This error display is cleared when a data recall, data save, or data resave command is set.

③① Data resave (Data resave)

File name	Control message	Data request message	Output message
0 ⋮ 99	RSVΔ0 ⋮ RSVΔ99	None	_____

Restrictions

Control message Invalid in the following case:
 When floppy disk is being accessed

Examples

Control message OUTPUT 700; "RSVΔ0"
 Overwrites the data using the specified file name under the
 conditions of the set memory function.

Note: When the specified file does not exist, an error is generated.

This error display is cleared when a data recall, data save, or data resave command is set.

11.5 Rear Panel Section

The rear panel section GP-IB commands are described on the following pages. The Δ in the text means a space.

③2 Number of external pattern input channel (1/8 Speed ch select)

Set value	Control message	Data request message	Output message
1 to 8 ch (INT) 1 ch only (EXT) ⋮	SCHΔ#B00000000 SCHΔ#B00000001 ⋮	SCH?	SCHΔ#B00000000 SCHΔ#B00000001 ⋮
2 to 8 ch (EXT) 1 to 8 ch (EXT)	SCHΔ#B11111110 SCHΔ#B11111111		SCHΔ#B11111110 SCHΔ#B11111111

Sets INT or EXT of each number of external pattern input channel from the value of each bit of the NR field.
 0: INT (internal pattern input)
 1: EXT (external pattern input)

Restrictions

Control message Invalid in the following case:
 When floppy disk is being accessed

Data request message No restrictions

Examples

Control message OUTPUT 700; "SCHΔ#B00000001"
 Sets number of external pattern input channel: 1 ch only to EXT.

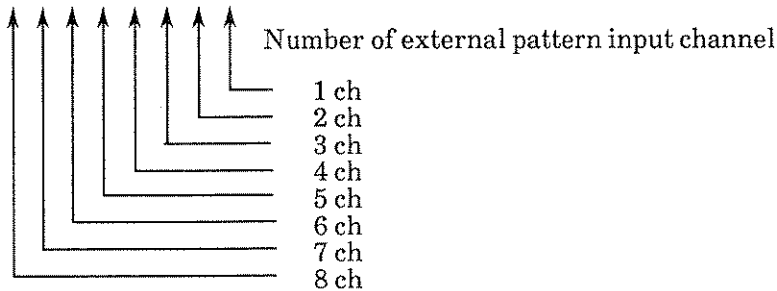
Data request message When number of external pattern input channel: 1 ch and 5 ch only are EXT
 OUTPUT 700; "SCH?"
 ENTER 700;B\$
 PRINT B\$



Outputs SCHΔ#B00010001 (CR/LF).

Note:

SCHΔ#B * * * * *



When multiple conditions are set, the channels are combined.

③ Error additional channel (Error addition channel)

MP1701B, MP1755A			
Set value	Control message	Data request message	Output message
1 ch	ECHΔ 1	ECH?	ECHΔΔ1
:	:		:
32 ch	ECHΔ32		ECHΔ32

MP1608A, MP1650A			
Set value	Control message	Data request message	Output message
1 ch	ECHΔ 1	ECH?	ECHΔΔ1
:	:		:
16 ch	ECHΔ16		ECHΔ16

Restrictions

- Control message Invalid in the following cases:
 When code error addition function is OFF
 When floppy disk is being accessed
- Data request message Invalid in the following cases and ERR is output:
 When code error addition function is OFF

Examples

- Control message When code error addition function is other than OFF
 OUTPUT 700; "ECHΔ1"
 Sets the number of error addition channel to 1 ch.
- Data request message When number of error addition channel is 1 ch
 OUTPUT 700; "ECH?"
 ENTER 700;B\$
 PRINT B\$
 ↓
 Outputs ECHΔΔ1 (CR/LF).
 When code error addition function is OFF
 OUTPUT 700; "ECH?"
 ENTER 700;B\$
 PRINT B\$
 ↓
 Outputs ERR (CR/LF).

34 Number of mark ratio AND bit shifts (Mark ratio bit shift)

Set value	Control message	Data request message	Output message
1 bit shift	SFTΔ0	SFT?	SFTΔ0
3 bit shift	SFTΔ1		SFTΔ1

Restrictions

- Control message Invalid in the following cases:
 When output pattern is in PRGM. (WORD or DATA) mode
 When floppy disk is being accessed.
- Data request message Invalid in the following case and ERR is output:
 When output pattern is in PRGM. (WORD or DATA) mode

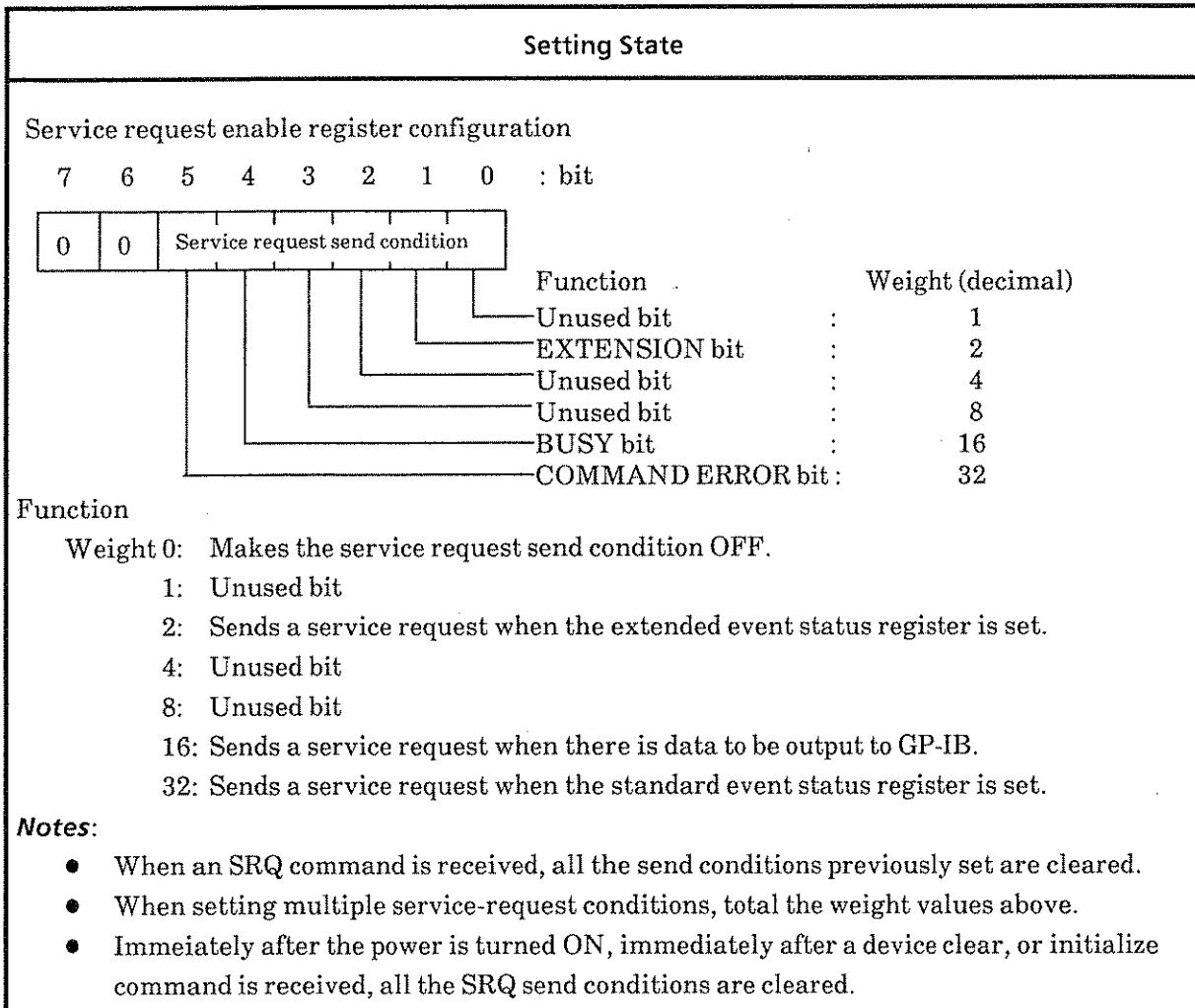
Examples

- Control message When output pattern is in PRBS mode
 OUTPUT 700; "SFTΔ0"
 Sets the number of mark ratio AND bit shifts to a 1 bit shift.
- Data request message When number of mark ratio AND bit shifts is 1 bit shift
 OUTPUT 700; "SFT?"
 ENTER 700;B\$
 PRINT B\$
 ↓
 Outputs SFTΔ0 (CR/LF).
 When output pattern is in PRGM. (WORD or DATA) mode
 OUTPUT 700; "SFT?"
 ENTER 700;B\$
 PRINT B\$
 ↓
 Outputs ERR (CR/LF).

11.6 Other Sections

The GP-IB commands of other sections are described on the following pages. The Δ in the text means a space.

③⑤ Service request enable register (Service request enable register)



Control message	Data request message	Output message
SRQ Δ 0	SRQ?	SRQ $\Delta\Delta$ 0
:		:
SRQ Δ 63		SRQ Δ 63

Restrictions

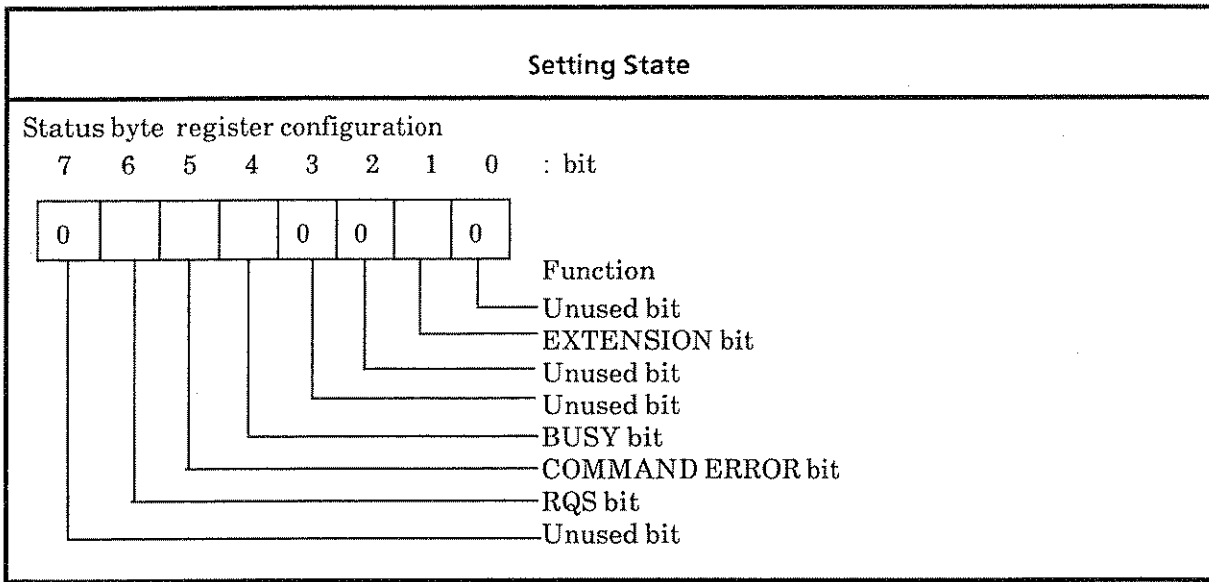
Control message	Invalid in the following case: When floppy disk is being accessed.
Data request message	No restrictions

Examples

Control message	OUTPUT 700; "SRQΔ2" Sends a service request when the extended event status register is set.
Data request message	When extended event status register is set, a service request send condition is sent. OUTPUT 700; "SRQ?" ENTER 700; B\$ PRINT B\$ ↓ Outputs SRQΔΔ2 (CR/LF).

Note: The service request enable register and GP-IB status byte are described in SECTION 12.

36 Status byte register (Status byte register?)



Control message	Data request message	Output message
_____	STB?	STBΔ#B*****

Restrictions

Data request message No restrictions

Examples

Data request message When GP-IB status byte is 01010000

OUTPUT 700; "STB?"

ENTER 700;B\$

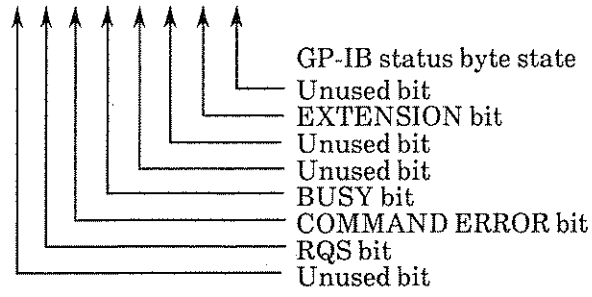
PRINT B\$

↓

Outputs STBΔ#B01010000 (CR/LF)

Note: This GP-IB status byte is the same as the data that is read by serial polling.

STBΔ#B * * * * * * * *



The status byte register and GP-IB status byte are described in SECTION 12.

37 Standard event status enable register (Event status enable register [Standard])

Setting State

Standard event status enable register configuration

7	6	5	4	3	2	1	0	: bit
0	0	Standard event status send condition						

	Function	Weight (decimal)
	Unused bit	1
	Unused bit	2
	QYE bit	4
	DDE bit	8
	EXE bit	16
	CMD bit	32

Function

- Weight 0: Turns the standard event status send condition to OFF.
- 1: Unused bit
- 2: Unused bit
- 4: Sends the standard event status send condition when a talker is designated even though the send buffer is empty or when the send buffer has overflowed.
- 8: Sends the standard even status send condition when the received command is outside the usage condition.
- 16: Sends the standard event status send condition in a numeric value error or when a HEX value is used in a command in which it cannot be used.
- 32: Sends the standard event status send condition when header error is present.

Note:

- When the ESE command is received, all the send conditions set previously are cleared.
- When setting multiple event-status-enable (ESE) conditions, total the weight values above.
- All the ESE send conditions are cleared immediately after the power is turned ON, and immediately after a device clear, or initialize command is received.

Control message	Data request message	Output message
ESE△0	ESE?	ESE△△0
⋮		⋮
ESE△63		ESE△63

Restrictions

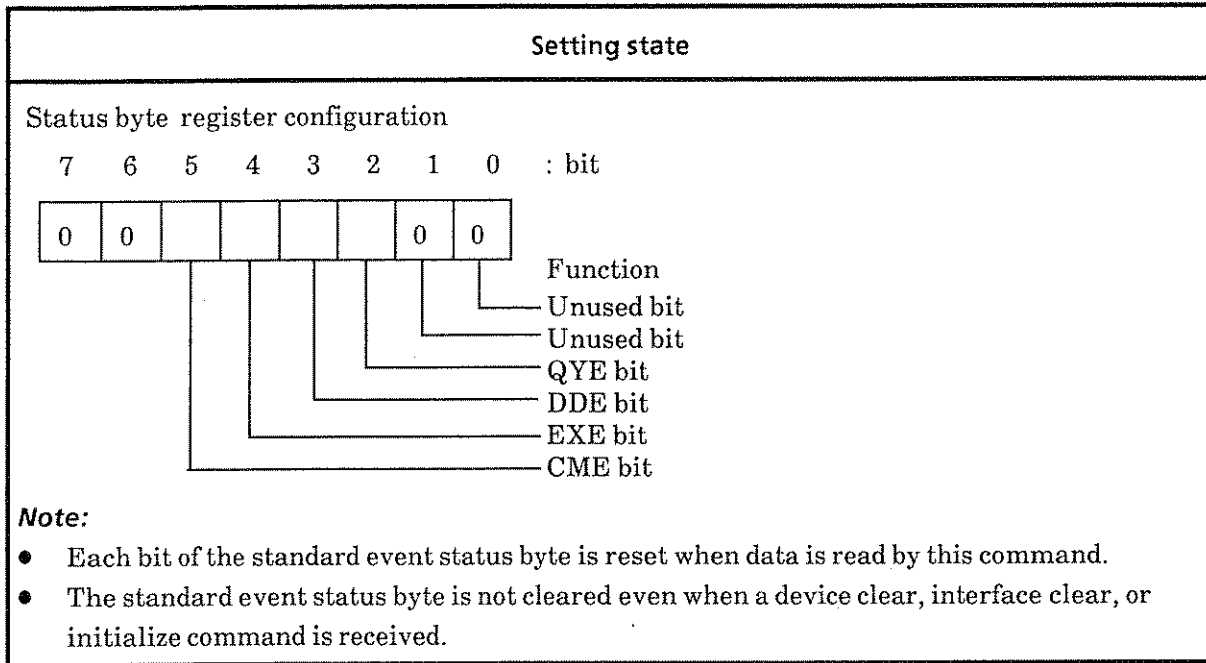
Control message	Invalid in the following case: When floppy disk is being accessed
Data request message	No restrictions

Examples

Control message	OUTPUT 700; "ESEΔ32" Sends the standard event status when a header error is generated.
Data request message	When standard event status send condition is sent at header error OUTPUT 700; "ESE?" ENTER 700;B\$ PRINT B\$ ↓ Outputs ESEΔ32 (CR/LF).

Note: The standard event status enable register and GP-IB status byte are described in SECTION 12.

38) Standard event status register (Event status byte register? [Standard])



Control message	Data request message	Output message
_____	ESR?	ESRΔ#B*****

Restrictions

Data request message No restrictions

Examples

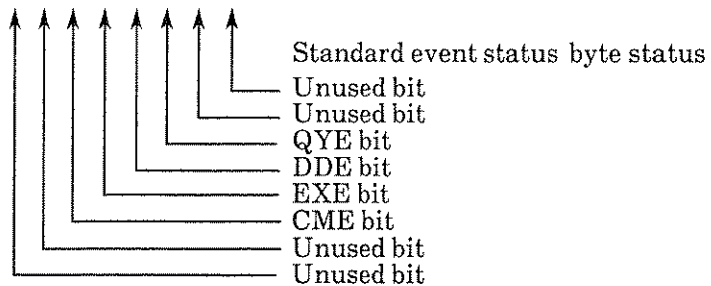
Data request message When status of standard event status byte is 00100000

```

OUTPUT 700; "ESR?"
ENTER 700; B$
PRINT B$
↓
Outputs ESRΔ#B00100000 (CR/LF).
    
```

Note: This standard event status byte is irrelevant when serial polling. This status is considered an extension byte of the COMMAND ERROR bit in the GP-IB status byte.

ESRΔ#B * * * * * * * *



The standard event status byte register and GP-IB status byte are described in SECTION 12.

39 Extended event status enable register (Event status enable register [Extension])

Setting state

Extended event status enable register configuration

7 6 5 4 3 2 1 0 : bit

0	0	Extended event status send condition
---	---	--------------------------------------

Function	Weight (decimal)
Unused bit	1
BACKUP ERROR bit	2
PLL UNLOCK bit	4
DELAY bit	8
PATTERN bit	16
FILE ACCESS bit	32

Function

Weight 0: Turns the extended event status (EES) send condition to OFF.

1: Unused bit

2: Sends the EES send condition at backup error.

4: Sends the EES send condition when the internal clock circuit PLL is unlocked.

8: Sends the EES send condition when the servo circuit of the clock delay circuit is changed from the BUSY to the READY state.

16: Sends the EES send condition at the end of a programmable pattern setting.

32: Sends the EES send condition at the end of floppy disk access function.

Note:

- When an EES command is received, all the send conditions set previously are cleared.
- When setting multiple EES conditions, total the weight values above.
- Immediately after the power is turned ON, immediately after a device clear or initialize command is received, all EES send conditions are cleared.

Control message	Data request message	Output message
EESΔ 0	EES?	EESΔΔ0
:		:
EESΔ63		EESΔ63

Restrictions

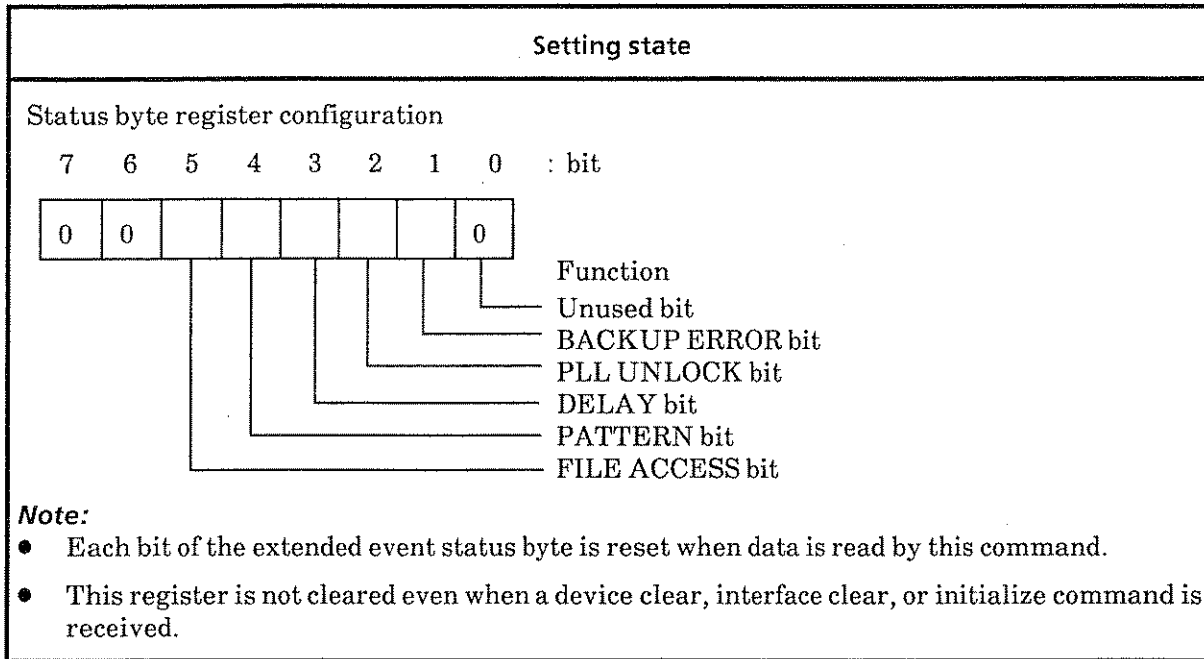
Control message	Invalid in the following case: When floppy disk is being accessed.
Data request message	No restrictions

Examples

Control message	OUTPUT 700; "EESΔ32" Sends the extended event status at the end of floppy disk access.
Data request message	When EES send condition is in the state to send at the end of floppy disk access function. OUTPUT 700; "EES?" ENTER 700;B\$ PRINT B\$ ↓ Outputs EESΔ32 (CR/LF).

Note: The extended event status enable register and GP-IB status byte are described in SECTION 12.

④ Extended event status register (Event status byte register? [Extension])



Control message	Data request message	Output message
—	EER?	EERΔ#B*****

Restrictions

Data request message No restrictions

Examples

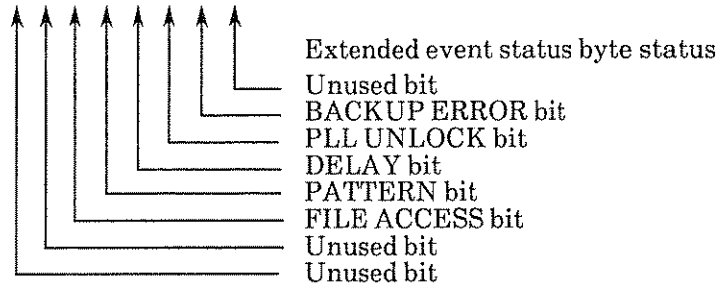
Data request message When status of extended event status byte is 01000000

```

OUTPUT 700; "EER?"
ENTER 700; B$
PRINT B$
↓
Outputs EERΔ#B00100000 (CR/LF).
    
```

Note: This extended event status byte is irrelevant when serial polling. This status is considered an extension byte of the EXTENSION bit in the GP-IB status byte.

EERΔ#B * * * * * * * *



The extended event status byte register and GP-IB status byte are described in SECTION 12.

④1 INITIALIZE (Initialize)

Setting state	Control message	Data request message	Output Message
Initialize	INI	_____	_____

Restrictions

Control message No restrictions

Examples

Control message OUTPUT 700; "INI"

 Sets the MP1701B/MP1755A/MP1608A/MP1650A to the factory setting state (forced initialization).

Note: "Forced initialization" is the same as the initialization operation when turning the power ON while pressing the [LOCAL] key.

If the floppy disk is being accessed, the access is aborted.

④ Internal timer setting (Real time setting)

Setting state	
RTMΔNR1, NR1, NR1, NR1, NR1, NR1	Function: Range
	Second: 0 to 59
	Minute: 0 to 59
	Hour: 0 to 23
	Day: 1 to 31 (leap year: 1 to 29)
	Month: 1 to 12
	Year: 0 to 99 (Gregorian calendar)

Control message	Data request message	Output message
RTMΔ00,01,01,00,00,00	RTM?	RTMΔ00,01,01,00,00,00
:		:
RTMΔ99,12,31,23,59,59		RTMΔ99,12,31,23,59,59

Restrictions

- Control message Invalid in the following case:
When floppy disk is being accessed
- Data request message No restrictions

Examples

- Control message OUTPUT 700; "RTMΔ90, 03, 10, 00, 10, 29"
Sets the internal timer to March 10, 1990, 0 hours 10 minutes 29 seconds, and starts the clock.
- Data request message When internal timer is March 10, 1990, 0 hours 10 minutes 29 seconds
OUTPUT 700; "RTM?"
ENTER 700; B\$
PRINT B\$
↓
Outputs RTMΔ90, 03, 10, 00, 10, 29 (CR/LF).

④ Power failure, power recovery (Power failure interval?)

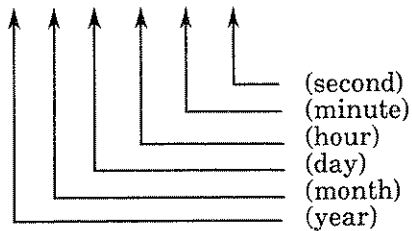
Setting state	Control message	Data request message
_____	_____	PWI?

Output message format

The power failure time, power recovery time, and interval time are output in the following format:

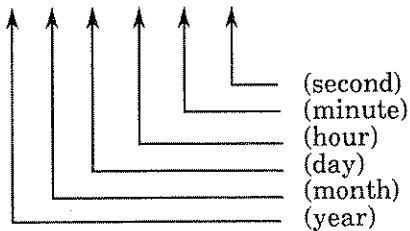
1. Power failure time

PWF△**, **, **, **, **, ** (CR/LF)



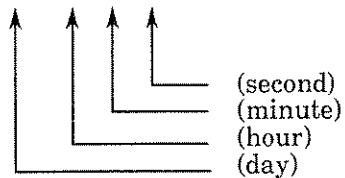
2. Power recovery time

PWR△**, **, **, **, **, ** (CR/LF)



3. Power failure interval time

PWI△△△△****, **, **, ** (CR/LF + EOI)



Example: An example of actual output is shown below.

PWF△89, 03, 01, 00, 00, 00 : Power failure time
 PWR△89, 03, 23, 13, 00, 00 : Power recovery time
 PWI△△△△△△△22, 13, 00, 00 : Power failure interval time

Restrictions

Data request message

Invalid in the following cases and ERR is output:

After device clear received

After initialization

Examples

Data request message

OUTPUT 700; "PWI?"

LOOP

ENTER 700;B\$

PRINT B\$

EXIT IF B\$[1,3]="PWI"

END LOOP

↓

Outputs the power failure time, power recovery time, and power failure interval time.

After device clear received or after initialization

OUTPUT 700; "PWI?"

ENTER 700;B\$

PRINT B\$

↓

Outputs ERR (CR/LF).

④ PLL lock status (PLL unlock?)

Setting state	Control message	Data request message	Output message
LOCK	—————	PLL?	PLLΔ0
UNLOCK			PLLΔ1

Restrictions

Data request message Invalid in the following case and ERR is output:
 When operation clock is externa clock (EXT)

Examples

Data request message When internal-clock-generator phase lock loop (PLL) is in LOCK state

```
OUTPUT 700; "PLL?"
ENTER 700; B$
PRINT B$
```

↓

Outputs PLLΔ0 (CR/LF).

When operation clock is an external clock (EXT)

```
OUTPUT 700; "PLL?"
ENTER 700; B$
PRINT B$
```

↓

Outputs ERR (CR/LF).

Note: "LOCK" indicates that the internal clock generator PLL is in the LOCK state.
 "UNLOCK" indicates that the internal clock generator PLL is in the UNLOCK state.

④5 Delay status (Delay unlock?)

Setting state	Control message	Data request message	Output message
READY	—	DLY?	DLYΔ0
BUSY			DLYΔ1

Restrictions

Data request message Invalid in the following case and ERR is output:
 When front panel/rear panel output switching (\bar{E} DISPLAY mode) is 1/4 SPEED

Examples

Data request message When servo subcircuit of clock delay circuit is in the READY state
 OUTPUT 700; "DLY?"
 ENTER 700;B\$
 PRINT B\$

↓

Outputs DLYΔ0 (CR/LF).

When DISPLAY mode is 1/4 SPEED

OUTPUT 700; "DLY?"
 ENTER 700;B\$
 PRINT B\$

↓

Outputs ERR (CR/LF).

Note: "READY" indicates that the servo subcircuit of the clock delay circuit is in the READY state. "BUSY" indicates that the servo subcircuit is in the BUSY state.

SECTION 12

GP-IB STATUS BYTE

The MP1701B/MP1755A/MP1608A/MP1650A has a status byte that outputs information on the bus when serial polling from the controller is received, and has a standard event status byte and extended event status byte (extension byte) that output information in response to data request command with no relation to serial polling.

Note: When multiple serial pollings are performed continuously as shown in the example below, always provide a WAIT time of about 0.1 s before performing the next serial polling.

This is because if serial polling is performed continuously without a WAIT time, this processing task is processed first and information is not sent to slave tasks because of the MP1701B/MP1755A/MP1608A/MP1650A software configuration.

```
Example:  ⋮
          LOOP
            A = SPOLL (700)
            EXIT IF BIT (A, 6) = 1
            WAIT . 1 ←———— Indicates WAIT time.
          END LOOP
          ⋮
```

Status information can also be read by data request message STB? without performing serial polling.

12.1 Status Byte Configuration

The configuration of each status byte is shown in Fig. 12-1.

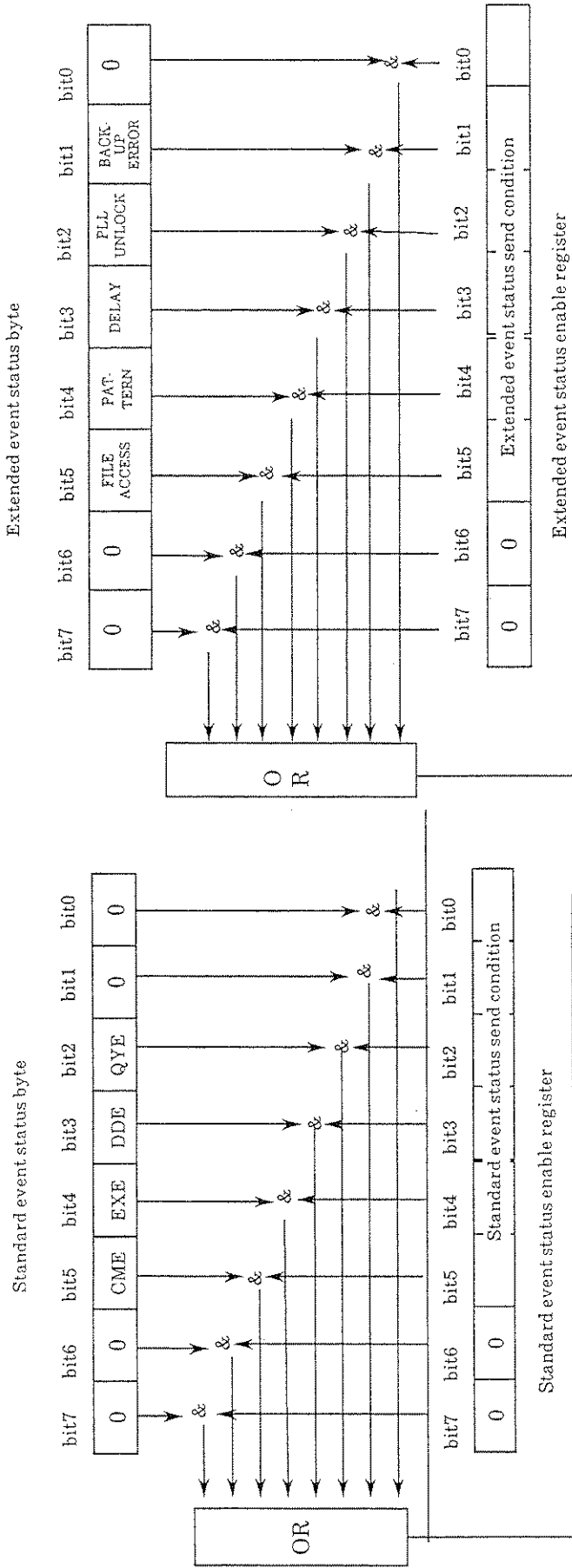
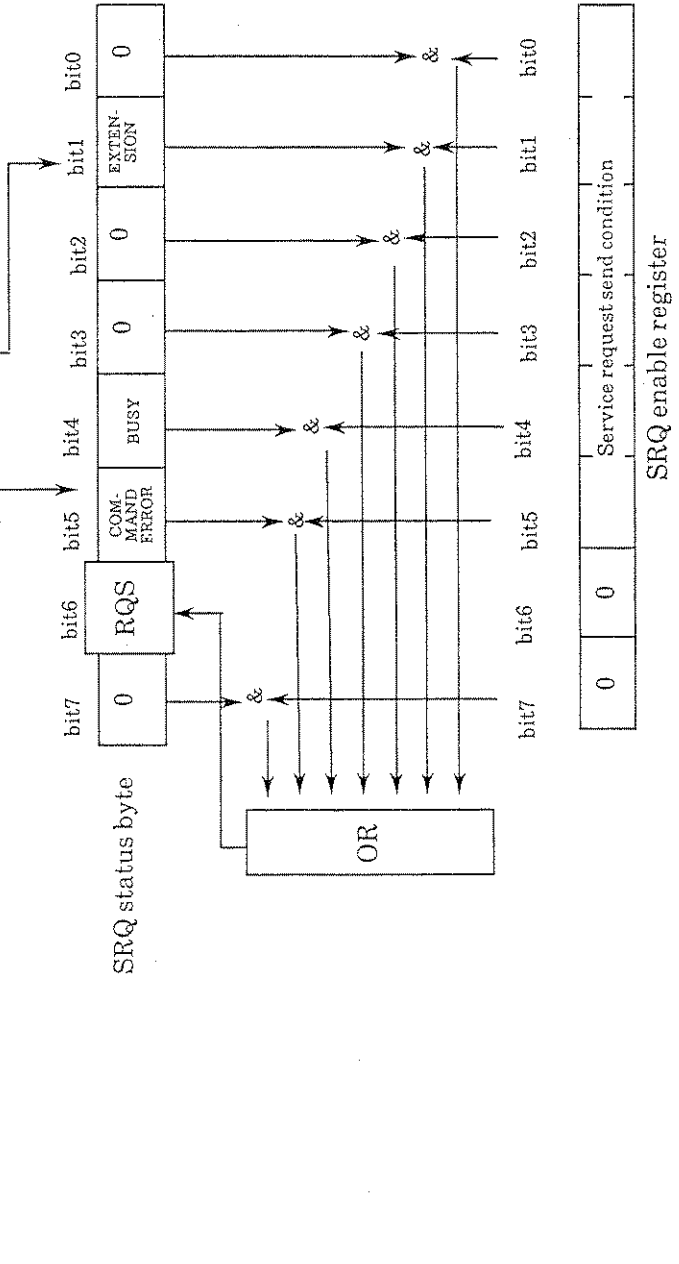


Fig. 12-1 Configuration of Each Status Byte and Enable B



12.2 Description of Each Register and Status Byte

12.2.1 SRQ enable register

Operation

This register indicates the GP-IB command service request send condition.

It turns the RQS bit ON and OFF in status byte bit units.

When the event of "1" bit at the SRQ enable register is generated, the RQS bit becomes "1" and an interrupt is sent to the controller.

- When an SRQ command is received, all the send conditions previously set are cleared.
- Multiple conditions can also be set.
- Immediately after the power is turned ON and immediately after a device clear or initialize command is received, all the SRQ send conditions are cleared.

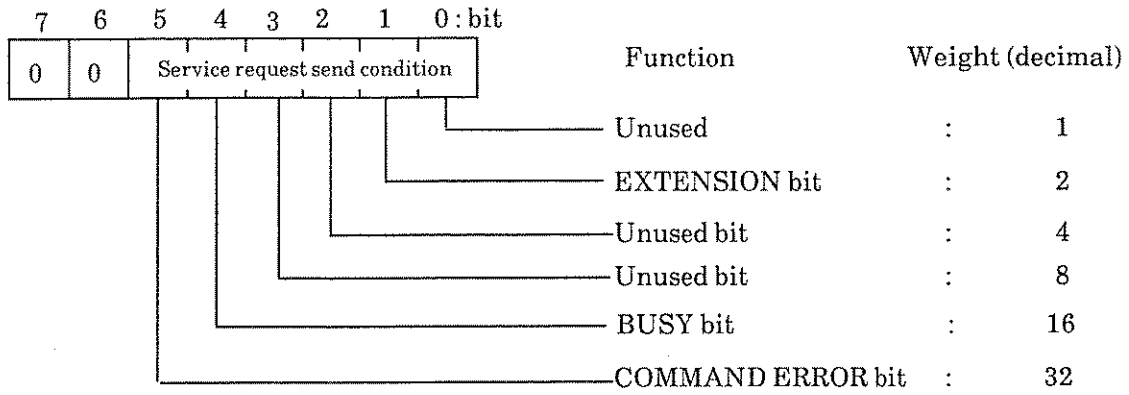
Command

Control message : SRQ Δ 0 to SRQ Δ 63

Data request message : SRQ?

Output message : SRQ $\Delta\Delta$ 0 (CR/LF) to SRQ Δ 63 (CR/LF)

Configuration



Function:

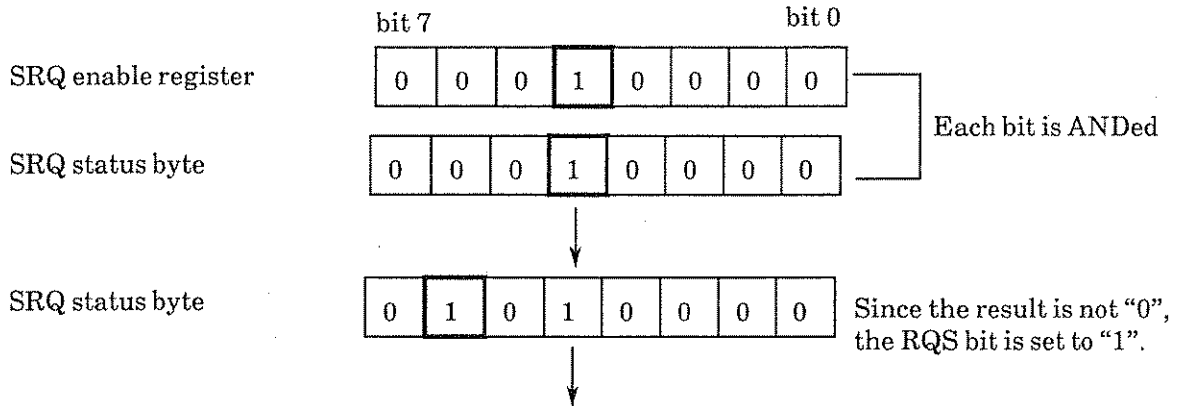
- weight 0 : Turns the service request send condition to OFF.
- 1: Unused bit
- 2: Sends a service request when the extended event status register is set.
- 4: Unused bit
- 8: Unused bit
- 16: Sends a service request when there is data to be output to GP-IB.
- 32: Sends a service request when the standard event status register is set.

12.2.2 SRQ status byte

Operation

Status byte is the information for the data that is to be output on the bus when serial polling is received from the controller.

When an event is generated, it is ANDed with the SRQ enable byte. If the result is not "0", the RQS bit (bit 6) is set to "1".



This data is output when serial polling.
This data is also output by a data request message.

Command

Control message :
Data request message : STB?
Output message : STBΔ#B***** (CR/LF)

(*: State at that time)

Configuration

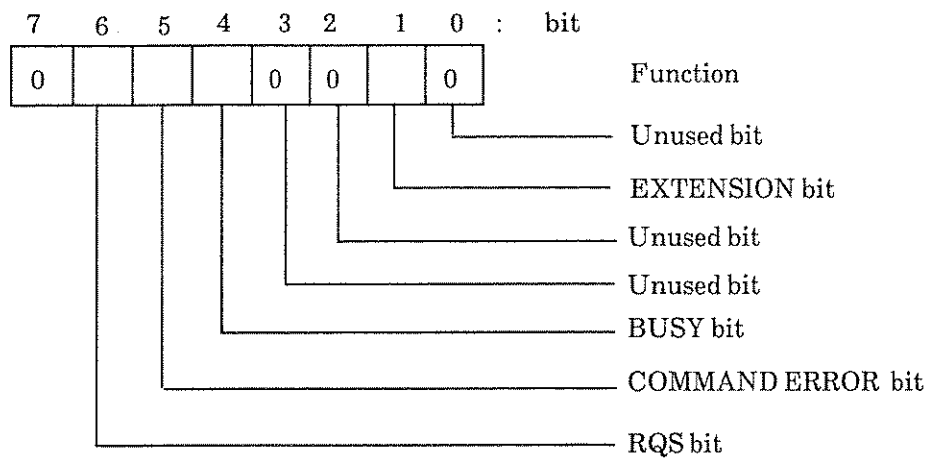


Table 12-1 GP-IB Status Byte

Name	Bit	Function
	7	Not used.
RQS	6	This bit indicates that a service request is issued. It is set when the bit set by SRQ command was set from 0 to 1, and is reset when there is no event requesting service or after the controller has read the status. It is also reset when a device clear or initialize command is received.
COMMAND ERROR	5	This bit indicates the error state. It is set when the condition is satisfied by standard event status byte and standard event status enable register conditions (See paragraph 12.2.3.). It is reset when the standard event status byte is read. It is also reset when a device clear or initialize command is received. When this bit is set after setting the COMMAND ERROR bit (weight 32) by SRQ command, SRQ is sent.
BUSY	4	This bit is set when there is data to be sent on the GP-IB. It is reset when data transfer is completed or when there is no data to be sent. It is also reset when a device clear or initialize command is received. When this bit is set after setting the BUSY bit (weight 16) by SRQ command, SRQ is sent.
	3	Not used.
	2	Not used.
EXTENSION	1	This bit indicates the MP1701B/MP1755A/MP1608A/MP1650A's unique status. It is set when the extended event status byte and extended event status enable register conditions are satisfied (See paragraph 12.2.5.). It is reset when the extended event status byte is read. It is also reset when a device clear or initialize command is received. When this bit is set after setting the EXTENSION bit (weight 2) by SRQ command, SRQ is sent.
	0	Not used.

12.2.3 Standard event status enable register

Operation

This register indicates the GP-IB command standard event status send condition.

The COMMAND ERROR bit ON/OFF of the SRQ status byte is carried out in bit units of the standard event status byte.

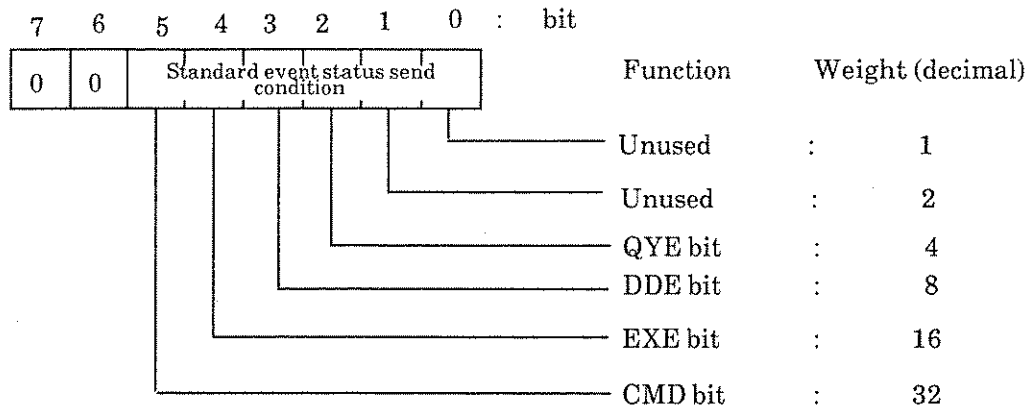
When an event of "1" bit at the standard event status register is generated, the COMMON ERROR bit is set to "1".

- When an ESE command is received, all the send conditions set previously are cleared.
- Multiple conditions can also be set.
- Immediately after the power is turned ON and immediately after a device clear or initialize command is received, all the ESE send conditions are cleared.

Command

Control message	:	ESE Δ 0 to ESE Δ 63
Data request message	:	ESE?
Output message	:	ESE $\Delta\Delta$ 0 (CR/LF) to ESE Δ 63 (CR/LF)

Configuration



Function:

Weight 0 : Sets the standard event status send condition to OFF.

- 1: Unused bit
- 2: Unused bit
- 4: Sends the standard event status send condition when a talker is designated event though the send buffer is empty or when the send buffer has overflowed.
- 8: Sends the standard event status send condition when the command received is outside the usage condition.
- 16: Sends the standard event status send condition at a numeric value error or when a HEX value is used in a command in which HEX cannot be used.
- 32: Sends the standard event status send condition at header error.

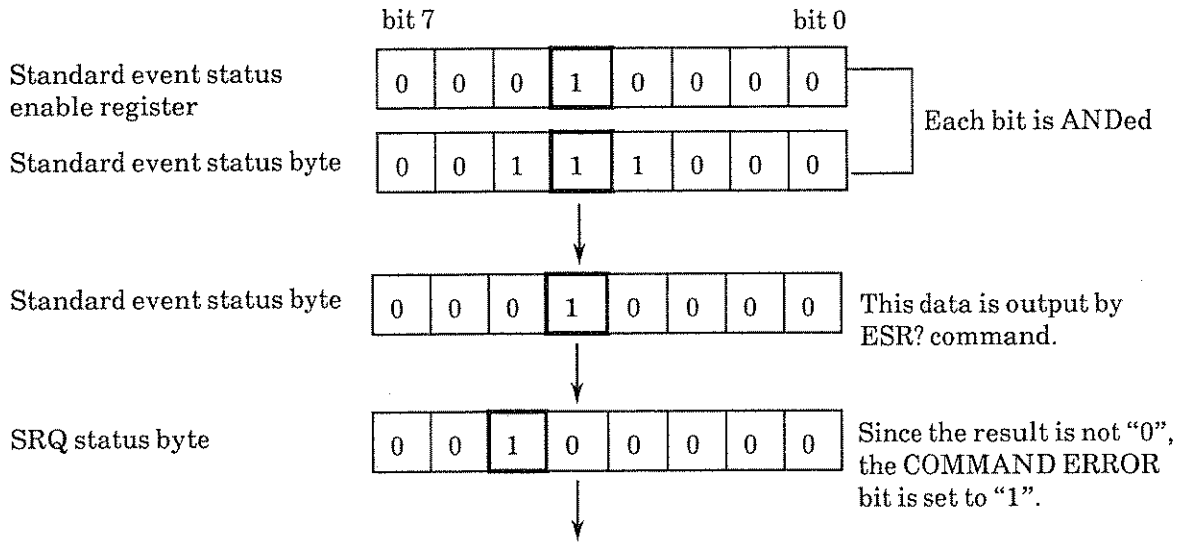
12.2.4 Standard event status byte

Operation

This byte is the information for the data which is to be output on the bus when a data request message is received.

When an event is generated, it is ANDed with the standard event status enable register. If the result is not "0", the COMMAND ERROR bit of the SRQ status byte is set to "1".

- Each bit of the standard event status byte is reset when data is read by data request message.
- This byte is not cleared even if a device clear, interface clear or initialize command is received.



This data is output when serial polling.
This data is also output by data request message.

Command

Control message :
Data request message : ESR?
Output message : ESRΔ#B***** (CR/LF)
(*: State at that time)

Configuration

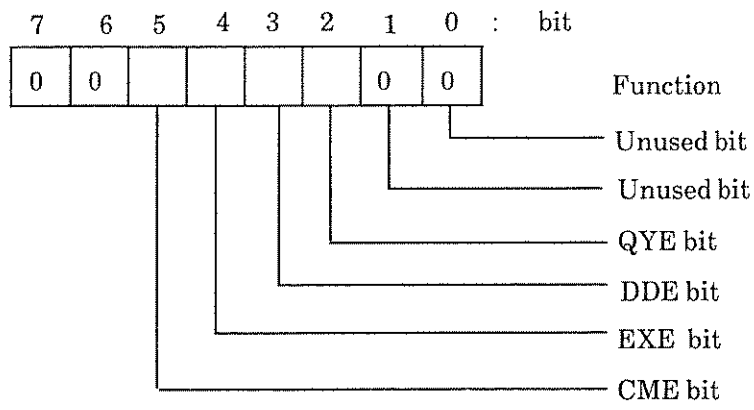


Table 12-2 Standard Event Status Byte

Name	Bit	Function
	7	Not used
	6	Not used
CME	5	<p>CME: Command error This bit is set when an undefined header is received. It is reset when the standard event status byte was read. It is not reset even if a device clear, interface clear, or initialize command is received. When this bit is set after setting the CME bit (weight 32) by the ESE command, the COMMAND ERROR bit of the status byte is set.</p>
EXE	4	<p>EXE: Execution error This bit is set when there is an error after the header (value range error or a HEX value used in a command in which HEX cannot be used). It is reset when the standard event status byte is read. It is not reset even if a device clear, interface clear, or initialize command is received. When this bit is set after setting the EXE bit (weight 16) by ESE command, the COMMAND ERROR bit of the status byte is set.</p>
DDE	3	<p>DDE: Device dependent error This bit is set when the received command is outside the usage condition. It is reset when the standard event status byte is read. It is not reset even if a device clear, interface clear, or initialize command is received. When this bit is set after setting the DDE bit (weight 8) by ESE command, the COMMAND ERROR bit of the status byte is set.</p>
QYE	2	<p>QYE: Query error This bit is set when the MP1701B/MP1755A/MP1608A/MP1650A is designated a talker even though the send buffer is empty or when the send buffer has overflowed. It is reset when the standard event status byte is read. It is not reset even if a device clear, interface clear, or initialize command is received. When this bit is set after setting the QYE bit (weight 4) by ESE command, the COMMAND ERROR bit of the status byte is set.</p>
	1	Not used
	0	Not used

12.2.5 Extended event status enable register

Operation

This register indicates the GP-IB command extended event status send condition.

It turns the SRQ status byte EXTENSION bit ON and OFF in bit units of extended event status byte.

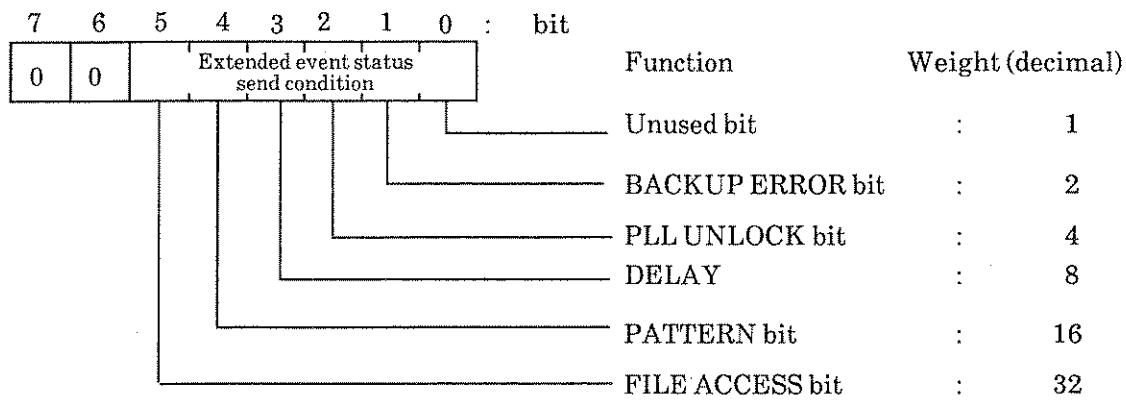
When an event of "1" bit at the extended event status register is generated, the EXTENSION bit is set to "1".

- When an EES command is received, all the send conditions set previously are cleared.
- Multiple conditions can also be set.
- All the EES send conditions are cleared immediately after the power is turned ON and immediately after a device clear or initialize command is received.

Command

Command message	:	EES Δ 0 to EES Δ 63
Data request message	:	EES?
Output message	:	EES $\Delta\Delta$ 0 (CR/LF) to EES Δ 63 (CR/LF)

Configuration



Function

Weight 0: Turns the extended event status send condition to OFF

- 1: Unused bit
- 2: Sends the extended event status send condition at backup error.
- 4: Sends the extended event status send condition when internal-clock phase lock loop circuit (PLL) is in unlock state.
- 8: Send the extended event status send condition when the servo subcircuit of the clock delay circuit is changed from the BUSY to READY state.
- 16: Sends the extended event status send condition at the end of programmable pattern setting.
- 32: Sends the extended event status send condition at the end of floppy disk access.

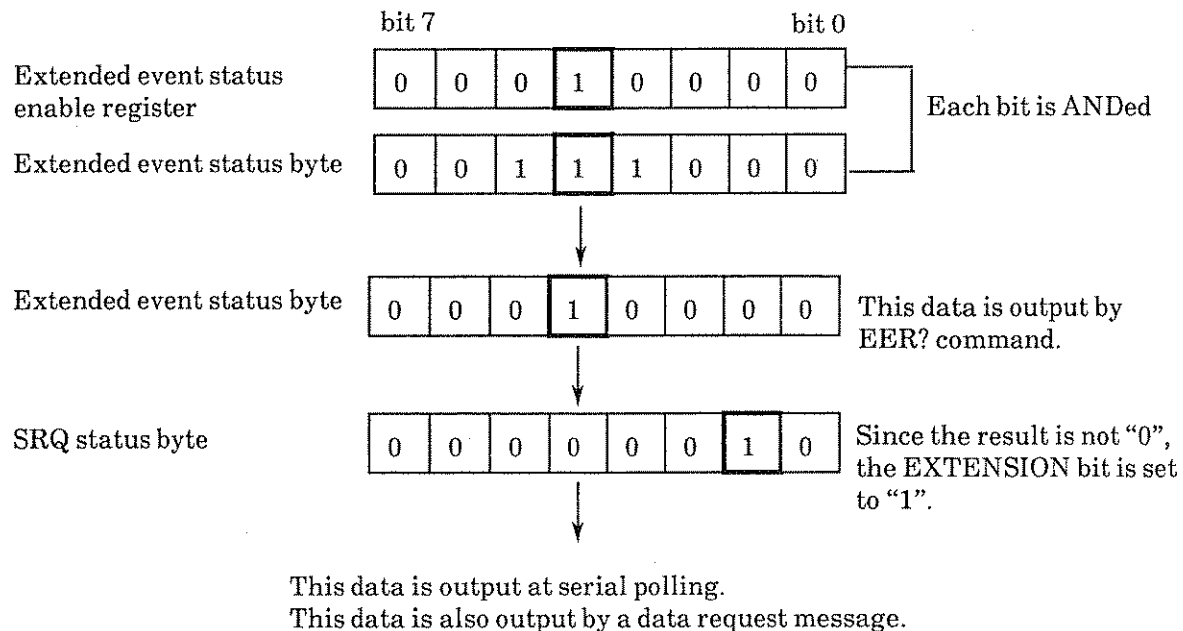
12.2.6 Standard event status byte

Operation

This byte is the information for the data that is sent on the bus when a data request message is received.

When an event is generated, it is ANDed with the extended event status enable register. If the result is not "0", the EXTENSION bit of the SRQ status byte is set to "1".

- Each bit of the extended event status byte is reset when data is read by data request message.
- This byte is not cleared even if a device clear, interface clear, or initialize command is received.



Command

Control message :
 Data request message : EER?
 Output message : EERΔ#B***** (CR/LF)
 (*: State at that time)

Configuration

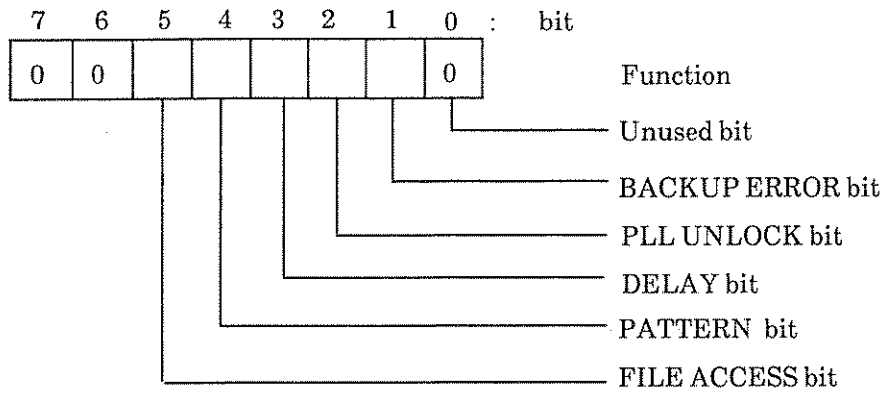


Table 12-3 Extended Event Status Byte

Name	Bit	Function
	7	Not used
	6	Not used
FILE ACCESS	5	This bit is set at the end of a floppy disk access. It is reset when the extended event status byte is read. It is not reset even if a device clear, interface clear, or initialize command is received. When this bit is set after setting the FILE ACCESS bit (weight 32) by EES command, the status byte EXTENSION bit is set.
PATTERN	4	This bit is set at the end of programmable pattern setting. It is reset when the extended event status byte is read. It is not reset even if a device clear, interface clear, or initialize command is received. When this bit is set after setting the PATTERN bit (weight 16) by EES command, the status byte EXTENSION bit is set.
DELAY	3	This bit is set when the servo subcircuit of the clock delay circuit is changed from the BUSY to the READY state. It is reset when the extended event status byte is read. However, it is not reset even if a device clear, interface clear, or initialize command is received. When this bit is set after setting the DELAY bit (weight 8) by EES command, the status byte EXTENSION bit is set.
PLL UNLOCK	2	This bit is set when the internal clock circuit PLL is in the UNLOCK state. It is reset when the extended event status byte is read. However, it is not reset even if a device clear, interface clear, or initialize command is received. When this bit is set after setting the PLL UNLOCK bit (weight 4) by EES command, the status byte EXTENSION bit is set.

Table 12-3 Extended Event Status Byte (Cont'd)

Name	Bit	Function
BACKUP ERROR	1	This bit is set when the contents of the backup RAM are checked at power ON and then an error is detected. It is reset when the extended event status byte is read. It is not reset even if a device clear, interface clear, or initialize command is received. When this bit is set after setting BACKUP ERROR bit (weight 2) is set by EES command, the status byte EXTENSION bit is set.
	0	Not used

SECTION 13

PATTERN DATA TRANSFER BY DMA

When up to 512k bits of programmable pattern data are transferred via GP-IB by BIT command, just as with conventional instruments (MP1601A/MP1604A), a large amount of time is required.

The MP1701B/MP1755A/MP1608A/MP1650A is equipped with a DMA transfer function to efficiently transfer this data pattern.

13.1 DMA

DMA is the abbreviation of Direct Memory Access and is a method of transferring a large volume of data at high speed (memory to memory transfer).

13.2 Commands for Number of Pattern Data Input Bytes and Number of Pattern Data Output Bytes

The number of pattern data input bytes (WRT command) and number of pattern data output bytes (RED command) GP-IB commands of the MP1701B/MP1755A/MP1608A/MP1650A are described below.

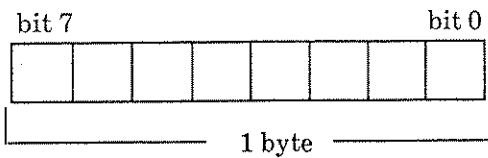
To transfer pattern data, the MP1701B/MP1755A/MP1608A/MP1650A must be given the following information by using the WRT and RED commands:

- Number of bytes of pattern data to be transferred
- The start address of the MP1701B/MP1755A/MP1608A/MP1650A internal RAM to store the transferred pattern data, or the start address of the MP1701B/MP1755A/MP1608A/MP1650A internal RAM to output the pattern data to be transferred.

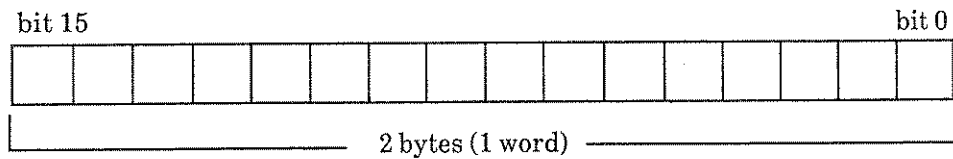
13.2.1 Number of bytes of pattern data to be transferred

A byte is an 8-bit data group.

The MP1701B/MP1755A/MP1608A/MP1650A has a 16-bit pattern data configuration. One word consists of two bytes.

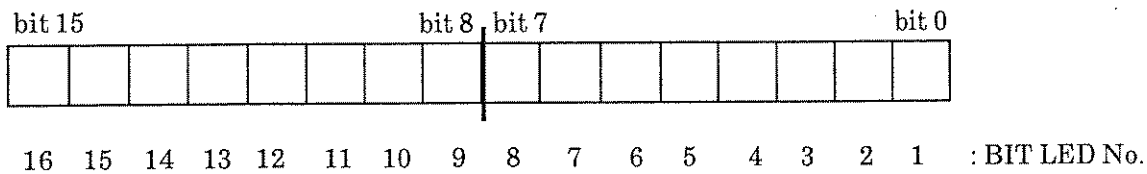


Value range: 0 to 255 (decimal)
0H to FFH (hexadecimal)



Value range: 0 to 65535 (decimal)
0H to FFFFH (hexadecimal)

The pattern data and bit correspondence is shown below.



The number of bytes of pattern data to be transferred equals the number of 8-bit data to be transferred.

Normally, 2 bytes (even number byte transfer) are transferred for one page displayed.

However, for odd number byte transfer, only the higher byte (bits 15 to 8) is set in the figure above.

13.2.2 Start address of the MP1701B/MP1755A/MP1608A/MP1650A internal RAM to store transferred pattern data and that to output pattern data to be transferred

The MP1701B/MP1755A/MP1608A/MP1650A internal RAM address range is from 0 to 32767. This range is common to WORD and DATA patterns.

The relationship between number of the page actually displayed and the corresponding address is shown below.

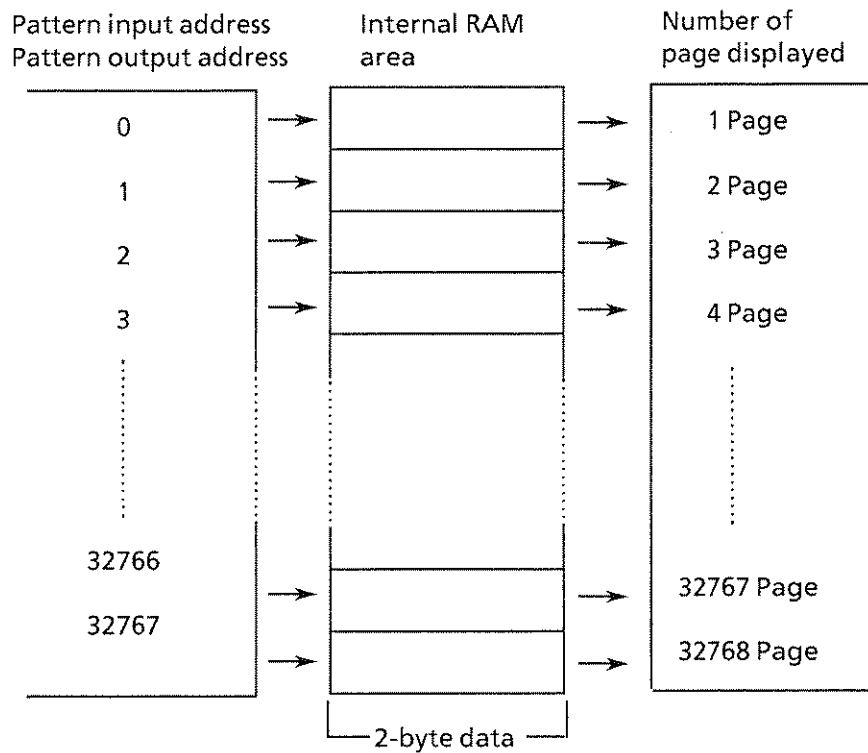


Fig. 13-1 Relationship between Number of Page and Address

13.3 Editing Program using DMA

Sample softwares using DMA function are shown in items (15) and (16) of paragraph 14.4 and 14.5. Learn the function and operation of each by referring to them.

The relationship between the pattern input/output start address and the number of the page actually set is described below.

For a 16-bit CPU, odd numbered addresses cannot be defined and, therefore, the following equation is established.

(Pattern input/output start address + 1) = number of the actual page

SECTION 14

SAMPLE PROGRAMS

Sample programs and their flowcharts for the MP1701B/MP1755A/MP1608A/MP1650A functions by GP-IB are described in this Section.

14.1 Sample Program List

Table 14-1 Sample Program List

Item No. of Paragraphs 14.4 and 14.5	Sample Program Control Function
(1)	Frequency setting
(2)	Frequency setting
(3)	Pattern setting
(4)	Output signal setting
(5)	Floppy disk file information read Floppy disk access status check → serial polling
(6)	Floppy disk file information read Floppy disk access status check → status byte register service request
(7)	Data save, resave, and recall
(8)	Standard event status byte check COMMAND ERROR bit check → serial polling
(9)	Standard event status byte check COMMAND ERROR bit check → status byte register service request
(10)	Internal timer setting
(11)	Power failure and recovery status check
(12)	Number of external pattern input channel setting
(13)	Error addition channel setting
(14)	Number of mark ratio AND bit shifts setting

Table 14-1 Sample Program List (Cont'd)

Item No. of Paragraphs 14.4 and 14.5	Sample Program Control Function
(15)	Pattern data transfer by DMA*
(16)	Pattern data transfer by DMA*

* The sample program of pattern data transfer by DMA uses HP9000 only as the controller.

14.2 Controller Used

Sample programs are made assuming that the controller will be the HP9000 Series Computer or IBM PC (or compatibles, equipped with the GP-IB interface card manufactured by National Instrument, INC. [N.I.]).

The programming languages are HP-BASIC and QUICK-BASIC (by Microsoft).

These programs are checked by the HP9000-200/300 Computer with HP-BASIC (V 5.12) and the Toshiba (Japan) J3100 Computer with QUICK-BASIC (V 3.00) on the GP-IB interface card (N.I.).

14.3 Preparation for Program Execution

Sample program execution preparations are shown below by controller used.

Controller	Program Execution Preparation
HP9000	<ul style="list-style-type: none"> • Set the MP1701B/MP1755A/MP1608A/MP1650A GP-IB address to "0". • Connect the MP1701B/MP1755A/MP1608A/MP1650A to the HP9000 with a GP-IB cable.
J3100	<ul style="list-style-type: none"> • Set the MP1701B/MP1755A/MP1608A/MP1650A GP-IB address to "0". • Setting IBCONF <ul style="list-style-type: none"> ① < Board Characteristics > <ul style="list-style-type: none"> Board: GPIB0 (Board is made "GPIB0".) Primary GPIB Address 10 Secondary GPIB Address NONE Timeout setting T30S EOS byte 0AH Terminal Read on EOS yes Set EOI with EOS on Write yes Type of compare on EOS 7-bit Set EOI w/last byte of Write yes GPIB-PC Model PC2A Board is System Controller yes Local Lockout on all devices no Desable Auto Serial Polling yes High-speed timing no Interrupt jumper setting 7 Base I/O Address 02E1H DMA channel NONE Internal clock Freq (in MHZ) 6 ② < Device Characteristics > <ul style="list-style-type: none"> Device: PPG (Device name is made "PPG".) Primary GPIB Address 0 Secondary GPIB Address NONE Timeout setting T30S EOS byte 0AH Terminal Read on EOS yes Set EOI with EOS on Write yes Type of compare on EOS 7-bit Set EOI w/last byte of Write yes ③ Device Map for Board GPIB0 <ul style="list-style-type: none"> This connects the device made "PPG" to Board: GPIB0 of ①. • Connect the MP1701B/MP1755A/MP1608A/MP1650A to the J3100 with a GP-IB cable.

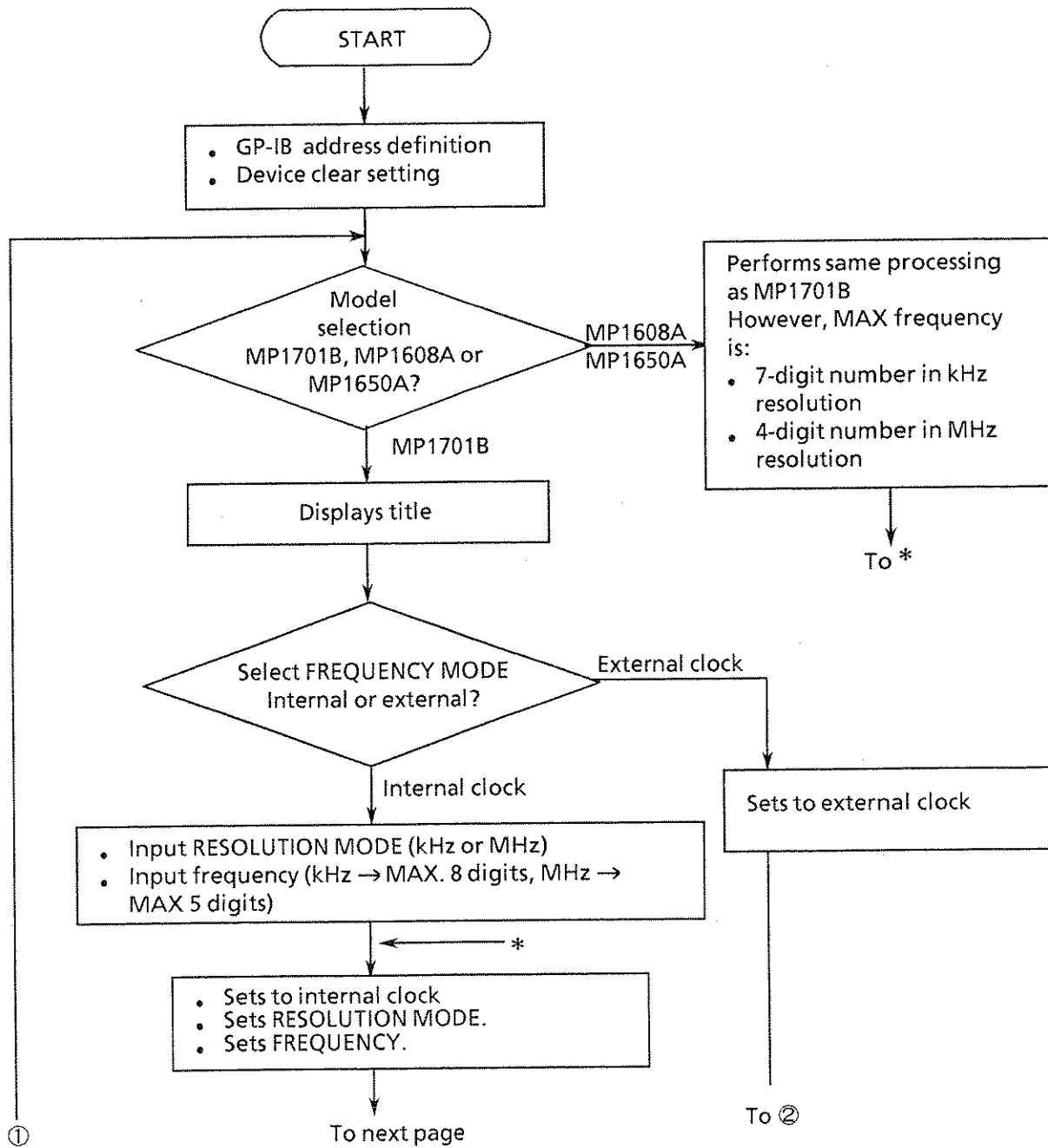
Sample programs using the HP9000 as the controller are shown in paragraph 14.4. Sample programs using the J3100 as the controller are shown in paragraph 14.5.

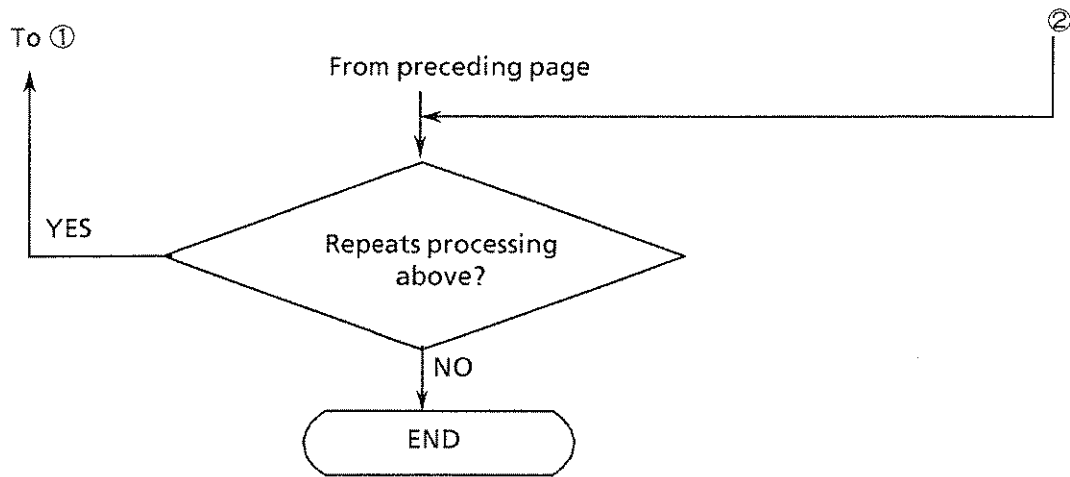
14.4 Sample Programs Using HP9000 as the Controller

(1) Frequency setting

This program performs clock frequency control.

It selects internal clock or external clock, and inputs the frequency resolution and clock frequency.





PROGRAM LISTING

```

10 !*****
20 !*
30 !* MP1701B/MP1608A/MP1650A FREQUENCY SAMPLE SOFT_1 *
40 !*
50 !* FRQ_1 *
60 !*****
70 !
80 !-----!
90 ! MAIN ROUTINE !
100 !-----!
110 !
120 LET Add=700 ! Set Device Address
130 CLEAR Add ! Device Clear
140 !
150 LOOP
160 !
170 INPUT "MODE SELECT [MP1701B=0, MP1608A=1, MP1650A=2] ?",Mode#
180 !
190 SELECT Mode#
200 CASE "0"
210 GOSUB P10g
220 CASE "1"
230 Ppg#="MP1608A"
240 GOSUB P5g3g
250 CASE "2"
260 Ppg#="MP1650A"
270 GOSUB P5g3g
280 END SELECT
290 !
300 INPUT "NEXT DATA SET [ YES=0 , NO=1 ] ?",Loop#
310 !
320 EXIT IF Loop#="1"
330 !
340 END LOOP
350 !
360 STOP
370 !
380 !-----!
390 ! SUB ROUTINE !
400 !-----!
410 !
420 P10g:!------- MP1701B DATA INPUT
430 !
440 PRINT "*** MP1701B FREQUENCY SAMPLE SOFT ***"
450 PRINT ""
460 INPUT "FREQUENCY MODE SELECT[ EXTERNAL=0, INTERNAL=1 ]?",Clk#
470 !
480 IF Clk#="1" THEN
490 INPUT "FREQUENCY RESOLUTION [ kHz=0 , MHz=1 ] ?",Res#
500 !
510 IF Res#="1" THEN
520 INPUT "FREQUENCY DATA [ MHz=5 figures,MAX ] ?",Frq#
530 ELSE
540 INPUT "FREQUENCY DATA [ kHz=8 figures,MAX ] ?",Frq#
550 END IF
560 !
570 GOSUB D_set
580 !
590 ELSE
600 OUTPUT Add;"CLK 0" ! Clock mode : External
610 !

```

```

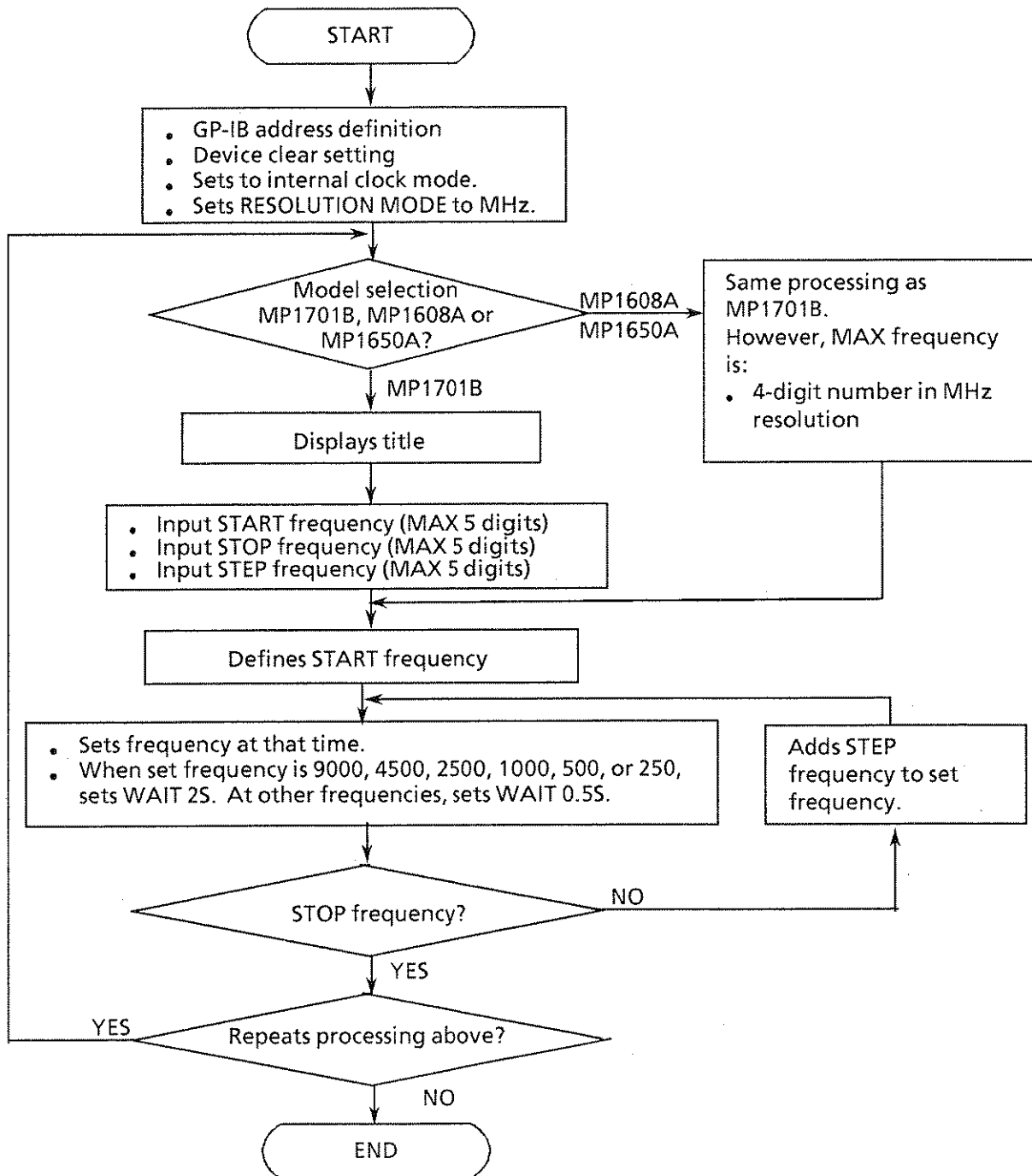
620 END IF
630 !
640 RETURN
650 !
660 !
670 P5g3g: !----- MP1608A/MP1650A DATA INPUT
680 !
690 PRINT "*** "&Ppg#&" FREQUENCY SAMPLE SOFT ***"
700 PRINT ""
710 INPUT "FREQUENCY MODE SELECT [ EXTERNAL=0, INTERNAL=1 ]?",Clk#
720 !
730 IF Clk#="1" THEN
740 INPUT "FREQUENCY RESOLUTION [ kHz=0 , MHz=1 ] ?",Res#
750 !
760 IF Res#="1" THEN
770 INPUT "FREQUENCY DATA [ MHz=4 figures,MAX ] ?",Frq#
780 ELSE
790 INPUT "FREQUENCY DATA [ kHz=7 figures,MAX ] ?",Frq#
800 END IF
810 !
820 GOSUB D_set
830 !
840 ELSE
850 OUTPUT Add;"CLK 0" ! Clock mode : External
860 !
870 END IF
880 !
890 RETURN
900 !
910 !
920 D_set: !----- MP1701B/MP1608A/MP1650A DATA OUTPUT
930 !
940 OUTPUT Add;"CLK 1" ! Clock mode : Internal
950 OUTPUT Add;"RES "&Res# ! Send Resolution mode
960 OUTPUT Add;"FRQ "&Frq# ! Send Frequency data
970 !
980 RETURN
990 !
1000 END

```


(2) Frequency setting

This program increases the internal clock frequency from a start frequency to a stop frequency in a certain step width.

The frequency switching speed is every 0.5 second, but the set state is held for 2 seconds at 250, 500, 1000, 2500, 4500, and 9000 MHz.



PROGRAM LISTING

```

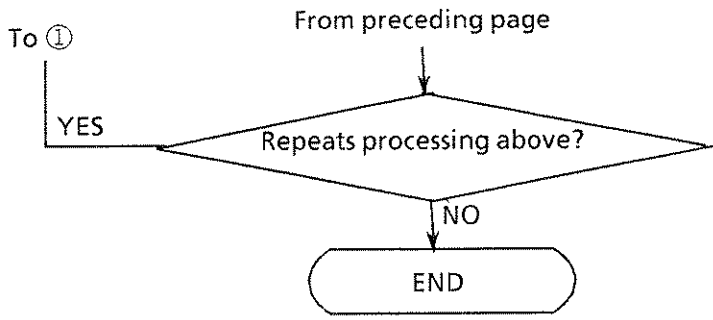
10  !*****
20  !*
30  !*      MP1701B/MP1608A/MP1650A      FREQUENCY SAMPLE SOFT_2      *
40  !*
50  !*
60  !*****
70  !
80  !
90  !-----!
100 !                               MAIN ROUTINE                               !
110 !-----!
120 !
130 LET Add=700                      ! Set Device Address
140 CLEAR Add                          ! Device Clear
150 OUTPUT Add;"CLK 1;RES 1"          ! Internal mode , Resolution : MHz
160 !
170 LOOP
180 !
190 INPUT "MODE SELECT [ MP1701B=0, MP1608A=1, MP1650A =2 ] ?",Mode#
200 !
210 SELECT Mode#
220 CASE "0"
230 GOSUB P10g
240 CASE "1"
250 Ppg#="MP1608A"
260 GOSUB P5g3g
270 CASE "2"
280 Ppg#="MP1650A"
290 GOSUB P5g3g
300 END SELECT
310 !
320 GOSUB D_set
330 !
340 INPUT "NEXT DATA SET ? [ YES=0 , NO=1 ]",Loop#
350 !
360 EXIT IF Loop#="1"
370 !
380 END LOOP
390 !
400 STOP
410 !
420 !
430 !-----!
440 !                               SUB ROUTINE                               !
450 !-----!
460 !
470 P10g:!------- MP1701B DATA INPUT
480 !
490 PRINT " ***** MP1701B FREQUENCY SAMPLE SOFT *****"
500 PRINT ""
510 !
520 INPUT " START FREQUENCY DATA [ MHz=5 figures,MAX ] ?",Startf#
530 INPUT " STOP FREQUENCY DATA [ MHz=5 figures,MAX ] ?",Stopf#
540 INPUT " STEP FREQUENCY DATA [ MHz=5 figures,MAX ] ?",Stepf#
550 !
560 RETURN
570 !
580 !

```

```

590 P5g3g: ! ----- MP1608A /MP1650A DATA INPUT
600 !
610 PRINT " ***** "&Ppg#&" FREQUENCY SAMPLE SOFT *****"
620 PRINT ""
630 !
640 INPUT " START FREQUENCY DATA [ MHz=4 figures,MAX ] ?",Startf#
650 INPUT " STOP FREQUENCY DATA [ MHz=4 figures,MAX ] ?",Stopf#
660 INPUT " STEP FREQUENCY DATA [ MHz=4 figures,MAX ] ?",Stepf#
670 !
680 RETURN
690 !
700 !
710 D_set: ! ----- MP1701B/MP1608A/MP1650A DATA OUTPUT
720 !
730 FOR I=VAL(Startf#) TO VAL(Stopf#) STEP VAL(Stepf#)
740 !
750 OUTPUT Add;"FRQ "&VAL$(I) ! Send Frequency Data
760 !
770 IF I=9000 OR I=4500 OR I=2500 OR I=1000 OR I=500 OR I=250 THEN
780 WAIT 2
790 !
800 ELSE
810 WAIT .5
820 !
830 END IF
840 !
850 NEXT I
860 !
870 RETURN
880 !
890 !
900 END

```

PROGRAM LISTING

```

10  !*****
20  !*
30  !*      MF1701B/MF1608A/MF1650A  PATTERN SAMPLE SOFT      *
40  !*
50  !*
60  !*****
70  !
80  !
90  !-----!
100 !                      MAIN ROUTINE                      !
110 !-----!
120 !
130 LET Add=700          ! Set Device Address
140 CLEAR Add           ! Device Clear
150 OUTPUT Add;"EAD 0"  ! Error Addition mode : OFF
160 !
170 PRINT "*** MF1701B/MF1608A/MF1650A  PATTERN SAMPLE SOFT ***"
180 PRINT ""
190 !
200 LOOP
210 !
220   GOSUB Pattern
230 !
240   INPUT " NEXT DATA SET [ YES=0 , NO=1 ] ?",Loop#
250 !
260   EXIT IF Loop#="1"
270 !
280 END LOOP
290 !
300 !
310 STOP
320 !
330 !-----!
340 !                      SUB ROUTINE                      !
350 !-----!
360 !
370 Pattern:!------- SET Logic , Pattern mode
380 !
390 LOOP
400 !
410   INPUT " LOGIC MODE [ POSITIVE=0 , NEGATIVE=1 ] ?",Lgc#
420   OUTPUT Add;"LGC "%Lgc#
430 !
440   PRINT "PATTERN MODE [ WORD=0, DATA=1, PN7=2, PN9=3, PN11=5, "
450   PRINT "          PN15=6, PN20=7, PN23=8, PN31=9 ] "
460   PRINT ""
470   INPUT "PATTERN MODE [ 0 OR 1 OR 2 OR 3 OR 5 OR 6 OR 7 OR 8 OR 9 ]?",Ptn#
480 !
490   EXIT IF Ptn#<>"4"
500 !
510 END LOOP
520 !
530 OUTPUT Add;"PTN "%Ptn#
540 !
550 !
560 IF Ptn#="0" OR Ptn#="1" THEN
570 !
580   GOSUB Prog_mode
590 !
600 ELSE
610 !

```

```

620     INPUT "MARK RATIO ( 0/8:8/8=0, 1/8:7/8=1, 1/4:3/4=2, 1/2:N1/2=3 ?",Mrk#
630     OUTPUT Add;"MRK "&Mrk#
640     !
650     END IF
660     !
670     RETURN
680     !
690     !
700 Prog_mode: !----- SET  PROG (Word),(Data)
710     !
720     IF Ptn#="1" THEN
730         !
740         PRINT "*** PATTERN MODE PROG (DATA) ***"
750         PRINT ""
760         !
770         INPUT " DATA LENGTH DATA ?",Dln#
780         OUTPUT Add;"DLN "&Dln#
790         !
800         INPUT " PAGE DATA ?",Pag#
810         OUTPUT Add;"PAG "&Pag#
820         !
830         INPUT " BIT PATTERN SET DATA  BIT16-->BIT1 [0/1] ?",Bit#
840         GOSUB Btoh
850         OUTPUT Add;"BIT #H"&B#
860         !
870     ELSE
880         !
890         PRINT "*** PATTERN MODE PROG (WORD) ***"
900         PRINT ""
910         !
920         INPUT " NUMBER OF WORD DATA ?",Wnb#
930         OUTPUT Add;"WNB "&Wnb#
940         !
950         INPUT " WORD LENGTH DATA ?",Wln#
960         OUTPUT Add;"WLN "&Wln#
970         !
980         INPUT " PAGE DATA ?",Pag#
990         OUTPUT Add;"PAG "&Pag#
1000        !
1010       INPUT " BIT PATTERN SET DATA  BIT16-->BIT1 [0/1] ?",Bit#
1020       GOSUB Btoh
1030       OUTPUT Add;"BIT #H"&B#
1040       !
1050     END IF
1060     !
1070     RETURN
1080     !
1090     !
1100 Btoh: !----- BIN TO HEX
1110     !
1120     A=IVAL(Bit#,2)           ! BIN --> DIG
1130     B#=IVAL#(A,16)         ! DIG --> HEX
1140     !
1150     RETURN
1160     !
1170     !
1180     END

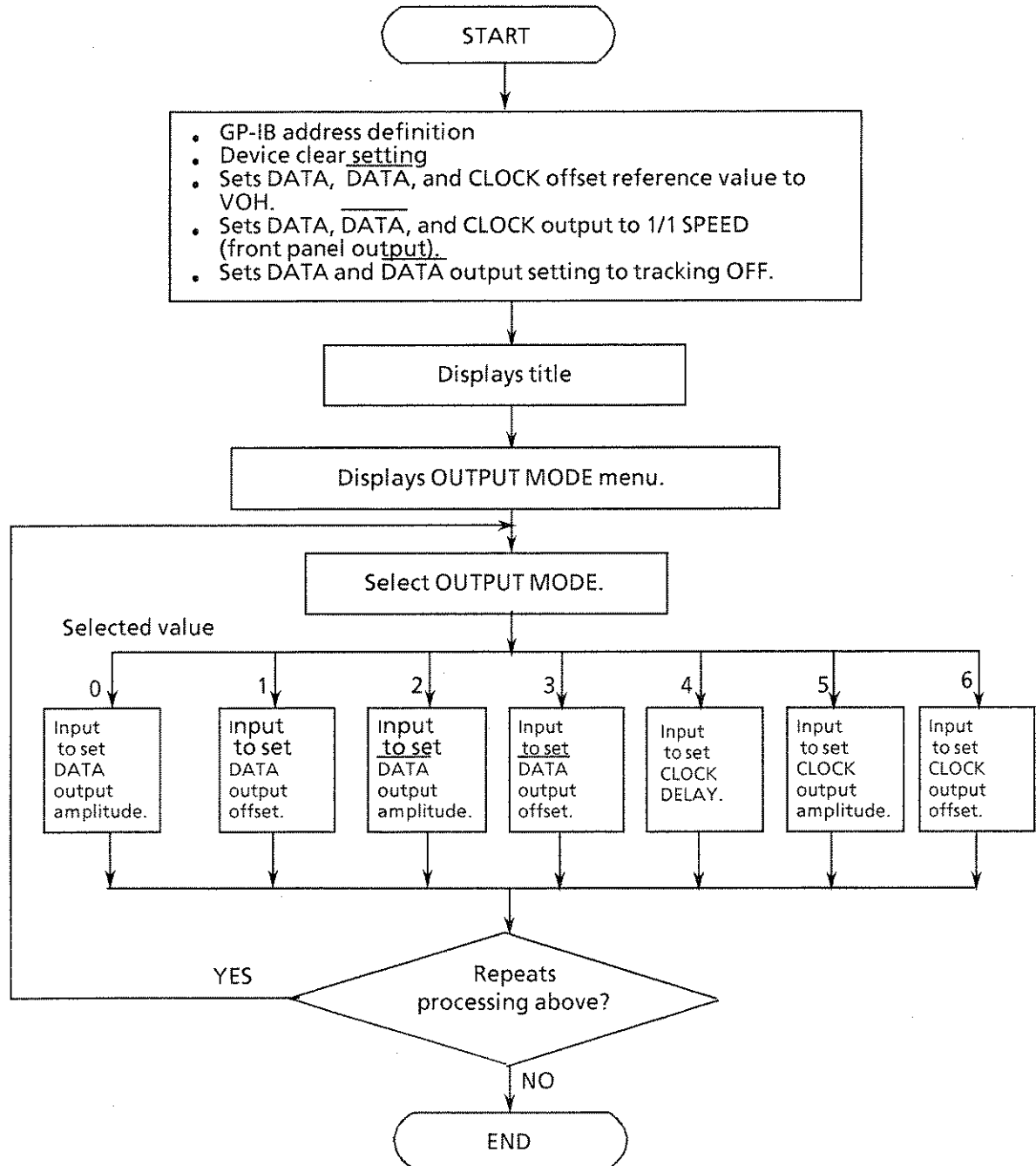
```

(4) Output signal setting

This program performs output signal control.

Select to set the necessary DATA, $\overline{\text{DATA}}$, and CLOCK output amplitudes and offsets.

The MP1701B/MP1755A/MP1608A/MP1650A is set to the offset reference value: VOH, 1/1SPEED (front panel output), and DATA/ $\overline{\text{DATA}}$ tracking: OFF state.



PROGRAM LISTING

```

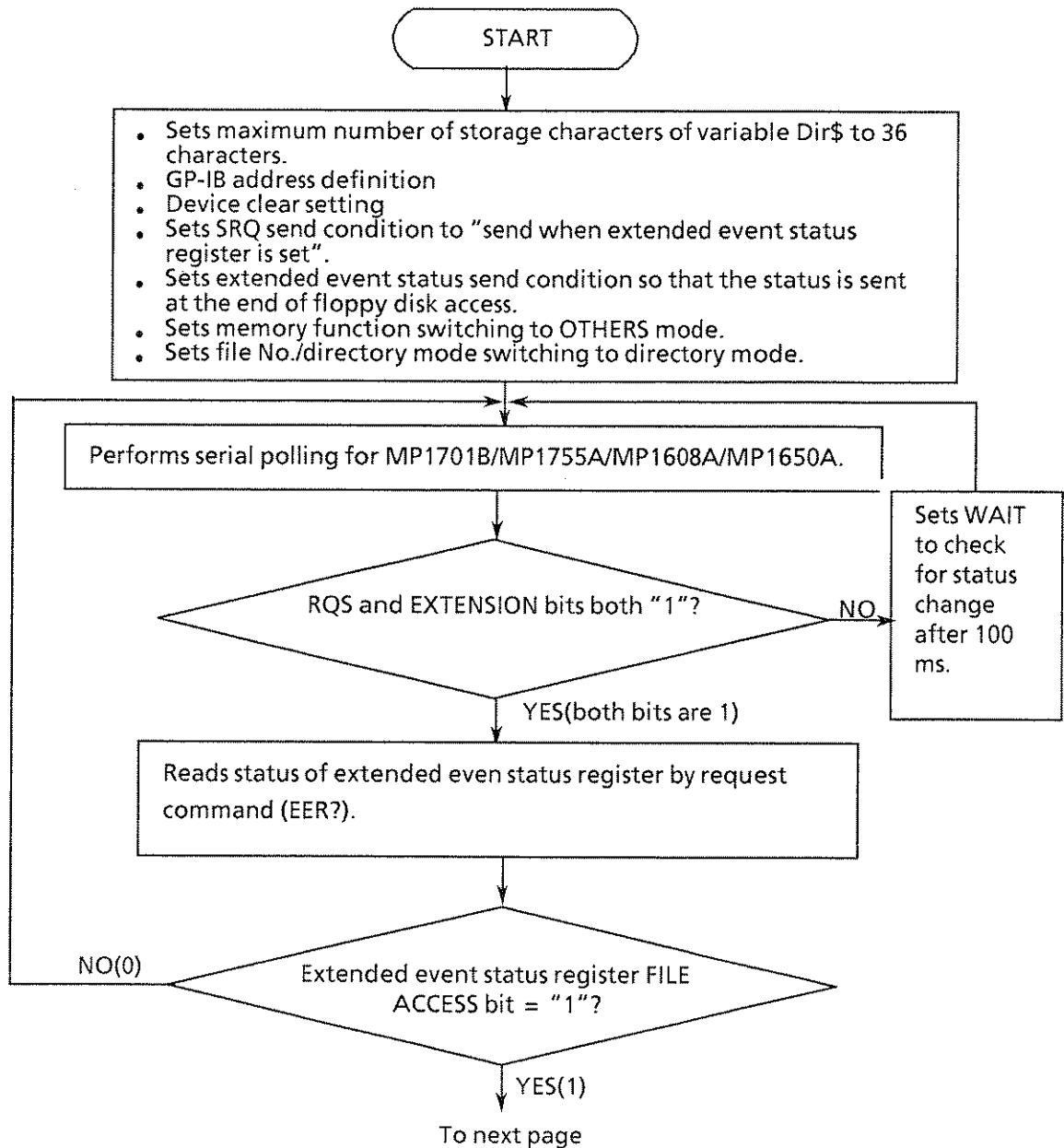
10  !*****
20  !*
30  !*      MP1701B/MP1608A/MP1650A  OUTPUT SAMPLE SOFT      *
40  !*      --- 1/1(Speed),Voh mode ---                      *
50  !*
60  !*****
70  !
80  LET Add=700
90  CLEAR Add
100 OUTPUT Add;"DFS 0;SFD 0;TRK 0" !SET Voh mode,1/1(Speed),Tracking off
110 !
120 PRINT "      ***** MP1701B/MP1608A/MP1650A  OUTPUT SAMPLE SOFT *****"
130 PRINT "      ----- 1/1(Speed) , Voh mode ----- "
140 PRINT ""
150 PRINT "OUTPUT MODE [DATA AMPLITUDE=0, DATA OFFSET=1, "
160 PRINT "      AMPLITUDE=2, NDATA OFFSET=3, CLOCK DELAY=4"
170 PRINT "      CLOCK AMPLITUDE=5, CLOCK OFFSET=6 ]"
180 PRINT
190 !
200 LOOP
210 !
220 INPUT "OUTPUT MODE SELECT [ 0 OR 1 OR 2 OR 3 OR 4 OR 5 OR 6 ]",Out#
230 !
240 SELECT Out#
250 !
260 CASE "0"
270     INPUT "* DATA AMPLITUDE [ +0.50 ~ 2.00 V ] STEP 0.01 ",Dap#
280     OUTPUT Add;"DAF "&Dap#
290     !
300 CASE "1"
310     INPUT "* DATA OFFSET [ -2.000 ~ 2.000 V ] STEP 0.005 ",Dos#
320     OUTPUT Add;"DOS "&Dos#
330     !
340 CASE "2"
350     INPUT "* NDATA AMPLITUDE [ +0.50 ~ 2.00 V ] STEP 0.01 ",Nap#
360     OUTPUT Add;"NAP "&Nap#
370     !
380 CASE "3"
390     INPUT "* NDATA OFFSET [ -2.000 ~ 2.000 V ] STEP 0.005 ",Nos#
400     OUTPUT Add;"NOS "&Nos#
410     !
420 CASE "4"
430     PRINT "<<<<  CLOCK DELAY  >>>>"
440     PRINT "MP1701B/MP1608A --> -500 ~ 500 ps    1ps STEP"
450     PRINT "MP1650A --> -1000 ~ 1000 ps    2ps STEP"
460     INPUT "* CLOCK DELAY ",Cd1#
470     OUTPUT Add;"CDL "&Cd1#
480     !
490 CASE "5"
500     INPUT "* CLOCK AMPLITUDE [ +0.50 ~ 2.00 V ] STEP 0.01 ",Cap#
510     OUTPUT Add;"CAP "&Cap#
520     !
530 CASE "6"
540     INPUT "* CLOCK OFFSET [ -2.000 ~ 2.000 V ] STEP 0.005",Cos#
550     OUTPUT Add;"COS "&Cos#
560     !
570 END SELECT
580 !
590 INPUT " NEXT DATA SET [ YES=0 , NO=1 ] ",Loop#
600 !
610 EXIT IF Loop#="1"
620 END LOOP
630 !
640 END

```

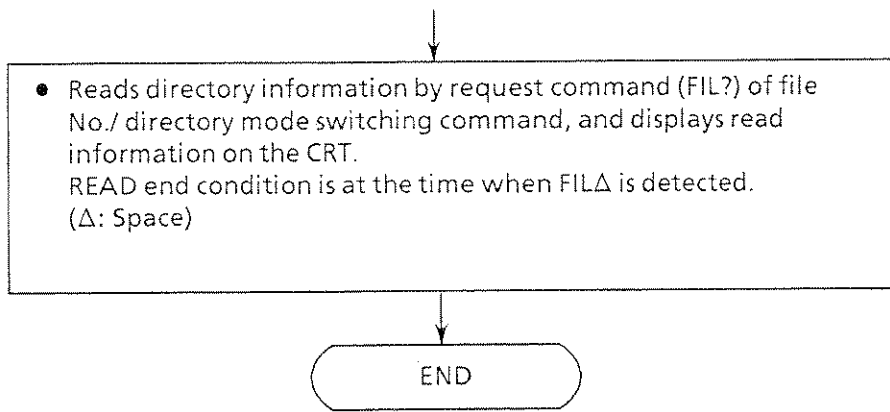
(5) Floppy disk file information read
(Floppy disk access status check → serial polling)

This program checks the directory information of the files stored on the floppy disk, and displays it on the CRT. It also shows an example of actual output.

Floppy disk access status check is performed by serial polling and by data request command that checks the extended event status information.



From preceding page



EXECUTED RESULT

T02	,PTN,	104,89-09-02,16:22
T04	,PTN,	104,89-09-27,15:53
T07	,OTH,	108,89-10-06,09:56
T21	,PTN,	104,89-09-27,15:53
T23	,PTN,	108,89-10-06,09:55
T24	,OTH,	108,89-09-27,15:54
T38	,PTN,	65640,89-09-29,15:35
T99	,PTN,	65640,89-10-06,09:58
FIL 1		

PROGRAM LISTING

```

10  !*****
20  !*
30  !* MP1701B/MP1608A/MP1650A  FILE DIRECTORY READ  SAMPLE SOFT_1  *
40  !*
50  !*
60  !*****
70  !
80  !-----!
90  !                               MAIN ROUTINE                               !
100 !-----!
110 !
120 DIM Dir#[36]
130 LET Add=700           ! Set Device Address
140 CLEAR Add            ! Device Clear
150 !
160 OUTPUT Add;"SRQ 2"   ! SRQ : Extension bit
170 OUTPUT Add;"EES 32" ! EES : Floppy Access End
180 OUTPUT Add;"MEM 1"  ! Memory mode : OTHERS
190 OUTPUT Add;"FIL 1"  ! FIL : Directory Mode
200 !
210 GOSUB Spoll_eer
220 GOSUB File_dir
230 !
240 STOP
250 !
260 !-----!
270 !                               SUB ROUTINE                               !
280 !-----!
290 !
300 Spoll_eer:!------- Check Status Byte
310 !
320 LOOP
330 !
340   LOOP
350   !
360     A=SPOLL(Add)      ! Send Serial Poll
370     !
380     EXIT IF BIT(A,6)=1 AND BIT(A,1)=1 ! RQS bit,Extension bit = 1
390     WAIT .1
400     !
410   END LOOP
420   !
430   OUTPUT Add;"EER?"  ! REQUEST Extension Event Register ?
440   ENTER Add;Eer#     ! READ   Extension Event Register
450   !
460   EXIT IF Eer#[9,9]="1" ! File Access bit = 1
470   !
480 END LOOP
490 !
500 RETURN
510 !
520 !

```

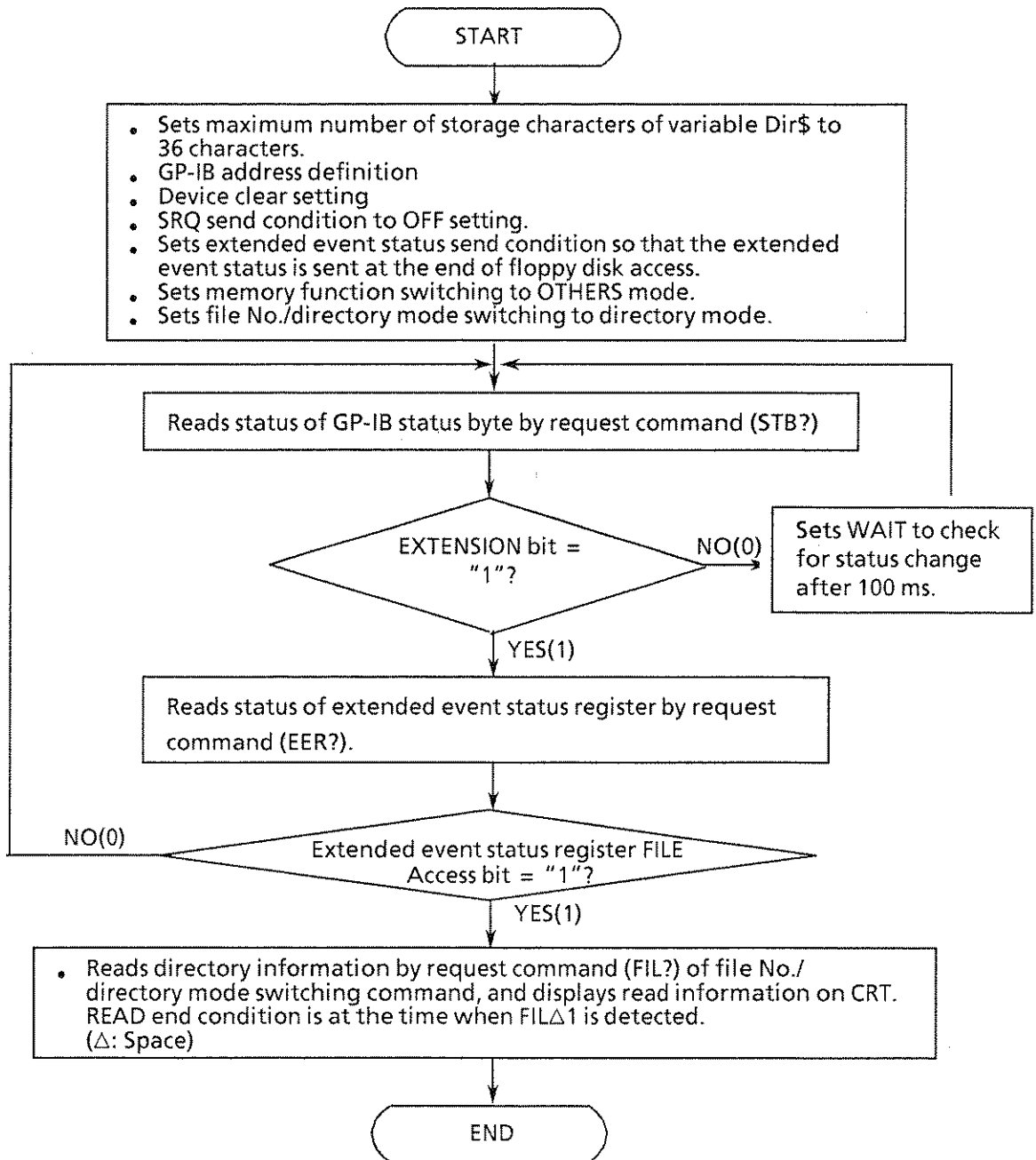
```
530 File_dir: ! ----- Read File Directory
540 !
550 OUTPUT Add;"FIL?" ! REQUEST Directory ?
560 !
570 LOOP
580 !
590 ENTER Add;Dir# ! READ Directory
600 PRINT Dir#
610 !
620 EXIT IF Dir#="FIL 1"
630 !
640 END LOOP
650 !
660 RETURN
670 !
680 !
690 END
```

(6) Floppy disk file information read

(Floppy disk access status check → status byte register service request)

This program checks the directory information of the files stored on floppy disk, and displays it on the CRT. It also shows an actual output example.

Floppy disk access status check is performed by data request command that checks the status of the GP-IB status byte and by data request command that checks the extended event status information.



EXECUTED RESULT

T02	,PTN,	104,89-09-02,16:22
T04	,PTN,	104,89-09-27,15:53
T07	,OTH,	108,89-10-06,09:56
T21	,PTN,	104,89-09-27,15:53
T23	,PTN,	108,89-10-06,09:55
T24	,OTH,	108,89-09-27,15:54
T38	,PTN,	65640,89-09-29,15:35
T99	,PTN,	65640,89-10-06,09:58
FIL 1		

PROGRAM LISTING

```

10  !*****
20  !*
30  !* MP1701B/MP1608A/MP1650A  FILE DIRECTORY READ  SAMPLE SOFT_2 *
40  !*
50  !*
60  !*****
70  !
80  !-----!
90  !                               MAIN ROUTINE                               !
100 !-----!
110 !
120 DIM Dir#[36]
130 LET Add=700                ! Set Device Address
140 CLEAR Add                  ! Device Clear
150 !
160 OUTPUT Add;"SRQ 0"        ! SRQ : OFF
170 OUTPUT Add;"EES 32"      ! EES : Floppy Access End
180 OUTPUT Add;"MEM 1"      ! Memory mode : OTHERS
190 OUTPUT Add;"FIL 1"      ! FIL : Directory Mode
200 !
210 GOSUB Stb_reg
220 GOSUB File_dir
230 !
240 STOP
250 !
260 !
270 !-----!
280 !                               SUB ROUTINE                               !
290 !-----!
300 !
310 Stb_reg:!------- Check Status Byte
320 !
330 LOOP
340 !
350 LOOP
360 !
370     OUTPUT Add;"STB?"      ! REQUEST Status Byte Register ?
380     ENTER Add;Stb#        ! READ   Status Byte Register
390     !
400     EXIT IF Stb#[13,13]="1" ! Extention bit = 1
410     WAIT .1
420     !
430     END LOOP
440     !
450     OUTPUT Add;"EER?"      ! REQUEST Extension Event Register ?
460     ENTER Add;Eer#        ! READ   Extension Event Register
470     !
480     EXIT IF Eer#[9,9]="1"  ! File Access bit = 1
490     !
500     END LOOP
510     !
520     RETURN
530     !
540     !

```

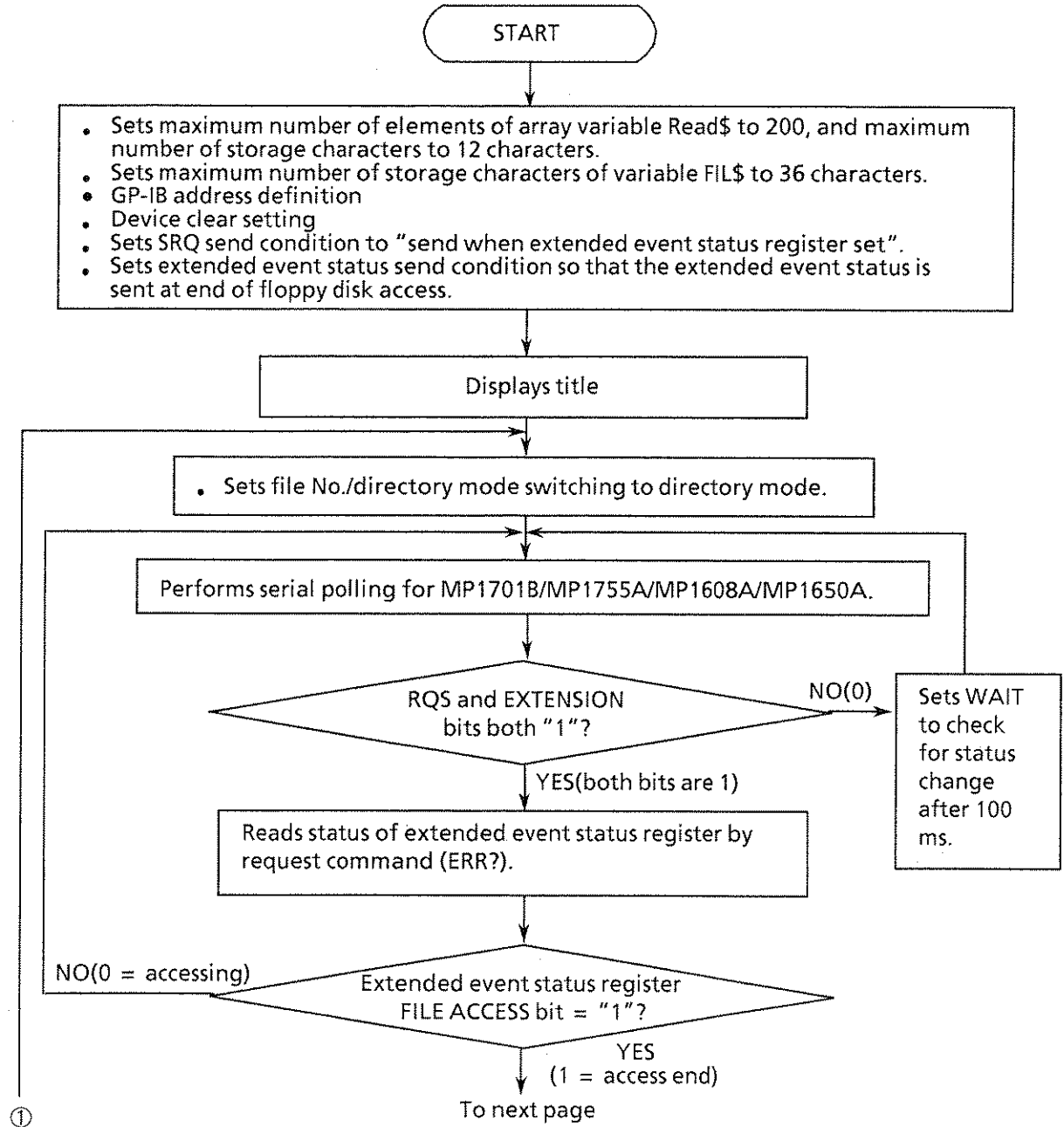


```
550 File_dir: ! ----- Read File Directory
560 !
570 OUTPUT Add;"FIL?" ! REQUEST Directory
580 !
590 LOOP
600 !
610 ENTER Add;Dir# ! READ Directory
620 PRINT Dir#
630 !
640 EXIT IF Dir#="FIL 1"
650 !
660 END LOOP
670 !
680 RETURN
690 !
700 END
```

(7) Data save, resave, and recall

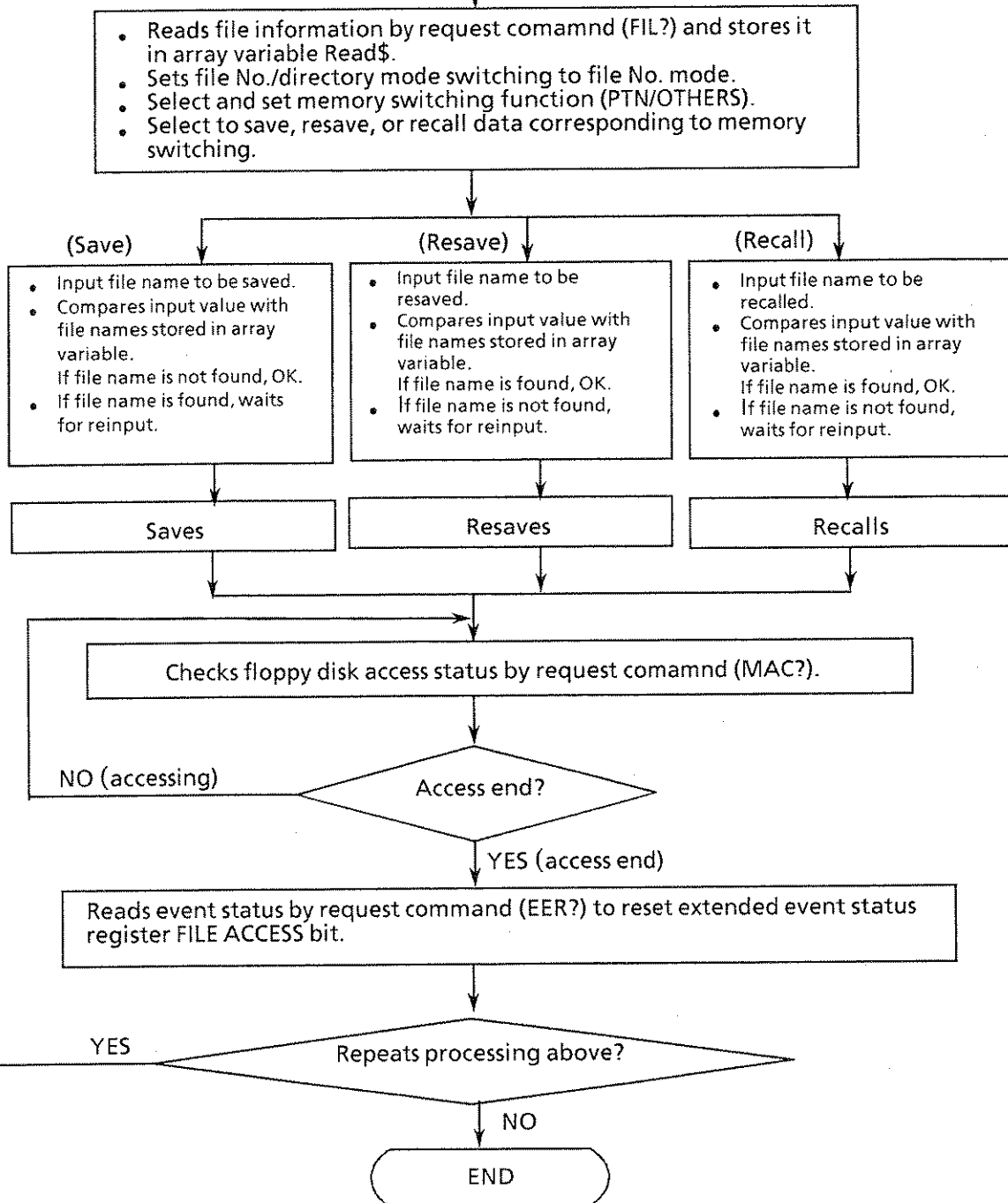
This program checks the directory information of the files stored on floppy disk, saves or resaves the statuses set at the MP1701B/MP1755A/MP1608A/MP1650A corresponding to the memory switching function (PTN/OTHERS), and recalls the file stored on floppy disk.

The access status of the floppy disk to obtain the directory information is checked by serial polling and by data request command to check the extended event status. At save, resave, and recall, the floppy disk access status is checked by request command MAC?



To ①

From preceding page



PROGRAM LISTING

```

10  !*****
20  !*
30  !*      MP1701B/MP1608A/MP1650A  MEMORY SAMPLE SOFT      *
40  !*
50  !*
60  !*****
70  !
80  !
90  !////////////////////
100 !/
110 !/
120 !/
130 !////////////////////
140 !
150 DIM Read$(199)[12]
160 DIM Fil$(36)
170 LET Add=700          ! Set Device Address
180 CLEAR Add           ! Device Clear
190 OUTPUT Add;"SRQ 2"  ! SRQ : EXTENSION bit
200 OUTPUT Add;"EES 32" ! Floppy Access End
210 !
220 PRINT " ** MP1701B/MP1608A/MP1650A  MEMORY SAMPLE SOFT **"
230 PRINT
240 !
250 LOOP
260 !
270     GOSUB S_poll
280     GOSUB D_set
290     GOSUB Access
300 !
310     INPUT " NEXT DATA SET [ YES=0 , NO=1 ] ?",Loop#
320 !
330 EXIT IF Loop#="1"
340 !
350 END LOOP
360 !
370 !
380 STOP
390 !
400 !////////////////////
410 !/
420 !/
430 !/
440 !////////////////////
450 !
460 !-----!
470 !   Serial Polling                               !
480 !               RQS,Extention bit=1 , File Access bit=1 !
490 !-----!
500 !
510 S_poll:
520 !
530 OUTPUT Add;"FIL 1"
540 !
550 LOOP
560 !
570     LOOP
580     !
590     A=SPOLL(Add)                                ! Serial polling
600     EXIT IF BIT(A,6)=1 AND BIT(A,1)=1          ! RQS,Extention bit=1
610     WAIT .1

```

```

620      !
630      END LOOP
640      !
650      OUTPUT Add;"EER?"          ! REQUEST Extension register ?
660      ENTER Add;Eer#           ! READ Extension register
670      !
680      EXIT IF Eer#[9,9]="1"      ! File access" is end
690      !
700      END LOOP
710      !
720      RETURN
730      !
740      !
750      !-----!
760      !       Read File directory , Set Memory mode
770      !           & Select SAVE or RESAVE or RECALL
780      !-----!
790      !
800 D_set: !
810      !----- Read File directory
820      I=0
830      WAIT .3
840      OUTPUT Add;"FIL?"
850      !
860      LOOP
870      !
880      ENTER Add;Fil#
890      Read$(I)=Fil#[1,12]
900      EXIT IF Fil#="FIL 1"
910      !
920      I=I+1
930      !
940      END LOOP
950      !
960      !----- Set Memory mode
970      !
980      OUTPUT Add;"FIL 0"
990      !
1000     LOOP
1010     INPUT "MEMORY MODE SELECT [ PTN=0 , OTHERS=1 ] ?",Mem
1020     EXIT IF Mem=0 OR Mem=1
1030     END LOOP
1040     !
1050     OUTPUT Add;"MEM "&VAL$(Mem)
1060     !
1070     IF Mem=0 THEN
1080     Mem#="PTN"
1090     ELSE
1100     Mem#="OTH"
1110     END IF
1120     !
1130     !
1140     !----- Select SAVE,RESAVE,RECALL
1150     !
1160     LOOP
1170     INPUT "SELECT [ SAVE=0 , RESAVE=1 , RECALL=2 ] ?",Dta
1180     EXIT IF Dta=0 OR Dta=1 OR Dta=2
1190     END LOOP
1200     !
1210     IF Dta=0 THEN GOSUB Dsave
1220     IF Dta=1 THEN GOSUB Dresave
1230     IF Dta=2 THEN GOSUB Drecall
1240     !
1250     RETURN
1260     !
1270     !

```

```

1280 !-----
1290 !      * DATA SAVE *
1300 !      SAME Memory mode & File name --> ERROR !
1310 !-----
1320 !
1330 Dsave: !
1340 !
1350 LOOP
1360 !
1370 I=0
1380 !
1390 INPUT "** DATA SAVE ** FILE NAME [ 0~99 ] ?",Nam
1400 !
1410 Result$=" "
1420 LOOP
1430 !
1440 IF Mem$=Read$(I)[10,12] AND Nam=VAL(Read$(I)[2,3]) THEN
1450 Result$="SAME"
1460 END IF
1470 !
1480 I=I+1
1490 !
1500 EXIT IF Read$(I)[1,5]="FIL 1"
1510 !
1520 END LOOP
1530 !
1540 EXIT IF Result$<>"SAME"
1550 !
1560 END LOOP
1570 !
1580 OUTPUT Add;"SAV "&VAL$(Nam)
1590 !
1600 RETURN
1610 !
1620 !
1630 !-----
1640 !      * DATA RESAVE *
1650 !      SAME Memory mode & File name --> OK !
1660 !-----
1670 !
1680 Dresave: !
1690 !
1700 LOOP
1710 !
1720 I=0
1730 INPUT "** DATA RESAVE ** FILE NAME [ 0~99 ] ?",Nam
1740 !
1750 Result$=" "
1760 LOOP
1770 !
1780 IF Mem$=Read$(I)[10,12] AND Nam=VAL(Read$(I)[2,3]) THEN
1790 Result$="SAME"
1800 END IF
1810 !
1820 I=I+1
1830 !
1840 EXIT IF Read$(I)[1,5]="FIL 1"
1850 !
1860 END LOOP
1870 !
1880 EXIT IF Result$="SAME"
1890 END LOOP
1900 !
1910 OUTPUT Add;"RSV "&VAL$(Nam)
1920 !
1930 RETURN

```

```

1940 !
1950 !
1960 !-----|
1970 !      * DATA RECALL *
1980 !      SAME Memory mode $ File name --> OK !
1990 !-----|
2000 !
2010 Drecall:|
2020 !
2030 LOOP
2040 !
2050     I=0
2060 !
2070     INPUT "** DATA RECALL ** FILE NAME [ 0~99 ] ?",Nam
2080 !
2090     Result$="    "
2100     LOOP
2110     !
2120         IF Mem$=Read$(I)[10,12] AND Nam=VAL(Read$(I)[2,3]) THEN
2130             Result$="SAME"
2140         END IF
2150     !
2160         I=I+1
2170     !
2180     EXIT IF Read$(I)[1,5]="FIL 1"
2190 !
2200     END LOOP
2210 !
2220 EXIT IF Result$="SAME"
2230 END LOOP
2240 !
2250 OUTPUT Add;"RCL "&VAL$(Nam)
2260 !
2270 RETURN
2280 !
2290 !
2300 !-----|
2310 !      MEMORY ACCESS CONDITION ?
2320 !      & RESET FILE ACCESS bit
2330 !-----|
2340 !
2350 Access:|
2360 !
2370 LOOP
2380     OUTPUT Add;"MAC?"
2390     ENTER Add;Mac$
2400     EXIT IF Mac$="MAC 0"           ! Memory access END
2410     END LOOP
2420 !
2430     OUTPUT Add;"EER?"           ! Reset FILE ACCESS bit
2440     ENTER Add;Eer$
2450 !
2460     RETURN
2470 !
2480 !
2490     END

```

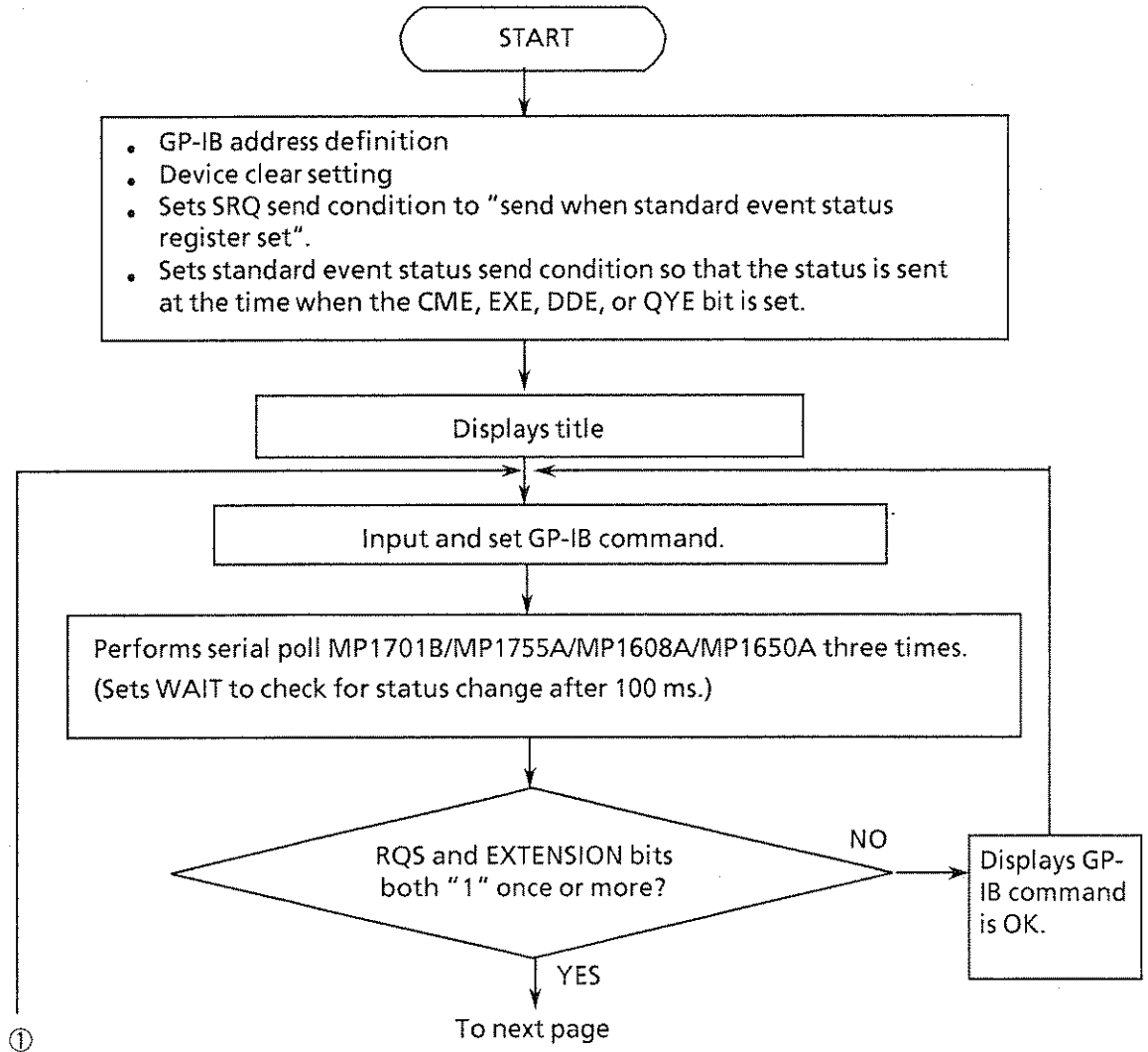
(8) Standard event status bytes check

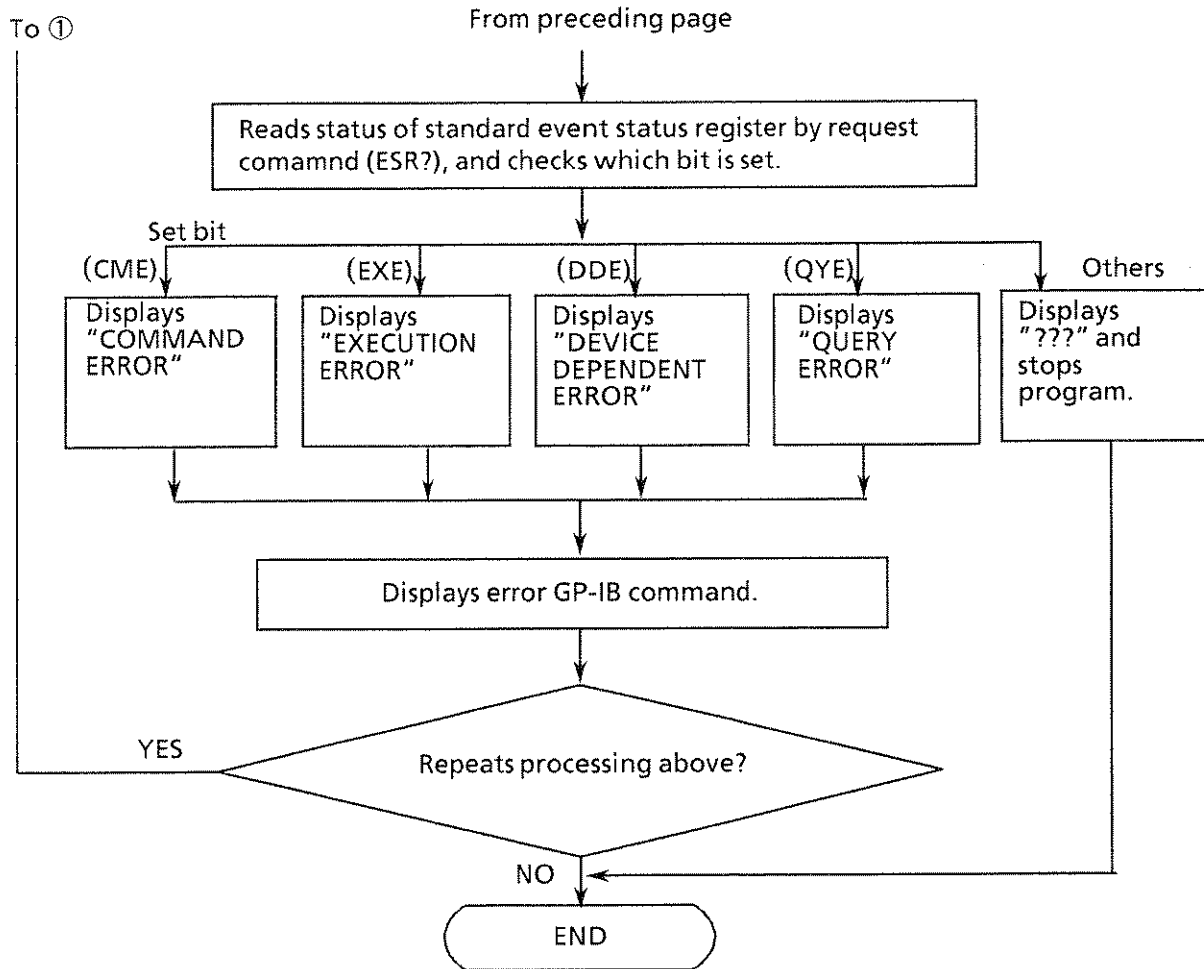
(COMMAND ERROR bit check → serial polling)

This program checks the CME, EXE, DDE, and QYE bits of the standard event status byte, and displays whether or not the input GP-IB command is normal on the CRT.

When the command is abnormal, the cause of the error at that time is displayed.

The GP-IB status byte is checked by serial polling and the status of the standard event status register which is obtained by data request command.





PROGRAM LISTING

```

10  !*****
20  !*
30  !*          MP1701B / MP1608A / MP1650A          *
40  !*  STANDARD EVENT STATUS REGISTER CHECK  SAMPLE SOFT  *
50  !*                                          ESE  *
60  !*****
70  !
80  !
90  !-----!
100 !          MAIN ROUTINE          !
110 !-----!
120 !
130 LET Add=700          ! Set Device Address
140 CLEAR Add           ! Device Clear
150 OUTPUT Add;"SRQ 32" ! SRQ : COMMAND ERROR bit
160 OUTPUT Add;"ESE 60" ! CME,EXE,DDE,QYE bit
170 !
180 PRINT "          ** MP1701B/MP1608A/MP1650A **"
190 PRINT "** STANDARD EVENT STATUS REGISTER CHECK **"
200 PRINT
210 !
220 LOOP
230 !
240   INPUT " INPUT ANY GP-IB COMMAND ?",Com#
250   OUTPUT Add;Com#
260   !
270   GOSUB S_poll
280   !
290   INPUT " NEXT COMMAND SET ? [ YES=0 , NO=1 ] ?",Loop#
300   !
310   EXIT IF Loop#="1"
320   !
330 END LOOP
340 !
350 STOP
360 !
370 !-----!
380 !          SUB ROUTINE          !
390 !-----!
400 !
410 S_poll:!------- Check Status byte
420 !
430 Byt=0
440 !
450 FOR I=0 TO 2
460   !
470   A=SPOLL(Add)          ! Serial polling
480   !
490   IF BIT(A,6)=1 AND BIT(A,5)=1 THEN Byt=A ! RQS,COMMAND ERROR bit=1
500   !
510   WAIT .1
520   !
530 NEXT I
540 !
550 IF BIT(Byt,6)=1 AND BIT(Byt,5)=1 THEN
560   GOSUB Err
570 ELSE
580   PRINT " GP-IB COMMAND IS OK "
590   PRINT
600 END IF
610 !
620 !
630 RETURN

```

```

640  !
650  !
660  Err: !----- ERROR
670  !
680  OUTPUT Add;"ESR?"
690  ENTER Add;Er#
700  !
710  IF Er#[9,9]="1" THEN PRINT " COMMAND ERROR !! "
720  IF Er#[10,10]="1" THEN PRINT " EXECUTION ERROR !! "
730  IF Er#[11,11]="1" THEN PRINT " DEVICE DEPENDENT ERROR !! "
740  IF Er#[12,12]="1" THEN PRINT " QUERY ERROR !! "
750  !
760  PRINT " INPUT COMMAND = "&Com#
770  PRINT
780  !
790  IF Er#[7,7]="1" OR Er#[8,8]="1" OR Er#[13,13]="1" OR Er#[14,14]="1" THEN
800    GOSUB Trap
810  END IF
820  !
830  RETURN
840  !
850  !
860  Trap: !-----
870  !
880  PRINT "???"
890  STOP
900  !
910  !
920  END

```

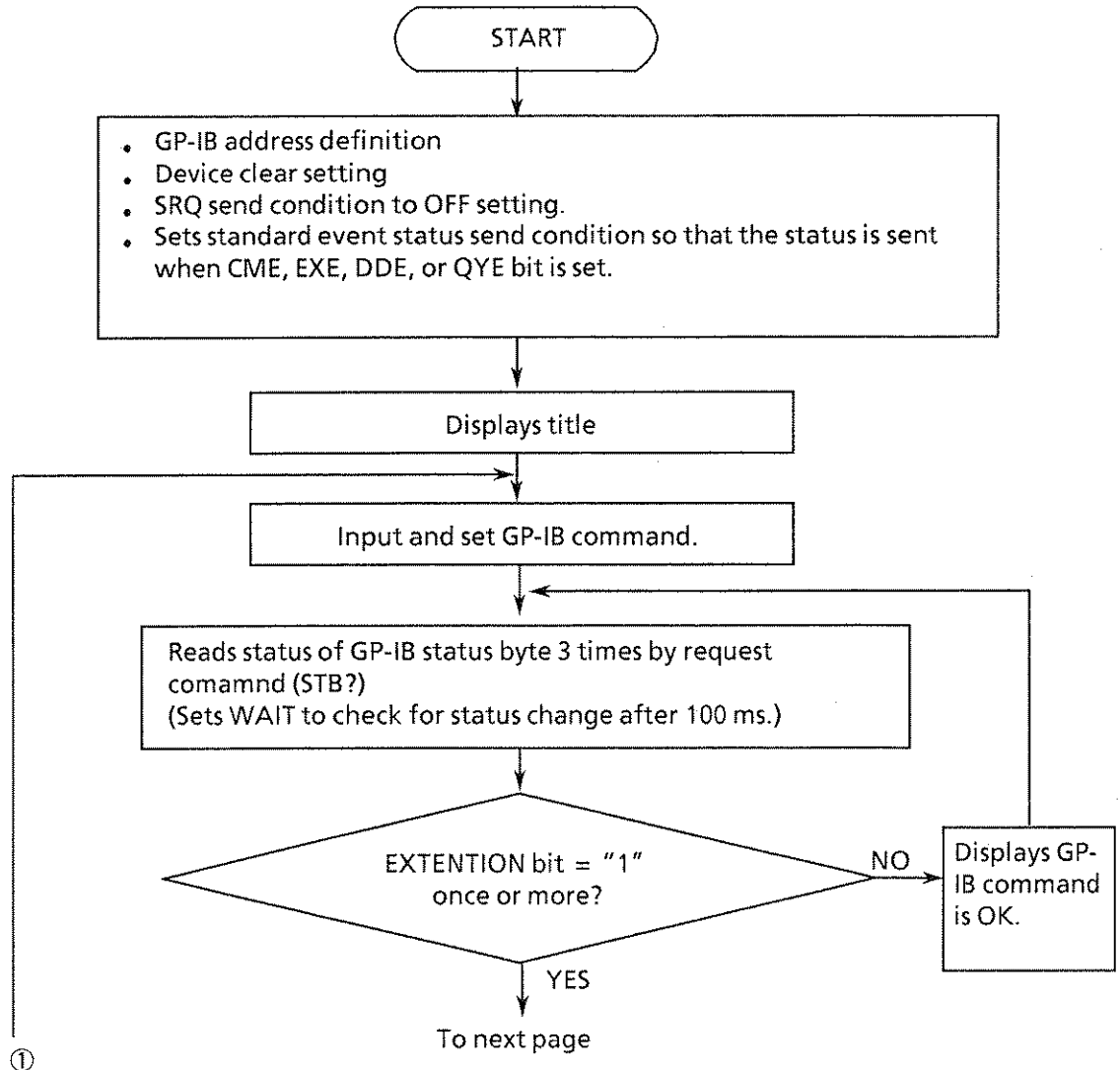
(9) Standard event status byte check

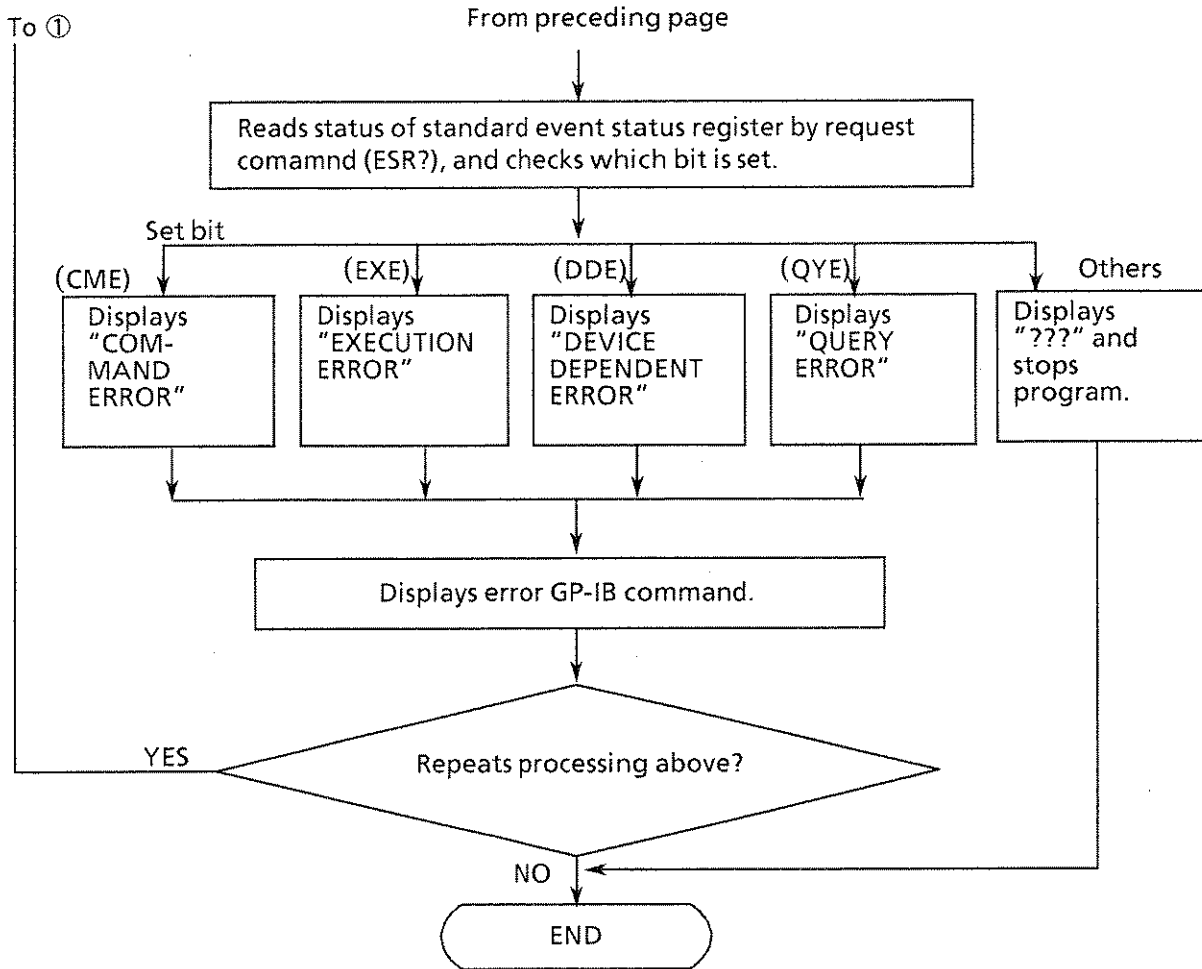
(COMAMND ERROR bit check → status byte register service request)

This program checks the CME, EXE, DDE, and QYE bits of the standard event status byte, and displays whether or not the input GP-IB command is normal on the CRT.

When the command is abnormal, the cause of the error at that time is displayed.

The GP-IB status byte and the status of the standard event status register are checked by data request command.





PROGRAM LISTING

```

10 !*****
20 !*
30 !*          MP1701B / MP1608A / MP1650A          *
40 !*          STANDARD EVENT STATUS REGISTER CHECK  SAMPLE SOFT  *
50 !*                                          ESE2  *
60 !*****
70 !
80 !
90 !-----!
100 !                      MAIN ROUTINE                      !
110 !-----!
120 !
130 Add=700          ! Set Device address
140 CLEAR Add        ! Device Clear
150 OUTPUT Add;"SRQ 0" ! SRQ : OFF
160 OUTPUT Add;"ESE 60" ! CME,EXE,DDE,QYE bit
170 !
180 PRINT "          ** MP1701B/MP1608A/MP1650A **"
190 PRINT "*** STANDARD EVENT STATUS REGISTER CHECK ***"
200 PRINT
210 !
220 LOOP
230 !
240     INPUT " INPUT ANY GP-IB COMMAND ? ",Com#
250     OUTPUT Add;Com#
260     !
270     GOSUB Stb_reg
280     !
290     INPUT " NEXT COMMAND SET ? [ YES=0 , NO=1 ] ? ",Loop#
300     !
310 EXIT IF Loop#="1"
320 !
330 END LOOP
340 !
350 STOP
360 !
370 !-----!
380 !                      SUB ROUTINE                      !
390 !-----!
400 !
410 Stb_reg: ! ----- Check Status byte
420 !
430 Byt#=""
440 !
450 FOR I=0 TO 2
460 !
470     OUTPUT Add;"STB?" ! REQUEST Status byte register ?
480     ENTER Add;Stb#    ! READ Status byte register
490     !
500     IF Stb#[9,9]="1" THEN ! COMMAND ERROR bit=1
510         Byt#="Stb#"
520     END IF
530     !
540     WAIT .1
550     !
560 NEXT I
570 !

```

```

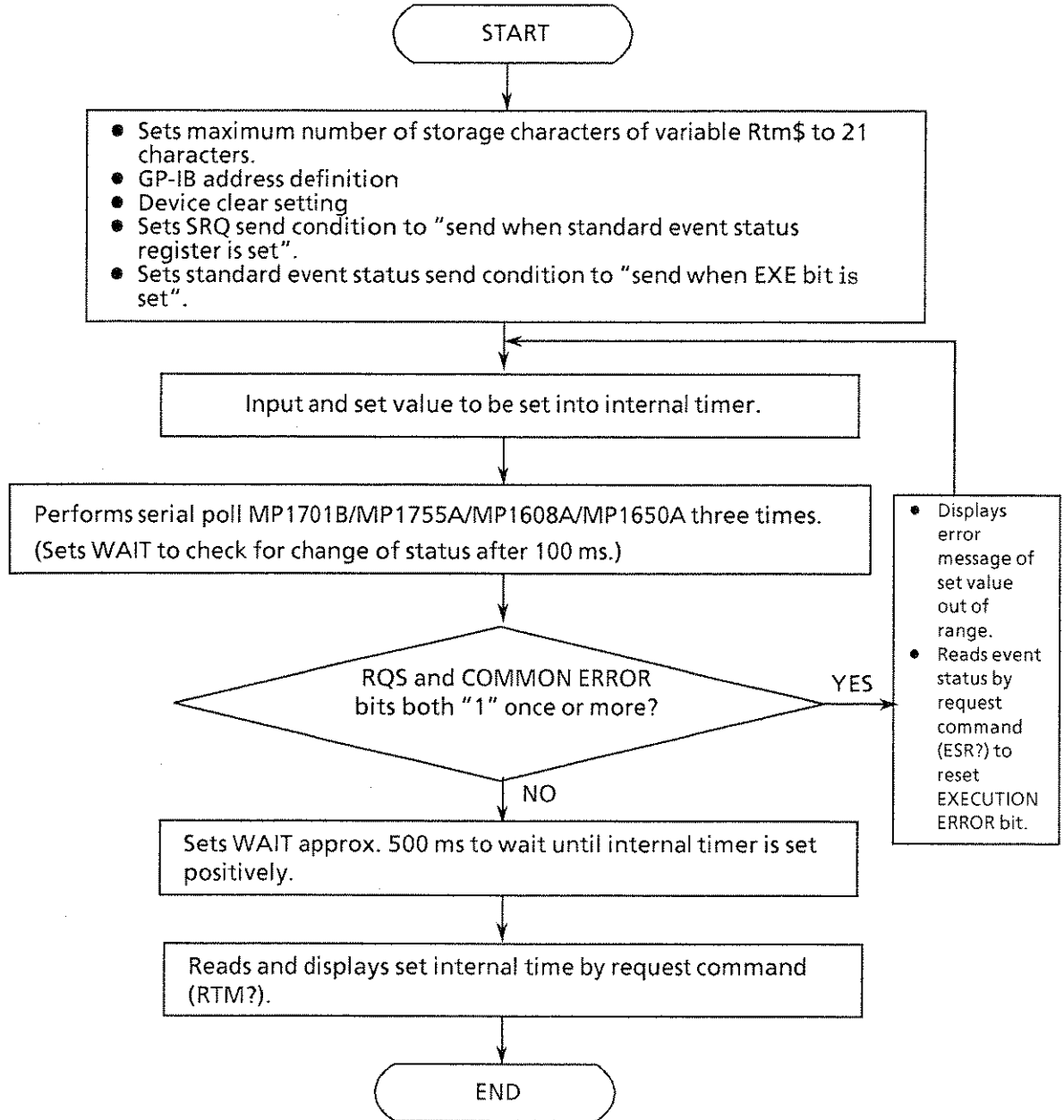
580 IF Byt#[9,9]="1" THEN
590   GOSUB Err
600 ELSE
610   PRINT " GF-IB COMMAND IS OK"
620   PRINT
630 END IF
640 !
650 !
660 RETURN
670 !
680 !
690 Err: !----- ERROR
700 !
710 OUTPUT Add;"ESR?"
720 ENTER Add;Er#
730 !
740 IF Er#[9,9]="1" THEN PRINT " COMMAND ERROR !! "
750 IF Er#[10,10]="1" THEN PRINT " EXECUTION ERROR !! "
760 IF Er#[11,11]="1" THEN PRINT " DEVICE DEPENDENT ERROR !! "
770 IF Er#[12,12]="1" THEN PRINT " QUERY ERROR !! "
780 !
790 PRINT " INPUT COMMAND = "&Com#
800 PRINT
810 !
820 IF Er#[7,7]="1" OR Er#[8,8]="1" OR Er#[13,13]="1" OR Er#[14,14]="1" THEN
830   GOSUB Trap
840 END IF
850 !
860 RETURN
870 !
880 !
890 Trap: !-----
900 !
910 PRINT "???"
920 STOP
930 !
940 !
950 END

```

(10) Internal timer setting

This program sets the internal timer, reads its value by data request command, and displays it on the CRT.

Input value is checked by the standard event status byte. The MP1701B/MP1755A/MP1608A/MP1650A needs several ms to set and start the timer positively. This program sets a WAIT of approximately 500 ms for this purpose.



PROGRAM LISTING

```

10  !*****
20  !*
30  !* MP1701B/MP1608A/MP1650A  REAL TIME SETTING SAMPLE SOFT *
40  !*
50  !*
60  !*****
70  !
80  !-----!
90  !                               MAIN  ROUTINE                               !
100 !-----!
110 !
120 DIM Rtm#[21]
130 LET Add=700           ! Set Device address
140 CLEAR Add            ! Device clear
150 OUTPUT Add;"SRQ 32"  ! SRQ : COMMAND ERROR bit
160 OUTPUT Add;"ESE 16"  ! Execution error
170 !
180 GOSUB D_set
190 !
200 STOP
210 !
220 !-----!
230 !                               SUB  ROUTINE                               !
240 !-----!
250 !
260 D_set: !----- DATA SET & DISPLAY REAL TIME
270 !
280 LOOP
290 !
300 INPUT "REAL TIME SETTING DATA  ** YEAR ** [ 0~99 ]",Y#
310 INPUT "REAL TIME SETTING DATA  ** MONTH ** [ 1~12 ]",M#
320 INPUT "REAL TIME SETTING DATA  ** DAY ** [ 1~31 ]",D#
330 INPUT "REAL TIME SETTING DATA  ** HOURS ** [ 0~23 ]",H#
340 INPUT "REAL TIME SETTING DATA  ** MINUTE ** [ 0~59 ]",Mi#
350 INPUT "REAL TIME SETTING DATA  ** SECOND ** [ 0~59 ]",S#
360 !
370 Rtm#=Y#&","&M#&","&D#&","&H#&","&Mi#&","&S#&
380 OUTPUT Add;"RTM "&Rtm#
390 !
400 GOSUB Check
410 !
420 EXIT IF Result#="OK"
430 !
440 END LOOP
450 !
460 WAIT .5
470 OUTPUT Add;"RTM?"
480 !
490 ENTER Add;Rtm#
500 PRINT Rtm#
510 !
520 RETURN
530 !
540 !

```

```

550 Check: !----- Check Set Data
560 !
570 Byt=0
580 !
590 FOR I=0 TO 2
600 !
610 A=SPOLL(Add) ! Serial polling
620 !
630 IF BIT(A,6)=1 AND BIT(A,5)=1 THEN ! RQS,COMMAND ERROR bit=1
640 Byt=A
650 END IF
660 !
670 WAIT .1
680 !
690 NEXT I
700 !
710 IF BIT(Byt,6)=1 AND BIT(Byt,5)=1 THEN
720 PRINT "EXECUTION ERROR !! INPUT AGAIN "
730 !
740 OUTPUT Add;"ESR?" ! Reset Execution error bit
750 !
760 ELSE
770 Result#="OK"
780 !
790 END IF
800 !
810 RETURN
820 !
830 !
840 END

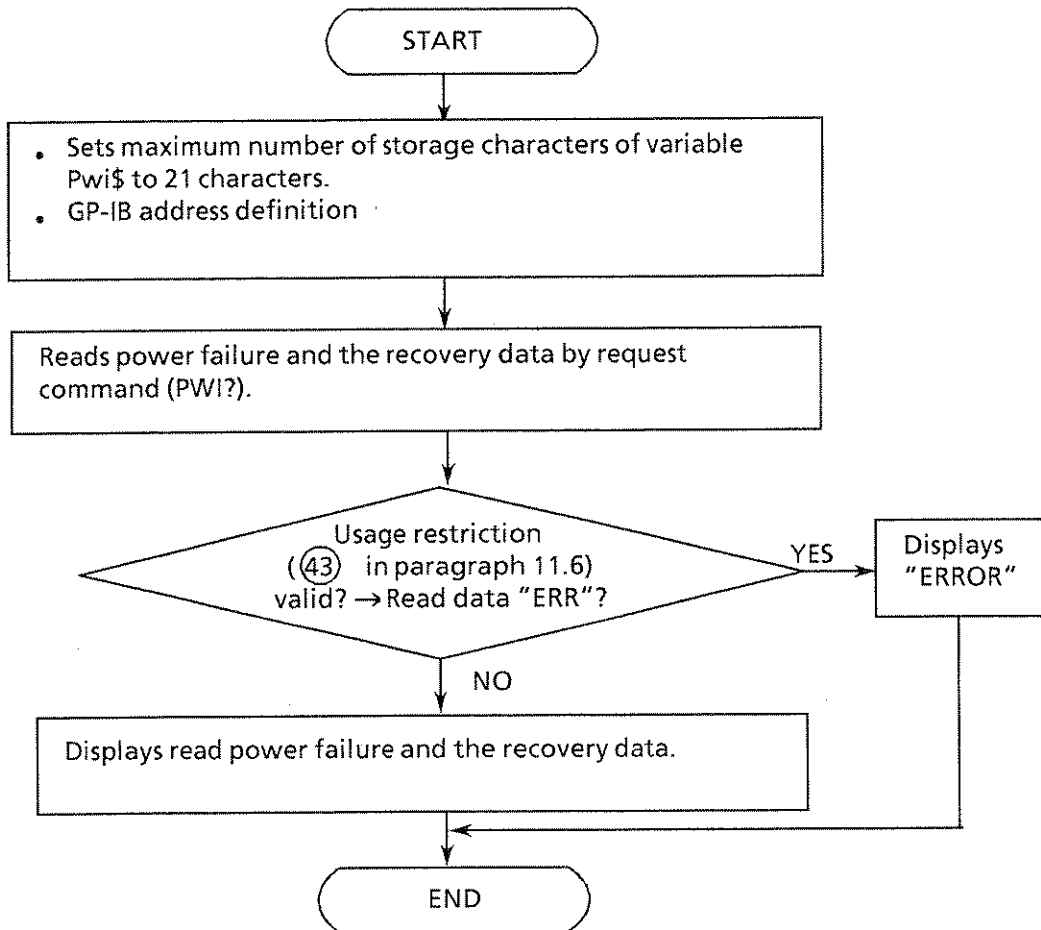
```

(11) Power failure and power recovery status check

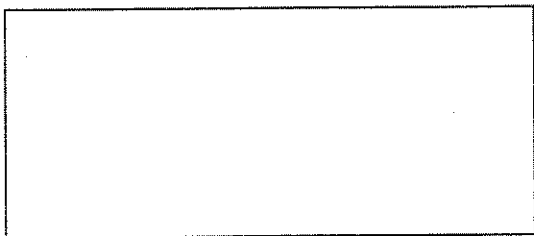
This program reads the MP1701B/MP1755A/MP1608A/MP1650A power failure and the recovery information by request command, and displays them on the CRT. It also shows an actual output example.

The time MP1701B/MP1755A/MP1608A/MP1650A power was turns OFF last can be checked by this.

The backup battery usage time can also be checked by recording the power failure interval time.



EXECUTED RESULT

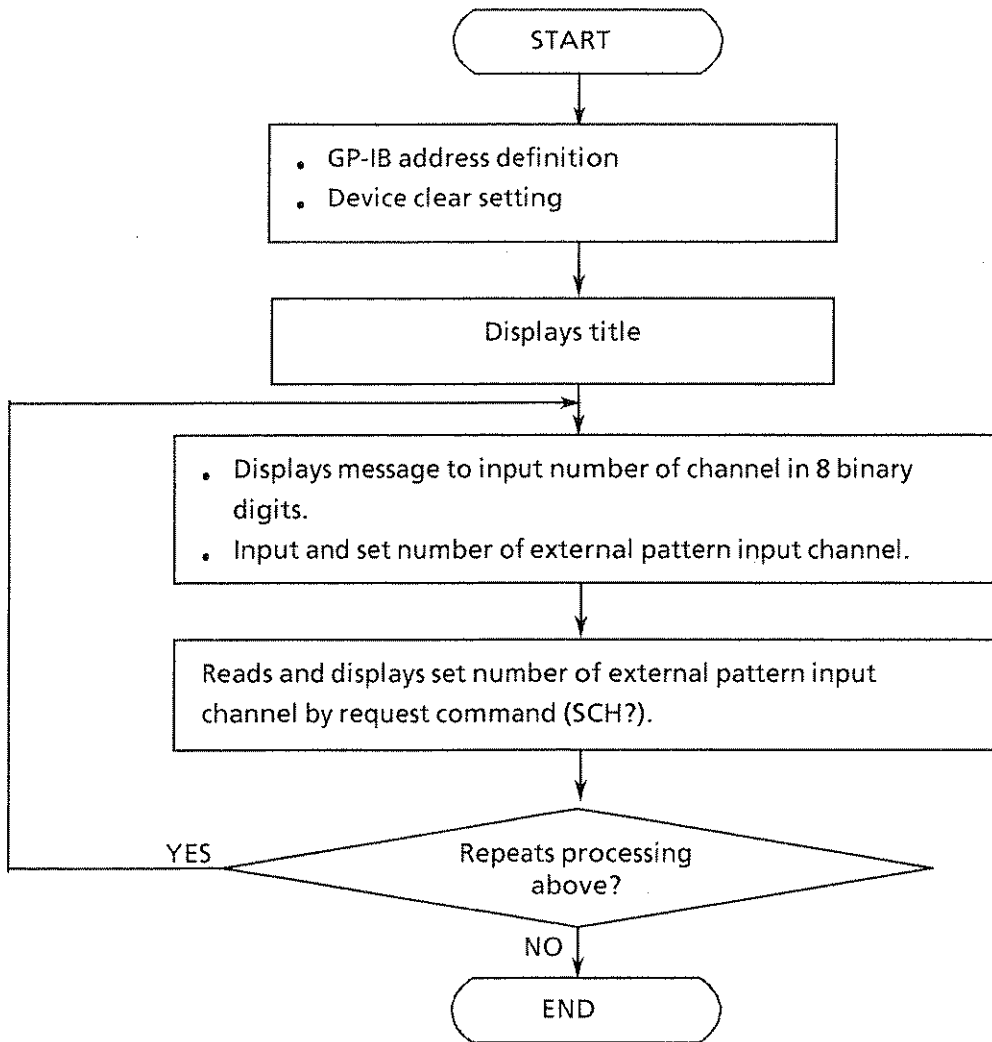


PROGRAM LISTING

```
10  !*****
20  !*
30  !* MP1701B/MP1608A/MP1650A  POWER FAIL INTERVAL? SAMPLE SOFT  *
40  !*
50  !*
60  !*****
70  !
80  DIM Pwi$(21)
90  LET Add=700
100 !
110 OUTPUT Add;"PWI?"
120 ENTER Add;Pwi$
130 !
140 IF Pwi$="ERR" THEN
150   PRINT "ERROR !!"
160 ELSE
170   PRINT Pwi$
180 !
190   FOR I=0 TO 1
200     ENTER Add;Pwi$
210     PRINT Pwi$
220   NEXT I
230 !
240 END IF
250 !
260 END
```

(12) Number of external pattern input channel setting

This program sets the number of external pattern input channel.



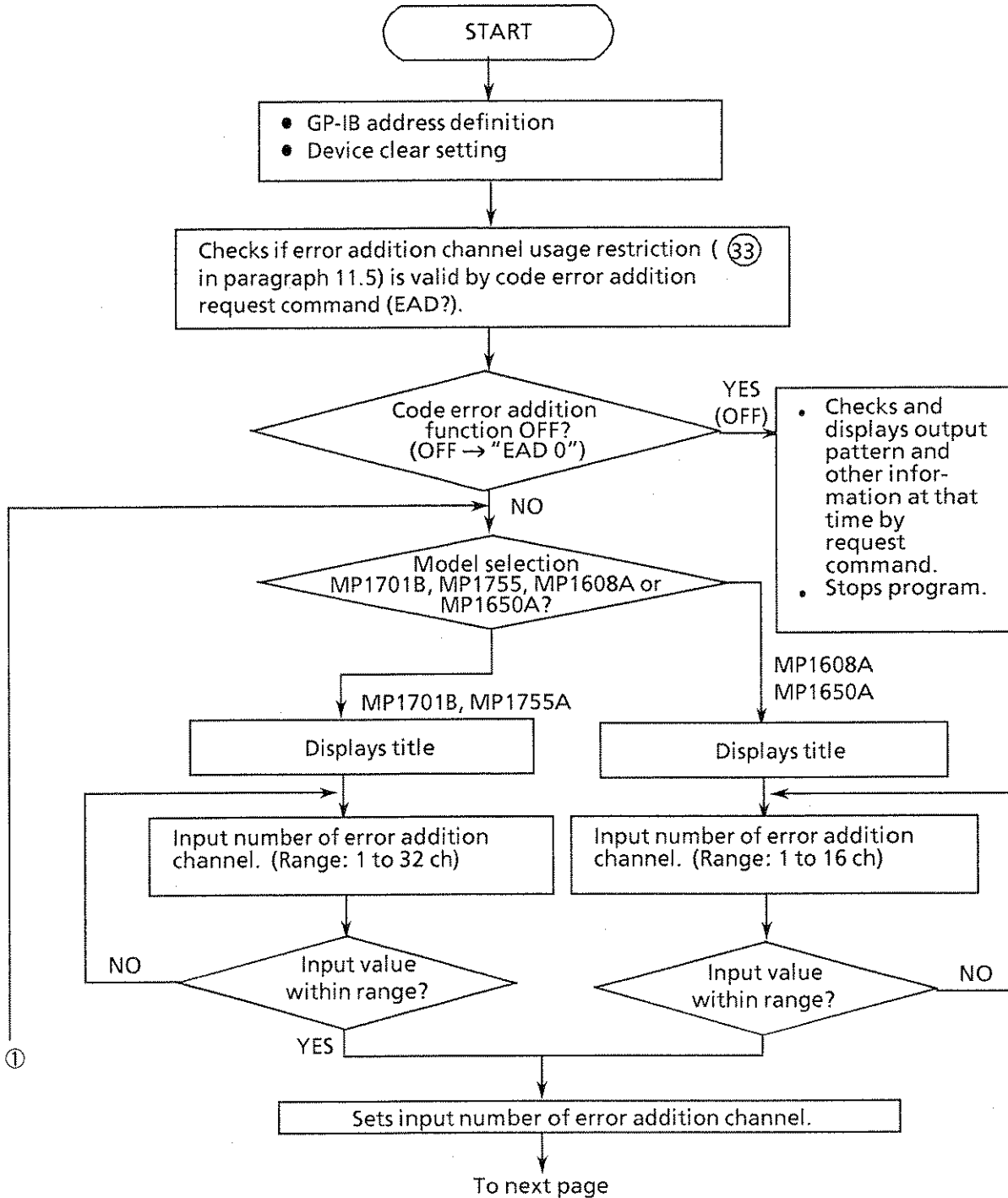
PROGRAM LISTING

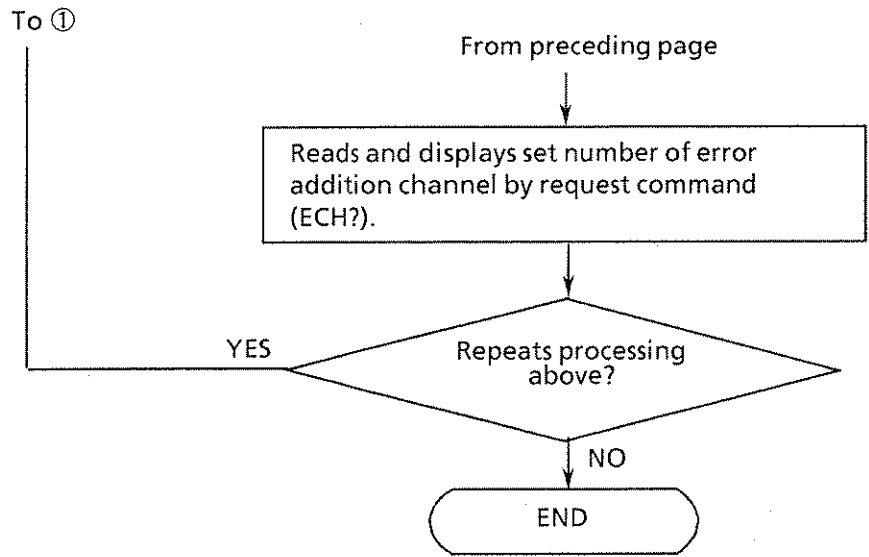
```
10      !*****
11      !*
12      !* MP1701B/MP1608A/MP1650A 1/8 SPEED CHANNEL SELECT SAMPLE SOFT *
13      !*
14      !*
15      !*
16      !*
17      !*
18      !*
19      !*
20      !*
21      !*
22      !*
23      !*
24      !*
25      !*
26      !*
27      !*
28      !*
29      !*
30      !*
31      !*
32      !*
33      !*
34      !*
35      !*
36      !*
37      !*
38      !*
39      !*
40      !*
41      !*
42      !*
43      !*
44      !*
45      !*
46      !*
47      !*
48      !*
49      !*
50      !*
51      !*
52      !*
53      !*
54      !*
55      !*
56      !*
57      !*
58      !*
59      !*
60      !*
61      !*
62      !*
63      !*
64      !*
65      !*
66      !*
67      !*
68      !*
69      !*
70      !*
71      !*
72      !*
73      !*
74      !*
75      !*
76      !*
77      !*
78      !*
79      !*
80      !*
81      !*
82      !*
83      !*
84      !*
85      !*
86      !*
87      !*
88      !*
89      !*
90      !*
91      !*
92      !*
93      !*
94      !*
95      !*
96      !*
97      !*
98      !*
99      !*
100     !*
101     !*
102     !*
103     !*
104     !*
105     !*
106     !*
107     !*
108     !*
109     !*
110     !*
111     !*
112     !*
113     !*
114     !*
115     !*
116     !*
117     !*
118     !*
119     !*
120     !*
121     !*
122     !*
123     !*
124     !*
125     !*
126     !*
127     !*
128     !*
129     !*
130     !*
131     !*
132     !*
133     !*
134     !*
135     !*
136     !*
137     !*
138     !*
139     !*
140     !*
141     !*
142     !*
143     !*
144     !*
145     !*
146     !*
147     !*
148     !*
149     !*
150     !*
151     !*
152     !*
153     !*
154     !*
155     !*
156     !*
157     !*
158     !*
159     !*
160     !*
161     !*
162     !*
163     !*
164     !*
165     !*
166     !*
167     !*
168     !*
169     !*
170     !*
171     !*
172     !*
173     !*
174     !*
175     !*
176     !*
177     !*
178     !*
179     !*
180     !*
181     !*
182     !*
183     !*
184     !*
185     !*
186     !*
187     !*
188     !*
189     !*
190     !*
191     !*
192     !*
193     !*
194     !*
195     !*
196     !*
197     !*
198     !*
199     !*
200     !*
201     !*
202     !*
203     !*
204     !*
205     !*
206     !*
207     !*
208     !*
209     !*
210     !*
211     !*
212     !*
213     !*
214     !*
215     !*
216     !*
217     !*
218     !*
219     !*
220     !*
221     !*
222     !*
223     !*
224     !*
225     !*
226     !*
227     !*
228     !*
229     !*
230     !*
231     !*
232     !*
233     !*
234     !*
235     !*
236     !*
237     !*
238     !*
239     !*
240     !*
241     !*
242     !*
243     !*
244     !*
245     !*
246     !*
247     !*
248     !*
249     !*
250     !*
251     !*
252     !*
253     !*
254     !*
255     !*
256     !*
257     !*
258     !*
259     !*
260     !*
261     !*
262     !*
263     !*
264     !*
265     !*
266     !*
267     !*
268     !*
269     !*
270     !*
271     !*
272     !*
273     !*
274     !*
275     !*
276     !*
277     !*
278     !*
279     !*
280     !*
281     !*
282     !*
283     !*
284     !*
285     !*
286     !*
287     !*
288     !*
289     !*
290     !*
291     !*
292     !*
293     !*
294     !*
295     !*
296     !*
297     !*
298     !*
299     !*
300     !*
```

(13) Error addition channel setting

This program sets the error addition channel.

The error addition channel cannot be set when the code error addition function is OFF. At this time, the output pattern related information is displayed on the CRT and program execution ends.





PROGRAM LISTING

```

10  !*****
20  !*
30  !* MP1701B/MF1608A/MF1650A  ERROR ADDITION ch SET SAMPLE SOFT *
40  !*
50  !*                               ERR_ADD *
60  !*****
70  !
80  !-----!
90  !                               MAIN ROUTINE                               !
100 !-----!
110 !
120 LET Add=700                ! Set Device Address
130 CLEAR Add                  ! Device Clear
140 !
150 OUTPUT Add;"EAD?"         ! REQUEST Error addition mode ?
160 ENTER Add;Ead#            ! READ Error addition mode
170 !
180 IF Ead#="EAD 0" THEN      ! OFF --> error
190   GOSUB Err
200   PRINT
210   PRINT "PROGRAM STOP !"
220   STOP
230 !
240 END IF
250 !
260 LOOP
270 !
280   INPUT " MODE SELECT [ MP1701B=0, MP1608A=1, MP1650A=2 ] ?",Mode#
290 !
300   SELECT Mode#
310   CASE "0"
320     GOSUB P10g
330   CASE "1"
340     Ppg#="MF1608A"
350     GOSUB P5g3g
360   CASE "2"
370     Ppg#="MF1650A"
380     GOSUB P5g3g
390   END SELECT
400 !
410   OUTPUT Add;"ECH "&VAL$(Ech)  ! Set Error addition channel
420 !
430   OUTPUT Add;"ECH?"
440   ENTER Add;Ech#
450   PRINT " ERROR ADDITION CHANNEL = "&Ech#[5,6]
460 !
470   INPUT " NEXT DATA SET [ YES=0 , NO=1 ] ?",Loop#
480 !
490   EXIT IF Loop#="1"
500 !
510 END LOOP
520 !
530 STOP
540 !

```

```

550  !-----
560  !                               SUB ROUTINE                               !
570  !-----
580  !
590  Mp1701b: !----- MP1701B DATA INPUT
600  !
610  PRINT "** MP1701B ERROR ADDITON CHANNEL SETTING **"
620  PRINT
630  LOOP
640  !
650  INPUT " ERROR ADDTION ch SET DATA [ 1~32 ] ?",Ech
660  !
670  EXIT IF Ech<33 AND Ech>0
680  END LOOP
690  !
700  RETURN
710  !
720  !
730  Mp1608a: !----- MP1608A DATA INPUT
740  !
750  PRINT "** MP1608A ERROR ADDITON CHANNEL SETTING **"
760  PRINT ""
770  LOOP
780  !
790  INPUT " ERROR ADDTION ch SET DATA [ 1~16 ] ?",Ech
800  !
810  EXIT IF Ech<17 AND Ech>0
820  END LOOP
830  !
840  RETURN
850  !
860  !
870  Err: !----- Display Pattern mode
880  !
890  PRINT "ERROR !! ERROR ADDTION MODE=OFF"
900  PRINT
910  !
920  OUTPUT Add;"PTN?"
930  ENTER Add;Ptn#
940  !
950  SELECT Ptn#[5,5]
960  !
970  CASE "0"
980  GOSUB Prog_w
990  !
1000 CASE "1"
1010 GOSUB Prog_d
1020 !
1030 CASE "2"
1040 PRINT "PATTERN MODE= PRBS.PN7"
1050 !
1060 CASE "3"
1070 PRINT "PATTERN MODE= PRBS.PN9"
1080 !
1090 CASE "5"
1100 PRINT "PATTERN MODE= PRBS.PN11"
1110 !
1120 CASE "6"
1130 PRINT "PATTERN MODE= PRBS.PN15"
1140 !

```

```

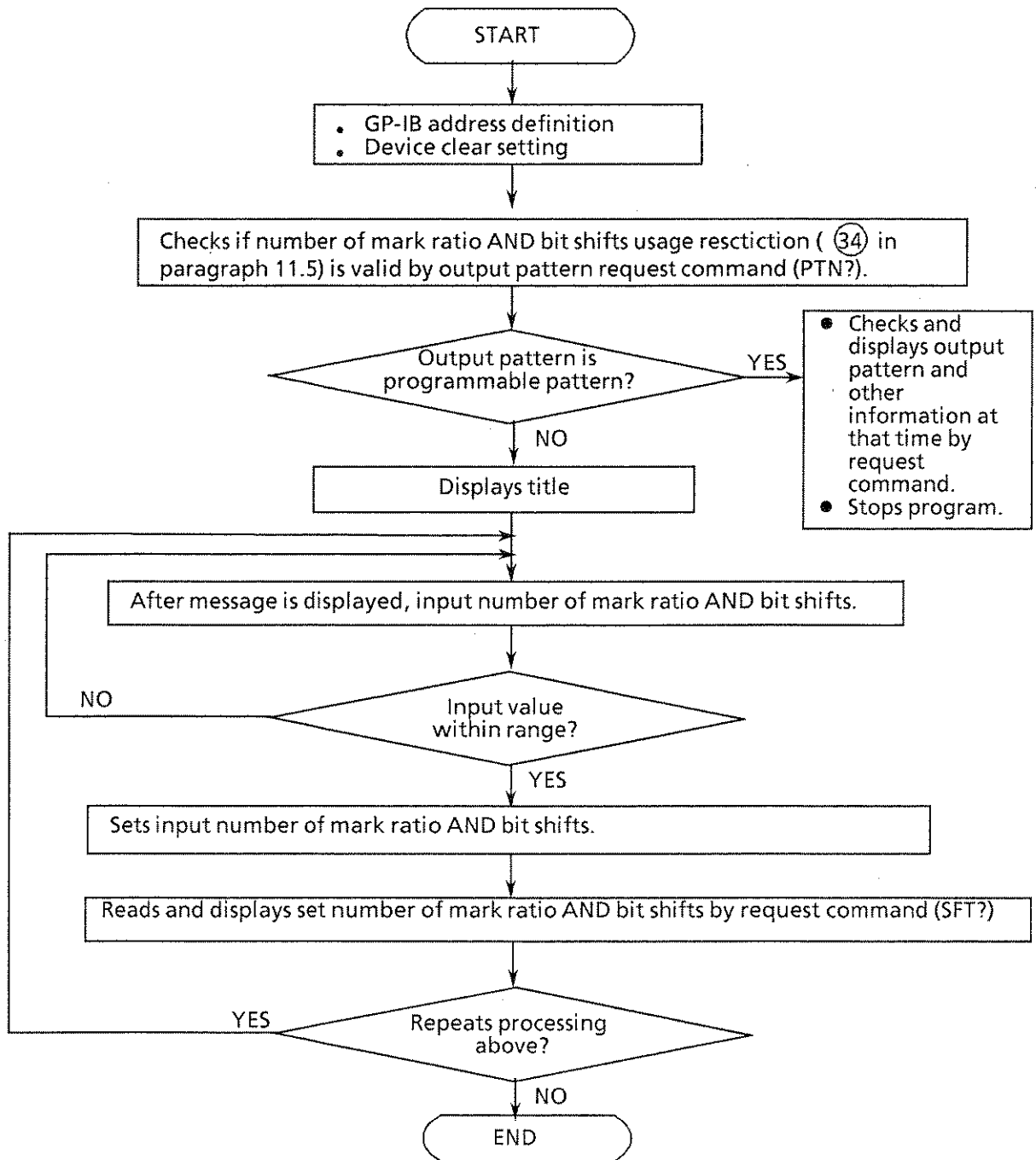
1150 CASE "7"
1160     PRINT "PATTERN MODE= PRBS.PN20"
1170     !
1180 CASE "8"
1190     PRINT "PATTERN MODE= PRBS.PN23"
1200     !
1210 CASE "9"
1220     PRINT "PATTERN MODE= PRBS.PN31"
1230     !
1240 END SELECT
1250     !
1260 RETURN
1270     !
1280 Prog_w: !----- WORD MODE
1290     !
1300 PRINT "PATTERN MODE=PROG.WORD MODE"
1310 PRINT
1320     !
1330 OUTPUT Add;"WNB?"
1340 ENTER Add;Wnb#
1350 OUTPUT Add;"WLN?"
1360 ENTER Add;Wln#
1370 OUTPUT Add;"PAG?"
1380 ENTER Add;Pag#
1390     !
1400 PRINT "NUMBER OF WORD="&Wnb#[4,9]
1410 PRINT "WORD LENGTH="&Wln#[4,6]
1420 PRINT "PAGE="&Pag#[4,9]
1430     !
1440 RETURN
1450     !
1460 Prog_d: !----- DATA MODE
1470     !
1480 PRINT "PATTERN MODE=PROG.DATA MODE"
1490 PRINT
1500     !
1510 OUTPUT Add;"DLN?"
1520 ENTER Add;Dln#
1530 OUTPUT Add;"PAG?"
1540 ENTER Add;Pag#
1550     !
1560 PRINT "DATA LENGTH="&Dln#[4,10]
1570 PRINT "PAGE="&Pag#[4,9]
1580     !
1590 RETURN
1600     !
1610     !
1620 END

```

(14) Number of mark ratio AND bit shifts setting

This program sets the number of mark ratio AND bit shifts.

The number of mark ratio AND bit shifts cannot be set when the output pattern is in the programmable mode. At that time, the information related to output pattern is displayed on the CRT and program execution is ended.



PROGRAM LITING

```

10  !*****
20  !*
30  !* MF1701B/MF1608A/MF1650A MARK RATIO AND BIT SHIFT  SAMPLE SOFT *
40  !*
50  !*
60  !*****
70  !
80  !
90  !-----!
100 !                               MAIN ROUTINE                               !
110 !-----!
120 !
130 LET Add=700                ! Set Device Address
140 CLEAR Add                  ! Device Clear
150 !
160 PRINT "** MF1701B/MF1608A/MF1650A  MARK RATIO AND BIT SHIFT **"
170 PRINT
180 !
190 OUTPUT Add;"PTN?"          ! REQUEST Pattern mode ?
200 ENTER Add;Ptn#            ! READ Pattern mode
210 !
220 IF Ptn#[5,5]="0" OR Ptn#[5,5]="1" THEN ! Word/Data Mode-->ERROR
230   GOSUB Err
240   PRINT " PROGRAM STOP !"
250   STOP
260   !
270 END IF
280 !
290 LOOP
300 !
310   GOSUB Mark
320   !
330   INPUT " NEXT DATA SET [ YES=0 , NO=1 ] ?",Loop#
340   !
350 EXIT IF Loop#="1"
360 !
370 END LOOP
380 !
390 !
400 STOP
410 !
420 !-----!
430 !                               SUB ROUTINE                               !
440 !-----!
450 !
460 Mark: !----- Set Mark ratio bit shift
470 !
480 LOOP
490 !
500   INPUT "MARK RATIO [ 1 BIT SHIFT=0, 3 BIT SHIFT=1 ] ?",Sft
510 EXIT IF Sft=0 OR Sft=1
520 !
530 END LOOP
540 !
550 OUTPUT Add;"SFT "&VAL$(Sft)
560 !
570 OUTPUT Add;"SFT?"
580 ENTER Add;Sft#
590 !

```

```

600 IF Sft#[5,5]="0" THEN
610   Sft#="1"
620 ELSE
630   Sft#="3"
640 END IF
650 !
660 PRINT "MARK RATIO AND BIT SHIFT = "&Sft#
670 !
680 RETURN
690 !
700 !
710 Err: !----- Display Pattern data
720 !
730 IF Ptn#[5,5]="0" THEN
740 !
750   PRINT " ERROR !!   PATTERN MODE=PROG.WORD MODE"
760   PRINT
770   OUTPUT Add;"WNB?"
780   ENTER Add;Wnb#
790   OUTPUT Add;"WLN?"
800   ENTER Add;Wln#
810   OUTPUT Add;"PAG?"
820   ENTER Add;Pag#
830   !
840   PRINT " NUMBER OF WORD ="&Wnb#[4,9]
850   PRINT " WORD LENGTH ="&Wln#[4,6]
860   PRINT " PAGE ="&Pag#[4,9]
870   PRINT
880   !
890 ELSE
900 !
910   PRINT " ERROR !!   PATTERN MODE=PROG.DATA MODE"
920   PRINT
930   OUTPUT Add;"DLN?"
940   ENTER Add;Dln#
950   OUTPUT Add;"PAG?"
960   ENTER Add;Pag#
970   !
980   PRINT " DATA LENGTH ="&Dln#[4,10]
990   PRINT " PAGE ="&Pag#[4,9]
1000  PRINT
1010  !
1020 END IF
1030 !
1040 RETURN
1050 !
1060 !
1070 END

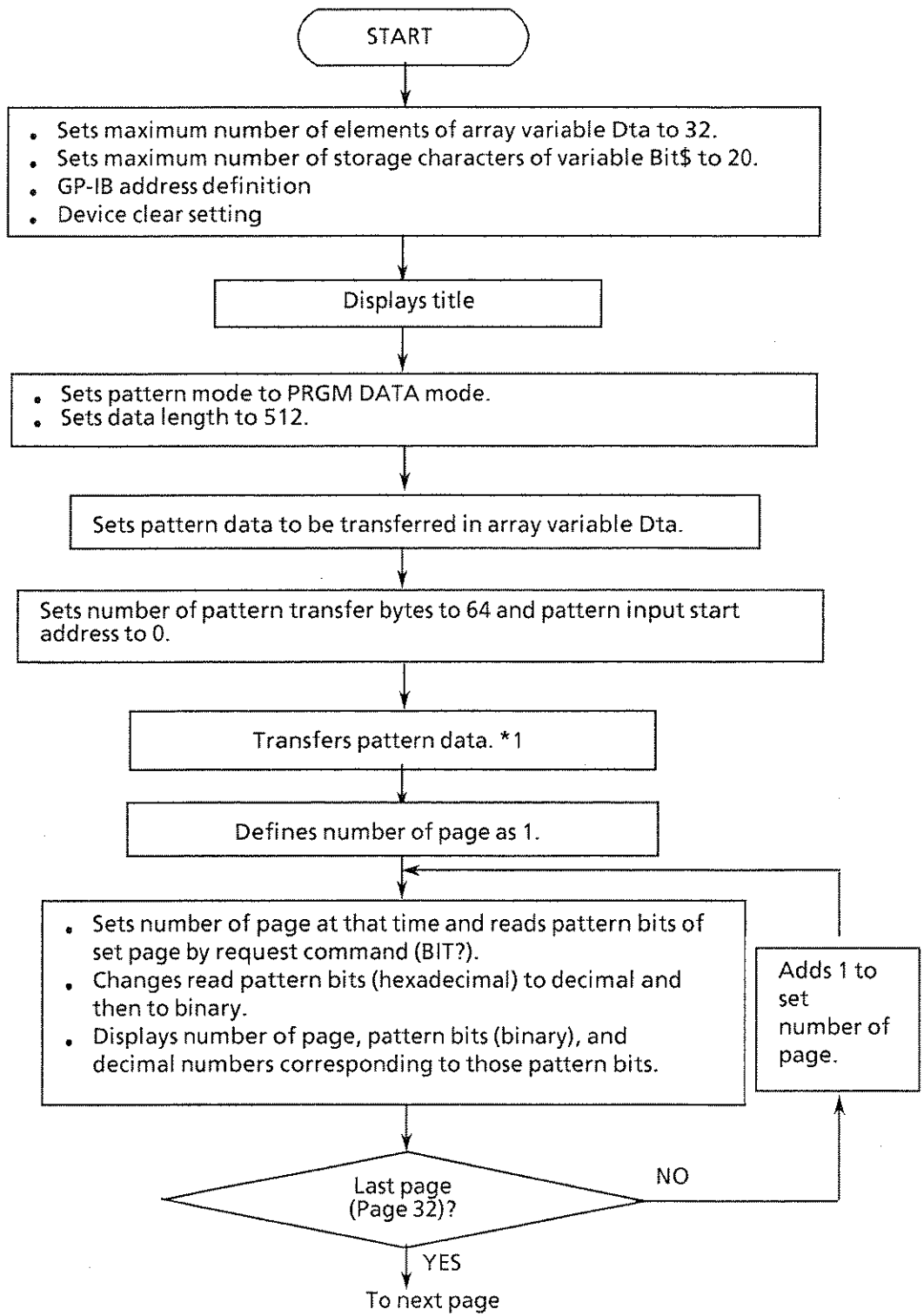
```

(15) Pattern data transfer by DMA

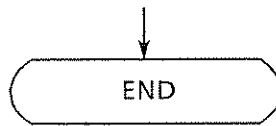
This program performs pattern data transfer by DMA with a HP9000 Series Computer as the controller. The actual executed result is shown. The relationship between each array variable Dta and the set value (decimal notation, binary notation, hexadecimal notation) in it is shown in Table 14-2. This relationship differs somewhat with the controller used.

Table 14-2 Relationship between Array Variable and Set

Array variable	Set value (Decimal number)	Corresponding binary number and BIT LED NO.																Corresponding hexadecimal number	Corresponding number of page
		16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1		
Dta(0)	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1H	1
Dta(1)	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2H	2
Dta(2)	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	4H	3
Dta(3)	8	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	8H	4
Dta(4)	16	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	10H	4
Dta(5)	32	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	20H	6
Dta(6)	64	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	40H	7
Dta(7)	128	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	80H	8
Dta(8)	256	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	100H	9
Dta(9)	512	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	200H	10
Dta(10)	1024	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	400H	11
Dta(11)	2048	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	800H	12
Dta(12)	4096	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1000H	13
Dta(13)	8192	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2000H	14
Dta(14)	16384	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000H	15
Dta(15)	32767	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	7FFFH	16
Dta(16)	-32768	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8000H	17
Dta(17)	-16384	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	C000H	18
Dta(18)	-8192	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	E000H	19
Dta(19)	-4096	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	F000H	20
Dta(20)	-2048	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	F800H	21
Dta(21)	-1024	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	FC00H	22
Dta(22)	-512	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	FE00H	23
Dta(23)	-256	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	FF00H	24
Dta(24)	-128	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	FF80H	25
Dta(25)	-64	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	FFC0H	26
Dta(26)	-32	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	FFE0H	27
Dta(27)	-16	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	FFF0H	28
Dta(28)	-8	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	FFF8H	29
Dta(29)	-4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	FFFCH	30
Dta(30)	-2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	FFFEH	31
Dta(31)	-1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	FFFFH	32



From preceding page



1 Pattern data transfer using <OUPUT Add USING“W”;Dta()>

W: Outputs an integer of the two's complement of 16 bits (two bits).

Since the GP-IB interface board is 8 bit I/O, the upper byte is transferred first and then the lower byte.

*: Outputs the entire defined array Dta.

EXECUTED RESULT

```
** MF1701B/MF1608A PATTERN DATA DMA TRANSFER **  
PATTERN BIT PAGE=1      0000000000000001      1  
PATTERN BIT PAGE=2      0000000000000010      2  
PATTERN BIT PAGE=3      0000000000000100      4  
PATTERN BIT PAGE=4      0000000000001000      8  
PATTERN BIT PAGE=5      0000000000010000     16  
PATTERN BIT PAGE=6      0000000000100000     32  
PATTERN BIT PAGE=7      0000000001000000     64  
PATTERN BIT PAGE=8      0000000010000000    128  
PATTERN BIT PAGE=9      0000000100000000    256  
PATTERN BIT PAGE=10     0000001000000000    512  
PATTERN BIT PAGE=11     0000010000000000   1024  
PATTERN BIT PAGE=12     0000100000000000   2048  
PATTERN BIT PAGE=13     0001000000000000   4096  
PATTERN BIT PAGE=14     0010000000000000   8192  
PATTERN BIT PAGE=15     0100000000000000  16384  
PATTERN BIT PAGE=16     0111111111111111  32767  
PATTERN BIT PAGE=17     1000000000000000 -32768  
PATTERN BIT PAGE=18     1100000000000000 -16384  
PATTERN BIT PAGE=19     1110000000000000 -8192  
PATTERN BIT PAGE=20     1111000000000000 -4096  
PATTERN BIT PAGE=21     1111100000000000 -2048  
PATTERN BIT PAGE=22     1111110000000000 -1024  
PATTERN BIT PAGE=23     1111111000000000 -512  
PATTERN BIT PAGE=24     1111111100000000 -256  
PATTERN BIT PAGE=25     1111111110000000 -128  
PATTERN BIT PAGE=26     1111111111000000 -64  
PATTERN BIT PAGE=27     1111111111100000 -32  
PATTERN BIT PAGE=28     1111111111110000 -16  
PATTERN BIT PAGE=29     1111111111111000 -8  
PATTERN BIT PAGE=30     1111111111111100 -4  
PATTERN BIT PAGE=31     1111111111111110 -2  
PATTERN BIT PAGE=32     1111111111111111 -1
```

PROGRAM LISTING

```
10  !*****
20  !
30  !           MP1701B / MP1608A / MP1650A           *
40  !   PROGRAMMABLE PATTERN DATA DMA TRANSFER   SAMPLE SOFT *
50  !                                           DMA *
60  !*****
70  !
80  DIM D(31)
90  DIM Bit#[20]
100 LET Add=700           ! DEVICE ADDRESS
110 CLEAR Add           ! DEVICE CLEAR
120 !
130 PRINT "** MP1701B/MP1608A/MP1650A PATTERN DATA DMA TRANSFER **"
140 PRINT
150 !
160 OUTPUT Add;"PTN 1"           ! PATTERN MODE : PROG.DATA
170 OUTPUT Add;"DLN 512"       ! DATA LENGTH : 512
180 !
190 !----- SET PATTERN BIT DATA -----
200 !
210 D(0)=1
220 D(1)=2
230 D(2)=4
240 D(3)=8
250 D(4)=16
260 D(5)=32
270 D(6)=64
280 D(7)=128
290 D(8)=256
300 D(9)=512
310 D(10)=1024
320 D(11)=2048
330 D(12)=4096
340 D(13)=8192
350 D(14)=16384
360 D(15)=32767
370 D(16)=-32768
380 D(17)=-16384
390 D(18)=-8192
400 D(19)=-4096
410 D(20)=-2048
420 D(21)=-1024
430 D(22)=-512
440 D(23)=-256
450 D(24)=-128
460 D(25)=-64
470 D(26)=-32
480 D(27)=-16
490 D(28)=-8
500 D(29)=-4
510 D(30)=-2
520 D(31)=-1
530 !
540 OUTPUT Add;"WRT 64,0"
550 !
560 OUTPUT Add USING "W";D(*)
570 !
580 !
```

```
590 FOR I=1 TO 32
600 !
610 OUTPUT Add;"PAG "&VAL$(I)
620 OUTPUT Add;"BIT?"
630 ENTER Add;Bit$
640 Bit$=Bit$[17,20]
650 !
660 Bt=IVAL(Bit$,16) ! HEX --> DIG
670 Bit$=IVAL$(Bt,2) ! DIG --> BIN
680 !
690 IMAGE "PATTERN BIT PAGE=",AA,XXX,AAAAAAAAAAAAAAAA,XX,DDDDDD
700 PRINT USING 690;VAL$(I);Bit$;Dta(I-1)
710 !
720 NEXT I
730 !
740 END
```

(16) Pattern data transfer by DMA

This program performs pattern data transfer by DMA with the HP9000 Series computer as the controller.

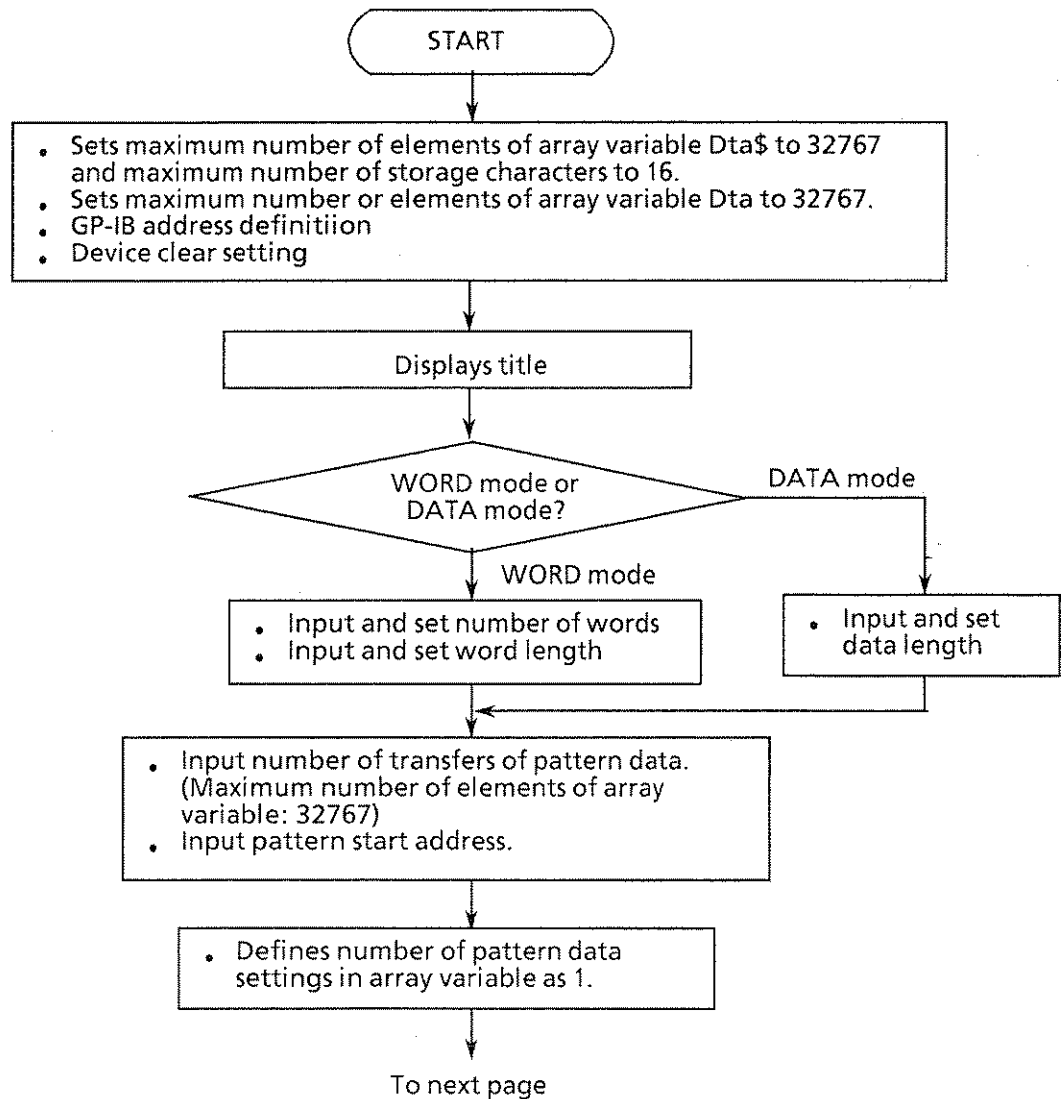
When this program is executed, output pattern selection and conditions setting are performed.

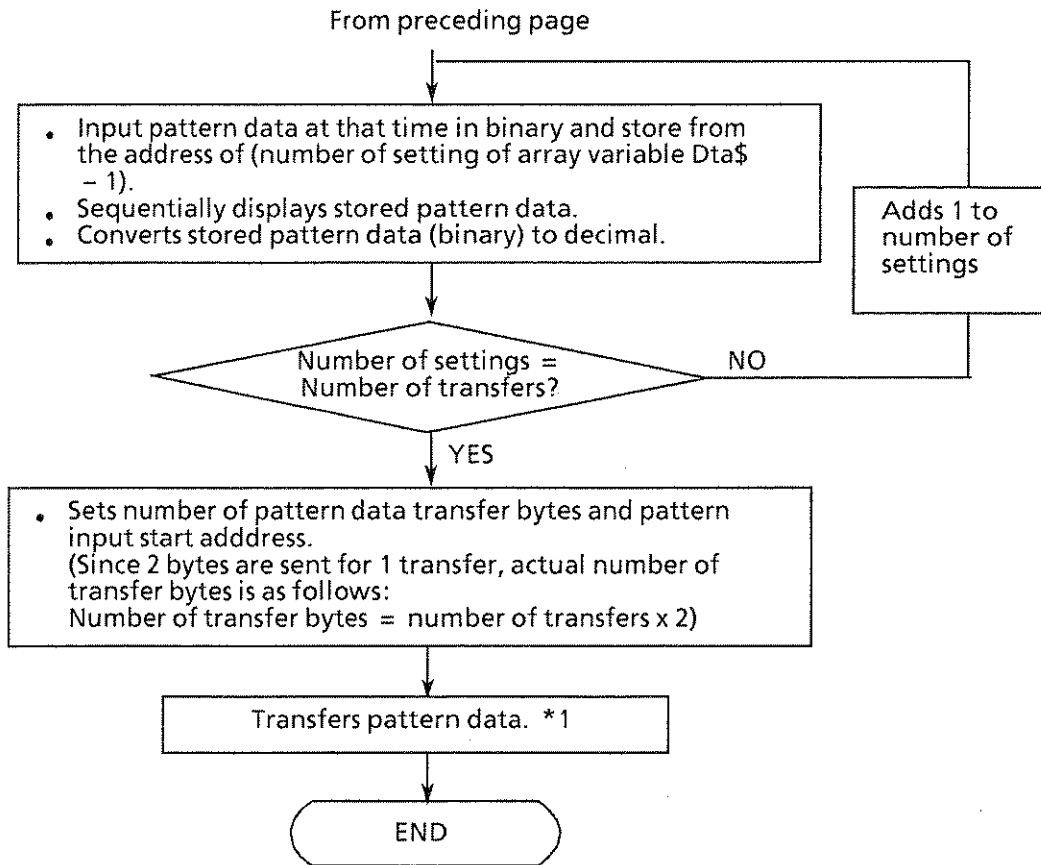
Next, the number of pattern data to be output (number of pages to be transferred = number of transfer operations) and the pattern input start address (start page: [number of start page - 1] = pattern input start address) are defined.

Next, the necessary number of pattern data are input in binary. When all the data has been input, the pattern data is transferred by DMA to the MP1701B/MP1755A/MP1608A/MP1650A.

The pattern data actually transferred uses the HP9000 Series command Dta(*), and all the necessary elements of the defined array variable are output.

Therefore, it takes 20 s or less for the program to end.





1: Pattern data transfer using <OUTPUT Add USING "W";Dta()>

W: Outputs an integer of the two's complement of 16 bytes (two bytes).

Since the GP-IB interface board is 8 bit I/O, the upper byte is transferred first and then the lower byte.

*: Outputs the entire defined array Dta.

PROGRAM LISTING

```

10  !*****
20  !
30  !           MP1701B / MP1608A / MP1650A           *
40  !   PROGRAMMABLE PATTERN DATA DMA TRANSFER  SAMPLE SOFT  *
50  !                                           DMA2 *
60  !*****
70  !
80  !-----!
90  !           MAIN ROUTINE                               !
100 !-----!
110 !
120 DIM Dta$(32766)[16]
130 DIM Dta(32766)
140 LET Add=700
150 CLEAR Add
160 !
170 PRINT " *** PROGRAMMABLE PATTERN BIT  DMA TRANSFER *** "
180 PRINT
190 !
200 LOOP
210   INPUT "PROGRAMMABLE PATTERN MODE [ WORD=0 , DATA=1 ]?",Ptn
220   EXIT IF Ptn=0 OR Ptn=1
230   END LOOP
240 !
250   OUTPUT Add;"PTN "&VAL$(Ptn)
260 !
270   IF Ptn=1 THEN
280     GOSUB Prog_d
290   ELSE
300     GOSUB Prog_w
310   END IF
320 !
330   GOSUB D_set
340 !
350   STOP
360 !
370 !-----!
380 !           SUB ROUTINE                               !
390 !-----!
400 !
410 Prog_w: !----- PROG.Word  mode
420 !
430 PRINT "           ** PROG.WORD MODE **"
440 PRINT
450 !
460 LOOP
470   INPUT "NUMBER OF WORD DATA [ 1~32768 ] ?",Wnb
480   EXIT IF Wnb>0 AND Wnb<32769
490   END LOOP
500 !
510   OUTPUT Add;"WNB "&VAL$(Wnb)
520 !
530 !
540 LOOP
550   INPUT "WORD LENGTH [ 2~16 ] ?",Wln
560   EXIT IF Wln>1 AND Wln<17
570   END LOOP
580 !
590   OUTPUT Add;"WLN "&VAL$(Wln)
600 !
610   RETURN

```

```

620  !
630  !
640 Prog_d:  !----- PROG.Data mode
650  !
660  PRINT "          ** PROG.DATA MODE **"
670  PRINT
680  !
690  LOOP
700  INPUT "DATA LENGTH [ 2~524288 ] ?",Dln
710  EXIT IF Dln>1 AND Dln<524289
720  END LOOP
730  !
740  OUTPUT Add;"DLN "&VAL$(Dln)
750  RETURN
760  !
770  !
780 D_set:  !----- SET Bit pattern data
790  !
800  INPUT "PATTERN DATA SETTING COUNT ? [ MAX 32767 ]",Cnt
810  INPUT "PATTERN DATA WRITE TOP ADDRESS ? [ 0 ~ 32767 ]",Top#
820  !
830  FOR I=1 TO Cnt
840  !
850  INPUT "BIT PATTERN DATA BIT16-->BIT1 [0/1]?",Dta$(I-1)
860  PRINT "BIT PATTERN DATA= "&Dta$(I-1)
870  !
880  Dta(I-1)=IVAL(Dta$(I-1),2)          ! BIN --> DIG
890  !
900  NEXT I
910  !
920  OUTPUT Add;"WRT "&VAL$(Cnt*2)&","&Top#
930  !
940  OUTPUT Add USING "W";Dta(*)
950  !
960  RETURN
970  !
980  !
990  END

```

14.5 Sample Programs Using J3100 as the Controller

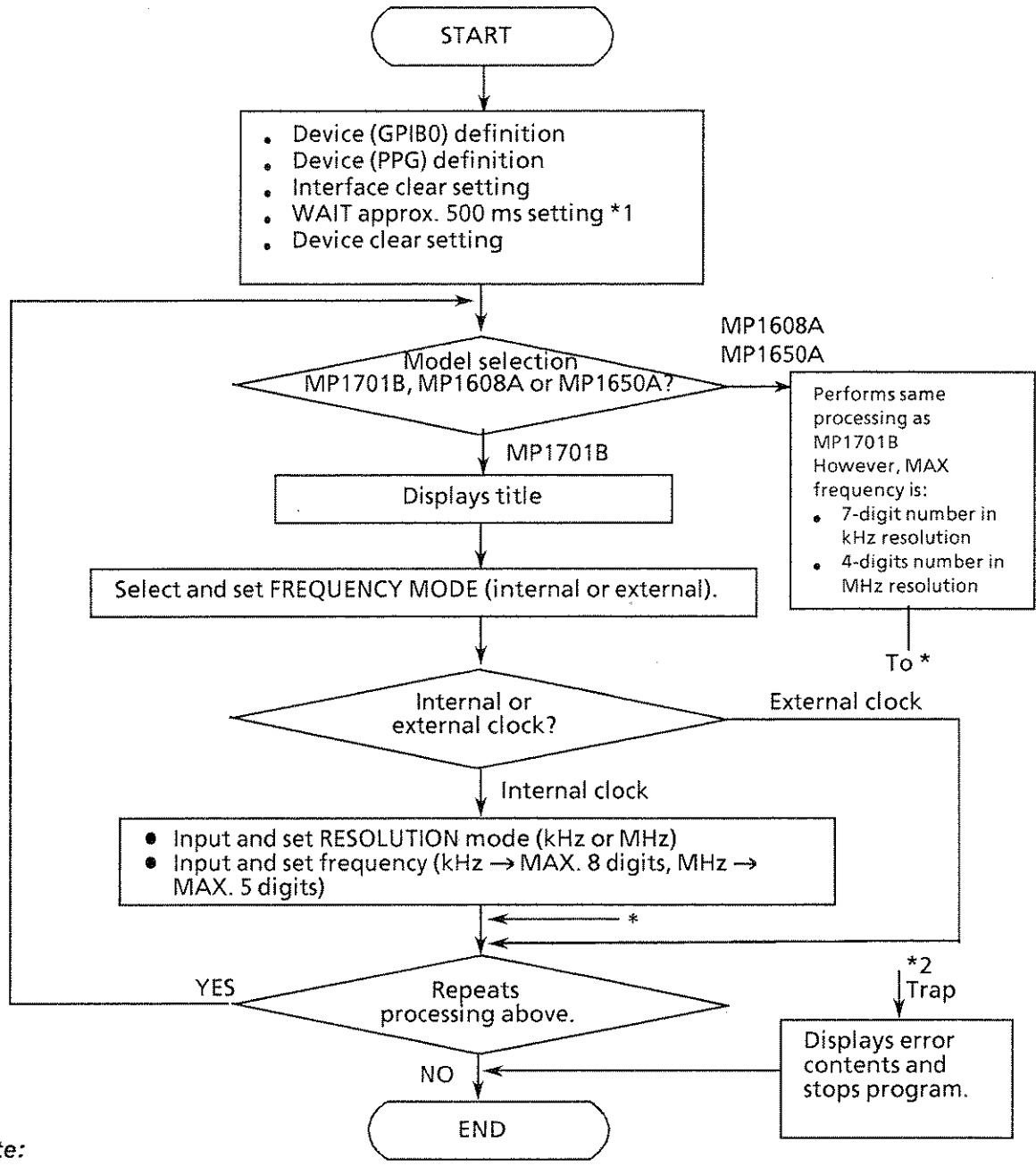
<Description of common parts of sample programs >

- ① <common shared IBSTA%, IBERR, IBCNT> is defined at the beginning of the program for GP-IB control.
- ② *1 500 ms WAIT after interface clear executed
Before the next command is executed after interface clear was executed, an approximately 500 ms WAIT is inserted until the MP1701B/MP1755A/MP1608A/MP1650A receives and executes interface clear from the J3100 and becomes stable.
- ③ *2 Trap
When a GP-IB command is received from the J3100 or when data is read from the MP1701B/MP1755A/MP1608A/MP1650A, the command's normal execution is checked. If there was an error, the error information is displayed and the program stops.
Similar processing is also performed for IBFIND (device definition), IBSIC (interface clear), and IBRSP (serial polling).
- ④ wrtcmd: '-----Write comamnd -----
wrt\$ = wrt\$ + chr\$(13) + chr\$(10)
CR, LF are connected to the end of the data stored in variable wrt\$ and reassigned to the same variable.
CR: chr\$(13)
LF: chr\$(10)
CALL IBWRT(PPG%, wrt\$) sends the data stored in variable wrt\$ to the MP1701B/MP1755A/MP1608A/MP1650A.
- ⑤ readcmd: '----- Read command -----
rd\$ = SPACE\$(○○)
Initializes variable rd\$ that stores the data read from the MP1701B/MP1755A/MP1608A/MP1650A to spaces.
○○ in SPACE\$(○○) is [number of characters of read data + 2 characters (CR and LF)].
CALL IBRD(PPG%,rd\$) reads data from the MP1701B/MP1755A/MP1608A/MP1650A and stores it in variable rd\$.

(1) Frequency setting

This program performs clock frequency control.

It selects internal or external clock, and inputs the frequency resolution and clock frequency.



Note:
See the preceding page for *1 and *2.

```

*****
*
*   MF1701B/MF1608A/MF1650A  FREQUENCY SAMPLE SOFT_1  *
*
*
*
*
*
*****
-----
MAIN ROUTINE
-----

common shared IBSTAX,IBERR%,IBCNT%      ' Setup GPIB-PC functions
GOSUB gpinit                             ' Setup GPIB interface

DO
  CLS

  INPUT " MODE SELECT [ MF1701B=0, MF1608A=1, MF1650A=2 ]";mode#
  PRINT

  SELECT CASE mode#
    CASE "0": GOSUB 10G
    CASE "1": ppg#="MF1608A": GOSUB 5G3G
    CASE "2": ppg#="MF1650A": GOSUB 5G3G
  END SELECT

  INPUT " NEXT DATA SET [ YES=0 , NO=1 ] ";loop#

LOOP UNTIL loop#="1"

STOP

-----
SUB ROUTINE
-----

10G: -----  MF1701B Data set  -----

PRINT " ***** MF1701B FREQUENCY SAMPLE SOFT *****"
PRINT ""
INPUT " FREQUENCY MODE [ EXTERNAL=0 , INTERNAL=1 ] ";clk#
wrt# = "CLK "+clk# : GOSUB wrtcmd

IF clk# = "1" THEN

  INPUT " FREQUENCY RESOLUTION [ kHz=0 , MHz=1 ] ";res#
  wrt# = "RES "+res# : GOSUB wrtcmd

  IF res# = "1" THEN
    INPUT " FREQUENCY DATA [ MHz=5 figures ] ";frq#
  ELSE
    INPUT " FREQUENCY DATA [ Khz=8 figures ] ";frq#
  END IF

  wrt# = "FRQ "+frq# : GOSUB wrtcmd

END IF

RETURN

```

```

5636:  -----  MP1608A,MP1650A Data set  -----
PRINT " ***** "+ppg#+ " FREQUENCY SAMPLE SOFT *****"
PRINT ""
INPUT " CLOCK MODE SELECT [ EXTERNAL=0 , INTERNAL=1 ] ";clk#
wrt# = "CLK "+clk# : GOSUB wrtcmd
IF clk# = "1" THEN
    INPUT " FREQUENCY RESOLUTION [ kHz=0 , MHz=1 ] ";res#
    wrt# = "RES "+res# : GOSUB wrtcmd
    IF res# = "1" THEN
        INPUT " FREQUENCY DATA [ MHz=4 figures ] ";frq#
    ELSE
        INPUT " FREQUENCY DATA [ Khz=7 figures ] ";frq#
    END IF
    wrt# = "FRQ "+frq# : GOSUB wrtcmd      Send Frequency data
END IF
RETURN

```

```

gpinit:  -----  Setup GPIB interface  -----
CALL IBFIND("GPIB0", GPIB0)      ' Open device (GPIB0)
IF GPIB0% < 0 THEN GOTO trap      ' system error
CALL IBFIND("PPG", PPG%)         ' Open device (PPG)
IF PPG% < 0 THEN GOTO trap        ' system error
CALL IBSIC(GPIB0%)              ' Interface clear
IF IBSTAX% < 0 THEN GOTO trap     ' system error
tim = 0.5
GOSUB waidly
CALL IBCLR(PPG%)                ' Device clear
RETURN

```

```

wrtcnd:  -----  Write command  -----
wrt# = wrt# + chr$(13) + chr$(10)
CALL IBWRT(PPG%, wrt#)          ' Write command
IF IBSTAX% < 0 THEN GOTO trap    ' Trap
RETURN

```

```

waidly:  -----  Wait delay  -----
stm = TIMER
etm = TIMER
WHILE etm - stm < tim
    etm = TIMER
    IF etm < stm THEN etm = etm + 86400
WEND
RETURN

```

```

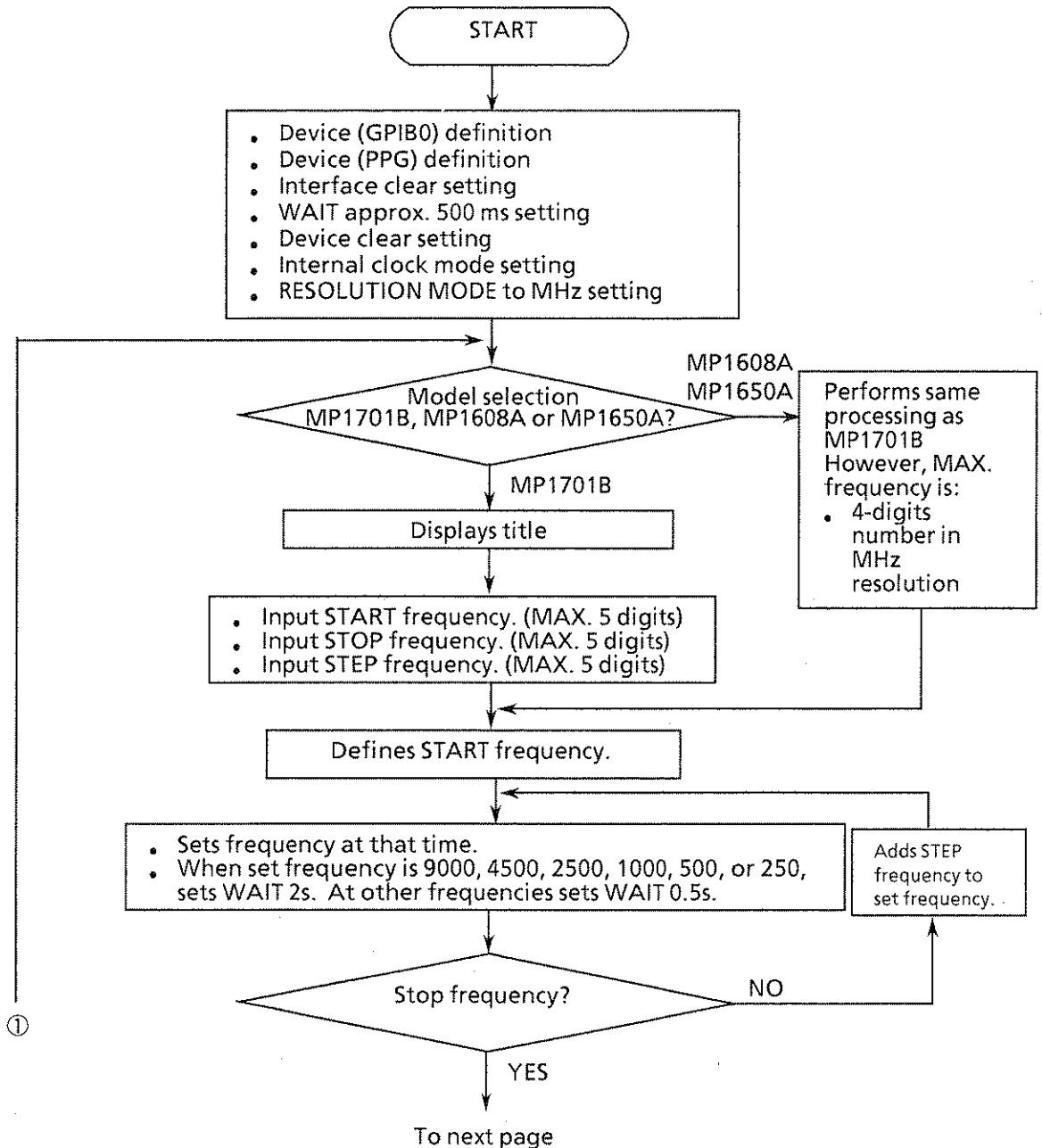
trap:  -----  System trap  -----
PRINT "IBERR%:" + STR$(IBERR%)
STOP
END

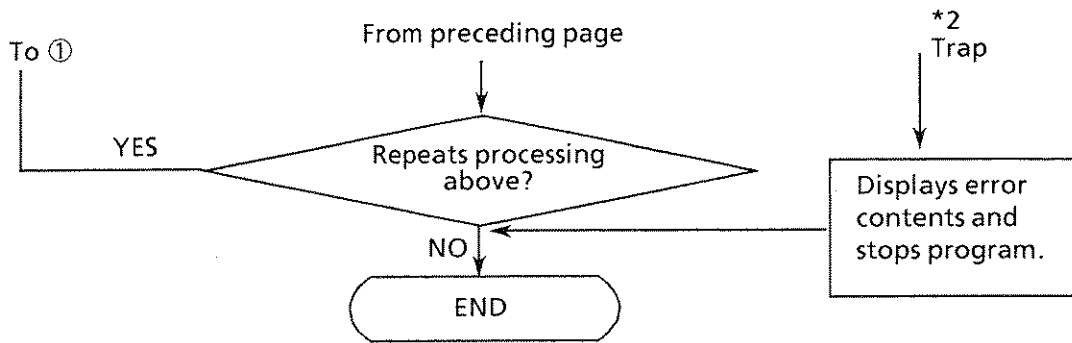
```

(2) Frequency setting

This program increases the internal clock frequency from a start frequency to a stop frequency in a certain step width.

The frequency switching speed is every 0.5 second, but the set state is held for 2 seconds at 250, 500, 1000, 2500, 4500, and 9000 MHz.





Note:

See the beginning of this paragraph 14.5 for *1 and *2.

```

*****
*
*      MP1701B/MP1608A/MP1650A  FREQUENCY SAMPLE SOFT_2
*
*
*      FRQ_2
*
*****

-----
MAIN ROUTINE
-----

common shared IBSTA%,IBERR%,IBCNT%  ' Setup GPIB-PC  functions
GOSUB gpinit                        ' Setup GPIB interface
wrt# = "CLK 1" : GOSUB wrtcmd        ' Send Clock mode : INTERNAL
wrt# = "RES 1" : GOSUB wrtcmd        ' Send Resolution mode : MHz

DO
  CLS

  INPUT " MODE SELECT [ MP1701B=0, MP1608A=1, MP1650A=2 ] ";mode#
  PRINT

  SELECT CASE mode#
    CASE "0": GOSUB 10G
    CASE "1": ppg#="MP1608A": GOSUB 563G
    CASE "2": ppg#="MP1650A": GOSUB 563G
  END SELECT

  INPUT " NEXT DATA SET [ YES=0 , NO=1 ] ";loop#

LOOP UNTIL loop#="1"

STOP

-----
SUB ROUTINE
-----

10G:  -----  MP1701B Data set  -----

PRINT " ***** MP1701B FREQUENCY SAMPLE SOFT *****"
PRINT ""

INPUT " START FREQUENCY DATA [ MHz=5 figures,MAX ] ";startf#
INPUT " STOP  FREQUENCY DATA [ MHz=5 figures,MAX ] ";stopf#
INPUT " STEP  FREQUENCY DATA [ MHz=5 figures,MAX ] ";stepf#

GOSUB SetFrq

RETURN

563G:  -----  MP1608A,MP1650A Data set  -----

PRINT " ***** "+ppg#+" FREQUENCY SAMPLE SOFT *****"
PRINT ""

INPUT " START FREQUENCY DATA [ MHz=4 figures,MAX ] ";startf#
INPUT " STOP  FREQUENCY DATA [ MHz=4 figures,MAX ] ";stopf#
INPUT " STEP  FREQUENCY DATA [ MHz=4 figures,MAX ] ";stepf#

GOSUB SetFrq

RETURN

```

```

SetFrq:  ----- Set frequency data -----
FOR I=VAL(startf#) TO VAL(stopf#) STEP VAL(stepf#)
    wrt# = "FRQ " + STR$(I) : GOSUB wrtcmd
    IF I=9000 OR I=4500 OR I=2500 OR I=1000 OR I=500 OR I=250 THEN
        tim = 2 : GOSUB waidly
    ELSE
        tim = 0.5 : GOSUB waidly
    END IF
NEXT I
RETURN

gpinit:  ----- Setup GPIB interface -----
CALL IBFIND("GPIB0", GPIB%) ' Open device (GPIB0)
IF GPIB% < 0 THEN GOTO trap ' system error

CALL IBFIND("PPG", PPG%) ' Open device (PPG)
IF PPG% < 0 THEN GOTO trap ' system error

CALL IBSID(GPIB%) ' Interface clear
IF IBSTA% < 0 THEN GOTO trap ' system error

tim = 0.5
GOSUB waidly

CALL IBCLR(PPG%) ' Device clear

RETURN

wrtcmd:  ----- Write command -----

wrt# = wrt# + chr$(13) + chr$(10)
CALL IBWRT(PPG%, wrt#) ' Write command
IF IBSTA% < 0 THEN GOTO trap ' Trap

RETURN

waidly:  ----- Wait delay -----

stm = TIMER
etm = TIMER

WHILE etm - stm < tim
    etm = TIMER
    IF etm < stm THEN etm = etm + 86400
WEND

RETURN

trap:  ----- System trap -----

FRINT "IBERR:" + STR$(IBERR%)
STOP

END

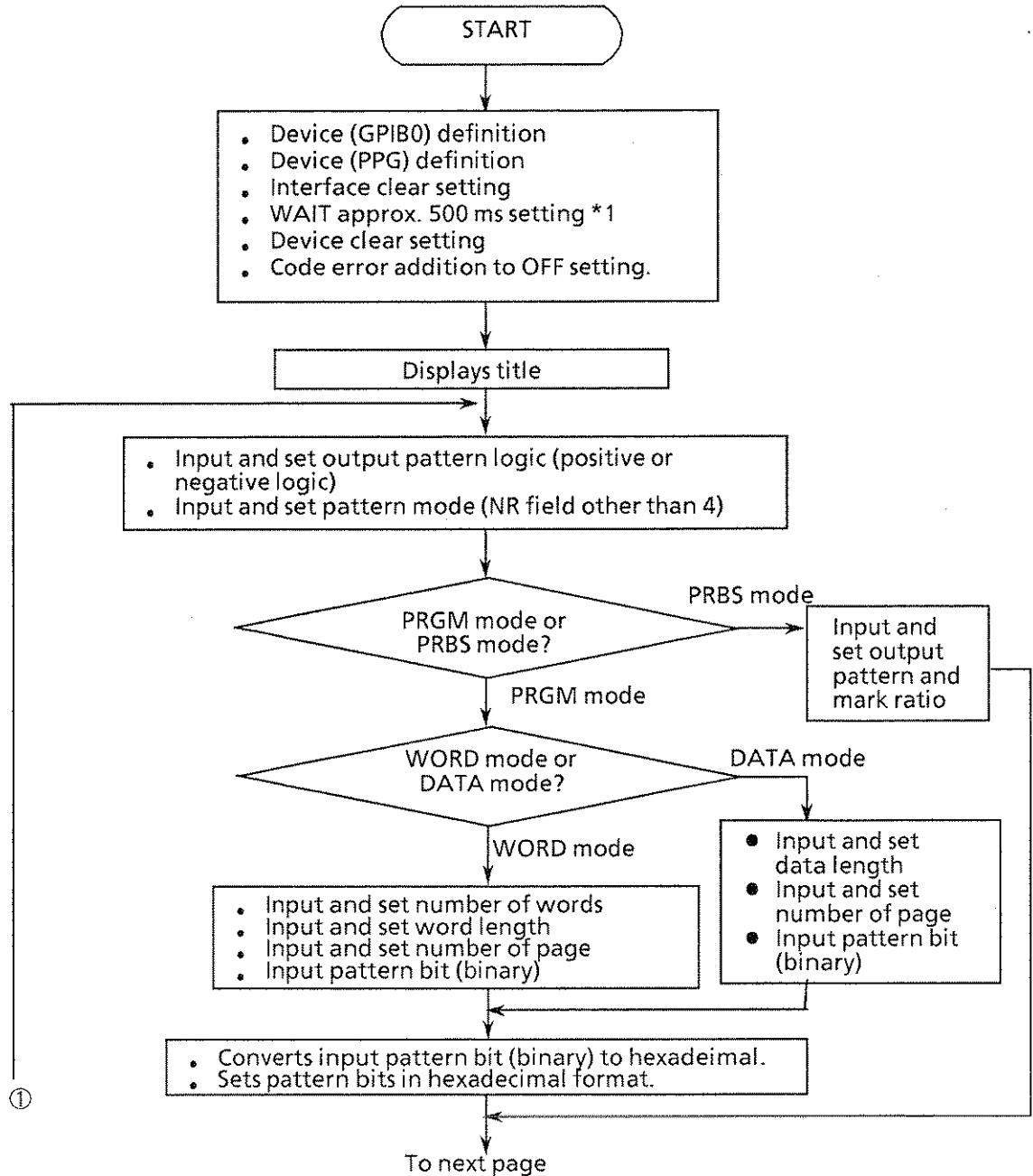
```

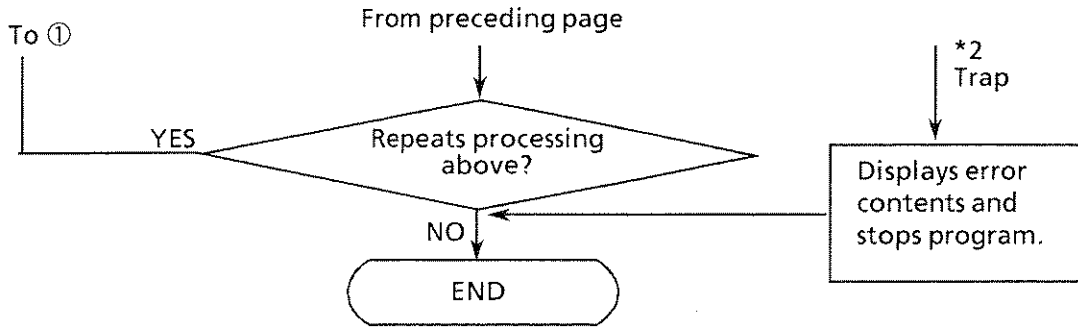
(3) Pattern setting

This program performs output pattern control.

It selects the output pattern logic, selects the PRGM or PRBS mode, and sets the necessary items of each output pattern for the MP1701B/MP1755A/MP1608A/MP1650A.

In the PRGM mode, the pattern data corresponding to the number of page input previously is set in a hexadecimal format.





Note: See the beginning of this paragraph 14.5 for *1 and *2.

PROGRAM LISTING

```

*****
*
*      MF1701B/MF1608A/MF1650A  PATTERN  SAMPLE  SOFT      *
*
*
*
*
*
*****

```

```

-----
MAIN  ROUTINE
-----

```

```

common shared IBSTA%,IBERR%,IBCNT%  ' Setup GPIB-PC functions
GOSUB gpinit                          ' Setup GPIB interface
wrt# = "EAD 0" : GOSUB wrtcmd         ' Error addition mode : OFF

```

```

DO
  CLS
  PRINT " *** MF1701B/MF1608A/MF1650A PATTERN SAMPLE SOFT ***"
  PRINT ""
  GOSUB pattern
  INPUT " NEXT DATA SET [ YES=0 , NO=1 ]";loop#
LOOP UNTIL loop# = "1"

```

```

STOP

```

```

-----
SUB ROUTINE
-----

```

```

pattern: ----- Set Logic,Pattern mode -----

```

```

INPUT " LOGIC MODE [ POSITIV=0 , NEGATIV=1 ]";lgc#
PRINT ""
WRT# = "LGC "+lgc# : GOSUB wrtcmd

```

```

DO
  PRINT " PATTERN MODE [ WORD=0, DATA=1, PN7=2, PN9=3, PN11=5"
  INPUT "                               PN15=6, PN20=7, PN23=8, PN31=9 ]";ptn#
  PRINT ""
  PRINT ""

```

```

LOOP UNTIL ptn# <> "4"

```

```

wrt# = "PTN "+ptn# : GOSUB wrtcmd

```

```

IF ptn# = "0" OR ptn# = "1" THEN

```

```

  GOSUB ProgMode

```

```

ELSE

```

```

  PRINT " MARK RATIO MODE [ 0/8:8/8 = 0, 1/8:7/8 = 1, "

```

```

  INPUT "                               1/4:3/4 = 2, 1/2:N1/2 = 3 ]";mrk#

```

```

  wrt# = "MRK "+mrk# : GOSUB wrtcmd

```

```

END IF

```

```

RETURN

```

ProgMode: ----- SET Prog(Word), (Data) -----

IF ptn# = "1" THEN

PRINT " *** PATTERN MODE PROG (DATA) ***"
PRINT ""

INPUT " DATA LENGTH DATA ";dln#
wrt# = "DLN "+dln# : GOSUB wrtcmd

INPUT " PAGE DATA ";pag#
wrt# = "PAG "+Pag# : GOSUB wrtcmd

INPUT " BIT PATTERN SET DATA BIT16-->BIT1 [0/1]";bit#
GOSUB BtoH
wrt# = "BIT #H"+B# : GOSUB wrtcmd

ELSE

PRINT " *** PATTERN MODE PROG (WORD) ***"
PRINT ""

INPUT " NUMBER OF WORD DATA ";wnb#
wrt# = "WNB "+wnb# : GOSUB wrtcmd

INPUT " WORD LENGTH DATA ";wln#
wrt# = "WLN "+wln# : GOSUB wrtcmd

INPUT " PAGE DATA ";pag#
wrt# = "PAG "+Pag# : GOSUB wrtcmd

INPUT " BIT PATTERN SET DATA BIT16-->BIT1 [0/1]";bit#
GOSUB BtoH
wrt# = "BIT #H"+B# : GOSUB wrtcmd

END IF

RETURN

gpinit: ----- Set up GP-IB functions -----

CALL IBFIND("GPIBO", GPIBO%) ' Open device (GPIBO)
IF GPIBO% < 0 THEN GOTO trap ' system error

CALL IBFIND("PPG", PPG%) ' Open device
IF PPG% < 0 THEN GOTO trap ' system error

CALL IBSIC(GPIBO%) ' Interface clear
IF IBSTAX% < 0 THEN GOTO trap ' system error

tim = 0.5
GOSUB waidly

CALL IBCLR(PPG%) ' Device clear

RETURN

BtoH: ----- Bin to Hex -----

```
A=0
FOR I=15 TO 0 STEP -1
  IF MID$(bit$,16-I,1) = "1" THEN
    A = A + 2^I
  END IF
NEXT I

B$ = HEX$(A)

RETURN
```

wrtcmd: ----- Write command -----

```
wrt$=wrt$+chr$(13)+chr$(10)
CALL IBWRT(PPG%, wrt$)           ' Write command
IF IBSTAZ < 0 THEN GOTO trap     ' Trap

RETURN
```

waidly: ----- Wait delay -----

```
stm = TIMER
etm = TIMER
WHILE etm - stm < tim
  etm = TIMER
  IF etm < stm THEN etm = etm + 86400
WEND

RETURN
```

trap: ----- System trap -----

```
PRINT "IBERR:" + STR$(IBERR%)
STOP

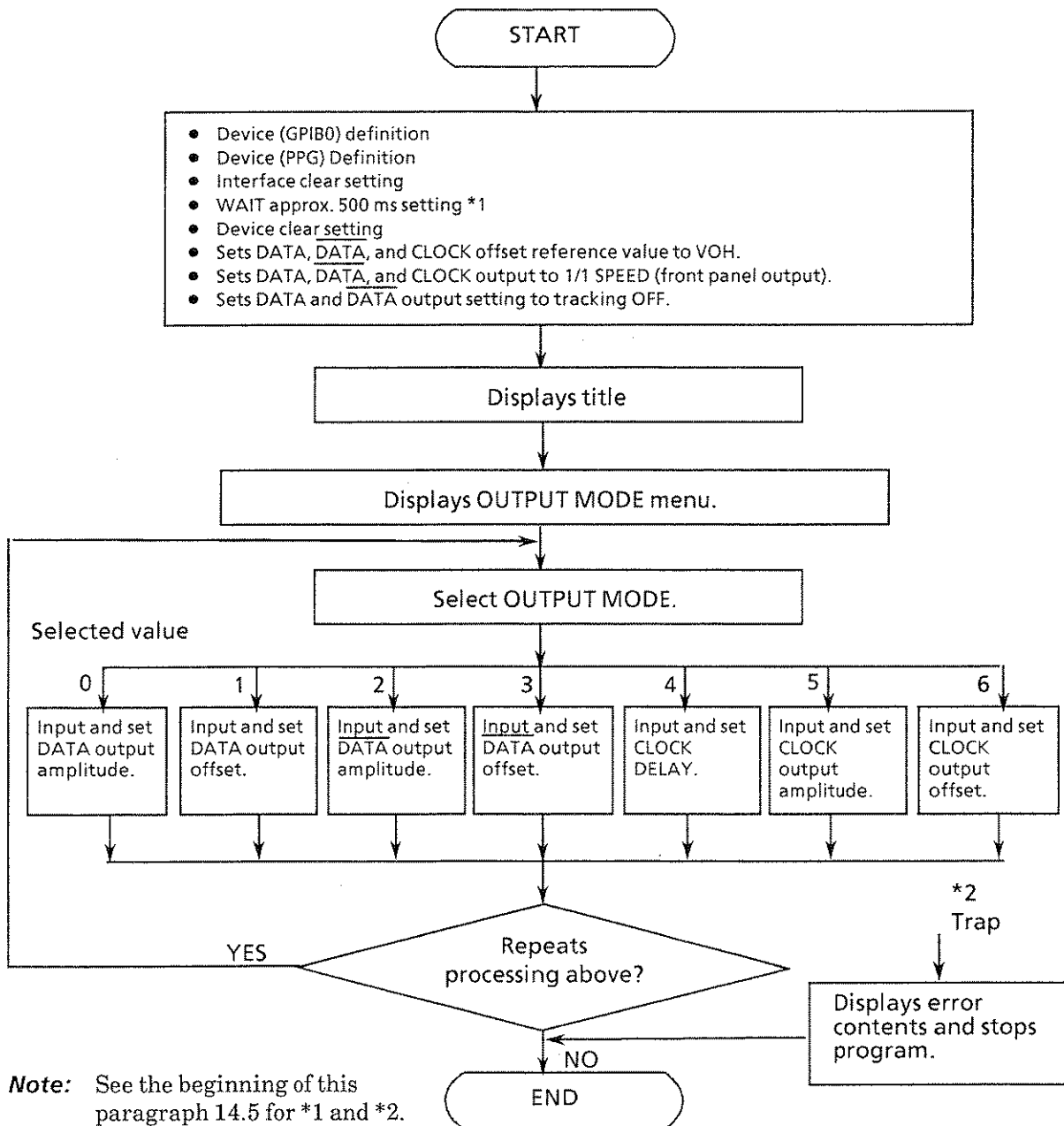
END
```

(4) Output signal setting

This program performs output signal control.

This selects and sets the necessary DATA, $\overline{\text{DATA}}$, and CLOCK output amplitudes and offsets.

The MP1701B/MP1755A/MP1608A/MP1650A is set to the offset reference value: VOH, 1/1 SPEED (front panel output), DATA/ $\overline{\text{DATA}}$ tracking: OFF.



PROGRAM LISTING

```

*****
*
*      MP1701B/MP1608A/MP1650A  OUTPUT SAMPLE SOFT_1      *
*      --- 1/1(Speed),Voh mode---                          *
*
*
*      OUTPUT
*
*****

common shared IBSTA%,IBERR%,IBCNT%      ' Setup GPIB-PC functions
GOSUB gpinit                            ' Setup GPIB interface
wrt# = "OFS 0" : GOSUB wrtcmd            ' Set Offset mode : Voh
wrt# = "SPD 0" : GOSUB wrtcmd            ' Set 1/1(speed)
wrt# = "TRK 0" : GOSUB wrtcmd            ' Set Tracking : off

DO
CLS
PRINT "**** MP1701B/MP1608A/MP1650A  OUTPUT SAMPLE SOFT ****"
PRINT "      --- 1/1(speed) , Voh mode ---"
PRINT ""
PRINT "      [ DATA AMPLITUDE=0, DATA OFF SET=1, NDATA AMPLITUDE=2 "
PRINT "      NDATA OFFSET=3,  CLOCK DELAY=4, CLOCK AMPLITUDE=5"
PRINT "      CLOCK OFFSET=6 ]"
PRINT ""
INPUT " MODE SELECT [ 0 or 1 or 2 or 3 or 4 or 5 or 6 ]";out#
PRINT ""

SELECT CASE out#

CASE "0"
INPUT "DATA AMPLITUDE [+0.50 ~ 2.00 V ] STEP 0.01 ";dap#
wrt# = "DAF " + dap# : GOSUB wrtcmd

CASE "1"
INPUT "DATA OFFSET [ -2.000 ~ 2.000 V ] STEP 0.005 ";dos#
wrt# = "DOS " + dos# : GOSUB wrtcmd

CASE "2"
INPUT "NDATA AMPLITUDE [ +0.50 ~ 2.00 V ] STEP 0.01 ";nap#
wrt# = "NAF " + nap# : GOSUB wrtcmd

CASE "3"
INPUT "NDATA OFFSET [ -2.000 ~ 2.000 V ] STEP 0.005 ";nos#
wrt# = "NOS " + nos# : GOSUB wrtcmd

CASE "4"
PRINT "      <<<  CLOCK DELAY  >>>"
PRINT "MP1701B,MP1608A --> -500 ~ 500 ps  1ps STEP"
PRINT "MP1650A -->      -1000 ~ 1000 ps  2ps STEP"
PRINT
INPUT "CLOCK DELAY ";cdl#
wrt# = "CDL " + cdl# : GOSUB wrtcmd

CASE "5"
INPUT "CLOCK AMPLITUDE [ +0.50 ~ 2.00 V ] STEP 0.01 ";cap#
wrt# = "CAP " + cap# : GOSUB wrtcmd

CASE "6"
INPUT "CLOCK OFFSET [ -2.000 ~ 2.000 V ] STEP 0.005 ";cos#
wrt# = "COS " + cos# : GOSUB wrtcmd

END SELECT

```

```

INPUT "NEXT DATA SET [ YES=0 , NO=1 ]";loop#
LOOP UNTIL loop# = "1"
STOP

gpinit:  ----- Setup GPIB interface -----
CALL IBFIND("GPIBO", GPIBO%)      ' Open device (GPIBO)
IF GPIBO% < 0 THEN GOTO trap      ' system error

CALL IBFIND("PPG", PPG%)          ' Open device (PPG)
IF PPG% < 0 THEN GOTO trap      ' system error

CALL IBSID(GPIBO%)                ' Interface clear
IF IBSTA% < 0 THEN GOTO trap    ' system error

tim = 0.5
GOSUB waidly

CALL IBCLR(PPG%)                  ' Device clear

RETURN

wrtcmd:  ----- Write command -----

wrt# = wrt# + chr$(13) + chr$(10)
CALL IBWRT(PPG%, wrt#)           ' Write command
IF IBSTA% < 0 THEN GOTO trap    ' Trap

RETURN

waidly:  ----- Wait delay -----

stm = TIMER
etm = TIMER
WHILE etm - stm < tim
  etm = TIMER
  IF etm < stm THEN etm = stm + 86400
WEND

RETURN

trap:  ----- System trap -----

PRINT "IBERR:" + STR$(IBERR%)
STOP

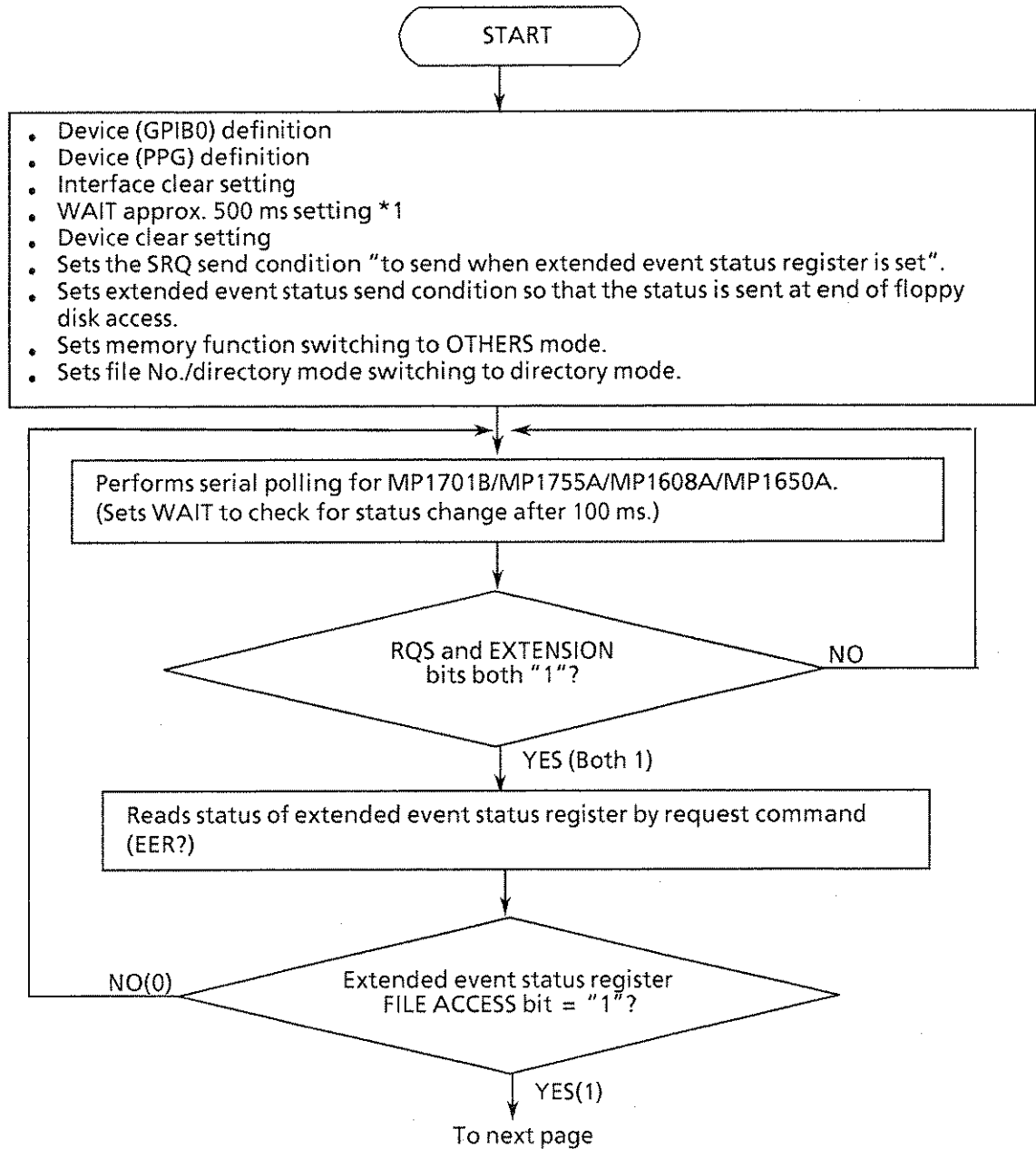
END

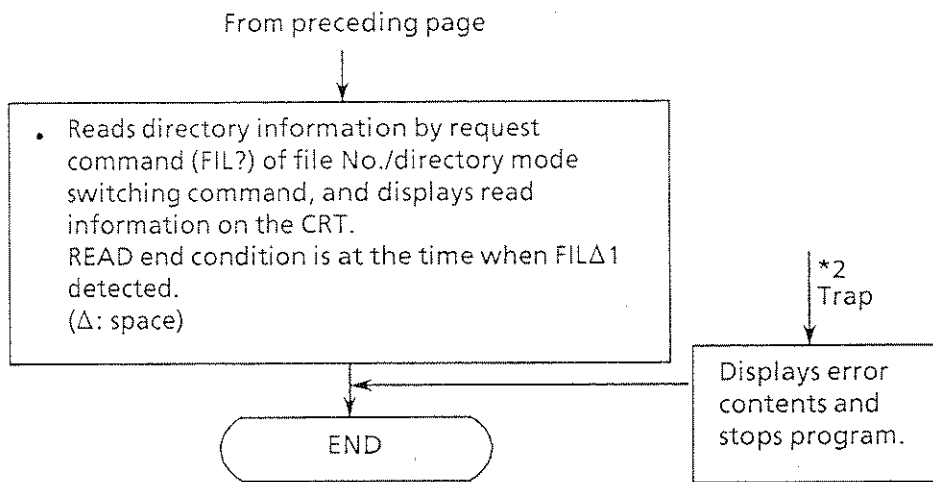
```

(5) Floppy disk file information read (Floppy disk access status check → serial polling)

This program checks the directory information of the files stored on the floppy disk, and displays it on the CRT. It also shows an example of actual output.

Floppy disk access status check is performed by serial polling and by data request command that checks the extended event status information.





Note: See the beginning of this paragraph 14.5 for *1 and *2.

EXECUTED RESULT

```

T02      ,PTN,      104,89-09-02,16:22
T04      ,PTN,      104,89-09-27,15:53
T07      ,OTH,      108,89-10-06,09:56
T21      ,PTN,      104,89-09-27,15:53
T23      ,PTN,      108,89-10-06,09:55
T24      ,OTH,      108,89-09-27,15:54
T38      ,PTN,      65640,89-09-29,15:35
T99      ,PTN,      65640,89-10-06,09:58
FIL 1
  
```

PROGRAM LISTING

```

*****
*
*           MF1701B / MF1608A / MF1650A
*           FILE DIRECTORY READ SAMPLE SOFT_1
*
*                                           F_DIR
*
*****

```

MAIN ROUTINE

```

common shared IBSTA%,IBERR%,IBCNT%      ' Setup GFIB-PC functions
GOSUB gpinit                             ' Setup GFIB interface

wrt# = "SRQ 2" : GOSUB wrtcmd            ' SRQ : Extension bit
wrt# = "EES 32" : GOSUB wrtcmd          ' Floppy access end
wrt# = "MEM 1" : GOSUB wrtcmd          ' Memory mode : OTHERS
wrt# = "FIL 1" : GOSUB wrtcmd          ' Directory mode

GOSUB SpollEer
GOSUB FileDir

STOP

```

SUB ROUTINE

SpollEer: ----- Check Status Byte -----

```

DO
  DO
    CALL IBRSP(PPG%,SPR%)      ' Send Serial poll
    IF IBSTA% < 0 THEN GOTO trap

    tim = 0.1 : GOSUB waidly

    LOOP UNTIL SPR%=66        ' RQS bit,Extension bit = 1

    wrt# = "EER?"
    GOSUB wrtcmd              ' REQUEST Extension Event Register ?
    GOSUB readcmd            ' READ Extension Event Register

    LOOP UNTIL MID$(rd#,9,1) = "1"    ' File Access bit = 1

  RETURN

```

FileDir: ----- File directory -----

```

wrt# = "FIL?" : GOSUB wrtcmd      ' REQUEST Directory ?

DO
  GOSUB readcmd                  ' READ Directory
  PRINT rd#
  LOOP UNTIL MID$(rd#,1,5) = "FIL 1"

RETURN

```

```

gpinit:  ----- Setup GPIB interface -----
        CALL IBFIND("GPIBO", GPIBO%)           ' Open device (GPIBO)
        IF GPIBO% < 0 THEN GOTO trap           ' system error

        CALL IBFIND("PPG", PPG%)             ' Open device(PPG)
        IF PPG% < 0 THEN GOTO trap           ' system error

        CALL IBSIC(GPIBO%)                   ' Interface clear
        IF IBSTA% < 0 THEN GOTO trap         ' system error

        tim = 0.5
        GOSUB waidly

        CALL IBCLR(PPG%)                     ' Device clear

        RETURN

wrtcmd:  ----- Write command -----
        wrt#=wrt#+chr$(13)+chr$(10)
        CALL IBWRT(PPG%, wrt#)              ' Write command
        IF IBSTA% < 0 THEN GOTO trap         ' Trap

        RETURN

readcmd: ----- Read command -----
        rd# = SPACE$(38)
        CALL IBRD(PPG%, rd#)                ' Read command
        IF IBSTA% < 0 THEN GOTO trap         ' Trap

        RETURN

waidly:  ----- Wait delay -----
        stm = TIMER
        etm = TIMER

        WHILE etm - stm < tim
            etm = TIMER
            IF etm < stm THEN etm = etm + 86400
        WEND

        RETURN

trap:   ----- System trap -----
        PRINT "IBERR%:" + STR$(IBERR%)
        STOP
        .
        .
        END

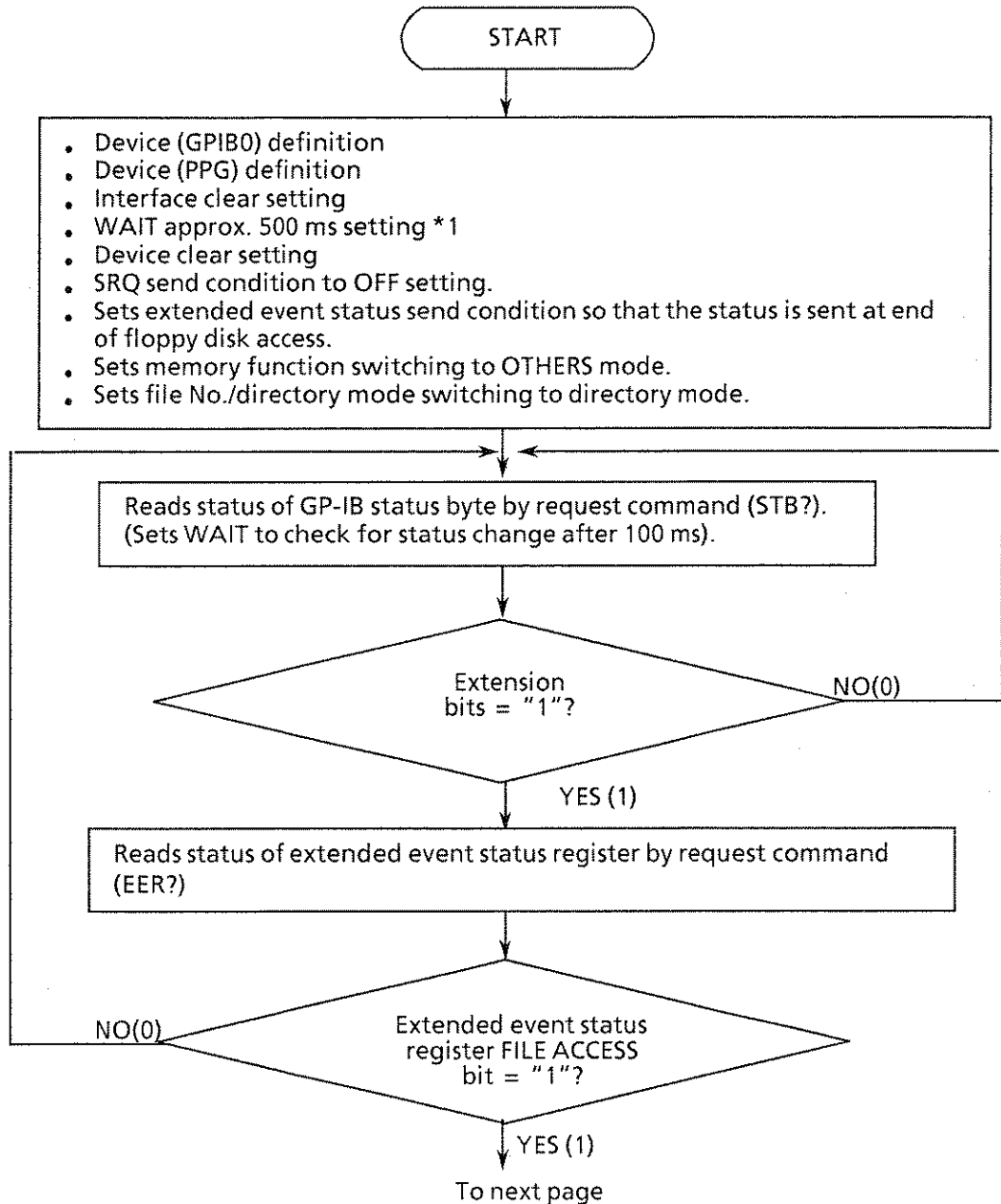
```

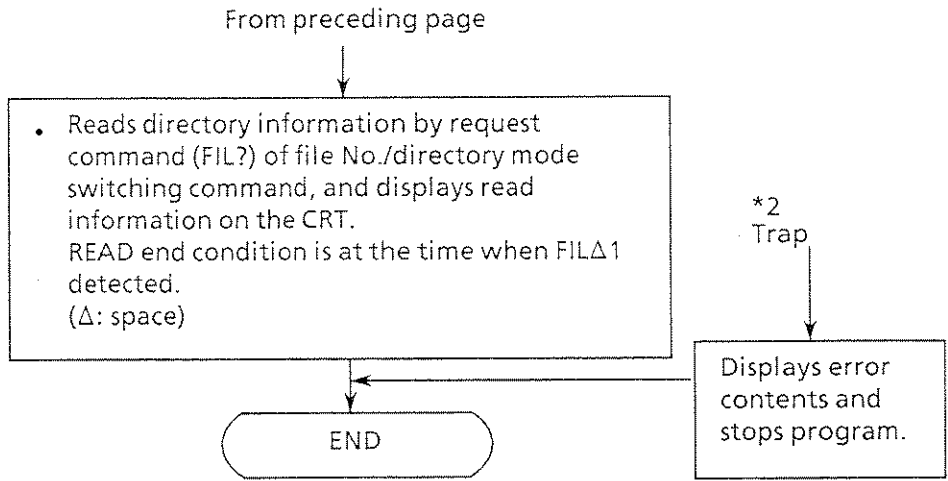
(6) Floppy disk file information read

(Floppy disk access status check → status byte register service request)

This program checks the directory information of the files stored on the floppy disk, and displays it on the CRT. It also shows an actual output example.

Floppy disk access status check is performed by data request command that checks the status of the GP-IB status byte and by data request command that checks the extended event status information.





Note: See the beginning of this paragraph 14.5 for *1 and *2.

EXECUTED RESULT

```

T02      ,PTN,      104,89-09-02,16:22
T04      ,PTN,      104,89-09-27,15:53
T07      ,OTH,      108,89-10-06,09:56
T21      ,PTN,      104,89-09-27,15:53
T23      ,PTN,      108,89-10-06,09:55
T24      ,OTH,      108,89-09-27,15:54
T38      ,PTN,      65640,89-09-29,15:35
T99      ,PTN,      65640,89-10-06,09:58
FIL 1
  
```

PROGRAM LISTING

```

*****
*
*           MP1701B / MP1608A / MP1650A
*           FILE DIRECTORY READ SAMPLE SOFT_2
*
*                               F_DIR2
*****

```

MAIN ROUTINE

```

common shared IBSTA%,IBERR%,IBCNT%      Set up GP-IB functions
GOSUB gpinit

```

```

wrt# = "SRQ 0" : GOSUB wrtcmd      ' SRQ : Off
wrt# = "EES 32" : GOSUB wrtcmd      ' EES : Floppy access end
wrt# = "MEM 1" : GOSUB wrtcmd      ' FIL : Memory mode : OTHERS
wrt# = "FIL 1" : GOSUB wrtcmd      ' FIL : Directory mode

```

```

GOSUB StbReg
GOSUB FileDir

```

```

STOP

```

SUB ROUTINE

StbReg: ----- Check Status Byte -----

```

DO
  DO
    wrt# = "STB?" : GOSUB wrtcmd      ' REQUEST Status Byte Register?
    GOSUB readcmd                      ' READ Status Byte Register
    tim = 0.1 : GOSUB waidly
    LOOP UNTIL MID$(rd#,13,1) = "1"    ' Extension bit = 1
    wrt# = "EER?" : GOSUB wrtcmd      ' REQUEST Extension Register?
    GOSUB readcmd                      ' READ Extension Register
    LOOP UNTIL MID$(rd#,9,1) = "1"    ' File Access bit = 1
  RETURN

```

FileDir: ----- File directory -----

```

wrt# = "FIL?" : GOSUB wrtcmd      ' REQUEST Directory?
DO
  GOSUB readcmd                      ' READ Directory
  PRINT rd#
  LOOP UNTIL MID$(rd#,1,5) = "FIL 1"  ' File Access bit = 1
RETURN

```

```

gpinit:  ----- Set up GP-IB functions -----
CALL IBFIND("GPIBO", GPIBO%)      ' Open device (GPIBO)
IF GPIBO% < 0 THEN GOTO trap      ' system error

CALL IBFIND("PPG", PPG%)          ' Open device (PPG)
IF PPG% < 0 THEN GOTO trap        ' system error

CALL IBSIC(GPIBO%)                ' Open device
IF IBSTA% < 0 THEN GOTO trap      ' system error

tim = 0.5
GOSUB waidly

CALL IBCLR(PPG%)                  ' Device clear

RETURN

```

```

wrtcmd:  ----- Write command -----
wrt$=wrt$+chr$(13)+chr$(10)
CALL IBWRT(PPG%, wrt$)            ' Write command
IF IBSTA% < 0 THEN GOTO trap      ' Trap

RETURN

```

```

readcmd: ----- Read command -----
rd$=SPACE$(38)
CALL IBRD(PPG%, rd$)             ' Read command
IF IBSTA% < 0 THEN GOTO trap      ' Trap

RETURN

```

```

waidly:  ----- Wait delay -----
stm = TIMER
etm = TIMER

WHILE etm - stm < tim
  etm = TIMER
  IF etm < stm THEN etm = etm + 86400
WEND

RETURN

```

```

trap:  ----- System trap -----
PRINT "IBERR:" + STR$(IBERR%)
STOP

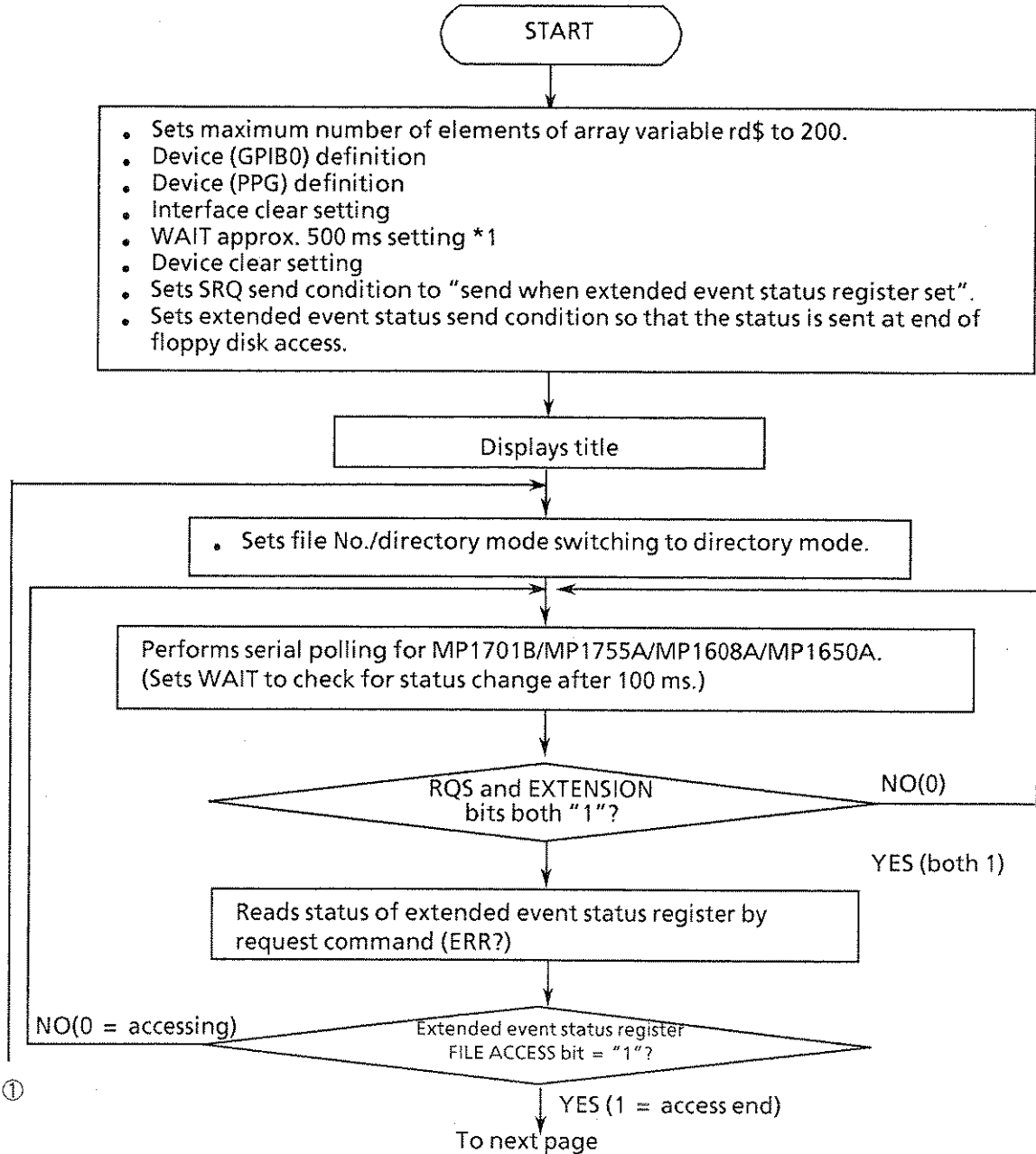
END

```

(7) Data save, resave, and recall

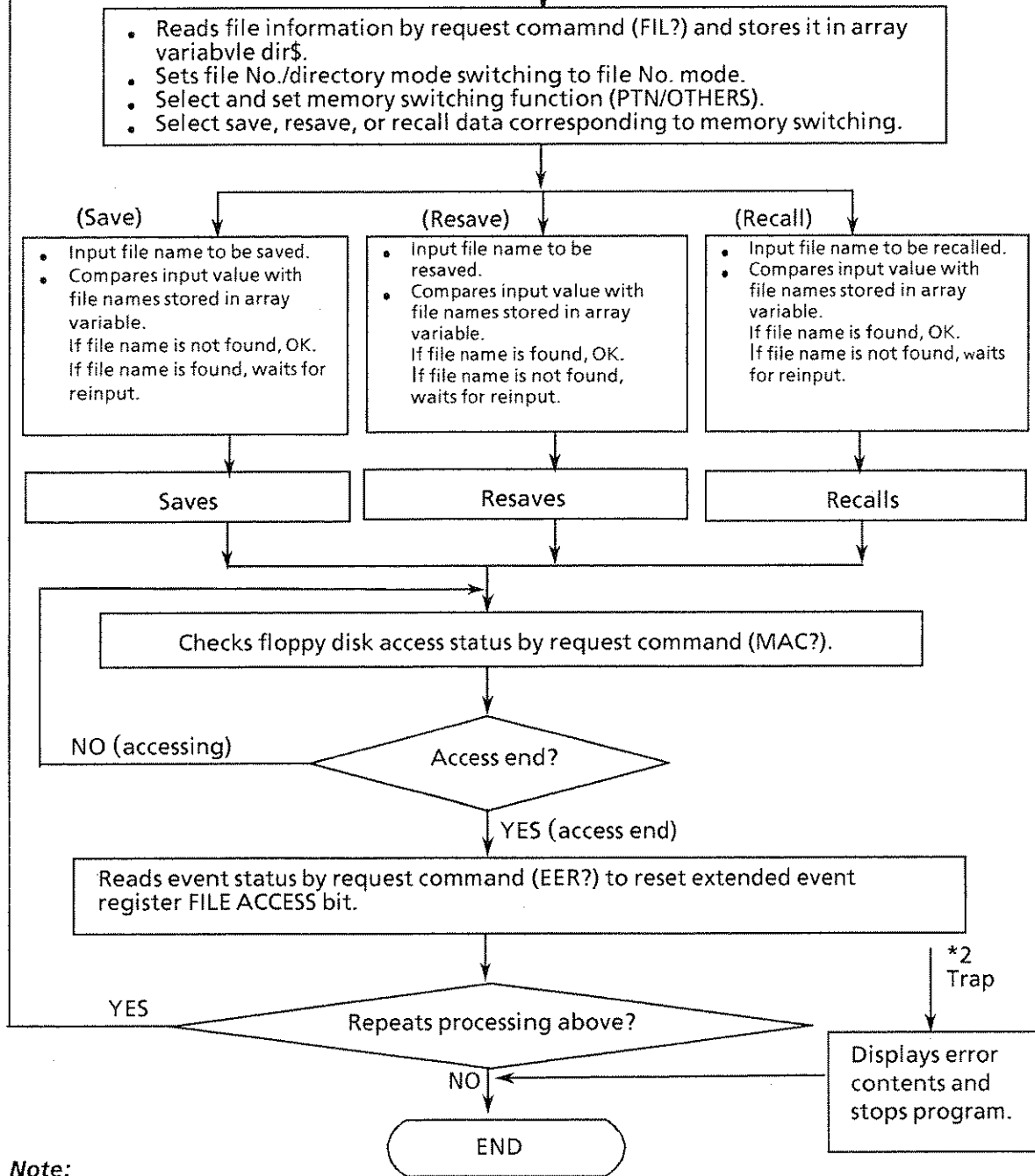
This program checks the directory information of the files stored on floppy disk, saves or resaves the statuses set at the MP1701B/MP1755A/MP1608A/MP1650A corresponding to the memory switching function (PTN/OTHERS), and recalls the file stored on floppy disk.

The access status of the floppy disk to obtain the directory information is checked by serial polling and by data request command to check the extended event status. At save, resave, and recall, the floppy disk access status is checked by request command MAC?



To ①

From preceding page



Note:

See the beginning of this paragraph 14.5 for *1 and *2.

PROGRAM LISTING

```

*****
*
*      MP1701B/MP1608A/MP1650A  MEMORY  SAMPLE SOFT
*
*
*      MEMORY
*****

-----
MAIN  ROUTINE
-----

common shared IBSTA%,IBERR%,IBCNT%  ' Setup GPIB-PC  functions
GOSUB gpinit                        ' Setup GPIB interface

DIM dir$(199)
wrt$ = "SRQ 2" : GOSUB wrtcmd        ' SRQ : Extension bit
wrt$ = "EES 32" : GOSUB wrtcmd      ' Floppy access end

PRINT " *** MP1701B/MP1608A/MP1650A  MEMORY  SAMPLE SOFT "
PRINT
DO

    GOSUB SPoll : GOSUB DSet : GOSUB Floppy

    INPUT " NEXT DATA SET [ YES=0 , NO=1 ] ";loop#
    PRINT

LOOP UNTIL loop#="1"

STOP

-----
SUB  ROUTINE
-----

SPoll:  ////////////////////////////////////////////////////////////////////
//
//      Serial polling
//      RQS,Extension bit = 1 , File access bit = 1
//
////////////////////////////////////////////////////////////////////

wrt$="FIL 1" : GOSUB wrtcmd          ' Directory mode

DO
    DO
        CALL IBRSP (PPG%,SPR%)      ' Serial polling
        IF IBSTA < 0 THEN GOTO TRAP

        tim = 0.1 : GOSUB waidly

    LOOP UNTIL SPR%=66              ' RQS,Extension bit = 1

    wrt$="EER?" : GOSUB wrtcmd      ' REQUEST Extension event register
    GOSUB readcmd                  ' READ Extension event register

LOOP UNTIL MID$(rd$,9,1)="1"      ' File access bit = 1

RETURN

```

```

DSet:  ////////////////////////////////////////////////////////////////////
//
//   Read file directory , Set Memory mode &
//           Select SAVE or RESAVE or RECALL
//
////////////////////////////////////////////////////////////////////
.
----- Read File directory
.
I=0
tim=0.3 : GOSUB waidly
.
wrt$="FIL?" : GOSUB wrtcmd           ' REQUEST File directory
.
DO
  GOSUB readcmd                     ' READ File directory
  dir$(I)=MID$(rd$,1,12)
  I=I+1
LOOP UNTIL MID$(rd$,1,5)="FIL 1"    ' File end
.
----- Set Memory mode
.
wrt$="FIL 0" : GOSUB wrtcmd
.
DO
  INPUT " MEMORY MODE SELECT [ PTN=0 , OTHERS=1 ] ";mem
LOOP UNTIL mem=0 OR mem=1
.
wrt$="MEM "+STR$(mem) : GOSUB wrtcmd
.
IF mem=0 THEN mem$="PTN"
IF mem=1 THEN mem$="OTH"
.
----- Select SAVE or RESAVE or RECALL
.
DO
  INPUT " SELECT [ SAVE=0 , RESAVE=1 , RECALL=2 ]";dta
LOOP UNTIL dta=0 OR dta=1 OR dta=2
.
IF dta=0 THEN GOSUB DSave
IF dta=1 THEN GOSUB DResave
IF dta=2 THEN GOSUB DRecall
.
RETURN
.
DSave: ////////////////////////////////////////////////////////////////////
//
//   * Data Save *
//           SAME Memory mode & File name --> ERROR !
//
////////////////////////////////////////////////////////////////////
.
DO
  I=0
  result$=SPACE$(4)
  INPUT " ** Data Save ** File Name [ 0^99 ] ";nam
.
  DO
    IF mem$=MID$(dir$(I),10,3) AND nam=VAL(MID$(dir$(I),2,2)) THEN
      result$="SAME"
    END IF
.
    I=I+1
  DO

```

```

        LOOP UNTIL MID$(dir$(I),1,5)="FIL 1"
    LOOP UNTIL result$ <> "SAME"
    wrt$="SAV "+STR$(nam) : GOSUB wrtcmd
    RETURN
DResave: ////////////////////////////////////////////////////////////////////
//
//      * Data Resave *
//              SAME Memory mode & File name --> OK !
//
////////////////////////////////////////////////////////////////////
DO
    I=0
    result$=SPACE$(4)
    INPUT " ** Data Resave ** File Name [ 0~99 ] ";nam
    DO
        IF mem$=MID$(dir$(I),10,3) AND nam=VAL(MID$(dir$(I),2,2)) THEN
            result$="SAME"
        END IF
        I=I+1
        LOOP UNTIL MID$(dir$(I),1,5)="FIL 1"
    LOOP UNTIL result$ = "SAME"
    wrt$="RSV "+STR$(nam) : GOSUB wrtcmd
    RETURN
DRecall: ////////////////////////////////////////////////////////////////////
//
//      * Data Recall *
//              SAME Memory mode & File name --> OK !
//
////////////////////////////////////////////////////////////////////
DO
    I=0
    result$=SPACE$(4)
    INPUT " ** Data Recall ** File Name [ 0~99 ] ";nam
    DO
        IF mem$=MID$(dir$(I),10,3) AND nam=VAL(MID$(dir$(I),2,2)) THEN
            result$="SAME"
        END IF
        I=I+1
        LOOP UNTIL MID$(dir$(I),1,5)="FIL 1"
    LOOP UNTIL result$ = "SAME"
    wrt$="RCL "+STR$(nam) : GOSUB wrtcmd
    RETURN

```

```

Floppy: ////////////////////////////////////////////////////////////////////
//
//      Memory Access Condition ?
//      & Reset FILE ACCESS bit
//
//////////////////////////////////////////////////////////////////
DO
  wrt#="MAC?" : GOSUB wrtcmd : GOSUB readcmd
LOOP UNTIL MID$(rd#,1,5)="MAC 0"

wrt#="EER?" : GOSUB wrtcmd : GOSUB readcmd
RETURN

gpinit:  ----- Setup GPIB interface -----
CALL IBFIND("GPIBO", GPIBO%)      ' Open device (GPIBO)
IF GPIBO% < 0 THEN GOTO trap      ' system error

CALL IBFIND("PPG", PPG%)          ' Open device (PPG)
IF PPG% < 0 THEN GOTO trap      ' system error

CALL IBSIC(GPIBO%)               ' Interface clear
IF IBSTA% < 0 THEN GOTO trap    ' system error

tim = 0.5
GOSUB waidly

CALL IBCLR(PPG%)                 ' Device clear

RETURN

wrtcmd:  ----- Write command -----
wrt#=wrt#+chr$(13)+chr$(10)
CALL IBWRT(PPG%, wrt#)           ' Write command
IF IBSTA% < 0 THEN GOTO trap    ' Trap

RETURN

readcmd: ----- Read command -----
rd#=SPACE$(38)
CALL IBRD(PPG%, rd#)            ' Read command
IF IBSTA% < 0 THEN GOTO trap    ' Trap

RETURN

waidly:  ----- Wait delay -----
stm = TIMER
etm = TIMER

WHILE etm - stm < tim
  etm = TIMER
  IF etm < stm THEN etm = etm + 86400
WEND

RETURN

trap:   ----- System trap -----
PRINT "IBERR:" + STR$(IBERR%)
STOP

END

```

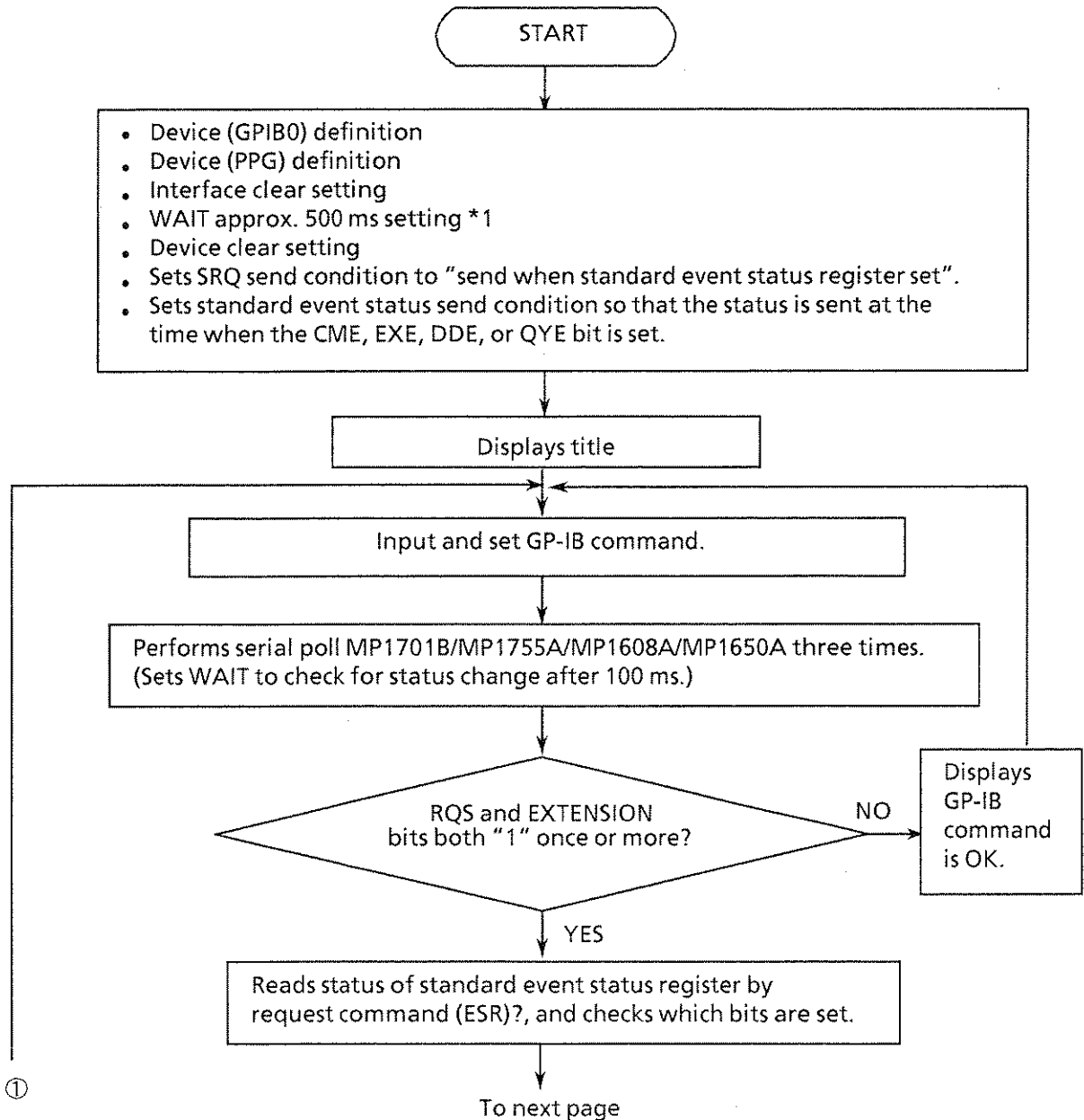
(8) Standard event status byte check

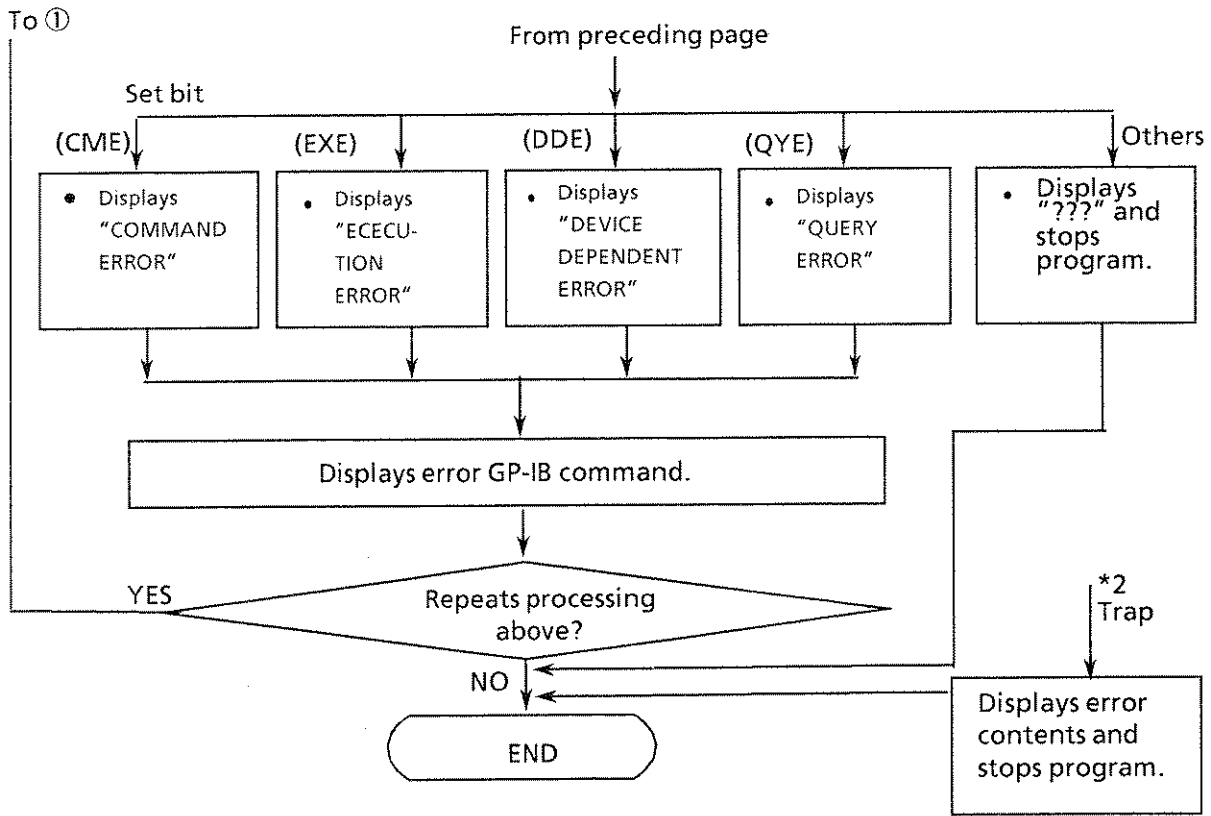
(COMMAND ERROR bit check → serial polling)

This program checks the CME, EXE, DDE, and QYE bits of the standard event status byte, and displays whether or not the input GP-IB command is normal on the CRT.

When the command is abnormal, the cause of the error at that time is displayed.

The GP-IB status byte is checked by serial polling and by the status of the standard event status register which is obtained by data request command.





Note:

See the beginning of this paragraph 14.5 for *1 and *2.

PROGRAM LISTING

```

*****
*
*           MF1701B / MF1608A / MF1650A
*   STANDERD EVENT STATUS REGISTER CHECK  SAMPLE SOFT
*
*                                           ESE
*****

-----
MAIN ROUTINE
-----

common shared IBSTA%,IBERR%,IBCNT%      ' Setup GPIB-PC functions
GOSUB gpinit                             ' Setup GPIB interface

wrt# = "SRQ 32" : GOSUB wrtcmd           ' SRQ : Command error bit
wrt# = "ESE 60" : GOSUB wrtcmd         ' CME,EXE,DDE,QYE bit

PRINT "* MF1701B/MF1608A/MF1650A STANDERD STATUS REGISTER CHECK *"
PRINT

DO
  INPUT " INPUT ANY GP-IB COMMAND,Please.",com#

  wrt# =com# : GOSUB wrtcmd

  GOSUB Check

  INPUT " NEXT COMMAND SET? [ YES=0 , NO=1 ]";loop#
  PRINT

LOOP UNTIL loop#="1"

STOP

-----
SUB ROUTINE
-----

Check:  ----- Serial polling -----

byt=0
FOR I=0 TO 2
  CALL IBRSP(PPG%,SPR%)                ' Serial polling
  IF IBSTA% < 0 THEN GOTO trap

  tim = 0.1 : GOSUB waidly

  IF SPR%=96 THEN byt=SPR%              ' RQS,Comand Error bit=1
NEXT I

IF byt=96 THEN
  GOSUB Ero
ELSE
  PRINT " GP-IB COMMND IS OK !"
END IF

RETURN

```


Ero: ----- ERROR -----

```
wrt# = "ESR?" : GOSUB wrtcmd     ' REQUEST standard event status?
GOSUB readcmd                 ' READ standard event status
```

```
IF MID$(rd#,9,1)="1" THEN PRINT "* COMMAND ERROR !!"
IF MID$(rd#,10,1)="1" THEN PRINT "* EXECUTION ERROR !!"
IF MID$(rd#,11,1)="1" THEN PRINT "* DEVICE DEPENDENT ERROR !!"
IF MID$(rd#,12,1)="1" THEN PRINT "* QUERY ERROR !!"
PRINT " INPUT COMMAND = "+com#
```

```
IF MID$(rd#,7,1)="1" OR MID$(rd#,8,1)="1" OR MID$(rd#,13,1)="1" OR MID$(rd#,
14,1)="1" THEN GOSUB Bug
```

```
RETURN
```

gpinit: ----- Setup GPIB interface -----

```
CALL IBFIND("GPIBO", GPIBO%)     ' Open device (GPIBO)
IF GPIBO% < 0 THEN GOTO trap     ' system error
```

```
CALL IBFIND("PPG", PPG%)         ' Open device(PPG)
IF PPG% < 0 THEN GOTO trap       ' system error
```

```
CALL IBSID(GPIBO%)               ' Interface clear
IF IBSTA% < 0 THEN GOTO trap     ' system error
```

```
tim = 0.5
GOSUB waidly
```

```
CALL IBCLR(PPG%)                 ' Device clear
```

```
RETURN
```

wrtcmd: ----- Write command -----

```
wrt#=wrt#+chr$(13)+chr$(10)
CALL IBWRT(PPG%, wrt#)           ' Write command
IF IBSTA% < 0 THEN GOTO trap     ' Trap
```

```
RETURN
```

readcmd: ----- Read command -----

```
rd#=SPACE$(16)
CALL IBRD(PPG%, rd#)             ' Read command
IF IBSTA% < 0 THEN GOTO trap     ' Trap
```

```
RETURN
```

waidly: ----- Wait delay -----

```
etm = TIMER
etm = TIMER
```

```
WHILE etm - stm < tim
  etm = TIMER
  IF etm < stm THEN etm = etm + 86400
WEND
```

```
RETURN
```

Bug: / ----- BUG -----

PRINT "???"

STOP

trap: / ----- System trap -----

PRINT "IBERR%:" + STR\$(IBERR%)

STOP

.

END

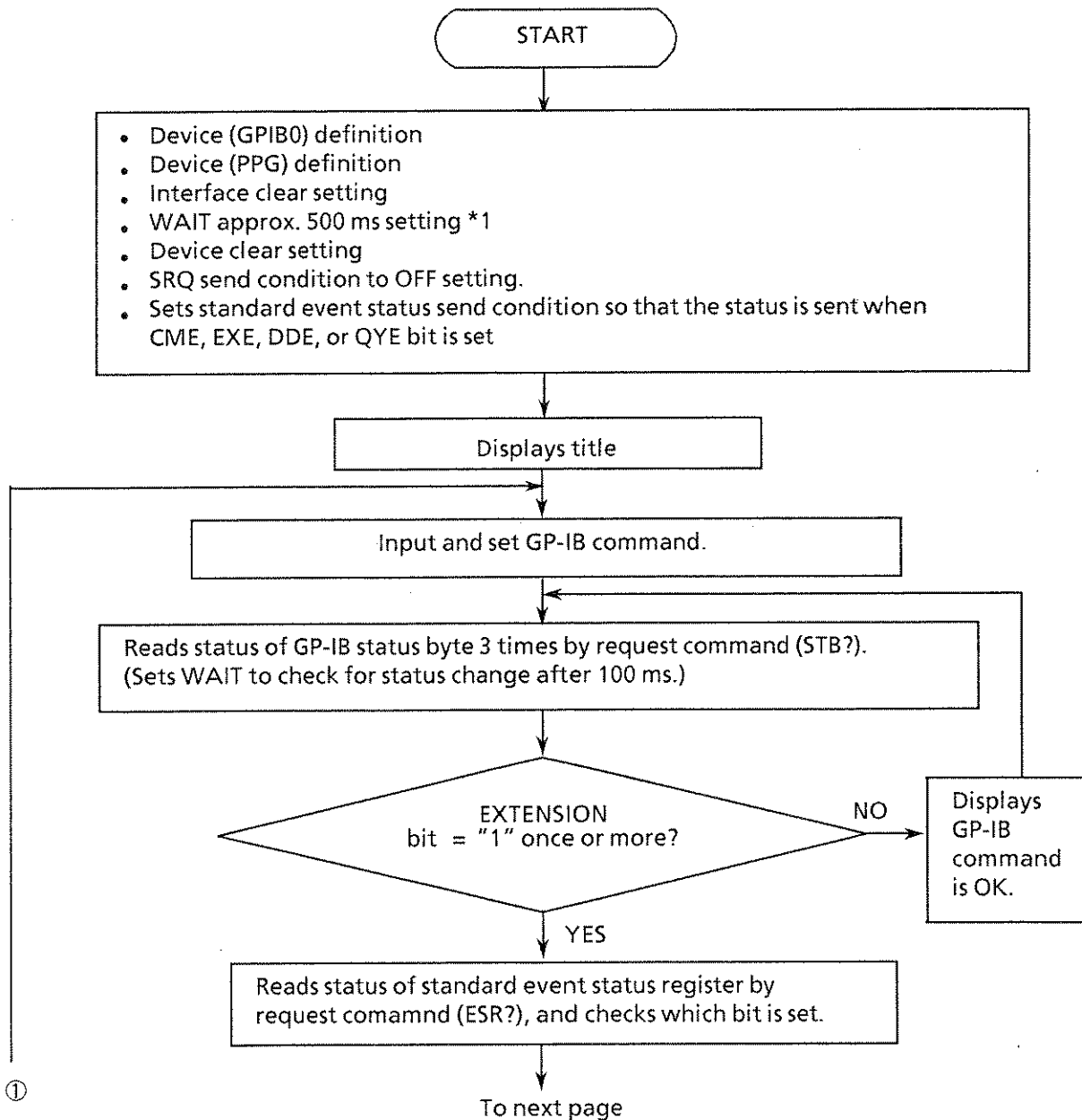
(9) Standard event status byte check

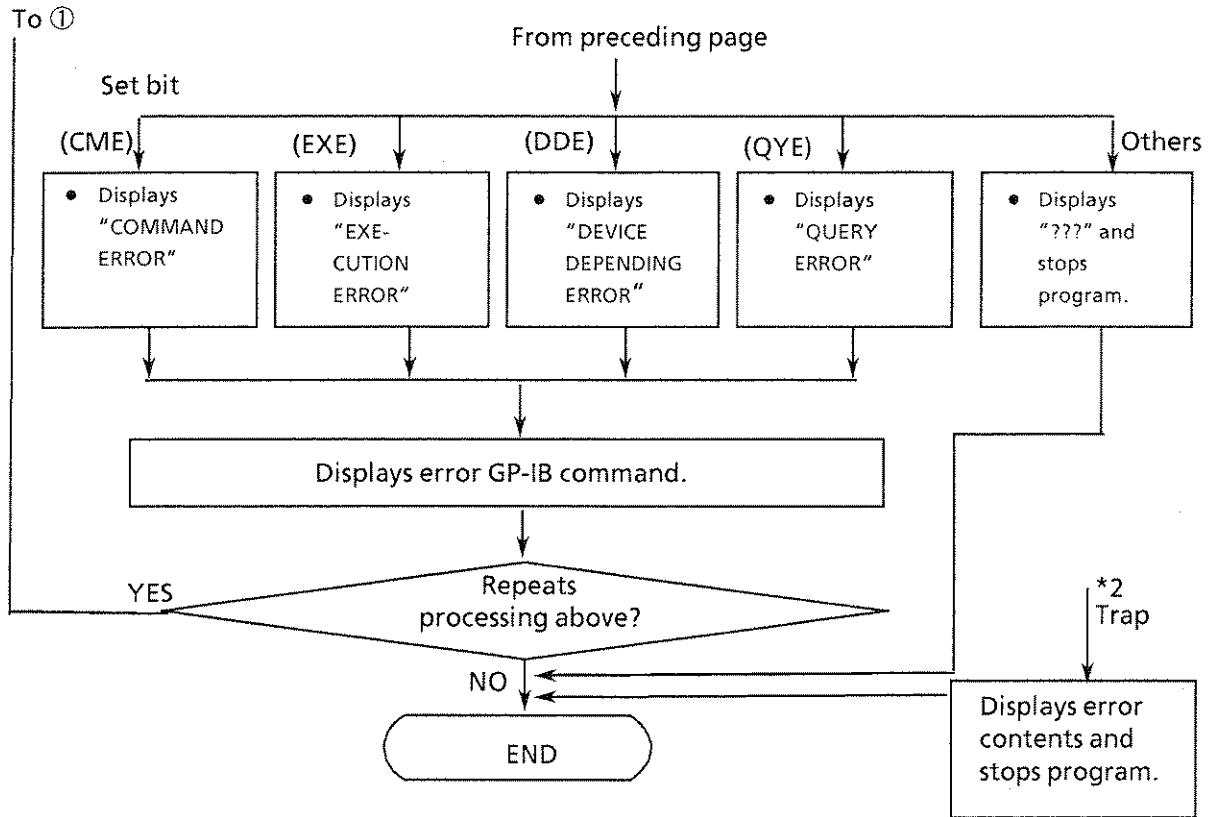
(COMMAND ERROR bit check → status byte register service request)

This program checks the CME, EXE, DDE, and QYE bits of the standard event status byte, and displays whether or not the input GP-IB command is normal on the CRT.

When the command is abnormal, the cause of the error at that time is displayed.

The GP-IB status byte and the status of the standard event status register are checked by data request command.





Note:

See the beginning of this paragraph 14.5 for *1 and *2.

PROGRAM LISTING

```

/ *****
/ *
/ *          MP1701B / MP1608A / MP1650A          *
/ *          STANDERD EVENT STATUS REGISTER CHECK  SAMPLE SOFT *
/ *                                          ESE2          *
/ *****
/-----/
/              MAIN  ROUTINE
/-----/
/
common shared IBSTA%,IBERR%,IBCNT%      ' Setup GPIB-PC functions
GOSUB gpinit                            ' Setup GPIB interface

wrt# = "SRQ 0 " : GOSUB wrtcmd           ' SRQ :OFF
wrt# = "ESE 60" : GOSUB wrtcmd         ' CME,EXE,DDE,0YE bit

PRINT "* MP1701B/MP1608A/MP1650A STANDERD STATUS REGISTER CHECK *"
PRINT
DO
  INPUT " INPUT ANY GP-IB COMMAND,Please.",com#
  wrt# =com# : GOSUB wrtcmd
  GOSUB Check
  INPUT " NEXT COMMAND SET? [ YES=0 , NO=1 ] ";loop#
  PRINT
LOOP UNTIL loop#="1"
STOP
/-----/
/              SUB  ROUTINE
/-----/

```

Check: ----- Check Status byte -----

```

bytf=SPACE$(16)
FOR I=0 TO 2
  wrt#="STB?" : GOSUB wrtcmd      ' REQUEST Status byte register?
  GOSUB readcmd                 ' READ Status byte register
  IF MID$(rd#,9,1)="1" THEN bytf=rd# ' Command error bit = 1
  tim=0.1 : GOSUB waidly
NEXT I
IF MID$(bytf,9,1)="1" THEN
  GOSUB Ero
ELSE
  PRINT " GP-IB COMMAND IS OK !"
END IF
RETURN

```

```

Ero: ----- ERROR -----

wrt# = "ESR?" : GOSUB wrtcmd      ' REQUEST Standard event status?
GOSUB readcmd      ' READ Standard event status

IF MID$(rd#,9,1)="1" THEN PRINT "* COMMAND ERROR !! "
IF MID$(rd#,10,1)="1" THEN PRINT "* EXECUTION ERROR !! "
IF MID$(rd#,11,1)="1" THEN PRINT "* DEVICE DEPENDENT ERROR !! "
IF MID$(rd#,12,1)="1" THEN PRINT "* QUERY ERROR !! "
PRINT " INPUT COMMAND = "+com#

IF MID$(rd#,7,1)="1" OR MID$(rd#,8,1)="1" OR MID$(rd#,13,1)="1" OR MID$(rd#,
14,1)="1" THEN GOSUB Bug

RETURN

gpinit: ----- Setup GPIB interface -----

CALL IBFIND("GPIBO", GPIBO%)      ' Open device (GPIBO)
IF GPIBO% < 0 THEN GOTO trap      ' system error

CALL IBFIND("PPG", PPG%)          ' Open device (PPG)
IF PPG% < 0 THEN GOTO trap      ' system error

CALL IBSIC(GPIBO%)              ' Interface clear
IF IBSTA% < 0 THEN GOTO trap      ' system error

tim = 0.5
GOSUB waidly

CALL IBCLR(PPG%)                ' Device clear

RETURN

wrtcmd: ----- Write command -----

wrt# = wrt# + chr$(13) + chr$(10)
CALL IBWRT(PPG%, wrt#)          ' Write command
IF IBSTA% < 0 THEN GOTO trap      ' Trap

RETURN

readcmd: ----- Read command -----

rd# = SPACE$(16)
CALL IBRD(PPG%, rd#)           ' Read command
IF IBSTA% < 0 THEN GOTO trap      ' Trap

RETURN

waidly: ----- Wait delay -----

stm = TIMER
etm = TIMER

WHILE etm - stm < tim
  etm = TIMER
  IF etm < stm THEN etm = etm + 86400
WEND

RETURN

```

```
Bug: / ----- BUG -----
```

```
FRINT "???"
```

```
STOP
```

```
trap: / ----- System trap -----
```

```
FRINT "IBERR:" + STR#(IBERR#)
```

```
STOP
```

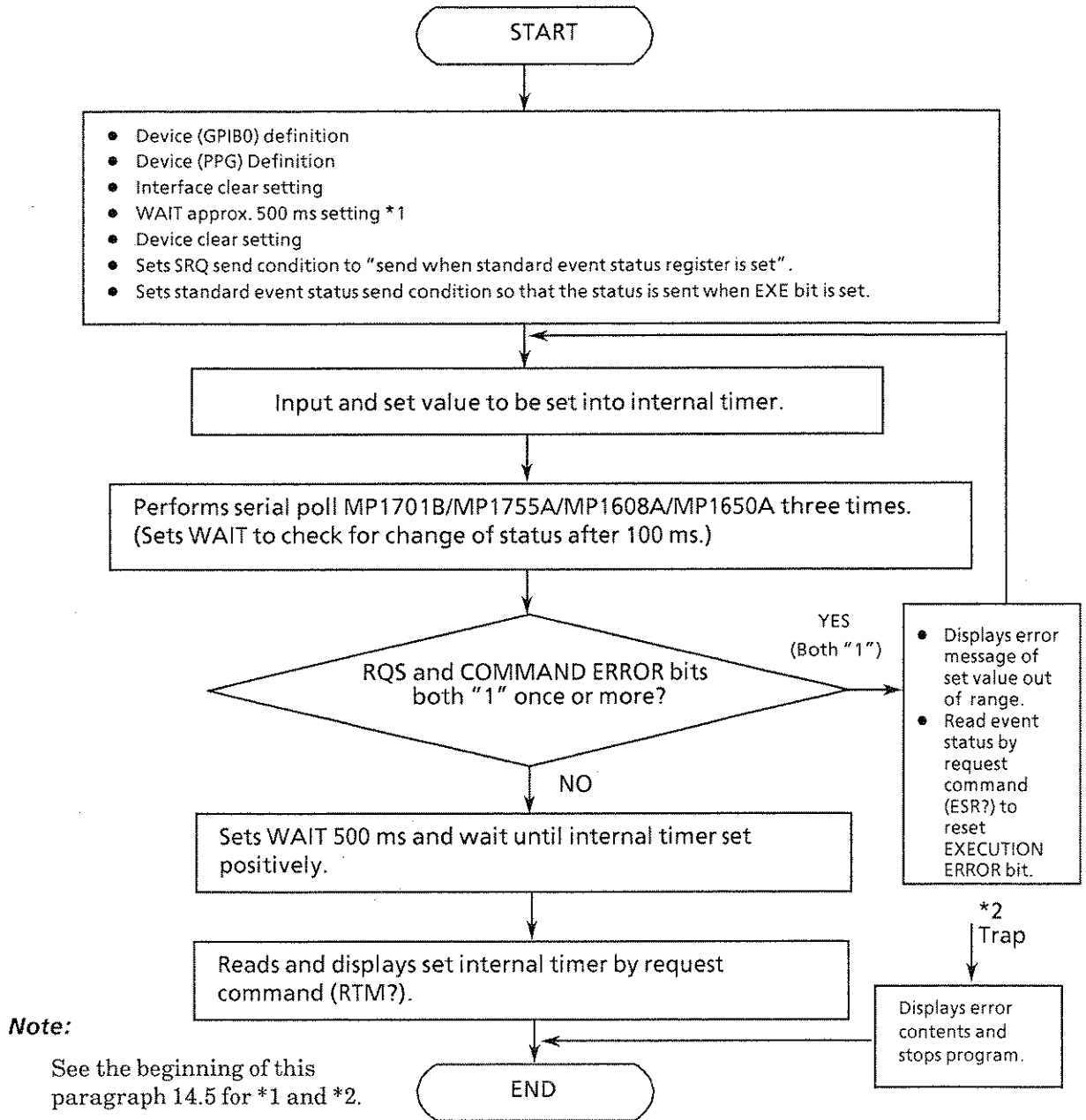
```
END
```

(10) Internal timer setting

This program sets the internal timer, reads its value by data request command, and displays it on the CRT.

Input value is checked by the standard event status byte.

The MP1701B/MP1755A/MP1608A/MP1650A needs several ms to set and start the timer positively. This program sets a WAIT of approximately 500 ms for this purpose.



PROGRAM LISTING

```

*****
*
*           MF1701B / MF1608A / MF1650A
*           REAL TIME SETTING SAMPLE SOFT
*
*                                           TIME
*****

```

```

-----
MAIN ROUTINE
-----

```

```

common shared IBSTAZ,IBERR%,IBCNT%      ' Setup GPIB-PC functions
GOSUB gpinit                             ' Setup GPIB interface

wrt# = "SRQ 32" : GOSUB wrtcmd           ' SRQ : Command error bit
wrt# = "ESE 16" : GOSUB wrtcmd           ' Extution error bit

GOSUB DSet
STOP

```

```

-----
SUB ROUTINE
-----

```

DSet: ----- Data Check & Display real time -----

```

DO
CLS
INPUT "Real time setting data * YEAR * [ 0~99 ] ";Y#
INPUT "Real time setting data * MONTH * [ 1~12 ] ";M#
INPUT "Real time setting data * DAY * [ 1~31 ] ";D#
INPUT "Real time setting data * HOURS * [ 0~23 ] ";H#
INPUT "Real time setting data * MINUTE * [ 0~59 ] ";Mi#
INPUT "Real time setting data * SECOND * [ 0~59 ] ";S#
PRINT

rtm#="Y#"+","+M#"+","+D#"+","+H#"+","+Mi#"+","+S#
wrt#="RTM "+rtm# : GOSUB wrtcmd

GOSUB check

LOOP UNTIL Result#="OK"

tim = 0.5 : GOSUB waidly

wrt#="RTM?": GOSUB wrtcmd
GOSUB readcmd : PRINT rd#

RETURN

```

check: ----- Check Set data -----

```

byt=0

FOR I=0 TO 2

CALL IBRSF(PPG%,SPR%)      ' Serial polling
IF IBSTAZ < 0 THEN GOTO trap

tim = 0.1 : GOSUB waidly

```

```

      IF SPR%=96 THEN byt=SPR%          ' RQS,Comand Error bit=1
NEXT I
IF byt=96 THEN
  INPUT " EXECUTION ERROR !! Press ENTER key ";A
  wrt#="ESR?" : GOSUB wrtcmd          ' Reset Execution error bit
ELSE
  Result#="OK"
END IF
RETURN

gpinit:  ' ----- Setup GPIB interface -----

CALL IBFIND("GPIBO", GPIBO%)          ' Open device (GPIBO)
IF GPIBO% < 0 THEN GOTO trap          ' system error

CALL IBFIND("PPG", PPG%)              ' Open device(PPG)
IF PPG% < 0 THEN GOTO trap          ' system error

CALL IBSIC(GPIBO%)                    ' Interface clear
IF IBSTA% < 0 THEN GOTO trap        ' system error

tim = 0.5
GOSUB waidly

CALL IBCLR(PPG%)                      ' Device clear

RETURN

wrtcmd:  ' ----- Write command -----

wrt#=wrt#+chr$(13)+chr$(10)
CALL IBWRT(PPG%, wrt#)                ' Write command
IF IBSTA% < 0 THEN GOTO trap          ' Trap

RETURN

readcmd: ' ----- Read command -----

rd#=SPACE$(23)
CALL IBRD(PPG%, rd#)                  ' Read command
IF IBSTA% < 0 THEN GOTO trap          ' Trap

RETURN

waidly:  ' ----- Wait delay -----

stm = TIMER
etm = TIMER

WHILE etm - stm < tim
  etm = TIMER
  IF etm < stm THEN etm = etm + 86400
WEND

RETURN

trap:   ' ----- System trap -----

PRINT "IBERR%:" + STR$(IBERR%)
STOP

END

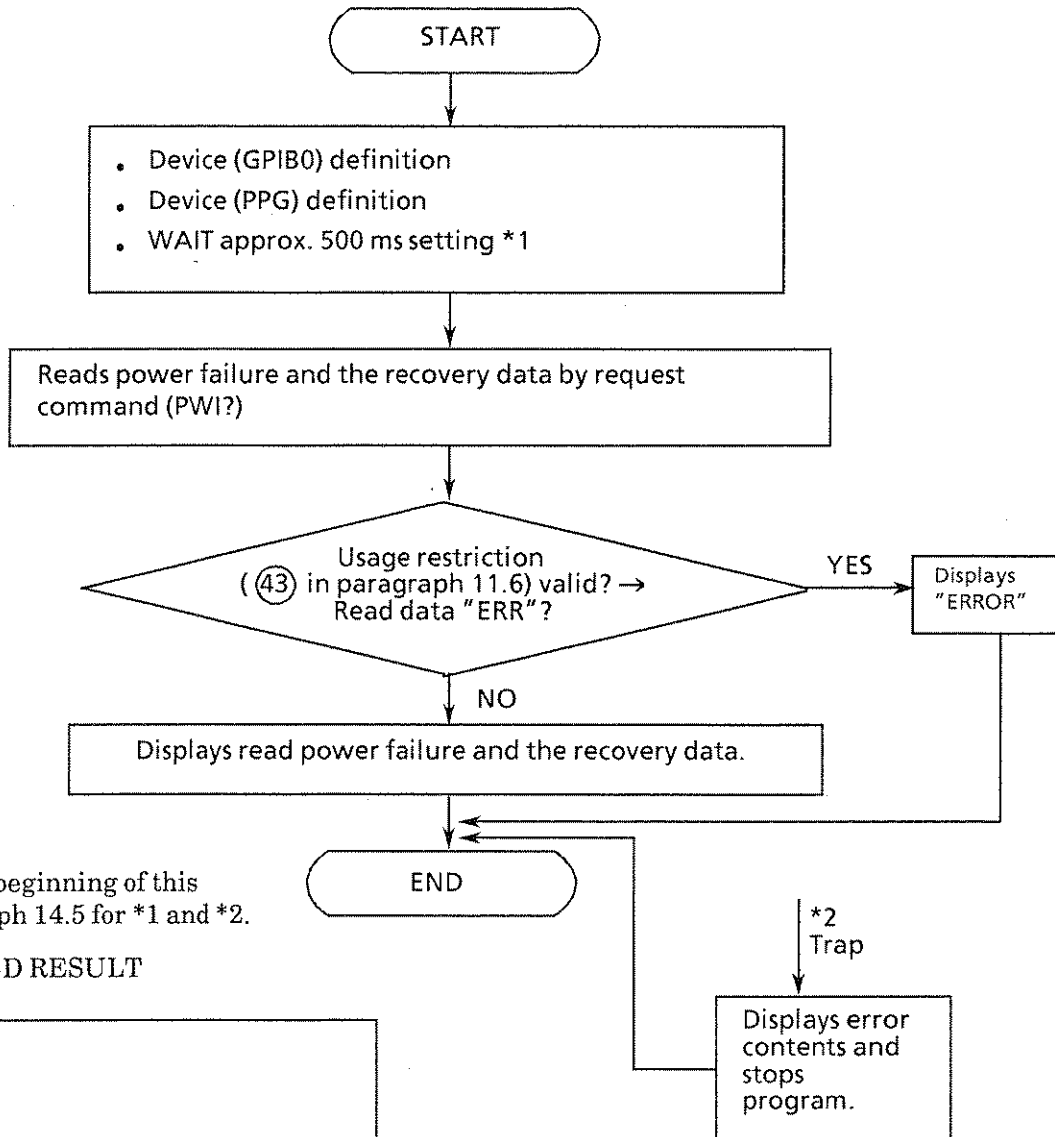
```

(11) Power failure and power recovery status check

This program reads the MP1701B/MP1755A/MP1608A/MP1650A power failure and the recovery information by request command, and displays them on the CRT. It also shows an actual output example.

The time the MP1701B/MP1755A/MP1608A/MP1650A power was turned OFF the last time can be checked by this.

The backup battery usage time can also be checked by recording the power failure interval time.



Note:

See the beginning of this paragraph 14.5 for *1 and *2.

EXECUTED RESULT



PROGRAM LISTING

```

/ *****
*
*          MP1701B / MP1600A / MP1750A
*          POWER FAIL INTERVAL ? SAMPLE SOFT
*
*
*          POWER
*
/ *****

common shared IBSTAX,IBERR%,IBCNT%      ' Setup GPIB-PC functions
GOSUB gpinit                            ' Setup GPIB interface
wrt# = "PWI?" : GOSUB wrtcmd             ' REQUEST Power fail interval?
GOSUB readcmd                           ' READ Power fail interval

IF MID$(rd#,1,3)="ERR" THEN
  PRINT "ERROR !!"
ELSE
  PRINT rd#
/
  FOR I=0 TO 1
    GOSUB readcmd : PRINT rd#
  NEXT I

END IF

STOP

gpinit:  ' ----- Setup GPIB interface -----

CALL IBFIND("GPIBO", GPIBO%)           ' Open device (GPIBO)
IF GPIBO% < 0 THEN GOTO trap           ' system error

CALL IBFIND("PPG", PPG%)                ' Open device (PPG)
IF PPG% < 0 THEN GOTO trap            ' system error

CALL IBSIC(GPIBO%)                      ' Interface clear
IF IBSTAX < 0 THEN GOTO trap          ' system error

tim = 0.5
GOSUB waidly

RETURN

wrtcmd:  ' ----- Write command -----

wrt# = wrt# + chr$(13) + chr$(10)
CALL IBWRT(PPG%, wrt#)                  ' Write command
IF IBSTAX < 0 THEN GOTO trap           ' Trap

RETURN

readcmd: ' ----- Read command -----

rd# = SPACE$(23)
CALL IBRD(PPG%, rd#)                    ' Read command
IF IBSTAX < 0 THEN GOTO trap           ' Trap

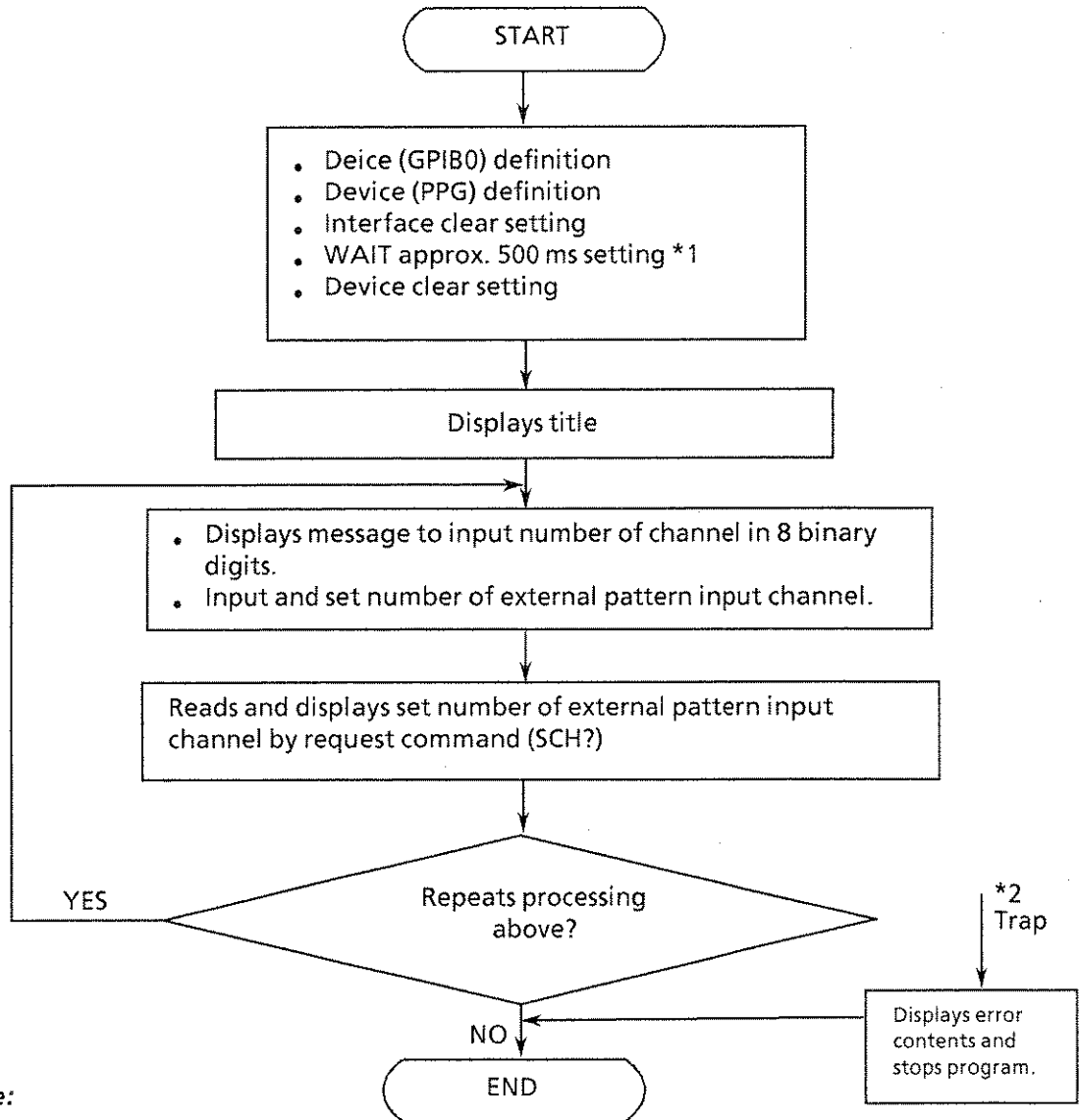
RETURN

```

```
waidly:  ----- Wait delay -----  
stm = TIMER  
etm = TIMER  
WHILE etm - stm < tim  
    etm = TIMER  
    IF etm < stm THEN etm = etm + 86400  
WEND  
RETURN  
trap:   ----- System trap -----  
PRINT "IBERR%:" + STR$(IBERR%)  
STOP  
END
```

(12) Number of external pattern input channel setting

This program sets the number of external pattern input channel.



Note:

See the beginning of this paragraph 14.5 for *1 and *2.

PROGRAM LISTING

```

' *****
' *
' *          MP1701B / MP1608A / MP1650A          *
' *          1/8 SPEED CHANNEL SELECT SAMPLE SOFT *
' *                                          CHANNEL *
' *****

common shared IBSTAX,IBERR%,IBCNT%      ' Setup GPIB-PC functions
GOSUB gpinit                            ' Setup GPIB interface

PRINT "* MP1701B/MP1608A/MP1650A 1/8 SPEED ch SELECT SAMPLE SOFT *"
FRINT
PRINT " CHANNEL 8 FIGURES [ INT =0 , EXT =1 ]
FRINT
FRINT
FRINT

DO
  INPUT "CHANNEL SET DATA BITS(8ch)-->BIT1(1ch) [0/1] ";SCH#
  wrt# = "SCH #B"+sch# : GOSUB wrtcmd

  wrt#="SCH?" : GOSUB wrtcmd
  GOSUB readcmd : PRINT rd#

  INPUT "NEXT DATA SET [ YES=0 , NO=1 ] ";loop#
  FRINT

LOOP UNTIL loop#="1"

STOP

gpinit:  ' ----- Setup GPIB interface -----

CALL IBFIND("GPIBO", GPIBO%)      ' Open device (GPIBO)
IF GPIBO% < 0 THEN GOTO trap      ' system error

CALL IBFIND("PPG", PPG%)          ' Open device (PPG)
IF PPG% < 0 THEN GOTO trap        ' system error

CALL IBSID(GPIBO%)               ' Interface clear
IF IBSTAX < 0 THEN GOTO trap      ' system error

tim = 0.5
GOSUB waidly

CALL IBCLR(PPG%)                 ' Device clear

RETURN

wrtcmd:  ' ----- Write command -----

wrt# = wrt# + chr$(13) + chr$(10)
CALL IBWRT(PPG%, wrt#)           ' Write command
IF IBSTAX < 0 THEN GOTO trap      ' Trap

RETURN

```

```

readcmd: ----- Read command -----
      rd$=SPACE$(16)
      CALL IBRD(PPG%, rd$)           / Read command
      IF IBSTA% < 0 THEN GOTO trap   / Trap
      RETURN

waidly: ----- Wait delay -----
      stm = TIMER
      etm = TIMER
      WHILE etm - stm < tim
        etm = TIMER
        IF etm < stm THEN etm = etm + 86400
      WEND
      RETURN

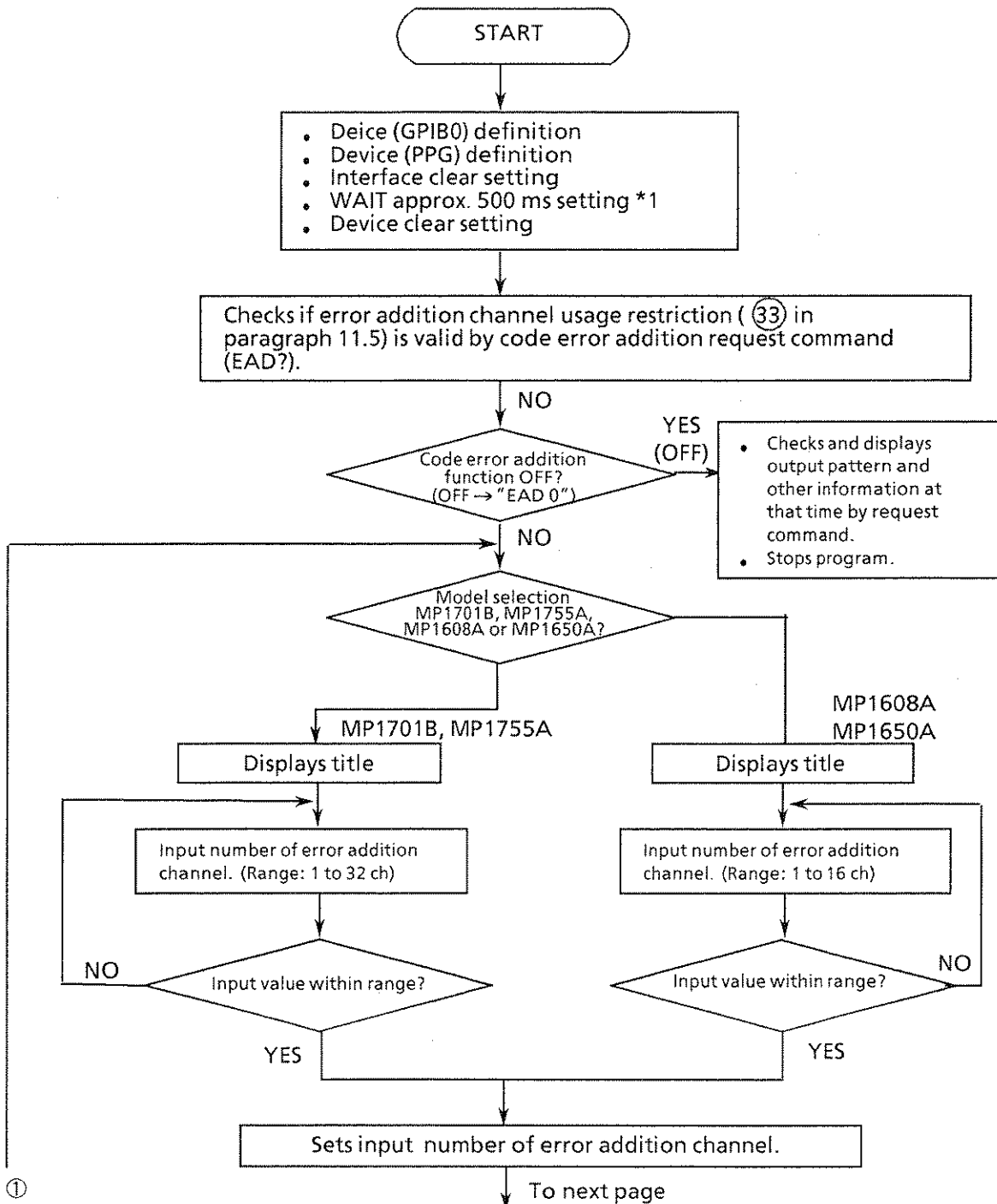
trap: ----- System trap -----
      PRINT "IBERR%:" + STR$(IBERR%)
      STOP
      END

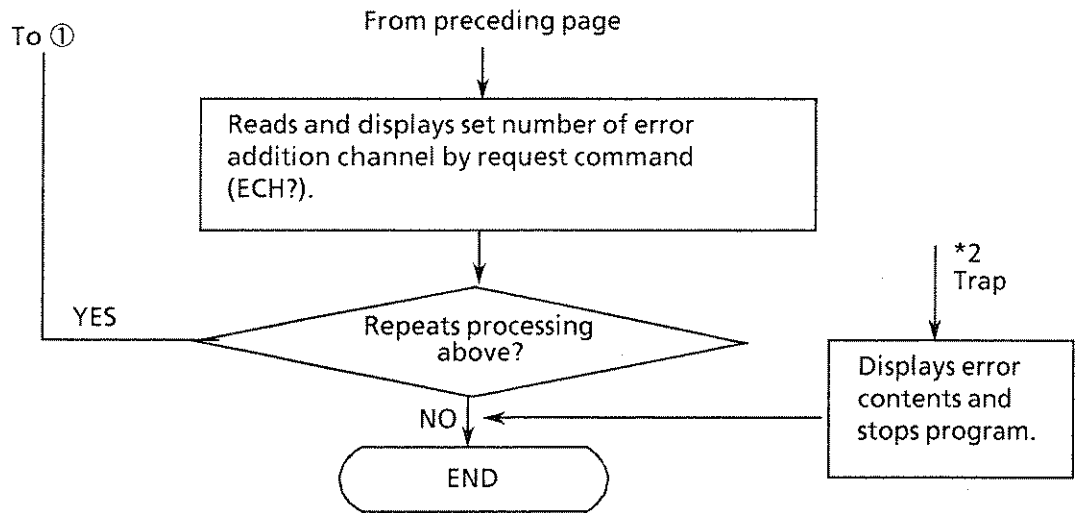
```


(13) Error addition channel setting

This program sets the error addition channel.

The error addition channel cannot be set when the code error addition function is OFF. At this time, the output pattern related information is displayed on the CRT and program execution ends.





Note:

See the beginning of this paragraph 14.5 for *1 and *2.

PROGRAMING LISTING

```

/ *****
/ *
/ * MP1701B/MP1608A/MP1650A ERROR ADDITION ch SET SAMPLE SOFT *
/ *
/ * ERR_ADD *
/ *****
/
/-----
/                               MAIN ROUTINE
/-----
/
common shared IBSTAZ,IBERRZ,IBCNTZ / Setup GPIB-PC functions
GOSUB gpinit / Setup GPIB interface
/
wrt# = "EAD?" : GOSUB wrtcmd / REQUEST Error addition mode
GOSUB readcmd / READ Error addition mode
IF MID$(rd#,1,5)="EAD Q" THEN / OFF --> ERROR
  GOSUB Err: PRINT " **PROGRAM STOP**"
  STOP
END IF
/
DO
  CLS
  /
  INPUT " MODE SELECT [ MP1701B=0, MP1608A=1, MP1650A=2 ]";mode#
  PRINT
  /
  SELECT CASE mode#
    CASE "0": GOSUB 106
    CASE "1": ppg#="MP1608A": GOSUB 5636
    CASE "2": ppg#="MP1650A": GOSUB 5636
  END SELECT
  /
  wrt# = "ECH "+STR$(ech) : GOSUB wrtcmd /Set Error addition ch
  /
  wrt#="ECH?" : GOSUB wrtcmd
  GOSUB readcmd
  PRINT " ERROR ADDTION CHANNEL = "+MID$(rd#,5,2)
  /
  INPUT " NEXT DATA SET [ YES=0 , NO=1 ]";loop#
  /
LOOP UNTIL loop#="1"
/
STOP
/
/-----
/                               SUB ROUTINE
/-----
/
106:----- MP1701B DATA INPUT -----
/
PRINT " *** MP1701B ERROR ADDTION ch SAMPLE SOFT ***"
PRINT
DO
  INPUT " ERROR ADDITION ch SET DATA [ 1~32 ] ";ech
/
LOOP UNTIL ech < 33 AND ech > 0
/
RETURN
/

```

```

5G3G: ----- MP1608A,MP1650A DATA INPUT -----
PRINT " *** "+ppg#+ " ERROR ADDTION ch SAMPLE SOFT ***"
PRINT
DO
  INPUT " ERROR ADDITION ch SET DATA [ 1~16 ] ";ech
LOOP UNTIL ech < 17 AND ech > 0
RETURN

Ero: ----- Display Pattern mode -----
PRINT "ERROR !! ERROR ADDITION MODE = OFF "
PRINT
wrt# = "PTN? " : GOSUB wrtcmd      ' REQUEST Pattern mode ?
GOSUB readcmd      ' READ Pattern mode ?

SELECT CASE MID$(rd#,5,1)
  CASE "0": GOSUB ProgW
  CASE "1": GOSUB ProgD
  CASE "2": PRINT "PATTERN MODE = PRBS.PN7 "
  CASE "3": PRINT "PATTERN MODE = PRBS.PN9 "
  CASE "5": PRINT "PATTERN MODE = PRBS.PN11 "
  CASE "6": PRINT "PATTERN MODE = PRBS.PN15 "
  CASE "7": PRINT "PATTERN MODE = PRBS.PN20 "
  CASE "8": PRINT "PATTERN MODE = PRBS.PN23 "
  CASE "9": PRINT "PATTERN MODE = PRBS.PN31 "
END SELECT

RETURN

ProgW: ----- PROG.WORD MODE -----
PRINT " Pattern mode is PROG.WORD MODE "
PRINT
wrt# = "WNB?": GOSUB wrtcmd : GOSUB readcmd
PRINT "NUMBER OF WORD =" +MID$(rd#,4,6)

wrt# = "WLN?": GOSUB wrtcmd : GOSUB readcmd
PRINT "WORD LENGTH =" +MID$(rd#,4,3)

wrt# = "PAG?": GOSUB wrtcmd : GOSUB readcmd
PRINT "PAGE =" +MID$(rd#,4,6)

RETURN

ProgD: ----- PROG.DATA MODE -----
PRINT "Pattern mode is PROG.DATA MODE "
PRINT

```

```
wrt# = "DLN?" : GOSUB wrtcmd : GOSUB readcmd
PRINT "DATA LENGTH =" + MID$(rd#,4,7)
```

```
wrt# = "PAG?" : GOSUB wrtcmd : GOSUB readcmd
PRINT "PAGE =" + MID$(rd#,4,6)
```

```
RETURN
```

```
gpinit:  ----- Set up GF-IB functions -----
```

```
CALL IBFIND("GPIBO", GPIBO%)      ' Open device (GPIBO)
IF GPIBO% < 0 THEN GOTO trap      ' system error
```

```
CALL IBFIND("PPG", PPG%)          ' Open device (PPG)
IF PPG% < 0 THEN GOTO trap      ' system error
```

```
CALL IBSIC(GPIBO%)               ' Interface clear
IF IBSTAX < 0 THEN GOTO trap    ' system error
```

```
tim = 0.5
GOSUB waidly
```

```
CALL IBCLR(PPG%)                 ' Device clear
```

```
RETURN
```

```
wrtcmd:  ----- Write command -----
```

```
wrt# = wrt# + chr$(13) + chr$(10)
CALL IBWRT(PPG%, wrt#)           ' Write command
IF IBSTAX < 0 THEN GOTO trap    ' Trap
```

```
RETURN
```

```
readcmd: ----- Read command -----
```

```
rd# = SPACE$(12)
CALL IBRD(PPG%, rd#)             ' Read command
IF IBSTAX < 0 THEN GOTO trap    ' Trap
```

```
RETURN
```

```
waidly:  ----- Wait delay -----
```

```
stm = TIMER
etm = TIMER
```

```
WHILE etm - stm < tim
  etm = TIMER
  IF etm < stm THEN etm = etm + 86400
WEND
```

```
RETURN
```

```
trap:   ----- System trap -----
```

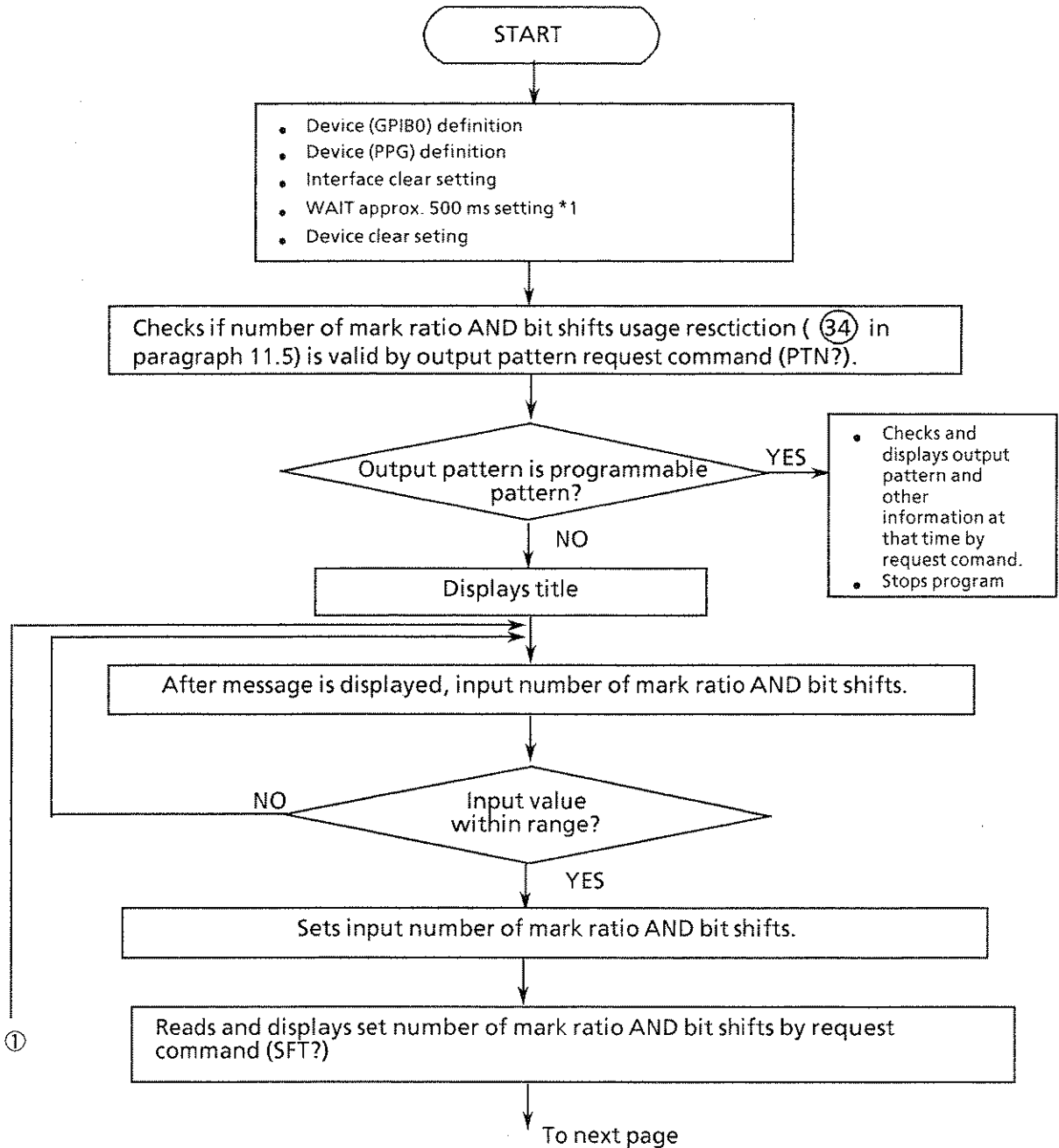
```
PRINT "IBERR:" + STR$(IBERR%)
STOP
```

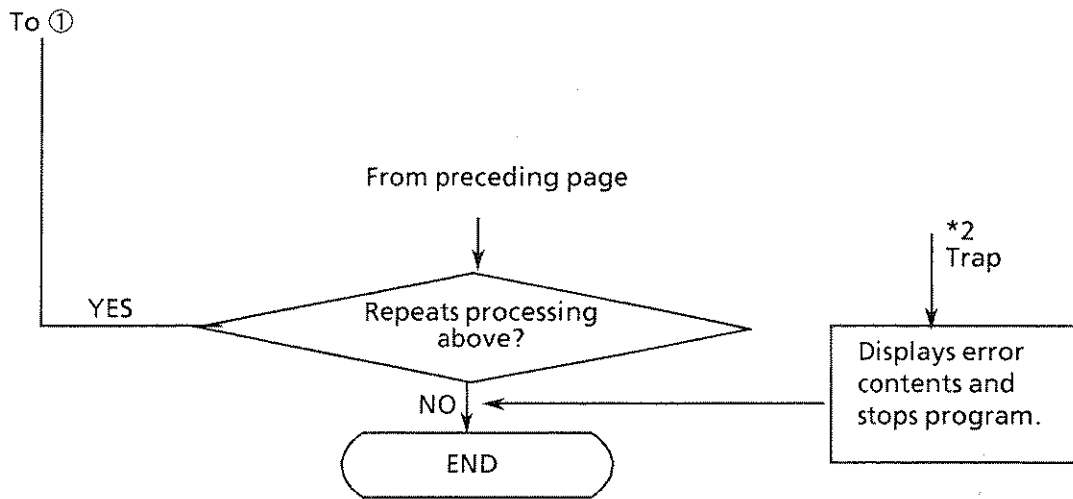
```
END
```

(14) Number of mark ratio AND bit shifts setting

This program sets the number of mark ratio AND bit shifts.

The number of mark ratio AND bit shifts cannot be set when the output pattern is in the programmable mode. At that time, the information related to output pattern is displayed on the CRT and program execution is ended.





Note:

See the beginning of this paragraph 14.5 for *1 and *2.

PROGRAMING LISTING

```

*****
*
*      MP1701B/MP1608A/MP1650A  MARK RATIO SAMPLE SOFT
*
*
*
*****
-----
MAIN ROUTINE
-----

common shared IBSTA%,IBERR%,IBCNT%      ' Setup GPIB-PC functions
GOSUB gpinit                             ' Setup GPIB interface

PRINT " *** MP1701B/MP1608A/MP1650A  MARK RATIO  SAMPLE SOFT ***"
PRINT

wrt# = "PTN?" : GOSUB wrtcmd              ' REQUEST Pattern mode ?
GOSUB readcmd                             ' READ Pattern mode

IF MID$(rd#,5,1)="0" OR MID$(rd#,5,1)="1" THEN
  GOSUB Ero                               ' Word/Data mode-->Error
  PRINT "*** PROGRAM STOP ** "
  STOP
END IF

DO
  GOSUB Mark

  INPUT " NEXT DATA SET [ YES=0 , NO=1 ]";loop#
  PRINT

LOOP UNTIL loop#="1"

STOP

-----
SUB ROUTINE
-----

Mark:  ----- SET Mark ratio bit shift -----

DO
  INPUT "MARK RATIO SHIFT [ 1 BIT Shift=0 , 3 BIT Shift=1 ] ";sft
LOOP UNTIL sft=0 OR sft=1

wrt# = "SFT " +STR$(sft) : GOSUB wrtcmd
wrt#="SFT?" : GOSUB wrtcmd : GOSUB readcmd

IF MID$(rd#,1,5)="SFT 0" THEN
  rd#="1"
ELSE
  rd#="3"
END IF

PRINT "MARK RATIO BIT SHIFT= "+rd#

RETURN

```


Ero: : ----- Error --> Display Pattern data -----

```
IF MID$(rd$,5,1)="0" THEN
PRINT "ERROR !! Pattern mode = PROG.WORD MODE "
PRINT
wrt$ = "WNB?" : GOSUB wrtcmd : GOSUB readcmd
PRINT "NUMBER OF WORD =" + MID$(rd$,4,6)
wrt$ = "WLN?" : GOSUB wrtcmd : GOSUB readcmd
PRINT "WORD LENGTH =" + MID$(rd$,4,3)
wrt$ = "PAG?" : GOSUB wrtcmd : GOSUB readcmd
PRINT "PAGE =" + MID$(rd$,4,6)
ELSE
PRINT "ERROR !! Pattern mode = PROG.DATA MODE "
PRINT
wrt$ = "DLN?" : GOSUB wrtcmd : GOSUB readcmd
PRINT "DATA LENGTH =" + MID$(rd$,4,7)
wrt$ = "PAG?" : GOSUB wrtcmd : GOSUB readcmd
PRINT "PAGE =" + MID$(rd$,4,6)
END IF
RETURN
```

gpinit: : ----- Set up GP-IB functions -----

```
CALL IBFIND("GPIBO", GPIBO%) ' Open device (GPIBO)
IF GPIBO% < 0 THEN GOTO trap ' system error
CALL IBFIND("PPG", PPG%) ' Open device (PPG)
IF PPG% < 0 THEN GOTO trap ' system error
CALL IBSIC(GPIBO%) ' Interface clear
IF IBSTAX < 0 THEN GOTO trap ' system error
tim = 0.5
GOSUB waidly
CALL IBCLR(PPG%) ' Device clear
RETURN
```

wrtcmd: : ----- Write command -----

```
wrt$=wrt$+chr$(13)+chr$(10)
CALL IBWRT(PPG%, wrt$) ' Write command
IF IBSTAX < 0 THEN GOTO trap ' Trap
RETURN
```

readcmd: : ----- Read command -----

```
rd$= SPACE$(12)
CALL IBRD(PPG%, rd$) ' Read command
IF IBSTAX < 0 THEN GOTO trap ' Trap
RETURN
```

```
waidly: / ----- Wait delay -----  
      .  
      stm = TIMER  
      etm = TIMER  
      .  
      WHILE etm - stm < tim  
        .  
        etm = TIMER  
        IF etm < stm THEN etm = etm + 86400  
      .  
      WEND  
      .  
      RETURN  
      .  
trap: / ----- System trap -----  
      .  
      PRINT "IBERR%:" + STR$(IBERR%)  
      .  
      STOP  
      .  
      END
```