ADVANTEST CORPORATION

NETWORK ANALYZER

PROGRAMMING MANUAL

MANUAL NUMBER OEA00 9509

Applicable Instruments
R3764/66
R3765/67

Before reselling to other corporations or re-exporting to other countries, you are required to obtain permission from both the Japanese Government under its Export Control Act.

First printing September 20, 1995
Printed in Japan

How to Use This Manual

The following describes the structure of this manual.

- Part 1: Built-in BASIC
- Part 2: GPIB

Reference : For details of the network analyzer section names functions and key operations, refer to the pertinent instruction manual.

- R3764/66 Network Analyzer Instruction Manual
  or
- R3765/67 Network Analyzer Instruction Manual

Part 1

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

1. INTRODUCTION

The BASIC language built into the network analyzer is equipped with general-purpose BASIC commands, GPIB control purpose commands, and exclusive built-in functions, enabling the network analyzer to be used for simple configuration of small GPIB systems.

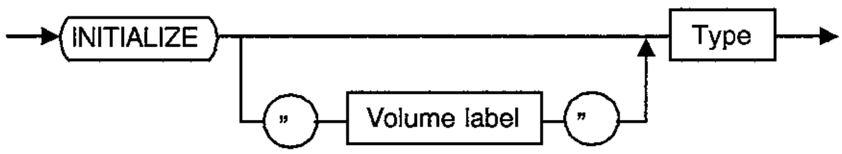
- Command and statement syntax

The syntax for the commands and statements used for this analyzer is explained in Chapters 3 and 4 of this manual with both schematic and descriptive representations for intuitive understanding.

CAUTION

How to read the syntax for commands and statements

(1) Schematic representation
To represent a syntax, the analyzer disassembles it into its elements and connects them with straight lines.
Statements should always be read in the direction of the arrows. If a statement jumps to multiple branches on the way, the analyzer will go to one of them. If a loop is formed in the representation, the loop can be passed any number of times.
Description example:



(2) Meanings of symbols used for descriptive representation

- Part enclosed with symbols [ ]: Indicates that the enclosed item is an option (omissible).
- Part enclosed with symbols < >: Indicates that the enclosed item is not an option (un-omissible).
- Part enclosed with symbols { }: Indicates that the enclosed item is repeatable 0 times or more.
- Symbol | : Indicates "or". (ex. A|B - A or B is selectable.)

Example of representation: INITIALIZE ["volume label"] <type>

(3) Meanings of words used for schematic and descriptive representations

- Numerical value representation expression: Any one of numeric value constant, numeric value variable, and expression
- Character string representation expression: Expression consisting of character string constant, character string variable, character string function, and sub-string
- Equipment address: Address of device connected to GPIB

- GPIB mode

The analyzer operates in either of two modes: ADDRESSABLE or CONTROL. The switching between the modes is performed using the CONTROL command or from the front panel.

For the use of the CONTROL command, refer to "3. BASIC COMMANDS". For the use of the front panel, refer to the instruction manual for the pertinent unit.

(1) ADDRESSABLE mode

The ADDRESSABLE mode is a normal mode. In this mode, the analyzer is controlled by an external controller.
If the built-in BASIC program of the analyzer is run in this mode, the analyzer will operate as follows:

① If "CONTROL 7;4" of the BASIC command has not been set:

Data can be transmitted/received between the built-in BASIC of the analyzer and an external controller.
However, since the ENTER and OUTPUT instructions of the built-in BASIC have higher priority, setting cannot be performed using a GPIB command from the external controller.
Perform setting using a GPIB command from the external controller, stop the built-in BASIC program or set "CONTROL 7;4".

② If "CONTROL 7;4" of the BASIC command has been set:

In contrast with ①, setting can be performed using a GPIB command from an external controller.
In other words, the system operates in the same manner as when the built-in BASIC is stopped. However, no data can be transmitted/received between the built-in BASIC and the external controller.

(2) SYSTEM CONTROLLER mode

The built-in BASIC program enables the analyzer to control the measurement function and the externally connected units.

Note: In this page, the BASIC built in the analyzer is called the built-in BASIC in order to distinguish from the external controller. But when the distinction from the external is not needed hereafter, it's called BASIC.

- Floppy Disk

The floppy disk is used for storing/reading the setting condition and the measured data or a BASIC program and the files from the BASIC program.
The floppy disk format complies with MS-DOS, enabling programs to be created or data to be analyzed using a personal computer corresponding to MS-DOS.
In the analyzer, the disks initialized with the following formats can be used:

2DD (Double-sided double-density):	720 Kbytes (512 bytes, 9 sectors)
2HD (Double-sided high-density):	1.2 Mbytes (1024 bytes, 8 sectors) 1.2 Mbytes (512 bytes, 15 sectors) 1.4 Mbytes (512 bytes, 18 sectors)

CAUTION

The analyzer automatically discriminates between 2DD and 2HD disks. 2DD floppy disks formatted to hold 1.2 Mbytes or 1.4 Mbytes and 2HD floppy disks formatted to hold 720 Kbytes cannot be used.

(1) External appearance and names of micro-floppy disk

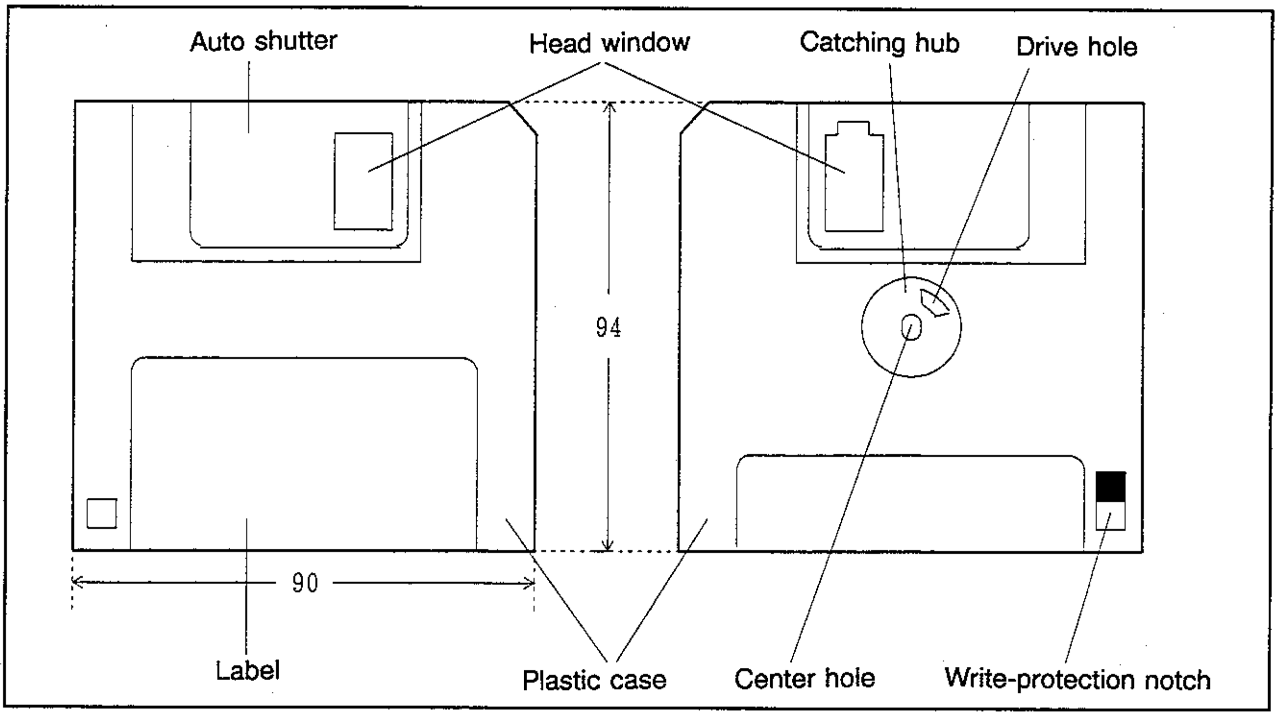


Figure 1-1 External Appearance and Names of Parts of Micro-Floppy Disk

- Label: Adhesive label for floppy disk
- Head window: The READ/WRITE head is positioned at the corresponding opening on the back of the floppy disk. The head is aligned with this slot.
  When the floppy disk is pulled out from the drive slot, the auto shutter closes to protect the disk.
- Catching hub (drive hole, center hole):
  When the floppy disk is inserted into the drive slot, a spindle which uses a catching magnet on the drive side fixes and rotates the floppy disk.
- Write-protect window: Writing can be prohibited to prevent important data from being erased by mistake.

(2) Insertion and handling of floppy disks

Insert the floppy into the disk drive with the label facing upwards, as shown in Figure 1-2. Check that the disk is fully inserted in the drive by pushing it in with a finger. The disk is ejected automatically when the eject button is pressed.

CAUTION

Never press the eject button while the floppy drive lamp is blinking, since this could result in incorrect operation or data loss.

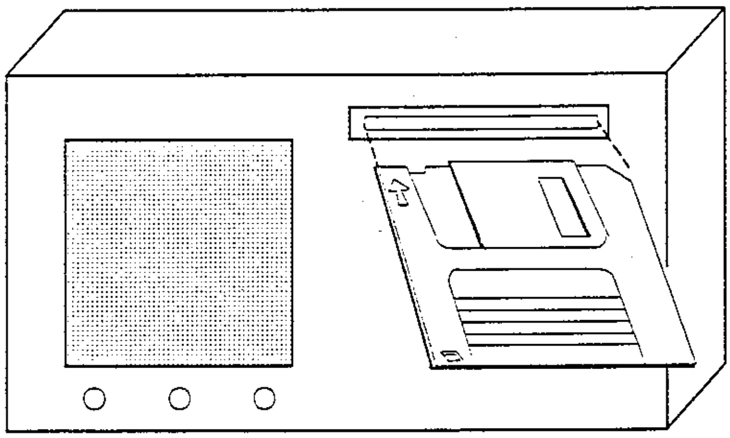


Figure 1-2 Inserting Floppy Disk (for R3765/67)

When handling floppy disks, pay attention to the following items.

① Keep away from materials which generate a strong magnetic field.

② Do not expose to extreme heat or direct sunlight.

③ Take care to avoid cigarette ash and other contaminants.

④ Do not touch the magnetic surface.

⑤ Do not place heavy objects on disks.

⑥ Damaged disks (wet, dripped, bent, etc.) or those which have been contaminated with foreign particles should be changed.

(3) Write protect

Important data should be protected from accidental erasure by using the write-protect shutter. To protect data, slide the write-protect tub (Figure 1-3).
Writing is possible when the tub is closed to the center hole and not possible when furthest from the center hole.

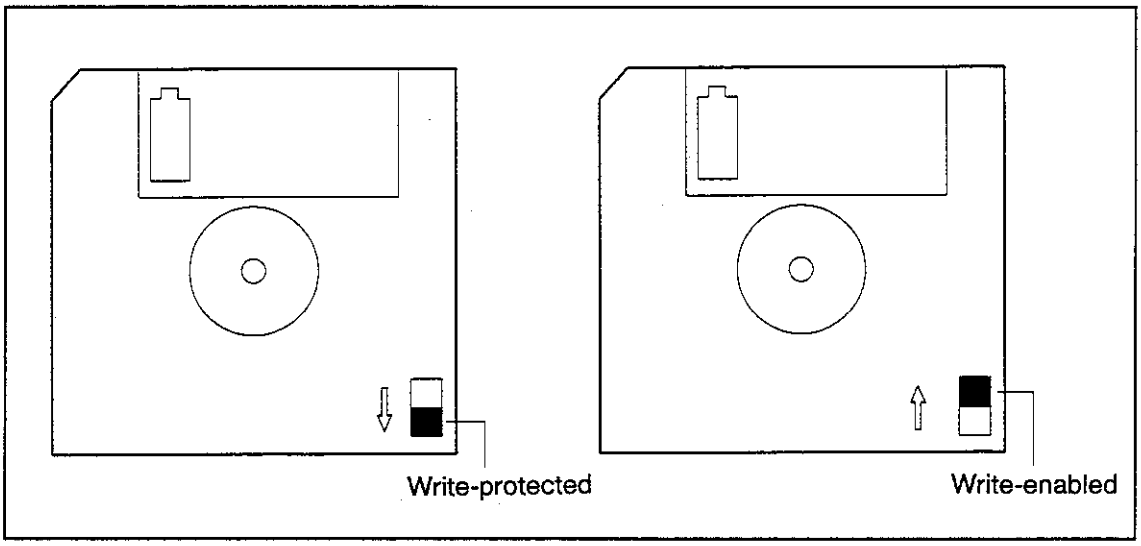


Figure 1-3 Write-Protect Tab Position

- File Management

The management of disk files for the analyzer is the same as for disk files created by MS-DOS. In other words, the analyzer can use MS-DOS-formatted floppy disks itself, and files created by the analyzer can be referenced from MS-DOS.

(1) File

Generally, a group of data is called a "file". BASIC programs edited on personal computers (PCs) and data created by BASIC are all stored as files.

(2) Directory

Each directory can manage the file.
The analyzer does not have a function to create the directory, but can refer to files in the sub-directory.

(3) Drive

Files are stored on disks such as floppy disks and memory disks. A unit which reads and writes files is called a "drive". Each drive manages one disk. The following four drives are provided for the analyzer:

A: Floppy disks
Same as floppy disks created using MS-DOS

B: Memory disks which cannot be backed up
These disks are automatically formatted when the analyzer is switched on. When the analyzer is switched off, the contents of the disk are lost.
BASIC can use up to 128 Kbytes, but when the register is used, the usable capacity decreases.

C: Memory disks which can be backed up
The contents of the disk can be maintained when the analyzer is switched off.
BASIC can use up to 900 Kbytes, but when the register is used, the usable capacity decreases.

D: Read-only memory disks
These disks maintain the system program of the analyzer.
BASIC cannot use these memory disks.

To select the current drive, refer to the instruction manual for each model of analyzer.

(4) Specifying files

The following shows how to specify a file containing drive and directory.

"drive name:/directory name/file name"

Usually, MS-DOS uses "¥" (" \" in English mode) as a delimiter of directory. But this analyzer uses "/" instead. As "\" in the character string is used in particular in this analyzer as described in "4. BASIC statement", the analyzer uses "/" but not "\".

(5) Initializing floppy disks

When a new floppy disk is to be used, it must first be initialized (formatted).
The following three initialization methods are possible:

① Execute the FORMAT command contained in MS-DOS by using the personal computer and use the formatted disk in the analyzer.

② Analyzer panel operation (Refer to the description of the panel operation.)

③ Execute the INITIALIZE command contained in the BASIC program of the analyzer.

Generally, the format of floppy disk has the following four types.

① 1.44 Mbytes type (2HD, 512 bytes, 18 sectors)
② 1.2 Mbytes type (2HD, 1024 bytes, 8 sectors)
③ 1.2 Mbytes type (2HD, 512 bytes, 15 sectors)
④ 720 Kbytes type (2DD, 512 bytes, 9 sectors)
⑤ 640 Kbytes type (2DD, 512 bytes, 8 sectors)

The analyzer can use these four types of floppy disk but ⑤.

Note: In PC9801 series, the default is ⑤ format when 2DD floppy is formatted by FORMAT command.
The floppy used in this analyzer must be formatted to be ④ format.

- Keyboard

101 type keyboard and 106 type keyboard prescribed by OADG (PC Open Architecture Developers' Group) can be connected.
In case of R3765/67 series, pressing PROGRAM key on the front panel, the keyboard for BASIC can be input.

CAUTION

The keyboard must be connected before turning the power on.
If it's connected after turning the power on, the normal operation cannot be guaranteed.

MEMO

2. OPERATING BASICS

How to create, carry out, and end the program are shown below.

2.1 Program Creating

① Creating with personal computer

The input and the edit are performed with personal computer, and the program is saved into the floppy disk in the form of ASCII.

② Creating with keyboard

The input is performed with the line numbers of program, and the program is saved into the floppy disk.

CAUTION

There's no constraint about the file extension, but in order to distinguish BASIC program files from others, use BAS for the extension.

The character code that can be handled in BASIC is 7 bits ASCII code.
But if the following characters are used in the program statement, the program loading is stopped at the line, for they are not used in BASIC. (Except the case enclosed in double quotation marks.)

2.2 Program Carrying Out

- R3764/66 series

① Mount the floppy disk, in which the program you want to carry out is saved, to the floppy disk drive of the analyzer.

② Press LOAD key (panel key) to display the files in the floppy disk.

③ Use ↑ or ↓ key (panel key) to move the cursor to the file name which you want to load.

④ Pressing ENT key (panel key), the program is loaded.

⑤ Pressing RUN key (panel key), the program is carried out.

- R3765/67 series

① Mount the floppy disk, in which the program you want to carry out is saved, to the floppy disk drive of the analyzer.

② Press RUN key (panel key) to display the controller menu.

③ Press LOAD MENU key (soft key) to display the files in the floppy disk.

④ Use ↑ or ↓ cursor (soft key) to move the cursor to the file name which you want to load.

⑤ Pressing LOAD key (soft key), the program is loaded.

⑥ Pressing RUN key (soft key), the program is carried out.

2.3 Program Ending

① R3764/66 series

Pressing STOP key (panel key), the program ends.

② R3765/67 series

1) Press RUN key (panel key) to display the controller menu.

2) Pressing STOP key (soft key), the program ends.

MEMO

3. BASIC COMMANDS

In the BASIC, commands and statements are used.
Commands are carried out directly (not in the program) basically, while statements are carried out in the program basically.
Here describes about commands first.

3.1 Various Commands

BASIC has commands to edit, carry out programs and operate files. The following shows the structure of the BASIC commands.

- Built-in BASIC commands
  - Direct commands
    - Program input
    - DEL
    - GLIST
    - GLISTN
    - LIST
    - LISTN
    - LLIST
    - LLISTN
    - PRINTER
    - REN
    - SCRATCH
  - Execution commands
    - CONT
    - CONTROL
    - PAUSE
    - STEP
    - STOP
    - RUN
  - File commands
    - CAT
    - COPY
    - INITIALIZE
    - LOAD
    - MERGE
    - PURGE
    - SAVE
    - RENAME

Note: A capital letter is used for command.

In these commands, some can be carried out in the program as statements.

3.1.1 List of Command Function

	Command	Function	Possible as statements
EDIT commands	Program input	Stores the statement as a program.	×
	DEL	Deletes the specified line number.	×
	GLIST	Outputs the program list to the GPIB.	○
	GLISTN	Outputs the program list to the GPIB.	○
	LIST	Displays the program list on the screen.	○
	LISTN	Displays the program list on the screen.	○
	LLIST	Outputs the program list to the serial port.	○
	LLISTN	Outputs the program list to the serial port.	○
	PRINTER	Sets the GPIB address of the printer.	○
	REN	Changes the line number.	○
	SCRATCH	Deletes the already input program.	×
EXECUTION commands	CONT	Runs the program again.	×
	CONTROL	Sets the BASIC control variables. (Environment setup)	○
	PAUSE	Suspends the program. (Enables CONT command)	○
	STEP	Runs the program one line.	×
	STOP	Stops the program. (Disables CONT command)	○
	RUN	Runs the program.	○
FILE commands	CAT	Displays the file name in the current drive onto the screen.	○
	COPY	Copies the file.	○
	INITIALIZE	Initializes the floppy disk.	○
	LOAD	Loads (Invokes) the program.	○
	MERGE	Loads (Invokes) the program to add it to the already input program.	○
			○
	PURGE	Purges the file.	○
	SAVE	Saves (Stores) the program.	○
	RENAME	Renames the file name.	○

3.1.2 List of Command Syntax

	Command	Syntax
EDIT commands	Program input	Line number Statement
	DEL	DEL Start line [, Last line]
	GLIST	GLIST [Start line] [, [Last line]]
	GLISTN	GLISTN [Start line] [, [Line number]]
	LIST	LIST [Start line] [, [Last line]]
	LISTN	LISTN [Start line] [, [Line number]]
	LLIST	LLIST [Start line] [, [Last line]]
	LLISTN	LLISTN [Start line] [, [Line number]]
	PRINTER	PRINTER Device address
	REN	REN [[Current line number] [, <New line number> [, <Increment>]]]
	SCRATCH	SCRATCH [1|2]
EXECUTION commands	CONT	CONT [Line number]
	CONTROL	CONTROL <Resistor number>;<Value>
	PAUSE	PAUSE
	STEP	STEP [Line number]
	STOP	STOP
	RUN	RUN [Line number|"File name"]
FILE commands	CAT	CAT ["DATE"]
	COPY	COPY "Current file name", "New file name"
	INITIALIZE	INITIALIZE ["Volume label"] <Type>
	LOAD	LOAD "File name"
	MERGE	MERGE "File name"
	PURGE	PURGE "File name"
	SAVE	SAVE "File name"
	RENAME	RENAME "Current file name", "New file name"

3.1.3 Precautions Common to All Commands

The following precautions are common to all of the built-in BASIC commands:

(1) Parameters

The character string representation expression and numeric value representation expression can be used to specify command parameters. In other words, variables used in the BASIC command can be used. If the number used is a real number, digits to the right of the decimal point will be omitted.
The description of each command uses representations such as integers and character strings for easy understanding.

(2) Boundary of expression

In principle, when the BASIC command uses multiple expressions continuously, a space can be used instead of a comma, as long as the boundary of the expressions can be interpreted in the syntax.

(3) Line number in LIST, LISTN, LLIST, LLISTN, GLIST, and GLISTN.

The line number setting range is 1 to 65535.
If 0 or any value below the first line number of the program is specified, the analyzer will interpret that the first line of the program has been specified.
If 65535 or any value over the last line number of the program is specified, the analyzer will interpret that the last line of the program has been specified.
If the number which has been specified does not exist, the nearest number over the specified line number is selected. The label can be specified instead of the line number.

3.2 Command Grammar and Application

1. Program Input

The commands and statements described in Chapters 3 and 4 can be entered as a program if line numbers are added to them.

If the same line number exists in a program which has already been input, the newly entered number will replace it. If the same line number does not exist, the new number will be added or inserted.

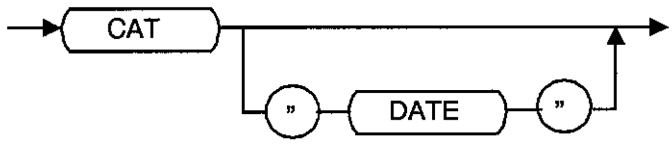
2. CAT

Outline

The CAT command is used to list the names of the files stored on the current drive.

Syntax

(1)-1



(1)-2
CAT ["DATE"]

Description

- The CAT command lists the names of the files and directories stored on the current drive.

  CAT: Displays the registered number, the file name, the number of bytes used, and the file attribute in that order from the left.
  CAT "DATE": Displays the registered number, the file name, and the date the file was created in that order from the left.

Note: For the information how to handle files, refer to "1. Preface • File Management".

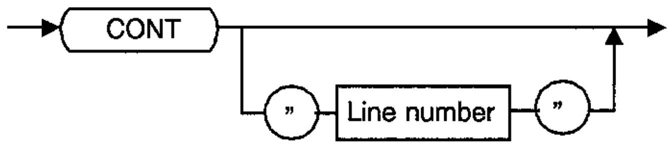
3. CONT

Outline

The CONT command is used to restart the BASIC program.

Syntax

(1)-1



(1)-2

CONT [Line number]

Description

- The CONT command restarts the BASIC program which is paused by the PAUSE command at the next of the line where the program pauses.
- The CONT command restarts the BASIC program at the desired (specified) line. Cannot be used to initialize variables.
- The CONT command cannot be used as a statement in the program.

Example

CONT

CONT 200

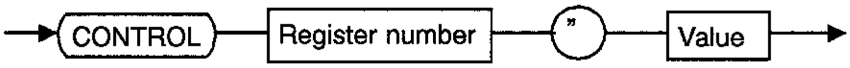
4. CONTROL

Outline

The CONTROL command is used to set the detailed values concerning the BASIC control (environment setup).

Syntax

(1)-1



(1)-2

CONTROL register number ; value

Description

- The CONTROL command specifies the items to be controlled by the register number. The value followed by a semicolon is the actual value.
- The value 1 to 9 can be set to the register number. The contents of each register are as follows. (However, the register 4 has not been used by means of internal structure.)

<Register 1> ··· Initial value: 79

Sets a serial I/O port. The total of values added up is used to specify the serial I/O port. The following underlined-value is each default value which has been already set when the analyzer is turned on.

① Baud rate:	0;	1200 baud	③ Parity:	0;	None
	1;	2400 baud		16;	Odd
	2;	4800 baud		48;	Even
	3;	9600 baud			
② Character length:	0;	5 bits	④ Stop-bit number:	0;	None
	4;	6 bits		64;	1 bit
	8;	7 bits		128;	1 1/2 bits
	12;	8 bits		192;	2 bits

Example: When 9600 bps for baud rate, 8 bits for character length, even parity for parity, and 2 bits for stop-bit number are used:

CONTROL 1;3 + 12 + 48 + 192
or
CONTROL 1;255

<Register 2> ··· Initial value: 0

With the command LLIST or GLIST, specifies the print position from the left side by entering the number of spaces.

Example: When the list output is moved to the right by five characters

Execute the CONTROL 2;5 first and the LLIST or GLIST, five spaces will be inserted immediately before the line number, then the list will be displayed after that.

<Register 3> ··· Initial value: 0

Specifies whether the BASIC program will be displayed in full name or short name.

0: Full name
1: Short name

For the relationship between the full and short names, refer to Table 4-2.

<Register 5> ··· Initial value: 0

Specifies whether the maintenance command POKE is available or not.

0: Not available
1: Available

<Register 7> ··· Initial value: 0

Used for GPIB setting. Each value must be set as follows:

0: Sets GPIB mode to ADDRESSABLE.
1: Sets GPIB mode to SYSTEM CONTROLLER.
2: Transits REQUEST CONTROL (request for control privilege).
4: Enables GPIB command setting from the external controller during BASIC operation.

<Register 8> ··· Initial value: 0

Sets ON/OFF of DMA transfer mode.

0: OFF
1: ON

<Register 9> ··· Initial value: 1

Specifies a desired output instrument for PRINT. The total of values added up is used to set up.

1: Default output (front panel indicator of each model)
2: Output to maintenance port (terminal)
4: Output to external monitor or R3765/67 LCD
8: Output to R3764/67 fluorescent character display tube

Example 1: Output to default and maintenance port
CONTROL 9;3

Example 2: Output to default, maintenance port and external monitor
CONTROL 9;7

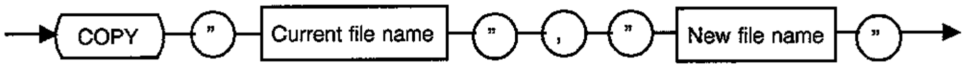
5. COPY

Outline

The COPY command is used to copy the files.

Syntax

(1)-1



(1)-2

COPY "current file name", "new file name"

Description

- The COPY command copies the contents of the current file name to a new file name.
- When a new file name has already existed, the contents of the current file is overwritten.
- If the new file name is the same as the current file name, then the error will be occurred.
- Both of two file names can be specified by using a character-string expression.
- If the drives are specified, the copy between the drives can be made. If there's no specification about the drive, the file copy is carried out in the current drive.

Note: For the information how to handle files, refer to "1. Preface • File Management".

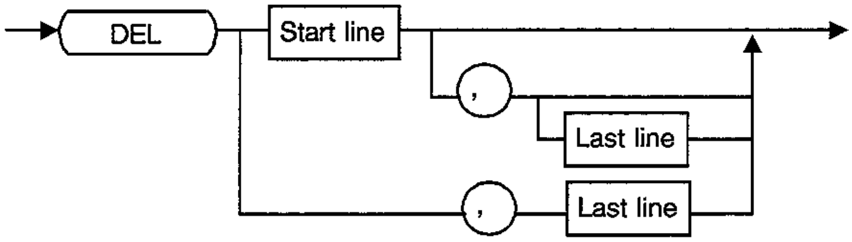
6. DEL

Outline

The DEL command is used to delete lines in the program.

Syntax

(1)-1



(1)-2
DEL <Start line [, [last line] >|<, last line>

Note: A space may be used instead of a comma.
The line number setting range is 1 through 65535.

Description

- The DEL command deletes the program from the start line to the last line.
- If the line number is omitted, the no operation will be performed.
- The DEL command cannot be used as a statement in the program.

Example

DEL 10	Deletes the 10th line only of the program.
DEL 10,	Deletes the program from line 10 to the last line.
DEL 10,100	Deletes the program from line 10 to line 100.
DEL , 100	Deletes the program from the start line to line 100.

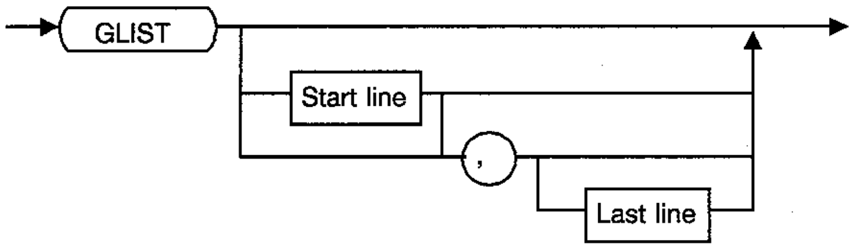
7. GLIST

Outline

The GLIST command is used to output a program list to peripheral devices such as a printer, etc. through the GPIB.

Syntax

(1)-1



(1)-2

GLIST [Start line [, [last line]]] | [, [last line]]

Note: A space may be used instead of a comma.
The line number setting range is 1 through 65535.
The label can be used instead of the line number.

Description

- The GLIST command outputs the BASIC programs list to peripheral devices such as a printer, etc. connected with the GPIB.
- The printer GPIB address can be define by the PRINTER statement or the panel key operation of R3764/66, R3765/67.
- SYSTEM CONTROLLER is made by the panel operation of the analyzer.

Example

GLIST	Outputs all lines of the program list.
GLIST 100	Outputs the 100th line only of the program list.
GLIST 100,	Outputs the program list from line 100 to the last line.
GLIST 100, 200	Outputs the program list from line 100 to line 200.
GLIST ,	Outputs all lines of the program list. (Same as GLIST)
GLIST , 200	Outputs the program list from the start line to line 200.

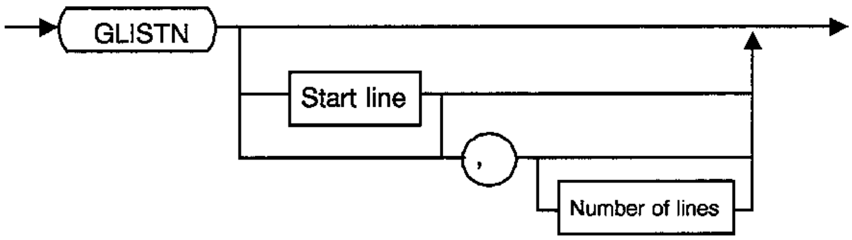
8. GLISTN

Outline

The GLISTN command is used to output a program list to peripheral devices such as a printer, etc. through the GPIB.

Syntax

(1)-1



(1)-2

GLISTN [Start line [, [number of lines]]] | [, [number of lines]]

Note: A space may be used instead of a comma.
The line number setting range is 1 through 65535.
The label can be used instead of the line number.

Description

- The GLISTN command outputs the BASIC programs list to peripheral devices such as a printer, etc. connected with the GPIB.
- The printer GPIB address can be define by the PRINTER statement or the panel key operation of R3764/67, R3765/67.
- SYSTEM CONTROLLER is made by the panel operation of the analyzer.
- The GLISTN command outputs specified lines of the program list from the start line number specified at the start line.
- When the line number is a negative value, this command outputs the program list toward the lower order numbers.

Example

GLISTN	Outputs all lines of the program list.
GLISTN 100	Outputs the 100th line only of the program list.
GLISTN 100,	Outputs the program list from line 100 to the last line.
GLISTN 100, 20	Outputs 20 lines of the program list from line 100.
GLISTN ,	Outputs all lines of the program list. (Same as GLISTN)
GLISTN , 20	Outputs 20 lines of the program list from the start line.

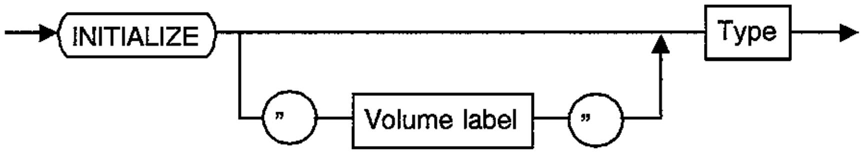
9. INITIALIZE (INIT)

Outline

The INITIALIZE command is used to initialize a floppy disk.

Syntax

(1)-1



(1)-2
INITIALIZE ["Volume label"] type

Description

- The INITIALIZE command initializes a new floppy disk or the floppy disk to be copied with the format specified by the floppy type setting.
- The volume label can be specified at the initialization.
  If omitted, there is no volume label
- Specify the types of floppy disks as follows:

Floppy type:	0;	720 KB (512 bytes, 9 sectors) 2DD
	1;	1.2 MB (1024 bytes, 8 sectors) 2HD
	2;	1.4 MB (512 bytes, 18 sectors) 2HD
	3;	1.2 Mbytes (512 bytes, 15 sectors) 2HD

CAUTION

The analyzer automatically discriminates between 2DD and 2HD disks.
If the different type (floppy disk) is inserted in the floppy disk drive, make sure to initialize it with the following default setting:

Default setting: 720 KB for 2DD (type 0)
1.2 MB for 2HD (type 1)

Note: For the information how to handle files, refer to "1. Preface • File Management".

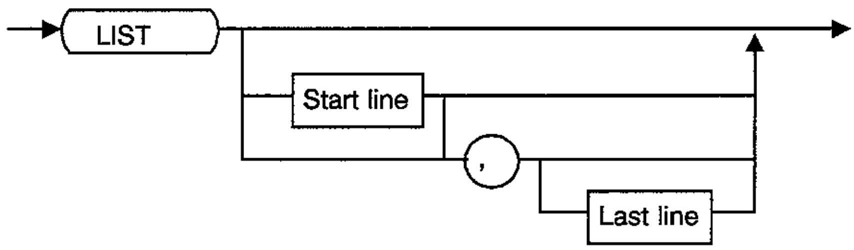
10. LIST

Outline

The LIST command is used to display a program list on the display.

Syntax

(1)-1



(1)-2

LIST [Start line [, [last line]]] | [, [last line]]

Note: A space may be used instead of a comma.
The line number setting range is 1 through 65535.
The label can be used instead of the line number.

Description

- The LIST command displays the BASIC program list specified by the parameters on the display.
- The display of the program list can be aborted using the STOP key. However, since the stop operation differs from the program operation, the program list cannot be re-displayed from the aborted line.

Example

LIST	Outputs all lines of the program list.
LIST 100	Outputs the 100th line only of the program list.
LIST 100,	Outputs the program list from line 100 to the last line.
LIST 100, 200	Outputs the program list from line 100 to line 200.
LIST ,	Outputs all lines of the program list. (Same as LIST)
LIST , 200	Outputs the program list from the start line to line 200.

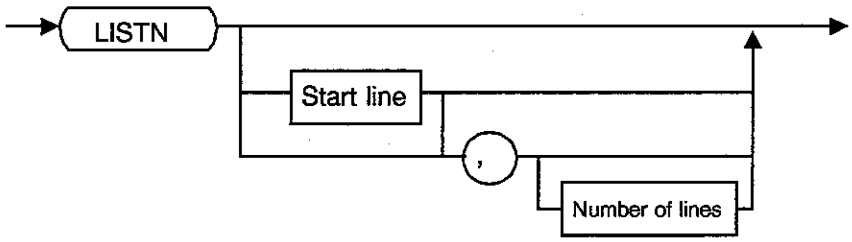
11. LISTN

Outline

The LISTN command is used to display a program list on the display.

Syntax

(1)-1



(1)-2

LISTN [Start line [, [number of lines]]] | [, [number of lines]]

Note: A space may be used instead of a comma.
The line number setting range is 1 through 65535.
The label can be used instead of the line number.

Description

- The LISTN command displays the BASIC program list specified by the parameters on the display.

Example

LISTN	Outputs all lines of the program list.
LISTN 100	Outputs the 100th line only of the program list.
LISTN 100,	Outputs the program list from line 100 to the last line.
LISTN 100, 20	Outputs 20 lines of the program list from line 100.
LISTN ,	Outputs all lines of the program list. (Same as LISTN)
LISTN , 20	Outputs 20 lines of the program list from the start line.

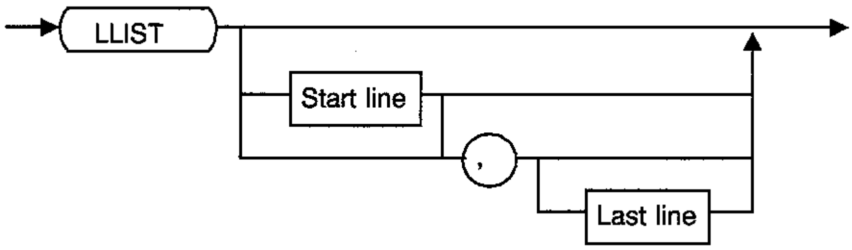
12. LLIST

Outline

The LLIST command is used to output a program list to peripheral devices such as a printer, etc. through the serial port.

Syntax

(1)-1



(1)-2

LLIST [Start line [, [last line]]] | [, [last line]]

Note: A space may be used instead of a comma.
The line number setting range is 1 through 65535.
The label can be used instead of the line number.

Description

- The LLIST command outputs the BASIC program list to peripheral devices such as a printer, etc. connected with the serial port.

Example

LLIST	Outputs all lines of the program list.
LLIST 100	Outputs the 100th line only of the program list.
LLIST 100,	Outputs the program list from line 100 to the last line.
LLIST 100, 200	Outputs the program list from line 100 to line 200.
LLIST ,	Outputs all lines of the program list. (Same as LLIST)
LLIST , 200	Outputs the program list from the start line to line 200.

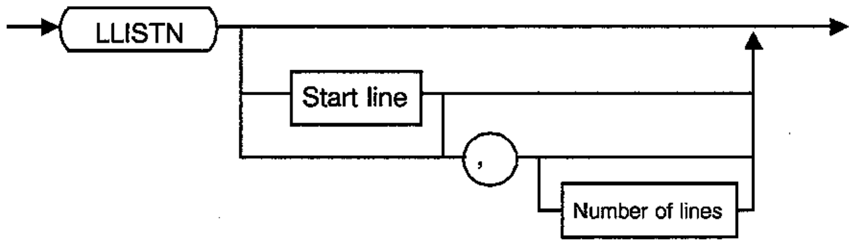
13. LLISTN

Outline

The LLISTN command is used to output a program list to peripheral devices such as a printer, etc through the serial port.

Syntax

(1)-1



(1)-2
LLISTN [Start line [, [number of lines]]] | [, [number of lines]]

Note: The line number setting range is 1 through 65535.
The label can be used instead of the line number.

Description

- The LLISTN command outputs the BASIC program list to peripheral devices such as a printer, etc. connected with the serial port.
- The LLISTN command outputs specified lines of the program list from the start line number specified at the start line.
- When the line number is a negative value, this command outputs the program list toward the lower order line numbers.

Example

LLISTN	Outputs all lines of the program list.
LLISTN 100	Outputs the 100th line only of the program list.
LLISTN 100,	Outputs the program list from line 100 to the last line.
LLISTN 100, 20	Outputs 20 lines of the program list from line 100.
LLISTN ,	Outputs all lines of the program list. (Same as LLISTN)
LLISTN , 20	Outputs 20 lines of the program list from the start line.

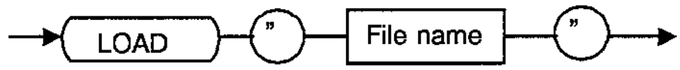
14. LOAD

Outline

The LOAD command is used to load the BASIC program file.

Syntax

(1)-1



(1)-2

LOAD "file name"

Description

- Loads the file specified by the file name. The files except BASIC must not be loaded.
- If there's no specification about the drive, loads from the current drive.
- If the program with no line number is loaded, the line number is attached automatically.

Note: For the information how to handle files, refer to "1. Preface ● File Management".

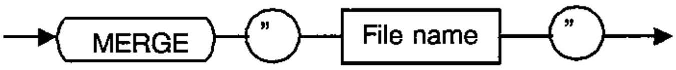
15. MERGE

Outline

The MERGE command is used to load the BASIC program file and overwrite onto the program in the memory.

Syntax

(1)-1



(1)-2
MERGE "file name"

Description

- The MERGE command differs from the LOAD command, since the BASIC buffer is not initialized before loading.
- The program already existing in the BASIC memory is not deleted unless the line number is the same.
- The program without line number cannot be loaded.
- The combination of the SCRATCH and MERGE commands represents the same function as the LOAD command.

Note: For the information how to handle files, refer to "1. Preface ● File Management".

16. PAUSE

Outline

The PAUSE command is used to pause (suspend) a program operation.

Syntax

(1)-1

(1)-2
PAUSE

Description

- The PAUSE command suspends the BASIC program temporally, or the BASIC program itself stops the program temporally.
- The program is restarted again at the next line of the suspended line by the CONT command.

Example

```
10  FOR I=1 TO 9
20    GOTO 60
30    GOTO *PRT
40  NEXT
50  PAUSE
60  !
70  X = I * I
80  GOTO 30
90  *PRT
100 PRINT I; "*" ;I; "=" ;X
110 GOTO 40
```

17. PRINTER

Refer to "44. PRINTER" in section 4.3.

18. PURGE

Outline

The PURGE command is used to purge files.

Syntax

(1)-1

(1)-2
PURGE "file name"

Description

- The PURGE command is used to purge files. Note that the purged files cannot be restored.
- If there's no specification about the drive, the object drive is the current one.

Note: For the information how to handle files, refer to "1. Preface ● File Management".

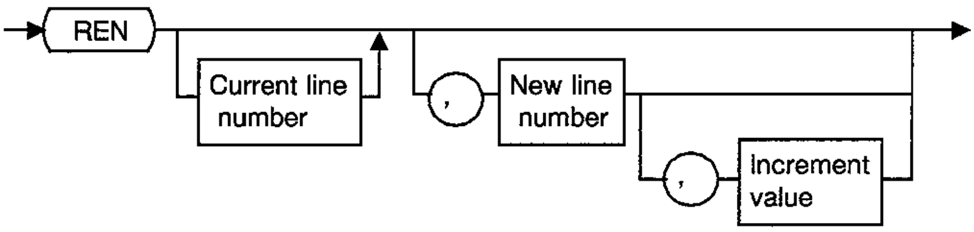
19. REN

Outline

The REN command is used to renew the line numbers of program.

Syntax

(1)-1



(1)-2

REN [[Current line number] [, New line number [, Increment value]]]

Note: A space may be used instead of a comma.
The setting range of the current line number, the new line number and the increment value is 1 through 65535.

Description

- The current line number specifies the head of the line number to be renewed in the current program.
- The new line number specifies the start of the renewed line number.
- The increment value specifies the step of the renewed line number.
- The REN command renews the line number used in the GOTO and GOSUB statements corresponding to the new line number.
- The REN command cannot be used to specify the line number exceeds 65535. Do not specify the program line with changing/modifying the order.

Example

REN:	Renews the start line to 10, and changes the line number by 10 steps till the last line.
REN 30, 50, 3:	Renews the line number 30 to 50, and changes the line number by 3 steps till the last line.

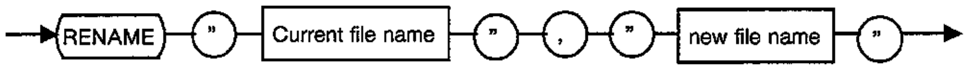
20. RENAME

Outline

The RENAME command is used to rename the file name stored on a drive.

Syntax

(1)-1



(1)-2

RENAME "current file name", "new file name"

Description

- The RENAME command renames only the file name stored without changing its contents.
- If the same file exists in a floppy which has already been created, then no operation will be performed.
- RENAME cannot be executed between the different drives. If there is no specification about the drive, the object drive is the current one.

Note: For the information how to handle files, refer to "1. Preface ● File Management".

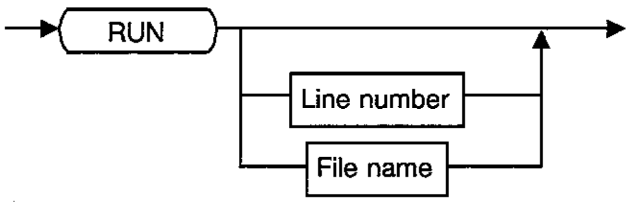
21. RUN

Outline

The RUN command is used to execute the BASIC program.

Syntax

(1)-1



(1)-2

RUN [line number | file name]

Description

- The RUN command executes the BASIC program from the specified line.
- If no line number is specified, the program will be executed from the start line.
- If a file name is specified, the program will be executed after the specified file loaded. The start line cannot be specified.
- When the RUN command is executed, all the variables are cleared and also the array declarations are forcibly cleared before program execution.

Example

RUN

RUN 200

22. SAVE

Outline

The SAVE command is used to save the BASIC program files.

Syntax

(1)-1

(1)-2

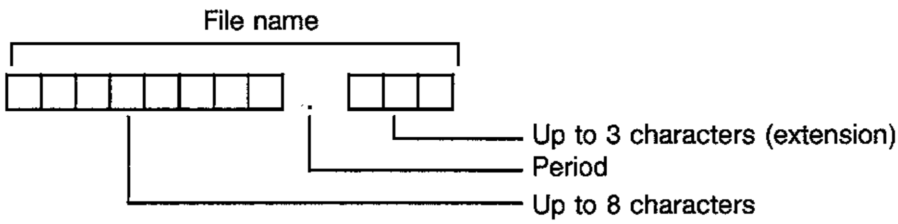
SAVE "file name"

Description

- The SAVE command stores the program (stored in the memory) into the file specified in the statement.
- If the already existed file name is specified, the specified file is assumed to update, then the file is overwritten.
- If there's no specification about the drive, the object drive is the current one.

CAUTION

The file name uses numerics, alphabets and symbols (except for double quotations), and specify the file name as follows:



Use .BAS as much as possible for the extension.

Note: For the information how to handle files, refer to "1. Preface ● File Management".

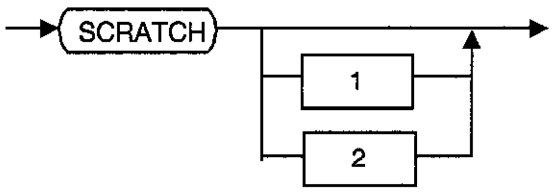
23. SCRATCH

Outline

The SCRATCH command is used to scratch (erase) the BASIC program stored in the memory.

Syntax

(1)-1



(1)-2

SCRATCH [1|2]

Example

SCRATCH:	Erases all the programs stored in the BASIC buffer.
SCRATCH 1:	Initializes the program data only stored in the BASIC buffer.
SCRATCH 2:	Initializes the program procedure only stored in the BASIC buffer.

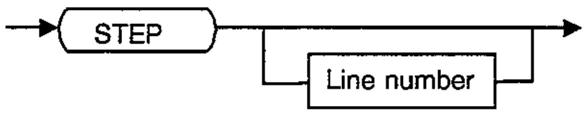
24. STEP

Outline

The STEP command is used to execute the only one line of the BASIC program.

Syntax

(1)-1



(1)-2

STEP [line number]

Description

- The STEP command executes the only one line of the BASIC program, however, no operation will be performed in the FOR statement.
- If the line number is omitted, the next line of currently suspended line is performed.

Example

```
STEP

STEP 100
```

25. STOP

Outline

The STOP command is used to stop the BASIC program.

Syntax

(1)-1

(1)-2
STOP

Description

- The STOP command stops the BASIC program execution or the BASIC program itself stops the program execution.

4. BASIC STATEMENT

4.1 Programming Rules

4.1.1 Program Structure

(1) Statement

The BASIC program consists of various statements.
The statements are grouped into two types; control statement and executable statement.
Each statement consists of key words and expressions. The decision of the construction is the syntax rule for grammar.

- Statement
  - Control statement
    - Declaration statement
    - Loop/branch statement
    - Instruction interrupt processing instruction
  - Executable statement
    - Execution command
    - Editing command
    - I/O execution command
    - File command *
    - GPIB execution command
    - Operational statement

*: Describes in "Chapter 3. BASIC Command".

(2) Key word

The term whose meaning and application are predetermined with BASIC is called a "key word". The same name as the key word cannot be used for any other purpose.
The key word that is frequently used and whose full name is long has a short name.
To change the appearance from the full name to the short name, CONTROL command should be used to set the control register 3 should be set to "0".

For information of key word list, refer to Table 4-1.
The relationship between the full and short names is shown in Table 4-2.

Table 4-1 Key Word List

AND	APPEND	AS	ASCII	BAND	BASIC (*)
BINARY	BNOT	BOR	BREAK	BUZZER	BXOR
CASE	CAT	CHKDSK	CIRCLE (*)	CLEAR	CLOSE
CLS	CMD	COLOR (*)	CONSOLE	CONT	CONTINUE
CONTROL	COPY	DELAY	COUNT	CSR	CURSOR
DATA	DEL	ELSE	DELIMITER	DIM	DISABLE
DSTAT	DUMP	ERROR	ENABLE	END	ENT
ENTER	GLISTN	GOSUB	EVENT	FOR	FORMAT
GLIST	INITIALIZE	INP	GOTO	GPRINT	IF
INIT	ISRQ	KEY	INPUT	INTEGER	INTERFACE
INTR	LISTEN	LISTN	LABEL (*)	LINE (*)	LINETYPE (*)
LIST	LPRINT	LOAD	LLIST	LLISTN	LOCAL
LOCKOUT	NOT	OFF	MERGE	MOVE (*)	NEXT
OUTPUT	OUT	PRF	ON	OPEN	OR
PRINT	PRINTER	RENAME	PAUSE	PEEK	POKE
RESTORE	PURGE	RUN	PRINTF	READ	RECTANGLE (*)
REQUEST	RETURN	SRQ	REM	REMOTE	REN
SEND	SPRINTF	THEN	SAVE	SCRATCH	SELECT
TALK	TEXT	UNTIL (*)	STEP	STOP	SYSTEM (*)
UNL	UNT		TIME	TO	TRIGGER
WAIT	XOR		USE	USING	VIEWPORT (*)

Note: A capital letter is used for keyword.

(*) : They are the reserved keywords. Though they are not used, they cannot be used for variable names.

Table 4-2 Correspondence Table between Full Name and Short Name

Full Name	Short Name
CURSOR	CSR
ENTER	ENT
INITIALIZE	INIT
INPUT	INP
OUTPUT	OUT
PRINTF	PRF
USING	USE
PRINT	?

(3) Expression

The expression consists of the object and operator and can be placed anywhere it can be grammatically specified to. (However, since the condition expression of 1F statement interpret the symbol "-" as equal sign because of the compatibility with the conventional BASIC, the assignment expression cannot be written.)
There are three kinds of expressions, depending on which kinds of data type is used for the final value as a result of computation.

<arithmetic expression> <character string expression> <logical expression>

Arithmetic expression:	Results in an integer value or real value,
logical expression:	Is determined by the syntax regardless of whether the expression includes the logical operator within itself and estimates the final value as logical value, i.e., "0" is false and "1" is true.

4.1.2 Object

The item to be processed by BASIC is called "object". The object may be a constant, variable, and function and each object type consists of:

- Object
  - (1) constant
    - integer number constant
    - real number constant
    - character string constant
    - Line number
    - Label
  - (2) variable
    - ① scalar variable
      - integer number variable
      - real number variable
      - character string variable
    - ② system variable
      - current line variable
      - built in variable
    - ③ array
      - integer number
      - real number
    - ④ File descriptor
  - (3) function

(1) Constant

●Integer number constant

The constant which has no decimal point within a program is considered as an integer number. Since the constant is represented using four bytes inside, it can range from -2,147,483,648 to +2,147,483,647.

•Real number constant

The constant which has a decimal point or is represented using a floating decimal point such as 1E+20 is considered as a real number. Since the constant is represented using eight bytes (1EEE) inside, it can range from approx. -1E+308 to approx. 1E+308 and has an accuracy of 15 digits.

•Character string constant

To represent a character string, it must be enclosed with double quotation marks ("). It is possible to specify any character string between the empty string "" and a maximum of 128 character string. The unit of the included character is 8 bits and it is possible to represent up to 256 kinds of character units of 0 to 255. ASCII codes are used as character codes, which register special symbols to codes from 128 to 255.

For the program to represent the codes which are not assigned to the keyboard or to enter the INPUT statement, the form field (\f) method is prepared using "\". Similarly, "\" can be written to include the double quotation mark " into the character string.

To represent the ASCII control characters, escape sequences are prepared, as follows:

Escape sequences	Meanings	total number	Decimal number
\b	Back space	010	8
\t	Horizontal TAB	011	9
\n	Line field (new line)	012	10
\v	Vertical TAB	013	11
\f	Form field (clear screen)	014	12
\r	Carriage return	015	13

•Line number

Line number is shown by integer 1 to 65535, and specifies the line of the BASIC program.

•Label

Label can be used instead of the line number. For declaration, an asterisk (*) should be added to the beginning of the program.

The usable character is the same as the variable. However, since it is not a variable, any character cannot be substituted. In addition, the positions where the label can be written are limited to the line number part described in "4.3 Statement Syntax and Use" or the part where "label" is written.

(2) Variable

The name of variable consists of up to 20 alphanumeric characters, starting with an alphabetic character.

If the last character of the variable name is $:	Character string variable
If the last character is (integer):	Array type variable
	If INTEGER statement does not declare the variable type, the variable is used as a real number type.

Table 4-3 Alphanumeric Characters

1, 2, 3, 4, 5, 6, 7, 8, 9, 0
a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, l, r, s, t, u, v, w, x, y, z
A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z
-

Example: Variable types

value, v123:	Real number variable
string$, s123$:	Character string variable
array(3):	Array type real number variable
INTEGER code:	Integer variable
INTEGER week(7):	Array type integer number variable

① Scalar variable

- Integer number variable
- Real number variable
- Character string variable

As long as the variable is not initialize, "0" is assigned to the numeric type variable. Therefore, if the variable is to be initialized to a specific value, it is necessary to specifically substitute a value in the program.
The value which can be stored each data type has the same amplitude as for the constant. The character string variable does not have the array. The character string has the length attribute similarly to the character string constant. To declare the length, DIM statement should be used.

DIM string$[100]

If the reference is made without the declaration, the variable is considered as 18 character string. A part of the character string can be handled using the sub-string operator ([ ]).

Refer to "(7) Sub-string operator" in section 4.1.3.

```
string$ = "ADVANTEST CORPORATION"
PRINT string$[1,14] ; "."
```

Result

```
ADVANTEST CORP.
```

② System variable

- Current line variable @
  Stores the line number of the program which is currently performed. Any value cannot be substituted.
  LIST @: Displays the line currently performed.

- Built -in variable
  Is the variable which is automatically registered when the BASIC starts. The variable is initialized to a specific value and can be changed by substituting a specific value. To return it to the value when the BASIC starts, substitute that value specifically or initialize the BASIC with SCRATCH 1,SCRATCH.

  PI: 3.14159.....
  EXP: 2.71828.....

③ Array

For declaration of the array, use DIM, INTEGER statement.

- Numeric value type array
  If the reference is made without any declaration, the amplitude of that array (number of elements) is 10 as shown in the declaration below. The attached character is always assigned starting at 1.

```
DIM     array(10)
INTEGER array(10)
```

Real number type array	DIM real(20)
Integer number type array	INTEGER int(30,40)

④ File descriptor

The BASIC reads and writes files by using the file descriptor. Declaration is not necessary, but OPEN connects to the real file name. After OPENed, specify the file descriptor by using ENTER or OUTPUT to refer to the file. Since the file descriptor is a special variable, it cannot perform operations or print like other variables can.

(3) Functions

All the functions are built-in type and grouped into the integer number type, real number type, and character string type, depending on its return value. In addition, since the function call can be written in an operation expression, it can be handled similarly to the variable.

```
string$ = "ADVANTEST"
PRINT string$
A = NUM("A")
a = NUM("a")
FOR idx = 1 to LEN(string$);
      b = NUM(string$[idx;1]) - A + a
       string$[idx;1]=CHR$(b)
NEXT idx
PRINT STRING$
```

Result

```
ADVANTEST
advantest
```

- Built-in functions

Functions	Descriptions
SIN (Arithmetic expression) COS (Arithmetic expression) TAN (Arithmetic expression) ATN (Arithmetic expression)	Sine (sin) Cosine (cos) Tangent (tan) Reverse tangent (tan^{-1}) Unit of angle = radian
LOG (Arithmetic expression)	Natural logarithm
SQR (Arithmetic expression)	Square root
ABS (Arithmetic expression)	Absolute value
NUM (Character string expression)	Returns ASCII code for the first one character of the character string expression. Example: NUM ("A")---> 65
CHR$ (Arithmetic expression)	Returns the character string of the ASCII code one character corresponding to the value of the arithmetic expression. Example: CHR$ (65)---> "A"
LEN (Character string expression)	Returns the length of the character string expression. Example: LEN ("ADVANTEST")---> 9
POS (Arithmetic expression 1, Arithmetic expression 2)	Returns the digit of the head character of the character string corresponding to the character string expression 2 in the character string expression 1. Example: POS ("ADVANTEST", "AN")---> 4
Built-in functions	Functions to handle the measurement value For details, refer to "4.4 Built-in Functions".

Though there is no built-in function to convert from character string to numeric variable and from numeric variable to character string, the conversion can be performed by assignment statement.

Example: A$ = A
         A = "123.4"

4.1.3 Operators

Operator are used to operate the object operand. An expression is coded by combining operators and objects.

Operators
- (1) Assignment operators
- (2) Unary arithmetic operators
- (3) Dyadic arithmetic operators
- (4) Logic operators
- (5) Bit processing operators
- (6) Relational operators
- (7) Sub-string operators

(1) Assignment operators

The key word existed in the standard BASIC, which is called "LET" is not provided for the assignment operator. Assignment expression contains has its values and and makes up an expression.

```
PRINT a=1              ---> 1.0
PRINT a$="ADVANTEST"   ---> "ADVANTEST"
PRINT (a=1)+a          ---> 2.0
```

The assignment operators are shown below:

=: Normal assignment
In the assignment for character-string variables, transmits the only effective value of right part.
Example: DIM string$ [20]
PRINT LEN (string$ = "12345")
Result
5

=: Converts the value depending on the data type of left part, then assigns it to variable.
Example: string$ = 123.456 --->"123.456"
numeric = "123" --->123
integer = 123.456 --->123

```
+=: a += 10    ---> a = a + 10
-=: a -= 10    ---> a = a - 10
*=: a *= 10    ---> a = a * 10
/=: a /= 10    ---> a = a / 10
%=: a %= 10    ---> a = a % 10
```

=<: Assigns the character strings left-justify to variables.
=>: Assigns the character strings right-justify to variables.

(2) Unary arithmetic operators

-: Minus sign
+: Plus sign
++: Front/Back Increment
Front b = ++a ... Adds 1 to a, then assigns ++a to b.
Back b = a++ ... Assigns a++ to b, then adds 1 to a.
--: Front/Back Decrement
Front b = --a ..... Subtracts 1 from a, then assigns --a to b.
Back b = a-- ..... Assigns a-- to b, then subtracts 1 from a.

Example: a = 10: PRINT a++: PRINT a: PRINT --a: PRINT --a: print a

Result
10.0
11.0
10.0
9.0
9.0

Note: The operations of front/back increment-decrement cannot be performed to the constant (real constant, integer constant).

(3) Dyadic arithmetic operators

+: Addition
-: Subtraction
*: Multiplication
/: Division
%: Modulo calculation (remainder)
^: Involution
&: Coupling characters

(4) Logic operators

NOT	Example	NOT 1	Result	0
AND	Example	1 AND 0	Result	0
OR	Example	1 OR 0	Result	0
XOR	Example	1 XOR 0	Result	0

(5) Bit processing operators

In numeric expressions, only the integer type is available. Real type may result in an error.

BNOT	Example	BNOT 1	Result	-1
BAND	Example	2 BAND 3	Result	2
BOR	Example	2 BOR 3	Result	3
BXOR	Example	2 BXOR 3	Result	1

(6) Relational operators

The following operators are provided, and the result of applying these operators is a boolean value, either TRUE or FALSE. At this case, TRUE is 1, and FALSE is 0. When the relational operation is resulted based on the BASIC syntax, if the value calculated finally resulted in 0, the result is determined as FALSE. All the values other than calculated values become TRUE.

=: Equal
< >: Not equal (or ! =)
<
>
< =
> =

Since the relational operations always perform the arithmetic operation according to the IF statement condition, the operator " = " is determined unconditionally as relational operator. Therefore, the assignment expression cannot be included in the IF statement conditional expression.

(7) Sub-string operators

Enables to specify the character-string expression in part as character string.

Character-string expression [arithmetic expression 1, arithmetic expression 2]:
The sub-string operator is considered (defined) as from.
"ADVANTEST" [1,5] ---> "ADVAN"

Character-string expression [arithmetic expression 1, arithmetic expression 2]:
The sub-string operator is considered (defined) as from.
"ADVANTEST" [6;4] ---> "TEST"

4.2 Various Statements

4.2.1 Statement Function List

(1) Basic (fundamental) statement

Statement	Function
BUZZER	Sounds the buzzer.
CLS	Clears the screen.
CONSOLE	Specifies the scroll area.
CURSOR	Moves the cursor.
DATA	Defines the numeric value or character string to be read out by READ statement.
DATE$	Reads out the date of timer (RTC) built into the analyzer.
DIM	Defines the array variable or character-string variable.
DISABLE INTR	Disables the acceptance of the interruption.
ENABLE INTR	Enables the acceptance of the interruption.
ERRM$	Returns the error message.
ERRN	Returns the error number.
FOR-TO-SETP, NEXT, BREAK, CONTINUE	Executes the loop processing.
FRE	Returns the BASIC program memory remaining capacity.
GOSUB, RETURN	Branches or returns to the subroutine.
GOTO	Branches to the specified line.
GPRINT	Outputs to the numeric value or character string to the GPIB.
IF-THEN, ELSEL END IF	Conditional branch
INPUT	Inputs from the panel key.
INTEGER	Defines the variable as an integer type.
KEY$	Returns the panel key code of the analyzer.
LPRINT	Outputs the numeric value or character string to the serial port.
LET	Substitutes the expression for variable.
OFF ERROR	Cancels the branch when detecting the BASIC error.
OFF ISRQ	Cancels the interruption branch by ISRQ.
OFF KEY	Cancels the interruption branch by key input.
OFF SRQ	Cancels the interruption branch by SRQ.
ON DELAY	Branches after the specified time elapses.
ON ERROR	Defines the branch when detecting the BASIC error.
ON ISRQ	Defines the interruption branch by the internal request.
ON KEY	Defines the interruption branch by key input.
ON SRQ	Defines the interruption branch by externally GPIB SRQ.

(Cont'd)

Statement	Function
PRINT [USING]	Displays the numeric value or character string.
PRINTER	Sets the printer GPIB address.
PRINTF	Displays the numeric value or character string.
READ	Assigns the constant of DATA statement to the variable.
REM	Annotation
RESTORE	Specifies the data line to be read in next READ statement.
SELECT, CASE, END SELECT	Executes the multi branches with condition of expression value.
SPRINTF	Assigns the result according to PRINTF format to the character string.
TIME$	Returns the value of timer (RTC) built into the analyzer.
TIMER	Reads out and resets the value of the built-in system timer.
WAIT	Waits for the specified time.
WAIT EVENT	Waits for the occurrence of the specified event.

(2) GPIB control statement

Statement	Function
CLEAR	Clears the device.
DELIMITER	Specifies the block delimiter.
ENTER	Inputs from the GPIB.
INTERFACE CLEAR	Clears the GPIB interface.
LOCAL	Cancels the remote control.
LOCAL LOCKOUT	Local lockout
OUTPUT	Outputs to the GPIB.
REMOTE	Remote control
REQUEST	Sets the status byte.
SEND	Outputs (sends) the command, data, and others to the GPIB.
SPOLL	Reads out the status byte.
TRIGGER	Outputs the group-execute trigger.

(3) File control statement

Statement	Function
CLOSE	Closes the file.
DSTAT	Obtains the directory contents of floppy disk for the BASIC variable.
ENTER [USING]	Reads out the data from the file.
OFF END	Cancels the processing specified by ON END statement.
ON END	Defines the processing at the end of file.
OPEN	Opens the file.
OUTPUT [USING]	Outputs (writes) the data to the file.

4.2.2 Statement Syntax List

(1) Basic statement

<table>
<tr><th>Statement</th><th>Syntax</th></tr>
<tr><td>BUZZER</td><td>BUZZER <tone><time></td></tr>
<tr><td>CLS</td><td>CLS</td></tr>
<tr><td>CONSOLE</td><td>CONSOLE <start line><last line></td></tr>
<tr><td>CURSOR</td><td>CURSOR <X axis><Y axis></td></tr>
<tr><td>DATA</td><td>DATA numeric constant | character-string constant
{, numeric constant | character-string constant}</td></tr>
<tr><td>DATE$</td><td>(1) DATE$
(2) DATE$ = "YY/MM/DD"</td></tr>
<tr><td>DIM</td><td>DIM <B>|<C> {, <B>|<C>}</td></tr>
<tr><td>DISABLE INTR</td><td>DISABLE INTR</td></tr>
<tr><td>ENABLE INTR</td><td>ENABLE INTR</td></tr>
<tr><td>ERRM$</td><td>ERRM$ (error number)</td></tr>
<tr><td>ERRN</td><td>ERRN</td></tr>
<tr><td>FOR-TO-SETP, NEXT,
BREAK, CONTINUE</td><td>FOR numeric variable = numeric expression TO
numeric expression [STEP numeric expression]
[BREAK]
[CONTINUE]
NEXT [numeric variable]</td></tr>
<tr><td>FRE</td><td>FRE (numeric)</td></tr>
<tr><td>GOSUB, RETURN</td><td>GOSUB line number | label expression
RETURN</td></tr>
</table>

B: numeric variable name [(numeric expression {, numeric expression})]
C: character-string variable [numeric expression]

(Cont'd)

<table>
<tr><th>Statement</th><th>Syntax</th></tr>
<tr><td>GOTO</td><td>GOTO line number | label</td></tr>
<tr><td>GPRINT</td><td>GPRINT [A {, | ;A}]</td></tr>
<tr><td>IF-THEN, ELSEL END IF</td><td>(1) IF<conditional expression> THEN <statement>
(2) IF<conditional expression> THEN
[ELSE IF <conditional expression> THEN]
[multi statements]
[ELSE]
[multi statements]
END IF</td></tr>
<tr><td>INPUT</td><td>INPUT ["<character-string>",] A {, A}</td></tr>
<tr><td>INTEGER</td><td>INTEGER <B> {, <B>}</td></tr>
<tr><td>KEY$</td><td>KEY$</td></tr>
<tr><td>LPRINT</td><td>LPRINT [A {, | ;A}]</td></tr>
<tr><td>LET</td><td>LET <D>|<E> {:<D>|<E>}</td></tr>
<tr><td>OFF ERROR</td><td>OFF ERROR</td></tr>
<tr><td>OFF ISRQ</td><td>OFF ISRQ</td></tr>
<tr><td>OFF KEY</td><td>OFF KEY [key code]</td></tr>
<tr><td>OFF SRQ</td><td>OFF SRQ</td></tr>
<tr><td>ON DELAY</td><td>ON DELAY time GOTO | GOSUB line number | label</td></tr>
<tr><td>ON ERROR</td><td>ON ERROR GOTO | GOSUB line number | label</td></tr>
<tr><td>ON ISRQ</td><td>ON ISRQ GOTO | GOSUB line number | label</td></tr>
<tr><td>ON KEY</td><td>ON KEY key code GOTO | GOSUB line number | label</td></tr>
<tr><td>ON SRQ</td><td>ON SRQ GOTO | GOSUB line number | label</td></tr>
<tr><td>PRINT [USING]</td><td>(1) PRINT [A {, | ;A}]
(2) PRINT USING format setup expression ; {, A}</td></tr>
<tr><td>PRINTER</td><td>PRINTER numeric expression</td></tr>
<tr><td>PRINTF</td><td>PRINTF format expression {, A}</td></tr>
<tr><td>READ</td><td>READ input item {, input item}</td></tr>
<tr><td>REM</td><td>REM [character string] or ![character string]</td></tr>
<tr><td>RESTORE</td><td>RESTORE line number | label</td></tr>
</table>

A: numeric expression | character-string expression

B: numeric variable name [(numeric expression {, numeric expression})]

D: numeric variable = Numeric expression

E: character-string variable = | = <| = >character-string expression

(Cont'd)

Statement	Syntax
SELECT, CASE, END SELECT	SELECT <numeric expression | character-string expression> CASE <numeric expression | character-string expression> multi statements [CASE ELSE] [multi statements] END SELECT
SPRINTF	SPRINTF character-string variable format specification {, A}
TIMER	TIMER (0|1)
TIME$	(1) TIME$ (2) TIME$ = "HH:MM:SS"
WAIT	WAIT time
WAIT EVENT	WAIT EVENT <event number>

A: numeric expression | character-string expression

- In PRINT USING format specification, specify the following image specifications by using a comma among images.
  image specifications

D:	Specifies the output digits with No. of D. A space is used to fill up the remaining blank in the specified field.
Z:	Specifies the output digits with No. of Z. A zero is used to fill up the remaining blank in the specified field.
K:	Displays the expression as it is.
S:	Displays the PRINT USING format with a + or - sign flag at the position of S.
M:	Displays the PRINT USING format with a - for negative and a space for positive at the position of M.
.:	Displays the PRINT USING format to match the position "." with coming the decimal point.
E:	Displays PRINT USING format with the exponent format (e, sign, exponent).
H:	Same as K. However, use a comma for a decimal point.
R:	Same as ".". However, use a comma for a decimal point.
*:	Specifies the output digits with the number of *. A space is used to fill up the remaining blank in the specified field.
A:	Displays one character.
k:	Displays the character-string expression as it is.
X:	Displays the character of one space.
Literal:	Encloses a literal with \" when writing it to the format expression.
B:	Displays the expression result using an ASCII code.
@:	Form lead
+:	Moves the display position to the top of the same line.
-:	Line feed
#:	Does not line feed.
n:	Specifies the number of repetition of each image by using numerics.

- In PRINTF format specification, specify the parameter immediately followed after % by using the following image.

  %[-] [0] [m] [. n] character

  -: Justifies the character with no space from left (if no specification, then from right).

  0: Sets the character, which is justified for the remaining blank in the specified field, to be 0.

  m: Reserves the field for the character "m".

  .n: Outputs the PRINT USING format with n-digit accuracy. In character string, this setup value is used for an actual character-string length.

  Character: d; decimal with sign, s; character string, o; octal, e; floating-point expression (exponent format), x; hexadecimal, f; floating-point expression

(2) GPIB statement

<table>
<tr><th>Statement</th><th>Syntax</th></tr>
<tr><td>CLEAR</td><td>CLEAR [unit address {, unit address}]</td></tr>
<tr><td>DELIMITER</td><td>DELIMITER numeric expression</td></tr>
<tr><td>ENTER</td><td>ENTER unit address ; B {, B}</td></tr>
<tr><td>INTERFACE CLEAR</td><td>INTERFACE CLEAR</td></tr>
<tr><td>LOCAL</td><td>LOCAL [unit address {, unit address}]</td></tr>
<tr><td>LOCAL LOCKOUT</td><td>LOCAL LOCKOUT</td></tr>
<tr><td>OUTPUT</td><td>OUTPUT unit address {, unit address} ;A {, A}</td></tr>
<tr><td>REMOTE</td><td>REMOTE [unit address {, unit address}]</td></tr>
<tr><td>REQUEST</td><td>REQUEST integer</td></tr>
<tr><td>SEND</td><td>SEND <C>|<D> {, <C>|<D>}</td></tr>
<tr><td>SPOLL</td><td>SPOLL (unit address)</td></tr>
<tr><td>TRIGGER</td><td>TRIGGER [unit address {, unit address}]</td></tr>
</table>

A: numeric expression | character-string expression

B: numeric variable [character-string expression

C: <CMD|DATA|LISTEN|TALK> [numeric expression {, numeric expression}]

D: UNL|UNT

(3) File control statement

<table>
<tr><th>Statement</th><th>Syntax</th></tr>
<tr><td>CLOSE</td><td>CLOSE #FD | *</td></tr>
<tr><td>DSTAT</td><td>(1) DSTAT 0 <number of file>
(2) DSTAT <index> <file name> <attribute> <size> <number of sector> <year> <month> <date> <time> <minute> <start sector>
(3) DSTAT ;SELECT <character string> COUNT <variable></td></tr>
<tr><td>ENTER [USING]</td><td>(1) ENTER #FD ; input item {, input item}
(2) ENTER #FD USING "image specification" ; input item {, input item} }</td></tr>
<tr><td>OFF END</td><td>OFF END #FD</td></tr>
<tr><td>ON END</td><td>ON END #FD GOTO|GOSUB integer|label expression</td></tr>
<tr><td>OPEN</td><td>OPEN "file name" FOR processing mode AS #FD [; type]</td></tr>
<tr><td>OUTPUT [USING]</td><td>(1) OUTPUT #FD ; output item {, output item}
(2) OUTPUT #FD USING "image specification" ; output item {, output item} }</td></tr>
</table>

FD: file descriptor
Processing mode: INPUT|OUTPUT
Type: BINARY|TEXT|ASCII

- ENTER USING image specification

  image specification

  D: Interprets the numeric of D as an input digit and reads out it, then assigns it to the variable of the input item.

  Z: Same as D.

  K: Reads one line and converts it to the numeric data, then assigns it to the variable of the input item.

  S: Same as D.

  M: Same as D.

  .: Same as D.

  E: Same as K

  H: Same as K. However, use a comma for a decimal point.

  *: Same as D.

  A: Reads the number of A and assigns it to the character-string variable.

  k: Reads one line and assigns it to the character-string variable.

  X: Skips one character.

  Literal: Skips the the character-string numeric data enclosed with \".

  B: Reads one character and assigns it to the input item using an ASCII code.

@: Skips one-byte data.
+: Same as @.
-: Same as @.
#: Ignored in ENTER statement.
n: Specifies the number of repetition of each image by using numerics.

- OUTPUT USING image specification

image specification

D: Specifies the output digits with No. of D. A space is used to fill up the remaining blank in the specified field.
Z: Specifies the output digits with No. of Z. A zero is used to fill up the remaining blank in the specified field.
K: Displays the expression as it is.
S: Displays the OUTPUT USING with a + or - sign flag at the position of S.
M: Displays the OUTPUT USING with a - for negative and a space for positive at the position of M.
.: Displays the OUTPUT USING to match the position "." with coming the decimal point.
E: Displays OUTPUT USING with the exponent format (e, sign, exponent).
H: Same as K. However, use a comma for a decimal point.
R: Same as ".". However, use a comma for a decimal point.
*: Specifies the output digit with the number of *. A space is used to fill up the remaining blank in the specified field.
A: Displays one character.
k: Displays the character-string expression as it is.
X: Displays the character of one space.
Literal: Encloses the literal with \" when writing it in the format expression.
B: Displays the expression result using an ASCII code.
@: Outputs the form lead.
+: Outputs the carriage return.
-: Outputs the line feed.
#: Does not hang the line feed immediately followed after the last item.
n: Specifies the number of repetition of each image by using numerics.

4.3 Statement Syntax and Use

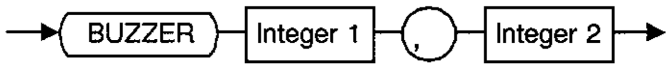
1. BUZZER

Outline

The BUZZER statement is used to sound alarm.

Syntax

(1)-1



(1)-2
BUZZER integer 1 integer 2

Note: An integer 1 is used to specify the tone at the range of 0 (high tone) to 65535 (low tone).
An integer 2 is used to specify the duration (unit: ms).

Description

- The BUZZER statement sounds the buzzer built into the analyzer in accordance with the specified range.

Example

```
10 FOR I=0 TO 255
20   BUZZER I, 10
30 NEXT I
40 STOP
```

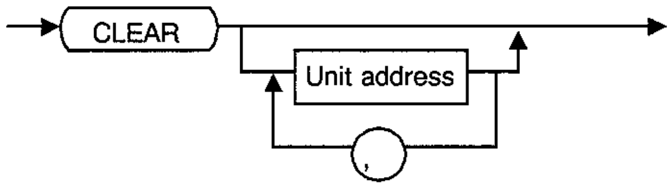
2. CLEAR

Outline

The CLEAR statement is used to set the all units connected to a GPIB or the selected particular units to an initial state. In other word, this statement clears the all setup values for units.

Syntax

(1)-1



(1)-2
CLEAR [unit address {, unit address}]

Description

- If only the CLEAR statement is performed without specifying the unit address, the universal Device Clear (DCL) command will be sent. By the DCL command, all the units, which is connected to a GPIB, could be set to the initial state.
- When the unit address is specified followed after the CLEAR statement, only the units which are specified by the unit address are addressed, then the Select Device Clear (SDC) command is sent. By the SDC command, only the particular units is set to the initial state. Multiple unit-address can be specified.
- The initial state that is defined for each unit in the CLEAR statement depends on each unit.

Example

```
10 CLEAR
20 CLEAR 2
30 CLEAR 1, 3, 5, 7
```

Note

The CLEAR statement is not available in ADDRESSABLE mode.

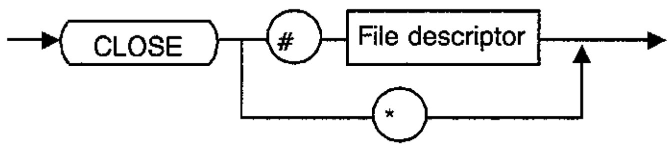
3. CLOSE

Outline

The CLOSE statement is used to close files assigned to a file descriptor.

Syntax

(1)-1



(1)-2

CLOSE <#file descriptor| *>

Description

- All files opened by the OPEN command must be closed before removing a floppy disk or turning off the power of units. If not, the files may be damaged.
- In BASIC program, when operation is suspended using the PAUSE or STOP key, files are not closed automatically. In other cases, all files are closed automatically after programming, also after termination with an error. However, if ON ERROR is set in instrument, the files will not be closed. By reasons above, be sure to perform the close operation certainly by using the following method (specification method for closing all files using the command) at the error termination.

  CLOSE *

- The files are closed automatically when command such as SCRATCH or LOAD is executed.

Note: For the information how to handle files, refer to "1. Preface ● File Management".

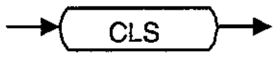
4. CLS

Outline

The CLS statement is used to clear the display on the screen.

Syntax

(1)-1



(1)-2
CLS

Description

- The CLS statement clears the characters displayed on the screen and immediately returns the cursor to the original position.
- The CLS statement clears the scroll range specified by CONSOLE.

Example

```
10 CLS
```

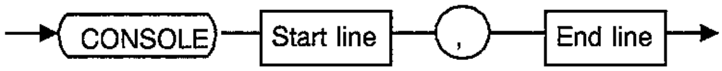
5. CONSOLE

Outline

The CONSOLE statement is used to specify the scroll range.

Syntax

(1)-1



(1)-2

CONSOLE start line , end line

Note: If any value below the start line is specified as the end line, the start line is assigned to the end line.

Description

- The CONSOLE statement sets the scroll range of the text screen.
- The range of start line and end line is specified as follows:

R3764/66 (fluorescent character display tube);	0 to 7
R3764/66 (external monitor);	0 to 29
R3765/67;	0 to 29

Example

```
10 CONSOLE 0,5
20 PRINT "This is Network Analyzer"
30 PRINT "....Sweep Check Program...."
40 STOP
```

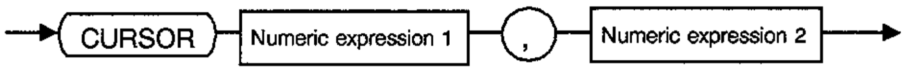
6. CURSOR

Outline

The CURSOR statement is used to move the cursor to the specified coordinate position.

Syntax

(1)-1



(1)-2

CURSOR numeric expression 1 , numeric expression 2

Note: Numeric expression 1: X-axis specification (column direction)
Numeric expression 2: Y-axis specification (line direction)
A space may be used instead of a comma.

Description

- The CURSOR statement moves the cursor to the specified position on the screen.
- The numeric expression 1 is used to specify X-axis coordinate, and the numeric expression 2 is used to specify Y-axis coordinate.
- The range of X-axis coordinate and Y-axis coordinate is specified as follows:

R3764/66 (fluorescent character display tube);	$0 \leqq X \leqq 31$	$0 \leqq Y \leqq 7$
R3764/66 (external monitor);	$0 \leqq X \leqq 79$	$0 \leqq Y \leqq 29$
R3765/67;	$0 \leqq X \leqq 66$	$0 \leqq Y \leqq 29$

Example

```
10 CLS
20 X=4:Y=4:X1=1:Y1=1
30 CURSOR X, Y:PRINT " ";
40 X=X+X1:Y=Y+Y1
50 CURSOR X, Y:PRINT "*"
60 IF X<=0 OR 67<=X THEN X1 *=-1
70 IF Y<=0 OR 29<=Y THEN Y1 *=-1
80 GOTO 30
90 STOP
```

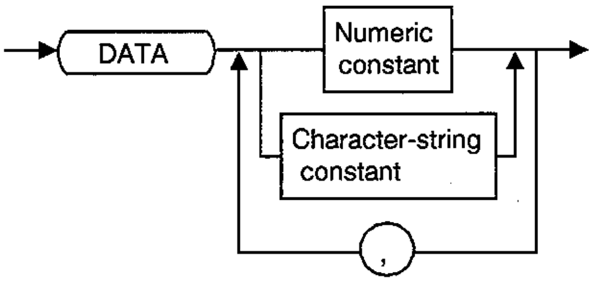
7. DATA

Outline

The DATA statement is used to define the numeric and the character string to be read out by the READ statement.

Syntax

(1)-1



(1)-2

DATA <numeric constant|character-string constant> {, <numeric constant|character-string constant> }

Description

- Since the DATA statement does not become the object to be executed, so it can be placed in any statement number. Generally, the DATA statement is necessary based on the order read out by the READ statement.
- The READ statement searches the DATA statement in the program and retrieves the data to be read.
- To change this order, use the RESTORE statement.
- In DATA statement, multiple constants can be defined, by using commas or spaces for separating the constants. The character string is enclosed with double quotation as character-string constant.
- After the DATA statement, multi-statement separated by a colon cannot be used.

Note

In DATA statement, the parameters (expressions) which include variables cannot be used.

8. DATE$

Outline

The DATE$ statement is used to read out date and to change the date.

Syntax

(1)-1

(1)-2
DATE$ = "

(2)-1

(2)-2
DATE$ = "year/month/day"

Description

- The DATE$ statement reads out the date of the system built-in timer (RTC).
- The read out date can be changed.
  Input as follows:

  DATE$ = " 93/1/1"
  or
  DATE$ = " 93/1/01"

Example

```
10 DIM D$[10]
20 D$=DATE$
30 PRINT "Date is ":D$
40 PRINT "Date Reset"
50 DATE$="93/1/1"
60 STOP
```

9. DELIMITER

Outline

The DELIMITER statement is used to select four types of delimiters and to set them.

Syntax

(1)-1

→(DELIMITER)→ Numeric expression →

(1)-2
DELIMITER numeric expression

Description

- The DELIMITER statement sets the delimiter corresponding to the number resulted by numeric expression.
  The following table shows the selection numbers and the types of delimiters.

Selection No.	Type of delimiter
0	Outputs 2-byte code of CR and LF. Also outputs single signal EOI immediately with LF output.
1	Outputs 1-byte code of LF.
2	Outputs single signal EOI immediately with end of data byte.
3	Outputs 2-byte code of CR and LF.

- If the result of numeric expression exceeds the range of 0 to 3, an error may occur.
  Numeric digits that follow after a decimal point are ignored and recognized as an integer.
- "DELIMITER = 0" is automatically set as a default value when the power is turned on.

Example

```
10 DELIMITER 0
20 DELIMITER 1
30 DELIMITER A*10
```

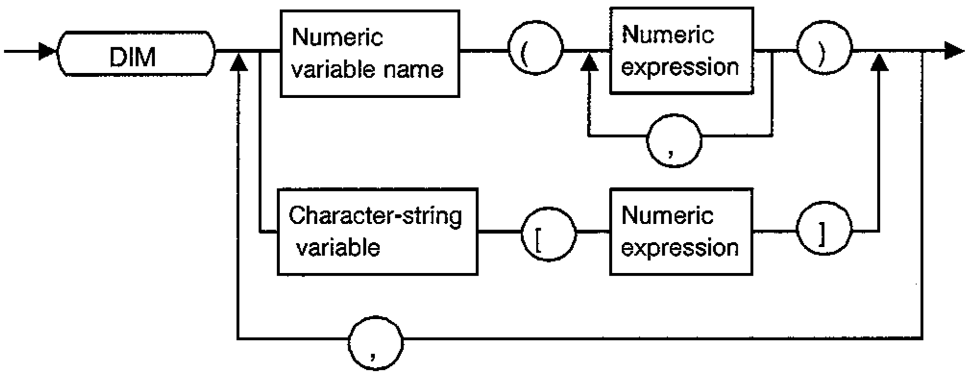
10. DIM

Outline

The DIM statement is used to define the array variable or character-string variable.

Syntax

(1)-1



(1)-2
DIM <A|B> {, <A|B>}

Note: A: numeric variable name [(numeric expression {, numeric expression})]
B: character-string variable [numeric expression]

Description

- When the array variable and character-string variable are used, the array variable name and the character length of array variable must be defined by DIM statement. If the array variable is used with no definition, the array variable will become 10 prime numbers in one dimension, and the character string will be the length of 18 characters.
- When the array declaration is performed by the DIM statement, the specified size array variable is reserved into memory. If more array declaration is performed, the remaining capacity (space) of BASIC program will be decreased and then the program may stop and will be resulted in an error (memory space full).
- The numeric expression that indicates an array variable size recognizes the real number as an integer by omitting the digit followed after a decimal point, even if the calculation has resulted in a real expression. A zero cannot be used for an array variable.
- Numeric expression is used to declare the length of character string for character-srting variable.

Example

```
10 DIM N(5)
20 FOR I = 1 TO 5
30      N(I) = I*I/2
40 NEXT I
50 FOR I = 1 TO 5
60      PRINT N(I)
70 NEXT I
```

<Result>
0.5
2.0
4.5
8.0
12.5

11. DISABLE INTR

Outline

The DISABLE INTR statement is used to prohibit the interruption reception.

Syntax

(1)-1

→ (DISABLE INTR) →

(1)-2
DISABLE INTR

Description

- The DISABLE INTR statement prohibits the interruption by ENABLE INTR statement.
- When the interruption is permitted again after the DISABLE INTR statement performs, the ENABLE INTR statement must be performed. At this case, the branch condition set by ON XXX statement is kept as the previous condition. However, if the condition of interruption branch is changed, it can be set using ON XX or OFF XXX statement before the ENABLE INTR performs.
- After immediately executing (running) the program, the interruption is prohibited until the ENABLE INTR is executed.

Example

```
10 ON KEY 1 GOTO 60
20 ENABLE INTR
30 ! LOOP
40 GOTO 30
50 !
60 DISABLE INTR
70 PRINT "KEY 1 INTERRUPT"
80 STOP
```

12. DSTAT

Outline

The DSTAT statement is used to obtain the contents of directory for BASIC variable.

Syntax

(1)
```
DSTAT <index> <variable>
```
(2)
```
DATAT <index> <fileattribute> <size> <sectors>
                <year> <month> <day> <hour> <minutes>
                <start-sector>
```
(3)
```
DSTAT ; SELECT <string> COUNT <variable>
```

Description

- Syntax of (1)
  The DSTAT statement checks the number of files stored in the directory of file system. A zero is specified for 1st parameter <index>, and numeric variable for 2nd parameter. The result is assigned to the 2nd parameter.

- Syntax of (2)
  The DSTAT statement obtains the directory information of file system for BASIC variable. The index of the directory is specified by 1st parameter <index>. The settable values are between 1 to the number of stored files (the number of stored file is the value obtained by syntax of (1)).
  For 2nd parameter, character-string variable is specified. The file name of result is stored for the 2nd parameter.
  For 3rd parameter and after, all of the parameters are specified with numeric variables. In these parameters, the following contentsare assigned:

fileattribute	File attribute (when file has multiple attributes, the parameter is output by adding each number.) 1. READ ONLY 8. VOLUME LABEL 2. HIDDEN FILE 16. DIRECTORY 4. SYSTEM FILE 32. ARCHIVE FILE
size	File size (number of byte)
sectors	Number of sector
year, month, day	Date of file created
hour, minutes	Time of file created
start-sector	Start sector of file

- Syntax of (3)

  The DSTAT statement assigns the number of file specified by parameter <character string> to the parameter <variable>.

  This syntax is used for searching files whether the specified file is existed in the directory or not.

  ? : Same as one character

  * : Same as one character or more

  [ ] :Same as any one character of character string enclosed with [ ].
  If parameter is specified with [character 1 - character 2], then it is the same as the character between character 1 and character 2.

13. ENABLE INTR

Outline

The ENABLE ENTER statement is used to permit the interruption reception.

Syntax

(1)-1

→ (ENABLE INTR) →

(1)-2
ENABLE INTR

Description

- The ENABLE ENTR statement permits the interruption reception, and enables the interruption branch defined by ON XXX statement.
- If the interruption is permitted again after performing the DISABLE INTR, then the ENABLE INTER statement must be executed.
- After immediately executing the program, the interruption cannot be performed until the ENABLE INTR statement is performed.

Example

```
10 ON KEY 1 GOTO 60
20 ENABLE INTR
30 ! LOOP
40 GOTO 30
50 !
60 PRINT "KEY 1"
70 GOTO 20
```

CAUTION

If the interruption defined by ON XXX statement occurs, then the interruption cannot be used after immediately the program branches, even if the ENABLE INTER statement is executed (same as DISABLE INTR statement). That is to prevent the Nest for the interruption processing, if the next interruption occurred during interruption.
To enable the interruption branch continuously, the ENABLE INTR statement is required again to permit the interruption.

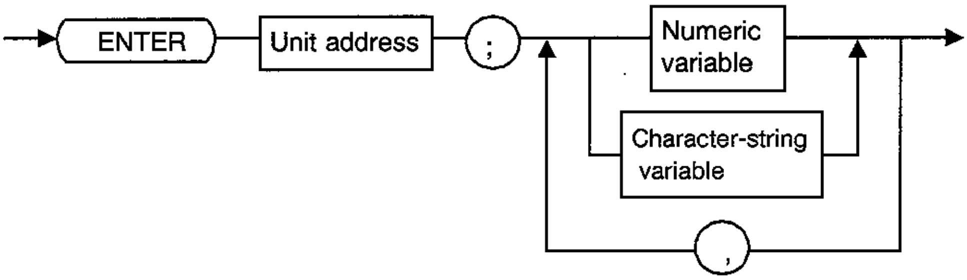
14. ENTER

Outline

(1) The ENTER statement obtains data from a GPIB and a parallel I/O.
(2) The ENTER statement read data from file and assigns the data to an input item.

Syntax

(1)-1



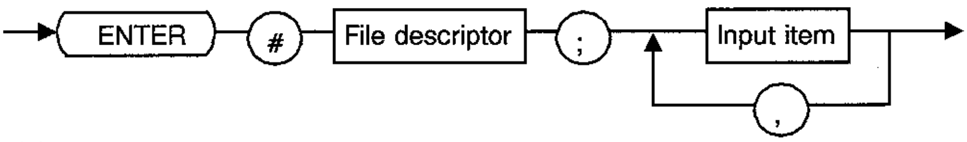
(1)-2

ENTER unit address; <numeric variable|character-string variable>
{, <numeric variable|character-string variable>}

Note: Unit address:

	0 to 30;	Unit address connected to an external GPIB.
	31;	Data input from measurement section of the analyzer.
	34;	Read out of parallel port Flip/Flop condition.
	35;	Data read out of parallel port C.
	36;	Data read out of parallel port D.
	37;	Data read out of parallel port CD.

(2)-1



(2)-2

ENTER # file descriptor ; input item {, input item}

Description

Syntax of (1)

- The ENTR statement inputs data from the unit specified by unit address through a GPIB and stores the data into BASIC variable as numeric variable or character string. Pay attention that the controller will stop the operation without completing handshake if talker function is not provided for the unit specified by the unit address.
  When character-string variable is used, it must be defined by DIM statement.
- In character staring input, pay attention that the input data will overflow and the overflowed data will be ignored, if the length of character string variable used for destination is not enough.

- Example

```
10 ENTER 1;A
20 DIM A$ (100), B$(20)
30 ENTER 2;A$
40 ENTER 3;B$
```

- Note

  When SYSTEM CONTROLLER mode is selected, the unit specified by the address is set as talker and the data are obtained.

Syntax of (2)

- The ENTER statement reads data as data-type format corresponding input item from the file assigned to the file descriptor, and assigns the data to the input item.

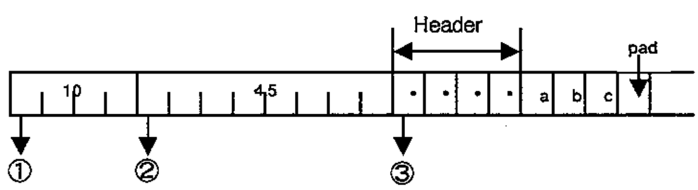
Note: For the information how to handle files, refer to "1. Preface • File Management".

- Example 1: BINARY file

The ENTER statement assigns an internal data as it is. It also enables to read the data of the number of byte indicated by the header contents after reading each header such as integer of 4 byte, real number of 8 byte, and character string of 4 byte.
Since the number of byte to be read is decided by the type of input item, the same type as OUTPUT is required for preventing the data difference

```
10 INTEGER I
20 DIM R
30 OPEN "FILE" FOR INPUT AS #FD
40 ENTER #FD;I,R,S$
```

Number of byte to be read differs according to the variable type to be assigned.

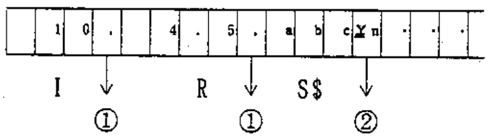


①: When the variable is an integer, 4-byte data is read and assigned to the variable.
②: When the variable is a real number, 8-byte data is read and assigned to the variable.
③: When the variable is a character string, 4-byte header and header length are read and assigned to the variable.

- Example 2: TEXT file

Regardless of the number of input items, the TEXT file is read out until the line field. The TEXT file is recognized as one data until a comma and converted into the input-item type, then it is assigned. If the number of input items is more, it cannot be assigned to the variables. Therefore, these values stored in advance are remaining. In reverse, if the number of variables is less than the number of actual data, the data are omitted.

```
10 INTEGER I
20 DIM R
30 OPEN "FILE" FOR INPUT AS #FD;TEXT
40 ENTER #FD;I,R,S$
```

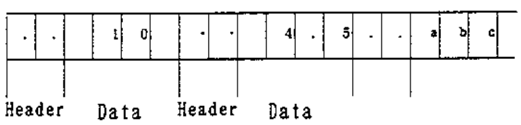


①: Each item is delimited with a string of commas.
②: LF followed after the final item is used.

- Example 3: ASCII file

The 2-byte header and its data according to the header length are read out. The ASCII file is converted into the variable type and assigned.

```
10 INTEGER I
20 DIM R
30 OPEN "FILE" FOR INPUT #FD;ASCII
40 ENTER #FD;I,R,S$
```



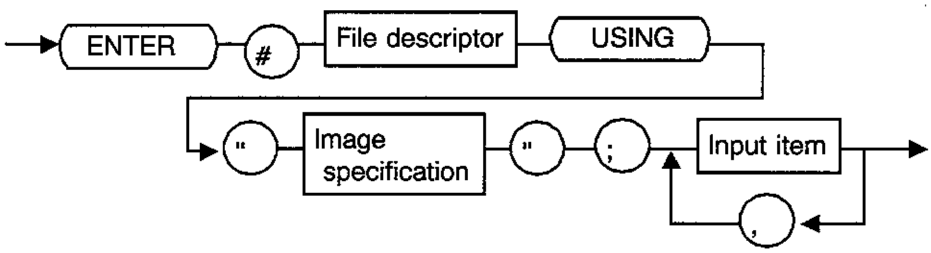
15. ENTER USING

Outline

The ENTER USING statement is used to enter data to the input item from the file by using the image specification format.

Syntax

(1)-1



(1)-2

ENTER # file descriptor USING "image specification" ; input item {, input item}

Note: ENT can be used instead of the ENTER, and USE for the USING.

Description

The ENTER USING statement enters the data to the input item from the file assigned to the file descriptor by using the image specification format.
It is effective only when opened as a TEXT file.

image specification

D: Recognizes the numeric of D as a numeric digit and reads out it, then assigns it to the variable of the input item.
Z: Same as D.
K: Reads out one line and converts it into the numeric data, then assigns it to the variable of the input item.
S: Same as D.
M: Same as D.
.: Same as D.
E: Same as K.
H: Same as K. However, use a comma for a decimal point.
*: Same as D.
A: Reads the number of A and assigns it to the character-string variable.
k: Reads one line and assigns it to the character-string variable.
X: Skips one-character data.
Literal: Skips the the character-string numeric data enclosed with \".
B: Reads one character and assigns it to the input item using an ASCII code.

Description

image specification

@: Skips one-byte data.

+: Same as @

-: Same as @

#: Ignored in ENTER statement.

n: Specifies the number of repetition of each image by using numerics. For example, 3D.2D is the same as for DDD.DD, and 4A for AAAA.

Note: For the information how to handle files, refer to "1. Preface ● File Management".

Example

```
10 INTEGER INT
20 DIM     REL
30 ENTER #FD USING "ZZZ,DD.D,3A";INT,REL,S$
```

0	1	0		4	.	5	a	b	c	\n	・・・		
	INT				REL			S$					

INT: Reads out 3-byte data and converts it into an integer-type data, then assigns it to the variable INT.

REL: The DD.D of image specification corresponds to the REL of the input item. Reads out 4-byte data and converts it into a real-type data, then assigns it to the variable REL. After the execution, the REL becomes 4.5.

S$: Reads out 3-byte data and assigns it to the variable S$. After the execution, the A$ becomes "abc".

```
10 DIM A,B
20 ENTER #FD USING "SDDD,X,MZZZ";A,B
```

		+	5		−	0	1	3	・・・
	A			X			B		

A,B: Reads out 4-byte data and converts it into a real-type data, then assigns it to the variables A and B.
After the execution, the A = 5.0, and the B = -13.0.
The image specification X can read 1-byte data, however, cannot assign it to the variable. Converts the data, which is input using an SDDD format, into a real-type data, and assigns it to the variable A.
The image specification X is not required for variable, it skips one character.
The MZZZZ corresponds to the variable B and enters 4-byte data to convert it into a real-type data, then assigns it to the variable B.

```
10 DIM A
20 ENTER #FD USING "K";A
```

S	T	R	I	N	G	1	2	3	.	5	#	#	\n	· ·

Execution result A = 123.5

The STRING123.5## is read out and converted into the real-type data of input variable A. When the input item is a real-type data, the preceding character strings other than numerics, signs (+ , -), and exponents (E, e) are ignored and only the numerics are obtained. Only the numerics can be detected. If the character other than numerics is detected, the conversion is terminated.

For the image specifications such as K, E, k, and H, since LF represents terminator, the data from the current file pointer to the LF as one data are assigned to the variables.

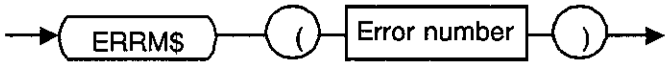
16. ERRM$

Outline

The ERRM$ statement is the system function which is used to return an error message of the number specified.

Syntax

(1)-1



(1)-2

ERRM$ (error number)

Description

- The ERRM$ statement returns the error message specified by parameters. Particularly, if 0 as a parameter is specified, the ERRM$ returns the error message immediately displayed.
- The error numbers are constructed from as follows:
  Error classes * 256 + error message number
  Error classes: 1; Data input
  2; Data calculation processing
  3; Built-in function
  4; BASIC syntax
  5; Others
- If the numbers which include the error classes are specified, only the error message numbers will be displayed. Therefore, the ERRN can be specified for the error numbers.

17. ERRN

Outline

The ERRN statement is the system variable which holds an error number.

Syntax

(1)-1

(1)-2
ERRN

Description

- The ERRN statement is the system variable, which holds the error number occurred when the BASIC program is being executed.
- The ERRN is initialized to 0 when the BASIC program starts, and if an error occurs, its number will be assigned to the ERRN. To initialize this assigned value to 0, forcibly assign 0 to the ERRN or re-start the BASIC program.
- The error numbers are constructed from as follows:
  Error classes * 256 + error message number
  Error classes: 1; Data input
  2; Data calculation processing
  3; Built-in function
  4; BASIC syntax
  5; Others

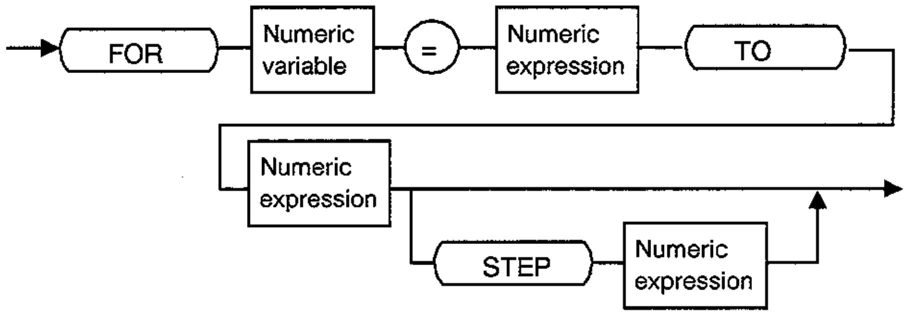
18. FOR - TO - STEP, NEXT, BREAK, CONTINUE

Outline

This statement consists of the program loop (loop processing) by combining with FOR statement and NEXT statement.

Syntax

(1)-1



(1)-2

FOR numeric variable = numeric expression TO numeric expression [STEP numeric expression]

(2)-1

(2)-2

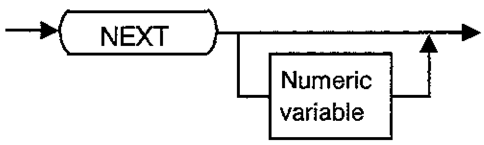
BREAK

(3)-1

(3)-2

CONTINUE

(4)-1



(4)-2

NEXT [numeric variable]

Description

- This statement uses the numeric variable specified as a loop counter (repetition) and enables to increase the value from the initial value to the final value by the increased step. If the counter value exceeds the final value, then the loop will terminate. The counter increment/decrement is performed by the NEXT statement. Therefore, the program created between FOR statement and NEXT statement is looped repeatedly.
- The values of the initial, final, step are as follows:
  FOR A = (initial value) TO (final value) STEP (increment)
- If STEP (increment) value is omitted, the value is automatically incremented by 1.
- Nest is available between FOR statement and NEXT statement.
- The numeric variable name of the loop counter used for a pair of FOR statement and NEXT statement, be sure to use the same name. If the numeric variable name is different, an error may occur.
- If the value of numeric variable used for the loop counter is changed when the loop processing is executed between FOR statement and NEXT statement, the normal loop processing could not be performed.
- If the numeric variable followed after NEXT statement is omitted, the NEXT statement will automatically correspond to immediately FOR statement.
- BREAK statement can be used to exit in FOR-NEXT loop.
- CONTINUE statement branches to the next step loop in FOR-NEXT loop.
- For example, if a loop like FOR I = 0 TO 10 STEP -1 is specified, the line in the loop ends without performed.

Example

```
10 FOR R=11 TO 0 STEP -5
20   FOR I=0 TO PI STEP PI/180
30     X=SIN(I)*R+23
40     Y=COS(I)*R+15
50     CURSOR X,Y:PRINT "*"
60   NEXT I
70 NEXT R
80 STOP
```

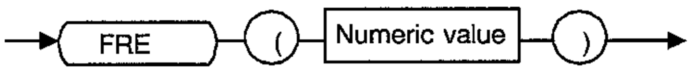
19. FRE

Outline

The FRE statement is the system function which returns the memory space of BASIC.

Syntax

(1)-1



(1)-2
FRE (numeric value)

Description

1. When the numeric value is 0.
- Returns the memory space roughly with the bite number to be used by the BASIC.
- This statement checks the memory space roughly and performs no re-structure strictly. Therefore, saving and re-loading the data may result in more memory capacity.

2. When the numeric value is 1.
- Returns the memory space roughly with the bite number to be used by the built-in function.

3. Others
- Returns 0.

Example

```
PRINT FRE(0)
```

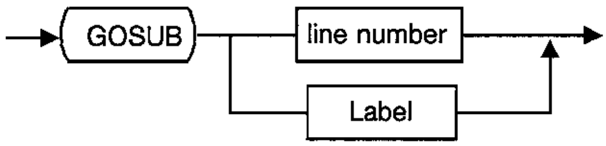
20. GOSUB, RETURN

Outline

This statement is used to branch/return to the specified subroutine.

Syntax

(1)-1



(1)-2

GOSUB <line number|label>

(2)-1

(2)-2

RETURN

Description

- Moves the processing control to the defined line number subroutine and returns to the next statement to the GOSUB statement by the RETURN statement.
- Be sure to input the RETURN statement at the end of subroutine and return the processing control to the main program.
- If the RETURN statement is executed without the branch to subroutine, an error may occur.
- Since Nest is available between the GOSUB statement and RETURN statement, the processing can branch to the other subroutine. If more Nest is performed, the remaining capacity (space) of BASIC program will be decreased and then an error may occur.
- If the line number or the label defined in GOTO/GOSUB does not exist, the program is not executed.
  When it runs, "Undefined LABEL" is displayed and the program stops by error without executing any line.

Example

```
10 FOR I=1 TO 9
20   GOSUB 60
30   GOSUB *PRT
40 NEXT I
50 STOP
60 ! SUB ROUTINE
70 X = I * I
80 RETURN
90 *PRT ! SUB ROUTINE
100 PRINT I; " * " ;I; " = " ;X
110 RETURN
```

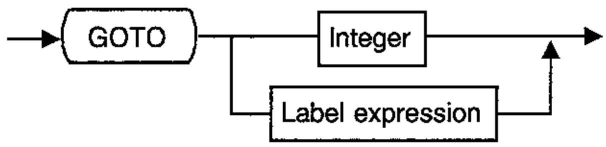
21. GOTO

Outline

The GOTO statement is used to branch to the specified line.

Syntax

(1)-1



(1)-2

GOTO <integer|label expression>

Description

- The GOTO statement branches to the specified line number unconditionally.
- If the line number or the label defined in GOTO/GOSUB does not exist, the program is not executed.
  When it runs, "Undefined LABEL" is displayed and the program stops by error without executing any line.

Example

```
10 FOR I=1 TO 9
20   GOTO 60
30   GOTO *PRT
40 NEXT I
50 STOP
60 !
70 X = I * I
80 GOTO 30
90 *PRT
100 PRINT I; " * " ;I; " = " ;X
110 GOTO 40
```

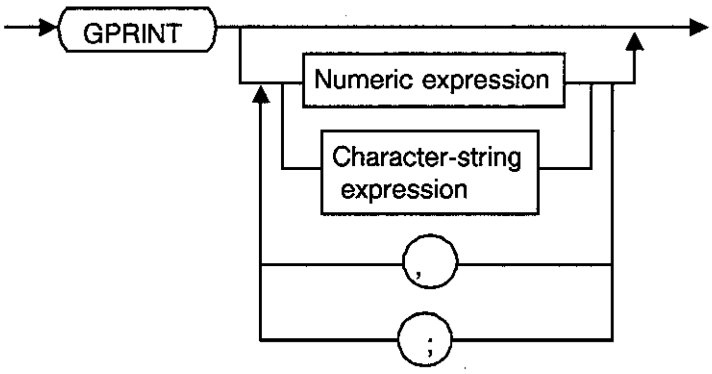
22. GPRINT, LPRINT

Outline

This statement is used to output numerics or character strings.
GPRINT: GPIB output
LPRINT: Serial output

Syntax

(1)-1



(1)-2

GPRINT [<numeric expression|character-string expression> {, | <numeric expression|character-string expression>}

(2)

The LPRINT is the same as the GPRINT.

Description

- This statement displays the numerics or character strings specified by the GPRINT or LPRINT.
- When the multiple numerics or character strings are delimited with a comma and specified, they are continuously output without LF.
- If a semicolon is used at the end of the GPRINT/LPRINT statement, LF could not be performed after the termination of print out. Therefore, if the next GPRINT/LPRINT statement is executed, the line followed after the previous output line will be output continuously.
- When GPRINT is used to output data to GPIB printer, be sure to set SYSTEM CONTROLLER by the analyzer panel operation and set up the printer address.

Example

```
100 PRINTER 1
110 FOR I=0 TO 20
120   GPRINT I
130   LPRINT I
140 NEXT I
150 STOP
```

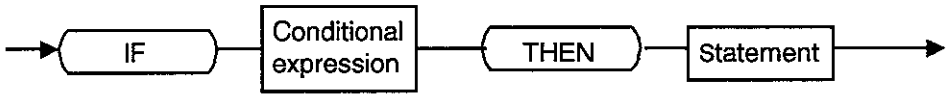
23. IF-THEN, ELSE, END IF

Outline

This statement is used to perform the branch based on the condition branch and the specified statement.

Syntax

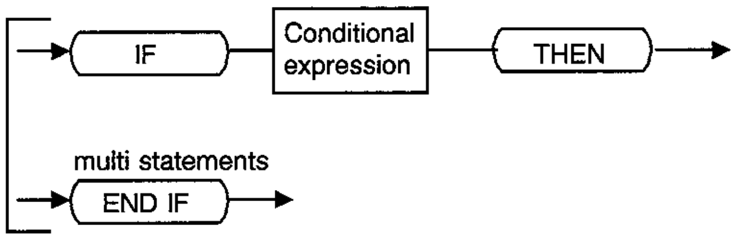
(1)-1



(1)-2

IF conditional expression THEN statement

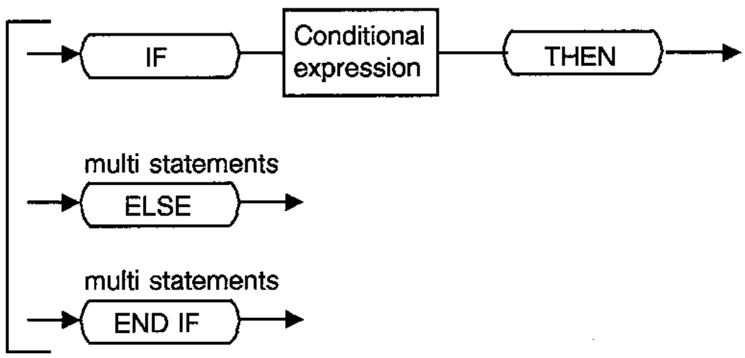
(2)-1



(2)-2

```
IF conditional expression THEN
  multi statements
END IF
```

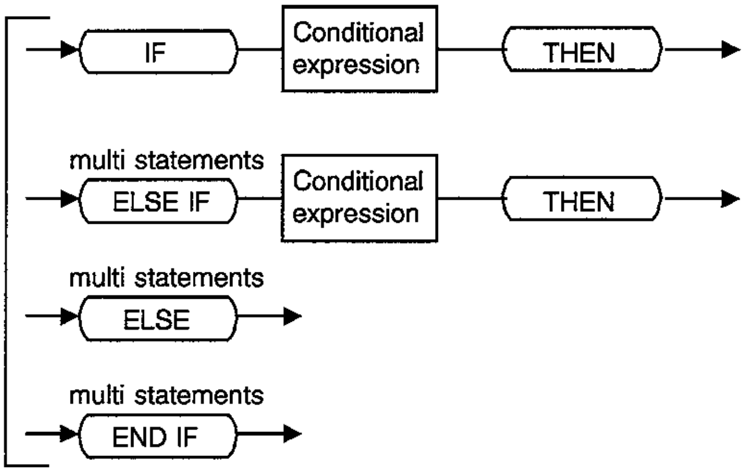
(3)-1



(3)-2

```
IF conditional expression THEN
  multi statements
ELSE
  multi statements
END IF
```

(4)-1



(4)-2

```
IF conditional expression THEN
  multi statements
ELSE IF conditional expression THEN
  multi statements
ELSE
  multi statements
END IF
```

Description

- Generally, the condition expression represents a logical expression, however, numeric expression can be used in this statement other than the logical expression used relational operators. In this case, when the calculation result becomes 0 only, the value is determined as FALSE, and the values other 0 is estimated as TRUE.
- Depending on the condition of logical expression, branching and processing the program can be performed.
- When the logical expression is defined, the THEN statement can be executed. The other statements can be followed after the THEN statement and the next statement can be executed.
- If the logical expression cannot be concluded, the next line is performed.
- The following six types of relational operators are provided:

A=B	Returns true if A equal to B; false otherwise.
A>B	Returns true if A is greater than B; false otherwise.
A<B	Returns true if A is less than B; false otherwise.
A> =B	Returns true if A is greater than or equal to B; false otherwise.
A< =B	Returns true if A is less than or equal to B; false otherwise.
A< >B	Returns true if A does not equal to B; false otherwise.

In the logical expression above, both values A and B consist of numeric expression. The comparison between numeric expression and character-string expression can be performed.

Example

```
10 FLG = 0
20 FOR I=0 TO 10
30  PRINT I;
40  IF (I % 2) =0 THEN FLG = 1
50  IF FLG = 1 THEN
60                  PRINT "  EVEN" ;
70                  FLG = 0
80                  END IF
90  PRINT
100 NEXT I
110 STOP
```

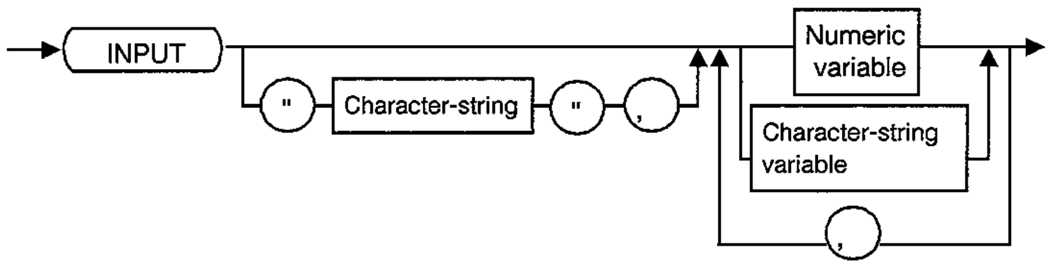
24. INPUT

Outline

The INPUT statement is used to assign the data entered by keys to numeric variables.

Syntax

(1)-1



(1)-2

INPUT ["character-string",] <numeric variable | character-string variable> {, <numeric variable | character-string variable>}

Description

- When the INPUT statement is executed, then the program is temporarily suspended and waits for next key to be input. The waiting state for the key input is continued until the ENTER key is pressed. If the ENTER key is pressed after data input, the data will be assigned to variables.
- Both numeric variable and character-string variable can be handled in the INPUT statement. In case of numeric variable input, if the characters other than numeric (such as alphabets, symbols, and others) are entered, then they will be ignored. If no numeric is existed, then 0 will be assigned to the variable. If only the ENTER key is pressed, no assignment can be performed. In other words, the value immediately before the INPUT statement has been remaining.
- To enter a character constant, it is not required to be enclosed with double quotation marks.

Example

```
10 OUTPUT 31; "OLDC OFF"
20 OUTPUT 31; "INIT:CONT OFF"
30 INPUT "CENTER FREQUENCY(MHz) ?" ,CF
40 INPUT "SPAN FREQUENCY(KHz)?" ,SF
50 OUTPUT 31; "FREQ:CENT " ,CF, "MHz"
60 OUTPUT 31; "FREQ:SPAN " ,SF, "KHz"
70 OUTPUT 31; "INIT"
80 PRINT "MAX = " ,MAX(0,1200,0)
90 STOP
```

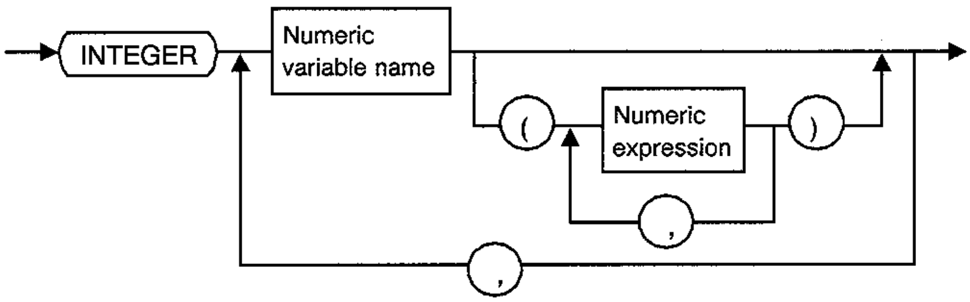
25. INTEGER

Outline

The INTEGER statement is used to declare that the variable or array variable is an integer type.

Syntax

(1)-1



(1)-2

INTEGER A[B] {, A[B] }

A: Numeric variable name
B: (Numeric expression {, Numeric expression})

Description

- When a numeric variable or an array variable is specified in the INTEGER statement, the variable is determined as an integer type after the specification.
- The numeric handled in the integer-type variable, it is the same as the range of an integer constant.
  -2147483648 to +2147483647
- In the variables which handle only the integers, the declaration in the INTEGER statement is recommended to shorten the processing time.
- When the array declaration is used in the INTEGER statement, the specified-size array variable is reserved on the memory. If larger array declaration is performed, an error may occur due to the rack of memory space (memory space full) and then the program execution will be forcibly terminated.
  (memory space full)
- When multiple subscripts are specified, the array variables are also specified according to the number of dimension. (Number of dimension is specified as long as the memory space is permitted.)

Example

```
10  INTEGER ARRAY(2,3)
20  PRINT "J/I " ;
30  PRINT USING "X,3D,3D,3D" ;1,2,3
40  PRINT " " ;
50  FOR I = 1 TO 2
60    FOR J = 1 TO 3
70        ARRAY(I,J) = I*10 + J
80    NEXT J
90  NEXT I
100 FOR I = I TO 2
110 PRINT
120 PRINT USING  " 2D,2X,# " ;I
130   FOR J = 1 TO 3
140       PRINT USING  "3D,#" ;ARRAY(I,J)
150   NEXT J
160 NEXT I
```

```
<Result>
 J/I  1  2  3

  1  11 12 13
  2  21 22 23
```

CAUTION

- The variable which is once specified as an integer type by the INTEGER statement, if the instruction is deleted by the DEL or comment statement, the specified variable (integer type) is not changed.
- To change the specified integer-type variable into a real-type variable again, add the DIM instruction or execute the SAVE/LOAD command once and then perform the RUN command.

26. INTERFACE CLEAR

Outline

The INTERFACE CLEAR statement is used to initialize the all GPIB interfaces connected with the analyzer.

Syntax

(1)-1

(1)-2

INTERFACE CLEAR

Description

- When the INTERFACE CLEAR statement is executed, the GPIB single signal IFC is output approximately 100μs.
  If the all GPIB interface devices connected with the analyzer receive the IFC signal, then the setting state of talker or listener will be canceled.

Example

```
10 INTERFACE CLEAR
```

Note

The INTERFACE CLEAR statement is not available in the ADDRESSABLE mode.

27. KEY$

Outline

The KEY$ statement is used to return the code of panel key.

Syntax

(1)-1

(1)-2
KEY$

Description

- The KEY$ statement returns the code pressed at the last operation. When this code is referred once, the contents of this variable is cleared.

Example

```
10  A$=KEY$
20  IF A$="1" THEN
30    GOSUB *TEST1
40  ELSE IF A$="2" THEN
50    GOSUB *TEST2
60  END IF
70  GOTO 10
80  STOP
100 *TEST1
110  PRINT "Check1 Start !!"
120  ......
130  RETURN
200 *TEST2
210  PRINT "Check2 Start !!"
220  ......
230  RETURN
```

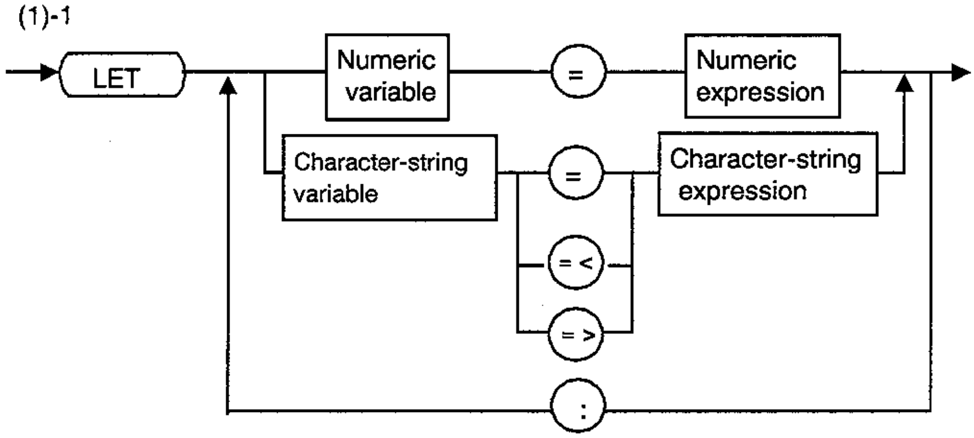
28. LET

Outline

(The LET statement is not used in the program, the assignment statement can be used directly.)
The LET statement is used to assign to the variable.

Syntax

(1)-1



(1)-2
LET <A | B> {: <A | B> }

A: numeric variable = numeric expression
B: character-string variable = | = < | = > character-string expression

Description

- The signs used in this statement indicate an assignment and differ from the sign used in arithmetic operation.
- If th left part of sign is a numeric, the numeric part of character string is converted and then assigned.
  Especially, when character string is assigned:
  when = : Only the length of right part is assigned.
  when = >: If the character string of the right part is shorter than the left one, spaces is used to assign the different values from the top of the left part.
  when = <: Spaces are used to fill up to the blank.
  Therefore, the signs = > and = < are assignment operators which are available only for character strings.

Example

```
10 DIM STR$                          <After the execution>
20 PRINT   "123456789012345678"       123456789012345678
30 STR$ = "ABC" :PRINT STR$           ABC
40 STR$ =< "OPQ" :PRINT STR$          OPQ
50 STR$ => "XYZ" :PRINT STR$                         XYZ
```

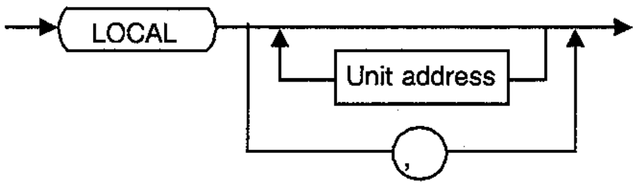
29. LOCAL

Outline

The LOCAL statement is used to cancel the specified device from the remote state or to set the remote-enable (REN) line to FALSE.

Syntax

(1)-1



(1)-2

LOCAL [unit address {, unit address}]

Description

- If only the LOCAL statement is executed without specifying the device address, then the GPIB remote-enable line will become FALSE (High level) and all the devices on the GPIB will be a local state.
  If the REN is FALSE, pay attention that the setting of GPIB device could not be performed (cannot be controlled by GPIB).
- To set the REN to TRUE (Low level) again, execute the REMOTE.
- If the device address is specified followed after the LOCAL, only the device specified by the device address could be addressed, and the remote state will be canceled.

Example

```
10 LOCAL
20 LOCAL 1
30 LOCAL 1,2,3
```

Note

The LOCAL state is not be available in the ADDRESS mode.

30. LOCAL LOCKOUT

Outline

The LOCAL LOCKOUT statement is used to prohibit the function which controls the local/remote state from the panel key of the device connected to the GPIB.

Syntax

(1)-1

→(LOCAL LOCKOUT)→

(1)-2
LOCAL LOCKOUT

Description

- When each device is remote state (controlled by GPIB), the panel key of each device is locked except for the LOCAL key and the data setting cannot be performed from each panel.
  When the LOCAL key is pressed during the remote state, the data setting is available since each device become local state. Therefore, various errors occur during the remote control and the control cannot be performed correctly.
  In this case, if the LOCAL LOCOUT statement is executed, its function enables to lock the all devices on the GPIB and the setting from each device panel can be completely prohibited.
- When the LOCAL LOCKOUT statement is executed, the local lockout (LLO) of universal command is sent to the GPIB.
- To cancel the local lockout state, use the LOCAL command to set the REN line to FALSE (High level).

Example

```
10 LOCAL LOCKOUT
```

Note

The LOCAL LOCKOUT statement is not available in the ADDRESSABLE mode.

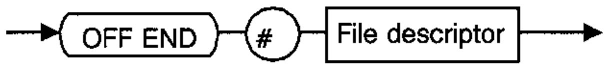
31. OFF END

Outline

The OFF END statement is used to cancel the processing of the end of file specified by the ON END statement.

Syntax

(1)-1



(1)-2

OFF END # file descriptor

Description

- After canceling the branch defined into file descriptor, if the end of file occurs, the following error message will be displayed and the program will be terminated.

  end of "DATAFILE" file

Note: For the information how to handle files, refer to "1. Preface ● File Management".

32. OFF ERROR

Outline

The OFF ERROR statement is used to cancel the branch function when an error occurs.

Syntax

(1)-1

→(OFF ERROR)→

(1)-2
OFF ERROR

Description

- The OFF ERROR statement prohibits the error branch defined by the ON ERROR statement.

Example

```
10  ON ERROR GOTO 100
      :
100 OFF ERROR
110 PRINT "Error Code" ,ERRN
120 STOP
```

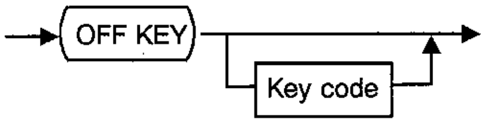
33. OFF KEY

Outline

The OFF KEY statement is used to cancel the branch function by interruption of KEY input.

Syntax

(1)-1



(1)-2
OFF KEY [key code]

Description

- The OFF KEY statement prohibits the branch by the interruption of the analyzer KEY input, which is permitted by the ON KEY statement.

Example

```
10  ON KEY 2 GOTO 100
20  ENABLE INTR
30  ! LOOP
40  GOTO 30
100 OFF KEY
110 PRINT "OFF KEY"
120 STOP
```

34. OFF SRQ, OFF ISRQ

Outline

This statement is used to cancel the function and definition by the interruption of SRQ or ISRQ.

Syntax

(1)-1

→ (OFF SRQ) →

(1)-2

OFF SRQ

(2)

The OFF ISRQ is the same as the OFF SRQ.

Description

- OFF SRQ
  This statement prohibits the branch by the interruption, which is permitted by the ON SRQ.
- OFF ISRQ
  This statement prohibits the branch by the interruption, which is permitted by the ON ISRQ.

Example

```
100 OUTPUT 31; "OLDC OFF"
110 OUTPUT 31; "START:OPER:ENAB 8;*SRE 128":SPOLL(31)
120 ON ISRQ GOTO *MAX
130 OUTPUT 31; "INIT:CONT OFF;:ABOR;:INIT"
140 ENABLE INTR
150 ! LOOP
160 GOTO 150
170 *MAX
180 DISABLE INTR
190 OFF ISRQ
200 PRINT MAX(0,1200,0)
210 STOP
```

Address	Contents
110	Enables the SRQ.
120	Sets the interruption branch of the internal SRQ.
130	Single sweep.
140	Interruption reception.
180	Interruption prohibition.
190	Cancels the interruption branch of the internal SRQ.
200	Displays the maximum level.

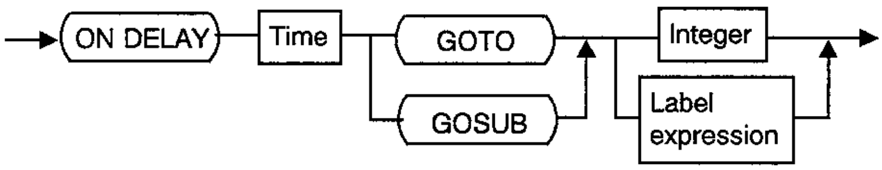
35. ON DELAY

Outline

The ON DELAY statement is used to branch after the specified time elapsed.

Syntax

(1)-1



(1)-2
ON DELAY time <GOTO | GOSUB> <integer | label expression>

Note: The unit of time is msec, and the setting range is between 0 to 65535.

Description

- The ON DELAY statement branches according to the statement after the specified time elapsed.
- Acceptance of the interruption should be permitted by the ENABLE INTR statement.

Example

```
10  INTEGER T
20  T=50
30  ENABLE INTR
40  ON DELAY T GOSUB *TEST
50  STOP
100 *TEST
110  PRINT T;"[msec] Delay"
120 RETURN
```

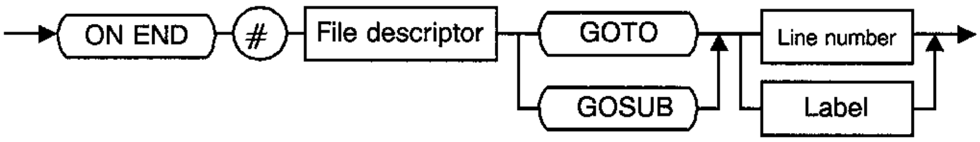
36. ON END

Outline

The ON END statement is used to define the processing (destination branch) at the end of file.

Syntax

(1)-1



(1)-2

ON END #file descriptor <GOTO | GOSUB> <line number | label>

Description

- The ON END statement reads out the data from the file by the ENTER command, if the data to be entered is not existed with reading out the end of file, the result will be the end of file.
  If the processing declaration is omitted in the ON END statement, after closing the file, an error message will be displayed and the program will terminate.

Note: For the information how to handle files, refer to "1. Preface • File Management".

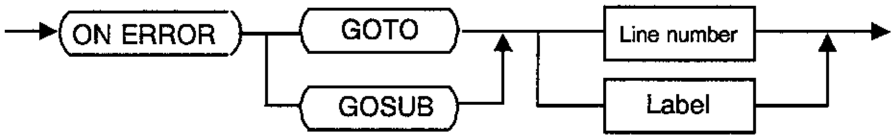
37. ON ERROR

Outline

The ON ERROR statement is used to permit the branch when an error occurs.

Syntax

(1)-1



(1)-2

ON ERROR <GOTO | GOSUB> <line number | label>

Description

- If an error occurs during the BASIC program, the statement number and error message of the program will be displayed and the program will terminate.
  Especially, if the built-in function error which demands the service request of the measuring device, only the error message will be displayed and the program will continue the operation. To detect the error to branch, use the ON ERROR statement is used.
- To categorize the generated error, the ERRN system variable which stores the error number is provided.
- After generating the error, if the error is not recovered by the error processing, then the endless loop will be performed. To prevent this trouble, the OFF ERROR statement must be used (written).

Example

```
ON ERROR GOTO 1000
```

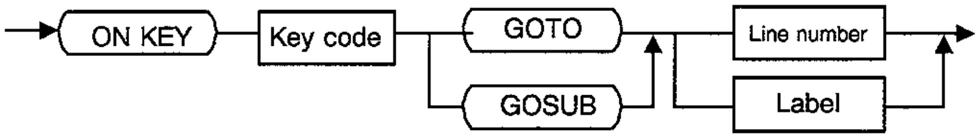
38. ON KEY

Outline

The ON KEY statement is used to permit the branch by the interruption of KEY input.

Syntax

(1)-1



(1)-2

ON KEY key code <GOTO | GOSUB> <line number | label>

Description

- The ON KEY statement branches by the interruption of KEY input during the program execution.
- The branch is executed after completing the processing of the statement being executed when the interruption is generated.
- The return position of the statement when the program branches to the subroutine is the next statement of the statement being executed when the interruption is generated.
- The key codes are constructed from the numerics of 1 to 6. They correspond to the function key on the front panel and the F1 to F6 on the key board. In addition, when the keyboard is connected to the analyzer, the key codes correspond to F1 to F6 on the key board.
- Acceptance of the interruption should be permitted by the ENABLE INTR statement.

Example

```
10   CLS
20   ON KEY 1 GOTO 1000
30   ON KEY 2 GOTO 1100
40   ON KEY 3 GOTO 1200
50   ON KEY 4 GOTO 1300
60   ON KEY 5 GOTO 1400
70   ON KEY 6 GOTO 1500
75   CNT = 10
80   *HERE:
85   I = 0: PRINT " "
90   IF I=CNT THEN FOTO *HERE
100  ++I: PRINT ">" ;
110  ENABLE INTR
120  GOTO 90
1000 PRINT "FIRST KEY"
1001 CNT = 1
1010 GOTO *HERE
1100 PRINT "SECOND KEY"
1101 CNT = 10
1110 GOTO *HERE
1200 PRINT "THIRD KEY"
1201 CNT = 20
1210 GOTO *HERE
1300 PRINT "FOURTH KEY"
1301 CNT = 30
1310 GOTO *HERE
1400 PRINT "FIFTH KEY"
1401 CNT = 40
1410 GOTO *HERE
1500 PRINT "SIXTH KEY"
1501 CNT = 50
1510 GOTO *HERE
```

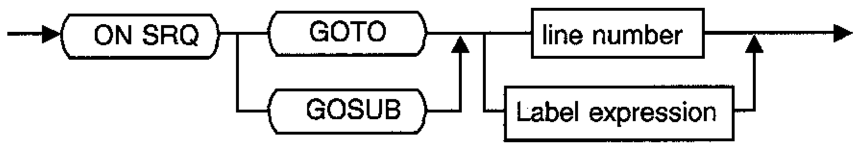
39. ON SRQ, ON ISRQ

Outline

The ON SRQ statement is used to permit the interruption branch by the GPIB external SRQ signal. (It is available in ON SRQ controller mode only.)
The ON ISRQ statement is used to permit the interruption branch when the internal interruption factor is generated.

Syntax

(1)-1



(1)-2
ON SRQ <GOTO | GOSUB> <line number | label expression>

(2)
The ON ISRQ is the same as the ON SRQ.

Description

- This statement branches by the interruption during the program execution.
- The branch is executed after completing the processing of the statement being executed when the interruption is generated.
- The return position of the statement when the program branches to the subroutine is the next statement of the statement being executed when the interruption is generated.
- The ON SRQ statement performs the interruption branch by the SRQ signal from the GPIB external during the controller mode in progress.
- Acceptance of the interruption should be permitted by the ENABLE INTR statement.

Example

Sample program which searches the MAX every single sweep.

```
100 OUTPUT 31;"OLDC OFF"
110 ON ISRQ GOTO *MAX
120 OUTPUT 31; "STAT:OPER;ENAB 8;*SRE 128" :SPOLL(31)
130 ENABLE INTR
135 OUTPUT 31; "INIT:CONT OFF;:ABOR;:INIT"
140 ! LOOP
150 GOTO 140
160 *MAX
170 DISABLE INTR:SPOLL(31)
180 PRINT MAX(0,1200,0)
190 GOTO 130
```

Address	Contents
110	Sets the interruption branch of the internal SRQ.
120	Enables the SRQ.
130	Interruption reception.
135	Single sweep.
170	Interruption prohibition.
180	Displays the maximum level.

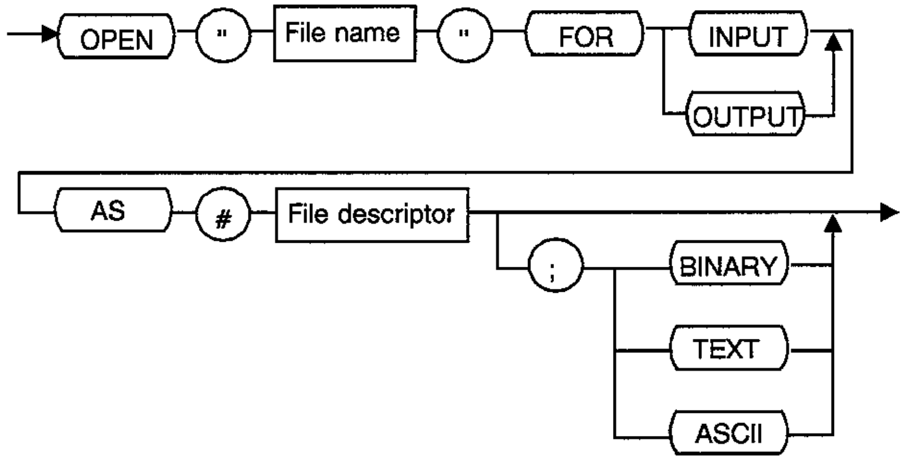
40. OPEN

Outline

The OPEN statement is used to assign the file descriptor to the file and to open the by with the specified processing mode.

Syntax

(1)-1



(1)-2

OPEN "file name" FOR processing mode AS #file descriptor [; file type]

Note: Processing mode: INPUT | OUTPUT
File type: BINARY | TEXT | ASCII

Description

- To recognize the file for the program, the OPEN statement assigns the file descriptor to the file and to open the by with the specified processing mode.

  Processing mode
  Two processing modes are provided.
  OUTPUT: Used for writing the data to files.
  INPUT: Used for reading out the data from files.

  # File descriptor
  Generally, writing/reading files uses the ENTER or OUTPUT mode.
  For these commands, the file descriptor is used to recognize the target files. To name the file descriptor, use alphanumerics followed after #.

File type

Three file types (BINARY, TEXT, and ASCII) are provided.

If the file type is not specified, BINARY type is automatically set.

BINARY: Stores the data without changes. An integer type is 4-byte data, a real type for 8-byte data, and a character-string type for header 4-byte. In case of the character-string type, ASCII data is followed after the header 4-byte. If the number of character data is an odd, then one space of 1-byte will be followed after the data.

TEXT: Converts data into ASCII codes and outputs the data, and "-" or space is followed before the numeric. The USING specification can be used for the TEXT file.

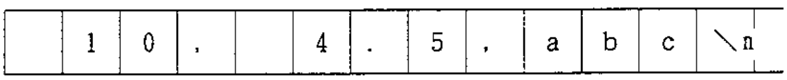
ASCII: Represents the input/output item using ASCII codes followed after 2-byte header. "-" or space is followed before the numeric. If the number of the character data is an even, then one space will be followed after the data.

- When the file descriptor already assigned the file to the other file is opened, the previous assigned file is closed and the specified file is newly opened.
- The same files cannot be opened using the multiple file-descriptor at the same time.

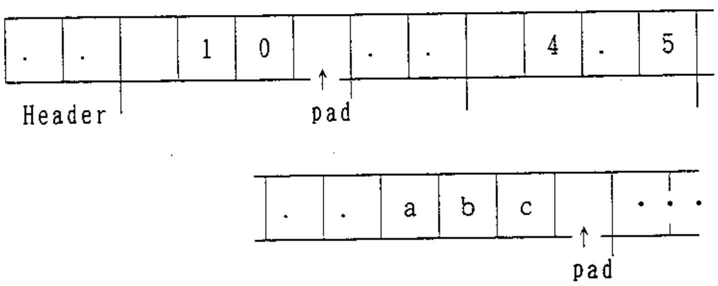
Note: For the information how to handle files, refer to "1. Preface ● File Management".

Example

```
10 OPEN "DATA.BAS" FOR OUTPUT AS #FD ; TEXT
20 OUTPUT #FD;10,4.5,"abc"
```




```
10 OPEN "DATA.BAS" FOR OUTPUT AS #FD ; ASCII
20 OUTPUT #FD;10,4.5,"abc"
```



41. OUTPUT

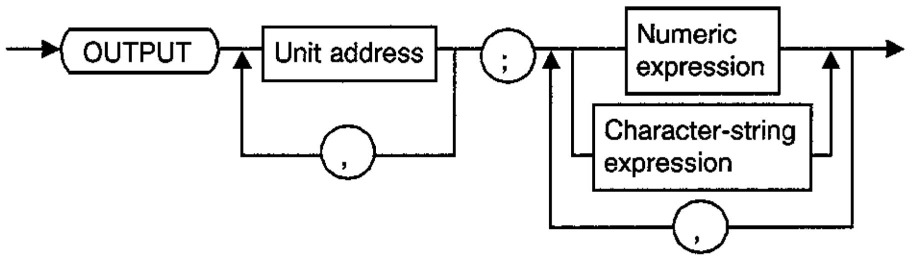
Outline

(1) The OUTPUT statement is used to output the data to GPIB or parallel port.

(2) The OUTPUT statement is used to output (write) the data to files.

Syntax

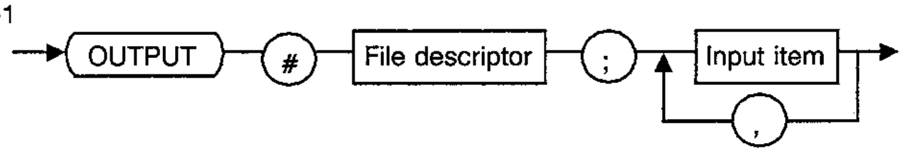
(1)-1



(1)-2
OUTPUT unit address {, unit address} ; <numeric expression | character-string expression> {, <numeric expression | character-string expression>}

Note: Unit address: 0 to 30; Address of the external GPIB device.
31; Output to the measurement section of the analyzer.
33; Output to the A port of parallel port.
34; Output to the B port of parallel port.
35; Output to the C port of parallel port and set/reset of Flip/Flop.
36; Output to the D port of parallel port and set of port mode.
37; Output to the CD port of parallel port.
Only when the unit addresses are between 0 and 30, plural unit addresses can be specified.

(2)-1



(2)-2
OUTPUT # file descriptor ; input item {, input item}

Description

Syntax of (1)

- The OUTPUT statement sends numeric and character string as an ASCII data to the specified device by the unit address.
  Multiple unit address can be specified by delimiting with a string of commas. The numeric expression and the character-string expression are used together by delimiting with a string of commas.

- If the OUTPUT statement is executed when the REN line is TRUE (Low level), the unit specified by the unit address will be automatically remote state. To cancel the remote state by the program, execute the LOCAL statement.

- Example

```
10 A=5
20 B=10
30 OUTPUT A;"STARTF", B,"MHz"
```

- Note

  In the SYSTEM CONTROLLER mode, the specified address device is set as the listener and the data is output.
  When the external listener is not existed, this command cannot be executed.

Syntax of (2)

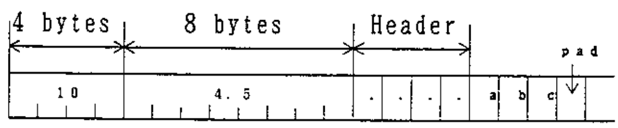
- The OUTPUT statement converts the data into the BASIC format and then outputs the file assigned to the file descriptor.
  The OUTPUT statement reads out the converted BASIC-format data and assigns it to its input item.

- Example 1: BINARY file

  Outputs data without changes. A character string is output with the header which indicates the length of 4-byte character string. If the number of character data is an odd, then one space of 1-byte will be followed after the data.

```
10 OPEN "FILE" FOR OUTPUT AS #FD
20 OUTPUT #FD;10,4.5,"abc"
```

Note: For the information how to handle files, refer to "1. Preface ● File Management".



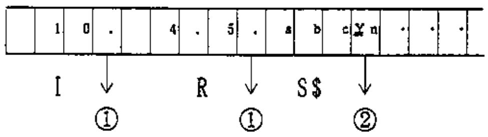
Header has each data length.

- Example 2: TEXT file

  Converts data into into ASCII codes and outputs the data.

  The signs (space or minus) for numeric data is placed to the top of the field.

```
10 OPEN "FILE" FOR OUTPUT AS #FD;TEXT
20 OUTPUT #FD;10,4.5,"abc"
```



①: Each item is delimited with a string of commas.

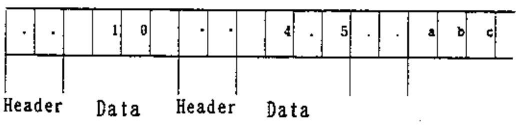
②: LF followed after the final item is output.

- Example 3: ASCII file

  Converts data into ASCII codes and outputs the data.

  The signs (space or minus) for numeric data is placed to the top of the field. If the number of character data is an odd, then one space of 1-byte will be followed after the data.

```
10 OPEN "FILE" FOR INPUT #FD;ASCII
20 OUTPUT #FD;10,4.5,"abc"
```



Header has each data length.

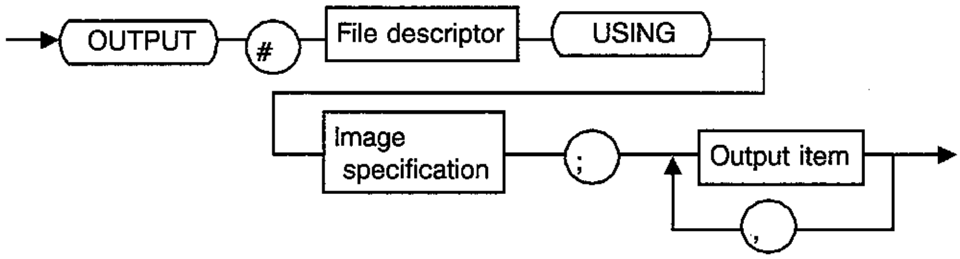
42. OUTPUT USING

Outline

The OUTPUT USING statement is used to output data with the specified data-type to the file assigned to the #file descriptor. Only the TEXT file is effective.

Syntax

(1)-1



(1)-2
OUTPUT # file descriptor USING image specification ; output item {, output item}

Note: OUT can be used instead of the OUTPUT, and USE for the USING.

Description

- When the USING and the image specification are specified, the format is converted and output. The image specification must be specified by character-string expression.
- The specified file descriptor when the file is opened is used. The file descriptor is assigned for the file to be objected at the file open. After that, the processing for the file can be performed through this file descriptor.

image specification

D: Specifies the output digits with No. of D. A space is used to fill up the remaining blank in the specified field.

Z: Specifies the output digits with No. of Z. A zero is used to fill up the remaining blank in the specified field.

K: Displays the expression as it is.

S: Displays the OUTPUT USING with a + or - sign flag at the position of S.

M: Displays the OUTPUT USING with a - for negative and a space for positive at the position of M.

.: Displays the OUTPUT USING to match the position "." with coming the decimal point.

E: Displays OUTPUT USING with the exponent format (e, sign, exponent).

H: Same as K. However, use a comma for a decimal point.

R: Same as ".". However, use a comma for a decimal point.

*: Specifies the output digit with the number of *. A space is used to fill up the remaining blank in the specified field.

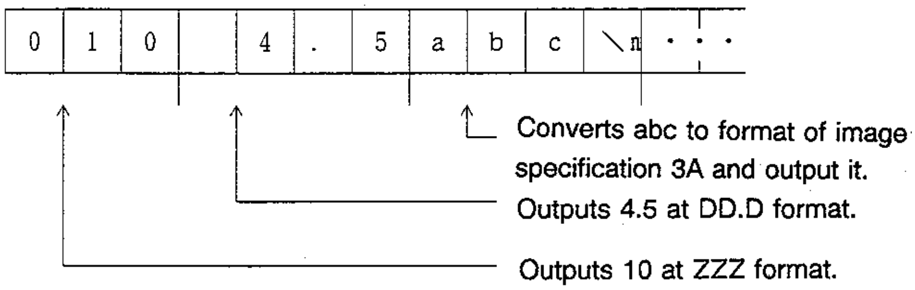
image specification

A:	Displays one character.
k:	Displays the character-string expression as it is.
Literal:	Encloses the literal with \" when writing it in the format expression.
X:	Displays the character of one space.
B:	Displays the expression result using an ASCII code.
@:	Outputs the form lead.
+:	Outputs the carriage return.
-:	Outputs the line feed.
#:	Does not hang the line feed immediately followed after the last item.
n:	Specifies the number of repetition of each image by using numerics. For example, 3D.2D is the same as for DDD.DD, and 4A for AAAA.

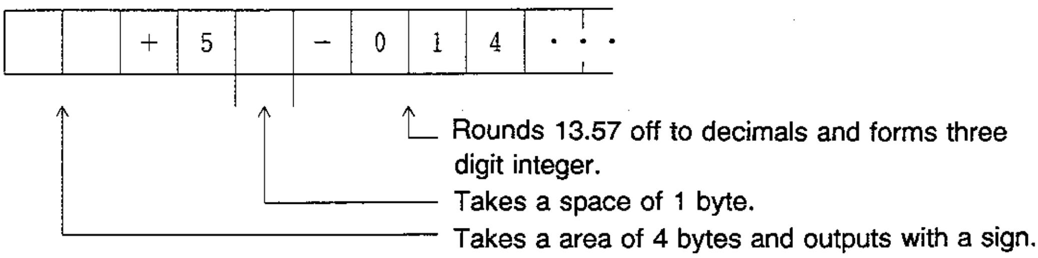
Note: For the information how to handle files, refer to "1. Preface ● File Management".

Example

```
OUTPUT #FD USING "ZZZ,DD.D,3A";10,4.5,"abc"
```




```
OUTPUT #FD USING "SDDD,X,MZZZ";+5,-13.57
```



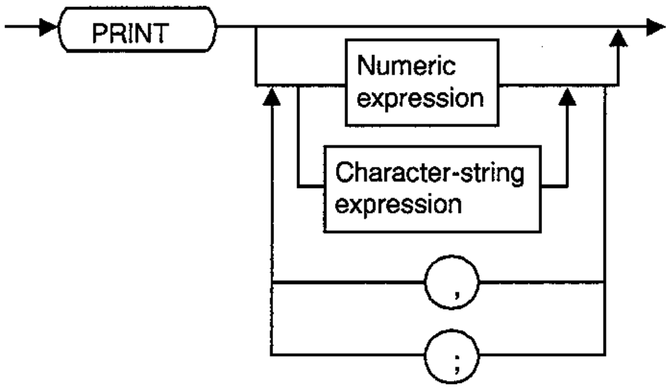
43. PRINT [USING]

Outline

The PRINT [USING] statement is used to display numerics or character strings.

Syntax

(1)-1



(1)-2

PRINT [numeric expression| character-string expression {, | ; numeric expression| character-string expression}]

Description

- The PRINT [USING] statement displays the specified numeric or character string.
- When the multiple numerics or character strings are delimited with a comma and specified, they are continuously output without LF.
- If a semicolon is used at the end of the PRINT statement, LF could not be performed after the termination of print out. Therefore, if the next PRINT statement is executed, the line followed after the previous output line will be output continuously.

Example

```
10 PRINT 123*456
20 PRINT "ABC"
30 PRINT "Freq.=",A, "Hz"
40 PRINT I,
```

- In PRINT USING format specification expression ; [[expression [···]]
  The format specification expression (character-string expression), specify the image specification by using a comma among image. The end of the format specification expression is automatically returned with line feed.

image specifications

D:	Specifies the output digits with No. of D. A space is used to fill up the remaining blank in the specified field.
Z:	Specifies the output digits with No. of Z. A zero is used to fill up the remaining blank in the specified field.
K:	Displays the expression as it is.
S:	Displays the PRINT USING format with a + or - sign flag at the position of S.
M:	Displays the PRINT USING format with a - for negative and a space for positive at the position of M.
.:	Displays the PRINT USING format to match the position "." with coming the decimal point.
E:	Displays PRINT USING format with the exponent format (e, sign, exponent).
H:	Same as K. However, use a comma for a decimal point.
R:	Same as ".". However, use a comma for a decimal point.
*:	Specifies the output digits with the number of *. A space is used to fill up the remaining blank in the specified field.
A:	Displays one character.
k:	Displays the character-string expression as it is.
X:	Displays the character of one space.
Literal:	Encloses a literal with \" when writing it to the format expression.
B:	Displays the expression result using an ASCII code.
@:	Form lead
+:	Moves the display position to the top of the same line.
-:	Line feed
#:	Does not line feed.
n:	Specifies the number of repetition of each image by using numerics. For example, 3D.2D is the same as for DDD.DD, and 4A for AAAA.

Example 1

```
10 PRINT USING "4Z,2X,5D,2X,5*" ;123,-444,567

<After the execution>
 0123   -444   **567
```

Example 2

```
10 PRINT USING "S3D,X,S3D" ;-4.5,465
20 PRINT USING "M3Z.Z,X,M3ZR3Z" ;1.26,-5.452

<After the execution>
  -5 +456
 001.3 -005.452
```

Example 3

```
10 PRINT USING "K,X,H" ;5.03884e+22,4.5563
```

<After the execution>

```
 5.03884e+22 4.5563
```

Example 4

```
10 PRINT USING "k,#" ;"character:"
20 PRINT USING "B" ;69
```

<After the execution>

```
 character:E
```

Example 5

```
10 PRINT USING "\" ............ \" ,+,A" ; "*"
20 PRINT USING "k,-, \" .END. \" " ;  "string"
```

<After the execution>

```
 *............
 string
 .END.
```

Example 6

	<After the execution>
100 PRINT USING "DDD.DD" ;1.2	1.20
110 PRINT USING "ZZZ.ZZ" ;1.2	001.20
120 PRINT USING "K" ;1.2	1.2
130 PRINT USING "SDDD.DD" ;1.2	+1.20
140 PRINT USING "MDDD.DD" ;1.2	1.20
150 PRINT USING "MDDD.DD" ;-1.2	-1.20
160 PRINT USING "H" ; 1.2	1,2
170 PRINT USING "DDDRDD" ; 1.2	1,20
180 PRINT USING "***.**" ; 1.2	**1.20
190 PRINT USING "A" ; "a"	a
200 PRINT USING "k" ; "string"	string
210 PRINT USING "B" ; 42	*
220 PRINT USING "3D.2D" ;1.2	1.20

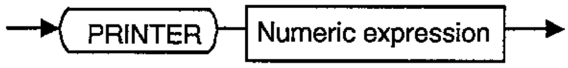
44. PRINTER

Outline

The PRINTER statement is used to specify the unit address for sending the data to the printer.

Syntax

(1)-1



(1)-2
PRINTER numeric expression

Description

- The PRINTER statement sets the printer unit address connected to the GPIB.
- Be sure to specify the printer unit address to the analyzer by the PRINTER statement before executing the GPRINT, GLIST and GLISTN statement.
- The unit address is the integers from 0 to 30.

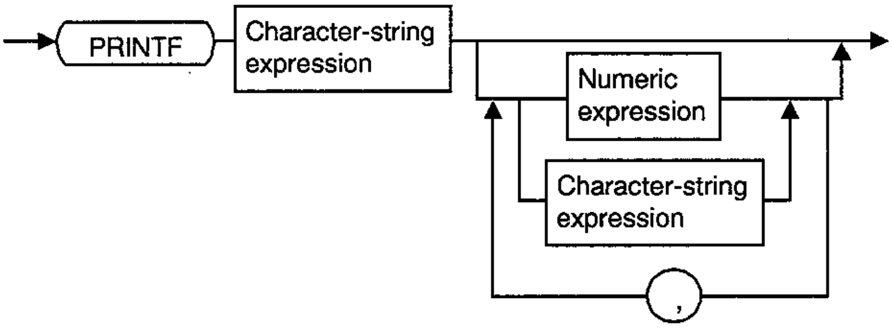
Example

```
10 PRINTER 1
```

45. PRINTF

Outline The PRINTF statement is used to display numerics or character strings.

Syntax (1)-1



(1)-2
PRINTF character-string expression [numeric expression | character-string expression {, numeric expression | character-string expression}]

Description

- The PRINTF statement displays the specified numeric or character string.
- When the multiple numerics or character strings are delimited with a comma and specified, they are continuously output without LF. To line feed, use a "*n" in the format specification expression.
- The first parameter character-string expression is used to specify the preceding parameter format.
- The following format specification are provided.

- PRINTF format specification expression ; [[expression [expression [···]]]
  The method of format specification is similarly to the Printf function of C language. The format specification expression is a character-string type and the output format is defined by the following method. The character string other than this format is normally output. If "%" is necessary, add "%" immediately followed after the "%".

  %[-] [0] [m] [. n] character

-:	Justifies the character with no space from left (if no specification, then from right).	
0:	Sets the character, which is justified for the remaining blank in the specified field, to be 0.	
m:	Reserves the field for the character "m".	
.n:	Outputs the PRINT USING format with n-digit accuracy. In character string, this setup value is used for an actual character-string length.	
Character:	d; decimal with sign	s; character string
	o; octal	e; floating-point expression (exponent format)
	x; hexadecimal	f; floating-point expression

Example

```
10 N = 500000
20 U = LOG(1+1/N)
30 V = U - 1 / N
40 PRINTF  "%7d %16.5e %16.5e \n" ,N,U,V
50 PRINTF  "%s\n" , "end"

<After the execution>
  500000 2.00000e-06 -1.99994e-12
 end
```

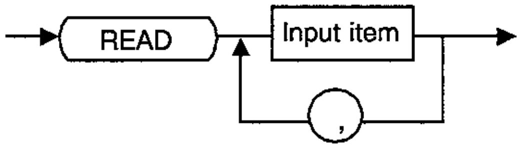
46. READ

Outline

The READ statement is used to assign the constant in the DATA statement to the variable.

Syntax

(1)-1



(1)-2

READ input item {, input item}

Description

- The READ statement reads the numeric or character string defined in the DATA statement to the variable specified by the argument.
- The READ statement catches the READ statement and searches the DATA statement in the program.
- In the first READ statement, basically (it must be changed by RESTORE statement), the READ searches the constant value from top line to final line in order, and the first searched value is assigned to the variable. After that, the constant corresponding to the DATA statement is searched and assigned to the variable.
- If the constant value specified the DATA statement is less, an error will occur.
- It is not necessary that the variable value read out by the READ statement and the constant value in one line of DATA statement are the same.

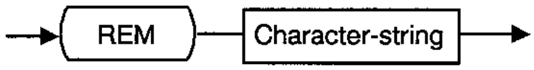
47. REM

Outline

The REM statement is an annotation for program.

Syntax

(1)-1



(1)-2
REM character-string

Description

- The REM statement is used to add the annotation to the program.
- Since the REM statement is no execution statement, any character string can be used followed after the REM statement. All the characters, numerics, and symbols can be used.
- An exclamation mark may be used instead of the REM statement.
- Multi statements using colons followed after the REM statement cannot be used. All the statements are determined as annotation statement.

Example

```
10 REM "PROGRAM 1"
20 ! 1983-JUN-02
30 A=A+1:! INCREMENT A
```

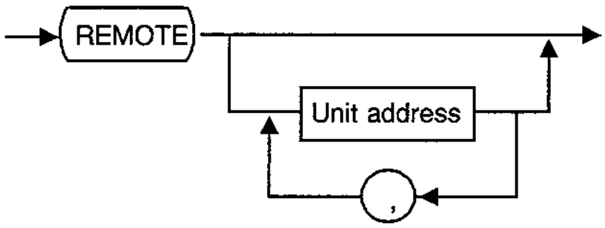
48. REMOTE

Outline

The REMOTE statement is used to set the specified unit to the remote state or to set the remote enable (REN) line to TRUE.

Syntax

(1)-1



(1)-2

REMOTE [unit address {, unit address }]

Description

- If only the REMOTE statement is executed without specifying the unit address, the remote enable (REN) line of the GPIB will become TRUE (Low level) and the unit connected on the GPIB will be set to the remote-controlled state. To set the REN line to FALSE (High level), execute the LOCAL statement.
- If the unit address followed after the REMOTE statement is specified, only the unit address specified by its unit address will be set to the remote-controlled state (only when the REN line is TRUE).
  Multiple unit addresses can be specified.
  To cancel the remote-controlled state, execute the LOCAL statement.
- The REMOTE statement is used to set the selected unit to the remote-controlled state, however, if the following statements are executed, then the specified unit will be automatically set to the remote-controlled state without executing the REMOTE statement.

  CLEAR [unit address {, unit address}]
  OUTPUT unit address {, unit address} ; <output data> {, <output data> }
  REMOTE [unit address {, unit address}]
  SEND LISTEN unit address {, unit address}
  TRIGGER unit address {, unit address}

Example

```
10 REMOTE 1
20 REMOTE 5
30 REMOTE 1 2 3
```

Note

The REMOTE statement is not available in the ADDRESSABLE mode.

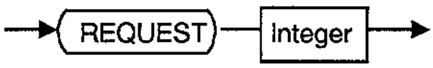
49. REQUEST

Outline

The REQUEST statement is used to set the status byte which is sent to the external GPIB controller in the ADDRESSABLE mode.

Syntax

(1)-1



(1)-2
REQUEST integer

Note: The setting range of integer is between 0 to 255.

Description

- The REQUEST statement sets the status byte which is sent to the external GPIB controller in the ADDRESSABLE mode.
- When the service request (SRQ) is transmitted, the values of 64 to 127 or 192 to 255 (bit 6 indicates "1") must be set.

Example

```
10 REQUEST 65
```

Note

- The REQUEST statement is not available in the SYSTEM CONTROLLER mode.
- Note that the serial pole is used to read (check) ? >the request signal< ? from an external controller. The STB? of the GPIB command cannot be used.
- When the SRQD of the GPIB command is executed, the bit 6 of the status byte is always transmitted with "0". Therefore, the SRQ is not transmitted.

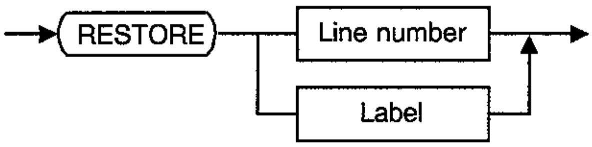
50. RESTORE

Outline

The RESTORE statement is used to specify the DATA line which is read out in the next READ statement.

Syntax

(1)-1



(1)-2
RESTORE

Description

- The line number is specified by the line number or label.
  Unless otherwise specified, the constant of the DATA statement is read out from the first line of the program in order, and the DATA statement which is objected for the next READ statement in the RESTORE statement.
- The line number of the argument is the first line number from which the DATA statement search is to start. Therefore, the DATA statement to be specified may be written on the line from which the DATA statement search is to start or any subsequent line.

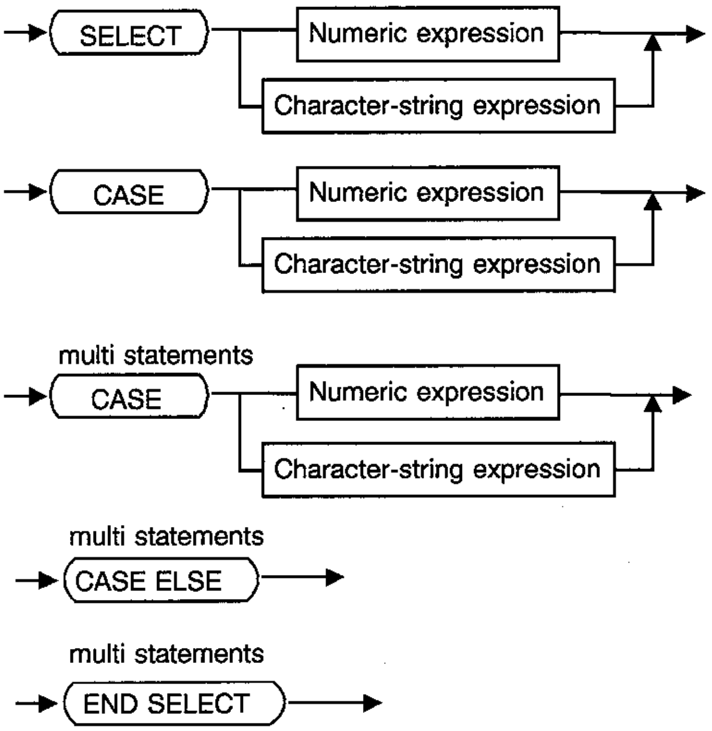
51. SELECT, CASE, ENS SELECT

Outline

This statement is used to perform the multiple brunches on condition of the one expression value.

Syntax

(1)-1



(1)-2

SELECT <numeric expression | character-string expression>
CASE <numeric expression | character-string expression>
 multi statements
CASE <numeric expression | character-string expression>
 multi statements
CASE ELSE
 multi statements
END SELECT

Description

- This statement executes the multiple statements which are agreed with the expression value specified by the SELECT statement followed after the CASE statement.
  The next statements such as CASE, CASE ELSE, or END SELECT can be objected for the execution.
- Nesting can be preformed in the SELECT statement. In this case, an internal SELECT statement includes the other statements.

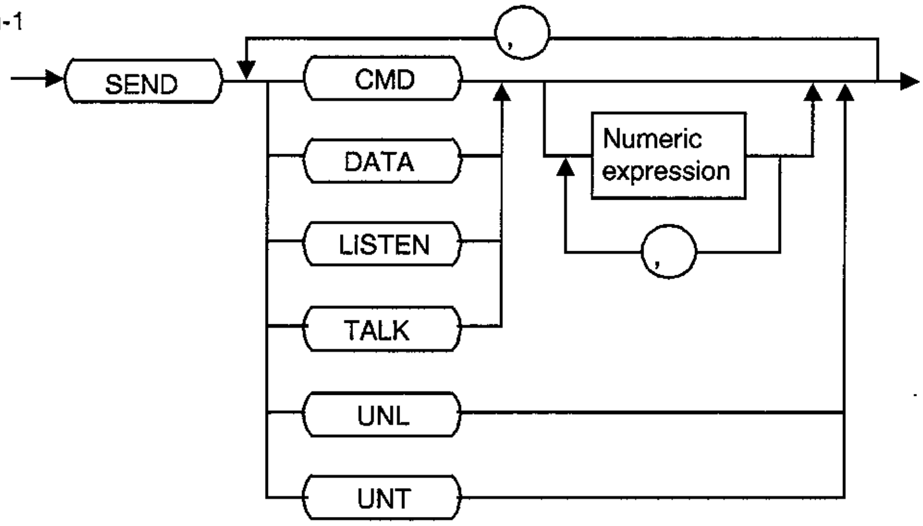
52. SEND

Outline

The SEND statement is used to output the command and data to a GPIB.

Syntax

(1)-1



(1)-2
SEND <A | B> {, <A | B> }

Note: A:<CMD | DATA | LISTEN | TALK> [numeric expression {, numeric expression}]
B:UNL | UNT

Description

- The SEND statement sends (transmits) the universal command, the address command, and the data independently to the GPIB.

CMD: Sets the ATN line to TRUE (Low level) and sends the numerics given to the GPIB. The numeric is converted into an 8-bit binary data and output to the GPIB. Therefore, the numerics to be used are the range of 0 to 255 and the numerics of decimal point e
xpression are automatically converted into integers.

DATA: Sets the ANT line to FALSE (High level) and sends the numerics given to the GPIB. The numerics to be used are the same as CMD.

LISTEN: Sends the numerics given to the GPIB as listener address group (LAG). Multiple numerics can be specified.

TALK: Sends the numerics given to the GPIB as talker address group (TAG). Multiple numerics cannot be specified.

UNT: Sends the UNT command to the GPIB. The talker (unit specified as talker before executing this command) can be canceled.

UNT: Sends the UNL command to the GPIB. The listener (unit specified as listener before executing this command) can be canceled.

Example

```
10 SEND UNT UNL LISTEN 1, 2, 3 TALK 4
20 SEND UNT CMD 63, 33 DATA 30,54
```

Note

The SEND statement is not available in the ADDRESSABLE mode.

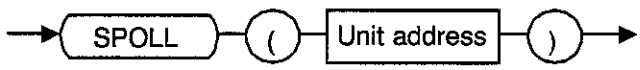
53. SPOLL

Outline

The SPOLL statement is used to perform the serial polling of the specified unit and to read out the status byte.

Syntax

(1)-1



(1)-2
SPOLL (unit address)

Description

- When the analyzer is set to the SYSTEM CONTROLLER mode, the SPOLL statement executes the serial polling for the other GPIB units.
- When the unit address is 0 to 30, the SPOLL statement executes the serial polling for the units corresponding to each address.
- When the unit address is 31, the SPOOL statement retrieves the status byte for the analyzer regardless of whether ?>the analyzer<? is set to the SYSTEM CONTROLLER mode or the ADDRESSABLE mode.

Example

```
10 OUTPUT 31;"OLDC ON"
20 ON ISRQ GOTO 70
30 ENABLE INTR
40 OUTPUT 31;"SRQE"
50 OUTPUT 31;"SINGLE"
60 GOTO 60
70 PRINT SPOLL(31)
80 STOP
```

Note

In the ADDRESSABLE mode, if the unit address between 0 to 30 is specified and the SPOLL is executed, the value "0" will be returned.

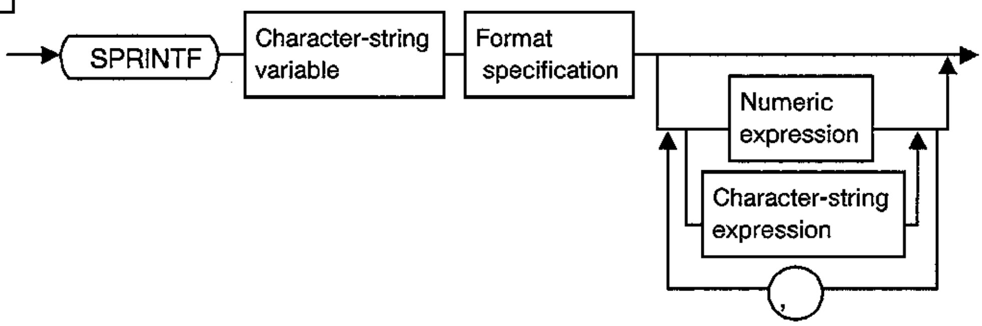
54. SPRINTF

Outline

The SPRINTF statement is used to convert the format in accordance with the format conversion of the PRINTF command and to assign the result to the character-string variable.

Syntax

(1)-1



(1)-2

SPRINTF character-string variable format specification [numeric expression | character-string expression {, numeric expression | character-string expression}]

Description

- The SPRINTF statement converts the expression value in accordance with the format conversion of the PRINTF command, and assigns the result to the character-string variable of first parameter.
- Pay attention to the format specification, the number of expression, and the character-string variable size for storing the result.
  If the character string for storing the result does not have enough capacity (free space), the BASIC buffer may be damaged.

  The method of format specification is refer to "45. PRINTF" of section 4.3.

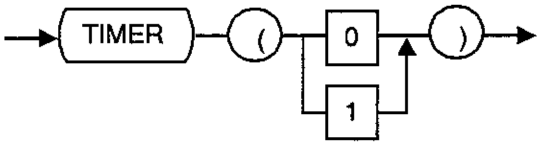
55. TIMER

Outline

The TIMER statement is used to read/reset the internal system time.

Syntax

(1)-1



(1)-2
TIMER (0 | 1)

Description

- The TIMER statement is the built-in function, which returns the internal system time with the unit of sec. This function is mainly used to check the measurement operation time.
  When the argument 0 is specified: Reads out the internal system time.
  When the argument 1 is specified: Resets the internal system time.
- The read out value with the resolution of 10msec includes an error of ±10msec.

Example

```
10  INTEGER I
20  TIMER(1)
30  FOR I=0 TO 10000
40  NEXT I
50  T1=TIMER(0)
60  !
70  TIMER(1)
80  FOR I=0 TO 10000
90    PRINT I
100 NEXT I
110 T2=TIMER(0)
120 !
130 PRINT "PRINT Command execute time is " ;T2-T1
140 STOP
```

56. TIME$

Outline

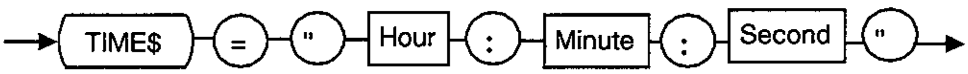
The TIME$ statement is used to read/set the time of the built-in timer.

Syntax

(1)-1

(1)-2
TIME$

(2)



(2)-2
TIME$ = "hour : minute : second"

Description

- The TIME$ statement reads out the time of the built-in timer (RTC).
- The TIME$ statement can change the time which is read out.
  Input as follows:

```
TIME$="23:43:12"
TIME$="11:5:6"
```

Example

```
10 DIM T$[10]
20 T$=TIME$
30 PRINT "Time is "; T$
40 PRINT "Time Reset"
50 TIME$="0:0:0"
60 STOP
```

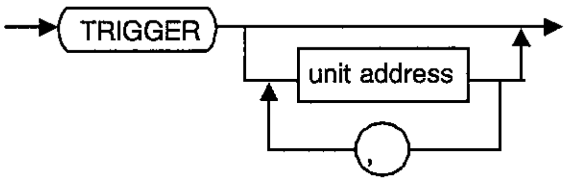
57. TRIGGER

Outline

The TRIGGER statement is used to send the group execute trigger (GET) of address command group (ACG) to the all units connected to the GPIB or to the particular unit selected.

Syntax

(1)-1



(1)-2
TRIGGER [unit address {, unit address }]

Description

- If only the TRIGGER statement is executed without specifying the unit address, only the the group execute trigger (GET) of address command will be transmitted. In this case, the unit to be triggered must be set as listener in advance.
- If the unit address followed after the TRIGGER statement is specified, the GET command will be transmitted to only the unit address specified by its unit address.

Example

```
10 TRIGGER 1
20 TRIGGER
```

Note

The TRIGGER statement is not available in the ADDRESSABLE mode.

58. WAIT

Outline

The WAIT statement is used to wait for the specified time.

Syntax

(1)-1

(1)-2

WAIT time

Description

- The WAIT statement waits for the specified time. The unit of time is msec. The setting range of time between 0 to 65535.

Example

```
10 INTEGER T
20 T=30
30 PRINT T;"[msec] Wait !!"
40 WAIT T
50 STOP
```

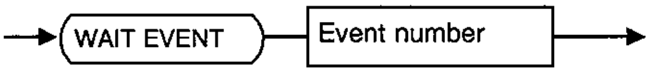
59. WAIT EVENT

Outline

The WAIT EVENT statement is used to wait the event until the specified event is generated.

Syntax

(1)-1



(1)-2
WAIT EVENT event number

Description

- The WAIT EVENT statement waits the event until the specified event number is generated.

  Event number: 1; sweep end

Example

```
10 INTEGER EV
20 EV=1
25 OUTPUT 31;"OLDC OFF"
30 OUTPUT 31;"INIT:CONT OFF;:ABOR;INIT"
40 WAIT EVENT EV
50 PRINT "SWEEP FINISHED"
60 STOP
```

4.4 Built-in Function

4.4.1 Outline

The Built-in function is a function which is built into the analyzer and can perform a high-speed processing. The data measured with a network analyzer by using the built-in function.
The built-in function is available for analyzing or judging the measured data. The basic function is used similarly as the existing network analyzer R3751, however, care is taken to partially added or deleted functions. Also the processing speed is improved.
The numeric values in the built-in function cannot specify the device. Any value is managed as a standard device.

Example: When calculating 10KHz address point
P = POINT2(10000,0)

Also the response data from the built-in function is similarly processed as the numeric value of the standard unit.

(1) Measurement data and address point

Use the address point for specifying the analysis range of the measurement data or the position in the measurement data. The address point specifies the measurement data by using the value of 0 through 1200. The measurement point is corresponded as follows:

- When the measurement point number is 1201

First data	Address point 0
2nd data	Address point 1
3rd data	Address point 2
⋮	
n-th data	Address point n-1
⋮	
1201st data	Address point 1200

- When the measurement point number is 601

First data	Address point 0
2nd data	Address point 2
3rd data	Address point 4
⋮	
n-th data	Address point 2(n-1)
⋮	
601st data	Address point 1200

- When the measurement point number is 301

First data	Address point 0
2nd data	Address point 4
3rd data	Address point 8
⋮	
n-th data	Address point 4(n-1)
⋮	
301st data	Address point 1200

Thus at the measurement point of 1200, the address point increases 1 and at the another point, it increases 1 or more.

Relation between measurement point number and addition value of address point is as follows:

Measurement point number	Addition value of address point	Measurement point number	Addition value of address point
1201	1	101	12
801*	1	51	24
601	2	21	60
401	3	11	120
301	4	6	240
201	6	3	600

Also this relation applies to user sweep and program sweep. When the user sweep and the program sweep are executed in the measurement point of 1201, the addition point of address point is always 1. The data is arranged at the beginning of the address point, 0. When the measurement point number is set to 601, further the total of the segment point number doesn't excess 601, the measurement data is arranged every other point. Also if an address point is specified when the measurement point number is changed, the specification of built-in function is not needed to be changed.

*: When the measurement point is 801, the addition value of address point is 1. If 801 to 1200 points are specified, error arises.

(2) Analysis channel

In the analysis channel, the analyzed data is specified by the built-in function. The data to be analyzed in the analyzer is as follows. The complex number data cannot be used for the analysis, but can be used for the data transmission.

① Display data
② Main trace data
③ Sub trace data
④ Main trace complex number data
⑤ Sub trace complex number data

Analysis channel specification for these data is as follows.

① Display data

In the display data, the displayed data is stored. The stored data is changed by the display format or the specification of the measure. The contents of memory data are unsettled.

Each measurement channel and analysis channel

CH1	CH2	CH3	CH4	
0	1	4	5	Measurement display first waveform data *1
8	9	12	13	Measurement display second waveform data *2
2	3	6	7	Memory display first waveform data *3
10	11	14	15	Memory display second waveform data *4

*1 : When 1 waveform is displayed in 1 screen, the display data is stored. When 2 waveforms are displayed in 1 screen, the first waveform is stored.
The first waveform : S11 when the format is LOGMAG&PHASE, further LOGMAG measure is S11&S21.

*2 : When 1 waveform is displayed in 1 screen, the contents are unsettled. When 2 waveforms are displayed in 1 screen, the second waveform is stored.
The second waveform : S21 when the format is LOGMAG&PHASE, further PHASE measure is S11&S21.

*3 : When the copy is not performed to the memory, the contents are unsettled.

*4 : Even if the copy is performed to the memory, if the waveform display is not the second one then, the contents are unsettled.

② Main trace data

The trace data is the data to be the display data. LOGMAG, phase, real number part, and imaginary number part data are stored as internal data. Since these internal data are kept regardless of the display format, it's effective to analyze the data which is not in the display data. This data is not changed even if the display data operates 'smoothing'.
When 1 screen has 2 measurement data like S11&S21, each waveform is called as follows in order to distinguish.
The trace data which corresponds to the first waveform : Main trace data
The trace data which corresponds to the second waveform : Sub trace data
In the case like S11 and S21, the trace data is always main one.

Each measurement channel and analysis channel

CH1	CH2	CH3	CH4	
32	36	48	52	LOGMAG data *1
33	37	49	53	Phase data *1
34	38	50	54	Real part data *1
35	39	51	55	Imaginary part data *1
40	44	56	60	LOGMAG data of memory *2
41	45	57	61	Phase data of memory *2
42	46	58	62	Real part data of memory *2
43	47	59	63	Imaginary part data of memory *2

*1 : If the measurement is not performed on the specified channel, the contents become indefinite.

*2 : If the copy to the memory is not performed, the contents become indefinite.

③ Sub trace data

Each measurement channel and analysis channel

CH1	CH2	CH3	CH4	
64	68	80	84	LOGMAG data *1
65	69	81	85	Phase data *1
66	70	82	86	Real part data *1
67	71	83	87	Imaginary part data *1
72	76	88	92	LOGMAG data of memory *2
73	77	89	93	Phase data of memory *2
74	78	90	94	Real part data of memory *2
75	79	91	95	Imaginary part data of memory *2

*1 : If the measurement is not performed on the specified channel, the contents become indefinite.

*2 : If the copy to the memory is not performed, the contents become indefinite.

④ Main trace complex number data

When treating the internal complex number data, only the data transmission like TRANSR or TRANSW can be performed.

Each measurement channel and analysis channel

CH1	CH2	CH3	CH4	
128	192	256	320	Trace data *1
132	196	260	324	Trace memory data *2
129	193	257	321	Data after corrective operation *1
130	194	258	322	Memory data after corrective operation *2
131	195	259	323	Data before corrective operation *1
133	197	261	325	Normalize standard data *3
134	198	262	326	1 port correction : Direction error coefficient *3
135	199	263	327	1 port correction : Source match error coefficient *3
136	200	264	328	1 port correction : Reflection tracking error coefficient *3
137	201	265	329	2 port correction : Forward direction error coefficient *4
138	202	266	330	2 port correction : Forward direction source match error coefficient *4
139	203	267	331	2 port correction : Forward direction reflection tracking error coefficient *4
140	204	268	332	2 port correction : Forward direction load match error coefficient *4
141	205	269	333	2 port correction : Forward direction transmission tracking error coefficient *4
142	206	270	334	2 port correction : Forward direction isolation error coefficient *4
143	207	271	335	2 port correction : Reverse direction error coefficient *4
144	208	272	336	2 port correction : Reverse direction source match error coefficient *4
145	209	273	337	2 port correction : Reverse direction reflection tracking error coefficient *4
146	210	274	338	2 port correction : Reverse direction load match error coefficient *4
147	211	275	339	2 port correction : Reverse direction transmission tracking error coefficient *4
148	212	276	340	2 port correction : Reverse direction isolation error coefficient *4
149	213	277	341	Normalize & Isolation correction : Normalize standard data *3
150	214	278	342	Normalize & Isolation correction : Isolation error coefficient *3

*1 : If the measurement is not performed on the specified channel, the contents become indefinite.

*2 : If the copy to the memory is not performed, the contents become indefinite.

*3 : If the correction is not performed, the contents become indefinite.

*4 : If the correction is not performed, the contents become indefinite. The contents of CH1 and CH3, and CH2 and CH4 correction data become the same.

⑤ Sub trace complex number data

Sub trace complex number data is assigned as follows.

Each measurement channel and analysis channel

CH1	CH2	CH3	CH4	
160	224	288	352	Trace data *1
164	228	292	356	Trace memory data *2
161	225	289	353	Data after corrective operation *1
162	226	290	354	Memory data after corrective operation *2
163	227	291	355	Data before corrective operation *1
165	229	293	357	Normalize standard data *3
166	230	294	358	1 port correction : Direction error coefficient *3
167	231	295	359	1 port correction : Source match error coefficient *3
168	232	296	360	1 port correction : Reflection tracking error coefficient *3
169	233	297	361	2 port correction : Forward direction error coefficient *4
170	234	298	362	2 port correction : Forward direction source match error coefficient *4
171	235	299	363	2 port correction : Forward direction reflection tracking error coefficient *4
172	236	300	364	2 port correction : Forward direction load match error coefficient* 4
173	237	301	365	2 port correction : Forward direction transmission tracking error coefficient *4
174	238	302	366	2 port correction : Forward direction isolation error coefficient *4
175	239	303	367	2 port correction : Reverse direction error coefficient *4
176	240	304	368	2 port correction : Reverse direction source match error coefficient *4
177	241	305	369	2 port correction : Reverse direction reflection tracking error coefficient *4
178	242	306	370	2 port correction : Reverse direction load match error coefficient *4
179	243	307	371	2 port correction : Reverse direction transmission tracking error coefficient *4
180	244	308	372	2 port correction : Reverse direction isolation error coefficient *4
181	245	309	373	Normalize & Isolation correction : Normalize standard data *3
182	246	310	374	Normalize & Isolation correction : Isolation error coefficient *3

*1 : If the measurement is not performed on the specified channel, the contents become indefinite.

*2 : If the copy to the memory is not performed, the contents become indefinite.

*3 : The command which can be used in controller mode was used in addressable mode.

*4 : The command which can be used in addressable mode was used in controller mode.

(3) Response formats for built-in function

Response formats for built-in function are provided for three types.

- Measurement point: Address point including measurement data.
  Example; MAX function
- Address point: At other than measurement point, interpolate to set the value of address point.
  Example; VALUE function
- Compensate: Interpolate to set a value.
  Example; CVALUE function

4.4.2 List of Built-In Function

- Address point relation

POINT1(F,C):	meas point;	Measurement point closed to specified frequency
POINT2(F,C):	address point;	Address point closed to specified frequency
DPOINT(F0,F1,C):	address point;	Address point width corresponding to specified frequency width
POINT1L(F,C):	meas point;	Max. measurement point less than specified frequency
POINT1H(F,C):	meas point;	Min. measurement point more than specified frequency
POINT2L(F,C):	address point;	Max. address point less than specified frequency
POINT2H(F,C):	address point;	Min. address point more than specified frequency
SWPOINT(C):	meas point;	Latest measurement point

- Frequency relation

FREQ(P,C):	address point;	Frequency corresponding to specified address point
DFREQ(P0,P1,C):	address point;	Frequency width corresponding to specified address point width
SWFREQ(C):	meas point;	Latest measurement frequency

- Response relation

VALUE(P,C):	address point;	Response value in specified address point
DVALUE(P0,P1,C):	address point;	Difference of response values between specified address points
CVALUE(F,C):	compensate;	Response value in specified frequency
DCVALUE(F0,F1,C):	compensate;	Difference of response values between specified frequencies
SWVALUE(C):	meas point;	Latest response value

- Max. value/Min. value relation

MAX(P0,P1,C):	meas point;	Max. response value between specified address points
FMAX(P0,P1,C):	meas point;	Max. response frequency between specified address points
PMAX(P0,P1,C):	meas point;	Measurement point in max. response between specified address points
MIN(P0,P1,C):	meas point;	Min. response value between specified address points
FMIN(P0,P1,C):	meas point;	Min. response frequency between specified address points
PMIN(P0,P1,C):	meas point;	Measurement point in min. response between specified address points

- Bandwidth relation

BND(P,X,C):	compensate;	Bandwidth attenuating specified data from specified address point
BNDL(P,X,C):	compensate;	Frequency in low frequency side attenuating specified data from specified address point
BNDH(P,X,C):	compensate;	Frequency in high frequency side attenuating specified data from specified address point
CBND(F,X,C):	compensate;	Bandwidth attenuating specified data from specified address point
CBNDL(F,X,C):	compensate;	Frequency in low frequency side attenuating specified data from specified frequency
CBNDH(F,X,C):	compensate;	Frequency in high frequency side attenuating specified data from specified frequency
MBNDI(P0,P1,P,N,La,Fa,C):	compensate;	Frequency in low frequency side, frequency in high frequency side, center frequency and bandwidth attenuating specified data from specified address point between specified address points
MBNSO(P0,P1,P,N,La,Fa,C):	compensate;	Frequency in low frequency side, frequency in high frequency side, center frequency and bandwidth attenuating specified data from specified address point between specified address points

- Ripple relation-1

RPL1(P0,P1,dX,dY,C):
meas point; Difference in max. value and min. value between specified address points

RPL2(P0,P1,dX,dY,C):
meas point; Max. value of difference in max. value and min. value adjoining between specified address points

RPL3(P0,P1,dX,dY,C):
meas point; Max. value adding difference in max. value and min. value adjoining between specified address points

RPL4(P0,P1,dX,dY,C):
meas point; Max. point of difference in max. value and min. value adjoining between specified address points

RPL5(P0,P1,dX,dY,C):
meas point; Largest value of max. value between specified address points

RPL6(P0,P1,dX,dY,C):
meas point; Smallest value of max. value between specified address points

RPLF(P0,P1,dX,dY,C):
meas point; Frequency difference in first max. value and min. value between specified points

RPLR(P0,P1,dX,dY,C):
meas point; Response difference in first max. value and min. value between specified points

RPLH(P0,P1,dX,dY,C):
meas point; Response value in first max. value between specified address points

FRPLH(P0,P1,dX,dY,C):
meas point; Frequency in first max. value between specified address points

PRPLH(P0,P1,dX,dY,C):
meas point; Measured point in first max. value between specified address points

RPLL(P0,P1,dX,dY,C):
meas point; Response value in first min. value between specified address points

FRPLL(P0,P1,dX,dY,C):
meas point; Frequency in first min. value between specified address points

FRPLL(P0,P1,dX,dY,C):
meas point; Measured point in first min. value between specified address points

- Ripple relation-2

NRPLH(P0,P1,dX,dY,C):	meas point;	Nos. of max. point between specified address points
NRPLL(P0,P1,dX,dY,C):	meas point;	Nos. of min. point between specified address points
PRPLHN(N,C):	meas point;	Measured point in N-th max. value with NRPLH
PRPLLN(N,C):	meas point;	Measured point in N-th min. value with NRPLL
FRPLHN(N,C):	meas point;	Frequency in N-th max. value with NRPLH
FRPLLN(N,C):	meas point;	Frequency in N-th min. value with NRPLL
VRPLHN(N,C):	meas point;	Response value in N-th max. value with NRPLH
VRPLLN(N,C):	meas point;	Response value in N-th min. value with NRPLL
PRPLHM(Pa,C):	meas point;	Measured point array in max. value with NRPLH
PRPLLM(Pa,C):	meas point;	Measured point array in min. value with NRPLL
FRPLHM(Xa,C):	meas point;	Frequency array in max. value with NRPLH
FRPLLM(Xa,C):	meas point;	Frequency array in min. value with NRPLL
VRPLHM(Xa,C):	meas point;	Response value array in max. value with NRPLH
VRPLLM(Xa,C):	meas point;	Response value array in min. value with NRPLL

- Direct search relation

DIRECT(P0,P1,X,C):	address point;	Address point closed to first detected data between specified address points
DIRECTL(P0,P1,X,C):	meas point;	Measured point in first detected data by search of low frequency side between specified address points
DIRECTH(P0,P1,X,C):	meas point;	Measured point in first detected data by search of high frequency side between specified address points
CDIRECT(F0,F1,X,C):	compensate;	Frequency in first detected data between specified frequencies
CDIRECTL(F0,F1,X,C):	compensate;	Frequency in first detected data by search of low frequency side
CDIRECTH(F0,F1,X,C):	compensate;	Frequency in first detected data by search of high frequency side between specified frequencies
DDIRECT(P0,P1,X,C):	address point;	Address point width in specified data between specified address points
CDDIRECT(F0,F1,X,C):	compensate;	Bandwidth in specified data between specified frequencies
ZEROPHS(P0,P1,C):	compensate;	Frequency in zero (0) phase between specified address points

- Data transfer relation

TRANSR(P0,P1,Xa,C):

meas point;	Transfer of measured data between specified address points to array

TRANSW(P0,P1,Xa,C):

meas point;	Transfer from array to specified address point

P,P0,P1:	Address point specification
F,F0,F1:	Frequency specification
C:	Analysis channel specification
dX:	Gradient horizontal axis specification
dY:	Gradient vertical axis specification
X:	Level specification
N:	Number(s) and N-th specification
Xa,La,Fa:	Array specification
Pa:	Integer array specification

4.4.3 Function Obtaining Address Point

(1) Functions which obtains measurement point POINT1, POINT1L, POINTIH

```
POINT1   (frequency, analysis channel)
POINT1L  (frequency, analysis channel)
POINT1H  (frequency, analysis channel)
```

Explanation: Obtain a measurement point in specified frequency.

POINT1 function:	Obtains the measurement point closed to specified frequency. Round to the nearest whole number by conversion to measured point.
POINT1L function:	Obtains the largest measurement point less than specified frequency. Omit the figures by conversion to measured point.
POINT1H function:	Obtains the smallest measurement point more than specified frequency. Raise to a unit by conversion to measured point.

Usage: Most built-in functions have set an address point to an argument. For using other built-in functions, convert a frequency to a measurement point. When analysis range is specified, raising to a unit or omitting is accurate for specifying the range.

Example: P0 = POINT1L(F0,0)
P1 = POINT1H(F1,0)
X = MAX(P0,P1,0) Search the max. value in the range including the frequency, F0, F1.

P = POINT1(F,0)
Y = VALUE(P,0) Read out the measured data closed to the frequency, F.

(2) Functions which obtains address point POINT2, POINT2L, POINT2H

POINT2	(frequency, analysis channel)
POINT2L	(frequency, analysis channel)
POINT2H	(frequency, analysis channel)

Explanation: Obtain an address point in specified frequency.

POINT2 function: Obtains the address point closed to specified frequency. Round to the nearest whole number by conversion to address point.

POINT2L function: Obtains the largest address point less than specified frequency. Omit the figures by conversion to address point.

POINT2H function: Obtains the smallest address point more than specified frequency. Raise to a unit by conversion to address point.

Usage: Most built-in functions have set an address point to an argument. For using other built-in functions, convert a frequency to an address point.

Example: P = POINT2(F,0)
Y = VALUE(P,0)

Read out the measured data closed to the frequency, F, measured data at measurement point and at other cases interpolate to read out.

(3) Function which obtains address point width DPOINT

DPOINT (frequency1, frequency2, analysis channel)

Explanation : Obtain an address point width corresponding to frequency width.

(4) Function which obtains the latest measurement point SWPOINT

SWPOINT (analysis channel)

Explanation: Calculate the latest measurement point during sweep.

Usage: Sweep condition is shown by using SWPOINT (analysis channel).
As the following example, the data swept during the sweep can be analyzed.

Example:

```
*SWEEPING1
 IF SWPOINT(0)<P1 THEN GOTO *SWEEPING1
 X=MAX(P0,P1,0)
```

CAUTION

When this unit is sweeping at high speed, the measured point is intermittently read out.

4.4.4 Function Obtaining Frequency

(1) Function which obtains frequency FREQ

FREQ (address point, analysis channel)

Explanation: Convert address point to frequency.

Usage: Convert the function value which returns address point to frequency

Example:

```
P=PMAX(0,1200,0)
F=FREQ(P,0)
X=VALUE(P,0)
```

Obtain the max. frequency and response value. Calculate at the higher speed since the search is once executed without using MAX, FMAX.

(2) Function which obtains frequency width DFREQ

DFREQ (address point1, address point2, analysis channel)

Explanation: Convert from specified address point to frequency width.

(3) Function which obtains latest width SWFREQ

SWFREQ (analysis channel)

Explanation: Obtain the latest measurement frequency during measurement.

Usage: Sweeping frequency are shown by using SWFREQ(analysis channel).

Example:
```
*SWEEPING1
IF SWFREQ(0)<F1 THEN GOTO *SWEEPING1
X=CVALUE(F1)
```

CAUTION

When this unit is sweeping at high speed, the measured point is intermittently read out.

4.4.5 Function Obtaining Response

(1) Function which obtains response VALUE

VALUE (address point, analysis channel)

Explanation: Read out response in specified address point. When address point is not measurement point, interpolate to obtain.

Usage: Convert the function value which returns address point to response value.

Example:
```
P=PMAX(0,1200,0)
F=FREQ(P,0)
X=VALUE(P,0)
```

Obtain the max. frequency and response value. Calculate at the higher speed since the search is once executed without using MAX, FMAX.

(2) Function which obtains response difference DVALUE

DVALUE (address point1, address point2, analysis channel)

Explanation: Obtain each difference of response value in specified address point.

(3) Function which obtains response value CVALUE

CVALUE (frequency, analysis channel)

Explanation: Obtain response value corresponding to specified frequency.

(4) Function which obtains response difference DCVALUE

DCVALUE (frequency1, frequency2, analysis channel)

Explanation: Calculate each difference of response values in specified frequency.

(5) Function which obtains latest response value SWVALUE

SWVALUE (analysis channel)

Explanation: Obtain the latest measured response value during measurement.

Usage: Available for adjustment by monitoring a response value.

Example:

```
*ADJUST
IF SWVALUE(33)< =PHASE1 THEN GOTO *ADJUST_END
OUTPUT 33;C
GOTO *ADJUST
*ADJUST_END
```

Output to parallel I/O till a phase value drops less than a designated value.

CAUTION

When this unit is sweeping at high speed, the measured point is intermittently read out.

4.4.6 Function calculating Max. value, Min. value

(1) Function which calculates max. response value MAX

MAX (start address point, end address point, analysis channel)

Explanation: Searches max. response value between specified address points.

Usage: Used when the response value of resonance point is calculated.

Example: X = MAX(0,1200,0)

(2) Function which obtains the frequency of max. response FMAX

FMAX (start address point, end address point, analysis channel)

Explanation: Calculates the frequency of max. response between specified address points.

Usage: Used when the frequency of resonance point is calculated.

Example: F = FMAX(0,1200,0)

(3) Function which obtains the measurement point of max. response PMAX

PMAX (start address point, end address point, analysis channel)

Explanation: Calculates the measurement point of max. response between specified address points.

Usage: Used when the frequency of resonance point, response value or also address point in another analysis is obtained.

Example 1:
```
P = PMAX(0,1200,0)
F = FREQ(P,0)
X = VALUE(P,0)
```

Obtain the frequency and response value from the measured point in the max. value. Calculate at the higher speed since the search is once executed, compared with the use of MAX, FMAX.

Example 2:
```
P = PMAX(0,1200,0)
FB = BND(P,3,0)
```

Obtain the bandwidth of -3dB from peak value.

(4) Function which obtains min. response value MIN

MIN (start address point, end address point, analysis channel)

Explanation: Search the min. response value between specified address points.

Usage: Used when the response value of anti-resonance point is obtained.

Example: X = MIN(0,1200,0)

(5) Function which obtains the frequency of min. response FMIN

FMIN (start address point, end address point, analysis channel)

Explanation: Calculates the frequency of min. response between specified address points.

Usage: Used when the frequency of anti-resonance point is obtained.

Example: F = FMIN(0,1200,0)

(6) Function which obtains the measurement point of min. response PMIN

PMIN (start address point, end address point, analysis channel)

Explanation: Calculates the measurement point of minx. response between specified address points.

Usage: Used when the frequency of anti-resonance point and response value are obtained.

Example:
```
P = PMIN(0,1200,0)
F = FREQ(P,0)
X = VALUE(P,0)
```

Obtain the frequency and response value from the measured point in the min. value. Calculate at the higher speed since the search is once executed, compared with the use of FMIN, MIN.

4.4.7 Function Obtaining Bandwidth, etc.

(1) Function which obtains bandwidth BND

BND (address point, attenuation level, analysis channel)

Explanation: Obtain the bandwidth by searching the point which attenuated the specified attenuation level value from the specified address point.
The search is executed outside the specified address point.

Usage: Obtain 3db less bandwidth, etc.

Example:
```
P=PMAX(0,1200,0)
F=BND(P,3,0)
```

Obtain 3db less bandwidth.

(2) Function which obtains frequency of low frequency side in bandwidth BNDL

BNDL (address point, attenuation level, analysis channel)

Explanation: Obtain the frequency by searching the point to the low frequency side, which attenuated the specified attenuation level value from the specified address point. The search is executed outside the specified address point.

Usage: Obtain center frequency, combined with BNDH.

(3) Function which obtains frequency of high frequency side in bandwidth BNDH

BNDH (address point, attenuation level, analysis channel)

Explanation: Obtain the frequency by searching the point to the high frequency side, which attenuated the specified attenuation level value from the specified address point. The search is executed outside the specified address point.

Usage: Obtain center frequency, combined with BNDL.

Example:
```
P=PMAX(0,1200,0)
FH=BNDH(P,3,0)
FL=BNDL(P,3,0)
FB=FH-FL
FC=(FL+FH)*0.5
```

(4) Function which obtains bandwidth CBND

CBND (frequency, attenuation level, analysis channel)

Explanation: Obtain the bandwidth by searching the point which attenuated the specified attenuation level value from the specified frequency.
The search is executed outside the specified address point.

Usage: Obtain 3db less bandwidth, etc.

Example: F = BND(F,3,0)

Obtain 3db less bandwidth.

(5) Function which obtains frequency of low frequency side in bandwidth CBNDL

CBNDL (frequency, attenuation level, analysis channel)

Explanation: Obtain the frequency by searching the point to the low frequency side, which attenuated the specified attenuation level value from the specified frequency.

Usage: Obtain center frequency, combined with CBNDH.

(6) Function which obtains frequency of high frequency side in bandwidth CBNDH

CBNDH (frequency, attenuation level, analysis channel)

Explanation: Obtain the frequency by searching the point to the low frequency side, which attenuated the specified attenuation level value from the specified frequency.

Usage: Obtain center frequency, combined with CBNDL.

Example:

```
FH = CBNDH(F,3,0)
FL = CBNDL(F,3,0)
FB = FH-FL
FC = (FL + FH)*0.5
```

(7) Function which obtains bandwidth analysis for multiple attenuation levels MBNDI

MBNDI (start address point, end address point, standard address point, nos, of attenuation level, attenuation level array, array storing analysis result such as bandwidth, analysis channel)

Explanation: Multiple attenuation levels are once analyzed. Outputs four types of frequency in low frequency side, frequency in high frequency side, center frequency and bandwidth to one attenuation level.
The attenuation level is specified in array and the analysis result is stored in array. The search is executed outside the specified address point. The array for attenuation level should be in order of low level.

Usage: Calculate at high speed when multiple attenuation levels are analyzed.
Available when four frequencies are required to one attenuation level.

Example:

```
DIM L(3), F(3,4)
L(1)=1.0
L(2)=3.0
L(3)=10.0
P   =PMAX(0,1200,0)
N   =MBNDI(0,1200,P,3,L(1),F(1,1),0)
```

In this case, the followings are stored in the array F.

F(1,1)	Frequency in low frequency side at attenuation level of 1.0
F(1,2)	Frequency in high frequency side at attenuation level of 1.0
F(1,3)	Center frequency at attenuation level of 1.0
F(1,4)	Bandwidth at attenuation level of 1.0
F(2,1)	Frequency in low frequency side at attenuation level of 3.0
F(2,2)	Frequency in high frequency side at attenuation level of 3.0
F(2,3)	Center frequency at attenuation level of 3.0
F(2,4)	Bandwidth at attenuation level of 3.0
F(3,1)	Frequency in low frequency side at attenuation level of 10.0
F(3,2)	Frequency in high frequency side at attenuation level of 10.0
F(3,3)	Center frequency at attenuation level of 10.0
F(3,4)	Bandwidth at attenuation level of 10.0

When the search ca not be executed, (0,0) is entered. To N, the nos. of attenuation level is entered.

(8) Function which obtains bandwidth analysis for multi attenuation levels MBNDO

MBNDO (start address point, end address point, standard address point, nos, of attenuation level, attenuation level array, array storing analysis result such as bandwidth, analysis channel)

Explanation: The function is the same as MBNDI, however, the search is executed from outside to inside.

Usage: Used when the search is executed from outside to inside.

Example:

```
DIM L(3), F(3,4)
L(1)=1.0
L(2)=3.0
L(3)=10.0
P  =PMAX(0,1200,0)
N  =MBNDO(0,1200,P,3,L(1),F(1,1),0)
```

In this case, the array F is stored similarly at MBNDI.

4.4.8 Ripple Analysis Function-1

(1) Function which obtains the difference between the max. value and min. value RPL1

RPL1 (start address point, end address point, gradient coefficient for horizontal axis, gradient coefficient for vertical axis, analysis channel)

Explanation: Calculates the difference between the max. value and min. value by detecting the highest or lowest value between the specified address points in accordance with the gradient coefficient for horizontal or vertical axis.

Usage: Analyzes the ripple to be measured.

Example: X=RPL1(0,1200,1,0.5,0)

Calculates the difference between the max. value and min. value in the ripple which drops or raise 0.5dB a point.

(2) Function which calculates the difference between the max. value and min. value RPL2

RPL2 (start address point, end address point, gradient coefficient for horizontal axis, gradient coefficient for vertical axis, analysis channel)

Explanation: Detects the max . value or min. by detecting the max. or min value between specified address points according to the gradient coefficient for horizontal or vertical axis. Calculate the max. value in the difference between the closed max. value and min. value.
The max. value is low frequency side to the closed max. and min. value.

Usage: Analyzes the ripple to be measured.

Example: P = PMAX(0,1200,0)
X = RPL2(0,P,1,0.5,0)

Calculates the difference between the max. value and min. value closed to the left to the peak point in the ripple which drops or raise 0.5dB a point.

(3) Function which calculates the max. for the value adding the difference between the max. value and min. value RPL3

RPL3 (start address point, end address point, gradient coefficient for horizontal axis, gradient coefficient for vertical axis, analysis channel)

Explanation: Detect the max. and min. value between the specified points in accordance with the gradient coefficient in the vertical and horizontal axis. Calculate the max. value by adding the difference between the max. and min. value or the difference between the min. and max. value.

Usage: Analyzes the ripple to be measured.

Example: X = RPL3(0,1200,1,0.5,0)

Analyzes the ripple which drops or raise 0.5dB a point.

(4) Function which calculates the difference between the max. value and min. value RPL4

RPL4 (start address point, end address point, gradient coefficient for horizontal axis, gradient coefficient for vertical axis, analysis channel)

Explanation: Detect the max . value or min. by detecting the max. or min value between specified address points according to the indent coefficient for horizontal or vertical axis. Calculate the max. value in the difference between the closed max. value and min. value.
The max. value is low frequency side to the closed max. and min. value.
The pair of the max. and min. is conversed to RPL2.

Usage: Analyze the ripple to be measured.

Example: P = PMAX(0,1200,0)
X = RPL4(P,1200,1,0.5,0)

Calculates the difference between the max. value and min. value closed to the left to the peak point in the ripple which drops or raise 0.5dB a point.

(5) Function which obtains the max. value in the highest mark. RPL5

RPL5 (start address point, end address point, gradient coefficient for horizontal axis, gradient coefficient for vertical axis, analysis channel)

Explanation: Detect the max . value between the specified points according to the indent coefficient for horizontal or vertical axis to calculate the max. value.

Usage: Analyze the ripple spurious to be measured.

Example: X = RPL5(P0,P1,1,0.5,0)

Obtain the max. value in the ripple which drops or raise 0.5dB a point.

(6) Function which obtains the min. value in the max. value RPL6

RPL6 (start address point, end address point, gradient coefficient for horizontal axis, gradient coefficient for vertical axis, analysis channel)

Explanation: Detect the max . value between the specified points in accordance with the gradient coefficient for horizontal or vertical axis to obtain the max. value in the min.

Usage: Analyze the ripple spurious to be measured.

Example: X = RPL6(P0,P1,1,0.5,0)

Obtain the max. value in the min. in the ripple which drops or raise 0.5dB a point.

(7) Function which calculates the frequency difference between the min. value and max. value RPLF

RPLF (start address point, end address point, gradient coefficient for horizontal axis, gradient coefficient for vertical axis, analysis channel)

Explanation: Detect the max . value between the specified points in accordance with the gradient coefficient for horizontal or vertical axis to calculate the frequency difference between the first max. value and next min. value.

Usage: Analyze the ripple to be measured.

Example: X = RPLF(P0,P1,1,0.5,0)

Calculate the frequency difference between the max. value and min. value in the ripple which drops or raise 0.5dB a point.

(8) Function which calculates the response difference between the max. value and min. value RPLR

RPLR (start address point, end address point, gradient coefficient for horizontal axis, gradient coefficient for vertical axis, analysis channel)

Explanation: Detect the max . value between the specified points in accordance with the gradient coefficient for horizontal or vertical axis to calculate the response difference between the first max. value and the next min. value.

Usage: Analyzes the ripple to be measured.

Example: X=RPLR(P0,P1,1,0.5,0)

Calculates the response difference between the max. value and min. in the ripple which drops or raise 0.5dB a point.

(9) Function which obtains the response value in the max. value RPLH

RPLH (start address point, end address point, gradient coefficient for horizontal axis, gradient coefficient for vertical axis, analysis channel)

Explanation: Detect the max . value between the specified points in accordance with the gradient coefficient for horizontal or vertical axis to obtain the response value in the first max. value.

Usage: Analyze the ripple to be measured.

Example: X=RPLH(P0,P1,1,0.5,0)

Obtain the max. response value in the ripple which drops or raise 0.5dB a point.

(10) Function which obtains frequency in max. value FRPLH

FRPLH (start address point, end address point, gradient coefficient for horizontal axis, gradient coefficient for vertical axis, analysis channel)

Explanation: Detect the max . value between the specified points in accordance with the gradient coefficient for horizontal or vertical axis to obtain the frequency in the first max. value.

Usage: Analyze the ripple to be measured.

Example: X=FRPLH(P0,P1,1,0.5,0)

Obtain the frequency in max. in the ripple which drops or raise 0.5dB a point.

(11) Function which obtains measurement point in the max. value PRPLH

PRPLH (start address point, end address point, gradient coefficient for horizontal axis, gradient coefficient for vertical axis, analysis channel)

Explanation: Detect the max . value between the specified points in accordance with the gradient coefficient for horizontal or vertical axis to obtain the measurement point in the first max. value.

Usage: Analyze the ripple to be measured.

Example: X=PRPLH(P0,P1,1,0.5,0)

Obtain the max. measurement value in the ripple which drops or raise 0.5dB a point.

(12) Function which obtains response value in min. value RPLL

RPLL (start address point, end address point, gradient coefficient for horizontal axis, gradient coefficient for vertical axis, analysis channel)

Explanation: Detect the min . value between the specified points in accordance with the gradient coefficient for horizontal or vertical axis to obtain the frequency in the first min. value.

Usage: Analyze the ripple to be measured.

Example: X=RPLL(P0,P1,1,0.5,0)

Obtain the response value in min. in the ripple which drops or raise 0.5dB a point.

(13) Function which obtains frequency in the min. value FRPLL

FRPLL (start address point, end address point, gradient coefficient for horizontal axis, gradient coefficient for vertical axis, analysis channel)

Explanation: Detect the min . value between the specified points in accordance with the gradient coefficient for horizontal or vertical axis to obtain the frequency in the first min. value.

Usage: Analyze the ripple to be measured.

Example: X=FRPLL(P0,P1,1,0.5,0)

Obtain the min. frequency in the ripple which drops or raise 0.5dB a point.

(14) Function which obtains measurement point in the min. value PRPLL

PRPLL (start address point, end address point, gradient coefficient for horizontal axis, gradient coefficient for vertical axis, analysis channel)

Explanation: Detect the min . value between the specified points in accordance with the gradient coefficient for horizontal or vertical axis to obtain the measurement point in the first min. value.

Usage: Analyze the ripple to be measured.

Example: X = PRPLL(P0,P1,1,0.5,0)

Obtain the min. measurement point in the ripple which drops or raise 0.5dB a point.

4.4.9 Ripple Analysis Function-2

(1) Function which obtains the number of the max. value NRPLH

NRPLH (start address point, end address point, gradient coefficient for horizontal axis, gradient coefficient for vertical axis, analysis channel)

Explanation: Detect the max . value between the specified points in accordance with the gradient coefficient for horizontal or vertical axis to calculate the number of the max. value by storing the max. value information inside.

Usage: Analyze the ripple to be measured.

Example: NH = NRPLH(0,1200,1,0.5,0)

Obtain the number of the max. value in the ripple which drops or raise 0.5dB a point.

(2) Function which obtain the number of the min. value NRPLL

NRPLL (start address point, end address point, gradient coefficient for horizontal axis, gradient coefficient for vertical axis, analysis channel)

Explanation: Detect the min . value between the specified points in accordance with the gradient coefficient for horizontal or vertical axis to calculate the number of the max. value by storing the min. value information inside.

Usage: Analyze the ripple to be measured.

Example: NL = NRPLL(0,1200,1,0.5,0)

Obtain the number of the min. value in the ripple which drops or raise 0.5dB a point.

(3) Function which obtains measurement point for the max. or min. value PRPLHN, PRPLLN

```
PRPLHN (number specification of ripple, analysis channel)
PRPLLN (number specification of ripple, analysis channel)
```

Explanation: PRPLHN; Calculate the measurement point for the N-th max. value in NRPLH.

PRPLLN; Calculate the measurement point for the N-th min. value in NRPLL.

Example:

```
NH  =NRPLH(0,1200,1,0.5,0)
NL  =NRPLL(0,1200,1,0.5,0)
PH2=PRPLHN(2,0)
PL2=PRPLLN(2,0)
```

Execute the NRPLH, NRPLL to calculate the measurement point for the second max. or min value.

(4) Function which obtains frequency for the max. or min. value FRPLHN, FRPLLN

```
FRPLHN (number specification of ripple, analysis channel)
FRPLLN (number specification of ripple, analysis channel)
```

Explanation: FRPLHN; Obtain the frequency for the N-th max. value in NRPLH.

FRPLLN; Obtain the frequency for the N-th min. value in NRPLL.

Usage: Analyze the ripple to be measured.

Example:

```
NH  =NRPLH(0,1200,1,0.5,0)
NL  =NRPLL(0,1200,1,0.5,0)
FH2=FRPLHN(2,0)
FL2=FRPLLN(2,0)
```

Execute the NRPLH, NRPLL to obtain the frequency for the second max. or min value.

(5) Function which obtains response value for the max. or min. value VRPLHN, VRPLLN

VRPLHN (number specification of ripple, analysis channel) VRPLLN (number specification of ripple, analysis channel)

Explanation: VRPLHN; Obtain the response value for the N-th max. value in NRPLH.
VRPLLN; Obtain the response value for the N-th min. value in NRPLL.

Usage: Analyze the ripple to be measured.

Example:

```
NH  =NRPLH(0,1200,1,0.5,0)
NL  =NRPLL(0,1200,1,0.5,0)
XH2=VRPLHN(2,0)
XL2=VRPLLN(2,0)
```

Execute the NRPLH, NRPLL to obtain the response value for the second max. or min value.

(6) Function which batches process of calculating measurement point for the max. or min. value PRPLHM, PRPLLM

PRPLHM(integer array, analysis channel) PRPLLM (integer array, analysis channel)

Explanation: PRPLHM; Calculate the measurement point in the max. value in NRPLH.
PRPLLM; Calculate the measurement point in the min. value in NRPLL.

Usage: Analyzes the ripple to be measured.

Example:

```
INTEGER PH(600),PL(600)
NH  =NRPLH(0,1200,1,0.5,0)
NL  =NRPLL(0,1200,1,0.5,0)
NH  =PRPLHM(PH(1),0)
NL  =PRPLLM(PL(1),0)
```

Execute the NRPLH, NRPLL to enter the measurement point in the max. and min value in the array.

(7) Function which batches process of obtaining frequency for the max. or min. value
FRPLHM, FRPLLM

```
FRPLHM(real array, analysis channel)
FRPLLM (real array, analysis channel)
```

Explanation: FRPLHM; Obtain the frequency in the max. value in NRPLH.
FRPLLM; Obtain the frequency in the min. value in NRPLL.

Usage: Analyze the ripple to be measured.

Example:

```
DIM FH(600),FL(600)
NH  =NRPLH(0,1200,1,0.5,0)
NL  =NRPLL(0,1200,1,0.5,0)
NH  =FRPLHM(FH(1),0)
NL  =FRPLLM(FL(1),0)
```

Execute the NRPLH, NRPLL to enter the frequency in the max. and min value in the array.

(8) Function which batches process of obtaining response value for the max. or min. value
VRPLHM, VRPLLM

```
VRPLHM(real array, analysis channel)
VRPLLM (real array, analysis channel)
```

Explanation: VRPLHM; Obtain the response value in the max. value in NRPLH.
VRPLLM; Obtain the response value in the min. value in NRPLL.

Usage: Analyze the ripple to be measured.

Example:

```
DIM XH(600),XL(600)
NH  =NRPLH(0,1200,1,0.5,0)
NL  =NRPLL(0,1200,1,0.5,0)
NH  =VRPLHM(XH(1),0)
NL  =VRPLLM(XL(1),0)
```

Execute the NRPLH, NRPLL to enter the response value in the max. and min value in the array.

4.4.10 Direct Search

(1) Function which obtains address point corresponding to specified response DIRECT

DIRECT (start address point, end address point, response value, analysis channel)

Explanation: Search the specified response value between specified address points to set the corresponded address point. The search direction is from low frequency to high frequency.

Example:
```
P=DIRECT(0,1200,-10.0,0)
```

Search the data position of -10dB.

(2) Function which calculates measurement point corresponding to specified response DIRECTL, DIRECTH

DIRECTL (start address point, end address point, response value, analysis channel) DIRECTH (start address point, end address point, response value, analysis channel)

Explanation: Search the specified response value between specified address points to set the corresponded measurement point. The search direction of DIRECTL is from low frequency to high frequency and of DIRECTH is from high frequency to low frequency. when a response corresponds to the specified response, the measurement point is returned. When it not corresponded, the measurement point more than the specified response value is returned. Therefore, The continuous search is easy to execute.

Example:
```
P0=DIRECTL(0,1200,-3.0,0)
P1=DIRECTH(0,1200,-3.0,0)
F  =DFREQ(P0,P1,0)
```

Search from outside to calculate the bandwidth.

(3) Function which obtains frequency corresponding to specified response CDIRECT

CDIRECT (start frequency, end frequency, response value,analysis channel)

Explanation: Search the specified response value between specified responses to calculate the corresponded address point. The search direction is from low frequency to high frequency.

Example:
```
F=CDIRECT(F0,F1,-10.0,0)
```

Obtain the data position of -10dB.

(4) Function which obtains frequency corresponding to specified response
CDIRECTL, CDIRECTH

CDIRECTL (start frequency, end frequency, response value,analysis channel) CDIRECTH (start frequency, end frequency, response value,analysis channel)

Explanation: Search the specified response value between specified address points to obtain the corresponded frequency. The search direction of CDIRECTL is from low frequency to high frequency and of CDIRECTH is from high frequency to low frequency.

Example:

```
F0 = CDIRECTL(F0,F1,-3.0,0)
F1 = CDIRECTH(F0,F1,-3.0,0)
F  = F1-F0
```

Search from outside to calculate the bandwidth.

(5) Function which obtains address point width in specified response DDIRECT

DDIRECT (start address point, end address point, response value,analysis channel)

Explanation: Search the specified response value between the specified address points to the high frequency side to obtain the address point width from two detected measured points.

(6) Function which obtains bandwidth in specified response CDDIRECT

CDDIRECT (start address point, end address point, response value,analysis channel)

Explanation: Search the specified response value between the specified frequencies to the high frequency side to calculate the bandwidth from two detected measured points.

(7) Function which obtains frequency in zero phase ZEROPHS

ZEROPHS (start frequency, end frequency, response value,analysis channel)

Explanation: Detect the phase zero between the specified address points to obtain the frequency.

4.4.11 Data Transfer

(1) Function which reads data of specified analysis channel to array TRANSR

TRANSR (start address point, end address point, real array, analysis channel)

Explanation: Read the measured data in the specified analysis channel by specifying the address point to the BASIC array to return the number of data.

Usage: Used when the measured data is secondary processed.

Example:
```
DIM X(1201)
N =TRANSR(0,1200,X(1),0)
```

(2) Function which writes description of array to specified analysis channel TRANSW

TRANSW (start address point, end address point, real array, analysis channel)

Explanation: Write the description of the BASIC array to the specified analysis channel.

Usage: Used when the measured data is secondary processed.

Example:
```
DIM X(1201)
N =TRANSW(0,1200,X(1),0)
```

5. PARALLEL I/O PORT

5.1 Parallel I/O Port

The parallel I/O port is the input/output port to communicate with the handler or peripherals.
The parallel I/O connector on the back panel is used for communication. Figure 5-1 shows the internal pin assignment and signals of the connector. These I/O port is controlled with ENTER and OUTPUT commands.

- Input/output port

  There are two output ports and two input/output ports, as follows:

·Port only for output:	A port: 8-bit width
	B port: 8-bit width
·Input/output port:	C port: 4-bit width
	D port: 4-bit width

- Port C status output, port D status output

  Shows the settings of the input of the input/output ports C and D. It is low when C or D port is set to input, it is high when it is set to output

- Write strobe output for output port

  By generating a negative pulse on the write strobe output, it shows which output port is used for data output.
  Figure 5-1 shows the timing chart of the write strobe output and data output.

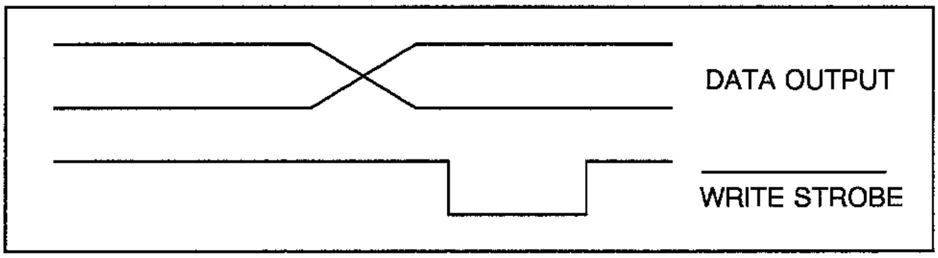


Figure 5-1 Timing Chart of WRITE STROBE

- INPUT 1 input

  By entering a negative pulse on the INPUT 1, the outputs 1 and 2 are set to LOW. The pulse width of the input signal to be entered in the INPUT 1 should be more than 1.

- OUTPUT 1 and 2

  These two signal lines are the latch output terminals set to LOW when a negative pulse is entered on the INPUT 1. It can be set to LOW or HIGH with the BASIC command (OUTPUT).

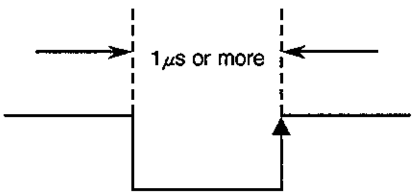
- PASS/FAIL output
  Generates LOW when the result of the limit test is PASS and HIGH when the result is FAIL. This function is available only when the limit test function is ON.

- Write strobe output for PASS/FAIL output
  When the limit test result is output to the PASS/FAIL output line, generates a negative pulse.

- SWEEP END
  When the analyzer finishes the sweeping, generates a negative pulse with a width of 10μs.

- +5V output
  +5 V output is provided for the external device. The maximum current to be supplied is 100mA. This line has a fuse which will be blown when overcurrent flows for circuit protection. The fuse needs to be replaced.

- EXT TRIG input
  By entering a negative pulse on this line, it is possible to trigger the sweeping measurement. The pulse width should be at least 1. The sweeping starts at the rising edge of the pulse. When this signal line is used, the trigger source should be set externally.



5.1.1 Connector Internal Pin Assigned and Signal Standard

Pin No.	Signal name	Function
1	GND	Ground
2	INPUT 1	Negative logic pulse input of TTL level (width: 1μs or more)
3	OUTPUT 1	Negative logic latch output of TTL level
4	OUTPUT 2	Negative logic latch output of TTL level
5	Output port A0	Negative logic latch output of TTL level
6	Output port A1	Negative logic latch output of TTL level
7	Output port A2	Negative logic latch output of TTL level
8	Output port A3	Negative logic latch output of TTL level
9	Output port A4	Negative logic latch output of TTL level
10	Output port A5	Negative logic latch output of TTL level
11	Output port A6	Negative logic latch output of TTL level
12	Output port A7	Negative logic latch output of TTL level
13	Output port B0	Negative logic latch output of TTL level
14	Output port B1	Negative logic latch output of TTL level
15	Output port B2	Negative logic latch output of TTL level
16	Output port B3	Negative logic latch output of TTL level
17	Output port B4	Negative logic latch output of TTL level
18	EXT TRIG	EXTERNAL TRIGGER input (width: 1μs or more),negative logic
19	Output port B5	Negative logic latch output of TTL level
20	Output port B6	Negative logic latch output of TTL level
21	Output port B7	Negative logic latch output of TTL level
22	Input/output port C0	Negative logic state input/latch output of TTL level
23	Input/output port C1	Negative logic state input/latch output of TTL level
24	Input/output port C2	Negative logic state input/latch output of TTL level
25	Input/output port C3	Negative logic state input/latch output of TTL level
26	Input/output port D0	Negative logic state input/latch output of TTL level
27	Input/output port D1	Negative logic state input/latch output of TTL level
28	Input/output port D2	Negative logic state input/latch output of TTL level
29	Input/output port D3	Negative logic state input/latch output of TTL level
30	Port C status	TTL level, Input mode: LOW, Output mode: HIGH
31	Port D status	TTL level, Input mode: LOW, Output mode: HIGH
32	Write strobe signal	TTL level, Negative logic, Pulse output
33	PASS/FAIL signal	TTL level, PASS: LOW, FAIL: HIGH, latch output
34	SWEEP END signal	TTL level, Negative logic, Pulse output (width: 10μs or more)
35	+5V	+5V 100mA MAX
36	Write strobe signal (PASS/FAIL)	TTL level, Negative logic, Pulse output

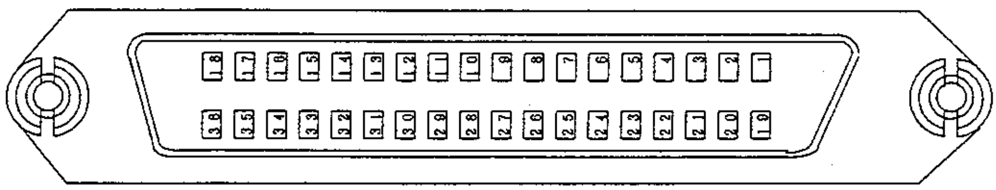
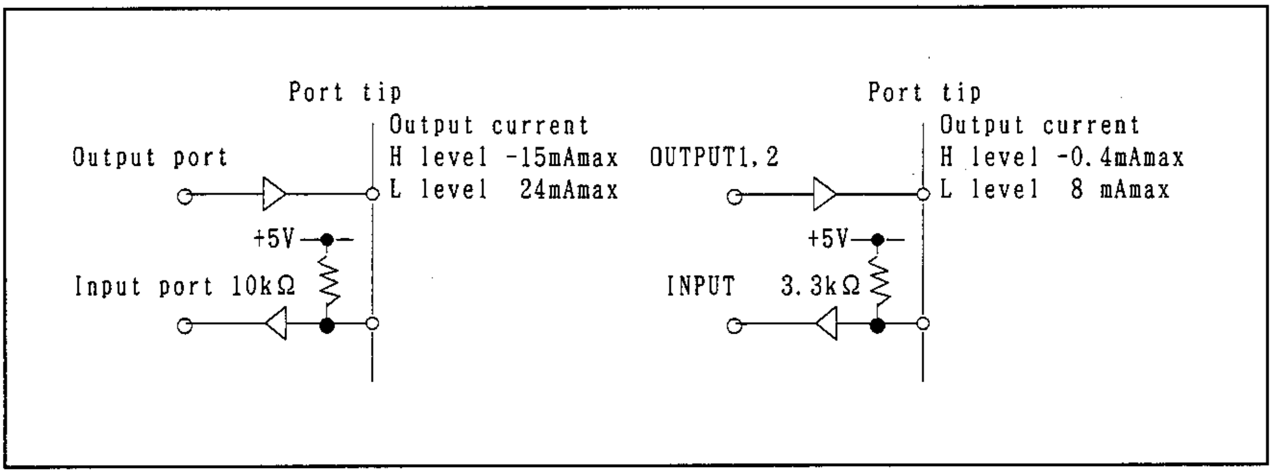


Figure 5-2 36-pin Connector Internal Pin Assignment and Signal



5.1.2 Mode Setting of Port

Command	Output port	Input port
OUTPUT 36 ;16	A, B, C, D	
OUTPUT 36 ;17	A, B, D	C
OUTPUT 36 ;18	A, B, C	D
OUTPUT 36 ;19	A, B	CD

To use a parallel I/O port, first set the mode setting of port. The combination of the setting command and the input port is referred the above table.

Example

```
10 OUTPUT  36;19
20 OUTPUT  33;255
30 ENTER   37;A
      ⋮
```

Set the output port for port A and port B, and the input port for port CD.

5.1.3 Each Port Operation Method

Describes the operation method by built-in BASIC.
OUTPUT statement (for output) and ENTER statement (for input) are used for data input/output.In the relationship between each port and BASICS command, the addresses used in each statement (OUTPUT and ENTER statements) is distinguished.

(1) BASIC format
OUTPUT (address); (data)
ENTER (address); [variable]
(An Input data becomes numeric value of variable name.)

(2) Address and data area

Address	Port to be used
33	Port A (Output only: OUTPUT statement only)
34	Port B (Output only: OUTPUT statement only)
35	Port C (Input/output: ENTER, OUTPUT)
36	Port D (Input/output: ENTER, OUTPUT)
37	Port CD (Input/output: ENTER, OUTPUT)

- OUTPUT 33, 34, 37
  OUTPUT ×× ; 0 to 255 (8bit)

- OUTPUT 35, 36
  OUTPUT ×× ; 0 to 15 (4bit)
  Note: The OUTPUT 35 concerns with the Set/Reset of Flip Flop.

- ENTER 35, 36
  ENTER ×× ; numeric variable (4bit) (Data from 0 to 15 are assigned.)

- ENTER 37
  ENTER 37 ; numeric variable (8bit) (Data from 0 to 255 are assigned.)

5.1.4 INPUT 1, OUTPUT 1, and OUTPUT 2 Terminals

By combining with the signal lines of INPUT1, OUTPUT 1, and OUTPUT 2, convenient functions are provided to easily control external devices.
The functions are; function which sets two latch outputs to LOW by pulse input to INPUT 1, and function which detects the state of variable OUTPUT 1 by INPUT 1. Also, the state of OUTPUTs 1 and 2 can be controlled by OUTPUT command.

(1) Setting OUTPUT 1 and OUTPUT 2, and Reset
The following four types are provided for set/reset as follows:

- Setting OUTPUT 1: OUTPUT 35 ; 16
- Setting OUTPUT 2: OUTPUT 35 ; 48
- Resetting OUTPUT 1: OUTPUT 35 ; 80
- Resetting OUTPUT 2: OUTPUT 35 ; 112

(2) INPUT 1 (external input)
The state of variable OUTPUT 1 by INPUT 1 can be observed by ENTER statement.

ENTER 34; (numeric variable)

If the numeric variable is set to 1, OUTPUT 1 will become ON (Low level: negative logic), if 0, the result will become OFF (High level).

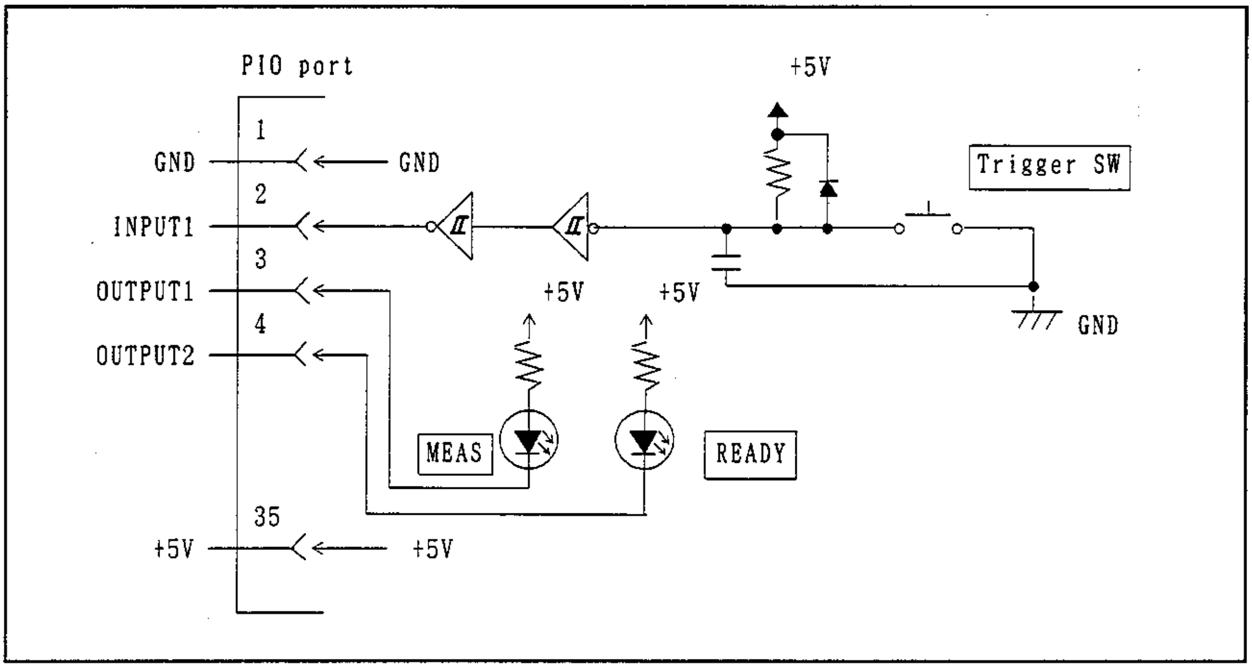
Example
```
10 OUTPUT 36 ; 16
20 ENTER 34 ; A
30 IF A< > 1 THEN GOTO 20
40 OUTPUT 33 ; 1
      ⋮
```

By observing the state of OUTPUT 1, if OUTPUT 1 is set to ON, then 1 is output to the port A.

① Examples of INPUT 1, OUTPUT 1, and OUTPUT2

When program is executed by trigger switch:

- Circuit example



- Program example

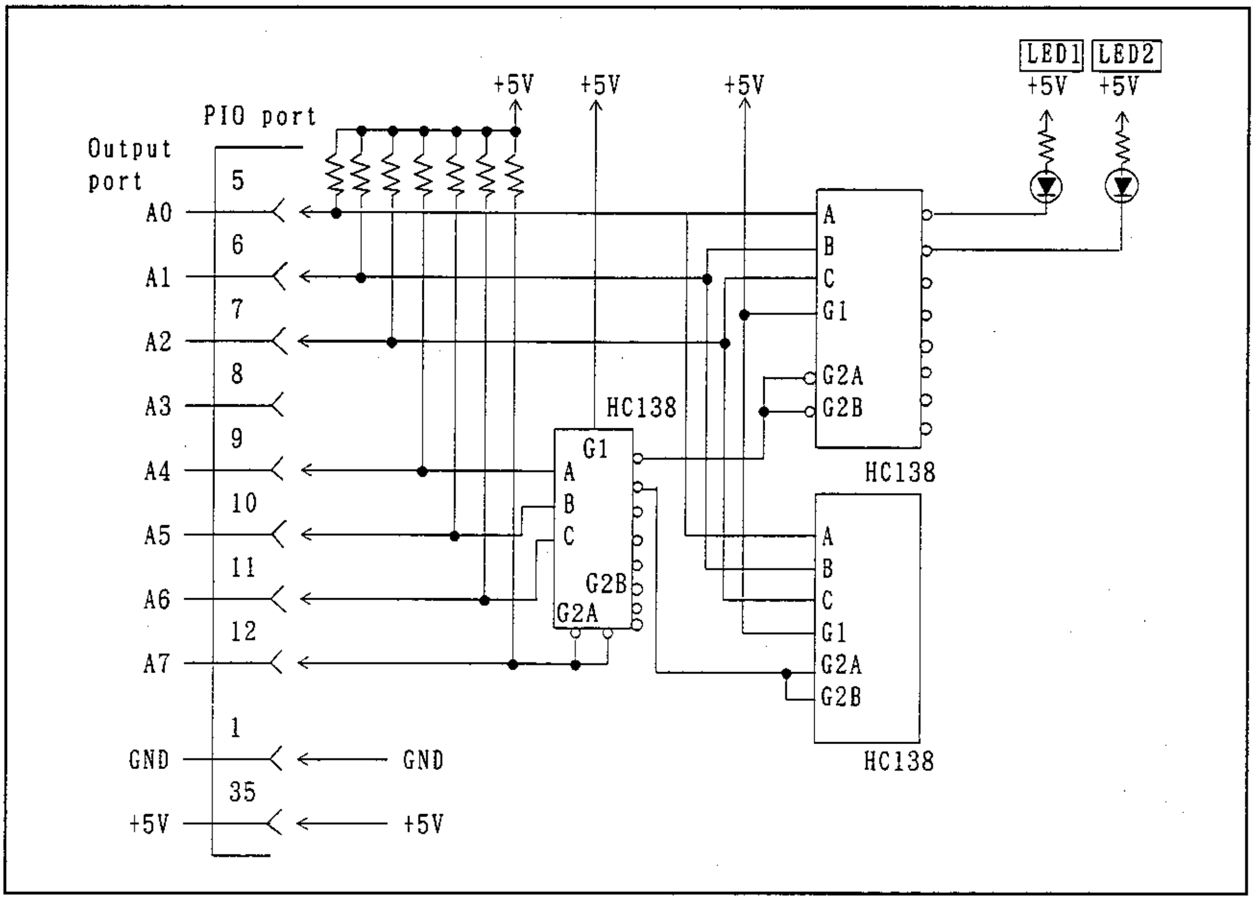
Waiting time for measurement: Represents READY .
During measurement operation: Represents MEAS .

10	OUTPUT 35 ; 80	
20	OUTPUT 35 ; 112	READY MEAS turns OFF.
:		Network analyzer initial setup
:		
100	OUTPUT 35 ; 48	READY turns ON.
110	ENTER 34 ; A	
120	IF A< >1 THEN GOTO 110	Recognition of Trigger SW
130	OUTPUT 35 ; 112	READY turns OFF.
:		
:		Measurement routine
500	OUTPUT 35 ; 80	MEAS turns OFF.
510	GOTO 100	When repeating the measurement
520	STOP	

② Usage example of output ports A and B .

When LED is used for selecting devices (when port A is used):

- Circuit example



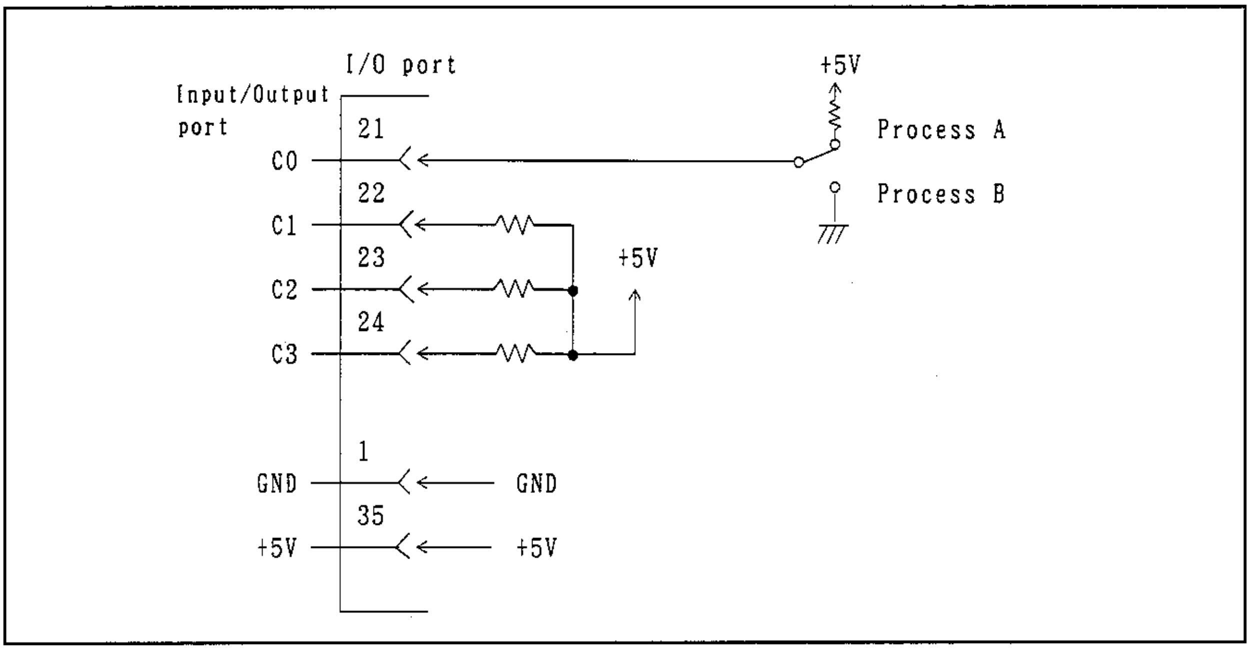
- Program example

10 OUTPUT 36 ; 16 Defines ports A, B, C, and D as output port.
20 OUTPUT 33 ; 0 Initializes LED.
30
⋮ Measurement and judgment
⋮ (measurement variable: A
⋮ judgment area: JED0 to JED1, JED1 to JED2···)
500 IF A> =JED0 AND A<JED1 THEN OUTPUT 33 ; 0_xFF
(when JED0 to JED1, lights up LED 1.)
510 IF A> =JED1 AND A<JED2 THEN OUTPUT 33 ; 0_xFF
(when JED1 to JED2, lights up LED 2.)
⋮
800 GOTO 30
810 STOP

③ Usage example of input ports C and D

Example to change routine whether bit 0 of I/O port C is 0 or 1

- Circuit example



- Program example (Check the port C by pressing [Trigger SW] in step ①.)

```
10    OUTPUT 36 ; 19              Defines ports A and B as output port.
20    OUTPUT 35 ; 80              Defines ports C and D as input port.
30    OUTPUT 35 ; 112
 :                                Network analyzer initial setup
100   *TRIG
110   ENTER 34 ; A
120   IF A< >1 THEN GOTO *TRIG
130   ENTER 35 ; B                Obtains value of port C.
140   IF B=1 THEN GOTO *ROUT_B
150   *ROUT_A
 :                                Process A
490   GOTO *TRIG
500   *ROUT_B
 :                                Process B
900   GOTO *TRIG
910   STOP
```

MEMO

6. ERROR MESSAGES

6.1 How to Check Error Message Line Number

When the PRINT ERRM$(0) statement is executed, the line number of suspended position and the last error message will be displayed.

6.2 How to Check Program Current Position

The symbol "@" is a system variable, which stores the the line number of the program being executed. The current line number, program position and suspended position of the program can be checked by using the @ system variable.

Example: PRINT @···Displays the paused position of the program.

6.3 Error Message List

Note 1: The error messages are described in the following table in the order of error class (error number).
(After the table, correspondence table in alphabetical order is also provided.)
Character strings are explained as XXX.
Numerics are described as YYY.

Note 2: Error class 1: Data input
2: Data calculation processing
3: Built-in function
4: BASIC syntax
5: Others

(1 of 5)

Error class (Error number)	Error message	Description
1(22)	xxx1(xxx2) error	xxx1 command is not available for xxx2 file.
1(23)	xxx1(xxx2, xxx3) error	xxx1 command is not available for xxx2 file and xxx3 file.
1(64)	"xxx" file cannot be opened	The file could not be opened or dose not exist.
1(65)	xxx: "xxx" file was opened with xxx mode.	The file was accessed with different mode from it was opened.
1(66)	cannot read data from "xxx" file.	The specified character number could not be read from xxx file.

(2 of 5)

Error class (Error number)	Error message	Description
1(67)	cannot write data into "xxx" file.	Data can not be written to xxx file.
1(69)	"xxx" file is already opened with another PATH.	The file already opened was tried to open again.
1(72)	file is NOT open.	File is not registered in the specified descriptor. (File has not been opened).
1(74)	end of "xxx" file	Data was read to EOF(End Of File).
1(75)	"xxx" file is already exist.	The existing file was tried to open with OUTPUT mode.
1(77)	Already 8 files are opened.	More than 8 files were tried to be opened.
1(79)	CANNOT assigned into this token	Cannot be assigned into the character variable.
1(95)	GPIB SYNTAX ERROR	The GPIB command is incorrect.
1(96)	Abort	The GPIB control statement was aborted in the execution, or an error occurred on the GPIB bus.
1(98)	Not controller	The command which can be used in controller mode was used in addressable mode.
1(99)	Not Talker/Listener	The command which can be used in addressable mode was used in controller mode.
2(1)	0 divide	0 division (n/0) was executed.
2(10)	xxx: CANNOT convert into string	Conversion into character string is not available.
2(32)	string length is too long	Declaration of character string variable exceeded the maximum value (128).

(3 of 5)

Error class (Error number)	Error message	Description
2(33)	Array's range error	Subscript of the array variable is out of declaration range.
2(41)	yyy: UNIT addr error in xxx	GPIB address is incorrectly specified.
2(43)	yyy is invalid value in xxx	yyy is invalid in xxx instruction.
2(48)	CANNOT move line.	The last line was specified exceeding 65535 in the REN command.
2(51)	Overflow value	The value of operation exceeded the allowable range
2(60)	yyy: Undefined Control Register	The register number of CONTROL instruction is not correct.
2(63)	Unmatched DATA's values and READ variable	Data read in READ statement does not exist.
2(85)	file format error	A terminator that should be within 256 characters is not.
3(11)	xxx function error	An parameter error was detected the built-in function.
3(94)	xxx function error. message	An error was detected the built-in function.
4(2)	xxx: invalid type in xxx	xxx contains an invalid type.
4(3)	NO operand in xxx	Operation format for xxx was set incorrectly.
4(5)	Program is NOT exist	Executed the program not exist.
4(6)	xxx: Syntax error	The syntax is not correct.
4(7)	Undefined ON condition	ON state was incorrectly defined.
4(9)	xxx: Invalid TARGET operand in xxx	The operand syntax in xxx contains an error.
4(12)	Unbalanced NEXT statement	NEXT statement does not exist even the existence of FOR statement.

(4 of 5)

Error class (Error number)	Error message	Description
4(13)	FOR's nest is abnormal.	Nesting to FOR statement could not execute properly.
4(14)	FOR variable does NOT exist.	The counter variable of FOR statement does not exit.
4(15)	FOR <init value> does NOT exist.	The initial value of FOR statement does not exist.
4(16)	Unbalanced FOR variable in NEXT	Relation between For statement and NEXT statement is not normal.
4(17)	Unbalanced BREAK	BREAK statement does not exist between FOR statement and NEXT statement.
4(18)	Uninstalled type (xxx)	Variable was incorrectly formatted.
4(19)	Label xxx is already exists.	Label for xxx is already exist.
4(20)	Unbalanced xxx	Statement construction is not balanced.
4(21)	Not available ASCII char(yyy)	ASCII code is not available.
4(24)	xxx: invalid first type in xxx	The first part of command syntax is incorrect.
4(25)	xxx: invalid second type in xxx	The second part of command syntax is incorrect.
4(26)	xxx: invalid source type in xxx	The type of source side is invalid for assignment of expression.
4(27)	xxx: invalid target type in xxx	The type of target variable is invalid for assignment.
4(29)	Invalid dimension parameter	Parameter of an array variable is not correct.
4(31)	string declaration error	[ ] was used in numeric variable.
4(34)	Unbalanced line No.	Specified line does not exist.
4(37)	Undefined label	Specified label does not exist.

(5 of 5)

Error class (Error number)	Error message	Description
4(38)	label not found	Specified label does not exist.
4(39)	Unknown line No.	Specifying line does not exist.
4(40)	expression format error	Expression is formatted incorrectly.
4(43)	yyy is invalid value in xxx	yyy is invalid in xxx instruction.
4(44)	Unbalanced xxx block	xxx block is not matched (FOR statement, IF statement, etc.).
4(45)	Not found THEN in xxx	THEN was not found after IF statement.
4(47)	Not found line No. yyy	Line No. yyy is not found.
4(49)	Substring error	Substring is incorrectly specified.
4(50)	parameter error	Parameter is not set correctly.
4(52)	Unmatched IMAGE-spec in USING	Specification of IMAGE in USING is unmatched
4(54)	yyy error(s) appeared.	The label line number is not correct.
4(55)	Program CANNOT be continued.	The terminated program was tried to restart again.
4(56)	Line No.yyy is out of range.	Specification of line number exceeded the program range.
4(68)	cannot specify "USING"	USING can not be specified by the specified file type.
4(70)	Not found DATA statement	DATA statement was not found in the direction of RESTORE.
4(71)	xxx nest overflow	The nesting exceeded the capacity.
4(78)	SELECT nesting overflow	Nesting to SELECT statement exceeded the capacity.
4(93)	Program cannot changed	Program change was tried in the execution of program.

Correspondence table in alphabetical order

(1 of 3)

Error message	Error class (Error number)
Abort	1(96)
Already 8 files are opened.	1(77)
Array's range error	2(33)
CANNOT assigned into this token	1(79)
CANNOT move line.	2(48)
cannot read data from "xxx" file.	1(66)
cannot specify "USING"	4(68)
cannot write data into "xxx" file.	1(67)
end of "xxx" file	1(74)
expression format error	4(40)
file format error	2(85)
file is NOT open.	1(72)
FOR <init value> does NOT exist.	4(15)
FOR variable does NOT exist.	4(14)
FOR's nest is abnormal.	4(13)
GPIB SYNTAX ERROR	1(95)
Invalid dimension parameter	4(29)
label not found	4(38)
Label xxx is already exists.	4(19)
Line No.yyy is out of range.	4(56)
NO operand in xxx	4(3)
Not available ASCII char(yyy)	4(21)
Not controller	1(98)
Not found DATA statement	4(70)
Not found line No. yyy	4(47)
Not found THEN in xxx	4(45)
Not Talker/Listener	1(99)
Overflow value	2(51)

(2 of 3)

Error message	Error class (Error number)
parameter error	4(50)
Program CANNOT be continued.	4(55)
Program cannot changed	4(93)
Program is NOT exist	4(5)
SELECT nesting overflow	4(78)
string declaration error	4(31)
string length is too long	2(32)
Substring error	4(49)
Unbalanced BREAK	4(17)
Unbalanced FOR variable in NEXT	4(16)
Unbalanced line No.	4(34)
Unbalanced NEXT statement	4(12)
Unbalanced xxx	4(20)
Unbalanced xxx block	4(44)
Undefined label	4(37)
Undefined ON condition	4(7)
Uninstalled type (xxx)	4(18)
Unknown line No.	4(39)
Unmatched DATA's values and READ variable	2(63)
Unmatched IMAGE-spec in USING	4(52)
xxx function error	3(11)
xxx function error. message	3(94)
xxx nest overflow	4(71)
xxx1(xxx2) error	1(22)
xxx1(xxx2, xxx3) error	1(23)
xxx: CANNOT convert into string	2(10)
xxx: invalid first type in xxx	4(24)
xxx: invalid second type in xxx	4(25)

(3 of 3)

Error message	Error class (Error number)
xxx: invalid source type in xxx	4(26)
xxx: Invalid TARGET operand in xxx	4(9)
xxx: invalid target type in xxx	4(27)
xxx: invalid type in xxx	4(2)
xxx: Syntax error	4(6)
xxx: "xxx" file was opened with xxx mode.	1(65)
"xxx" file cannot be opened	1(64)
"xxx" file is already exist.	1(75)
"xxx" file is already opened with another PATH.	1(69)
yyy error(s) appeared.	4(54)
yyy is invalid value in xxx	2(43), 4(43)
yyy: Undefined Control Register	2(60)
yyy: UNIT addr error in xxx	2(41)
0 divide	2(1)

ALPHABETICAL INDEX

Part 2

TABLE OF CONTENTS

LIST OF TABLES

1. INTRODUCTION

The network analyzer is equipped with a GPIB (General-Purpose Interface Bus) as standard, which complies with IEEE standards 488.1-1987 and 488.2-1987 and can be remotely controlled by means of an external controller. The analyzer also has a built-in control function, enabling easy configuration of small GPIB systems.

The following describes the method of control using the GPIB remote control functions.

1.1 GPIB

The GPIB is a high-performance interface bus used to connect the measuring instruments to the computer.

The operations of the GPIB are defined by IEEE standard 488.1-1987. Since the GPIB has a bus-configured interface, it can specify a device by assigning a specific address to each device. Up to 15 devices can be connected in parallel to a single bus. GPIB devices have one or more of the following functions:

- Talker: The talker is a device which is specified to send data to the bus. Only one active talker can exist on the GPIB bus.
- Listener: The listener is a device which is specified to receive data from the bus. Multiple active listeners can exist on the GPIB bus.
- Controller: The controller is a device which specifies the talker and listener. Only one active controller can operate on the GPIB bus. Controllers which control IFC and REN messages are called "system controllers".

  The GPIB bus can have only one system controller on it. If there are multiple controllers on the bus, the system controller becomes the active controller, while other devices which have a control function operate as addressable devices when the system is started up.

  The TCT (Take Control) interface message is used to set a controller other than the system controller as the active controller. After setting, the system controller will become the non-active controller.

  The controller controls the entire system by sending interface messages or device messages to each measuring instrument. The functions of the messages are:

  - Interface message: Control of the GPIB bus
  - Device message: Control of the measuring instrument

  To use the built-in BASIC, refer to Part 1 of this manual.

1.2 Command Modes

1.2.1 IEEE488.2-1987 Command Mode

In R3764/66 and R3765/67 series, the operation is possible in two command modes.

- IEEE standard 488. 2-1987 command mode
- IEEE standard 488. 1-1987 command mode

R3762/63 series can perform the operation only in IEEE standard 488. 1-1987 command mode.

The 488.2-1987 is defined by extending the following items to 488.1-1987.

- Syntax for programming the measuring instrument
- Communication protocol (procedure) of commands and data
- Common commands *
- Status data structure
- System synchronization protocol

*: The common commands refer to the commands that identically operate on all measuring instruments.

1.2.2 IEEE488.1-1987 Command Mode

Since the command syntax and the communication protocol used in IEEE488.1-1987 command mode are compatible with those of R3762/63 series, smooth transition from IEEE488.1-1987 command mode to R3764/R3766, R3765/67 series is possible. (However, because of changes in product specifications, some operations are performed using different commands.)

1.2.3 Switching of Command Mode

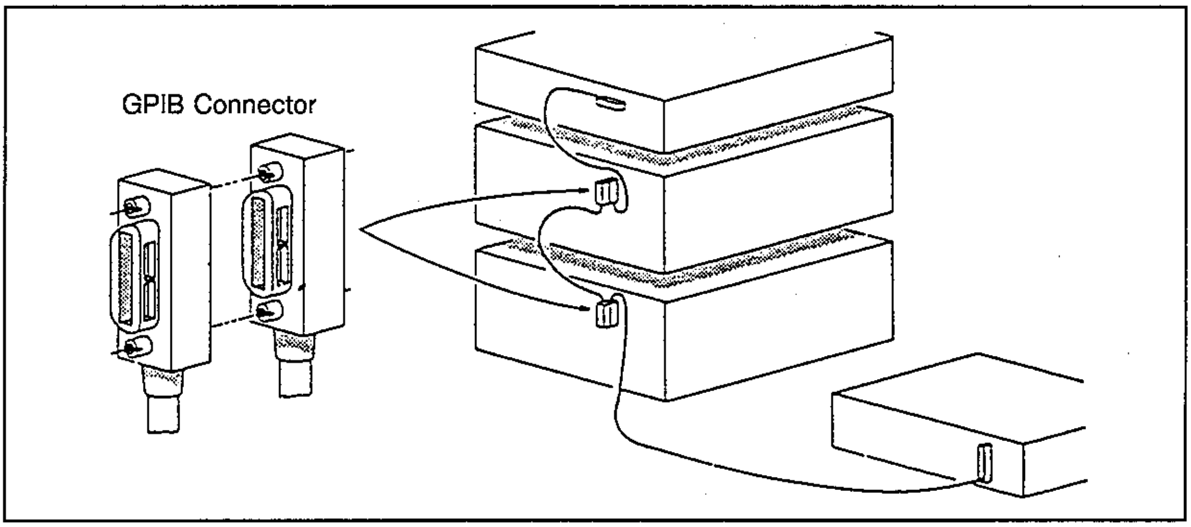
This instrument is set IEEE488.1-1987 command mode after activating (power on).
Execute switching of IEEE488.1-1987 command mode and IEEE488.2-1987 command mode is as follows:

- Send OLDC OFF → It enters IEEE488.2-1987 command mode.
- Send OLDC ON → It enters IEEE488.1-1987 command mode.

1.3 GPIB Setup

(1) Connecting GPIB

The following shows the standard GPIB connector. Secure the GPIB connector with the two screws to prevent it from coming loose during use.



The following precautions should be observed when using the GPIB interface:

- The total GPIB cable length in a single bus system should not exceed n x 2 meters, where n = the number of devices to be connected, including the GPIB controller. In no case should the cable length exceed 20 meters.
- Up to 15 devices can be connected to a single bus system.
- There are no restrictions concerning the method of connection between cables. However, no more than three GPIB connectors should be connected to a single device, since the use of excessive force could damage the connector mounting.

For example, the total cable length in a system with five devices should be 10 meters or less (2 meters x 5 devices = 10 meters).The total cable length can be distributed freely within the range of the maximum allowed cable length. However, if more than ten devices are to be connected, some of them should be connected using cables of less than 2 meters so that the total cable length does not exceed 20 meters.

(2) Setting GPIB address

The GPIB address is set using the keys on the front panel. The key operation depends on the model (R3764/66, R3765/67). For details, refer to the pertinent operation manual.

MEMO

2. GPIB BUS FUNCTIONS

2.1 GPIB Interface Functions

Code	Description
SH1	With source handshake function
AH1	With acceptor handshake function
T6	Basic talker function, serial polling function, listener-specified talker cancel function
TE0	Without extended talker function
L4	Basic listener function, talker-specified listener cancel function
LE0	Without extended listener function
SR1	With service request function
RL1	Remote function, local function, local lockout function
PP0	Without parallel polling function
DC1	Device clear function
DT1	Device trigger function
C1	System controller function
C2	IFC transmission, controller in charge function
C3	REN transmission function
C4	SRQ response function
C12	Transmission of interface messages, control transfer function
E1	Using open-collector bus driver

2.2 Controller Functions

R3764/66, R3765/67 has a system controller mode and an addressable mode. The features of each mode are as follows:

	System Controller Mode	Addressable Mode
At startup	Active controller	Non-active controller
IFC	Controllable	Not controllable
REN	Controllable	Not controllable

To be active in the addressable mode, R3764/66, R3765/67 must have received the TCT interface message.

Only one system controller is allowed on the GPIB bus. When a system connected through the GPIB bus is started up, the system controller becomes the active controller. Only one active controller at a time is allowed on the GPIB bus. The controller controls the devices on the bus by sending interface messages and receiving service requests (SQR). Note that the IFC and REN interface messages are sent by the system controller only.

Interface messages are used to send indications of talker and listener, serial poll, device clear, trigger, local, and the other information to the measuring instrument. Service requests are used to receive interruptions from the instrument.

The active controller can transfer control to any non-active controller. After specifying the talker as the device to which control is to be transferred, the active controller sends a TCT interface message to transfer control to the talker. This operation is called "pass control".

When the system controller sends an IFC interface message, control is returned from the active controller to the system controller.

2.3 Responses to Interface Messages

The responses of the analyzer to interface messages are defined by IEEE standards 488.1-1987 and 488.2-1987 and are described in this section.

For information on how to send interface messages to the analyzer, refer to the instruction manual of the controller to be used.

2.3.1 Interface Clear (IFC)

The IFC message is transmitted directly to the analyzer through a signal line. The message allows the analyzer to stop the operation of the GPIB bus. Although all input/output operation is stopped, the input/output buffer is not cleared. Note that the DCL is used to clear the buffer. If the analyzer is specified as an active controller at that time, control of the GPIB bus will be removed from the analyzer and transferred to the system controller.

2.3.2 Remote Enable (REN)

The REN message is transmitted directly to the analyzer through a signal line. If the analyzer is specified as a listener when the message is true, the analyzer is in the remote mode. The analyzer remains in the remote mode until the GTL message is received, or the REN becomes false, or the LOCAL key is pressed.

When the analyzer is in the local mode, it ignores all the received data. When the analyzer is in the remote mode, it ignores all key inputting other than LOCAL key inputting. When the analyzer is in the LOCAL LOCKOUT mode (LLO; see section 2.3.8), it ignores all key inputting.

2.3.3 Serial Polling Enable (SPE)

When the analyzer receives a message from external devices, it is in the serial polling mode. If the analyzer is specified as a talker in this mode, it sends status bytes instead of normal messages. The analyzer remains in the serial polling mode until the SPD (Serial Polling Disable) message or the IFC message is received.

When the analyzer sends an SRQ (Service Request) message to the controller, bit 6 (RQS bit) of the response data is set to 1 (true). When the analyzer has finished sending this message, the RQS bit reverts to 0 (false). The SRQ (Service Request) message is sent directly through a signal line.

2.3.4 Group Execute Trigger (GET)

If the following conditions are satisfied when this message triggers the analyzer, the analyzer will start the measuring operation.

- The trigger source becomes the GPIB bus (TRIG: SOUR BUS).
- The analyzer is in the trigger waiting state (see "5. TRIGGER SYSTEM").

The GET operates in the same manner as the *TRG but differently from TRIG:IMM and TRIG:SIG. The GET, *TRG, TRIG:IMM and TRIG:SIG are stacked in the input buffer and executed in order of reception.

2.3.5 Device Clear (DCL)

When the analyzer receives the DCL message, it performs the following:

- Clearing of the input and output buffers
- Resetting of syntax (?>program<?) analysis, execution control and response data generation
- Cancellation of all commands that prevent the remote command from being executed next
- Cancellation of commands that are paused to wait for other parameters
- Cancellation of *OPC and *OPC?

It does not perform the following:

- Changing of data set or stored in the analyzer
- Interruption of the front panel operation
- Modification or interruption of analyzer operations being executed
- Changing of status bytes other than MAV. (MAV becomes 0 when the output buffer is cleared.)

2.3.6 Selected Device Clear (SDC)

The SDC message operates in the same manner as the DCL message. However, it is executed only when the analyzer is as a listener.In other cases, it is ignored.

2.3.7 Go To Local (GTL)

The GTL message places the analyzer in the local mode. In the local mode, all the operations on the front panel are available.

2.3.8 Local Lockout (LLO)

The LLO message places the analyzer in the local lockout mode. If the analyzer is set to the remote mode in this mode, all the operations on the front panel will be inhibited. (Note that in the normal remote mode, front panel operations can be performed using the LOCAL key.)

The following three methods can be used to set the analyzer to the local mode from the local lockout mode:

- Sending a GTL message to the analyzer
- Setting the REN message to false (In this case, the local lockout mode will be canceled.)
- Switching on the analyzer power again

2.3.9 Take Control (TCT)

If the analyzer receives the TCT message when it is specified as a talker, it becomes the active controller through "pass control". On receiving the IFC message, the analyzer returns to the addressable mode.

2.4 Message Exchange Protocol

The analyzer receives program messages from controllers or other devices through the GPIB bus and generates response data. The program messages include commands, queries (commands used to query response data) and data. The procedure used to exchange these commands, queries and data is explained in this section.

2.4.1 GPIB Buffers

The analyzer is equipped with the following three buffers:

(1) Input buffer
The input buffer is used to store data temporarily for command analysis (1024 bytes).
Either of the following two methods can be used to clear the input buffer:

- Switching on the analyzer power
- Execution of the DCL or the SDC

(2) Output buffer
The output buffer is used to store data which are to be read from the controller (1024 bytes).
Either of the following two methods can be used to clear the output buffer:

- Switching on the analyzer power
- Execution of the DCL or the SDC

(3) Error queue
The error queue is available only for IEEE488.2-1987 command mode. It is used to store up to ten error messages for remote commands. Each time an error occurs during remote command analysis or in execution, an error message is stored in the queue. The SYST:ERR command is used to read out these messages. When a message is read out, it is removed from the queue.
Either of the following two methods can be used to clear the error queue:

- Switching on the analyzer power
- Execution of the *CLS

2.4.2 IEEE488.2-1987 Command Mode

IEEE488.2-1987 command mode performs the sending and receiving of messages in accordance with the message exchange protocol in compliance with IEEE standard 488.2-1987.

The following are the most important events when another controller or device receives messages from the analyzer in this mode:

- Response data are generated when a query is received.
- Data are generated in the order of query execution.

(1) Purser

The purser receives command messages in the order of reception from the input buffer, analyzes the syntax and determines what the received command is to execute.

The purser traces the tree structure of the commands when analyzing the command program. It memorizes which part of the tree structure is to be used to start analysis when analyzing the next command. This information is returned to the head of the structure when the purser is cleared.

Any of the following four methods can be used to clear the purser:

- Switching on the analyzer power
- Reception of the DCL or the SDC
- Reception of ":" following ";"
- Reception of the terminator or the EOI signal

(2) Generating response data

When the purser executes a query, the analyzer generates data in the output buffer in response to it (that is, to output data a query must be sent immediately before the data). The procedure implies that unless the controller reads out the data generated through the query, the data will never be cleared.

Apart from the controller read operation, there are two conditions under which the data are cleared. A query error will occur under the following conditions:

- Unterminated condition: When the controller has read the response data without terminating (LF code of ASCII or END message of GPIB) or sending the query
- Interrupted condition: When the controller has received the next program message before reading the response data

2.4.3 IEEE488.1-1987 Command Mode

In IEEE488.1-1987 command mode, the analyzer uses the same protocol for message exchange as R3762/63. In this mode, the command stored in the input buffer can be analyzed, and no command string longer than the input buffer can be received (such commands are ignored).

When the analyzer is specified as a talker, the analyzer generates response data. It is necessary for the query to specify the items of the response data in advance. Each time the analyzer is specified as a talker, response data are generated and formatted on the output buffer. It is impossible to answer multiple queries simultaneously.

2.4.4 BASIC Mode

The analyzer supports a function enabling the analyzer to program itself or to be programmed by external devices with a built-in BASIC interpreter. When the BASIC interpreter is in operation, the GPIB interface of the analyzer enters a special mode and the interpreter controls the command messages from the external devices and data output from the analyzer.

For information on data input/output, refer to "ENTER and OUTPUT" in Part 1 of this manual. For information on how the BASIC interpreter does not control the GPIB, refer to "CONTROL Command" in Part 1 of this manual.

The analyzer enables the use of a special method whereby the addressable mode controls the built-in BASIC interpreter.

@BASIC statement

Note: The character "@" must be at the beginning of the input message.

There are no restrictions concerning the BASIC statement to be executed using this method. Also, the BASIC statements described here are not confined to commands. That is, statements such as the following can be executed:

- @100 PRINT "Hello World"
- @VAR = 1000

Using this method, it is possible to download the built-in BASIC program from the external controller through the GPIB bus.

The GPIB bus is controlled by the BASIC interpreter when the BASIC interpreter is in operation. Under these conditions, the external controller can execute the statements in the same manner as above. (However, there are some restrictions on BASIC command execution.)

In other words, no character string beginning with "@" can be received through the GPIB bus in the addressable mode. (This restriction does not apply in the system controller mode, and there is no way to avoid it in the addressable mode.)

3. COMMAND SYNTAX

3.1 IEEE488.2-1987 Command Mode

For characters input in IEEE488.2-1987 command mode other than character string data and block data, no distinction is made between upper case and lower case.

3.1.1 Command Syntax

The command program for IEEE488.2-1987 command mode is defined in the following format:

Header	→	Space (space characters)	⇒	data

Note: "⇒" indicates repetition.

(1) Header
The header has a hierarchical structure consisting of multiple mnemonics separated by a colon. A four-character (or three-character) "short form" is provided for each mnemonic consisting of four characters or more. (Mnemonics which are not abbreviated are called "long forms".) It is possible to use any form in any combination.

Any command with a header followed immediately by "?" becomes a query command.

(2) Space (space character)
One space or more is required in this field; otherwise, a syntax error will occur.

(3) Data
When the command requires multiple data, the data should be separated with commas. A space may be inserted before or after the each comma.

For details of data types, refer to "3.1.2 Data Formats".

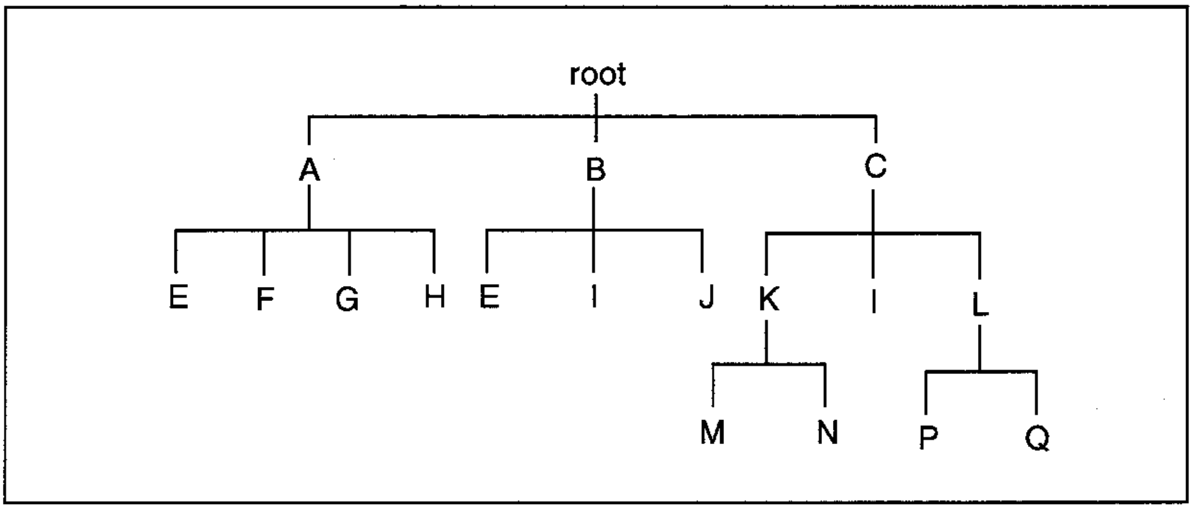
(4) Writing multiple commands
In IEEE488.2-1987 command mode, it is possible to write multiple commands by separating them with semicolons.If commands are written in this way, they should be executed while changing the current path in the hierarchical structure of the header.

(5) Changing the current path

The current path should be changed in accordance with the following rules:

- Switching on: The current path is set to "root".
- Terminator: The current path is set to "root".
- Colon (:): The current path is changed to the layer immediately below in the command tree. If the colon is at the beginning of the command, the current path will be changed to "root".
- Semicolon (;): The current path is not changed.
- Common command: The command can be executed regardless of the current path position. When the *RST command is executed, the current path is set to "root". (See the example below.)

The following header structure is given as an example:



In this example, the current path is changed as follows:

① :A:E;:B:E

Since the colon in the second command changes the current path to "root", commands "A:E" and "B:E" are both valid.

② :A:E<END> B:E

Since <END> (terminator) changes the current path to "root", commands "A:E" and "B:E" are both valid.

③ :A:E;F;G;H

Since the semicolon does not change the current path, ":A:E;F;G;H" results in the four commands "A:E", "A:F", "A:G" and "A:H".

④ :C:I;K:N;M
Since the colon changes the current path, "K:N" is viewed from the ":C:" layer. Therefore, "K:N" results in "C:K:N". At the same time, since "K:N" includes a colon, the current path is changed to ":C:K:" and the last "M" is interpreted as "C:K:M".

⑤ :A:E;*ESR 16
Since the common command is independent of the current path, "*ESR 16" will be executed correctly.

⑥ :A:E;*ESR 16;F;G;H
Since the common command does not change the current path, the third item, "F", will be searched for using the current path ":A:" set by the first item ":A:E". Therefore, "F", "G" and "H" result in "A:F", "A:G" and "A:H", respectively.

The following examples show syntax errors.

① :A:E;B:E
Since "A:E" changes the current path to ":A:", "B:E" will be searched for in the layer of ":A:". However, because the mnemonic "B" is not found, an error will occur.

② :C:K:M;L:P
Since ":C:K:M" changes the current path to ":C:K:", "L:P" will be searched for in the layer of ":C:K:". However, because the mnemonic "L" is not found, an error will occur.

3.1.2 Data Formats

In IEEE488.2-1987 command mode, the analyzer uses the data formats for data input/output shown in this section.

(1) Numeric data

There are three numeric data formats, any of which can be used for numeric data input. (The data are rounded up or down in accordance with the data format to be input.)
Some commands add the units to the data at data inputting. For information on units, refer to (5) below.
The following shows the format of the character data.

- Integer type: NR1 format

  [Symbol] → Number ⇒

- Fixed-point type: NR2 format

  [Symbol] → Number ⇒ . → Number ⇒

- Floating-point type: NR3 format

  [Symbol] → Number ⇒ . → Number ⇒ E/e → Symbol → Number ⇒

Note: "⇒" indicates repetition. Symbols at the beginning may be omitted.

(2) Character data

Alphabetic character → Alphanumeric character / _ ⇒

Note: "⇒" indicates repetition.

(3) Character string data

There are two character string data formats.

$\boxed{"} \rightarrow \boxed{\text{Character / " "}} \Rightarrow \boxed{"}$

$\boxed{'} \rightarrow \boxed{\text{Character / ' '}} \Rightarrow \boxed{'}$

Each format can be used as an ASCII 7-bit code character in the character string data.

Notes: In character string data starting with ["], ["] must be represented by [""]. In character string data starting with ['], ['] must be represented by ['']. "⇒" indicates repetition.

When the response data are character string data, character string data starting with ["] should be output.

(4) Block data

There are two block data formats. Either can be used for inputting into the analyzer.

Fixed-length format: $\boxed{\#} \rightarrow \boxed{\text{Number}} \rightarrow \boxed{\text{Number}} \Rightarrow \boxed{\text{DATA}} \Rightarrow$

Undefined-length format: $\boxed{\#} \rightarrow \boxed{0} \rightarrow \boxed{\text{DATA}} \Rightarrow \boxed{\text{LF\^{}EOI}}$

Note: "⇒" indicates repetition.

In the fixed-length format, the one-digit number following "#" represents the number of digits for the bytes in the data following that number. "0" cannot be used, because it indicates the undefined-length format.

Example: Block data #3128 <data byte>
"3" following "#" represents the number of digits in the character string (128) following "3", while "128" represents the number of bytes in <data byte> following that number.

(5) Units

Units are the suffix following a numeric value. The suffix can be used as a prefix for the unit. The table below lists the suffixes and the units which can be used.

Suffixes		Unit	Commands with which Usable
1E18 1E15 1E12 1E9 1E6 1E3 1E-3 1E-6 1E-9 1E-12 1E-15 1E-18	EX PE T G MA K M * U N P F A	HZ	[SENSe:]BANDwidth[:RESolution] [SOURce:]FREQuency:CENTer [SOURce:]FREQuency:CW [SOURce:]FREQuency:SPAN [SOURce:]FREQuency:STARt [SOURce:]FREQuency:STOP [SOURce:]PSWeep:FREQuency
		DEC	[SENSe:]CORRection:OFFSet:PHASe
		DB	INPut:ATTenuation OUTPut:ATTenuation
		DBM	[SOURce:]POWer[:LEVel][:AMPLitude] [SOURce:]POWer:STARt [SOURce:]POWer:STOP
		M	[SENSe:]CORRection:EDELay:DISTance
		S	[SENSe:]CORRection:EDELay[:TIME] [SENSe:]CORRection:PEXTension:TIME [SOURce:]SWEep:TIME TRIGger[:SEQuence]:DELay
		OHM	CALCulate:TRANsform:IMPedance:CIMPedance INPut:IMPedance

Note: For commands not listed in the table, only the suffix can be used.

*: If HZ or OHM is used as the unit, the command will be executed using the suffix 1E6 (equivalent to MA).

3.2 IEEE488.1-1987 Command Mode

The following shows the program message structure for IEEE488.1-1987 command mode. For IEEE488.1-1987 command mode, a lower-case letter is used as the separator, except in character string data.

3.2.1 Command Syntax

The program for IEEE488.1-1987 command mode is defined in the following format.

[Header] →[[Separato]] ⇒ [Data]

The separator can be a space of zero or more characters, a comma, or a semicolon. The following three data formats can be used:

- Numeric value data format:

  [[Symbol]] → [Number] ⇒ [.] → [Number] ⇒ [EXP/E/e] → [Symbol] → [Number] ⇒[[Unit]]

- Binary data format:

  [ON | OFF | 1 | 0]

- Character string data format:

  [Character] ⇒ [LF^EOI]

Note: "⇒" indicates repetition.

The units below can be used for numeric value data:

GHZ	MHZ	KHZ	HZ
DEG			
DP	DM	DB	
METER	CM		
SEC	MSEC	USEC	NSEC
VOLT	MV	UV	NV
MOHM	KOHM	OHM	
UNIT			
DIV			
PER			

In character string data, the characters from the character immediately after the header to the last character of the input data are regarded as a character string. If "?" is added immediately after the header, the command will become a query command.

4. STATUS BYTES

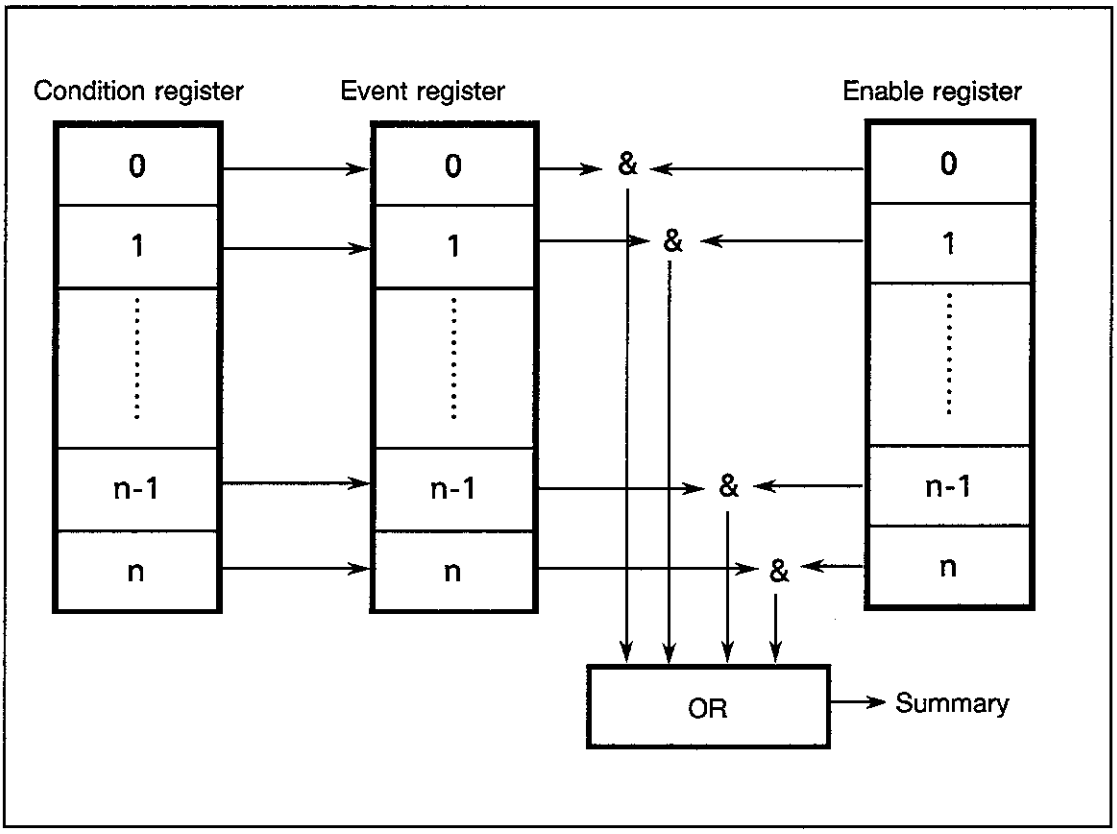
The analyzer has a hierarchical status register structure in compliance with IEEE standard 488.2-1987, which is used to send various device status information to the controller. This chapter explains the operational models of the status byte and event assignments.

Note: The status structure differs from that of R3762/63, irrespective of the command mode.

4.1 Status Register

4.1.1 Status Register Structure

The analyzer employs the status register model defined by IEEE standard 488.2-1987 and consists of a condition register, an event register and an enable register.



(1) Condition register
The condition register continuously monitors the status of devices, that is, retains the latest status of devices. No data can be written into this register.

(2) Event register
The event register latches and retains the status information from the condition register. (In some cases, it retains status changes.)
Once the register is set, the condition is maintained until a query command reads out the information or the register is reset by means of the *CLS command. No data can be written into this register.

(3) Enable register
The enable register specifies which bit in the event register is to be used as the valid status to generate a summary. The enable register is ANDed with the event register. The OR of the result of the AND operation is generated as a summary. The summary is written into the following status registers. Any data can be written into these registers.

4.1.2 Status Register Types

The following eight types of status register are used in the analyzer:

(1) Status byte register; See Section 4.2.

(2) Standard event register; See Section 4.3.

(3) Standard operation status register; See Section 4.4.

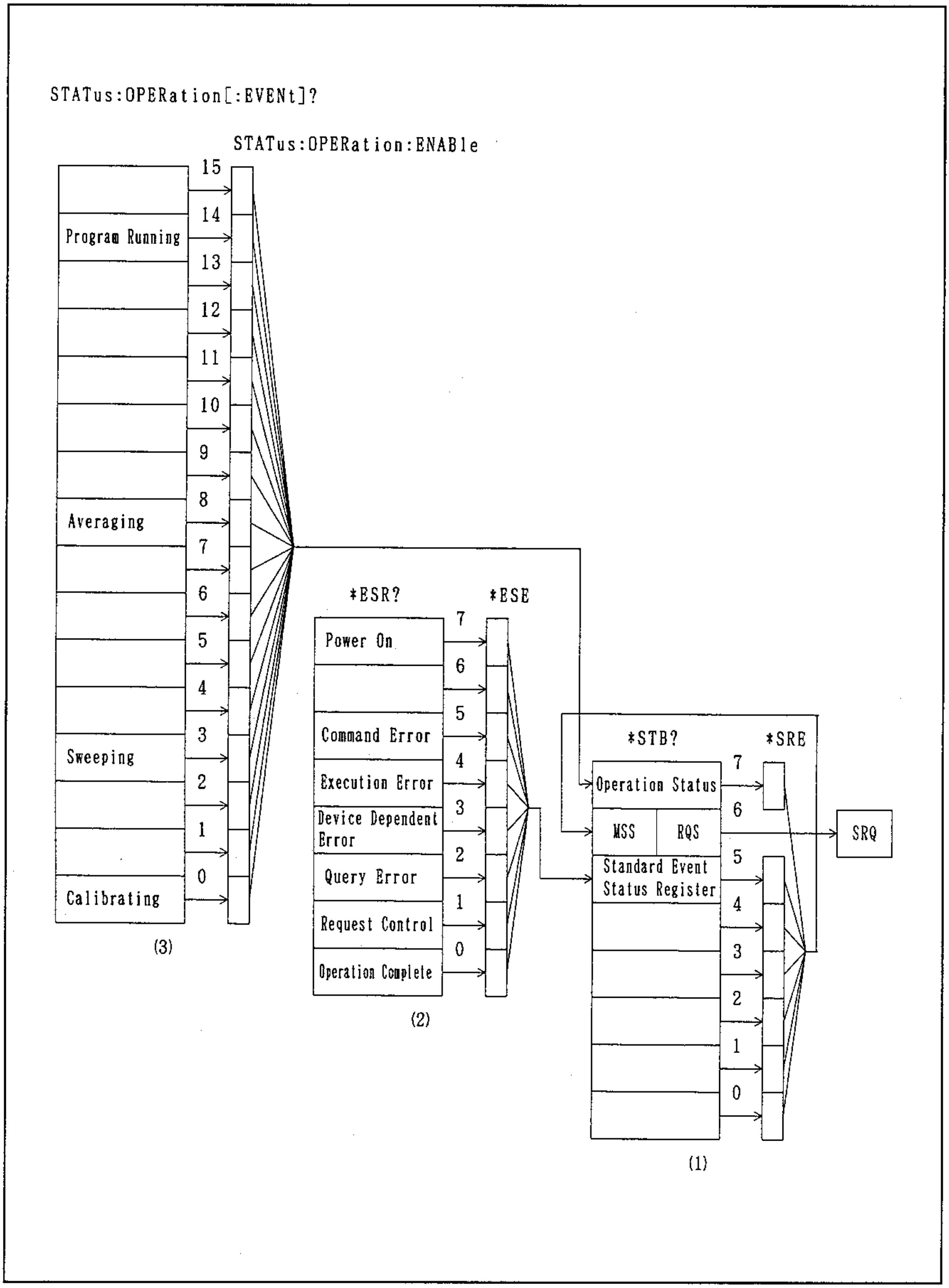
(4) Questionable status register;

(5) Device status register See Section 4.5.

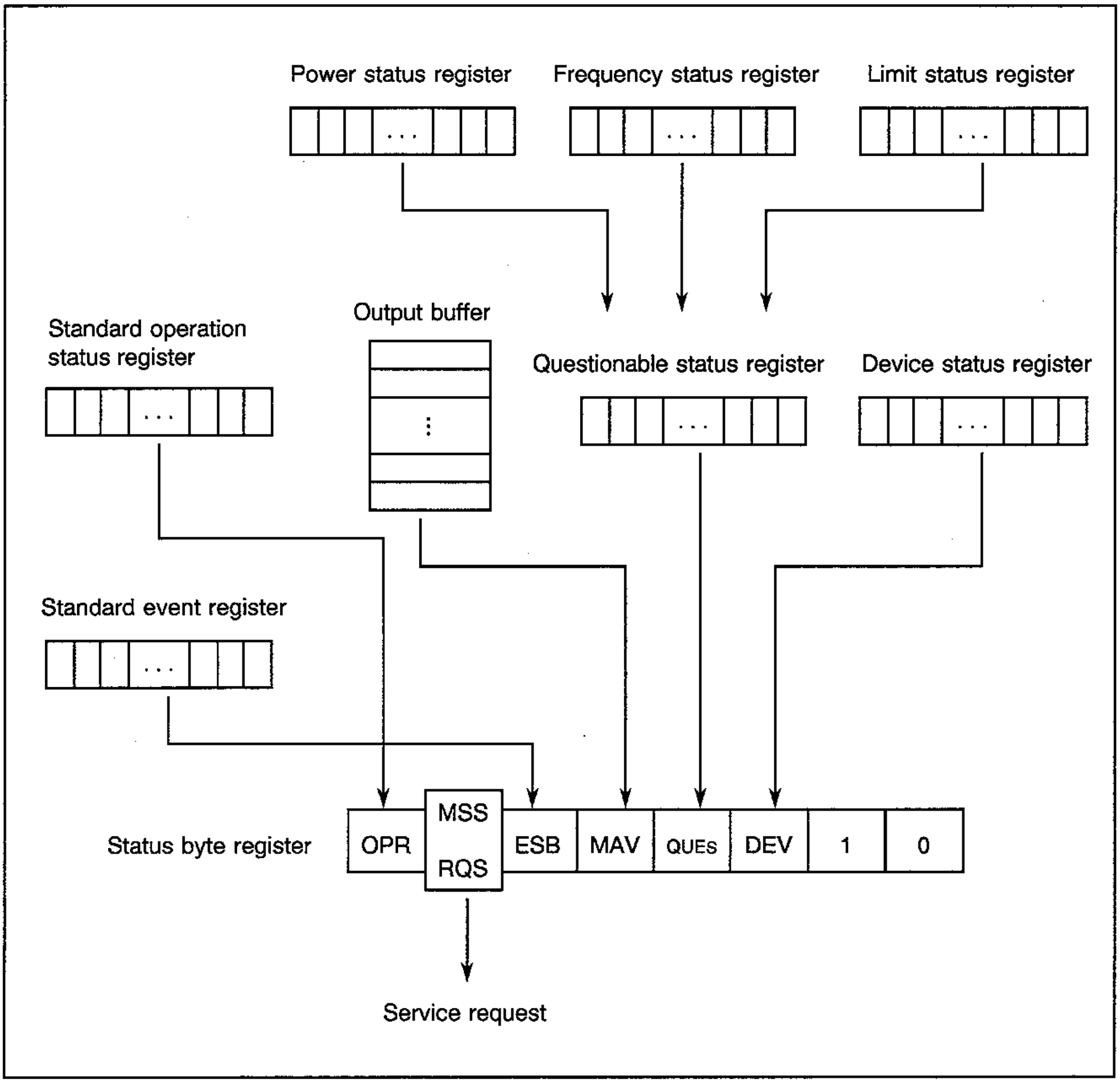
(6) Power status register; See Section 4.6.

(7) Frequency status register; See Section 4.7.

(8) Limit status register; See Section 4.8.


STATus:OPERation[:EVENt]?
STATus:OPERation:ENABle
15
14
Program Running
13
12
11
10
9
8
Averaging
7
6
5
4
3
Sweeping
2
1
0
Calibrating
(3)
*ESR?
*ESE
7
Power On
6
5
Command Error
4
Execution Error
3
Device Dependent Error
2
Query Error
1
Request Control
0
Operation Complete
(2)
*STB?
*SRE
7
Operation Status
6
MSS
RQS
SRQ
5
Standard Event Status Register
4
3
2
1
0
(1)

The figure below shows the arrangement of the status registers in the analyzer.



4.2 Status Byte Register

The status byte register summarizes the information from the status register (see section 4.1.1). In addition, a summary of the status byte register is sent to the controller as a service request. Therefore, the register operates slightly differently from the status register. This section explains the status byte register.

The figure below shows the structure of the status byte register.

Status byte register	OPR	MSS RQS	ESB	MAV	QUES	DEV	1	0	
				⊗					
SRQ enable register	7	×	5	4	3	2	1	0	⊕→MSS

The register has the same functions as the status register explained in section 4.1.1, except with regard to the following three points:

① The summary of the status byte register is written in bit 6 of the status byte register.

② Bit 6 of the enable register is always valid and cannot be changed.

③ Bit 6 (MSS) of the status byte register writes the RQS of the service request.

The register responds to the serial polling from the controller. On doing so, bits 0 to 5 and bit 7 of the status byte register and the RQS are read out, then the RQS is reset to 0. The other bits are not cleared until each factor has been reset to 0.

When the *CLS command is executed, the status byte register, the RQS bit and the MSS bit can be cleared.

The table below explains the meanings of the bits in the status byte register.

bit		Description
7	OPR	• The OPR bit is a summary of the standard operation status register.
6	MSS	• The RQS bit is true when the MSS bit of the status byte register is set to 1. The MSS bit is the summary bit for the entire status data structure. • The service request cannot read out the MSS bit. (However, the MSS bit is understood to be 1 when the RQS bit is 1.) • To read the MSS bit, use the common command *STB?.The *STB? command can read out bits 0 to 5 and bit 7 of the status byte register and the MSS bit. In this case, neither the status byte register nor the MSS bit can be cleared. • The MSS bit cannot become 0 until all the unmasked factors in the status register structure have been cleared.
5	ESB	• The ESB bit is a summary of the standard event register.
4	MAV	• The MAV bit is a summary bit for the output buffer. • When data exist in the buffer, this bit is set to 1. When the data are read out, it is set to 0.
3	QUES	• The QUES bit is a summary of the questionable status register.
2	DEV	• The DEV bit is a summary of the device status register.
0 to 1		• Always 0

4.3 Standard Event Register

The table below shows the assignments of the standard event register.

bit		Description
7	Power on	Set to 1 when the analyzer is switched on
6		Always 0
5	Command Error	Set to 1 when the purser finds a syntax error.
4	Execution Error	Set to 1 when the system fails to execute the instruction received as a GPIB command for some reason (such as out-of-range parameter).
3	Device Dependent Error	Set to 1 when errors other than command errors, execution errors, or query errors occur.
2	Query Error	Set to 1 when no data exist or data have been deleted when the controller attempts to read out data from the analyzer.
1	Request Control	Set to 1 when the analyzer is required to be the active controller.
0	Operation Control	Set to 1 when the analyzer has no command to be executed after receiving an *OPC command.

4.4 Standard Operation Status Register

(1) Condition register
The table below shows the assignments of the condition register for the standard operation status.

bit		Description
15		Always 0
14	Program running	Set to 1 when the built-in BASIC language is running.
4 to 13		Always 0
3	Sweeping	Set to 1 when sweeping is being executed.
1 to 2		Always 0
0	Calibrating	Set to 1 when calibration data are being acquired.

Note: Unlike the event register, the bit 8 (Averaging) is always 0.

(2) Event register
The event register for the standard operation status is used to hold the change from 1 to 0 of the corresponding condition register. The table below shows the assignments of the event register for the standard operation status.

bit		Description
15		Always 0
14	Program running	Set to 1 when the built-in BASIC language stops.
9 to 13		Always 0
8	Averaging	Set to 1 when averaging finishes.
4 to 7		Always 0
3	Sweeping	Set to 1 when sweeping finishes.
1 to 2		Always 0
0	Calibrating	Set to 1 when calibration data acquisition finishes.

4.5 Device Status Register

The table below shows the assignments of the condition register.

bit		Description
0	Cooling Fan Stopped	Sets to 1 when the cooling fan stops.
Others		Always 0

4.6 Power Status Register

The table below shows the assignments of the condition register.

bit		Description
0	Input-R Overloaded	• Sets to 1 when the input-R is overloaded.
1	Input-R Tripped	• Sets to 1 when the protection circuit of the input-R is in operation.
2	Input-A Overloaded	• Sets to 1 when the input-A is overloaded.
3	Input-A Tripped	• Sets to 1 when the protection circuit of the input-A is in operation.
4	Input-B Overloaded	• Sets to 1 when the input-B is overloaded.
5	Input-B Tripped	• Sets to 1 when the protection circuit of the input-B is in operation.
Others		• Always 0

Event register latches the change of the corresponding condition register 0→1. That is, 1 is set when the input is overloaded (or the protection circuit are put into operation).

4.7 Frequency Status Register

The table below shows the assignments of the condition register.

bit		Description
0	Local 1 Unlocked	• Sets to 1 when the local 1 is unlocked.
1	Local 2 Unlocked	• Sets to 1 when the local 2 is unlocked.
2	Synthe Unlocked	• Sets to 1 when the synthesizer is unlocked.
3	External Standard In	• Sets to 1 when the external standard frequency is input.
4	VCXO Unlocked	• Sets to 1 when VCXO is unlocked.
Others		• Always 0

Event register latches the change of the corresponding condition register 0→1. That is, 1 is set when the lock is unlocked.

4.8 Limit Status Register

The table below shows the assignments of the condition register.

bit		Description
0	CH1 1st Limit Failed	• Sets to 1 when the first waveform of the channel 1 is FAIL.
1	CH1 2nd Limit Failed	• Sets to 1 when the second waveform of the channel 1 is FAIL.
2	CH2 1st Limit Failed	• Sets to 1 when the first waveform of the channel 2 is FAIL.
3	CH2 2nd Limit Failed	• Sets to 1 when the second waveform of the channel 2 is FAIL.
4	CH3 1st Limit Failed	• Sets to 1 when the first waveform of the channel 3 is FAIL.
5	CH3 2nd Limit Failed	• Sets to 1 when the second waveform of the channel 3 is FAIL.
6	CH4 1st Limit Failed	Sets to 1 when the first waveform of the channel 4 is FAIL.
7	CH4 2nd Limit Failed	Sets to 1 when the second waveform of the channel 4 is FAIL.

Event register latches the change of the corresponding condition register 0→1. That is, 1 is set when the FAIL arose in each waveform.

4.9 SRQE/SRQD Operation

The analyzer incorporates an expansion which is not specified in IEEE standard 488.2-1987 in the service request system to support R3762/63 compatible mode. The items described here are not applicable to IEEE488.2-1987 command mode.

In R3762/63, the SRQE/SRQD command is used to permit/inhibit service requests. However, since IEEE standard 488.2-1987 uses a status data structure, the enable register can be used to permit/inhibit the service requests. However, since the enable register cannot perform exactly the same functions as the SRQE/SRQD command because of the nature of the register (that is, if the enable register is set to "enable" when its factor is 1, a request will be generated), IEEE standard 488.2-1987 has been expanded only for the SRQE/SRQD signal in R3762/63 command mode.

The SRQE/SRQD command in IEEE488.1-1987 command mode operates as RQS enable/disable of the status data structure. The SRQE command ignores existing requests and does not issue a request. It sends an RQS message to the controller only when a new MSS occurs. The SRQD command always stops origination of the RQS message. Therefore, if the SRQD command and the SRQE command are executed continuously when the RQS state is TRUE, the RQS state will be set to FALSE. Since the controller cannot read out the RQS state at that time, a serial polling must be performed on the analyzer before executing the SRQD command if it is necessary to use the RQS state.

MEMO

5. TRIGGER SYSTEM

This chapter describes the trigger system.

The trigger system is used to synchronize measurement with a specified event. The event may be a GET interface message, a GPIB command such as the *TRG command, or an external trigger signal. The delay time from an event to the start of measurement can also be specified using the trigger system.

5.1 Trigger Model

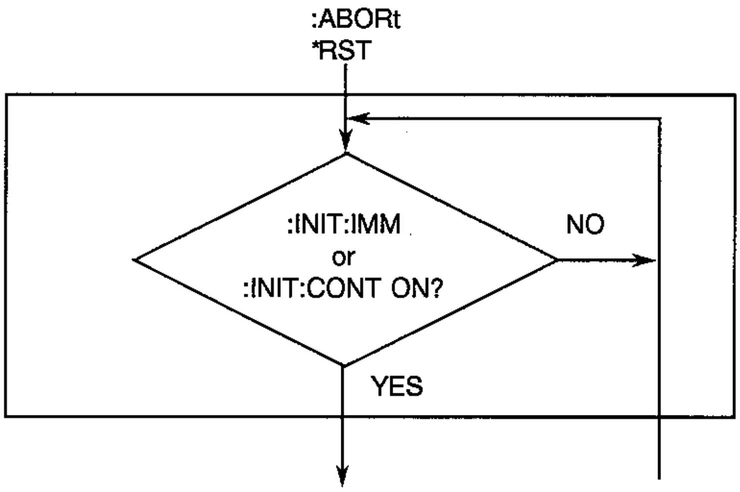
The following shows the model of the trigger system for the analyzer.

```
Switching on power
:ABORt→    | Idle state             |  : See section 5.2.
*RST
                ↓              ↑
           | Trigger waiting state  |  : See section 5.3.
                ↓              ↑
           | Measuring state        |  : See section 5.4.
```

When the analyzer is switched on or when the :ABORt command or the *RST command is executed, the trigger state changes to the idle state. The idle and trigger waiting states wait for conditions that are required for measurement.

5.2 Idle State

When the analyzer is switched on, the trigger system of the analyzer changes to the idle state. Also, the execution of the :ABORt command or the *RST command forcibly changes the trigger system to the idle state. The state changes as follows:

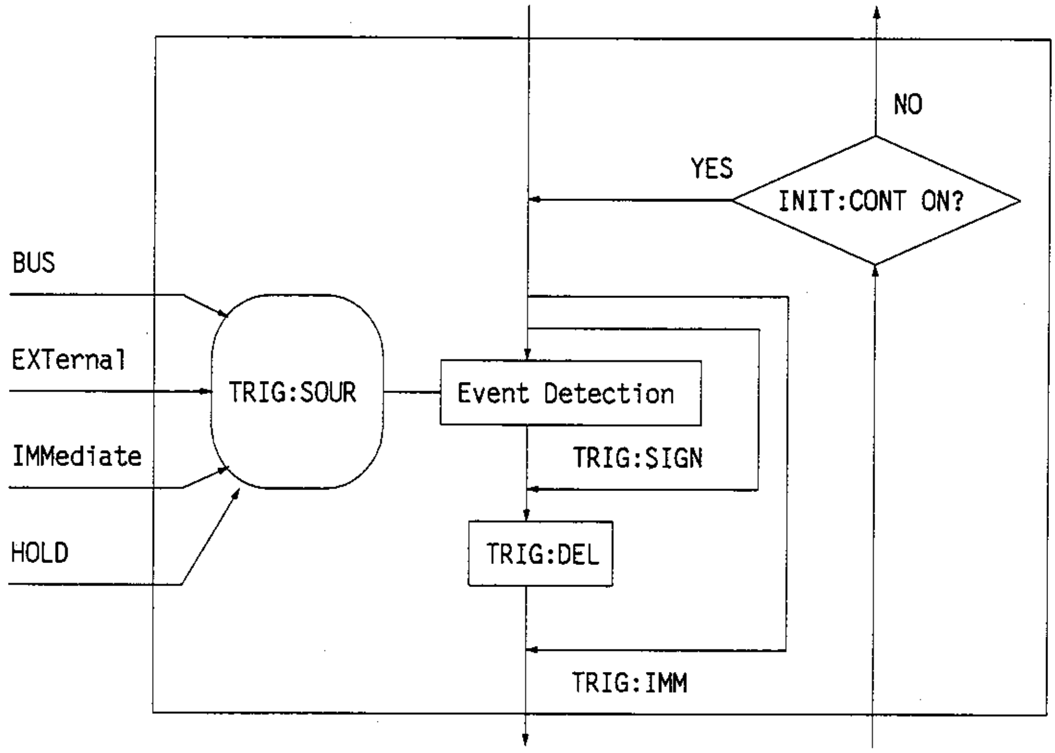


The trigger system does not leave this state until INITiate [:IMMediate] or INITiate:CONTinuous ON. Either of these conditions changes the trigger system to the trigger waiting state.

Note: Since the execution of the *RST command sets INITiate:CONTinuous to OFF, measurement stops.

When the trigger system exits the idle state, the operation pending flag of the analyzer is always set. Also, when the analyzer enters in the idle state, the operation pending flag is cleared. *OPC, *OPC? and *WAI refer to the operation pending flag.

5.3 Trigger Waiting State

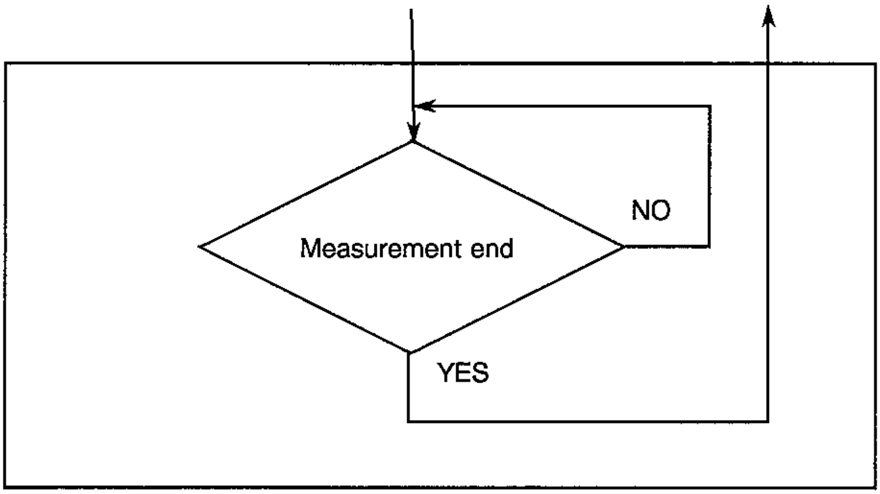


The above is a flowchart of the trigger waiting state of the analyzer. The TRIGger:SOURce command sets the trigger source, and the event detection detects a trigger factor. When the analyzer is triggered and leaves the event detection state, it enters the next state after the time specified by the TRIGger:DELay command has elapsed.

If the analyzer receives the TRIGger:SIGNal command in the trigger waiting state, it will enter the measuring state immediately without entering the event detection state. If it receives the TRIGger [:IMMediate] command in the trigger waiting state, it will enter the measuring state immediately without entering the TRIGger:DELay state.

If the INITiate:CONTinuous signal is set to OFF when the analyzer exits the measuring state, the analyzer will not return to the idle state but will directly enter the next trigger waiting state.

5.4 Measuring State



The analyzer performs measurement in this state. When the analyzer enters the measuring state, it performs sweeping and acquires measurement data.

5.5 IEEE488.1-1987 Command Mode

When the analyzer is in IEEE488.1-1987 command mode, it cannot utilize all of the functions for the trigger system described above. It can utilize only the following four macro commands for the trigger system.

The actual operations of each command in IEEE488.2-1987 command codes are shown on the right. They differ slightly from those used in the actual operation.

CONT	INITiate:CONTinuous ON
SINGLE	INITiate:CONTinuous OFF;:ABORt;INITiate
MEAS	ABORt;INITiate
SWPHLD	INITiate:CONTinuous OFF;:ABORt

6. SAMPLE PROGRAMS

The following are three sample programs:

(1) Program 1: Inputs the center frequency and the span frequency, obtains in levels at all points of the waveform, and substitutes them for variables. After obtaining in all the levels, displays them in the order of 1 to 1201.

(2) Program 2: This is a basic program which performs sweeping once, waits until it has received an SRQ signal indicating the sweeping end while forming a loop, and exits the loop and proceeds to the next loop on receiving the SRQ signal.

(3) Program 3: Inputs the center frequency and the span frequency, searches for a maximum level of the waveform and the frequency at the maximum level, and displays the result.

(1) Program 1

```
100 !************************
110 !*                      *
120 !* BINARY DATA TRANSFER *
130 !*     TEST PROGRAM     *
140 !*                      *
150 !************************
160 !
170 DIM DA(1201)
180 INTEGER N,LP
190 ADD=31
195 OUTPUT ADD;"OLDC OFF"
200 OUTPUT ADD;"DISP:ACT 1;:CALC:FORM MLOG"
210 OUTPUT ADD;"SWE:POIN 1201"
220 OUTPUT ADD;"INIT:CONT OFF"
230 CLS
240 INPUT "CENTER FREQ ? [MHz] =",CF
250 INPUT "SPAN   FREQ ? [KHz] =",SP
260 OUTPUT ADD;"FREQ:CENT ",CF,"MHz"
270 OUTPUT ADD;"FREQ:SPAN ",SP,"KHz"
280 OUTPUT ADD;"FREQ:STAR?"
290 ENTER ADD;STA
300 OUTPUT ADD;"FREQ:STOP?"
310 ENTER ADD;STP
320   P1=POINT1(STA,0)
330   P2=POINT1(STP,0)
340   N=TRANSR(P1,P2,DA(1),0)
350   FOR LP=1 TO 1201
360     PRINT "POINT ";(LP-1);" = ";DA(LP)
370   NEXT LP
380 PRINT "DATA COUNT = ";N
390 STOP
```

Line	Description
100 to 160	Comment lines.
170	Declares the variable arrangement (waveform data are substituted).
180	Declares the variable to be an integer.
190	Substitutes the address of the network analyzer for the variable.
195	Sets the IEEE488.2-1987 command mode.
200	Sets the format of channel 1 to LOGMAG.
210	Sets the measurement point to 1201.
220	Sets the sweeping to the single mode.
230	Deletes characters on the display.
240	Inputs the center frequency and substitutes it for the variable (unit: MHz).
250	Inputs the span frequency and substitutes it for the variable (unit: kHz).
260	Sets to the input center frequency.
270	Sets to the input span frequency.
280	Takes in the start frequency from the analyzer.
290	Substitutes the taken-in value for the variable.
300	Takes in the stop frequency from the analyzer.
310	Substitutes the taken-in value for the variable.
320	Converts the taken-in start frequency into an address point.
330	Converts the taken-in stop frequency into an address point.
340	Substitutes the waveform data (LOGMAG) for the variable.[Data at address point 0 = DA (1): up to 1200 below]
350	Displays data from 1 to 1201 in that order.
360	Displays the variable DA (1 to 1201) for which waveform data are substituted on the display.
370	Repeats until the LP reaches 1201.
380	Finally displays the number of times that data are transferred (1201 times).
390	Program ends.

(2) Program 2

```
100 !*****************
110 !*               *
120 !* SRQ SWEEP TEST *
130 !*               *
140 !*****************
150 !
160 CLS
162 OUTPUT 31;"OLDC OFF"
165 OUTPUT 31;"STAT:OPER:ENAB 8"
170 OUTPUT 31;"SWE:POIN 1201"
180 OUTPUT 31;"SWE:TIME 1S"
190 OUTPUT 31;"INIT:CONT OFF;:ABOR"
200 INPUT "HIT ENT KEY TO SWEEP START !",DUMMY$
210  GOSUB *SWP
220 PRINT "SWEEP TEST FINISHED !!!"
230 STOP
240 !
250 !*****************
260 !
270 *SWP
280   ON ISRQ GOTO *PATH
290   OUTPUT 31;"*SRE 128":SPOLL(31)
300   ENABLE INTR
310   OUTPUT 31;"INIT"
320  *LOOP
330   GOTO *LOOP
340 !
350  *PATH
360   SPOLL(31):DISABLE INTR
370   OUTPUT 31;"*SRE 0"
380 RETURN
```

Line	Description
100 to 150	Comment lines.
160	Deletes characters on the display.
162	Sets the IEEE488.2-1987 command mode.
165	Enables bit 3 (Sweep End) of OPER status.
170	Sets measurement point of network analyzer to 1201.
180	Sets the sweeping time to one second.
190	Sets the sweeping to the single mode.
200	Displays a comment on the CRT. (Go to next with ENTER key.)
210	Calls subroutine (*SWP).
220	Displays a comment on the CRT.
230	Program ends.
240	Comment line
250	Comment line
260	Comment line
270	Subroutine (*SWP)
280	On receiving ISRQ, go to *PATH.
290	Enables SRQ transmission of the standard operation status register.
300	Permits reception of interruption.
310	Sets the sweeping to the single mode. (In this case, performs sweeping once.)
320	*LOOP
330	Goes to *LOOP.(Forms a loop until an ISRQ is received.)
340	Comment line
350	*PATH (Jump destination name when an ISRQ is received.)
360	Inhibits reception of interruptions.
370	Inhibits transmission of all SRQ commands.
380	Returns to the point where the subroutine (*SWP) was called.

(3) Program 3

```
100 !*****************************
110 !*                           *
120 !* MAX SEARCH SAMPLE PROGRAM *
130 !*                           *
140 !*****************************
150 !
155 OUTPUT 31;"OLDC OFF"
160 OUTPUT 31;"DISP:ACT 1;:CALC:FORM MLOG"
170 OUTPUT 31;"SWE:POIN 1201"
180 OUTPUT 31;"SWE:TIME 1S"
190 CLS
200 INPUT "ENTER CENTER FREQ ? [MHz] =",CF
210 INPUT "ENTER SPAN   FREQ ? [KHz] =",SF
220 OUTPUT 31;"FREQ:CENT ",CF,"MHz"
230 OUTPUT 31;"FREQ:SPAN ",SF,"KHz"
240 OUTPUT 31;"FREQ:STAR?"
250 ENTER 31;S1
260 OUTPUT 31;"FREQ:STOP?"
270 ENTER 31;S2
280   PO1=POINT1(S1,0)
290   PO2=POINT1(S2,0)
300   FR=FMAX(PO1,PO2,0)
310   LV=MAX(PO1,PO2,0)
320   FR=FR/10-6
330 PRINT "***** PROGRAM RESULT *****"
340 PRINT "MAX FREQ  [MHz] = ";FR
350 PRINT "MAX LEVEL [dB]  = ";LV
360 STOP
```

Line	Description
100 to 150	Comment lines.
155	Sets the IEEE488.2-1987 command mode.
160	Sets channel 1 of network analyzer to LOGMAG.
170	Sets the number of measurement points to 1201.
180	Sets the sweeping time to one second.
190	Deletes characters on the display.
200	Inputs the center frequency and substitutes it for the variable (unit: MHz).
210	Inputs the span frequency and substitutes it for the variable (unit: KHz).
220	Sets to the input center frequency.
230	Sets to the input span frequency.
240	Takes in the start frequency from the analyzer.
250	Substitutes the taken-in value for the variable.
260	Takes in the stop frequency from the analyzer.
270	Substitutes the taken-in value for the variable.
280	Converts the taken-in start frequency into an address point.
290	Converts the taken-in stop frequency into an address point.
300	Searches for the frequency with the maximum response (level) in the bandwidth.
310	Searches for the maximum response (level) in the bandwidth.
320	Converts the searched-for value into a value in MHz.
330	Displays a comment on the display.
340	Displays a comment and the frequency value of the maximum response.
350	Displays a comment and the maximum response value.
360	Program ends.

MEMO

7. COMMAND REFERENCE

This chapter explains the program for all the remote commands of the analyzer (command program, query program, or both), formats of response data (when there is a query), and details of commands.

Note:
- When referring to a command, consider that part of the command mnemonic can be omitted.

  Example: Although the two following commands are represented differently, they are the same:
  SOURCE:SWEEP:TIME 1S
  SWEEP:TIME 1S

- If you were unable to refer to the command references using a description of SWEEP:TIME, search for a complete description of the command using the attached command list, then refer to the references. If you have a complete description of the command, you can search for it in the table of contents.

7.1 Command Description Format

The following are detailed descriptions for each command mode of IEEE488.2-1987 and IEEE488.1-1987. The following precautions should be taken:

CAUTION

1. The command and response data formats are described using the following symbols:
   < >: Indicates an element of syntax. The contents are written after the symbol.
   |: Indicates selection of one item from among multiple items.
   Example: A | B | C Means that A, B, or C is selectable.
   []: Indicates that the enclosed item is an option (omissible).
   {}: Indicates that the enclosed item is a group of selections separated by | and that you can select one of them.
2. The presence of commands and queries is described in the following:
   Command/Query: Indicates that both a command and a query exist.
   Command: Indicates that only a command exists.
   Query: Indicates that only a query exists.
3. A mnemonic with four characters or more has a short form. In this document, upper-case letters indicate the short form.
   Example: SOURce:SWEep:TIME
   short form: SOUR, SWE
   long form: SOURCE, SWEEP
   Since the term "TIME" consists of four characters, there is no difference between its short form and its long form.

CAUTION

(Continued)

4. Query commands must have "?" as their header. For a query which requires parameters, the query format must be described.
5. The parameter formats commonly used in this chapter are as follows:
   <int>: This is numeric data and can be input in NR1, NR2, or NR3 format. When the analyzer has received the data, they are rounded to a whole number.
   <real>: This is numeric data and can be entered in NR1, NR2, or NR3 format. When the analyzer has received the data, they are rounded to a real number with the valid number of digits.
   <bool>: On/off switch (0: OFF; 1: ON)
   <str>: Character string Indicates an alphanumeric symbol enclosed by " or '. (For IEEE488.1-1987 command mode, do not use " and '.)
   <block>: Block data type
   The contents of data are eight-bit binary data strings.

   For the format, refer to the description of IEEE488.2-1987 command mode.
6. The parameters to be added to a part of the parameter header are shown below. They are commonly used for each command.
   <chno>: 0: active channel
   1: Channel 1
   2: Channel 2
   3: Channel 3
   4: Channel 4
   (Note) It causes error to specify 3 or 4 for <chno>when sub-measure is OFF.
   <trace>: Analysis channel
   (Note) For the command which can specify this, the specifications of <chno>are ignored. In these analysis channels, the channels which can be specified are limited by the command kinds.

CH1	CH2	CH3	CH4	
0	1	4	5 ;	Display data (The first waveform)
2	3	6	7 ;	Memory data (The first waveform)
8	9	12	13 ;	Display data (The second waveform)
10	11	14	15 ;	Memory data (The second waveform)
32	36	48	52 ;	LOGMAG data
33	37	49	53 ;	Phase data
34	38	50	54 ;	LOGMAG data of memory
35	39	51	55 ;	Phase data of memory
40	44	56	60 ;	Real part

CAUTION

(Continued)

CH1	CH2	CH3	CH4	
41	45	57	61 ;	Imaginary part
42	46	58	62 ;	Real part of memory
43	47	59	63 ;	Imaginary part of memory (Hereafter, complex number data)
128	192	256	320 ;	Data array before formatted
129	193	257	321 ;	Data array
130	194	258	322 ;	Memory array
131	195	259	323 ;	Raw data array
133	197	261	325 ;	Normalized standard data array
134	198	262	326 ;	Direction error coefficient array
135	199	263	327 ;	Source match error coefficient array
136	200	264	328 ;	Reflection tracking error coefficient array
137	201	265	329 ;	Forward direction: Directive error coefficient array
138	202	266	330 ;	Forward direction: Source match error coefficient array
139	203	267	331 ;	Forward direction: Reflection tracking error coefficient array
140	204	268	332 ;	Forward direction: Load match error coefficient array
141	205	269	333 ;	Forward direction: Transmission tracking error coefficient array
142	206	270	334 ;	Forward direction: Isolation error coefficient array
143	207	271	335 ;	Reverse direction: Directive error coefficient array
144	208	272	336 ;	Reverse direction: Source match error coefficient array
145	209	273	337 ;	Reverse direction: Reflection tracking error coefficient array
146	210	274	338 ;	Reverse direction: Load match error coefficient array
147	211	275	339 ;	Reverse direction: Transmission tracking error coefficient array
148	212	276	340 ;	Reverse direction: Isolation error coefficient array

<input>: 1: R channel
2: A channel
3: B channel

<port>: 1: PORT 1
2: PORT 2

<eport>: 1: R channel
2: A channel
3: B channel
4: PORT 1
5: PORT 2

CAUTION

(Continued)

<n>: n: Integer value defined by each command

Example: To set the measurement format of channel 1 to MLOG using CALCulate[<chno>]:FORMat, input the following:

CALCulate1:FORMat MLOG

<parano>: In case that the display format is the type of rectangular coordinates.

0: Main trace

1: Sub trace

In case that the display format is the type of polar coordinates

0: Amplitude or real part

1: Phase or imaginary part

7.2 Common Commands

1.	*CLS	IEEE488.1-1987 command mode *CLS

- Function — Clearing of status byte and related data
- Presence of command and query — Command
- Command — *CLS
- Description — The *CLS command clears the status data structure and forcibly cancels *OPC and *OPC?. It also clears the error queue. However, since this command does not clear the output buffer, the MAV bit is not cleared when there are output data. However, since the data are cleared if this command is executed at the beginning of the line, all the status bits, including the MAV status bit, are cleared.
  The *CLS command also clears the error queue.

2. *DDT

- Function: Macro definition for GET
- Presence of command and query: Command / Query
- Command: *DDT <block>
- Parameter: <block>
- Response type: <block>
- Description: The *DDT command defines the command sequence which is to be executed when the *TRG interface message or the *GET interface message is received. That is, it replaces the *TRG operation with a series of commands which has been written into the <block> data. The length of the sequence to be defined must not exceed 255 characters.
  If the *DDT command defines block data (#10) with a length of 0, the *TRG interface message or the GET interface message will execute nothing. The macro can be canceled by executing the *RST command.
  Block data are used to respond a query. If the *DDT? command is executed with the macro not yet defined, block data (#10) with a length of 0 will be returned.
- Note: Do not use the *TRG interface message in this definition. If it is used in the definition with the *DDT command, the sequence set by the *DDT command will be called instead of the trigger, and thus an endless loop will be formed. (Actually, a macro error will occur because of nesting limitation.)
- Example: When the *DDT command is #214INIT;TRIG:SIGN, *TRG replaces INIT;TRIG:SIGN.

3. *DMC

- Function — Macro definition

- Presence of command and query — Command

- Command — *DMC <str>,<block>

- Parameter — <str>
  <block>

- Description — The *DMC command defines the command sequence in the macro label specified by <str>. When <str> is received, the definition allows the system to operate as if it has received <block> itself. (However, *EMC must be 1.)
  A hierarchical command can be used for this macro label. In addition, it is possible to overwrite the macro on R3764/66, R3765/67 command defined in advance. (However, it is not possible to overwrite on the common command.) Then, when the macro is enabled by *EMC 1, the system will perform the original operation by disabling a series of commands which has been replaced with the macro using *EMC 0. Use the *PMC command to delete the macro which has been defined by the *DMC command. Once registered, a macro cannot be re-registered until it has been cleared by the *PMC command.
  Follow the grammar of R3764/66, R3765/67 command to write the macro body. Up to nine parameters ($1 to $9) can be given to the macro command. "1" must be given to the parameter following the macro command, "2" to the next parameter, and so on. Also, the macro definition can include the macro. Up to nine levels of nesting are supported. Up to 30 macros can be registered as new macros (depending on the condition).
  See *PMC, *GMC?, *LMC? and *EMC.

- Example — When the *DMC command is "SWPINIT",#221FREQ:START $1;STOP $2, SWPINIT 100MHZ,500MHZ replaces FREQ:START 100MHZ:STOP 500MHZ.

4. *EMC

- Function: Permission for macro execution
- Presence of command and query: Command / Query
- Command: *EMC <int>
- Parameter: <int>
- Response type: 0 | 1
- Description: The *EMC command permits (1) or inhibits (0) the execution of the macro.
  This command does not affect the contents of the macro definition. It is used to execute an original command which has been overwritten by the macro.
  *RST inhibits the execution of the macro.
  See *DMC, *PMC, *GMC? and *LMC?.

5. *ESE — IEEE488.1-1987 command mode *ESE

- Function: Setting of standard event status enable register
- Presence of command and query: Command / Query
- Command: *ESE <int>
- Parameter: <int>
- Response type: NR1 (integer value)
- Description: The *ESE command sets the enable register in the standard event status register. The standard event status register corresponding to the bit set to 1 in this register is reflected in the status byte register as a valid bit.
  For details, see the description of the status data structure and *ESR?.
- Example: When the operation control bit (bit 3) and the device dependent error bit (bit 0) are set to "enable", calculate:
  $2^3 + 2^0 = 8 + 1 = 9$ and set *ESE 9.

6. *ESR? — IEEE488.1-1987 command mode *ESR?

- Function: Readout of standard event status register
- Presence of command and query: Query
- Query: *ESR?
- Response type: NR1 (integer value)
- Description: The *ESR command reads out the standard event status register value. When the register is read out, it is cleared and the corresponding bit (bit 5) of the status byte is cleared.
  For details, see the description of the status data structure.

Table Standard Event Register Assignmen

bit		Description
7	Power on	Set to 1 when the system is switched on
6		Always 0
5	Command Error	Set to 1 when the purser detects a grammar error
4	Execution Error	Set to 1 when the system fails to execute the instruction which has been received as a GPIB command for some reason (such as parameter out of range)
3	Device Dependent Error	Set to 1 when an error other than a command error, an execution error, or a query error occurs
2	Query Error	Set to 1 if there are no data or if data have been deleted when the controller attempts to read out data from the analyzer
1	Request Control	Set to 1 when the analyzer is required to be active controller
0	Operation Control	Set to 1 when the analyzer has no command to be executed after it has received the *OPC command

7. *GMC?

- Function: Query of macro definition
- Presence of command and query: Query
- Query: *GMC? <name>
- Parameter: <name>
- Response type: <block>
- Description: The *GMC? command reads out the macro definition specified by <name>.
  If the command reads out an undefined <name> macro, block data (#10) with a length of 0 will be returned.
  See *DMC, *PMC?, *LMC? and *EMC.

8. *IDN? — IEEE488.1-1987 command mode IDNT?

- Function: Query of devices
- Presence of command and query: Query
- Query: *IDN?
  IDNT?
- Response type: "<manufacturer>,<model>,<serial number>,<firmware level>"
  <manufacturer> = ADVANTEST
  <model> = Model name
  <serial number> = Serial number
  <firmware level> = System version
- Description: The *IDN? extracts system identification information. This command outputs four items in the character string format, as shown in the response format above.

9. *LMC?

- Function: Readout of all macros
- Presence of command and query: Query
- Query: *LMC?
- Response type: "<macro label>"[,"<macro label>"...]
  <macro label> = Macro header
- Description: Answers all the macro headers in the character string format. When multiple macros are defined, they are separated by ",". If there is no defined macro, the system responds with a character string with a length of 0 ("").
  See *DMC, *PMC, *GMC? and *EMC.

10. *OPC — IEEE488.1-1987 command mode *OPC

- Function: Notification of end of all operations in progress
- Presence of command and query: Command / Query
- Command: *OPC
- Response type: 1
- Description: The *OPC command sets the 'Operation Control' bit of the standard event status register to 1 when all commands being executed have been completed. If the next command is received before the command being executed finishes, the *OPC command waits until the execution of that command has been completed. Therefore, if the analyzer does not execute a command after receiving the *OPC command, the status register will be set.

  The *OPC? writes 1 into the output buffer while the *OPC command above sets the 'Operation Control' bit. Therefore, the *OPC? command allows the command to be finished when the controller receives the response from the analyzer.

  Both *OPC and *OPC? can be canceled by using a DCL interface message, the *CLS command, or the *RST command.
  See *WAI.

11. *PCB   IEEE488.1-1987 command mode *PCB

- Function: Setting of the GPIB address used to return the right of control
- Presence of command and query: Command
- Command: *PCB <primary>[,<secondary>]
- Parameter: <primary>
  <secondary>
  Note: In IEEE488.1-1987 command mode, <secondary> cannot be input and must always be omitted.
- Description: The *PCB command sets the address of the external controller to which the analyzer is connected.

12. *PMC

- Function: Deletion of all macro definitions
- Presence of command and query: Command
- Command: *PMC
- Description: The *PMC command deletes all the macro definitions. This command deletes all the macro headers and bodies from the memory of the analyzer, making it possible to register new macros.
  See *DDT *DMC *GMC?, *LMC? and *EMC.

13.	*RCL	IEEE488.1-1987 command mode RECLREG{1|2|3|4|5|6|7|8|9|10} RECLPOFF

- Function — Recall of device settings
- Presence of command and query — Command
- IEEE488.2-1987 command mode
  - Command: *RCL {<int> | POFF}
  - Parameter: <int> = register number
    POFF = Setting at previous switching-off
- IEEE488.1-1987 command mode
  - Command: RECLREG{1|2|3|4|5|6|7|8|9|10}
    RECLPOFF
- Description — The *PMC command recalls the setting condition of the analyzer from the specified internal register. If a register number 0 or POFF (or RECLPOFF) is used, this command recalls the settings at the previous switching-off.

14.	*RST	IEEE488.1-1987 command mode *RST

- Function — Resetting of devices
- Presence of command and query — Command
- Command — *RST
- Description — The *RST command resets the analyzer. The following operations are performed on the system:

① System initialization (See "A.3 Initialization".)
② Initialization of the macro defined by the *DDT command.
③ Invalidation of the macro (Same as *EMC 0)
④ Invalidation of the *OPC bit and the *OPC? bit
⑤ Resetting of the trigger system

The resetting does not affect:
① GPIB bus condition
② GPIB address
③ Output buffer
④ Status data structure
⑤ Macro defined by the *DMC command
⑥ Calibration data of the device
See SYSTem:PRESet(IP).

15.	*SAV	IEEE488.1-1987 command mode SAVEREG{1|2|3|4|5|6|7|8|9|10}

- Function — Saving of device settings
- Presence of command and query — Command
- IEEE488.2-1987 command mode
  - Command — *SAV <int>
  - Parameter — <int>
- IEEE488.1-1987 command mode
  - Command — SAVEREG{1|2|3|4|5|6|7|8|9|10}
- Description — The *SAV command saves the setting condition of the analyzer in an internal register with a specified number.
  The internal register is backed up with a built-in battery. However, calibration data are not backed up. When the analyzer is switched off, the calibration data and their related settings are cleared.
  If the register already contains data, the new setting will be written over them.

16.	*SRE	IEEE488.1-1987 command mode *SRE

- Function — Setting of service request enable register
- Presence of command and query — Command / Query
- Command — *SRE <int>
- Parameter — <int>
- Response type — NR1 (integer value)
- Description — The *SRE command sets the service request enable register. The status byte register corresponding to the bit in this register which is set to 1 is reflected in the MSS bit as a valid bit.
  Bit 6 of the response data for the query command is always 0.
  For details, see the description of the status data structure.
  See *STB?.
- Example — If the OPR bit (bit 7), the ESB bit (bit 5) and the MAV bit (bit 4) are set to "enable", calculate:
  $2^7 + 2^5 + 2^4 = 128 + 32 + 16 = 176$ and set *SRE 176.

17. *STB? IEEE488.1-1987 command mode *STB?

- Function: Readout of status byte register
- Presence of command and query: Query
- Query: *STB?
- Response type: NR1 (integer value)
- Description: The *STB? command reads out the contents of the status byte register.
  The summary bit of the request to be read out here is the MSS bit.
  This register and the MSS bit are not cleared, even if the register is read out.
  For details, see the description of the status data structure.

Status Byte Register Assignments

bit		
7	OPR	• OPR is a summary of the standard operation status register.
6	MSS	• When the MSS bit of the status byte register is set to 1, the RQS bit is TRUE and the MSS bit is the summary bit for all of the status data structure. • The service request cannot read out the MSS bit. (However, when the RQS bit is 1, it is understood that the MSS bit is 1.) • To read the MSS bit, the common command *STB? should be used. The *STB? command can read out bits 0 to 5 and bit 7 of the status byte register and the MSS bit. In this case, the status byte register and the MSS bit are not cleared. • The MSS bit does not become 0 until all the unmasked factors in the status register structure are cleared.
5	ESB	• The ESB bit is a summary of the standard event register.
4	MAV	• The MAV bit is a summary bit of the output buffer. • The MAV bit is 1 when the output buffer has data to be output and it is 0 when the data are read out.
3	QUES	• The QUES is a summary of the questionable status register.
2	DEV	• The DEV is a summary of the device status register.
0 to 1		• Always 0

18. *TRG — IEEE488.1-1987 command mode *TRG

- Function: Triggering device
- Presence of command and query: Command
- Command: *TRG
- Description: The *TRG command triggers devices. This command has exactly the same effect as the GET interface message. If the analyzer receives the *TRG interface message when TRIG:SOUR is set to BUS and the analyzer is in the trigger waiting state (see "5. TRIGGER SYSTEM"), it starts measurement. Under conditions other than above, this command is ignored.
Both the *TRG interface message and the GET interface message are stored in the input buffer and they are processed in the order of inputting.

19. *TST? — IEEE488.1-1987 command mode *TST?

- Function: Query of self test result
- Presence of command and query: Query
- Query: *TST?
- Response type: 0 | error code
- Description: The *TST? command allows the analyzer to start the self test and return the result. Answering with 0 indicates that the test has been passed, while other answers indicate error codes. For the analyzer, answers other than "0" are not returned in response to "*TST?".

20.	*WAI	IEEE488.1-1987 command mode *WAI

- Function — Waiting for end of all operations being performed
- Presence of command and query — Command
- Command — *WAI
- Description — The *WAI command is used to wait for the completion of all the commands which are being executed. If this command is executed, all commands input after that time will be delayed until all the commands being executed have been completed.
  *WAI can be canceled by means of the DCL interface message.

7.3 ABORt Subsystem

1. ABORt

• Function	Resetting trigger module
• Presence of command and query	Command
• Command	ABORt
• Description	The ABORt command resets the trigger system and forcibly sets the trigger state to the idle state. At the same time, the measurement is stopped and the average count is reset. Also, the device operation pending flag is cleared. The use of this command does not change INITiate:CONTinuous. Therefore, when CONTinuous is set to ON, the system moves immediately to the next trigger waiting state. See INITiate Subsystem and TRIGger Subsystem.

7.4 CALCulate Subsystem

1.	CALCulate[<chno>]:FORMat	IEEE488.1-1987 command mode LOGMAG,PHASE,DELAY,LINMAG,SWR,REAL, IMAG,UNWRAP,LINMP,LOGMP,LOGMD,POLAR, SRJX,SGJB

- Function — Selection of measurement format
- Presence of command and query — Command / Query
- IEEE488.2-1987 command mode
  - Command — CALCulate[<chno>]:FORMat <format>
  - Parameter — <format> = {MLOGarithmic | PHASe | GDELay | MLINear | SWR | REAL | IMAGinaly | UPHase | MLIPhase | MLOPhase | MLODelay | POLar | SCHart | ISCHart}
  - Response type — MLOG | PHAS | GDEL | MLIN | SWR | REAL | IMAG | UPH | MLIP | MLOP | MLOD | POL | SCH | ISCH
- IEEE488.1-1987 command mode
  - Command — LOGMAG
    PHASE
    DELAY
    LINMAG
    SWR
    REAL
    IMAG
    UNWRAP
    LINMP
    LOGMP
    LOGMD
    POLAR
    SRJX
    SGJB
  - Response type — 0 | 1
- Description — Specifies measurement formats such as amplitude, phase and group delay.

  Initial value MLOPhase

  The input signal is measured as a complex number in the form X + jY, and the signal is calculated in accordance with the specified measurement format, as shown in the table below:

R3762/63 command	R3764/66, R3765/67 command parameter	Calculation expression: (unit · relative measurement/absolute value)	Contents
LOGMAG	MLOG	$10\log_{10}(X^2+Y^2)$:(dB/dBm)	Amplitude (logarithm)
PHASE	PHAS	arctan(Y/X):(deg/deg)	Phase
DELAY	GDEL	$\dfrac{-\Delta\ (\text{phase})}{360\times\Delta\ (\text{frequency})}$:(Sec/Sec)	Group delay
LINMAG	MLIN	$\sqrt{X^2+Y^2}$:(Unit/Vrms)	Amplitude
SWR	SWR	$\dfrac{1+\Gamma}{1-\Gamma}$:(Unit/Unit) $\Gamma=\sqrt{X^2+Y^2}$	Reflection coefficient
REAL	REAL	X:(Unit/Unit)	Real part
IMAG	IMAG	Y:(Unit/Unit)	Imaginary part
UNWRAP	UPH	arctan(Y/X):(deg/deg)	Phase PHASE indicates a value within a range of ±180°. UNWRAP indicates a continuous value using the first measurement point as reference without turning back at ±180°.
LINMP	MLIP	pair(r1,r2) r1 = $\sqrt{X^2+Y^2}$:(Unit/Vrms) r2 = arctan(Y/X):(deg/deg)	Amplitude and phase pair rectangular coordinate display
LOGMP	MLOP	pair(r1,r2) r1 = $10\log_{10}(X^2+Y^2)$:(dB/dBm) r2 = arctan(Y/X):(deg/deg)	Amplitude (logarithm) and phase pair rectangular coordinate display
LOGMD	MLOD	pair(r1,r2) r1 = $10\log_{10}(X^2+Y^2)$:(dB/dBm) r2 = $\dfrac{-\Delta\ (\text{phase})}{360\times\Delta\ (\text{frequency})}$:(Sec/Sec)	Amplitude (logarithm) and group delay pair rectangular coordinate display
POLAR	POLar	X:(Unit/Unit) Y:(Unit/Unit)	Real part Imaginary part
SRJX	SCHart	X:(Unit/Unit) Y:(Unit/Unit)	Real part Imaginary part
SGJB	ISCHart	X:(Unit/Unit) Y:(Unit/Unit)	Real part Imaginary part

2. CALCulate[<chno>]:GDAPerture:APERture — IEEE488.1-1987 command mode APERTP

- Function: Group delay aperture setting
- Presence of command and query: Command / Query
- Command: CALCulate[<chno>]:GDAPerture:APERture <real>
  APERTP<real>
- Parameter: <real>
- Response type: NR3 (real value)
- Description:

Sets the aperture of the group delay.

Initial value: 10%
Setting range: 0.01% to 50%
Setting resolution: 0.01%

The group delay can be calculated using the expression below, in which △ (frequency) is called "aperture".

$$\text{Group delay} = \frac{-\triangle\ (\text{phase})}{360 \times \triangle\ (\text{frequency})}$$

The aperture (△ (frequency)) is converted into the measurement point (horizontal axis) and determined for the setting value <real> as follows:

$$\triangle(\text{frequency}) = \triangle(\text{point})$$
$$= \frac{\text{number of measurement point-1}}{100} \times <\text{real}>$$

That is, the setting value <real> is set as a percentage of the number of measurement points. The value is maintained even if the number of measurement points is changed. The △ point is calculated internally again using the number of measurement points after the change.

- Example

Number of measurement points: 101 point

Aperture: $2(\%) \rightarrow \triangle(\text{point}) = \frac{101-1}{100} \times 2$
$= 2$

Measurement points:

n-1 n n+1
○ ○ ○ ○ ○
△(point) = 2

3. CALCulate[<chno>]:MATH[:EXPRession]:NAME   IEEE488.1-1987 command mode DISPDDM

- Function — Data (+, -, x, /) memory setting
- Presence of command and query — Command / Query
- IEEE488.2-1987 command mode
  - Command — CALCulate[<chno>]:MATH[:EXPRession]:NAME <type>
  - Parameter — <type> = {NONE | DDM | DMM | DAM | DSM}
  - Response type — NONE | DDM | DMM | DAM | DSM
- IEEE488.1-1987 command mode
  - Command — DISPDDM<bool>
  - Parameter — <bool> = {ON | OFF}
  - Response type — 0 | 1
- Description — Calculates the relationship between the measurement data and the memory data.

R3762/63 command	R3764/66, R3765/67 command parameter	Calculation
DISPDDM ON	DDM	÷
	DMM	×
	DAM	+
	DSM	-
DISPDDM OFF	NONE	NONE

- Note — The calculation is valid only when the relationship between the data and the memory in the same channel is calculated. (It is not possible to calculate the relationship between the data and the memory in different channels.)
  DDM (÷) is used to normalize the data.
  The calculation is performed on the vector quantity (complex number data) before formatting.

4. CALCulate[<chno>]:SMOothing:APERture — IEEE488.1-1987 command mode SMOOAPER

• Function	Smoothing span setting
• Presence of command and query	Command / Query
• Command	CALCulate[<chno>]:SMOothing:APERture <real> SMOOAPER<real>
• Parameter	<real>
• Response type	NR3 (real number value)

• Description

Sets the smoothing aperture.

Initial value: 10%

Setting range: 0.01% to 50%

Setting resolution: 0.01%

The smoothing is provided by the algorithm below. (2m) is called "aperture".

Smoothing algorithm

$$\bar{D}_{(n)} = \frac{D_{(n-m)} + \cdots + D_{(n)} + \cdots + D_{(n+m)}}{2m+1}$$

$\bar{D}_{(n)}$: Smoothed nth data after formatting

$D_{(n)}$: nth data before smoothing

2m: Smoothing aperture

The aperture is obtained for the setting value <real> using the expression below:

Aperture(2m)

$$= \frac{\text{(number of measurement point)-1}}{100} \times \text{<real>}$$

That is, the setting value <real> is set as a percentage of the number of measurement points. The setting value <real> is maintained even if the number of measurement points is changed and the aperture (2m) is calculated internally again using the number of measurement points after the change.

• Example

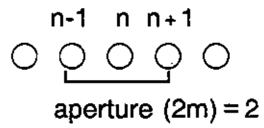
Number of measurement points: 101 point

Aperture: 2(%)→aperture (2m)

$$= \frac{101-1}{100} \times 2$$

$$= 2$$

Measurement points:



5. CALCulate[<chno>]:SMOothing:STATe — IEEE488.1-1987 command mode SMOO

- Function: ON/OFF of smoothing
- Presence of command and query: Command / Query
- Command: CALCulate[<chno>]:SMOothing:STATe <bool>
  SMOO<bool>
- Parameter: <bool>
- Response type: 0 | 1
- Description:

Performs smoothing.
Smoothing is used to obtain the moving average between adjacent formatted data.
By smoothing the noise component, the average of the noise can be obtained.
In contrast to this, since the averaging obtains the time average of the data before formatting (vector quantity), the noise is reduced rather than averaged.
Smoothing algorithm

$$\bar{D}_{(n)} = \frac{D_{(n-m)} + \cdots + D_{(n)} + \cdots + D_{(n+m)}}{2m+1}$$

$\bar{D}_{(n)}$: Smoothed nth data after formatting
$D_{(n)}$: nth data before smoothing
2m: Smoothing aperture

- Note

When the measurement format is set to 2 traces (MLOP, MLOD, MLIP) or the memory trace is set to ON, smoothing is performed for all the traces.

- Example

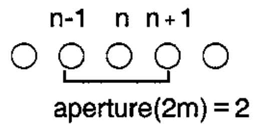
Number of measurement points: 101 point

Aperture: 2(%)→Aperture(2m)

$$= \frac{101-1}{100} \times 2$$

$$= 2$$

Measurement points:



6. CALCulate[<chno>]:TRANSform:IMPedance:CIMPedance — IEEE488.1-1987 command mode: SETZ0, MKRZ0{50|75}

- Function: Z conversion characteristic impedance setting
- Presence of command and query: Command / Query
- Command: CALCulate[<chno>]:TRANSform:IMPedance:CIMPedance <real>
  SETZ0 <real>
  MKRZ0{50|75}
- Parameter: <real>
- Response type: NR3 (real number value)
  0 | 1 (MKRZ0{50|75})
- Description: Sets the characteristic impedance for the impedance measurement.

  Initial value: 50Ω
  Setting range: 100pΩ to 1GΩ
  Setting resolution: 0.001pΩ

  The measurement value is obtained using the value normalized by the characteristic impedance of the measurement system (1 Ω). Therefore, to obtain the absolute value, it is necessary to specify the characteristic impedance of the measurement system.
- Example: To obtain the impedance using the reflection coefficient.

  Normalized impedance: $\frac{1+\Gamma}{1-\Gamma} \times 1(\Omega)$

  Absolute value impedance : $\frac{1+\Gamma}{1-\Gamma} \times Z_0$

  Γ: Reflection coefficient gain
  Z_0: Characteristic impedance

7. CALCulate[<chno>]:TRANsform:IMPedance:TYPE   IEEE488.1-1987 command mode CONV{OFF|RZ|RY|TZ|TY|1DS}

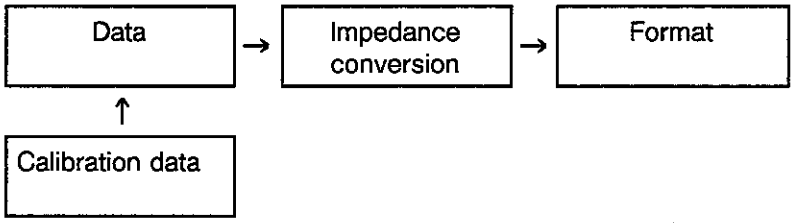
- Function — Z conversion type setting
- Presence of command and query — Command / Query
- IEEE488.2-1987 command mode
  - Command — CALCulate[<chno>]:TRANsform:IMPedance:TYPE <type>
  - Parameter — <type> = { NONE | ZREFlection | YREFlection | ZTRansmit | YTRansmit | INVersion }
  - Response type — NONE | ZREF | YREF | ZTR | YTR | INV
- IEEE488.1-1987 command mode
  - Command — CONV{OFF|RZ|RY|TZ|TY|1DS}
  - Response type — 0 | 1
- Description — Obtains the impedance from the reflection coefficient and the transfer characteristics using the table below:

R3762/63 command	R3764/66, R3765/67 command parameter	Converted value	Conversion expression
CONVOFF	NONE	No conversion	
CONVRZ	ZREF	Reflection impedance	$\frac{1+\Gamma}{1-\Gamma} \times Z_0$
CONVRY	YREF	Reflection admittance	$\frac{1-\Gamma}{1+\Gamma} \times \frac{1}{Z_0}$
CONVTZ	ZTR	Transfer impedance	$\frac{2(1-T)}{T} \times Z_0$
CONVTY	YTR	Transfer admittance	$\frac{T}{2(1-T)} \times \frac{1}{Z_0}$
CONV1DS	INV	Reverse S parameter	$\frac{1}{S}$

Γ: Reflection coefficient gain
T: Gain
S: Γ or T
Z_0: Characteristic impedance

•Note

The data processing flow is as follows:



7.5 DISPlay Subsystem

1. DISPlay:ACTive — IEEE488.1-1987 command mode CH{1|2|3|4}

- Function: Active channel specification

- Presence of command and query: Command / Query

- IEEE488.2-1987 command mode
  - Command: DISPlay:ACTive <int>
  - Parameter: <int>
  - Response type: NR1 (integer value)

- IEEE488.1-1987 command mode
  - Command: CH{1|2|3|4}
  - Response type: 0 | 1

- Description

Selects the active channel (Initial setting channel 1)
The analyzer is equipped with four measurement channels, which can be used independently for measurement and data display.
For the functions dependent on these channels, it is possible to specify <chno> as the header parameter of the command. When <chno> is omitted or IEEE488.1-1987 command is used, all the other commands are applied to the active channel specified here.

R3762/63 command	R3764/66, R3765/67 command parameter	Operation
CH1	1	Channel 1 is active.
CH2	2	Channel 2 is active.
CH3	3	Channel 3 is active.
CH4	4	Channel 4 is active.

(Note) When sub measure is OFF, the sub channel cannot be switched to active. The sub measure must be switched ON previously.
When the sub measure is switched ON/OFF, sometimes the active channel is switched automatically.
(Refer to 7.10.19 [SENSe:]FUNCtion[<chno>][:ON] and 7.10.20 [SENSe:]FUNCtion[<chno>]:POWer.)

2.	DISPlay:DUAL	IEEE488.1-1987 command mode DUAL

- Function — ON/OFF of dual channel
- Presence of command and query — Command / Query
- Command — DISPlay:DUAL <bool>
  DUAL<bool>
- Parameter — <bool>
- Response type — 0 | 1
- Description — Selects whether two measurement channels (CH1 and CH2) are to be displayed simultaneously or one of the channels is to be displayed.
  When the sub measure is selected, channel 3 and channel 4 are displayed too.

  Initial setting DUAL OFF

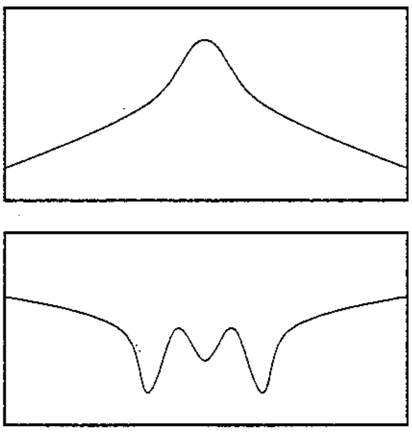
3. DISPlay:FORMat — IEEE488.1-1987 command mode SPLIT

- Function — Split/overlap selection
- Presence of command and query — Command / Query
- IEEE488.2-1987 command mode
  - Command — DISPlay:FORMat <type>
  - Parameter — <type> = {ULOWer | FBACk}
  - Response type — ULOW | FBAC
- IEEE488.1-1987 command mode
  - Command — SPLIT <bool>
  - Parameter — <bool> = {ON | OFF}
  - Response type — 0 | 1
- Description — Selects the split display or the overlap display.
  Initial setting SPLIT OFF

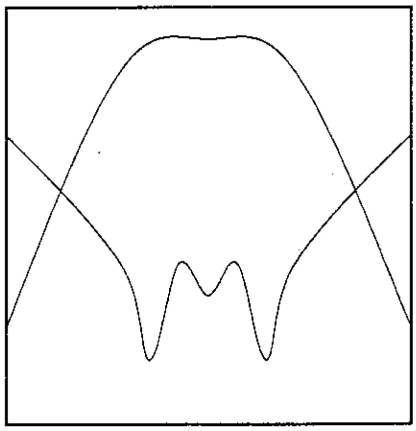
R3762/63 command	R3764/66, R3765/67 command parameter	Operation
SPLIT ON	ULOW	Split display
SPLIT OFF	FBAC	Overlap display

- Example

Split display



Overlap display



4. DISPlay[:WINDow[<chno>]]:TEXT[:DATA] — IEEE488.1-1987 command mode LABEL

- Function — Label setting
- Presence of command and query — Command / Query
- Command — DISPlay[:WINDow[<chno>]]:TEXT[:DATA] {<str>|<block>}
  LABEL<str>
- Parameter — {<str>|<block>}
- Response type — <str> = string
- Description — Sets the label.
  The label is set for the active channel.
  Number of characters to be set: 80

5. DISPlay[:WINDow[<chno>]]:TRACe:ASSign — IEEE488.1-1987 command mode DISP{DATA|MEM|DM}

- Function — ON/OFF of trace display
- Presence of command and query — Command / Query
- IEEE488.2-1987 command mode
  - Command — DISPlay[:WINDow[<chno>]]:TRACe:ASSign <type>
  - Parameter — <type> = {DATA | MEMory | DMEMory}
  - Response type — DATA | MEM | DMEM
- IEEE488.1-1987 command mode
  - Command — DISP{DATA|MEM|DM}
  - Response type — 0 | 1
- Description — Specifies the type of trace display.
  Initial setting DISPDATA

R3762/63 command	R3764/66, R3765/67 command parameter	Operation
DISPDATA	DATA	Displays the data trace only
DISPMEM	MEM	Displays the memory trace only
DISPDM	DMEM	Displays both the data trace and the memory trace

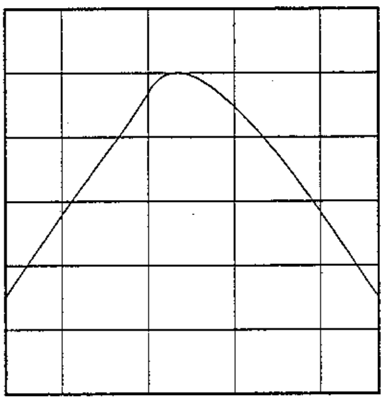
6. DISPlay[:WINDow[<chno>]]:TRACe:GRATicule[:STATe] IEEE488.1-1987 command mode GRAT

- Function: ON/OFF of graticule
- Presence of command and query: Command / Query
- Command: DISPlay[:WINDow[<chno>]]:TRACe:GRATicule[:STATe] <bool>
  GRAT<bool>
- Parameter: <bool>
- Response type: 0 | 1
- Description: Selects whether or not the graticule is displayed.
  Initial setting GRAT ON

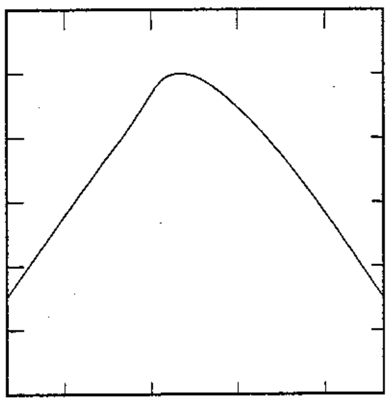
R3762/63 command	R3764/66, R3765/67 command parameter	Operation
GRAT ON	ON	Displays the graticule
GRAT OFF	OFF	Does not display the graticule

- Example

GRAT ON



GRAT OFF

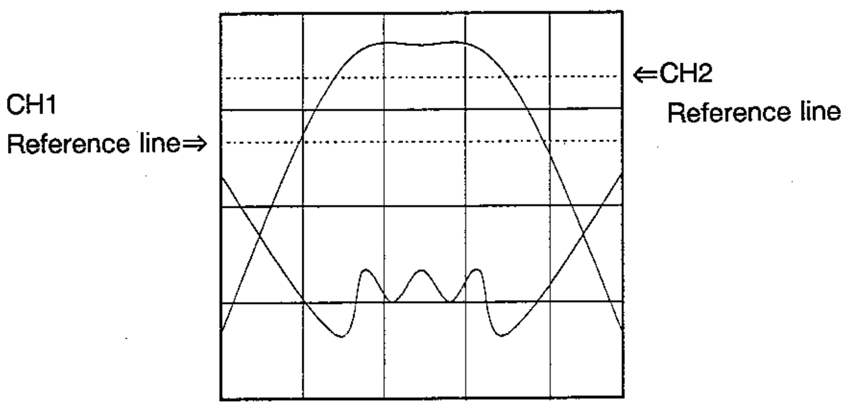


7. DISPlay[:WINDow[<chno>]]:Y[<trace>]:RLINe IEEE488.1-1987 command mode REFL

- Function: ON/OFF of Y-axis reference line display
- Presence of command and query: Command / Query
- Command: DISPlay[:WINDow[<chno>]]:Y[<trace>]:RLINe <bool>
  REFL<bool>
- Parameter: <bool>
- Response type: 0 | 1
- Description: Selects ON/OFF of the Y-axis reference line display.
  The Y-axis reference line indicates the reference value for the Y-axis graticule.
  Initial setting REFL ON

R3762/63 command	R3764/66, R3765/67 command parameter	Operation
REFL ON	ON	Displays the Y-axis reference line
REFL OFF	OFF	Does not display the Y-axis reference line

- Example



8.	DISPlay[:WINDow[<chno>]]:Y[<trace>][:SCALe]:AUTO	IEEE488.1-1987 command mode AUTO SCALF{1ST|2ND}

- Function: Y-axis automatic setting

- Presence of command and query: Command

- Command: DISPlay[:WINDow[<chno>]]:Y[<trace>][:SCALe]:AUTO
  ONCE
  AUTO
  SCALF{1ST|2ND}

- Parameter: ONCE

- Description: Automatically adjusts the Y-axis setting.
  The Y axis is set to an optimum value so that all the data which were displayed before the execution of this command fit into the scale screen. (Only the PDIV, RLEV setting is updated.)
  <trace> and SCALF{1ST|2ND} of IEEE488.1-1987 command mode are used to specify the trace whose scale is to be changed when the measurement format is set to 2 traces (MLOP, MLOD, MLIP). If the measurement format is not set to 2 traces, the specification will be ignored.

<trace>		
	=0 First waveform of CH1	SCALF1ST
	=1 First waveform of CH2	
	=4 First waveform of CH3	
	=5 First waveform of CH4	
	=8 Second waveform of CH1	SCALF2ND
	=9 Second waveform of CH2	
	=12 Second waveform of CH3	
	=13 Second waveform of CH4	

First waveform: LOGMAG when the display format is LOGMAG&PHASE and LOGMAG&DELAY,
LINMAG when it's LINMAG&PHASE,
S11 when the measure mode is S11&S21(FWD),
S22 when it's S22&S12(REV).

Second waveform: PHASE when the display format is LOGMAG&PHASE,
DELAY when it's LOGMAG&DELAY,
DELAY when it's LINMAG&DELAY,
S21 when the measure mode is S11&S21(FWD),
S12 when S22&S12(REV).

9. DISPlay[:WINDow[<chno>]]:Y[<trace>][:SCALe]:PDIVision   IEEE488.1-1987 command mode
SDIV
SCALF{1ST|2ND}

- Function: Y-axis grid scale setting
- Presence of command and query: Command / Query
- Command: DISPlay[:WINDow[<chno>]]:Y[<trace>][:SCALe]:PDIVision <real>
  SDIV<real>
  SCALF{1ST|2ND}
- Parameter: <real>
- Response type: NR3 (real value)
- Description: Sets the scale value of the Y-axis grid (scale per graticule).
  The command is ineffective in polar coordinate and Smith chart displays.
  <trace> and SCALF{1ST|2ND} of IEEE488.1-1987 command mode are used to specify the trace whose scale is to be changed when the measurement format is set to 2 traces (MLOP, MLOD, MLIP).
  If the measurement format is not set to 2 traces, the specification will be ignored.

<trace>	=0 First waveform of CH1	SCALF1ST
	=1 First waveform of CH2	
	=4 First waveform of CH3	
	=5 First waveform of CH4	
	=8 Second waveform of CH1	SCALF2ND
	=9 Second waveform of CH2	
	=12 Second waveform of CH3	
	=13 Second waveform of CH4	

First waveform: LOGMAG when the display format is LOGMAG&PHASE and LOGMAG&DELAY,
LINMAG when it's LINMAG&PHASE,
S11 when the measure mode is S11&S21(FWD),
S22 when it's S22&S12(REV).

Second waveform: PHASE when the display format is LOGMAG&PHASE,
DELAY when it's LOGMAG&DELAY,
DELAY when it's LINMAG&DELAY,
S21 when the measure mode is S11&S21(FWD),
S12 when S22&S12(REV).

The initial value depends on the measurement format.
See "A3. INITIAL SETTING".

10. DISPlay[:WINDow[<chno>]]:Y[<trace>][:SCALe]:RLEVel — IEEE488.1-1987 command mode: REFV, SCALF{1ST|2ND}

- Function: Y-axis reference level setting
- Presence of command and query: Command / Query
- Command: DISPlay[:WINDow[<chno>]]:Y[<trace>][:SCALe]:RLEVel <real>
  REFV<real>
  SCALF{1ST|2ND}
- Parameter: <real>
- Response type: NR3 (real value)
- Description:

Sets the level of the Y-axis reference line.
The Y-axis reference line indicates the reference value for the Y-axis graticule.
In polar coordinate and Smith chart displays, the value is set to the full-scale value on the outside circle.
<trace> and SCALF{1ST|2ND} of IEEE488.1-1987 command mode are used to specify the trace whose scale is to be changed when the measurement format is set to 2 traces (MLOP, MLOD, MLIP).
If the measurement format is not set to 2 traces, the specification will be ignored.

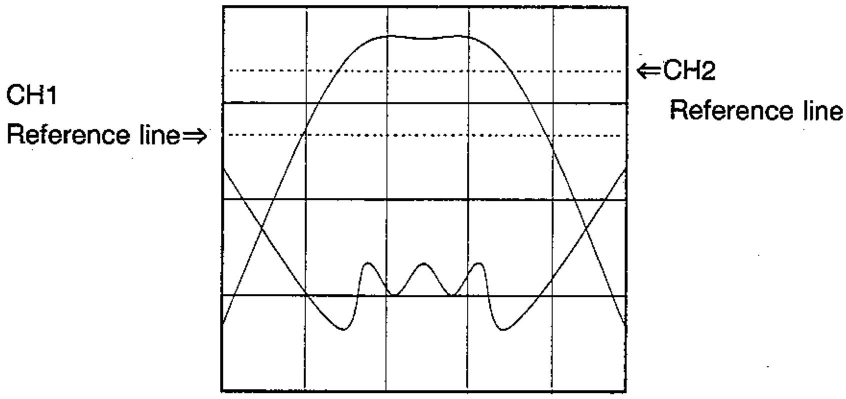
<trace>	=0 First waveform of CH1	SCALF1ST
	=1 First waveform of CH2	
	=4 First waveform of CH3	
	=5 First waveform of CH4	
	=8 Second waveform of CH1	SCALF2ND
	=9 Second waveform of CH2	
	=12 Second waveform of CH3	
	=13 Second waveform of CH4	

First waveform: LOGMAG when the display format is LOGMAG&PHASE and LOGMAG&DELAY,
LINMAG when it's LINMAG&PHASE,
S11 when the measure mode is S11&S21(FWD),
S22 when it's S22&S12(REV).

Second waveform: PHASE when the display format is LOGMAG&PHASE,
DELAY when it's LOGMAG&DELAY,
DELAY when it's LINMAG&DELAY,
S21 when the measure mode is S11&S21(FWD),
S12 When S22&S12(REV),

The initial value depends on the measurement format.
See "A3. INITIAL SETTING".

● Example



11.	DISPlay[:WINDow[<chno>]]:Y[<trace>][:SCALe]:RPOSition	IEEE488.1-1987 command mode REFP SCALF{1ST|2ND}

- Function — Y-axis reference line position specification
- Presence of command and query — Command / Query
- Command — DISPlay[:WINDow[<chno>]]:Y[<trace>][:SCALe]:RPOSition <real>
  REFP<real>
  SCALF{1ST|2ND}
- Parameter — <real> = 0 to 100
- Response type — NR3 (real value)
- Description — Specifies the position of the Y-axis reference line.
  <trace> and SCALF{1ST|2ND} of IEEE488.1-1987 command mode are used to specify the trace whose scale is to be changed when the measurement format is set to 2 traces (MLOP, MLOD, MLIP).
  If the measurement format is not set to 2 traces, the specification will be ignored.

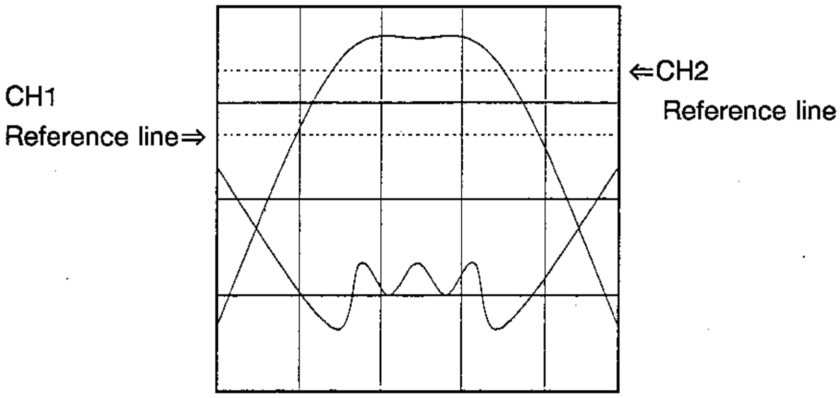
<trace>	= 0 First waveform of CH1	SCALF1ST
	= 1 First waveform of CH2	SCALF1ST
	= 4 First waveform of CH3	SCALF1ST
	= 5 First waveform of CH4	SCALF1ST
	= 8 Second waveform of CH1	SCALF2ND
	= 9 Second waveform of CH2	SCALF2ND
	= 12 Second waveform of CH3	SCALF2ND
	= 13 Second waveform of CH4	SCALF2ND

First waveform: LOGMAG when the display format is LOGMAG&PHASE and LOGMAG&DELAY,
LINMAG when it's LINMAG&PHASE,
S11 when the measure mode is S11&S21(FWD),
S22 when it's S22&S12(REV).

Second waveform: PHASE when the display format is LOGMAG&PHASE,
DELAY when it's LOGMAG&DELAY,
DELAY when it's LINMAG&DELAY,
S21 when the measure mode is S11&S21(FWD),
S12 when S22 & S12(REV)

The initial value depends on the measurement format. See "A3. INITIAL SETTING".
The value should be specified as a percentage, with 100% at the top of the screen, 50% in the middle, and 0% at the bottom.

• Example



7.6 FILE Subsystem

1.	FILE:DELete	IEEE488.1-1987 command mode PURGE

- Function — Deletion of a stored file
- Presence of command and query — Command
- IEEE488.2-1987 command mode
  - Command — FILE:DELete <str>
  - Parameter — <str> = File name
- IEEE488.1-1987 command mode
  - Command — PURGE<str>
  - Response type — <str> = File name
- Description — Deletes a file stored by the FILE:STORe command or the STFILE command.

2.	FILE:LOAD	IEEE488.1-1987 command mode LDFILE

- Function — Loading of a stored file

- Presence of command and query — Command

- IEEE488.2-1987 command mode
  - Command — FILE:LOAD <str>
  - Parameter — <str> = File name

- IEEE488.1-1987 command mode
  - Command — LDFILE<str>
  - Response type — <str> = File name

- Description — Loads a file stored by the FILE:STORe command or the STFILE command.

  If the specified file is stored when the FILE:STATe:RAW or the FILE:STATe:DATA is ON, the sweeping is forcibly in the hold mode after loading because the measured waveform data are also loaded.

3.	FILE:STATe:CONDition	IEEE488.1-1987 command mode DSSTATE

- Function — Definition of the conditions for the file to store
- Presence of command and query — Command / Query
- Command — FILE:STATe:CONDition <bool>
  DSSTATE <bool>
- Parameter — <bool>
- Response type — 0 | 1
- Description — Selects whether or not to store the setting conditions of the file by the FILE:STORe command.

4. FILE:STATe:CORRection — IEEE488.1-1987 command mode CORARY

- Function: Definition of the conditions for the file to store
- Presence of command and query: Command / Query
- Command: FILE:STATe:CORRection <bool>
  CORARY <bool>
- Parameter: <bool>
- Response type: 0 | 1
- Description: Selects whether or not to store the calibration data in the file by the FILE:STORe command.

5.	FILE:STATe:DATA	IEEE488.1-1987 command mode DATAARY

• Function	Definition of the conditions for the file to store
• Presence of command and query	Command / Query
• Command	FILE:STATe:DATA <bool> DATAARY <bool>
• Parameter	<bool>
• Response type	0 | 1
• Description	Selects whether or not to store the measured waveform data in the file by the FILE:STORe command.

6.	FILE:STATe:MEMory	IEEE488.1-1987 command mode MEMARY

- Function — Definition of the conditions for the file to store
- Presence of command and query — Command / Query
- Command — FILE:STATe:MEMory <bool>
  MEMARY <bool>
- Parameter — <bool>
- Response type — 0 | 1
- Description — Selects whether or not to store the memory waveform data in the file by the FILE:STORe command.

7. FILE:STATe:RAW — IEEE488.1-1987 command mode RAWARY

- Function — Definition of the conditions for the file to store
- Presence of command and query — Command / Query
- Command — FILE:STATe:RAW <bool>
  RAWARY <bool>
- Parameter — <bool>
- Response type — 0 | 1
- Description — Selects whether or not to store the raw data of the measured waveform in the file by the FILE:STORe command.

8.	FILE:STORe	IEEE488.1-1987 command mode STFILE

- Function — Storing the file

- Presence of command and query — Command

- IEEE488.2-1987 command mode
  - Command — FILE:STORe <str>
  - Parameter — <str> = File name

- IEEE488.1-1987 command mode
  - Command — STFILE<str>
  - Response type — <str> = File name

- Description — Setting conditions, calibration data, waveform data, etc. of this equipment can be stored to a floppy disk.

  The information to be stored is defined by the FILE:STATe command. For details, refer to FILE:STATe command.

7.7 FORMat Subsystem

1.	FORMat:BORDer	IEEE488.1-1987 command mode FORM{0|2|3|5|6|7|8}

- Function — Setting of byte order

- Presence of command and query — Command / Query

- IEEE488.2-1987 command mode
  - Command — FORMat:BORDer <border>
  - Parameter — <border> = {NORMal | SWAPped}
    - Response type — NORM | SWAP

- IEEE488.1-1987 command mode
  - Command — FORM{0|2|3|5|6|7|8}
  - Response type — None

- Description — The FORMat:BORDer(FORM{0|2|3|5|6|7|8}) command is used to set the data format to be input/output by the TRACe:DATA command. For detailed information on this command, see the description of the FORMat[:DATA] command.

  For details, see "2. FORMat[:DATA]".

2. FORMat[:DATA] — IEEE488.1-1987 command mode FORM{0|2|3|5|6|7|8}

- Function — Setting of data format

- Presence of command and query — Command / Query

- IEEE488.2-1987 command mode
  - Command — FORMat[:DATA] <format>,<len>
  - Parameter — <format> = {ASCii | REAL | MBINary}
    <len> = {32 | 64}
  - Response type — {ASC | REAL | MBIN},<int>
    <int> = NR1 (integer value)

- IEEE488.1-1987 command mode
  - Command — FORM{0|2|3|5|6|7|8}
  - Response type — None

- Description

The FORMat[:DATA] command is used in combination with the FORMat:BORDer command. Using these commands, the format of the trace data input/output using the TRACe:DATA command can be changed. (For IEEE488.1-1987 command mode, using the FORM {0|2|3|5|6|7|8} command, the input/output format of IN {1|2} etc or OT {1|2} etc can be changed.)

The format for data transfer using a combination of these commands is shown in the table below. If BORDer is set to NORMal, the data will be transferred in descending order from the highest byte. If it is set to SWAPped, the data will be transferred in ascending order from the lowest byte.

Note: If N88BASIC is used on an NEC personal computer, use the Microsoft floating-point format for the binary format.

FORM:DATA	FORM:BORD	
	NORMal	SWAPped
ASCii	ASCII(FORM0)	
REAL,32	IEEE 32bit binary(FORM2)	IEEE 32-bit binary order exchange (FORM5)
REAL,64	IEEE 64bit binary(FORM3)	IEEE 64-bit binary order exchange (FORM6)
MBIN,32	Microsoft single precision floating point binary (FORM7)	
MBIN,64	Microsoft double precision floating point binary (FORM8)	

7.8 INITiate Subsystem

1. INITiate:CONTinuous

- Function: ON/OFF of trigger system state
- Presence of command and query: Command / Query
- Command: INITiate:CONTinuous <bool>
- Parameter: <bool>
- Response type: 0 | 1
- Description: The INITiate:CONTinuous command controls the start of the trigger system.
  If CONTinuous is set to ON, the system does not return to the idle state and changes to the trigger waiting state.
  If CONTinuous is set to OFF, it changes to the trigger waiting state through the idle state. In this case, use the INITiate[:IMMediate] command to go to the trigger waiting state.

  For details, see "5. TRIGGER SYSTEM".

2. INITiate[:IMMediate]

- Function: Trigger system start
- Presence of command and query: Command
- Command: INITiate[:IMMediate]
- Description: The INITiate[:IMMediate] command starts the trigger system.

  The trigger system changes from the idle state to the trigger waiting state to wait for the occurrence of an event.

  For details, see "5. TRIGGER SYSTEM".

7.9 REGister Subsystem

1.	REGister:CLEar	IEEE488.1-1987 command mode CLRREG{1|2|3|4|5|6|7|8|9|10}

- Function — Clearing of the register

- Presence of command and query — Command

- IEEE488.2-1987 command mode
  - Command — REGister:CLEar <int>
  - Parameter — <int>

- IEEE488.2-1987 command mode
  - Command — CLRREG{1|2|3|4|5|6|7|8|9|10}

- Description — Clears the register data stored by the *SAV, the REGister:SAVE <int> or the SAVEREG{1|2|3|4|5|6|7|8|9|10}command.

2.

REGister:RECall	IEEE488.1-1987 command mode RECLREG{1|2|3|4|5|6|7|8|9|10}

- Function — Recalling (reading) the register

- Presence of command and query — Command

- IEEE488.2-1987 command mode
  - Command — REGister:RECall {<int>|POFF}
  - Parameter — <int> = Register number
    POFF = Setting before power off

- IEEE488.1-1987 command mode
  - Command — RECLREG{1|2|3|4|5|6|7|8|9|10}

- Description — Recalls the register data stored by the *SAV, the REGister:SAVE <int> or the SAVEREG{1|2|3|4|5|6|7|8|9|10}command.

  This command has the same function as the *RCL.

3.	REGister:SAVE	IEEE488.1-1987 command mode SAVEREG{1|2|3|4|5|6|7|8|9|10}

- Function — Saving of the register data

- Presence of command and query — Command

- IEEE488.2-1987 command mode
  - Command — REGister:SAVE <int>
  - Parameter — <int>

- IEEE488.1-1987 command mode
  - Command — SAVEREG{1|2|3|4|5|6|7|8|9|10}

- Description — Saves the setting conditions and the calibration data of this equipment into the specified number of register.

  This command has the same function as the *SAV.

7.10 SENSe Subsystem

1.	[SENSe:]AVERage[<chno>]:COUNt	IEEE488.1-1987 command mode AVERFACT AVR{2|4|8|16|32|64|128}

- Function — Setting of number of averaging times

- Presence of command and query — Command / Query

- IEEE488.2-1987 command mode

Command	[SENSe:]AVERage[<chno>]:COUNt <int>
Parameter	<int>
Response type	NR1 (integer value)

- IEEE488.1-1987 command mode

Command	AVERFACT<int> AVR{2|4|8|16|32|64|128}
Parameter	<int>
Response type	NR1 (AVERFACT command) 0 | 1 (AVR{2|4|8|16|32|64|128} command)

- Description — Sets the number of averaging times.

The averaging averages the data by adding time weight to the measured data before formatting. Since this method averages the data in accordance with the vector quantity, the noise level can be reduced.

The averaging process is as follows:

$$\bar{Y}_{(n)} = \frac{n-1}{n} \cdot \bar{Y}_{(n-1)} + \frac{1}{n} \cdot Y_{(n)} \qquad (n \leqq N)$$

$$\bar{Y}_{(n)} = \frac{N-1}{N} \cdot \bar{Y}_{(n-1)} + \frac{1}{N} \cdot Y_{(n)} \qquad (n > N)$$

$\bar{Y}_{(n)}$: nth averaged data
$Y_{(n)}$: nth data
N: Number of averaging times

2. [SENSe:]AVERage[<chno>]:RESTart — IEEE488.1-1987 command mode AVERREST

- Function: Averaging restart
- Presence of command and query: Command
- Command:
  [SENSe:]AVERage[<chno>]:RESTart
  AVERREST
- Description:
  Clears the average counter and restarts the averaging.
  The averaging averages the data by adding time weight to the measured data before formatting. Since this method averages the data in accordance with the vector quantity, the noise level can be reduced.

  The averaging process is as follows:

$$\bar{Y}_{(n)} = \frac{n-1}{n} \cdot \bar{Y}_{(n-1)} + \frac{1}{n} \cdot Y_{(n)} \qquad (n \leqq N)$$

$$\bar{Y}_{(n)} = \frac{N-1}{N} \cdot \bar{Y}_{(n-1)} + \frac{1}{N} \cdot Y_{(n)} \qquad (n > N)$$

$\bar{Y}_{(n)}$: nth averaged data
$Y_{(n)}$: nth data
N: Number of averaging times

3. [SENSe:]AVERage[<chno>][:STATe] IEEE488.1-1987 command mode
AVERAGE
AVER

- Function: ON/OFF of averaging
- Presence of command and query: Command / Query
- Command: [SENSe:]AVERage[<chno>][:STATe] <bool>
  AVERAGE
  AVER<bool>
- Parameter: <bool>
- Response type: 0 | 1
- Description: Sets ON/OFF of the averaging.

Initial setting OFF

The averaging averages the data by adding time weight to the measured data before formatted. Since this method averages the data in accordance with the vector quantity, the noise level can be reduced.

The averaging process is as follows:

$$\bar{Y}_{(n)} = \frac{n-1}{n} \cdot \bar{Y}_{(n-1)} + \frac{1}{n} \cdot Y_{(n)} \qquad (n \leqq N)$$

$$\bar{Y}_{(n)} = \frac{N-1}{N} \cdot \bar{Y}_{(n-1)} + \frac{1}{N} \cdot Y_{(n)} \qquad (n > N)$$

$\bar{Y}_{(n)}$: nth averaged data
$Y_{(n)}$: nth data
N: Number of averaging times

AVERAGE of R3762/63 command is identical to AVER OFF.

- Note: Smoothing obtains the moving average between adjacent formatted data. Since the method averages the scalar quantity, it reduces the noise width but does not reduce the noise level.

4.	[SENSe:]BANDwidth[<chno>][:RESolution]	IEEE488.1-1987 command mode RBW RBW{1K|300|100|30|10}HZ

- Function — Bandwidth setting

- Presence of command and query — Command / Query

- IEEE488.2-1987 command mode
  - Command — [SENSe:]BANDwidth[<chno>][:RESolution] <int>
  - Parameter — <int>
  - Response type — NR1 (integer value)

- IEEE488.1-1987 command mode
  - Command — RBW<int>
    RBW{1K|300|100|30|10}HZ
  - Parameter — <int>
  - Response type — NR1 (RBW command)
    0 | 1 (RBW{1K|300|100|30|10}HZ command)

- Description — Sets the resolution bandwidth of the receiver.

  Initial setting 10kHz

  The resolution bandwidth can be selected in the range 10kHz to 3Hz, as shown below. The maximum sweeping speed and noise level per point depend on the resolution bandwidth selected.

Resolution bandwidth	Maximum sweeping speed per point
10kHz	0.1ms/POINT
3kHz	0.35ms/POINT
1kHz	1.0ms/POINT
300Hz	3.5ms/POINT
100Hz	10ms/POINT
30Hz	35ms/POINT
10Hz	100ms/POINT
3Hz	350ms/POINT

- Note — If the resolution bandwidth is set to 10kHz, 3kHz, or 3Hz in IEEE488.1-1987 command mode, the setting must be performed and the query must be made by using an RBW command.

5. [SENSe:]BANDwidth[<chno>][:RESolution]:AUTO   IEEE488.1-1987 command mode RBWAUTO

- Function: Automatic bandwidth setting
- Presence of command and query: Command / Query
- Command: [SENSe:]BANDwidth[<chno>][:RESolution]:AUTO <bool>
  RBWAUTO
- Parameter: <bool>
- Response type: 0 | 1
- Description: Automatically sets the resolution bandwidth in accordance with the measurement frequency.

  The maximum sweeping speed and noise level per point depend on the resolution bandwidth selected.

Resolution bandwidth	Maximum sweeping speed per point
10kHz	0.1ms/POINT
3kHz	0.35ms/POINT
1kHz	1.0ms/POINT
300Hz	3.5ms/POINT
100Hz	10ms/POINT
30Hz	35ms/POINT
10Hz	100ms/POINT
3Hz	350ms/POINT

- Note: The maximum sweeping speed per point depends on the resolution bandwidth. Since at particularly low frequencies the resolution bandwidth is low and the sweeping speed is reduced, do not set the frequency too low.

6. [SENSe:]CORRection[<chno>]:COLLect[:ACQuire] — IEEE488.1-1987 command mode
NORM,NORMS
OPEN,SHORT,LOAD
S11OPEN,S11SHORT,S11LOAD,
S22OPEN,S22SHORT,S22LOAD,
FWDTRNS,FWDMATCH,
REVTRNS,REVMATCH,
OMITISO,FWDISO,REVISO

- Function — Calibration data acquisition

- Presence of command and query — Command

- Command — [SENSe:]CORRection[<chno>]:COLLect[:ACQuire] <standard>
{NORM| SNOR},S11O,S11S,S11L,S22O,S22S,S22L,FTR,
FMAT,RTR,RMAT,GTHRU,OIS,FIS,RIS <bool>
OPEN,SHORT,LOAD,S11OPEN,S11SHORT,S11LOAD,
S22OPEN,S22SHORT,S22LOAD,FWDTRNS,FWDMATCH,
REVTRNS,REVMATCH,OMITISO,FWDISO,REVISO

- Parameter — <standard> = {NORMalize | SNORmalize | OPEN | SHORt | LOAD}

- Description — Acquires the calibration data.

This command restarts the sweeping and acquires the calibration data.

If the averaging function is set to ON, the calibration data are acquired after the sweeping has been repeated the number of times specified.

If the calibration data have already been acquired, the data will be updated. However, when one-port full calibration and two-port full calibration are in progress, the data cannot be updated. In this case, the data should be cleared then updated.

R3762/63 command	R3764/66, R3765/67 command parameter	Operation (acquired data)
NORM ON	NORM	Normalize: Acquired and finished simultaneously
NORMS ON	SNOR	Short normalize: Acquired and finished simultaneously
OPEN	OPEN	One-port full calibration Open data
SHORT	SHOR	One-port full calibration Short data
LOAD	LOAD	One-port full calibration Load data
S11OPEN	S11O	Two-port full calibration Open data (S11)
S11SHORT	S11S	Two-port full calibration Short data (S11)
S11LOAD	S11L	Two-port full calibration Load data (S11)
S22OPEN	S22O	Two-port full calibration Open data (S22)
S22SHORT	S22S	Two-port full calibration Short data (S22)
S22LOAD	S22L	Two-port full calibration Load data (S22)
FWDTRNS	FTR	Two-port full calibration Forward direction through characteristic data
FWDMATCH	FMAT	Two-port full calibration Forward direction port matching characteristic data
REVTRNS	RTR	Two-port full calibration Reverse direction through characteristic data
REVMATCH	RMAT	Two-port full calibration Reverse direction port matching characteristic data
---	GTHRU	Two-port full calibration Acquires the above four (transmission characteristics) together.
OMITISO	OIS	Two-port full calibration Isolation data (OMIT)
FWDISO	FIS	Two-port full calibration Isolation data (Forward)
REVISO	RIS	Two-port full calibration Isolation data (Reverse)

7. [SENSe:]CORRection[<chno>]:COLLect:DELete — IEEE488.1-1987 command mode: CLEAR

- Function: Calibration data clearing
- Presence of command and query: Command
- Command: [SENSe:]CORRection[<chno>]:COLLect:DELete
  CLEAR
- Description: Clears the calibration data.

  For one-port full calibration and two-port full calibration, once the calibration has finished, it is impossible to acquire the data again until the data have been cleared. Therefore, to acquire the calibration data again, the data should be cleared.
  Note that if the calibration data are to be cleared, the correction measurement should be set to OFF.

8. [SENSe:]CORRection[<chno>]:COLLect:SAVE — IEEE488.1-1987 command mode: DONE, DONE1PORT, DONE2PORT

- Function: Calculation of error coefficient from calibration data
- Presence of command and query: Command
- Command: [SENSe:]CORRection[<chno>]:COLLect:SAVE
  DONE
  DONE1PORT
  DONE2PORT
- Description: Calculates the error coefficient from the calibration data acquired and sets the correction measurement function to ON.

9.	[SENSe:]CORRection[<chno>]:CSET:STATe	IEEE488.1-1987 command mode

- Function — ON/OFF of correction measurement

- Presence of command and query — Command / Query

- Command — [SENSe:]CORRection[<chno>]:CSET:STATe <bool>
  CORRECT<bool>

- Parameter — <bool>

- Response type — 0 | 1

- Description — Selects ON/OFF of correction measurement using the calibration data.

  If the calibration data have already been gained, this command should be used to perform the correction measurement. Since the stored calibration data are not cleared when this command is set to OFF, it is possible to perform the correction measurement by setting the command to ON at any time.

10. [SENSe:]CORRection[<chno>]:EDELay:DISTance IEEE488.1-1987 command mode LENGTH1987

- Function: Electrical length (distance) setting
- Presence of command and query: Command / Query
- Command: [SENSe:]CORRection[<chno>]:EDELay:DISTance <real>
  LENGVAL<real>
- Parameter: <real>
- Response type: NR3 (real value)
- Description: Sets the value of the electrical length correction by inputting the distance.

Correction value $\phi(\text{deg}) = \frac{L}{c} \times \frac{1}{V_f} \times f \times 360$

$= S \times f \times 360$

L: Electrical length (distance)
V_f: Transfer constant
c: Velocity of light
f: Frequency
S: Electrical length (time)

11. [SENSe:]CORRection[<chno>]:EDELay:STATe   IEEE488.1-1987 command mode LENGTH

- Function: ON/OFF of electrical length correction
- Presence of command and query: Command / Query
- Command: [SENSe:]CORRection[<chno>]:EDELay:STATe <bool>
  LENGTH<bool>
- Parameter: <bool>
- Response type: 0 | 1
- Description: Selects ON/OFF of the electrical length correction.

Corrects the phase variation of the measurement data in accordance with the electrical length already set.
This command is used to remove the phase variation of the connection cable so that only the phase variation of the object can be measured.

$$\text{Correction value} \quad \phi(\text{deg}) = \frac{L}{c} \times \frac{1}{V_f} \times f \times 360$$
$$= S \times f \times 360$$

L: Electrical length (distance)
V_f: Transfer constant
c: Velocity of light
f: Frequency
S: Electrical length (time)

12. [SENSe:]CORRection[<chno>]:EDELay[:TIME] IEEE488.1-1987 command mode ELED

- Function: Electrical length (time) setting
- Presence of command and query: Command / Query
- Command: [SENSe:]CORRection[<chno>]:EDELay[:TIME] <real>
  ELED<real>
- Parameter: <real>
- Response type: NR3 (real value)
- Description: Sets the value of the electrical length in time.

Correction value $\phi(\mathrm{deg}) = \frac{L}{c} \times \frac{1}{V_f} \times f \times 360$

$= S \times f \times 360$

L: Electrical length (distance)
V_f: Transfer constant
c: Velocity of light
f: Frequency
S: Electrical length (time)

13. [SENSe:]CORRection[n]:GPHase:STATe   IEEE488.1-1987 command mode
SRCCOR

- Function: ON/OFF of frequency characteristic calibration in the receiving part.

- Presence of command and query: Command / Query

- IEEE488.2-1987 command mode
  - Command: [SENSe:]CORRection[n]:GPHase:STATe <bool>
  - Parameter: <bool>
  - Response type: 0 | 1

- IEEE488.1-1987 command mode
  - Command: INPCOR<bool>
  - Parameter: <bool>
  - Response type: 0 | 1

- Description: Selects whether or not the frequency characteristics in the receiving part are to be calibrated. (ON or OFF)

14. [SENSe:]CORRection[<chno>]:OFFSet:PHASe IEEE488.1-1987 command mode PHAO

- Function — Phase offset value setting
- Presence of command and query — Command / Query
- Command — [SENSe:]CORRection[<chno>]:OFFSet:PHASe <real>
  PHAO<real>
- Parameter — <real>
- Response type — NR3 (real value)
- Description — Sets the value of the phase offset.

  A constant value is added to the phase data. Unlike the electrical length correction, the command always corrects the set value, regardless of the frequency.
- Note — If 0 is set, CORR:OFFS:STAT is automatically set to OFF.
  If the value other than 0 is set, CORR:OFFS:STAT is automatically set to ON.

15. [SENSe:]CORRection[<chno>]:OFFSet:STATe — IEEE488.1-1987 command mode PHAOFS

• Function	ON/OFF of phase offset function
• Presence of command and query	Command / Query
• Command	[SENSe:]CORRection[<chno>]:OFFSet:STATe <bool> PHAOFS<bool>
• Parameter	<bool>
• Response type	0 | 1
• Description	Selects ON/OFF of the phase offset function. A constant value is added to the phase data. Unlike the electrical length correction, the command always corrects the set value, regardless the frequency.
• Note	If OFF is set, CORR:OFFS:PHAS is automatically set to 0.

16.	[SENSe:]CORRection[<chno>]:PEXTension:TIME[<eport>]	IEEE488.1-1987 command mode EPORT{R|A|B|1|2}

- Function — Setting of extension correction value of measurement end face
- Presence of command and query — Command / Query
- Command — [SENSe:]CORRection[<chno>]:PEXTension:TIME[<eport>] <real>
  EPORT{R|A|B|1|2} <real>
- Parameter — <real>
- Response type — NR3 (real value)
- Description — Sets the extension value of the measurement end face.
  The command corrects the extension in accordance with the input port. While the electrical correction simply corrects the set value, this command corrects in accordance with the input port condition by setting the value corresponding to the input port.
  For example, this command automatically sets the correction value to two times the port extension value for reflection measurement and to one time the port extension value for transfer measurement.

17. [SENSe:]CORRection[<chno>]:PEXTension:STATe — IEEE488.1-1987 command mode PORE

- Function: ON/OFF of extension calibration of measurement end face
- Presence of command and query: Command / Query
- Command: [SENSe:]CORRection[<chno>]:PEXTension:STATe <bool>
  PORE<bool>
- Parameter: <bool>
- Response type: 0 | 1
- Description: Selects ON/OFF of the extension calibration function of the measurement end face.
  The command calibrates the extension in accordance with the input port. While the electrical calibration simply calibrates the set value, this command calibrates in accordance with the input port condition by setting the value corresponding to the input port.
  For example, this command automatically sets the calibration value to two times the port extension value for reflection measurement and to one time the port extension value for transfer measurement.

18. [SENSe:]CORRection[<chno>]:RVELocity:COAX — IEEE488.1-1987 command mode VELOFACT

- Function — Cable transfer coefficient setting
- Presence of command and query — Command / Query
- Command — [SENSe:]CORRection[<chno>]:RVELocity:COAX <real>
  VELOFACT <real>
- Parameter — <real>
- Response type — NR3 (real value)
- Description — Sets the cable transfer coefficient value.

Calibration quantity $\phi(\mathrm{deg}) = \frac{L}{c} \times \frac{1}{V_f} \times f \times 360$

$$= S \times f \times 360$$

$$V_f = \frac{L}{\sqrt{\varepsilon_R}}$$

l: Electrical length (distance)
V_f: Transfer constant
c: Velocity of light
f: Frequency
S: Electrical length (time)
ε_R: Dielectric constant

19. [SENSe:]FUNCtion[<chno>][:ON] IEEE488.1-1987 command mode {R|A|B|AR|BR|AB|BDC|BDCR}IN S11,S12,S21,S22,SFWD,SREV, SMEAS

- Function: Specification of the measure mode and ON/OFF of the sub measure mode

- Presence of command and query: Command / Query

- IEEE488.2-1987 command mode
  - Command: [SENSe:]FUNCtion[<chno>][:ON] <input>
  - Parameter: <input> = {"POWer:{AC | DC} {1 | 2 | 3}" |
    "POWer:{AC | DC}:RATio {2,1 | 3,1 | 2,3}" |
    "POWer:{S11 | S12 | S22 | S21 | SFWD | SREV}" |
    "POWer:NONE" }
  - Response type: "POW:AC | DC} {1 | 2 | 3}" |
    "POW:AC | DC}:RAT {2,1 | 3,1 | 3,2}" |
    "POW:{S11 | S12 | S22 | S21 | SFWD | SREV}" |
    "POW:NONE"

- IEEE488.1-1987 command mode
  - Command: {R|A|B|AR|BR|AB|BDC|BDCR}IN
    S11,S12,S21,S22,SFWD,SREV
    SMEAS<bool>
  - Response type: 0 | 1

- Description: Specifies the measure mode for measurement/analysis, and switches the sub measure's ON/OFF.

In IEEE488.2-1987 command mode, specifies the measure mode by specifying the channel by <chno>. Specifying 3 or 4 for <chno> when the sub measure is OFF, the sub measure becomes ON.
Mode setting of the sub measure is performed by specifying 3 or 4 for <chno>, or after setting the active channel to 3 or 4. When the sub measure is changed to OFF, sets the active channel to 3 or 4, or specifies 3 or 4 for <chno> and sets the parameter "POW:NONE."
Then the active channel is switched to the corresponding main channel.

In IEEE488.1-1987 command mode, the setting is performed to the active channel. The setting must be performed after switching the active channel.

To set the sub measure to ON, sends SMEASON. Then the sub measure mode becomes the same as the corresponding main measure mode, and the active channel is switched to the sub channel.

To set the sub measure to OFF, sends SMEASOFF. The active channel is switched to the corresponding main channel.

(Note) When the sub measure is OFF, the sub channel cannot be switched to active.

R3762/63 command	R3764/66, R3765/67 command parameter	Operation (input port)
RIN	POW:AC 1	Sets R input
AIN	POW:AC 2	Sets A input
BIN	POW:AC 3	Sets B input
ARIN	POW:AC:RAT 2,1	Sets A/R input (ratio measurement)
BRIN	POW:AC:RAT 3,1	Sets B/R input (ratio measurement)
ABIN	POW:AC:RAT 2,3	Sets A/B input (ratio measurement)
BDCIN	POW:DC 3	Sets B (DC) input (DC measurement)
BDCRIN	POW:DC:RAT 3,1	Sets B (DC)/R input (ratio measurement)
S11	POW:S11	Sets S11
S12	POW:S12	Sets S12
S21	POW:S21	Sets S21
S22	POW:S22	Sets S22
SFWD	POW:SFWD	Sets S11 & S21 (REFL&TRANS)
SREV	POW:SREV	Sets S22 & S12
SMEASON	Specifies 3 or 4 for <chno>.	Sets the sub measure to ON.
SMEASOFF	POW:NONE	Sets the sub measure to OFF.

Refer to "7.5.1 DISPlay:ACTive", too.

20. [SENSe:]FUNCtion[<chno>]:POWer — IEEE488.1-1987 command mode {R|A|B|AR|BR|AB|BDC|BDCR}IN S11,S12,S21,S22,SFWD,SREV, SMEAS

- Function — Measure mode specification and ON/OFF of sub measure

- Presence of command and query — Command / Query

- IEEE488.2-1987 command mode
  - Command: [SENSe:]FUNCtion[<chno>]:POWer <input>
  - Parameter: <input> = {R | A | B | AR | BR | AB | BDC | BDCR | S11 | S12 | S21 | S22 | SFWD | SREV | NONE}
  - Response type: R | A | B | AR | BR | AB | BDC | BDCR | S11 | S12 | S21 | S22 | SFWD | SREV | NONE

- IEEE488.1-1987 command mode
  - Command: {R|A|B|AR|BR|AB|BDC|BDCR}IN
    S11,S12,S21,S22,SFWD,SREV
    SMEAS<bool>
  - Response type: 0 | 1

- Description

Specifies the measure mode for measurement/analysis, and switches the sub measure's ON/OFF.

In IEEE488.2-1987 command mode, specifies the measure mode by specifying the channel by <chno>. Specifying 3 or 4 for <chno> when the sub measure is OFF, the sub measure becomes ON.
Mode setting of the sub measure is performed by specifying 3 or 4 for <chno>, or after setting the active channel to 3 or 4.
When the sub measure is changed to OFF, sets the active channel to 3 or 4, or specifies 3 or 4 for <chno> and sets the parameter NONE.
Then the active channel is switched to the corresponding main channel.
In IEEE488.1-1987 command mode, the setting is performed to the active channel. The setting must be performed after switching the active channel.
To set the sub measure to ON, sends SMEASON. Then the sub measure mode becomes the same as the corresponding main measure mode, and the active channel is switched to the sub channel.
To set the sub measure to OFF, sends SMEASOFF. The active channel is switched to the corresponding main channel.

(Note) When the sub measure is OFF, the sub channel cannot be switched to active.

R3762/63 command	R3764/66, R3765/67 command parameter	Operation (input port)
RIN	R	Sets R input
AIN	A	Sets A input
BIN	B	Sets B input
ARIN	AR	Sets A/R input (ratio measurement)
BRIN	BR	Sets B/R input (ratio measurement)
ABIN	AB	Sets A/B input (ratio measurement)
BDCIN	BDC	Sets B (DC) input
BDCRIN	BDCR	Sets B (DC)/R input
S11	S11	Sets S11
S12	S12	Sets S12
S21	S21	Sets S21
S22	S22	Sets S22
SFWD	SFWD	Sets S11 & S21 (REFL&TRANS)
SREV	SREV	Sets S22 & S12
SMEASON	Specifies 3 or 4 for <chno>.	Sets the sub measure to ON.
SMEASOFF	NONE	Sets the sub measure to OFF.

Refer to "7.5.1 DISPlay:ACTive", too.

7.11 SOURce Subsystem

1. [SOURce:]CORRection[n]:GAIN:STATe — IEEE488.1-1987 command mode SRCCOR

- Function: ON/OFF of frequency characteristic calibration in the signal source part.

- Presence of command and query: Command / Query

- IEEE488.2-1987 command mode
  - Command: [SOURce:]CORRection[n]:GAIN:STATe <bool>
  - Parameter: <bool>
  - Response type: 0 | 1

- IEEE488.1-1987 command mode
  - Command: SRCCOR<bool>
  - Parameter: <bool>
  - Response type: 0 | 1

- Description: Selects whether or not the frequency characteristics in the signal source part are to be calibrated. (ON or OFF)

2. [SOURce:]COUPle IEEE488.1-1987 command mode COUPLE

- Function: ON/OFF of connecting channels for output signal
- Presence of command and query: Command / Query
- Command: [SOURce:]COUPle <bool>
  COUPLE<bool>
- Parameter: <bool>
- Response type: 0 | 1
- Description: Selects whether or not the same measurement conditions are to be used for measurement channels 1 and 2.

  Initial setting: COUPLE ON

  The measurement conditions include:
  - Sweeping type
  - Frequency
  - Output level
  - Sweeping time
  - Number of points for measurement
  - Resolution bandwidth

  If the command is set to COUPLE OFF, it measures measurement channel 1 first then measurement channel 2. In other words, it measures channel 1 and 2 alternately.
  When the sub measure is selected, channel 3 and channel 1, and channel 4 and channel 2 are always measured simultaneously regardless of COUPLE ON/OFF.

  If the command is set to COUPLE ON, channel 1 and channel 2 are measured simultaneously.
  When the sub measure is selected, the four screens are measured simultaneously.

3.	[SOURce:]FREQuency[<chno>]:CENTer	IEEE488.1-1987 command mode CENTERF

- Function — Central frequency setting
- Presence of command and query — Command / Query
- Command — [SOURce:]FREQuency[<chno>]:CENTer <real>
  CENTERF<real>
- Parameter — <real>
- Response type — NR3 (real value)
- Description — Sets the central frequency when the frequency is swept.

Initial setting	1.92GHz (R3764/66) 4.02GHz (R3765/67)
Setting range	20MHz to 3.8GHz (R3764/66) 20MHz to 8.0GHz (R3765/67)
Setting resolution	1Hz

4.	[SOURce:]FREQuency[<chno>]:CW	IEEE488.1-1987 command mode CWFREQ

- Function — Fixed frequency setting
- Presence of command and query — Command / Query
- Command — [SOURce:]FREQuency[<chno>]:CW <real>
  CWFREQ<real>
- Parameter — <real>
- Response type — NR3 (real value)
- Description — Sets the frequency for level sweeping.

Initial setting	1GHz (R3764/66) 1GHz (R3765/67)
Setting range	20MHz to 3.8GHz (R3764/66) 20MHz to 8.0GHz (R3765/67)
Setting resolution	1Hz

5. [SOURce:]FREQuency[<chno>]:MODE — IEEE488.1-1987 command mode LINFREQ LOGFREQ

- Function — Sweeping type setting
- Presence of command and query — Command / Query
- IEEE488.2-1987 command mode
  - Command — [SOURce:]FREQuency[<chno>]:MODE <mode>
  - Parameter — <mode> = SWEep
  - Response type — CW | SWE | PSW
- IEEE488.1-1987 command mode
  - Command — LINFREQ
    LOGFREQ
  - Response type — 0 | 1
- Description — This command must be set by combining each item as shown in the table below:
  Initial setting Linear frequency sweeping

Command	PSW: MODE	FREQ: MODE	POW: MODE	SWE: SPAC	Sweeping type	Corresponding R3762/63 command
Parameter	(NONE)	SWE	(FIX)	LIN	Linear frequency sweeping	LINFREQ
				LOG	Log frequency sweeping	LOGFREQ
		(CW)	SWE	(LIN)	Level sweeping	LEVEL
	FREQ	(PSW)	(FIX)	(LIN)	Program sweeping (frequency only)	USRFSWP
	ALL	(PSW)	(PSW)	(LIN)	Program sweeping	USRARWP

Note: The value in parentheses indicates the value which is returned for a query. Do not use this value for setting.

Sweeping type	Linear frequency sweeping:	Sweeps the frequency at a constant interval at a fixed level.
	Log frequency sweeping:	Sweeps the frequency at a log interval at a fixed level.
	Level sweeping:	Sweeps the output level at a fixed frequency.
	Program sweeping (frequency only):	Arbitrarily sets the frequency only for each interval.
	Program sweeping:	Arbitrarily sets the frequency, the output level, the resolution bandwidth and the settling time for each interval.

However, the log frequency sweeping cannot be set for R3764, R3766.

6. [SOURce:]FREQuency[<chno>]:SPAN — IEEE488.1-1987 command mode SPANF

- Function: Span frequency setting
- Presence of command and query: Command / Query
- Command: [SOURce:]FREQuency[<chno>]:SPAN <real>
  SPANF<real>
- Parameter: <real>
- Response type: NR3 (real value)
- Description: Sets the span frequency for frequency sweeping.

Initial setting	3.76GHz (R3764/66) 7.96GHz (R3765/67)
Setting range	0 to 3.78GHz (R3764/66) 0 to 7.98GHz (R3765/67)
Setting resolution	1Hz

7. [SOURce:]FREQuency[<chno>]:STARt — IEEE488.1-1987 command mode STARTF

- Function: Start frequency setting
- Presence of command and query: Command / Query
- Command: [SOURce:]FREQuency[<chno>]:STARt <real>
  STARTF<real>
- Parameter: <real>
- Response type: NR3 (real value)
- Description: Sets the start frequency for frequency sweeping.

Initial setting	40MHz (R3764/66) 40MHz (R3765/67)
Setting range	20MHz to 3.8GHz (R3764/66) 20MHz to 8.0GHz (R3765/67)
Setting resolution	1Hz

8. [SOURce:]FREQuency[<chno>]:STOP   IEEE488.1-1987 command mode
STOPF

- Function: Stop frequency setting
- Presence of command and query: Command / Query
- Command: [SOURce:]FREQuency[<chno>]:STOP <real>
  STOPF<real>
- Parameter: <real>
- Response type: NR3 (real value)
- Description: Sets the stop frequency for frequency sweeping.

Initial setting	3.8GHz (R3764/66) 8.0GHz (R3765/67)
Setting range	20MHz to 3.8GHz (R3764/66) 20MHz to 8.0GHz (R3765/67)
Setting resolution	1Hz

9. [SOURce:]POWer[<chno>][:LEVel][:AMPLitude]   IEEE488.1-1987 command mode
OUTLEV

- Function: Output level setting
- Presence of command and query: Command / Query
- Command: [SOURce:]POWer[<chno>][:LEVel][:AMPLitude]<real>
  OUTLEV<real>
- Parameter: <real>
- Response type: NR3 (real value)
- Description: Sets the output level for frequency sweeping.
  Setting resolution 0.01dB

	Initial setting	Setting range	
		SRC COR ON	SRC COR OFF
A type	0dBm	-13dBm to +17dBm	-16dBm to +24.95dBm
B type	0dBm	-15dBm to +15dBm	-13dBm to +22.95dBm
C type (A type + S parameter)	10dBm	-20dBm to +15dBm	-23dBm to +17.95dBm

10. [SOURce:]POWer[<chno>]:MODE   IEEE488.1-1987 command mode
LEVEL

- Function   Sweeping type setting

- Presence of command and query   Command / Query

- IEEE488.2-1987 command mode
  - Command   [SOURce:]POWer[<chno>]:MODE <mode>
  - Parameter   <mode> = {SWEep}
  - Response type   FIX | SWE | PSW

- IEEE488.1-1987 command mode
  - Command   LEVEL
  - Response type   0 | 1

- Description   This command must be set by combining each item as shown in the table below:
  Initial setting   Linear frequency sweeping

Command	PSW: MODE	FREQ: MODE	POW: MODE	SWE: SPAC	Sweeping type	Corresponding R3762/63 command
Parameter	(NONE)	SWE	(FIX)	LIN	Linear frequency sweeping	LINFREQ
				LOG	Log frequency sweeping	LOGFREQ
		(CW)	SWE	(LIN)	Level sweeping	LEVEL
	FREQ	(PSW)	(FIX)	(LIN)	Program sweeping (frequency only)	USRFSWP
	ALL	(PSW)	(PSW)	(LIN)	Program sweeping	USRARWP

Note: The value in parentheses indicates the value which is returned for a query. Do not use this value for setting.

Sweeping type
- Linear frequency sweeping: Sweeps the frequency at a constant interval at a fixed level.
- Log frequency sweeping: Sweeps the frequency at a log interval at a fixed level.
- Level sweeping: Sweeps the output level at a fixed frequency.
- Program sweeping (frequency only): Arbitrarily sets the frequency only for each interval.
- Program sweeping: Arbitrarily sets the frequency, the output level, the resolution bandwidth and the settling time for each interval.

However, the log frequency sweeping cannot be set for R3764/66.

11. [SOURce:]POWer[<chno>]:STARt — IEEE488.1-1987 command mode STLEVEL

- Function: Start level setting
- Presence of command and query: Command / Query
- Command: [SOURce:]POWer[<chno>]:STARt <real>
  STLEVEL<real>
- Parameter: <real>
- Response type: NR3 (real value)
- Description: Sets the start level for level sweeping.

	Initial setting		Setting range	
	Start	Stop	SRC COR ON	SRC COR OFF
A type	-13dBm	0dBm	-13dBm to +17dBm	-16dBm to +24.95dBm
B type	-15dBm	0dBm	-15dBm to +15dBm	-13dBm to +22.95dBm
C type (A type + S parameter)	-20dBm	0dBm	-20dBm to +15dBm	-23dBm to +17.95dBm

Setting resolution 0.01dB

12. [SOURce:]POWer[<chno>]:STOP — IEEE488.1-1987 command mode SPLEVEL

- Function: Stop level setting
- Presence of command and query: Command / Query
- Command: [SOURce:]POWer[<chno>]:STOP <real>
  SPLEVEL<real>
- Parameter: <real>
- Response type: NR3 (real value)
- Description: Sets the stop level for level sweeping.

	Initial setting		Setting range	
	Start	Stop	SRC COR ON	SRC COR OFF
A type	-13dBm	0dBm	-13dBm to +17dBm	-16dBm to +24.95dBm
B type	-15dBm	0dBm	-15dBm to +15dBm	-13dBm to +22.95dBm
C type (A type + S parameter)	-20dBm	0dBm	-20dBm to +15dBm	-23dBm to +17.95dBm

Setting resolution 0.01dB

13. [SOURce:]PSWeep[<chno>]:BANDwidth[<n>] IEEE488.1-1987 command mode
USEG
URBW

- Function: Inputting of segment bandwidth for program sweeping
- Presence of command and query: Command / Query
- IEEE488.2-1987 command mode
  - Command: [SOURce:]PSWeep[<chno>]:BANDwidth[<n>] <int>
  - Parameter: <int>
  - Response type: NR1 (integer value)
- IEEE488.1-1987 command mode
  - Command: USEG<int>
    URBW<int>
  - Parameter: <int>
  - Response type: NR1 (integer value)
- Description: Sets the segment bandwidth for the program sweeping.

R3762/63 command	R3764/66, R3765/67 command parameter	Operation
USEG	<n>	Specifies the segment number
URBW	<int>	Sets the bandwidth

- Note: The bandwidth setting is reflected in (USRASWP) only when PSWeep[<chno>]:MODE is ALL. When the mode is FREQ, it is not reflected in (USRFSWP).

14. [SOURce:]PSWeep[<chno>]:CLEar[<n>]

- Function: Clearing of specified segment for program sweeping
- Presence of command and query: Command
- IEEE488.2-1987 command mode
  - Command: [SOURce:]PSWeep[<chno>]:CLEar[<n>]
- Description: Clears the setting condition of the nth segment for program sweeping.

15.	[SOURce:]PSWeep[<chno>]:CLEar[<n>]:ALL	IEEE488.1-1987 command mode USEGCL

- Function — Clearing of all segments for program sweeping
- Presence of command and query — Command
- Command — [SOURce:]PSWeep[<chno>]:CLEar[<n>]:ALL
  USEGCL
- Description — Clears the setting condition of all the segments for program sweeping.

16. [SOURce:]PSWeep[<chno>]:FREQuency[<n>] — IEEE488.1-1987 command mode
USEG
UFREQ
U{START | STOP}

- Function — Inputting of segment frequency for program sweeping

- Presence of command and query — Command / Query

- IEEE488.2-1987 command mode
  - Command: [SOURce:]PSWeep[<chno>]:FREQuency[<n>] <start>[,<stop>]
  - Parameter: <start>
    <stop>
  - Response type: <start>,<stop>
    <start> = <stop> = NR3 (real value)

- IEEE488.1-1987 command mode
  - Command: USEG<int>
    UFREQ<real>
    U{START | STOP}<real>
  - Response type: NR1 (USEG command)
    NR3 (UFREQ | USTART | USTOP command)

- Description — Sets the segment frequency for program sweeping.

R3762/63 command	R3764/66, R3765/67 command parameter	Operation
USEG	<n>	Specifies the segment number
UFREQ	*1	Sets the fixed frequency
USTART	<start>	Sets the start frequency
USTOP	<stop>	Sets the stop frequency

*1: Corresponds to <start> when <stop> is omitted. If <stop> is omitted, <stop> = <start> and the segment point number (PSWeep[<chno>]:POINts[<n>]) will automatically be set to 1.

17. [SOURce:]PSWeep[<chno>]:MODE — IEEE488.1-1987 command mode USR{FSWP|ASWP}

- Function — Sweeping type setting
- Presence of command and query — Command / Query
- IEEE488.2-1987 command mode
  - Command — [SOURce:]PSWeep[<chno>]:MODE <mode>
  - Parameter — <mode> = {FREQuency|ALL}
  - Response type — NONE|FREQ|ALL
- IEEE488.1-1987 command mode
  - Command — USR{FSWP|ASWP}
  - Response type — 0 | 1
- Description

This command must be set by combining each item as shown in the table below:

Initial setting   Linear frequency sweeping

If PSW:MODE is set to FREQ or ALL, the segments already input are searched. And then, the segments are internally rearranged in the ascending order of the frequency and are executed.

In this case, if the STOP frequency of a segment is larger than the START frequency of the following segment after the rearrangement, an error occurs.

Command	PSW: MODE	FREQ: MODE	POW: MODE	SWE: SPAC	Sweeping type	Corresponding R3762/63 command
Parameter	(NONE)	SWE	(FIX)	LIN	Linear frequency sweeping	LINFREQ
				LOG	Log frequency sweeping	LOGFREQ
		(CW)	SWE	(LIN)	Level sweeping	LEVEL
	FREQ	(PSW)	(FIX)	(LIN)	Program sweeping (frequency only)	USRFSWP
	ALL	(PSW)	(PSW)	(LIN)	Program sweeping	USRARWP

Note: The value in parentheses indicates the value which is returned by a query. Do not use this value for setting.

Sweeping type	Linear frequency sweeping:	Sweeps the frequency at a constant interval at a fixed level.
	Log frequency sweeping:	Sweeps the frequency at a log interval at a fixed level.
	Level sweeping:	Sweeps the output level at a fixed frequency.
	Program sweeping (frequency only):	Arbitrarily sets the frequency only for each interval.
	Program sweeping:	Arbitrarily sets the frequency, the output level, the resolution bandwidth and the settling time for each interval.
	However, the log frequency sweeping cannot be set for R3764/66.	

18. [SOURce:]PSWeep[<chno>]:POINts[<n>] — IEEE488.1-1987 command mode
USEG
UPOINT

- Function — Inputting of number of segment points for program sweeping
- Presence of command and query — Command / Query
- IEEE488.2-1987 command mode
  - Command — [SOURce:]PSWeep[<chno>]:POINts[<n>] <int>
  - Parameter — <int>
  - Response type — NR1 (integer value)
- IEEE488.1-1987 command mode
  - Command — USEG<int>
    UPOINT<int>
  - Parameter — <int>
  - Response type — NR1 (integer value)
- Description — Sets the number of segment points for program sweeping.

R3762/63 command	R3764/66, R3765/67 command paramerter	Operation
USEG	<n>	Specifies the segment number
UPOINT	<int>	Sets the number of points

19. [SOURce:]PSWeep[<chno>]:POWer[<n>] — IEEE488.1-1987 command mode: USEG, ULEVEL

- Function: Inputting of segment output level for program sweeping

- Presence of command and query: Command / Query

- IEEE488.2-1987 command mode
  - Command: [SOURce:]PSWeep[<chno>]:POWer[<n>] <real>
  - Parameter: <real>
  - Response type: NR3 (real value)

- IEEE488.1-1987 command mode
  - Command: USEG<int>
    ULEVEL<real>
  - Parameter: <int>
    <real>
  - Response type: NR1 (USEG command)
    NR3 (ULEVEL command)

- Description: Sets the segment output level for program sweeping.

R3762/63 Command	R3764/66, R3765/67 command paramerter	Operation
USEG	<n>	Specifies the segment number
ULEVEL	<real>	Sets the output level

- Note: The setting value for the output level is reflected in (USRASWP) only when PSWeep[<chno>]:MODE is set to ALL. When the mode is FREQ, it is not reflected in (USRFSWP).

20. [SOURce:]PSWeep[<chno>]:SETTling[<n>] — IEEE488.1-1987 command mode
USEG
USETLT

- Function — Inputting of segment settling time for program sweeping

- Presence of command and query — Command / Query

- IEEE488.2-1987 command mode
  - Command — [SOURce:]PSWeep[<chno>]:SETTling[<n>] <real>
  - Parameter — <real>
  - Response type — NR3 (real value)

- IEEE488.1-1987 command mode
  - Command — USEG<int>
    USETLT<real>
  - Parameter — <int>
    <real>
  - Response type — NR1 (USEG command)
    NR3 (USETLT command)

- Description — Sets the segment settling time for program sweeping.

R3762/63 command	R3764/66, R3765/67 command paramerter	Operation
USEG	<n>	Specifies the segment number
USETLT	<real>	Sets the settling time

- Note — The setting value for the settling time is reflected in (USRASWP) only when PSWeep[<chno>]:MODE is set to ALL. When the mode is FREQ, it is not reflected in (USRFSWP).

21.	[SOURce:]SWEep[<chno>]:POINts	IEEE488.1-1987 command mode POIN M{1201|601|301|201|101|51|21|11|6|3}P

- Function — Setting of numbers of points for sweeping

- Presence of command and query — Command / Query

- IEEE488.2-1987 command mode
  - Command: [SOURce:]SWEep[<chno>]:POINts <int>
  - Parameter: <int>
  - Response type: NR1 (integer value)

- IEEE488.1-1987 command mode
  - Command: POIN<int>
    M{1201|601|301|201|101|51|21|11|6|3}P
  - Parameter: <int>
  - Query: POIN?
    M{1201|601|301|201|101|51|21|11|6|3}P?
  - Response type: NR1 (POIN? command)
    0 | 1 (M{1201|601|301|201|101|51|21|11|6|3}P? command)

- Description — Sets the numbers of the points for sweeping.

  The numbers of the points to be set are:
  3,6,11,21,51,101,201,301,401,601,801,1201

22. [SOURce:]SWEep[<chno>]:SPACing — IEEE488.1-1987 command mode
LINFREQ
LOGFREQ

- Function — Sweeping type specification
- Presence of command and query — Command / Query
- IEEE488.2-1987 command mode
  - Command — [SOURce:]SWEep[<chno>]:SPACing <mode>
  - Parameter — <mode> = {LINear | LOGarithmic}
  - Response type — LIN | LOG
- IEEE488.1-1987 command mode
  - Command — LINFREQ
    LOGFREQ
  - Response type — 0 | 1
- Description — This command must be set by combining each item as shown in the table below:
  Initial setting   Linear frequency sweeping

<table>
<tr><th>Command</th><th>PSW: MODE</th><th>FREQ: MODE</th><th>POW: MODE</th><th>SWE: SPAC</th><th>Sweeping type</th><th>Corresponding R3762/63 command</th></tr>
<tr><td rowspan="5">Parameter</td><td rowspan="3">(NONE)</td><td rowspan="2">SWE</td><td rowspan="2">(FIX)</td><td>LIN</td><td>Linear frequency sweeping</td><td>LINFREQ</td></tr>
<tr><td>LOG</td><td>Log frequency sweeping</td><td>LOGFREQ</td></tr>
<tr><td>(CW)</td><td>SWE</td><td>(LIN)</td><td>Level sweeping</td><td>LEVEL</td></tr>
<tr><td>FREQ</td><td>(PSW)</td><td>(FIX)</td><td>(LIN)</td><td>Program sweeping (frequency only)</td><td>USRFSWP</td></tr>
<tr><td>ALL</td><td>(PSW)</td><td>(PSW)</td><td>(LIN)</td><td>Program sweeping</td><td>USRARWP</td></tr>
</table>

Note: The value in parentheses indicates the value which is returned for a query. Do not use this value for setting.

Sweeping type	Linear frequency sweeping:	Sweeps the frequency at a constant interval at a fixed level.
	Log frequency sweeping:	Sweeps the frequency at a log interval at a fixed level.
	Level sweeping:	Sweeps the output level at a fixed frequency.
	Program sweeping (frequency only):	Arbitrarily sets the frequency only for each interval.
	Program sweeping:	Arbitrarily sets the frequency, the output level, the resolution bandwidth and the settling time for each interval.

However, the log frequency sweeping cannot be set for R3764/66.

23. [SOURce:]SWEep[<chno>]:TIME — IEEE488.1-1987 command mode STIME

- Function: Sweeping time setting
- Presence of command and query: Command / Query
- IEEE488.2-1987 command mode
  - Command: [SOURce:]SWEep[<chno>]:TIME <real>
    STIME<real>
  - Parameter: <real>
  - Response type: NR3 (real value)
- Description: Sets the sweeping time. Setting of "0" indicates AUTO.

Initial setting	30ms
Setting range	0.2ms to 3932.1s
Setting resolution	0.05ms

24. [SOURce:]SWEep[<chno>]:TIME:AUTO — IEEE488.1-1987 command mode STIMEAUTO

- Function: Automatic setting of sweeping time
- Presence of command and query: Command / Query
- IEEE488.2-1987 command mode
  - Command: [SOURce:]SWEep[<chno>]:TIME:AUTO <bool>
    STIMEAUTO
  - Parameter: <bool>
  - Response type: 0 | 1
- Description: Automatically sets the sweeping time to the minimum value which has been determined by the resolution bandwidth.
  If the sweeping time is set in the AUTO mode, the mode will be canceled.

7.12 STATus Subsystem

1. STATus:DEVice:CONDition?

- Function: DEV status referring
- Presence of command and query: Query
- Query: STATus:DEVice:CONDition?
- Response type: NR1 (integer value)
- Description: Returns the contents of condition register of the device status register. This register is not cleared even though it is read out.

For details, see "4. STATUS BYTE."

Condition register assignments

bit		Description
15		Always 0
14	Program Running	Sets to 1 during built-in BASIC program running.
4 to 13		Always 0
3	Sweeping	Sets to 1 during sweeping.
1 to 2		Always 0
0	Cooling Fan Stopped	Sets to 1 when the cooling fan is stopped.
Others		Always 0

2. STATus:DEVice:ENABle

• Function	OPER status referring
• Presence of command and query	Command/Query
• Command	STATus:DEVice:ENABle <int>
• Parameter	<int>
• Response type	NR1 (integer value)
• Description	Sets the contents of enable register of the device status register. The event register corresponding to the bit set to 1 in this register is reflected in 2 in the status byte register as a valid bit. For details, see "4. STATUS BYTE."
• Example	If the the Cooling Fan Stopped (bit 1) is to be set to 'enable', set STAT:DEV:ENAB 1.

3. STATus:DEVice[:EVENt]?

- Function: OPER status query (with clear)
- Presence of command and query: Query
- Query: STATus:DEVice:EVENt]?
- Response type: NR1 (integer value)
- Description: Returns the contents of event register of the device status register. When this register is read out, it's cleared and also bit 2 of the corresponding status byte register is cleared.

  For details, see "4. STATUS BYTE."

Event register assignments

bit		Description
15		Always 0
14	Program Running	Sets to 1 when the built-in BASIC program running stops.
9 to 13		Always 0
8	Averaging	Sets to 1 when the averaging ends.
4 to 7		Always 0
3	Sweeping	Sets to 1 when the sweeping ends.
1 to 2		Always 0
0	Cooling Fan Stopped	Sets to 1 when the cooling fan stops.
Others		Always 0

4. STATus:FREQuency:CONDition?

- Function — FREQ status referring
- Presence of command and query — Query
- Query — STATus:FREQuency:CONDition?
- Response type — NR1 (integer value)
- Description — Returns the contents of condition register of the frequency status register. Even though this register is read out, it's not cleared.

For details, see "4. STATUS BYTE."

Condition register assignments

bit		Description
0	Local 1 Unlocked	Sets to 1 when local 1 is unlocked.
1	Local 2 Unlocked	Sets to 1 when local 2 is unlocked.
2	Synthe Unlocked	Sets to 1 when synthesizer is unlocked.
3	External Standard In	Sets to 1 when external standard frequency is input.
4	VCXO Unlocked	Sets to 1 when VCXO is unlocked.
Others		Always 0

5. STATus:FREQuency:ENABle?

• Function	FREQ status enable register setting
• Presence of command and query	Command/Query
• Command	STATus:FREQuency:ENABle <int>
• Parameter	<int>
• Response type	NR1 (integer value)
• Description	Sets the contents of enable register of the frequency status register. The event register corresponding to the bit set to 1 in this register is reflected in the bit 5 in the questionable status register as a valid bit. For details, see "4. STATUS BYTE."
• Example	If the the External Standard In (bit 3) is to be set to 'enable', calculate 2**3 = 8 and set STAT:FREQ:ENAB 8.

6. STATus:FREQuency[:EVENt]?

- Function: FREQ status reading
- Presence of command and query: Query
- Query: STATus:FREQuency[:EVENt]?
- Response type: NR1 (integer value)
- Description: Returns the contents of event register of the frequency status register. When this register is read out, it's cleared, as is bit 5 of the corresponding questionable status register.

  For details, see "4. STATUS BYTE."

Event register assignments

bit		Description
0	Local 1 Unlocked	Sets to 1 when local 1 is unlocked.
1	Local 2 Unlocked	Sets to 1 when local 2 is unlocked.
2	Synthe Unlocked	Sets to 1 when synthesizer is unlocked.
3	External Standard In	Sets to 1 when external standard frequency is input.
4	VCXO Unlocked	Sets to 1 when VCXO is unlocked.
Others		Always 0

7. STATus:LIMit:CONDition?

- Function: LIM status referring
- Presence of command and query: Query
- Query: STATus:LIMit:CONDition?
- Response type: NR1 (integer value)
- Description: Returns the contents of condition register of the limit status register. Even if this register is read out, it's not cleared.

For details, see "4. STATUS BYTE."

Condition register assignments

bit		Description
0	CH1 1st Limit Failed	Sets to 1 when the first waveform of channel 1 is FAIL.
1	CH1 2nd Limit Failed	Sets to 1 when the second waveform of channel 1 is FAIL.
2	CH2 1st Limit Failed	Sets to 1 when the first waveform of channel 2 is FAIL.
3	CH2 2nd Limit Failed	Sets to 1 when the second waveform of channel 2 is FAIL.
4	CH3 1st Limit Failed	Sets to 1 when the first waveform of channel 3 1 is FAIL.
5	CH3 2nd Limit Failed	Sets to 1 when the second waveform of channel 3 1 is FAIL.
6	CH4 1st Limit Failed	Sets to 1 when the first waveform of channel 4 is FAIL.
7	CH4 2nd Limit Failed	Sets to 1 when the second waveform of channel 4 is FAIL.
Others		Always 0

8. STATus:LIMit:ENABle

- Function — LIM status enable register setting
- Presence of command and query — Command/Query
- Command — STATus:LIMit:ENABle <int>
- Parameter — <int>
- Response type — NR1 (integer value)
- Description — Sets the contents of enable register of the limit status register. The event register corresponding to the bit set to 1 in this register is reflected in the bit 9 in the questionable status register as a valid bit.

  For details, see "4. STATUS BYTE."
- Example — If the CH1 1st Limit Failed (bit 0) and the CH3 1st Limit Failed (bit 4) are to be set to 'enable', calculate 2**0 + 2**4 = 17 and set STAT:LIN:ENAB 17.

9. STATus:LIMit[:EVENt]?

- Function — LIM status reading
- Presence of command and query — Query
- Query — STATus:LIMit[:EVENt]?
- Response type — NR1 (integer value)
- Description — Returns the contents of event register of the limit status register. When this register is read out, it's cleared, as is bit 9 of the corresponding questionable status register.

  For details, see "4. STATUS BYTE."

Event register assignments

bit		Description
0	CH1 1st Limit Failed	Sets to 1 when the first waveform of channel 1 is FAIL.
1	CH1 2nd Limit Failed	Sets to 1 when the second waveform of channel 1 is FAIL.
2	CH2 1st Limit Failed	Sets to 1 when the first waveform of channel 2 is FAIL.
3	CH2 2nd Limit Failed	Sets to 1 when the second waveform of channel 2 is FAIL.
4	CH3 1st Limit Failed	Sets to 1 when the first waveform of channel 3 1 is FAIL.
5	CH3 2nd Limit Failed	Sets to 1 when the second waveform of channel 3 1 is FAIL.
6	CH4 1st Limit Failed	Sets to 1 when the first waveform of channel 4 is FAIL.
7	CH4 2nd Limit Failed	Sets to 1 when the second waveform of channel 4 is FAIL.
Others		Always 0

10. STATus:OPERation:CONDition?

- Function: OPER status referring
- Presence of command and query: Query
- Query: STATus:OPERation:CONDition?
- Response type: NR1 (integer value)
- Description: Returns the contents of condition register of the operation status register. Even if this register is read out, it's not cleared.

For details, see "4. STATUS BYTE."

Condition register assignments

bit		Description
0	Calibrating	Sets to 1 during calibrating.
3	Sweeping	Sets to 1 during sweeping.
14	Program Running	Sets to 1 during built-in BASIC program running.
Others		Always 0

11. STATus:OPERation:ENABle

• Function	OPER status enable register setting
• Presence of command and query	Command/Query
• Command	STATus:OPERation:ENABle <int>
• Parameter	<int>
• Response type	NR1 (integer value)
• Description	Sets the contents of enable register of the operation status register. The event register corresponding to the bit set to 1 in this register is reflected in the bit 7 in the status byte register as a valid bit. For details, see "4. STATUS BYTE."
• Example	If the Program Running (bit 14) and the Sweeping (bit 3) are to be set to 'enable', calculate 2**14 + 2**3 = 16392 and set STAT:OPER = ENAB 16392.

12. STATus:OPERation[:EVENt]?

- Function: OPER status reading
- Presence of command and query: Query
- Query: STATus:OPERation[:EVENt]?
- Response type: NR1 (integer value)
- Description: Returns the contents of event register of the operation status register. When this register is read out, it's cleared, as is bit 7 of the corresponding status byte register.

  For details, see "4. STATUS BYTE."

Event register assignments

bit		Description
0	Calibrating	Sets to 1 when the calibration ends.
3	Sweeping	Sets to 1 when the sweeping ends
14	Program Running	Sets to 1 when the built-in BASIC program stops.
Others		Always 0

13. STATus:OPERation:CONDition?

- Function: POW status referring
- Presence of command and query: Query
- Query: STATus:POWer:CONDition?
- Response type: NR1 (integer value)
- Description: Returns the contents of condition register of the power status register. This register is not cleared even if it is read out.

For details, see "4. STATUS BYTE."

Condition register assignments

bit		Description
0	Input-R Overloaded	Sets to 1 when the input-R is overloaded.
1	Input-R Tripped	Sets to 1 when the protection circuit of the input-R is in operation.
2	Input-A Overloaded	Sets to 1 when the input-A is overloaded.
3	Input-A Tripped	Sets to 1 when the protection circuit of the input-A is in operation.
4	Input-B Overloaded	Sets to 1 when the input-B is overloaded.
5	Input-B Tripped	Sets to 1 when the protection circuit of the input-B is in operation.
Others		Always 0

14. STATus:POWer:ENABle

• Function	POW status enable register setting
• Presence of command and query	Command/Query
• Command	STATus:OPERation:ENABle <int>
• Parameter	<int>
• Response type	NR1 (integer value)
• Description	Sets the contents of enable register of the power status register. The event register corresponding to the bit set to 1 in this register is reflected in the bit 3 in the questionable status register as a valid bit. For details, see "4. STATUS BYTE."
• Example	If the Input-A Overloaded (bit 2) is to be set to 'enable', calculate 2**2 = 4 and set STAT:POW:ENAB 4.

15. STATus:POWer[:EVENt]?

- Function: POW status reading
- Presence of command and query: Query
- Query: STATus:POWer[:EVENt]?
- Response type: NR1 (integer value)
- Description: Returns the contents of event register of the power status register. When this register is read out, it's cleared, as is bit 3 of the corresponding questionable status register.

For details, see "4. STATUS BYTE."

Event register assignments

bit		Description
0	Input-R Overloaded	Sets to 1 when the input-R is overloaded.
1	Input-R Tripped	Sets to 1 when the protection circuit of the input-R is in operation.
2	Input-A Overloaded	Sets to 1 when the input-A is overloaded.
3	Input-A Tripped	Sets to 1 when the protection circuit of the input-A is in operation.
4	Input-B Overloaded	Sets to 1 when the input-B is overloaded.
5	Input-B Tripped	Sets to 1 when the protection circuit of the input-B is in operation.
Others		Always 0

16. STATus:QUEStionable:ENABle

• Function	QUES status enable register setting
• Presence of command and query	Command/Query
• Command	STATus:QUEStionable:ENABle <int>
• Parameter	<int>
• Response type	NR1 (integer value)
• Description	Sets the contents of enable register of the questionable status register. The event register corresponding to the bit set to 1 in this register is reflected in the bit 3 in the status byte register as a valid bit. For details, see "4. STATUS BYTE."
• Example	If the POW (bit 3) and LIM (bit 9) summary bits are to be set to 'enable', calculate 2**3 + 2**9 = 520 and set STAT:QUES:ENAB 520.

17. STATus:QUEStionable[:EVENt]?

- Function — QUES status reading
- Presence of command and query — Query
- Query — STATus:QUEStionable[:EVENt]?
- Response type — NR1 (integer value)
- Description — Returns the contents of event register of the questionable status register. When this register is read out, it's cleared, as is the corresponding status byte register.

  For details, see "4. STATUS BYTE."

Event register assignments

bit		Description
3	POW Summary Bit	Sets to 1 when the summary of power status register is 1.
5	FREQ Summary Bit	Sets to 1 when the summary of frequency status register is 1.
9	LIM Summary Bit	Sets to 1 when the summary of limit status register is 1.
Others		Always 0

7.13 SYSTem Subsystem

1.	SYSTem:DATE	IEEE488.1-1987 command mode YEAR MONTH DAY

- Function — Date setting

- Presence of command and query — Command / Query

- IEEE488.2-1987 command mode

Command	SYSTem:DATE <year>,<month>,<day>
Parameter	<year> = Numeric data is 1900 to 2099
	<month> = Numeric data is 1 to 12
	<day> = Numeric data is 1 to 31
Response type	<year>,<month>,<day>
	<year> = <month> = <day> = NR1 (integer value)

- IEEE488.1-1987 command mode

Command	YEAR<int>
	MONTH<int>
	DAY<int>
Parameter	<int>
Response type	NR1 (integer value)

- Description — Sets the date on the timer built into the analyzer.

  Use the Christian calendar (four digits) to set the year (examples: 1990, 1993)

2. SYSTem:ERRor?

- Function: Query of error
- Presence of command and query: Query
- Query: SYSTem:ERRor?
- Response type: <errno>,<errmsg>
  <errno> = NR1 (integer value)
  <errmsg> = error messege
- Description: The system can store information on up to 10 errors in the error queue. If more than nine errors occur, the indication of 10th error will be replaced with:

  -350, "Queue overflow"

  The 10th and subsequent errors cannot be maintained.SYSTem:ERRor? removes the error information from the queue.
  Since the queue stores errors using the FIFO (First-In First-Out) method, the command removes error information in the order of occurrence of errors.

  When error information is removed from the queue, the information is deleted from the queue, and the queue is ready for the next error information.
  If there is no error, the system responds with:

  0, "No error"

  The *CLS command clears the error queue.

3.	SYSTem:PRESet	IEEE488.1-1987 command mode IP

- Function — System initialization
- Presence of command and query — Command
- Command — SYSTem:PRESet
IP
- Description — The SYSTem:PRESet (IP) command initializes the setting of the analyzer and resets the trigger system.
  The initial values set using this command are different from those set using the *RST command.For actual setting values, see "A3. INITIAL SETTING".
  The items this command performs are the same as those performed using the PRESET key on the front panel.

4.	SYSTem:TIME	IEEE488.1-1987 command mode HOUR MINUTE RTC30ADJ

• Function	Time setting
• Presence of command and query	Command / Query
• IEEE488.2-1987 command mode	
Command	SYSTem:TIME <hour>,<minute>,<second>
Parameter	<hour> = Numeric data is 0 to 23 <minute> = Numeric data is 0 to 59 <second> = Numeric data is 0 to 59
Response type	<hour>,<minute>,<second> <hour> = <minute> = <second> = NR1 (integer value)
• IEEE488.1-1987 command mode	
Command	HOUR<int> MINUTE<int> RTC30ADJ
Parameter	<int>
Response type	NR1 (integer value) There is no query for the RTC30ADJ command.
• Description	Sets the time on the timer built into the analyzer. A 24-hour clock is used. The RTC30ADJ command of IEEE488.1-1987 command mode always sets the second to "0".

7.14 TRACe Subsystem

1.	TRACe[<chno>]:COPY	IEEE488.1-1987 command mode DTOM

- Function — Trace copying
- Presence of command and query — Command
- Command — TRACe[<chno>]:COPY <name>
  DTOM
- Parameter — <name> = DATA
- Description — The command copies the data waveform onto the memory waveform.

2.	TRACe[<chno>][:DATA]?	IEEE488.1-1987 command mode OT{1|2|3|4}{DRAT|CORED|MRAT|NORED|DFOR|MFOR|CORNR|CORDI|CORSO|CORTR}

- Function — Query of trace (output)
- Presence of command and query — Query
- IEEE488.2-1987 command mode
  - Query — TRACe[<chno>][:DATA]?{<name>|<trace>}[,{<name>|<trace>}...]
  - Parameter — <name> = {RAW|DATA|MEM|UDAT|FDAT1|FDAT2|FMEM1|FMEM2|NORM|EDIR|ESM|ERTR|EDF|ESF|ERF|ELF|ETF|EXF|EDR|ESR|ERR|ELR|ETR|EXR}
    <trace> = Analysis channel
- IEEE488.1-1987 command mode
  - Query — OT{1|2|3|4}{DRAT|CORED|MRAT|NORED|DFOR|MFOR|CORNR|CORDI|CORSO|CORTR}
- Description — Outputs the specified trace data.Multiple <names> or <trace> can be specified by separating them with a comma. In such cases, the data per trace are output in the specified order. (After the data corresponding to one trace are output, outputting of the data of next trace is begun.)

3.	TRACe[<chno>][:DATA]	IEEE488.1-1987 command mode IN{1|2|3|4}{DRAT|CORED|MRAT|NORED|DFOR| MFOR|CORNR|CORDI|CORSO|CORTR}

- Function — Trace inputting
- Presence of command and query — Command
- IEEE488.2-1987 command mode

  Command: TRACe[<chno>][:DATA]{<name>|<trace>},
  {<block>|<real>[,<real>...]}

  Parameter: <name> = {RAW|DATA|MEM|UDAT|FDAT1|FDAT2|FMEM1|
  FMEM2|NORM|EDIR|ESM|ERTR|EDF|ESF|ERF|
  ELF|ETF|EXF|EDR|ESR|ERR|ELR|ETR|EXR}
  <trace> = Analysis channel

- IEEE488.1-1987 command mode

  Command: IN{1|2|3|4}{DRAT|CORED|MRAT|NORED|DFOR|MFOR|
  CORNR|CORDI|CORSO|CORTR}

- Description — Inputs the data into the specified trace.
  Unlike trace outputting, multiple <name> or <trace> cannot be specified.

* Trace input/output command parameters

R3762/63 command	R3764/66, R3765/67 command parameter		Object traces	Data format*2
	<name>*1	<trace>		
OT{1|2|3|4}DRAT	RAW	{131|195|259|323}	Raw data array	Complex number
OT{1|2|3|4}CORED	DATA	{129|193|257|321}	Data array	Complex number
OT{1|2|3|4}MRAT	MEMory	{130|194|258|322}	Memory array	Complex number
OT{1|2|3|4}NORED	UDATa	{128|192|256|320}	Data array before formatting	Complex number
OT{1|2|3|4}DFOR	FDATa1	{0|1|4|5}	Data array after formatting 1	First waveform
	FDATa2	{8|9|12|13}	Data array after formatting 2	Second waveform
OT{1|2|3|4}MFOR	FMEMory1	{2|3|6|7}	Memory array after formatting 1	First waveform
	FMEMory2	{10|11|14|15}	Memory array after formatting 2	Second waveform
{OT|IN}{1|2|3|4}CORNR	NORMalize	{133|197|261|325}	Normalized reference data array	Complex number
{OT|IN}{1|2|3|4}CORDI	EDIRectivity	{134|198|262|326}	Direction error coefficient array	Complex number
{OT|IN}{1|2|3|4}CORSO	ESMatch	{135|199|263|327}	Source match error coefficient array	Complex number
{OT|IN}{1|2|3|4}CORTR	ERTRacking	{136|200|264|328}	Reflection tracking error coefficient array	Complex number
	EDForward	{137|201|265|329}	Forward direction: Direction error coefficient array	Complex number
	ESForward	{138|202|266|330}	Forward direction: Source match error coefficient array	Complex number
	ERForward	{139|203|267|331}	Forward direction: Reflection tracking error coefficient array	Complex number
	ELForward	{140|204|268|332}	Forward direction: load match error coefficient array	Complex number
	ETForward	{141|205|269|333}	Forward direction: Transfer tracking error coefficient array	Complex number
	EXForward	{142|206|270|334}	Forward direction: Isolation error coefficient array	Complex number
	EDReverse	{143|207|271|335}	Reverse direction: Direction error coefficient array	Complex number
	ESReverse	{144|208|272|336}	Reverse direction: Source match error coefficient array	Complex number
	ERReverse	{145|209|273|337}	Reverse direction: Reflection tracking error coefficient array	Complex number
	ELReverse	{146|210|274|338}	Reverse direction: load match error coefficient array	Complex number
	ETReverse	{147|211|275|339}	Reverse direction: Transfer tracking error coefficient array	Complex number
	EXReverse	{148|212|276|340}	Reverse direction: Isolation error coefficient array	Complex number

*1: If <name> is specified using R3764/66, R3765/67 command, the channel should be specified using the parameter <chno>.

*2: The data type depends on the trace type (see below).

Complex number: Complex numbers are output in the order real, imaginary, real, imaginary, and so on. Therefore, the total number of data output is doubled.

First waveform: When the format is set to LOGMAG&PHASE or LOGMAG&DELAY, the first waveform is LOGMAG; when the format is set to LINMAG&PHASE, the first waveform is LINMAG; when the format is set to SMITH or POLAR, the first waveform is real; when the measure mode is S11&S21, the first waveform is S11; and when the measure mode is S22&S12, the first waveform is S22.

Second waveform: When the format is set to LOGMAG&PHASE or LINMAG&PHASE, the second waveform is PHASE; when the format is set to LOGMAG&DELAY, the second waveform is DELAY; when the format is set to SMITH or POLAR, the second waveform is imaginary part; when the measure mode is S11&S21, the second waveform is S21; and when the measure mode is S22&S12, the second waveform is S12.
In other cases, the data are invalid.

7.15 TRIGger Subsystem

1. TRIGger[:SEQuence]:DELay — IEEE488.1-1987 command mode SETLTIME

● Function	Trigger delay setting
● Presence of command and query	Command / Query
● Command	TRIGger[:SEQuence]:DELay <real> SETLTIME<real>
● Parameter	<real>
● Response type	NR3 (real value)
● Description	This command sets the delay time between the detection of the trigger and the start of measurement. The delay time is available only when TRIGger[:SEQuence]:DELay:STATe is set to ON. See "TRIGger[:SEQuence]:DELay:STATe".
● Note	If 0 is set, TRIG:DEL:STAT is automatically set to OFF. If the value other than 0 is set, TRIG:DEL:STAT is automatically set to ON.

2. TRIGger[:SEQuence]:DELay:STATe — IEEE488.1-1987 command mode SETLVARI

● Function	ON/OFF of trigger delay
● Presence of command and query	Command / Query
● Command	TRIGger[:SEQuence]:DELay:STATe <bool> SETLVARI<bool>
● Parameter	<bool>
● Response type	0 | 1
● Description	This command enables/disables the trigger delay times set by the TRIGger[:SEQuence]:DELay (SETLTIME) command. Setting this command to OFF is identical to TRIGger[:SEQuence]:DELay 0 (SETLTIME 0).
● Note	If OFF is set, TRIG:DEL is automatically set to 0.

3. TRIGger[:SEQuence][:IMMediate]

• Function	Event detection path (not delay)
• Presence of command and query	Command
• Command	TRIGger[:SEQuence][:IMMediate]
• Description	This command bypasses the trigger waiting state. If the trigger system is in the trigger waiting state, the command starts the measurement immediately. In this case, the delay time set by the TRIGger[:SEQuence]:DELay (SETLTIME) command becomes invalid. For details, see "5. TRIGGER SYSTEM".

4. TRIGger[:SEQuence]:SIGNal

• Function	Event detection path (with delay)
• Presence of command and query	Command
• Command	TRIGger[:SEQuence]:SIGNal
• Description	This command bypasses the event detection of the trigger waiting state. If the trigger system is in the trigger waiting state, the command starts the measurement after the delay time set by TRIGger[:SEQuence]:DELay (SETLTIME) has elapsed. For details, see "5. TRIGGER SYSTEM".

5. TRIGger[:SEQuence]:SOURce — IEEE488.1-1987 command mode: FREE, EXTERN

- Function: Trigger source setting
- Presence of command and query: Command / Query
- IEEE488.2-1987 command mode
  - Command: TRIGger[:SEQuence]:SOURce <source>
  - Parameter: <source> = {IMMediate | EXTernal | BUS | HOLD}
  - Response type: IMM | EXT | BUS | HOLD
- IEEE488.1-1987 command mode
  - Command: FREE
    EXTERN
  - Response type: 0 | 1
- Description

This command selects the trigger source. The event detection ends when all of the conditions below are satisfied.

IMMediate: Has no event. This condition immediately ends the event detection of the trigger waiting state.

EXTernal: Waits for the external signal.

BUS: Waits for the *TRG interface message or the GET interface message.

HOLD: Does not end the event detection of the trigger waiting state.

If the analyzer receives TRIGger[:IMMediate] or TRIGger:SIGNal in the trigger waiting state, it starts the measurement regardless of the trigger source setting.

For details, see "5. TRIGGER SYSTEM".

FREE and EXTERN of IEEE488.1-1987 command mode select the same trigger sources as IMMediate and EXTernal of IEEE488.2-1987 command mode, respectively.

7.16 R3762/63 Command

1. CONT

- Function: Sets the sweeping mode to CONT
- Presence of command and query: Command
- Command: CONT
- Description: Performs continuous sweeping and measurement.

2. MEAS

- Function: Performs measurement
- Presence of command and query: Command
- Command: MEAS
- Description: If the system is in the process of sweeping, it resets the sweeping and performs the sweeping and the measurement once.If the sweeping mode is set to CONT, it continuously performs the sweeping and the measurement.

3. SINGLE

- Function: Sets the sweeping mode to SINGLE
- Presence of command and query: Command
- Command: SINGLE
- Description: The system performs the sweeping and the measurement once.

4. SWPHLD

- Function — Holds the sweeping
- Presence of command and query — Command
- Command — SWPHLD
- Description — The system immediately stops the sweeping.

7.17 R3765/67 MARKer Subsystem

1.	MARKer[<chno>]:ACTivate[:NUMBer]	IEEE488.1-1987 command mode MKR{1|2|3|4|5|6|7|8|9|10}A

- Function — Setting of active marker

- Presence of command and query — Command / Query

- IEEE488.2-1987 command mode

Command	MARKer[<chno>]:ACTivate[:NUMBer] <n>[,<real>]	
Parameter	<n> = 1 to 10 (marker number)	
	<real> = Setting value (stimulus value)	
Response type	NR1 (integer value):	0 to 10 (marker number)
	NR3 (real value):	Setting value (stimulus value)

- IEEE488.1-1987 command mode

Command	MKR{1|2|3|4|5|6|7|8|9|10}A	
Response type	NR3 (real value):	Setting value (stimulus value)
	NR3 (real value):	Measurement value (data A, B, C)
	NR1 (integer value):	Status

- Description

Specifies a number of the active marker. The specified marker will automatically be ON.

In IEEE488.2-1987 command mode, the maker number and the setting value are returned by the query. If no marker is ON, 0 is set as the marker number.
Setting value can be obtained by the FETch? query.
In IEEE488.1-1987 command mode, setting value and measurement value are returned by the query.
Refer to "7.18 FETch? Subsystem" for details of data and format.

2.	MARKer[<chno>]:ACTivate:STATe	IEEE488.1-1987 command mode MKROFF

- Function — ON/OFF of marker
- Presence of command and query — Command / Query
- Command — MARKer[<chno>]:ACTivate:STATe <bool>
  MKROFF
- Parameter — <bool>
- Response type — 0 | 1
- Description — If the active marker is set to OFF and the other markers are set to ON, the marker having the smallest number is changed as an active marker.

  In IEEE488.2-1987 command mode, the marker 1 is set to ON only when the parameters are ON and the marker 1 is OFF.

3. MARKer[<chno>]:ACTivate:STIMulus — IEEE488.1-1987 command mode MKR{1|2|3|4|5|6|7|8|9|10}A

- Function: Setting of marker stimulus value.
- Presence of command and query: Command / Query
- IEEE488.2-1987 command mode
  - Command: MARKer[<chno>]:ACTivate:STIMulus <real>
  - Parameter: <real> = Stimulus value
  - Response type: NR3(real value): Stimulus value
- IEEE488.1-1987 command mode
  - Command: MKR{1|2|3|4|5|6|7|8|9|10}A <real>
  - Parameter: <real> = Stimulus value
  - Response type:
    - NR3 (real value): Setting value (stimulus value)
    - NR3 (real value): Measurement value (data A, B, C)
    - NR1 (integer value): Status
- Description: Sets the stimulus value of the active marker.

  In IEEE488.2-1987 command mode, setting value is returned by the query.
  Measurement data can be obtained by the RETch? query.
  In IEEE488.1-1987 command mode, setting value and measurement value are returned by the query.

4.	MARKer[<chno>]:AOFF	IEEE488.1-1987 command mode MKRAOFF

• Function	OFF of all markers
• Presence of command and query	Command
• Command	MARKer[<chno>]:AOFF MKRAOFF
• Description	Sets all markers to OFF.

5.	MARKer[<chno>]:COMPensate	IEEE488.1-1987 command mode MKRCMP MKRUCMP

- Function — ON/OFF of marker interpolation mode

- Presence of command and query — Command / Query

- IEEE488.2-1987 command mode
  - Command — MARKer[<chno>]:COMPensate <bool>
  - Parameter — <bool>
  - Response type — 0 | 1

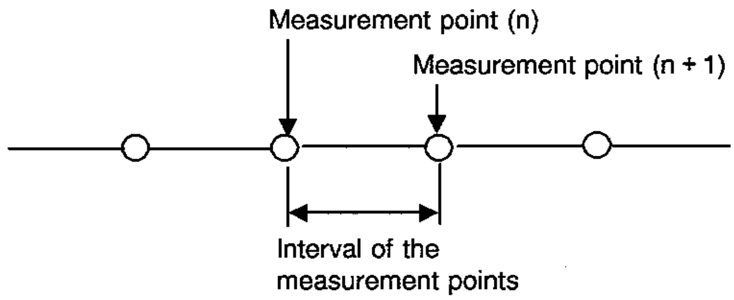
- IEEE488.1-1987 command mode
  - Command — MKRCMP→ ON
    MKRUCMP→ OFF
  - Response type — 0 | 1

- Description

Marker interpolation mode is used to interpolate the data between measurement points in linear approximation.

OFF: Marker can be set only to the measurement point. If you set the stimulus value to the point other than the measurement point, it is automatically changed to the nearest measurement point.

ON: Marker between the measurement points can be set with interpolating.



6. MARKer[<chno>]:CONVert[:MODE] IEEE488.1-1987 command mode ZYMK{DFLT|LIN|RI|LC}

- Function: Setting of marker conversion mode
- Presence of command and query: Command / Query
- IEEE488.2-1987 command mode
  - Command: MARKer[<chno>]:CONVert[:MODE] <format>
  - Parameter: <format> = {DEFault|LINear|RIMaginary}
  - Response type: DEF|LIN|RIM
- IEEE488.1-1987 command mode
  - Command: ZYMK{DFLT|LIN|RI|LC}
  - Response type: 0 | 1
- Description: Sets the format of the measurement marker value irrespective of the measurement format. This command is effective when the parameter conversion of the measurement value is in execution.

R3762/63 command mode	R3764/66, R3765/67 command parameter	Marker Format
ZYMKDFLT	DEFault	The same format as the measurement format
ZYMKLIN	LINear	Linear impedance
ZYMKRI	RIMaginary	Imaginary impedance

7. MARKer[<chno>]:COUPle — IEEE488.1-1987 command mode: MKRCOUP, MKRUCOUP

- Function: Setting of marker couple mode
- Presence of command and query: Command / Query
- IEEE488.2-1987 command mode
  - Command: MARKer[<chno>]:COUPle <bool>
  - Parameter: <bool>
  - Response type: 0 | 1
- IEEE488.1-1987 command mode
  - Command: MKRCOUP→ ON
    MKRUCOUP→ OFF
  - Response type: 0 | 1
- Description: Sets ON/OFF the marker coupling of the channel 1, 2, 3 and 4.

  ON: The marker set to the active channel is automatically set to the other channels.

  OFF: Marker is set to the channel 1, 2, 3 and 4 each.

8. MARKer[<chno>]:DELTa[:MODE] — IEEE488.1-1987 command mode DMKR{C|A|F|OF}

- Function: Setting of delta marker
- Presence of command and query: Command / Query
- IEEE488.2-1987 command mode
  - Command: MARKer[<chno>]:DELTa[:MODE] <type>
  - Parameter: <type> = {OFF|CHILd|COMPare|FIXed}
  - Response type: OFF|CHIL|COMP|FIX
- IEEE488.1-1987 command mode
  - Command: DMKRC
    DMKRA
    DMKRF
    DMKROF
  - Response type: 0 | 1
- Description: Sets the mode of the delta marker.

R3762/63 command mode	R3764/66, R3765/67 command parameter	Mode
DMKRC	CHIL	Sets the child marker to the point of the active marker and obtains the difference between the active marker and the child marker.
DMKRA	COMP	Obtains the difference between the active marker and the other marker.
DMKRF	FIX	Obtains the difference between the fixed marker (FIX MKR) and the active marker.
DMKROF	OFF	Sets the delta maker mode to OFF.

Note: Before setting the delta mode to COMP, specify the compare marker.

Delta stimulus cannot be set in IEEE488.1-1987 command mode.

9. MARKer[<chno>]:DELTa:COMPare — IEEE488.1-1987 command mode DMKR{1|2|3|4|5|6|7|8|9|10}O

- Function: Compare marker specification
- Presence of command and query: Command / Query
- IEEE488.2-1987 command mode
  - Command: MARKer[<chno>]:DELTa:COMPare <n>[,<real>]
  - Parameter: <n> = 1 to 10 (marker number)
    <real> = Stimulus value (relative value from the active marker)
  - Response type: <NR1> (integer value): 1 to 10 (marker number)
    <NR3> (real value): Stimulus value (relative value from the active marker)
- IEEE488.1-1987 command mode
  - Command: DMKR{1|2|3|4|5|6|7|8|9|10}O
  - Parameter: <real> = Stimulus value (relative value from the active marker)
  - Response type: 0 | 1
- Description: Specifies the marker to be compared when the delta marker is set to the COMPare mode. And, sets the position in the relative value from the active marker.

10. MARKer[<chno>]:FANalysis:DIRection   IEEE488.1-1987 command mode
TIN
TOUT

- Function — Setting the direction for the filter analysis

- Presence of command and query — Command / Query

- IEEE488.2-1987 command mode
  - Command: MARKer[<chno>]:FANalysis:DIRection <type>
  - Parameter: <type> = {IN|OUT}
  - Response type: IN|OUT

- IEEE488.1-1987 command mode
  - Command: TIN
    TOUT
  - Response type: 0 | 1

- Description — Sets the direction for the filter analysis.

R3762/63 command mode	R3764/66, R3765/67 command parameter	Direction
TIN	IN	Searching outward from the active marker.
TOUT	OUT	Searching toward the active marker.

11. MARKer[<chno>]:FANalysis[:STATe] — IEEE488.1-1987 command mode FLTANA

- Function: ON/OFF of filter analysis
- Presence of command and query: Command / Query
- Command: MARKer[<chno>]:FANalysis[:STATe] <bool>
  FLTANA <bool>
- Parameter: <bool>
- Response type: 0 | 1
- Description: Sets the filter analysis to ON or OFF.

  The following items can be measured by the filter analysis.
  · Center frequency of pass band specified with the analysis width (loss) from the active marker.
  · Pass bandwidth
  · Left frequency of pass band
  · Right frequency of pass band
  · Quality factor (Q factor)
  · Selectivity (shaping factor)

  Quality factor (Q factor) and selectivity (shaping factor) are obtained from the loss minimum value.

12.	MARKer[<chno>]:FANalysis:WIDTh	IEEE488.1-1987 command mode T{3|6|60|X}DB T{3|6|X}DEG

- Function — Setting the analysis band of the filter analysis

- Presence of command and query — Command / Query

- IEEE488.2-1987 command mode

Command	MARKer[<chno>]:FANalysis:WIDTh <real>
Parameter	<real> = Analysis band (pass bandwidth)
Response type	NR3(real value): Analysis band (pass bandwidth)

- IEEE488.1-1987 command mode

Command	T3DB	T3DEG
	T6DB	T6DEG
	T60DB	TXDEG<real>
	TXDB<real>	
Parameter	<real> = Analysis band (pass bandwidth)	
Response type	NR3 (real value):	CENTER
	NR3 (real value):	LEFT
	NR3 (real value):	RIGHT
	NR3 (real value):	BAND
	NR3 (real value):	QUALITY FACTOR
	NR3 (real value):	SHQPE FACTOR
	NR1 (integer value):	Status

- Description

Sets the analysis band (pass bandwidth) of the filter analysis.

To set 3dB, 6dB or 60dB in IEEE488.1-1987 command mode, execute each of them by T3DB, T6DB, and T60DB command. Only when the TXDB command is used, set a <real> value.

If 3deg or 6deg is set in phase, use T3DEG or T6DEG command. Only when the TXDEG command is used, set a <real> value.

13. MARKer[<chno>]:FIXed:STIMulus — IEEE488.1-1987 command mode FMKRS

- Function: Setting the X axis value of the fixed marker (FIX MKR)
- Presence of command and query: Command / Query
- Command: MARKer[<chno>]:FIXed:STIMulus <real>
  FMKRS <real>
- Parameter: <real> = X axis value
- Response type: <NR3> real value: X axis value
- Description: Sets the X axis value of the fixed marker (FIX MKR) in the rectangular coordinates display.

  The fixed marker (FIX MKR) is available only when the parameter conversion is OFF or 1/S.

14.	MARKer[<chno>]:FIXed:VALue	IEEE488.1-1987 command mode FMKRV

- Function — Setting the Y axis value of the fixed marker (FIX MKR)
- Presence of command and query — Command / Query
- Command — MARKer[<chno>]:FIXed:VALue <real>
  FMKRV <real>
- Parameter — <real> = Y axis value
- Response type — <NR3> real value: Y axis value
- Description — Sets the Y axis value of the fixed marker (FIX MKR) in the rectangular coordinates display.

  Sets the value of the real part in the Smith chart or the polar coordinates display.

15. MARKer[<chno>]:FIXed:AVALue

- Function — Setting of the imaginary part of the fixed marker (FIX MKR)
- Presence of command and query — Command / Query
- Command — MARKer[<chno>]:FIXed:AVALue <real>
- Parameter — <real> = Imaginary part
- Response type — <NR3> real value: Imaginary part
- Description — Sets the imaginary part of the fixed marker (FIX MKR) in the Smith chart or the polar coordinates display.

16. MARKer[<chno>]:LET   IEEE488.1-1987 command mode
MKR{REF|CENT|STAR|STOP|SPAN|FIX}

- Function   Marker assignment function.

- Presence of command and query   Command

- IEEE488.2-1987 command mode
  Command   MARKer[<chno>]:LET <type>
  Parameter   <type> = {STARt|STOP|CENTer|SPAN|RLEVel|FIXed}

- IEEE488.1-1987 command mode
  Command   MKR{REF|CENT|STAR|STOP|SPAN|FIX}

- Description   Assigns the setting value and the measurement value of the active marker to each setting parameter.

R3762/63 command mode	R3764/66, R3765/67 command parameter	Operation
MKRREF	RLEV	Assigns the Y axis value (measurement value) of the active marker to the reference value.
MKRCENT	CENT	Assigns the X axis value (setting value) of the active marker to the center value of the sweep. This command is available only in the frequency sweep.
MKRSTAR	STAR	Assigns the X axis value (setting value) of the active marker to the start value of the sweep.
MKRSTOP	STOP	Assigns the X axis value (setting value) of the active marker to the stop value of the sweep.
MKRSPAN	SPAN	Assigns the delta marker value (setting value) to the span value of the sweep. This command is available only in the frequency sweep.
MKRFIX	FIX	Assigns the position of the active marker to the fixed marker (FIX MKR).

17. MARKer[<chno>]:LIST

• Function	ON/OFF of marker list display
• Presence of command and query	Command / Query
• Command	MARKer[<chno>]:LIST <bool>
• Parameter	<bool>
• Response type	0 | 1
• Description	Switches the marker list display to ON or OFF.

18. MARKer[<chno>]:POLar — IEEE488.1-1987 command mode PMKR{LIN|LOG|RI}

- Function — Setting the marker mode for the polar display
- Presence of command and query — Command / Query
- IEEE488.2-1987 command mode
  - Command — MARKer[<chno>]:POLar <type>
  - Parameter — <type> = {MLINear|MLOGarithm|RIMaginary}
  - Response type — MLIN|MLOG|RIM
- IEEE488.1-1987 command mode
  - Command — PMKR{LIN|LOG|RI}
  - Response type — 0 | 1
- Description — Sets the marker mode for the polar display.

R3762/63 command mode	R3764/66, R3765/67 command parameter	Mode
PMKRLIN	MLIN	Linear value
PMKRLOG	MLOG	Logarithm value
PMKRRI	RIM	Complex value

19. MARKer[<chno>]:SMITh — IEEE488.1-1987 command mode SMKR{LIN|LOG|RI|RX|GB}

- Function — Setting the marker mode for the smith chart display
- Presence of command and query — Command / Query
- IEEE488.2-1987 command mode
  - Command — MARKer[<chno>]:SMITh <type>
  - Parameter — <type> = {MLINear|MLOGarithm|RIMaginary|IMPedance |ADMittance}
  - Response type — MLIN|MLOG|RIM|IMP|ADM
- IEEE488.1-1987 command mode
  - Command — SMKR{LIN|LOG|RI|RX|GB}
  - Response type — 0 | 1
- Description — Sets the marker mode for the smith chart display.

R3762/63 command mode	R3764/66, R3765/67 command parameter	Mode
SMKRLIN	MLIN	Linear value
SMKRLOG	MLOG	Logarithm value
SMKRRI	RIM	Complex value
SMKRRX	IMP	Impedance value
SMKRGB	ADM	Admittance value

20. MARKer[<chno>]:SEARch[:MODE]

IEEE488.1-1987 command mode
SRCHOFF
{MAX|MIN}SRCH
ZRPSRCH
DRIPPL1

- Function: Marker search function

- Presence of command and query: Command / Query

- IEEE488.2-1987 command mode
  - Command: MARKer[<chno>]:SEARch[:MODE] <type>
  - Parameter: <type> = {OFF|MAX|MIN|TARGet|RIPPle}
  - Response type: OFF|MAX|MIN|TARG|RIPP

- IEEE488.1-1987 command mode
  - Command:
    SRCHOFF
    {MAX|MIN}SRCH
    ZRPSRCH
    DRIPPL1
  - Response type:

SRCHOFF:	0 | 1	
{MAX|MIN}SRCH:	NR3 (real value):	Setting value (stimulus value)
ZRPSRCH	NR3 (real value):	Measurement value (data A, B, C)
DRIPPL1	NR1 (integer value):	Status

- Description

Sets the marker search function.

R3762/63 command mode	R3764/66, R3765/67 command parameter	Search Mode
SRCHOFF	OFF	OFF
MAXSRCH	MAX	Maximum value
MINSRCH	MIN	Minimum value
ZRPSRCH	TARG	Target value
DRIPPL1	RIPP1	Ripple value

In IEEE488.2-1987 command mode, the search mode is returned by the query.
Measurement value can be obtained by the FETch? query.
In IEEE488.1-1987 command mode, measurement value is returned by the query.

21. MARKer[<chno>]:SEARch:PARTial:SRANge

- Function — Area for the partial marker searching specification
- Presence of command and query — Command
- IEEE488.2-1987 command mode
  Command — MARKer[<chno>]:SEARch:PARTial:SRANge
- Description — Specifies the area for the partial marker searching. The searching is executed in the area specified between the delta markers.
  This command is disabled if the delta marker is OFF.

  This command is used only to specify the area for searching. Set the partial search to ON or OFF by the MARK:SEAR:PART:STAT command.

  Note: In IEEE488.1-1987 command mode, this function is automatically executed by the MKRPART ON.

22.	MARKer[<chno>]:SEARch:PARTial[:STATe]	IEEE488.1-1987 command mode MKRPART

● Function	ON/OFF of partial marker searching
● Presence of command and query	Command / Query
● Command	MARKer[<chno>]:SEARch:PARTial[:STATe] <bool> MKRPART <bool>
● Parameter	<bool>
● Response type	0 | 1
● Description	Specifies the partial marker search to ON or OFF.

23. MARKer[<chno>]:SEARch:RIPPle[:MODE]   IEEE488.1-1987 command mode
DRIPPL1
DMAXMIN

- Function: Mode specification of ripple search
- Presence of command and query: Command / Query
- IEEE488.2-1987 command mode
  - Command: MARKer[<chno>]:SEARch:RIPPle[:MODE] <type>
  - Parameter: <type> = {MAX|MIN|BOTH|PPEak}
  - Response type: MAX|MIN|BOTH|PPEak
- Description: Specifies a mode for the ripple search.

R3762/63 command	R3764/66, R3765/67 command parameter	Mode
	MAX	Obtains the maximum value of local maximum values.
	MIN	Obtains the minimum value of local minimum values.
DRIPPL1	BOTH	Obtains the difference between the maximum value of local maximum values and the minimum value of local minimum values.
DMAXMIN	PPEak	Obtains the difference between the maximum value and the minimum value.

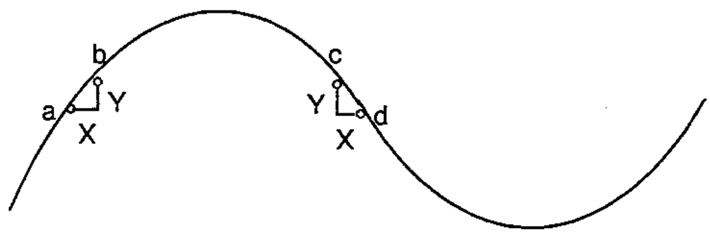
(Note) DRIPPL2 is not supported.

24.	MARKer[<chno>]:SEARch:RIPPle{:DX|:DY}	IEEE488.1-1987 command mode DLT{X|Y}

- Function — Setting the detectivity of the ripple search
- Presence of command and query — Command / Query
- IEEE488.2-1987 command mode
  - Command — MARKer[<chno>]:SEARch:RIPPle{:DX|:DY} <real>
    DLT{X|Y} <real>
  - Parameter — <real> = Setting value
  - Response type — <NR3> real value: Setting value
- Description — Sets the detectivity of the ripple search.

If the detectivity is set to $\Delta Y/\Delta X$, first obtain the a point of which the gradient of the waveform (Y/X) is $\Delta Y/\Delta X$ or more, then obtain the d point of which the reverse gradient is $\Delta Y/\Delta X$ or more. And finally obtain a maximum value between the a point and the d point as the local maximum peak.
Obtain a minimum value in the same way of obtaining a maximum value with the reverse gradient.

IEEE488.2-1987 command mode; DX→Set the ΔX
DY→Set the ΔY
IEEE488.1-1987 command mode; DLTX→Set the ΔX
DLTY→Set the ΔY



25. MARKer[<chno>]:SEARch:TARGet[:MODE] IEEE488.1-1987 command mode ZRPSRCH

- Function: Mode specification of the target search
- Presence of command and query: Command / Query
- IEEE488.2-1987 command mode
  - Command: MARKer[<chno>]:SEARch:TARGet[:MODE] <type>
  - Parameter: <type> = {ZERO|PI|VALue}
  - Response type: ZERO|PI|VALue
- IEEE488.1-1987 command mode
  - Command: ZRPSRCH
  - Response type:
    - NR3 (real value): Setting value (stimulus value)
    - NR3 (real value): Measurement value (data A, B, C)
    - NR1 (integer value): Status
- Description: Specifies a mode of the target search.

R3762/63 command mode	R3764/66, R3765/67 command parameter	Mode
ZRPSRCH	ZERO	Searches the phase of 0deg.
	PI	Searches the phase of ±180deg.
	VAL	Searches the specified value.

26. MARKer[<chno>]:SEARch:TARGet:VALue

• Function	Value specification of the target search
• Presence of command and query	Command / Query
• Command	MARKer[<chno>]:SEARch:TARGet:VALue <real>
• Parameter	<real>
• Response type	<NR3> real value
• Description	Specifies a value when the mode for the target search is set to search the specified value.

27. MARKer[<chno>]:SEARch:TARGet:LEFT

- Function: Left frequency searching
- Presence of command and query: Command
- Command: MARKer[<chno>]:SEARch:TARGet:LEFT
- Description: Searches the next frequency leftward in the target search mode.

28. MARKer[<chno>]:SEARch:TARGet:RIGHt

- Function: Right frequency searching
- Presence of command and query: Command
- Command: MARKer[<chno>]:SEARch:TARGet:RIGHt
- Description: Searches the next frequency rightward in the target search mode.

29. MARKer[<chno>]:SEARch:TRACking — IEEE488.1-1987 command mode MKRTRAC

- Function: ON/OFF of tracking mode
- Presence of command and query: Command / Query
- Command: MARKer[<chno>]:SEARch:TRACking <bool>
  MKRTRAC
- Parameter: <bool>
- Response type: 0 | 1
- Description: When the tracking mode is;

  ON: Marker search is executed every time a sweep ends.
  Note: Set the tracking mode to ON before specifying the marker search.

  OFF: Marker search is executed only once when the marker search is specified.

7.18 FETCh? Subsystem

1. FETCh[<chno>][:MARKer][:ACTivate]?

- Function — Active marker output
- Presence of command and query — Query
- Command — FETCh[<chno>][:MARKer][:ACTivate]?
- Description — Outputs the latest data of the active marker.
  The output data is transferred in the ASCII format.

Output format

SN.NNNNNNNNNNNNNNNNESNN, SN.NNNNNNNNNNNNNNNNESNN,
<Stimulus> <Data A>

SN.NNNNNNNNNNNNNNNNESNN, SN.NNNNNNNNNNNNNNNNESNN,
<Data B> <Data C>

SN NL^END
<Status>

<Stimulus> Shows the X axis value at the marker point.

The format is the following fixed length format of 22 characters.

SN.NNNNNNNNNNNNNNNNESNN
(S: +/-, N:0 to 9, E: Exponent characteristic)

If the active marker is disabled, the stimulus is +1.000000000000000E+38.

If the delta marker is enabled, the stimulus is the difference between the markers.

<Data A, B> The data A is the operation data of the first waveform. The data B is the operation data of the second waveform.
The memory waveform is the data B.

When the polar coordinates or the smith chart display is set, the data A is the value for the real part and the data B is the value for the imaginary part.

The data format is the same as that of the stimulus.
If no available data, the data A and B are +1.000000000000000E+38.

<Data C> The data C is available when the polar coordinates or the smith chart display is set. In this case, the data c is the reactance value or the capacitance value.

The data format is the same as that of the stimulus.
If no available data, the data C is +1.000000000000000E+38.

<Status> The status of the operation data is as follows.

-1: No data.
0: Data for the normal operation.
1: Measurement data cannot be operated.
2: Level 1 error in the filter analysis.
3: Level 2 error in the filter analysis.
4: Level 3 error in the filter analysis.
5: Level 4 error in the filter analysis.

The status is in the format of 1 or 2 integers.

2. FETCh[<chno>][:MARKer]:FANalysis?

- Function: Filter analysis output
- Presence of command and query: Query
- Command: FETCh[<chno>][:MARKer]:FANalysis?
- Description: Outputs the results for the filter analysis.

The filter analysis is executed with the first waveform data. If the data waveform is OFF, however, the memory waveform data is used.

The output data is transferred in the ASCII format.

Output format

SN.NNNNNNNNNNNNNNNNESNN, SN.NNNNNNNNNNNNNNNNESNN,
<CENTER FREQ> <LEFT FREQ>

SN.NNNNNNNNNNNNNNNNESNN, SN.NNNNNNNNNNNNNNNNESNN,
<LIGHT FREQ> <BANDWIDTH>

SN.NNNNNNNNNNNNNNNNESNN, SN.NNNNNNNNNNNNNNNNESNN,
<QUALITYFACTOR> <SHAPEFACTOR>

SN NL^END
<Status>

<CENTER FREQ> Center frequency of the filter

The format is the following fixed length format of 22 characters.

SN.NNNNNNNNNNNNNNNNESNN
(S: +/-, N:0 to 9, E: Exponent characteristic)

If the active marker is disabled, the CENTER FREQ is +1.000000000000000E+ 38.
If the delta marker is enabled, the frequency difference between the markers cannot be transferred.

<LEFT FREQ>	Left frequency of the searched bandwidth The format is the same as that of the CENTER FREQ. If no available data, the LEFT FREQ is +1.000000000000000E+38.
<LIGHT FREQ>	Right frequency of the searched bandwidth The format is the same as that of the CENTER FREQ. If no available data, the RIGHT FREQ is +1.000000000000000E+38.
<BANDWIDTH>	Searched bandwidth The format is the same as that of the CENTER FREQ. If no available data, the BANDWIDTH is +1.000000000000000E+38.
<QUALITYFACTOR>	Quality factor The format is the same as that of the CENTER FREQ. If no available data, the QUALITYFACTOR is +1.000000000000000E+38.
<SHAPEFACTOR>	Selectivity The format is the same as that of the CENTER FREQ. If no available data, the SHAPEFACTOR is +1.000000000000000E+38.
<Status>	The status of the operation data is as follows. -1: No data. 0: Data for the normal operation. 1: Measurement data cannot be operated. The status is in the format of 1 or 2 integers.

3. FETCh[<chno>][:MARKer]:NUMBer<n>?

• Function	Data output of the specified marker.
• Presence of command and query	Query
• Command	FETCh[<chno>][:MARKer]:NUMBer<n>?
• Parameter	<n> =0 to 10
• Description	Outputs the marker data of the specified number. Number 0 is the active marker. The format is the same as that of the active marker output.

7.19 LIMit Subsystem

1.	DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:BEEP	IEEE488.1-1987 command mode FAILBEEP

- Function: ON/OFF of beep sound at the time of limit FAIL
- Presence of command and query: Command / Query
- IEEE488.2-1987 command mode
  - Command: DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:BEEP <bool>
  - Parameter: <bool>
  - Response type: 0 | 1
- IEEE488.1-1987 command mode
  - Command: FAILBEEP<bool>
  - Parameter: <bool>
  - Response type: 0 | 1
- Description: Selects ON or OFF of beep sound at the time of limit test FAIL.

  When the limit test function (DISP:LIM) is ON and the beep function (SYST:BEEP:STAT) is ON, the beep sound is available by setting this command to ON.
  Specifying the parameter (parano) is invalid.

2. DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:CLEar   IEEE488.1-1987 command mode LSEGCL

- Function: Clear of all segments in the limit table
- Presence of command and query: Command
- IEEE488.2-1987 command mode
  Command: DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:CLEar
- IEEE488.1-1987 command mode
  Command: LSEGCL
- Description: Clears the contents of all segments in the limit table.

  Two groups (parameter) of limit table exist in each channel.
  When the second parameter table is to be the target, specify 2 for <parano>.
  In order to clear the segment partially, use DISP:LIM:SEGM:DEL.

3. DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:DATA<block>

- Function — Information setting of all segments in the limit table.
- Presence of command and query — Command / Query
- IEEE488.2-1987 command mode

Command: DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:DATA <block>

Parameter:

<block> = #<byte><length><data>
<byte> = Describes byte number of the next character string showing the block length with ASCII numeral (1 character).
<length> = Describes character number of the character string showing the data string with ASCII numeral.
<data> = Describes each element of all the necessary segments in order of <stimulus>, <upper>, <lower>, <type>, <color>, <wcolor>...
<stimulus> = Stimulus value
<upper> = Upper limit value
<lower> = Lower limit value
<type> = Line type{SLINe |FLINe |SPOint}
<color> = Limit line display color {1-7}
<wcolor> = Display color of signal waveform {1-7}

Response type: <block>

- Description — Sets all segment information of the limit table in perfect form. The last segment information is lost.
Receiving all the data, sorts the segments in ascending sequence of stimulus value.
If some description error is found in the data, the segments up to there is valid, but the rest is ignored.
Refer to "3.1.2 Data Format" for block data <block>.

- Example — LISP: LIM:DATA #2463GHz,5dB,-5dB,SLIN,2,6,6GHz,10dB,-10dB,
SPO,2,6

4. DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:LINE   IEEE488.1-1987 command mode
LIMILINE

- Function   ON/OFF of limit line screen display

- Presence of command and query   Command / Query

- IEEE488.2-1987 command mode
  - Command   DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:LINE<bool>
  - Parameter   <bool>
  - Response type   0 | 1

- IEEE488.1-1987 command mode
  - Command   LIMILINE<bool>
  - Parameter   <bool>
  - Response type   0 | 1

- Description   Selects ON/OFF of limit line screen display.
Setting this command to ON, the limit line is displayed on the display scale.
In order to perform the limit test, it is necessary to set DISP:LIM to ON.

5.	DISPlay[:WINDow[<chno>]]:LIMit[<parano>] :OFFSet:AMPLitude	IEEE488.1-1987 command mode LIMIAMPO

- Function — Adding/Subtracting offset value to/from the all segment limit values.

- Presence of command and query — Command / Query

- IEEE488.2-1987 command mode
  - Command — DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:OFFSet:AMPLitude <real>
  - Parameter — <real>
  - Response type — NR3 (real value)

- IEEE488.1-1987 command mode
  - Command — LIMIAMPO
  - Parameter — <real>
  - Response type — NR3 (real value)

- Description — Moves the limit line up and down according to the specified offset value.
  In order to add the offset value to the stimulus value, use DISP:LIM:OFFS:STIM command.

6. DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:OFFSet:STIMulus <real> — IEEE488.1-1987 command mode LIMISTIO

- Function: Adding/Subtracting offset value to/from the all segment stimulus values.

- Presence of command and query: Command / Query

- IEEE488.2-1987 command mode
  - Command: DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:OFFSet:STIMulus <real>
  - Parameter: <real>
  - Response type: NR3 (real value)

- IEEE488.1-1987 command mode
  - Command: LIMISTIO<real>
  - Parameter: <real>
  - Response type: NR3 (real value)

- Description: Moves the limit line up and down according to the specified offset value.
In order to add the offset value to the response value, use DISP:LIM:OFFS:AMPL command.

7. DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:ParallelIO — IEEE488.1-1987 command mode LIMPIO

- Function: ON/OFF of limit line judged result output to parallel I/O.

- Presence of command and query: Command / Query

- IEEE488.2-1987 command mode
  - Command: DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:ParallelIO <bool>
  - Parameter: <bool>
  - Response type: 0 | 1

- IEEE488.1-1987 command mode
  - Command: LIMPIO<bool>
  - Parameter: <bool>
  - Response type: 0 | 1

- Description: Selects ON/OFF of the limit test result output to parallel I/O (PIO). If this command is set to ON when the limit test (DISP:LIM) is ON, the result output to PIO is enabled.

8. DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:PARameter:PolarLIMit — IEEE488.1-1987 command mode LIMPLIN|LIMPLOG

- Function: Selecting the combination of judgement parameter when Polar is displayed on the format.

- Presence of command and query: Command / Query

- IEEE488.2-1987 command mode
  - Command: DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:PARameter:PolarLIMit <select>
  - Parameter: <select> = {LINear|LOGarithmic}
  - Response type: LIN|LOG

- IEEE488.1-1987 command mode
  - Command: LIMPLIN|LIMPLOG
  - Response type: 0 | 1

- Description: When Polar display (CALCulate[:FORMat]POLar) is selected on the display format, the judgement parameter becomes the combination of amplitude and phase.
  This command selects the amplitude of linear type or log type.

R3762/63 command	R3764/66, R3765/67 command parameter	Judgement parameter <parano>
LIMPLIN	LINear	0; Amplitude (Linear) 1; Phase
LIMPLOG	LOGarithmic	0; Amplitude (Log) 1; Phase

If the display format of the corresponding channel is the type of rectangular coordinates, the setting here has no influence.

9. DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:PARameter:SmithLIMit — IEEE488.1-1987 command mode LIMSLIN|LIMSLOG

- Function — Selecting the combination of judgement parameter when Smith is displayed on the format.

- Presence of command and query — Command / Query

- IEEE488.2-1987 command mode
  - Command — DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:PARameter:SmithLIMit <select>
  - Parameter — <select> = {LINear|LOGarithmic}
  - Response type — LIN|LOG

- IEEE488.1-1987 command mode
  - Command — LIMSLIN|LIMSLOG
  - Response type — 0 | 1

- Description — When Smith chart (CALCulate[:FORMat]SCHart|ISCHart) is selected on the display format, the judgement parameter becomes the combination of amplitude and phase.
This command selects the amplitude of linear type or log type.

R3762/63 command	R3764/66, R3765/67 command parameter	Judgement parameter <parano>
LIMPLIN	LINear	0; Amplitude (Linear) 1; Phase
LIMPLOG	LOGarithmic	0; Amplitude (Log) 1; Phase

If the display format of the corresponding channel is the type of rectangular coordinates, the setting here has no influence.

10.	DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:PARameter[:STATE]	IEEE488.1-1987 command mode LIMPAR

● Function — ON/OFF of each judgement parameter setting

● Presence of command and query — Command / Query

● IEEE488.2-1987 command mode

Command: DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:PARameter[:STATe] <bool>

Parameter: <bool>

Response type: 0 | 1

● IEEE488.1-1987 command mode

Command: LIMPAR<bool>

Parameter: <bool>

Response type: 0 | 1

● Description

Sets ON/OFF of each judgement parameter <parano>.

<parano>	Judgement parameter
0	Main trace/real part/amplitude
1	Sub trace/imaginary part/phase

In order to execute the limit test, set the test ON with DISP:LIM ON after setting the limit. Even if the parameter is set to ON, if no segment is set, the setting here becomes invalid.

11. DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:REPort?

- Function — Reporting PASS/FAIL information of all segments

- Presence of command and query — Query

- IEEE488.2-1987 command mode

  Query — DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:REPort?

  Response type — <block>

  The output type is different according to the data format setting (FORMat[:DATA]).

  In case of ASCII format (FORMat[:DATA] ASCii).
  <block> = <segment>[,<segment>,....]
  <segment> =0 to 30 numeral (ASCII character string)

  In case of binary format (FORMat[:DATA] {REAL|MBIN}, {32|64}).

  <block> = #<byte>[<length>]<data>
  <byte> = Specifies the character number of the character string showing the block length, with 1 character of ASCII numeral.
  <length> = Specifies the character number of the character string showing the data length, with ASCII numeral.
  <data> = Segment number of FAIL (Order of 1 byte integer, ascending order)

- Description — Reports PASS/FAIL information about all segments together.
  In order to know the test results, use DISP:LIM:RES?.

  Refer to "3.1.2 Data Format" for the block data <block>.
  Refer to "7.7.2 FORMat[:DATA]" for the data format.

12.	DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:RESult?	IEEE488.1-1987 command mode LIMRES?

- Function — Reporting PASS/FAIL information of test results
- Presence of command and query — Query
- IEEE488.2-1987 command mode
  - Query — DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:RESult?
  - Response type — PASS|FAIL|OFF|UND
- IEEE488.1-1987 command mode
  - Query — LIMRES?
  - Response type — PASS|FAIL|OFF|UND
- Description — Returns the test results of PASS/FAIL.
  But when the limit test is OFF, returns OFF, and when the limit value is undefined, returns UNDefined.

13. DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n>

- Function: Setting all information of the specified segment together
- Presence of command and query: Command / Query
- IEEE488.2-1987 command mode

Query: DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n> <block>

Parameter:

<block>	=	#<byte><length><data>
<byte>	=	Describes byte number of the next character string showing the block length with ASCII numeral (1 character).
<length>	=	Describes character number of the character string showing the data string with ASCII numeral.
<data>	=	Describes each element of all the necessary segments in order of <stimulus>, <upper>, <lower>, <type>, <color>, <wcolor>, ...
<stimulus>	=	Stimulus value
<upper>	=	Upper limit value
<lower>	=	Lower limit value
<type>	=	Line type {SLINe |FLINe |SPOint}
<color>	=	Limit line display color {1-7}
<wcolor>	=	Display color of signal waveform {1-7}

Response type: <block>

- Description: Sets the necessary information for 1 segment all together. If the specified segment is not empty, a new content is overwritten.
  Segment number <n> is set within the limits of 0 to 30.
  When all the data is received, the order of the segments is changed to the ascending order of stimulus values.
  If the specified segments are more than the segment number specified beforehand, the specification of segment is ignored and they are added to the original empty segment.
- Example: DISP:LIM:SEGM1 #2224GHz, 5dB,-5dB, SLIN, 2, 6

14.	DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n>:COLor	IEEE488.1-1987 command mode LIMC

- Function — Setting the line color of the specified segment

- Presence of command and query — Command / Query

- IEEE488.2-1987 command mode
  - Command: DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n>:COLor<int>
  - Parameter: <int>
  - Response type: NR1 (integer value)

- IEEE488.1-1987 command mode
  - Command: LIMC<int>
  - Parameter: <int>
  - Response type: NR1 (integer value)

- Description — Sets the limit line display colors of the specified segments.

Parameter	Display color
1	Gray
2	Red
3	Purple
4	Green
5	Blue
6	Yellow
7	White

In IEEE488.2-1987 command mode, segment number <n> is specified within the limits of 0 to 30. In IEEE488.1-1987 command mode, the segment number is specified by LSEG command in advance.

15. DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n>:DELete

- Function: Deleting the contents of the specified segment
- Presence of command and query: Command
- IEEE488.2-1987 command mode
  - Command: DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n>:DELete
- Description: Deletes the specified segment and shifts up the next segment.
  In order to delete the previous segment, use DISP:LIM:CLEar.

16. DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n>:LOWer — IEEE488.1-1987 command mode: LIML

- Function: Setting the lower limit value of the specified segment
- Presence of command and query: Command / Query
- IEEE488.2-1987 command mode
  - Command: DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n>:LOWer<real>
  - Parameter: <real>
  - Response type: NR3 (real value)
- IEEE488.1-1987 command mode
  - Command: LIML<real>
    Refer to "7.19.25 LSEG" too.
  - Parameter: <real>
  - Response type: NR3 (real value)
- Description: Sets the limit judgement lower limit value of the specified segment.
  If the specified lower limit value is larger than the upper limit value, the lower limit value becomes the same as the upper limit value.
  In IEEE488.2-1987 command mode, segment number <n> is specified within the limits of 0 to 30. In IEEE488.1-1987 command mode, the segment number is specified by LSEG command in advance.

17. DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n>:LOWer:REPort?

- Function: Reporting the information of the point where FAIL occurred at the lower limit value of the specified segment

- Presence of command and query: Query

- IEEE488.2-1987 command mode

  Query: DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n>:LOWer:REPort?

  Response type: <block>

  The output form is different according to the data format setting (FORMat[:DATA]).

  In case of ASCII format (FORMat[:DATA] ASCii).

  <block> = <point>[,<point>,....]

  <point> = <stimulus>,<amplitude>,<failed> (ASCII character string)

  In case of binary format (FORMat[:DATA] {REAL|MBIN}, {32|64}).

  <block> = #<byte><length>[<point>...]

  <byte> = Specifies the character number of the character string showing the block length, with 1 character of ASCII numeral.

  <length> = Specifies the character number of the character string showing the data length, with ASCII numeral.

  <point> = <stimulus>,<amplitude>,<failed> (binary format)

  <stimulus> = Stimulus value of FAIL point <real>

  <amplitude> = Response value of FAIL point <real>

  <failed> = The difference between the response value and the lower limit value <real>

- Description: Reports the information of the point where FAIL occurred at the lower limit value of the specified segment.

  The output data format follows the specification of FORM[:DATA] command.

  The units of the stimulus value and the response value correspond to the current display format.

  Refer to "3.1.2 Data Format" for the block data <block>.

  Refer to "7.7.2 FORMat[:DATA]" for the data format.

18. DISPLAY[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n>:REPort?

- Function: Reporting the information of the point where FAIL of the specified segment occurred

- Presence of command and query: Query

- IEEE488.2-1987 command mode

  Query: DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n>:REPort?

  Response type: <block>

  The output form is different according to the data format setting (FORMat[:DATA]).

  In case of ASCII format (FORMat[:DATA] ASCii).

  <block> = <point>[,<point>,....]

  <point> = <stimulus>,<amplitude>,<failed> (ASCII character string)

  In case of binary format (FORMat[:DATA] {REAL|MBIN}, {32|64}).

  <block> = #<byte><length>[<point>...]

  <byte> = Specifies the character number of the character string showing the block length, with 1 character of ASCII numeral.

  <length> = Specifies the character number of the character string showing the data length, with ASCII numeral.

  <point> = <stimulus>,<amplitude>,<failed> (binary format)

  <stimulus> = Stimulus value of FAIL point <real>

  <amplitude> = Response value of FAIL point <real>

  <failed> = The difference between the response value and the lower limit value <real>

- Description: Reports the information of the point where FAIL of the specified segment occurred.

  The output data format follows the specification of FORM[:DATA] command.

  The units of the stimulus value and the response value correspond to the current display format.

  Refer to "3.1.2 Data Format" for the block data <block>.

  Refer to "7.7.2 FORMat[:DATA]" for the data format.

19. DISPLAY[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n>:STIMulus — IEEE488.1-1987 command mode LSTIM

- Function: Setting the stimulus value of the specified segment
- Presence of command and query: Command / Query
- IEEE488.2-1987 command mode
  - Command: DISPLAY[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n>:STIMulus<real>
  - Parameter: <real>
  - Response type: NR3 (real value)
- IEEE488.1-1987 command mode
  - Command: LSTIM<real>
    Refer to "7.19.25 LSEG" too.
  - Response type: NR3 (real value)
- Description: Sets the stimulus value of the specified segment.
  In IEEE488.2-1987 command mode, segment number <n> is specified within the limits of 0 to 30. In IEEE488.1-1987 command mode, the segment number is specified by LSEG command in advance.

20. DISPLAY[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n>:TYPE — IEEE488.1-1987 command mode LIMTFLT|LIMTSLP|LIMTSP

- Function — Setting the line type of the specified segment.
- Presence of command and query — Command / Query
- IEEE488.2-1987 command mode
  - Command — DISPLAY[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n>:TYPE<type>
  - Parameter — <type> = SLINe|FLINe|SPOint
  - Response type — SLIN|FLIN|SPO
- IEEE488.1-1987 command mode
  - Command — LIMTFLT|LIMTSLP|LIMTSP
  - Response type — 0 | 1
- Description — Sets the line type of the specified segment.

R3762/63 command	R3764/66, R3765/67 command parameter	Type
LIMTFLT	FLINe	Flat line
LIMTSLP	SLINe	Slope line
LIMTSP	SPOint	Single line

If other than single point is selected in the polar coordinate display format, the same limit value is adapted for all measurement points in the segment.
In IEEE488.2-1987 command mode, segment number <n> is specified within the limits of 0 to 30. In IEEE488.1-1987 command mode, the segment number is specified by LSEG command in advance.

21. DISPLAY[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n>:UPPer — IEEE488.1-1987 command mode LIMU

- Function: Setting the upper limit value of the specified segment

- Presence of command and query: Command / Query

- IEEE488.2-1987 command mode

Command	DISPLAY[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n>:UPPer<real>
Parameter	<real>
Response type	NR3 (real value)

- IEEE488.1-1987 command mode

Command	LSTIM<real> Refer to "7.19.25 LSEG" too.
Parameter	<real>
Response type	NR3 (real value)

- Description: Sets the limit judgement upper limit value of the specified segment.
  If the specified upper limit value is smaller than the lower limit value, the upper limit value becomes the same as the lower limit value.
  In IEEE488.2-1987 command mode, segment number <n> is specified within the limits of 0 to 30. In IEEE488.1-1987 command mode, the segment number is specified by LSEG command in advance.

22. DISPLAY[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n>:UPPer:REPort?

- Function: Reporting the information of the point where FAIL occurred at the upper limit value of the specified segment

- Presence of command and query: Query

- IEEE488.2-1987 command mode

  Query: DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n> :UPPer:REPort?

  Response type: <block>

  The output form is different according to the data format setting (FORMat[:DATA]).

  In case of ASCII format (FORMat[:DATA] ASCii).

  <block> = <point>[,<point>,....]

  <point> = <stimulus>,<amplitude>,<failed> (ASCII character string)

  In case of binary format (FORMat[:DATA] {REAL|MBIN}, {32|64}).

  <block> = #<byte><length>[<point>...]

  <byte> = Specifies the character number of the character string showing the block length, with 1 character of ASCII numeral.

  <length> = Specifies the character number of the character string showing the data length, with ASCII numeral.

  <point> = <stimulus>,<amplitude>,<failed> (binary format)

  <stimulus> = Stimulus value of FAIL point <real>

  <amplitude> = Response value of FAIL point <real>

  <failed> = The difference between the response value and the upper limit value <real>

- Description: Reports the information of the point where FAIL occurred at the upper limit value of the specified segment.

  The output data format follows the specification of FORM[:DATA] command.

  The units of the stimulus value and the response value correspond to the current display format.

  Refer to "3.1.2 Data Format" for the block data <block>.

  Refer to "7.7.2 FORMat[:DATA]" for the data format.

23. DISPLAY[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n>:WCOLor — IEEE488.1-1987 command mode LIMWC

- Function — Setting the waveform color in the specified segment

- Presence of command and query — Command / Query

- IEEE488.2-1987 command mode
  - Command — DISPLAY[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n>:WCOLor<int>
  - Parameter — <int>
  - Response type — NR1 (integer value)

- IEEE488.1-1987 command mode
  - Command — LIMWC<int>
    Refer to "7.19.25 LSEG" too.
  - Parameter — <int>
  - Response type — NR1 (integer value)

- Description

Sets the display color of measurement waveform in the specified segment.
Within the stimulus limit of the segment, if the judgement is within the limit of PASS, the measurement waveform is displayed with the color specified here. If the judgment is within the limit of FAIL, the measurement waveform is displayed with red regardless of the setting here.

Parameter	Display color
1	Gray
2	Red
3	Purple
4	Green
5	Blue
6	Yellow
7	White

In IEEE488.2-1987 command mode, segment number <n> is specified within the limits of 0 to 30. In IEEE488.1-1987 command mode, the segment number is specified by LSEG command in advance.

24. DISPLAY[:WINDow[<chno>]]:LIMit[<parano>][:STATe] — IEEE488.1-1987 command mode LIMITEST

● Function — ON/OFF of the limit test function

● Presence of command and query — Command / Query

● IEEE488.2-1987 command mode

Command	DISPLAY[:WINDow[<chno>]]:LIMit[<parano>][:STATe] <bool>
Parameter	<bool>
Response type	0 | 1

● IEEE488.1-1987 command mode

Command	LIMITEST<bool>
Parameter	<bool>
Response type	0 | 1

● Description

When the limit test is set to ON, the judgement of trace data is performed with the set limit value.
In order to display the limit line on the screen, DISP:LIM:LINE must be set to ON.
The specification of parameter <parano> is invalid.

25. IEEE488.1-1987 command mode
LSEG

- Function: Selecting a segment number
- Presence of command and query: Command / Query
- IEEE488.1-1987 command mode
  - Command: LSEG<int>
  - Parameter: <int> = 0 to 30
  - Response type: NR1 (integer value)
- Description: Specifies a segment number in IEEE488.1-1987 command mode.
  In the set commands of segment, LIMC, LIML, LSTIM, LIMIT, LIMU and LIMWC, the setting is performed for the segment numbers specified here.
  In IEEE488.2-1987 command mode, the setting by this command is invalid because the setting follows the segment by the header parameter <n> in each command.

MEMO

APPENDIX

A.1 List of Command

A1.1 Common Commands

*CLS

*DDT <blk>
*DMC <str>,<blk>

*EMC <num>
*ESE <num>
*ESR?

*GMC? <name>

*IDN?

*LMC?

*OPC

*PCB <primary>[,<secondary>]
*PMC

*RCL{<num>|POFF}
*RST

*SAV <num>
*SRE <num>
*STB?

*TRG
*TST?

*WAI

A1.2 R3764/66, R3765/67 Commands

ABORt

CALCulate[<chno>]:FORMat{MLOGarithmic|PHASe|GDELay|POLar|MLINear|SWR|REAL
|IMAGinary|UPHase|SCHart|ISCHart|MLIPhase|MLOPhase|MLODeley}
CALCulate[<chno>]:GDAPerture:APERture <real>
CALCulate[<chno>]:MATH[:EXPRession]:NAME{NONE|DDM|DMM|DAM|DSM}
CALCulate[<chno>]:SMOothing:APERture <real>
CALCulate[<chno>]:SMOothing:STATe <bool>
CALCulate[<chno>]:TRANsform:IMPedance:CIMPedance <real>
CALCulate[<chno>]:TRANsform:IMPedance:TYPE{NONE|ZREFlection|YREFlecion|ZTRansmit|
|YTRansmit|INVersion}

DISPlay:ACTive <int>
DISPlay:DUAL <bool>
DISPlay:FORMat{ULOWer|FBACk}
DISPlay[:WINDow[<chno>]]:LIMit[pn]:BEEP <bool>
DISPlay[:WINDow[<chno>]]:LIMit[pn]:CLEar
DISPlay[:WINDow[<chno>]]:LIMit[pn]:DATA <block>
DISPlay[:WINDow[<chno>]]:LIMit[pn]:LINE <bool>
DISPlay[:WINDow[<chno>]]:LIMit[pn]:OFFSet:AMPLitude <real>
DISPlay[:WINDow[<chno>]]:LIMit[pn]:OFFSet:AMPLitude <real>
DISPlay[:WINDow[<chno>]]:LIMit[pn]:OFFSet:STIMulus <real>
DISPlay[:WINDow[<chno>]]:LIMit[pn]:OFFSet:STIMulus <real>
DISPlay[:WINDow[<chno>]]:LIMit[pn]:PARameter[:STATe] <bool>
DISPlay[:WINDow[<chno>]]:LIMit[pn]:PARameter:PLIMit{LINear|LOGarithmic}
DISPlay[:WINDow[<chno>]]:LIMit[pn]:PARameter:SLIMit{LINear|LOGarithmic}
DISPlay[:WINDow[<chno>]]:LIMit[pn]:ParallelIO <bool>
DISPlay[:WINDow[<chno>]]:LIMit[pn]:REPort?
DISPlay[:WINDow[<chno>]]:LIMit[pn]:RESult?
DISPlay[:WINDow[<chno>]]:LIMit[pn]:SEGMent<n> <block>
DISPlay[:WINDow[<chno>]]:LIMit[pn]:SEGMent<n>:COLor <int>
DISPlay[:WINDow[<chno>]]:LIMit[pn]:SEGMent<n>:DEL
DISPlay[:WINDow[<chno>]]:LIMit[pn]:SEGMent<n>:LOWer <real>
DISPlay[:WINDow[<chno>]]:LIMit[pn]:SEGMent<n>:LOWer:REPort?
DISPlay[:WINDow[<chno>]]:LIMit[pn]:SEGMent<n>:REPort?
DISPlay[:WINDow[<chno>]]:LIMit[pn]:SEGMent<n>:STIMulus <real>
DISPlay[:WINDow[<chno>]]:LIMit[pn]:SEGMent<n>:TYPE{SLINe|FLINe|SPOint}
DISPlay[:WINDow[<chno>]]:LIMit[pn]:SEGMent<n>:UPPer <real>
DISPlay[:WINDow[<chno>]]:LIMit[pn]:SEGMent<n>:UPPer:REPort?
DISPlay[:WINDow[<chno>]]:LIMit[pn][:STATe] <bool>
DISPlay[:WINDow[<chno>]]:TEXT[:DATA]{<str>|<block>}

```
DISPlay[:WINDow[<chno>]]:TRACe:ASSign{DATA|MEMory|DMEMory}
DISPlay[:WINDow[<chno>]]:TRACe:GRATiclue[:STATe] <bool>
DISPlay[:WINDow[<chno>]]:Y[trace]:RLINe <bool>
DISPlay[:WINDow[<chno>]]:Y[trace][:SCALe]:AUTO ONCE
DISPlay[:WINDow[<chno>]]:Y[trace][:SCALe]:PDIVision <real>
DISPlay[:WINDow[<chno>]]:Y[trace][:SCALe]:RLEVel <real>
DISPlay[:WINDow[<chno>]]:Y[trace][:SCALe]:RPOSition <real>

FETCh[<chno>][:MARKer]:FANalysis?
FETCh[<chno>][:MARKer]:NUMBer<n>?
FETCh[<chno>][:MARKer][:ACTivate]?
FILE:DELete <str>
FILE:LOAD <str>
FILE:STATe:CONDition <bool>
FILE:STATe:CORRection <bool>
FILE:STATe:DATA <bool>
FILE:STATe:MEMory <bool>
FILE:STATe:RAW <bool>
FILE:STORe <str>
FORMat:BORDer{NORMal|SWAPped}
FORMat[:DATA]{ASCii|REAL,32|REAL,64|MBINary,32|MBINary,64}

INITiate:CONTinuous <bool>
INITiate[:IMMediate]

MARKer[<chno>]:ACTivate:STATe <bool>
MARKer[<chno>]:ACTivate:STIMulus <real>
MARKer[<chno>]:ACTivate[:NUMBer] <n>[,<real>]
MARKer[<chno>]:AOFF
MARKer[<chno>]:COMPensate <bool>
MARKer[<chno>]:CONVert[:MODE]{DEFault|LINear|RIMaginary}
MARKer[<chno>]:COUPle <bool>
MARKer[<chno>]:DELTa:COMPare <n>[,<real>]
MARKer[<chno>]:DELTa[:MODE]{OFF|CHIld|COMPare|FIXed}
MARKer[<chno>]:FANalysis:DIRection{IN|OUT}
MARKer[<chno>]:FANalysis:WIDTh <real>
MARKer[<chno>]:FANalysis[:STATe] <bool>
MARKer[<chno>]:FIXed:AVALue <real>
MARKer[<chno>]:FIXed:STIMulus <real>
MARKer[<chno>]:FIXed:VALue <real>
MARKer[<chno>]:LET{STARt|STOP|CENTer|SPAN|RLEVel|FIXed}
```

MARKer[<chno>]:LIST <bool>
MARKer[<chno>]:POLar{MLINear|MLOGarithmic|RIMaginary}
MARKer[<chno>]:SEARch:PARTial:SRANge
MARKer[<chno>]:SEARch:PARTial[:STATe] <bool>
MARKer[<chno>]:SEARch:RIPPle:DX <real>
MARKer[<chno>]:SEARch:RIPPle:DY <real>
MARKer[<chno>]:SEARch:RIPPle[:MODE]{MAX|MIN|BOTH|PPEak}
MARKer[<chno>]:SEARch:TARGet:LEFT
MARKer[<chno>]:SEARch:TARGet:RIGHt
MARKer[<chno>]:SEARch:TARGet:VALue <real>
MARKer[<chno>]:SEARch:TARGet[:MODE]{ZERO|PI|VALue}
MARKer[<chno>]:SEARch:TRACking <bool>
MARKer[<chno>]:SEARch[:MODE]{OFF|MAX|MIN|TARGet|RIPPle}
MARKer[<chno>]:SMITH{MLINear|MLOGarithmic|RIMaginary|IMPedance|ADMittance}

REGister:CLEar <int>
REGister:RECall{<int>|POFF}
REGister:SAVE <int>

[SENSe:]AVERage[<chno>]:COUNt <int>
[SENSe:]AVERage[<chno>]:RESTart
[SENSe:]AVERage[<chno>][:STATe] <bool>
[SENSe:]BANDwidth[<chno>][:RESolution] <int>
[SENSe:]BANDwidth[<chno>][:RESolution]:AUTO <bool>
[SENSe:]CORRection[<chno>]:CKIT:TERMinal[port]{MALe|FEMale}
[SENSe:]CORRection[<chno>]:CKIT[:TYPE]{0-4}
[SENSe:]CORRection[<chno>]:COLLect:DELete
[SENSe:]CORRection[<chno>]:COLLect:SAVE
[SENSe:]CORRection[<chno>]:COLLect[:ACQuire]{NORMalize|SNORromalize|OPEN|SHORt
|LOAD|S11Oopen|S11Sshort|S11Load
|S22Oopen|S22Sshort|S22Load|FTRansmit
|FMATch|RTRansmit|RMATch|GTHRU
|OISolation|FISolation|RISolation}
[SENSe:]CORRection[<chno>]:CSET:INTerpolate <bool>
[SENSe:]CORRection[<chno>]:CSET:STATe <bool>
[SENSe:]CORRection[<chno>]:EDELay:DISTance <real>
[SENSe:]CORRection[<chno>]:EDELay:STATe <bool>
[SENSe:]CORRection[<chno>]:EDELay[:TIME] <real>
[SENSe:]CORRection[n]:GPHase:STATe <bool>
[SENSe:]CORRection[<chno>]:OFFSet:PHASe <real>
[SENSe:]CORRection[<chno>]:OFFSet:STATe <bool>

[SENSe:]CORRection[<chno>]:PEXTension:STATe <bool>
[SENSe:]CORRection[<chno>]:PEXTension:TIME[eport] <real>
[SENSe:]CORRection[<chno>]:RVELocity:COAX <real>
[SENSe:]CORRection[n]:GPHase:STATe <bool>
[SENSe:]FUNCtion[<chno>]:POWer{AR|BR|AB|R|A|B|BDC|BDCR|S11|S21|S12|S22
|SF WD|SREV|NONE}
[SENSe:]FUNCtion[<chno>][:ON]{"POWer:{AC|DC}{1|2|3}"|"POWer:RATio:{AC|DC}
{2,1|3,1|2,3}"|"POWer:{S11|S12|S21|S22}"
|"POWer:{SFWD|SREV}"|"POWer:NONE"}
[SOURce:]POWer[<chno>]:BANDwidth[n] <int>
[SOURce:]CORRection[n]:GAIN:STATe <bool>
[SOURce:]COUPle <bool>
[SOURce:]FREQuency[<chno>]:CENTer <real>
[SOURce:]FREQuency[<chno>]:CW <real>
[SOURce:]FREQuency[<chno>]:MODE SWEep
[SOURce:]FREQuency[<chno>]:SPAN <real>
[SOURce:]FREQuency[<chno>]:STARt <real>
[SOURce:]FREQuency[<chno>]:STOP <real>
[SOURce:]POWer[<chno>]:STARt <real>
[SOURce:]POWer[<chno>]:STOP <real>
[SOURce:]POWer[<chno>]MODE SWEep
[SOURce:]POWer[<chno>][:LEVel][:AMPlitude] <real>
[SOURce:]PSWeep[<chno>]:CLEar
[SOURce:]PSWeep[<chno>]:CLEar:ALL
[SOURce:]PSWeep[<chno>]:FREQuency<n> <real>[,<real>]
[SOURce:]PSWeep[<chno>]:MODE{FREQuency|ALL}
[SOURce:]PSWeep[<chno>]:POINts<n> <int>
[SOURce:]PSWeep[<chno>]:POWer<n> <real>
[SOURce:]PSWeep[<chno>]:SETTling<n> <real>
[SOURce:]SWEep[<chno>]:POINts <num>
[SOURce:]SWEep[<chno>]:SPACing{LINear|LOGarithmic}
[SOURce:]SWEep[<chno>]:TIME <real>
[SOURce:]SWEep[<chno>]:TIME:AUTO <bool>

STATus:DEVice:CONDition?
STATus:DEVice:ENABle
STATus:DEVice[:EVENt]?
STATus:FREQuency:CONDition?
STATus:FREQuency:ENABle
STATus:FREQuency[:EVENt]?
STATus:LIMit:CONDition?

STATus:LIMit:ENABle
STATus:LIMit[:EVENt]?
STATus:OPERation:CONDition?
STATus:OPERation:ENABle <num>
STATus:OPERation[:EVENt]?
STATus:POWer:CONDition?
STATus:POWer:ENABle
STATus:POWer[:EVENt]?
STATus:QUEStionable:ENABle
STATus:QUEStionable[:EVENt]?
SYSTem:DATE <year>,<month>,<day>
SYSTem:ERRor?
SYSTem:PRESet
SYSTem:TIME <hour>,<minute>,<second>

TRACe[<chno>]:COPY DATA
TRACe[<chno>][:DATA]{<name>|<trace>},{<block>|<real>[,<real>...]}
TRACe[<chno>][:DATA]?{<name>|<trace>}[,{<name>|<trace>}...]
TRIGger[:SEQuence]:DELay <real>
TRIGger[:SEQuence]:DELay:STATe <bool>
TRIGger[:SEQuence]:SIGNal
TRIGger[:SEQuence]:SOURce{IMMediate|EXTernal|BUS|HOLD}
TRIGger[:SEQuence][:IMMediate]

A1.3 R3762/63 Commands

R3762/63 Commands	Corresponding R3764/66, R3765/67 commands
AB	[SENSe:]FUNCtion[<chno>]:POWer AB
ABIN	[SENSe:]FUNCtion[<chno>]:POWer AB
ADDRCONT<int>	*PCB <int>
AIN	[SENSe:]FUNCtion[<chno>]:POWer A
ALTAB	[SOURCe:]COUPle OFF
APERTP<real>	CALCulate[<chno>]:GDAPerture:APERture <real>
AR	[SENSe:]FUNCtion[<chno>]:POWer AR
ARIN	[SENSe:]FUNCtion[<chno>]:POWer AR
AUTO	DISPlay[:WINDow[<chno>]]:Y[trace][:SCALe]:AUTO ONCE
AVER<bool>	[SENSe:]AVERage[<chno>][:STATe] <bool>
AVERAGE	[SENSe:]AVERage[<chno>][:STATe] OFF
AVERFACT<int>	[SENSe:]AVERage[<chno>]:COUNt <int>
AVERREST	[SENSe:]AVERage[<chno>]:RESTart
AVR128	[SENSe:]AVERage[<chno>]:COUNt 128; STATe ON
AVR16	[SENSe:]AVERage[<chno>]:COUNt 16; STATe ON
AVR2	[SENSe:]AVERage[<chno>]:COUNt 2; STATe ON
AVR32	[SENSe:]AVERage[<chno>]:COUNt 32; STATe ON
AVR4	[SENSe:]AVERage[<chno>]:COUNt 4; STATe ON
AVR64	[SENSe:]AVERage[<chno>]:COUNt 64; STATe ON
AVR8	[SENSe:]AVERage[<chno>]:COUNt 8; STATe ON
BDCIN	[SENSe:]FUNCtion[<chno>]:POWer BDC
BDCRIN	[SENSe:]FUNCtion[<chno>]:POWer BDCR
BEEPFAIL<bool>	DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:BEEP <bool>
BIN	[SENSe:]FUNCtion[<chno>]:POWer B
BR	[SENSe:]FUNCtion[<chno>]:POWer BR
BRIN	[SENSe:]FUNCtion[<chno>]:POWer BR
CALN	[SENSe:]CORRection[<chno>]:CSET:STATe OFF
CENT<real>	[SOURce:]FREQuncy[<chno>]:CENTer <real>
CENTERF<real>	[SOURce:]FREQuncy[<chno>]:CENTer <real>
CH1	DISPlay:ACTive 1
CH2	DISPlay:ACTive 2
CH3	DISPlay:ACTive 3
CH4	DISPlay:ACTive 4
CHAN1	DISPlay:ACTive 1
CHAN2	DISPlay:ACTive 2

R3762/63 Commands	Corresponding R3764/66, R3765/67 commands
CKIT0	[SENSe:]CORRection[<chno>]:CKIT[:TYPE] 0
CKIT1	[SENSe:]CORRection[<chno>]:CKIT[:TYPE] 1
CKIT2	[SENSe:]CORRection[<chno>]:CKIT[:TYPE] 2
CKIT3	[SENSe:]CORRection[<chno>]:CKIT[:TYPE] 3
CKIT4	[SENSe:]CORRection[<chno>]:CKIT[:TYPE] 4
CKIT5	[SENSe:]CORRection[<chno>]:CKIT[:TYPE] 5
CLEA1	REGister:CLEar 1
CLEA2	REGister:CLEar 2
CLEA3	REGister:CLEar 3
CLEA4	REGister:CLEar 4
CLEA5	REGister:CLEar 5
CLEAR	[SENSe:]CORRection[<chno>]:COLLect:DELete
CLES	*CLS
CLRREG1	REGister:CLEar 1
CLRREG10	REGister:CLEar 10
CLRREG2	REGister:CLEar 2
CLRREG3	REGister:CLEar 3
CLRREG4	REGister:CLEar 4
CLRREG5	REGister:CLEar 5
CLRREG6	REGister:CLEar 6
CLRREG7	REGister:CLEar 7
CLRREG8	REGister:CLEar 8
CLRREG9	REGister:CLEar 9
CLS	*CLS
CONT	INITiate:CONTinuous ON
CONV1DS	CALCulate[<chno>]:TRANsform:IMPedance:TYPE INVersion
CONVOFF	CALCulate[<chno>]:TRANsform:IMPedance:TYPE NONE
CONVRY	CALCulate[<chno>]:TRANsform:IMPedance:TYPE YREFlection
CONVRZ	CALCulate[<chno>]:TRANsform:IMPedance:TYPE ZREFlection
CONVTY	CALCulate[<chno>]:TRANsform:IMPedance:TYPE YTRansmit
CONVTZ	CALCulate[<chno>]:TRANsform:IMPedance:TYPE ZTRansmnit
CONVYREF	CALCulate[<chno>]:TRANsform:IMPedance:TYPE YREFlection
CONVYTRA	CALCulate[<chno>]:TRANsform:IMPedance:TYPE YTRansmit
CONVZREF	CALCulate[<chno>]:TRANsform:IMPedance:TYPE ZREFlection
CONVZTRA	CALCulate[<chno>]:TRANsform:IMPedance:TYPE ZTRansmnit
CORARY<bool>	FILE:STATe:CORRection <bool>
CORR<bool>	[SENSe:]CORRection[<chno>]:CSET:STATe <bool>

R3762/63 Commands	Corresponding R3764/66, R3765/67 commands
CORRECT<bool>	[SENSe:]CORRection[<chno>]:CSET:STATe <bool>
COUC<bool>	[SOURCe:]COUPle <bool>
COUPLE<bool>	[SOURCe:]COUPle <bool>
CWFREQ<real>	[SOURce:]FREQuency[<chno>]:CW <real>
DATAARY<bool>	FILE:STATe:DATA <bool>
DATI	TRACe[<chno>]:COPY DATA
DAY<int>	SYSTem:DATE <year>,<month>,<day>
DELA	CALCualte[<chno>]:FORMat GDELay
DELAY	CALCualte[<chno>]:FORMat GDELay
DELO	MARKer[<chno>]:DELTa[:MODE] OFF
DELR1	MARKer[<chno>]:DELTa:COMPare 1
DELR2	MARKer[<chno>]:DELTa:COMPare 2
DELR3	MARKer[<chno>]:DELTa:COMPare 3
DELR4	MARKer[<chno>]:DELTa:COMPare 4
DELRFIXM	MARKer[<chno>]:DELTa[:MODE] FIXed
DISM<bool>	MARKer:LIST <bool>
DISPDATA	DISPlay[:WINDow[<chno>]]:TRACe:ASSign DATA
DISPDATM	DISPlay[:WINDow[<chno>]]:TRACe:ASSign DMEMory
DISPDDM<bool>	CALCulate[<chno>]:MATH[:EXPRession]:NAME{DDM|NONE}
DISPDM	DISPlay[:WINDow[<chno>]]:TRACe:ASSign DMEMory
DISPDMM	CALCulate:MATH[:EXPRession]:NAME DSM
DISPMEM	DISPlay[:WINDow[<chno>]]:TRACe:ASSign MEMory
DISPMEMO	DISPlay[:WINDow[<chno>]]:TRACe:ASSign MEMory
DIVI	CALCulate[<chno>]:MATH[:EXPRession]:NAME DDM
DL0	(CR + LF/EOI; none)
DL1	(LF; none)
DL2	(EOI; none)
DL3	(CR + LF; none)
DLTX<real>	MARKer[<chno>]:SEARch:RIPPle:DX <real>
DLTY<real>	MARKer[<chno>]:SEARch:RIPPle:DY <real>
DMAXMIN	MARKer[<chno>]:SEARch:RIPPle[:MODE] PPEak
DMKR10O[real]	MARKer[<chno>]:DELTa:COMPare 10[,<real>]
DMKR1O[real]	MARKer[<chno>]:DELTa:COMPare 1[,<real>]
DMKR2O[real]	MARKer[<chno>]:DELTa:COMPare 2[,<real>]
DMKR3O[real]	MARKer[<chno>]:DELTa:COMPare 3[,<real>]
DMKR4O[real]	MARKer[<chno>]:DELTa:COMPare 4[,<real>]
DMKR5O[real]	MARKer[<chno>]:DELTa:COMPare 5[,<real>]

R3762/63 Commands	Corresponding R3764/66, R3765/67 commands
DMKR6O[real]	MARKer[<chno>]:DELTa:COMPare 6[,<real>]
DMKR7O[real]	MARKer[<chno>]:DELTa:COMPare 7[,<real>]
DMKR8O[real]	MARKer[<chno>]:DELTa:COMPare 8[,<real>]
DMKR9O[real]	MARKer[<chno>]:DELTa:COMPare 9[,<real>]
DMKRA	MARKer[<chno>]:DELTa[:MODE] COMPare
DMKRC	MARKer[<chno>]:DELTa[:MODE] CHILd
DMKRF	MARKer[<chno>]:DELTa[:MODE] FIXed
DMKROF	MARKer[<chno>]:DELTa[:MODE] OFF
DONE	[SENSe:]CORRection[<chno>]:COLLect:SAVE
DONE	[SENSe:]CORRection[<chno>]:COLLect:SAVE
DONE1PORT	[SENSe:]CORRection[<chno>]:COLLect:SAVE
DONE2PORT	[SENSe:]CORRection[<chno>]:COLLect:SAVE
DONEISO	[SENSe:]CORRection[<chno>]:COLLect:SAVE
DONEREFL	[SENSe:]CORRection[<chno>]:COLLect:SAVE
DONETRNS	[SENSe:]CORRection[<chno>]:COLLect:SAVE
DRIPPL1	MARKer[<chno>]:SEARch[:MODE] RIPPle
DSSTATE<bool>	FILE:STATe:CONDition <bool>
DTOM	TRACe[<chno>]:COPY DATA
DUAC<bool>	DISPlay:DUAL <bool>
DUAL<bool>	DISPlay:DUAL <bool>
ELED<real>	[SENSe:]CORRection[<chno>]:EDELay[:TIME] <real>
ELED<val>	[SENSe:]CORRection[<chno>]:EDELay:DISTance <real>
EPORT1<real>	[SENSe:]CORRection[<chno>]:PEXTension:TIME4 <real>
EPORT2<real>	[SENSe:]CORRection[<chno>]:PEXTension:TIME5 <real>
EPORTA<real>	[SENSe:]CORRection[<chno>]:PEXTension:TIME2 <real>
EPORTB<real>	[SENSe:]CORRection[<chno>]:PEXTension:TIME3 <real>
EPORTR<real>	[SENSe:]CORRection[<chno>]:PEXTension:TIME1 <real>
ESE	*ESE
ESR?	*ESR?
EXTERN	TRIGger[:SEQuence]:SOURce EXTernal
EXTTOFF	TRIGger[:SEQuence]:SOURce IMMediate
EXTTON	TRIGger[:SEQuence]:SOURce EXTernal
FAILBEEP<bool>	DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:BEEP <bool>
FLTANA<bool>	MARKer[<chno>]:FANnalsis[:STATe] <bool>
FMKRS<real>	MARKer[<chno>]:FIXed:STIMulus <real>
FMKRV<real>	MARKer[<chno>]:FIXed:VALue <real>

<table>
<tr><th>R3762/63 Commands</th><th>Corresponding R3764/66, R3765/67 commands</th></tr>
<tr><td>FORM0</td><td>FORMat:DATA ASCii;BORDer NORMal</td></tr>
<tr><td>FORM2</td><td>FORMat:DATA REAL,32;BORDer NORMal</td></tr>
<tr><td>FORM3</td><td>FORMat:DATA REAL,64;BORDer NORMal</td></tr>
<tr><td>FORM4</td><td>FORMat:DATA ASCii;BORDer NORMal</td></tr>
<tr><td>FORM5</td><td>FORMat:DATA REAL,32;BORDer SWAPped</td></tr>
<tr><td>FORM6</td><td>FORMat:DATA REAL,64;BORDer SWAPped</td></tr>
<tr><td>FORM7</td><td>FORMat:DATA MBINary,32;BORDer NORMal</td></tr>
<tr><td>FORM8</td><td>FORMat:DATA MBINary,64;BORDer NORMal</td></tr>
<tr><td>FREE</td><td>TRIGger[:SEQuence]:SOURce IMMediate</td></tr>
<tr><td>FRER</td><td>INITiate:CONTinuous ON</td></tr>
<tr><td>FWDISO</td><td>[SENSe:]CORRection[<chno>]:COLLect[:ACQuire] FISolation</td></tr>
<tr><td>FWDMATCH</td><td>[SENSe:]CORRection[<chno>]:COLLect[:ACQuire] FMATch</td></tr>
<tr><td>FWDTRNS</td><td>[SENSe:]CORRection[<chno>]:COLLect[:ACQuire] FTRansmit>|</td></tr>
<tr><td>GRAT<bool></td><td>ISPlay[:WINDow[<chno>]]:TRACe:GRATicule[:STATe] <bool></td></tr>
<tr><td>GRPTHRU</td><td>[SENSe:]CORRection[<chno>]:COLLect[:ACQuire] GTHRU</td></tr>
<tr><td>HOLD</td><td>INITiate:CONTinuous OFF;:ABORt</td></tr>
<tr><td>HOUR<int></td><td>SYSTem:TIME <hour>,<minute>,<second></td></tr>
<tr><td>IDN?</td><td>*IDN?</td></tr>
<tr><td>IDNT</td><td>*IDN?</td></tr>
<tr><td>IFBW<int></td><td>[SENSe:]BANDwidth[:RESolution] <int></td></tr>
<tr><td>IMAG</td><td>CALCulate[<chno>]:FORMat IMAGinary</td></tr>
<tr><td>IN1CORDI</td><td>TRACe[<chno>][:DATA]{EDIRectivity|134},{<block>|<real>[,<real>...]}</td></tr>
<tr><td>IN1CORDI</td><td>TRACe[<chno>][:DATA]{EDIRrectivity|134},{<block>|<real>[,<real>...]}</td></tr>
<tr><td>IN1CORED</td><td>TRACe[<chno>][:DATA]{DATA|129},{<block>|<real>[,<real>...]}</td></tr>
<tr><td>IN1CORNR</td><td>TRACe[<chno>][:D0TA]{NORMalize|133},{<block>|<real>[,<real>...]}</td></tr>
<tr><td>IN1CORNR</td><td>TRACe[<chno>][:DATA]{NORMalize|133},{<block>|<real>[,<real>...]}</td></tr>
<tr><td>IN1CORSO</td><td>TRACe[<chno>][:DATA]{ESMatch|135},{<block>|<real>[,<real>...]}</td></tr>
<tr><td>IN1CORSO</td><td>TRACe[<chno>][:DATA]{ESMatch|135},{<block>|<real>[,<real>...]}</td></tr>
</table>

R3762/63 Commands — Corresponding R3764/66, R3765/67 commands

IN1CORTR
TRACe[<chno>][:DATA]{ERTRacking|136},{<block>|<real>[,<0eal>...]}

IN1CORTR
TRACe[<chno>][:DATA]{ERTRacking|136},{<block>|<real>[,<real>...]}

IN1DFOR
TRACe[<chno>][:DATA]{FDATa1|0},{<block>|<real>[,<real>...]}

IN1DRAT
TRACe[<chno>][:DATA]{RAW|131},{<block>|<real>[,<real>...]}

IN1MFOR
TRACe[<chno>][:DATA]{FMEMory1|2},{<block>|<real>[,<real>...]}

IN1MRAT
TRACe[<chno>][:DATA]{MEMory|130},{<block>|<real>[,<real>...]}

IN1NORED
TRACe[<chno>][:DATA]{UDATa|128},{<block>|<real>[,<real>...]}

IN2CORDI
TRACe[<chno>][:DATA]{EDIRectivity|198},{<block>|<real>[,<real>...]}

IN2CORDI
TRACe[<chno>][:DATA]{EDIRrectivity|198},{<block>|<real>[,<real>...]}

IN2CORED
TRACe[<chno>][:DATA]{DATA|193},{<block>|<real>[,<real>...]}

IN2CORNR
TRACe[<chno>][:DATA]{NORMalize|197},{<block>|<real>[,<real>...]}

IN2CORNR
TRACe[<chno>][:DATA]{NORMalize|197},{<block>|<real>[,<real>...]}

IN2CORSO
TRACe[<chno>][:DATA]{ESMatch|199},{<block>|<real>[,<real>...]}

IN2CORSO
TRACe[<chno>][:DATA]{ESMatch|199},{<block>|<real>[,<real>...]}

IN2CORTR
TRACe[<chno>][:DATA]{ERTRacking|200},{<block>|<real>[,<real>...]}

IN2CORTR
TRACe[<chno>][:DATA]{ERTRacking|200},{<block>|<real>[,<real>...]}

IN2DFOR
TRACe[<chno>][:DATA]{FDATa1|1},{<block>|<real>[,<real>...]}

IN2DRAT
TRACe[<chno>][:DATA]{RAW|195},{<block>|<real>[,<real>...]}

IN2MFOR
TRACe[<chno>][:DATA]{FMEMory1|3},{<block>|<real>[,<real>...]}

<table>
<tr><th>R3762/63 Commands</th><th>Corresponding R3764/66, R3765/67 commands</th></tr>
<tr><td>IN2MRAT</td><td>TRACe[<chno>][:DATA]{MEMory|194},{<block>|<real>[,<real>...]}</td></tr>
<tr><td>IN2NORED</td><td>TRACe[<chno>][:DATA]{UDATa|192},{<block>|<real>[,<real>...]}</td></tr>
<tr><td>IN3CORDI</td><td>TRACe[<chno>][:DATA]{EDIRectivity|262},{<block>|<real>[,<real>...]}</td></tr>
<tr><td>IN3CORED</td><td>TRACe[<chno>][:DATA]{DATA|257},{<block>|<real>[,<real>...]}</td></tr>
<tr><td>IN3CORNR</td><td>TRACe[<chno>][:DATA]{NORMalize|261},{<block>|<real>[,<real>...]}</td></tr>
<tr><td>IN3CORSO</td><td>TRACe[<chno>][:DATA]{ESMatch|263},{<block>|<real>[,<real>...]}</td></tr>
<tr><td>IN3CORTR</td><td>TRACe[<chno>][:DATA]{ERTRacking|264},{<block>|<real>[,<real>...]}</td></tr>
<tr><td>IN3DFOR</td><td>TRACe[<chno>][:DATA]{FDATa1|4},{<block>|<real>[,<real>...]}</td></tr>
<tr><td>IN3DRAT</td><td>TRACe[<chno>][:DATA]{RAW|259},{<block>|<real>[,<real>...]}</td></tr>
<tr><td>IN3MFOR</td><td>TRACe[<chno>][:DATA]{FMEMory1|6},{<block>|<real>[,<real>...]}</td></tr>
<tr><td>IN3MRAT</td><td>TRACe[<chno>][:DATA]{MEMory|258},{<block>|<real>[,<real>...]}</td></tr>
<tr><td>IN3NORED</td><td>TRACe[<chno>][:DATA]{UDATa|256},{<block>|<real>[,<real>...]}</td></tr>
<tr><td>IN4CORDI</td><td>TRACe[<chno>][:DATA]{EDIRectivity|326},{<block>|<real>[,<real>...]}</td></tr>
<tr><td>IN4CORED</td><td>TRACe[<chno>][:DATA]{DATA|321},{<block>|<real>[,<real>...]}</td></tr>
<tr><td>IN4CORNR</td><td>TRACe[<chno>][:DATA]{NORMalize|325},{<block>|<real>[,<real>...]}</td></tr>
<tr><td>IN4CORSO</td><td>TRACe[<chno>][:DATA]{ESMatch|327},{<block>|<real>[,<real>...]}</td></tr>
<tr><td>IN4CORTR</td><td>TRACe[<chno>][:DATA]{ERTRacking|328},{<block>|<real>[,<real>...]}</td></tr>
<tr><td>IN4DFOR</td><td>TRACe[<chno>][:DATA]{FDATa1|5},{<block>|<real>[,<real>...]}</td></tr>
<tr><td>IN4DRAT</td><td>TRACe[<chno>][:DATA]{RAW|323},{<block>|<real>[,<real>...]}</td></tr>
</table>

<table>
<tr><th>R3762/63 Commands</th><th>Corresponding R3764/66, R3765/67 commands</th></tr>
<tr><td>IN4MFOR</td><td>TRACe[<chno>][:DATA]{FMEMory1|7},{<block>|<real>
[,<real>...]}</td></tr>
<tr><td>IN4MRAT</td><td>TRACe[<chno>][:DATA]{MEMory|322},{<block>|<real>
[,<real>...]}</td></tr>
<tr><td>IN4NORED</td><td>TRACe[<chno>][:DATA]{UDATa|320},{<block>|<real>
[,<real>...]}</td></tr>
<tr><td>INPCOR</td><td>[SENSe:]CORRection[n]:GPHase:STATe <bool></td></tr>
<tr><td>INTERPOL</td><td>[SENSe:]CORRection[<chno>]:CSET:INTerpolate <bool></td></tr>
<tr><td>IP</td><td>SYSTem:PRESet</td></tr>
<tr><td>LABEL<str></td><td>DISPlay[:WINDow[<chno>]]:TEXT[:DATA]{<str>|<block>}</td></tr>
<tr><td>LDFILE<str></td><td>FILE:LOAD <str></td></tr>
<tr><td>LENGTH<bool></td><td>[SENSe:]CORRection[<chno>]:EDELay:STATe <bool></td></tr>
<tr><td>LENGVAL<real></td><td>[SENSe:]CORRection[<chno>]:EDELay:DISTance <real></td></tr>
<tr><td>LEVEL</td><td>[SOURce:]POWer[<chno>]:MODE SWEep</td></tr>
<tr><td>LIMC<int></td><td>DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n>
:COLor <int></td></tr>
<tr><td>DLIMIAMPO<real></td><td>DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:OFFSet
:AMPLitude <real></td></tr>
<tr><td>LIMILINE<bool></td><td>DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:LINE <bool></td></tr>
<tr><td>LIMISTIO<real></td><td>DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:OFFSet
:STIMulus <real></td></tr>
<tr><td>LIMITEST<bool></td><td>DISPlay[:WINDow[<chno>]]:LIMit[<parano>][:STATe] <bool></td></tr>
<tr><td>LIML<real></td><td>DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n>
:LOWer <real></td></tr>
<tr><td>LIMPLIN</td><td>DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:PARameter
:PLIMit LINear</td></tr>
<tr><td>LIMPLOG</td><td>DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:PARameter
:PLIMit LOGarithmic</td></tr>
<tr><td>LIMSLIN</td><td>DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:PARameter
:SLIMit LINear</td></tr>
<tr><td>LIMSLOG</td><td>DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:PARameter
:SLIMit LOGarithmic</td></tr>
<tr><td>LIMS<real></td><td>DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n>
:STIMulus <real></td></tr>
<tr><td>LIMTFL</td><td>DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n>
:TYPE FLINe</td></tr>
</table>

R3762/63 Commands	Corresponding R3764/66, R3765/67 commands
LIMTFLT	DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n> :TYPE FLINe
LIMTSL	DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n> :TYPE SLINe
LIMTSLP	DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n> :TYPE SLINe
LIMTSP	DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n> :TYPE SPOint
LIMU<real>	DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n> :UPPer <real>
LINFREQ	[SOURce:]FREQuency[<chno>]:MODE SWEep;:[SOURce:]SWEep [<chno>]:SPACing LINear
LINM	CALCulate[<chno>]:FORMat MLINear
LINMAG	CALCulate[<chno>]:FORMat MLINear
LINMP	CALCulate[<chno>]:FORMat MLIPhase
LISFREQ	[SOURce:]PSWeep[<chno>]:MODE FREQuency
LOAD	[SENSe:]CORRection[<chno>]:COLLect[:ACQuire] LOAD
LOGFREQ	[SOURce:]FREQuency[<chno>]:MODE SWEep;:[SOURce:]SWEep [<chno>]:SPACing LOGarithmic
LOGM	CALCulate[<chno>]:FORMat MLOGarithmic
LOGMAG	CALCulate[<chno>]:FORMat MLOGarithmic
LOGMD	CALCulate[<chno>]:FORMat MLODelay
LOGMP	CALCulate[<chno>]:FORMat MLOPhase
LSEG	(segment number is specified by <n> in each command)
LSEGCL	DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:CLEar
LSTIM<real>	DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n> :STIMulus <real>
M101P	[SOURce:]SWEep[<chno>]:POINts 101
M11P	[SOURce:]SWEep[<chno>]:POINts 11
M1201P	[SOURce:]SWEep[<chno>]:POINts 1201
M201P	[SOURce:]SWEep[<chno>]:POINts 201
M21P	[SOURce:]SWEep[<chno>]:POINts 21
M301P	[SOURce:]SWEep[<chno>]:POINts 301
M3P	[SOURce:]SWEep[<chno>]:POINts 3

R3762/63 Commands	Corresponding R3764/66, R3765/67 commands
M51P	[SOURce:]SWEep[<chno>]:POINts 51
M601P	[SOURce:]SWEep[<chno>]:POINts 601
M6P	[SOURce:]SWEep[<chno>]:POINts 6
MARK1<val>	MARKer[<chno>]:ACTivate[:NUMBer] 1[,<real>]
MARK2<val>	MARKer[<chno>]:ACTivate[:NUMBer] 2[,<real>]
MARK3<val>	MARKer[<chno>]:ACTivate[:NUMBer] 3[,<real>]
MARK4<val>	MARKer[<chno>]:ACTivate[:NUMBer] 4[,<real>]
MARKCONT	MARKer[<chno>]:COMPensate OFF
MARKCOUP	MARKer[<chno>]:COUPle ON
MARKCW	MARKer[<chno>]:LET CENTer
MARKDISC	MARKer[<chno>]:COMPensate ON
MARKFAUV<val>	MARKer:FIXed:AVALue <val>
MARKFSTI<val>	MARKer[<chno>]:FIXed:STIMulus <real>
MARKFVAL<val>	MARKer[<chno>]:FIXed:VALue <real>
MARKMAXI	MARKer[<chno>]:SEARch[:MODE] MAX
MARKMINI	MARKer[<chno>]:SEARch[:MODE] MIN
MARKOFF	MARKer[<chno>]:AOFF
MARKREF	MARKer[<chno>]:LET RLEVel
MARKSPAN	MARKer[<chno>]:LET SPAN
MARKSTAR	MARKer[<chno>]:LET STARt
MARKSTOP	MARKer[<chno>]:LET STOP
MARKUNCO	MARKer[<chno>]:COUPle OFF
MARKZERO	MARKer[<chno>]:LET FIXed
MAXSRCH	MARKer[<chno>]:SEARch[:MODE] MAX
MEAS	ABORt;INITiate[:IMMediate]
MEASA	[SENSe:]FUNCtion[<chno>]:POWer A
MEASB	[SENSe:]FUNCtion[<chno>]:POWer B
MEASR	[SENSe:]FUNCtion[<chno>]:POWer R
MEMARY<bool>	FILE:STATe:MEMory <bool>
MINSRCH	MARKer[<chno>]:SEARch[:MODE] MIN
MINU	CALCulate:MATH[:EXPRession]:NAME DSM
MINUTE<int>	SYSTem:TIME <hour>,<minute>,<second>
MKR10A<real>	MARKer[<chno>]:ACTivate[:NUMBer] 10[,<real>]
MKR1A<real>	MARKer[<chno>]:ACTivate[:NUMBer] 1[,<real>]
MKR2A<real>	MARKer[<chno>]:ACTivate[:NUMBer] 2[,<real>]
MKR3A<real>	MARKer[<chno>]:ACTivate[:NUMBer] 3[,<real>]
MKR4A<real>	MARKer[<chno>]:ACTivate[:NUMBer] 4[,<real>]
MKR5A<real>	MARKer[<chno>]:ACTivate[:NUMBer] 5[,<real>]

<table>
<tr><th>R3762/63 Commands</th><th>Corresponding R3764/66, R3765/67 commands</th></tr>
<tr><td>MKR6A<real></td><td>MARKer[<chno>]:ACTivate[:NUMBer] 6[,<real>]</td></tr>
<tr><td>MKR7A<real></td><td>MARKer[<chno>]:ACTivate[:NUMBer] 7[,<real>]</td></tr>
<tr><td>MKR8A<real></td><td>MARKer[<chno>]:ACTivate[:NUMBer] 8[,<real>]</td></tr>
<tr><td>MKR9A<real></td><td>MARKer[<chno>]:ACTivate[:NUMBer] 9[,<real>]</td></tr>
<tr><td>MKRAOFF</td><td>MARKer[<chno>]:AOFF</td></tr>
<tr><td>MKRCENT</td><td>MARKer[<chno>]:LET CENTer</td></tr>
<tr><td>MKRCMP</td><td>MARKer[<chno>]:COMPensate ON</td></tr>
<tr><td>MKRCOUP</td><td>MARKer[<chno>]:COUPle ON</td></tr>
<tr><td>MKRFIX</td><td>MARKer[<chno>]:LET FIXed</td></tr>
<tr><td>MKROFF</td><td>MARKer[<chno>]:ACTivate:STATe OFF</td></tr>
<tr><td>MKRPART<bool></td><td>MARKer[<chno>]:SEARch:PARTial[:STATe] <bool></td></tr>
<tr><td>MKRREF</td><td>MARKer[<chno>]:LET RLEVel</td></tr>
<tr><td>MKRSPAN</td><td>MARKer[<chno>]:LET SPAN</td></tr>
<tr><td>MKRSTAR</td><td>MARKer[<chno>]:LET STARt</td></tr>
<tr><td>MKRSTOP</td><td>MARKer[<chno>]:LET STOP</td></tr>
<tr><td>MKRTRAC<bool></td><td>MARKer[<chno>]:SEARch:TRACking <bool></td></tr>
<tr><td>MKRUCMP</td><td>MARKer[<chno>]:COMPensate OFF</td></tr>
<tr><td>MKRUCOUP</td><td>MARKer[<chno>]:COUPle OFF</td></tr>
<tr><td>MKRZO50</td><td>CALCulate[<chno>]:TRANsform:IMPedance:CIMPedance 50OHM</td></tr>
<tr><td>MKRZO75</td><td>CALCulate[<chno>]:TRANsform:IMPedance:CIMPedance 75OHM</td></tr>
<tr><td>MONTH<int></td><td>SYSTem:DATE <year>,<month>,<day></td></tr>
<tr><td></td><td></td></tr>
<tr><td>NORM<ON></td><td>[SENSe:]CORRection[<chno>]:COLLect[:ACQuire] NORMalize</td></tr>
<tr><td>NORMS<ON></td><td>[SENSe:]CORRection[<chno>]:COLLect[:ACQuire] SNORmalize</td></tr>
<tr><td></td><td></td></tr>
<tr><td>OMITISO</td><td>[SENSe:]CORRection[<chno>]:COLLect[:ACQuire] OISolation</td></tr>
<tr><td>OPC</td><td>*OPC</td></tr>
<tr><td>OPEN</td><td>[SENSe:]CORRection[<chno>]:COLLect[:ACQire] OPEN</td></tr>
<tr><td>OT1CORDI</td><td>TRACe[<chno>][:DATA]?{EDIRectivity|134}</td></tr>
<tr><td>OT1CORED</td><td>TRACe[<chno>][:DATA]?{DATA|129}</td></tr>
<tr><td>OT1CORNR</td><td>TRACe[<chno>][:DATA]?{NORMalize|133}</td></tr>
<tr><td>OT1CORSO</td><td>TRACe[<chno>][:DATA]?{ESMatch|135}</td></tr>
<tr><td>OT1CORTR</td><td>TRACe[<chno>][:DATA]?{ERTRacking|136}</td></tr>
<tr><td>OT1DFOR</td><td>TRACe[<chno>][:DATA]?{FDATa1|0}</td></tr>
<tr><td>OT1DRAT</td><td>TRACe[<chno>][:DATA]?{RAW|131}</td></tr>
<tr><td>OT1MFOR</td><td>TRACe[<chno>][:DATA]?{FMEMory1|2}</td></tr>
<tr><td>OT1MRAT</td><td>TRACe[<chno>][:DATA]?{MEMory|130}</td></tr>
<tr><td>OT1NORED</td><td>TRACe[<chno>][:DATA]?{UDATa|128}</td></tr>
</table>

R3762/63 Commands	Corresponding R3764/66, R3765/67 commands
OT2CORDI	TRACe[<chno>][:DATA]?{EDIRectivity|198}
OT2CORED	TRACe[<chno>][:DATA]?{DATA|193}
OT2CORNR	TRACe[<chno>][:DATA]?{NORMalize|197}
OT2CORSO	TRACe[<chno>][:DATA]?{ESMatch|199}
OT2CORTR	TRACe[<chno>][:DATA]?{ERTRacking|200}
OT2DFOR	TRACe[<chno>][:DATA]?{FDATa1|1}
OT2DRAT	TRACe[<chno>][:DATA]?{RAW|195}
OT2MFOR	TRACe[<chno>][:DATA]?{FMEMory1|3}
OT2MRAT	TRACe[<chno>][:DATA]?{MEMory|194}
OT2NORED	TRACe[<chno>][:DATA]?{UDATa|192}
OT3CORDI	TRACe[<chno>][:DATA]?{EDIRectivity|262}
OT3CORED	TRACe[<chno>][:DATA]?{DATA|257}
OT3CORNR	TRACe[<chno>][:DATA]?{NORMalize|261}
OT3CORSO	TRACe[<chno>][:DATA]?{ESMatch|263}
OT3CORTR	TRACe[<chno>][:DATA]?{ERTRacking|264}
OT3DFOR	TRACe[<chno>][:DATA]?{FDATa1|4}
OT3DRAT	TRACe[<chno>][:DATA]?{RAW|259}
OT3MFOR	TRACe[<chno>][:DATA]?{FMEMory1|6}
OT3MRAT	TRACe[<chno>][:DATA]?{MEMory|258}
OT3NORED	TRACe[<chno>][:DATA]?{UDATa|256}
OT4CORDI	TRACe[<chno>][:DATA]?{EDIRectivity|326}
OT4CORED	TRACe[<chno>][:DATA]?{DATA|321}
OT4CORNR	TRACe[<chno>][:DATA]?{NORMalize|325}
OT4CORSO	TRACe[<chno>][:DATA]?{ESMatch|327}
OT4CORTR	TRACe[<chno>][:DATA]?{ERTRacking|328}
OT4DFOR	TRACe[<chno>][:DATA]?{FDATa1|5}
OT4DRAT	TRACe[<chno>][:DATA]?{RAW|323}
OT4MFOR	TRACe[<chno>][:DATA]?{FMEMory1|7}
OT4MRAT	TRACe[<chno>][:DATA]? {MEMory|322}
OT4NORED	TRACe[<chno>][:DATA]? {UDATa|320}
OUTLEV<real>	[SOURce:]POWer[<chno>][:LEVel][:AMPLitude] <real>
PCB<int>	*PCB <int>
PHAO<real>	[SENSe:]CORRection[<chno>]:OFFSet:PHASe <real>
PHAOFS<bool>	[SENSe:]CORRection[<chno>]:OFFSet:STATe <bool>
PHAS	CALCulate[<chno>]:FORMat PHASe
PHASE	CALCulate[<chno>]:FORMat PHASe
PMKRLIN	MARKER[<chno>]:POLar MLINear

R3762/63 Commands	Corresponding R3764/66, R3765/67 commands
PMKRRI	MARKER[<chno>]:POLar RIMaginary
PMKRRLOG	MARKER[<chno>]:POLar MLOGarithmic
POIN<int>	[SOURce:]SWEep[<chno>]:POINts <int>
POLA	CALCulate[<chno>]:FORMat POLar
POLAR	CALCulate[<chno>]:FORMat POLar
POLMLIN	MARKER[<chno>]:POLar MLINear
POLMLOG	MARKER[<chno>]:POLar MLOGarithmic
POLMRI	MARKER[<chno>]:POLar RIMaginary
PORE<bool>	[SENSe:]CORRection[<chno>]:PEXTension:STATe <bool>
PORT1FEM	[SENSe:]CORRection[<chno>]:CKIT:TERMinal1 FEMale
PORT1MAL	[SENSe:]CORRection[<chno>]:CKIT:TERMinal1 MALe
PORT2FEM	[SENSe:]CORRection[<chno>]:CKIT:TERMinal2 FEMale
PORT2MAL	[SENSe:]CORRection[<chno>]:CKIT:TERMinal3 MALe
PORTA<real>	[SENSe:]CORRection[<chno>]:PEXTension:TIME2 <real>
PORTB<real>	[SENSe:]CORRection[<chno>]:PEXTension:TIME3 <real>
POWE<real>	[SOURce:]POWer[<chno>][:LEVel][:AMPLitude] <real>
POWS	[SOURce:]POWer[<chno>]:MODE SWEep
POWTOFF	[SENSe:]POWer:AC:PROTection:CLEar
PRES	SYSTem:PRESet
PURGE<str>	FILE:DELete <str>
RAWARY<bool>	FILE:STATe:RAW <bool>
RBW100HZ	[SENSe:]BANDwidth[:RESolution] 100HZ
RBW10HZ	[SENSe:]BANDwidth[:RESolution] 10HZ
RBW1KHZ	[SENSe:]BANDwidth[:RESolution] 1KHZ
RBW300HZ	[SENSe:]BANDwidth[:RESolution] 300HZ
RBW30HZ	[SENSe:]BANDwidth[:RESolution] 30HZ
RBW<int>	[SENSe:]BANDwidth[:RESolution] <int>
RBWAUTO	[SENSe:]BANDwidth[:RESolution]:AUTO ON
REAL	CALCulate[<chno>]:FORMat REAL
RECA1	REGister:RECall 1
RECA2	REGister:RECall 2
RECA3	REGister:RECall 3
RECA4	REGister:RECall 4
RECA5	REGister:RECall 5
RECLPOFF	REGister:RECall{0|POFF}

R3762/63 Commands	Corresponding R3764/66, R3765/67 commands
RECLREG1	REGister:RECall 1
RECLREG10	REGister:RECall 10
RECLREG2	REGister:RECall 2
RECLREG3	REGister:RECall 3
RECLREG4	REGister:RECall 4
RECLREG5	REGister:RECall 5
RECLREG6	REGister:RECall 6
RECLREG7	REGister:RECall 7
RECLREG8	REGister:RECall 8
RECLREG9	REGister:RECall 9
REFL<bool>	DISPlay[:WINDow[<chno>]]:Y[trace]:RLINe <bool>
REFP<real>	DISPlay[:WINDow[<chno>]]:Y[trace][:SCALe]:RPOSition <real>
REFV<real>	DISPlay[:WINDow[<chno>]]:Y[trace][:SCALe]:RLEVel <real>
REST	ABORt;INITiate[:IMMediate]
REVISO	[SENSe:]CORRection[<chno>]:COLLect[:ACQuire] RISolation
REVMATCH	[SENSe:]CORRection[<chno>]:COLLect[:ACQuire] RMATch
REVTRNS	[SENSe:]CORRection[<chno>]:COLLect[:ACQuire] RTRansmit
RIN	[SENSe:]FUNCtion[<chno>]:POWer R
RST	*RST
RTC30ADJ	SYSTem:TIME <hour>,<minute>,<second>
S11	[SENSe:]FUNCtion[<chno>]:POWer S11
S11LOAD	[SENSe:]CORRection[<chno>]:COLLect[:ACQuire] S11Load
S11OPEN	[SENSe:]CORRection[<chno>]:COLLect[:ACQuire] S11Open
S11SHORT	[SENSe:]CORRection[<chno>]:COLLect[:ACQuire] S11Short
S12	[SENSe:]FUNCtion[<chno>]:POWer S12
S21	[SENSe:]FUNCtion[<chno>]:POWer S21
S22	[SENSe:]FUNCtion[<chno>]:POWer S22
S22LOAD	[SENSe:]CORRection[<chno>]:COLLect[:ACQuire] S22Load
S22OPEN	[SENSe:]CORRection[<chno>]:COLLect[:ACQuire] S22Oopen
S22SHORT	[SENSe:]CORRection[<chno>]:COLLect[:ACQuire] S22Short
SAVE1	REGister:SAVE 1
SAVE2	REGister:SAVE 2
SAVE3	REGister:SAVE 3
SAVE4	REGister:SAVE 4
SAVE5	REGister:SAVE 5
SAVEREG1	REGister:SAVE 1
SAVEREG10	REGister:SAVE 10

R3762/63 Commands	Corresponding R3764/66, R3765/67 commands
SAVEREG2	REGister:SAVE 2
SAVEREG3	REGister:SAVE 3
SAVEREG4	REGister:SAVE 4
SAVEREG5	REGister:SAVE 5
SAVEREG6	REGister:SAVE 6
SAVEREG7	REGister:SAVE 7
SAVEREG8	REGister:SAVE 8
SAVEREG9	REGister:SAVE 9
SCAL<real>	DISPlay[:WINDow[<chno>]]:Y[trace][:SCALe]:PDIVision <real>
SCALF1ST	DISPlay[:WINDow[<chno>]]:Y[trace]...
SCALF2ND	DISPlay[:WINDow[<chno>]]:Y[trace]...
SDIV<real>	DISPlay[:WINDow[<chno>]]:Y[trace][:SCALe]:PDIVision <real>
SEAMAX	MARKer[<chno>]:SEARch[:MODE] MAX
SEAMIN	MARKer[<chno>]:SEARch[:MODE] MIN
SEAOFF	MARKer[<chno>]:SEARch[:MODE] OFF
SETLTIME<real>	TRIGger[:SEQuence]:DELay <real>
SETLVARI<bool>	TRIGger[:SEQuence]:DELay:STATe <bool>
SETZ	CALCulate[<chno>]:TRANsform:IMPedance:CIMPedance <real>
SETZ0<real>	CALCulate[<chno>]:TRANsform:IMPedance:CIMPedance <real>
SFWD	[SENSe:]FUNCtion[<chno>]:POWer SFWD
SGJB	CALCulate[<chno>]:FORMat ISCHart
SHORT	[SENSe:]CORRection[<chno>]:COLLect[:ACQuire] SHORt
SING	INITiate:CONTinuous OFF;:ABORt;INITiate
SINGLE	INITiate:CONTinuous OFF;:ABORt;INITiate
SMEAS<bool>	[SENSe:]FUNCtion[<chno>]:POWer <input>
SMIC	CALCulate[<chno>]:FORMat SCHart
SMIMGB	MARKer[<chno>]:SMITh ADMittance
SMIMLIN	MARKer[<chno>]:SMITh MLINear
SMIMLOG	MARKer[<chno>]:SMITh MLOGarithmic
SMIMRI	MARKer[<chno>]:SMITh RIMaginary
SMIMRX	MARKer[<chno>]:SMITh IMPedance
SMKRGB	MARKer[<chno>]:SMITh ADMittance
SMKRLIN	MARKer[<chno>]:SMITh MLINear
SMKRLOG	MARKer[<chno>]:SMITh MLOGarithmic
SMKRRI	MARKer[<chno>]:SMITh RIMaginary
SMKRRX	MARKer[<chno>]:SMITh IMPedance
SMOO<bool>	CALCulate[<chno>]:SMOothing:STATe <bool>
SMOOAPER<real>	CALCulate[<chno>]:SMOothing:APERture <real>
SPAN<real>	[SOURce:]FREQuency[<chno>]:SPAN <real>

R3762/63 Commands	Corresponding R3764/66, R3765/67 commands
SPANF<real>	[SOURce:]FREQuency[<chno>]:SPAN <real>
SPLD<bool>	DISPlay:FORMat {ULOWer|FBACk}
SPLEVEL<real>	[SOURce:]POWer[<chno>]:STOP <real>
SPLIT<bool>	DISPlay:FORMat {ULOWer|FBACk}
SRCCOR	[SOURce:]CORRection[n]:GAIN:STATe <bool>
SRCHOFF	MARKer[<chno>]:SEARch[:MODE] OFF
SRE	*SRE
SREV	[SENSe:]FUNCtion[<chno>]:POWer SREV
SRJX	CALCulate[<chno>]:FORMat SCHart
SRQD	(none)
SRQE	(none)
STAR<real>	[SOURce:]{FREQuency|POWer}[<chno>]:STARt <real>
STARTF<real>	[SOURce:]FREQuency[<chno>]:STARt <real>
STB?	*STB?
STFILE<str>	FILE:STORe <str>
STIME<real>	[SOURce:]SWEep[<chno>]:TIME <real>
STIMEAUTO	[SOURce:]SWEep[<chno>]:TIME:AUTO ON
STLEVEL<real>	[SOURce:]POWer[<chno>]:STARt <real>
STOP<real>	[SOURce:]{FREQuency|POWer}[<chno>]:STOP <real>
STOPF<real>	[SOURce:]FREQuency[<chno>]:STOP <real>
SWEA	[SOURce:]SWEep[<chno>]:TIME:AUTO ON
SWET<real>	[SOURce:]SWEep[<chno>]:TIME <real>
SWPHLD	INITiate:CONTinuous OFF;:ABORt
SWR	CALCulate[<chno>]:FORMat SWR
T3DB	MARKer[<chno>]:FANalysis:WIDTh 3DB
T3DEG	MARKer[<chno>]:FANalysis:WIDTh 3DEG
T60DB	MARKer[<chno>]:FANalysis:WIDTh 60DB
T6DB	MARKer[<chno>]:FANalysis:WIDTh 6DB
T6DEG	MARKer[<chno>]:FANalysis:WIDTh 6DEG
TIN	MARKer[<chno>]:FANalysis:DIRection IN
TITL<str>	DISPlay[:WINDow[<chno>]]:TEXT[:DATA] <str>
TOUT	MARKer[<chno>]:FANalysis:DIRection OUT
TRACK<bool>	MARKer[<chno>]:SEARch:TRACking <bool>
TST?	*TST?
TXDB<real>	MARKer[<chno>]:FANalysis:WIDTh <real>
TXDEG<real>	MARKer[<chno>]:FANalysis:WIDTh <real>;:MARKer[<chno>] :SEARch[:MODE] TARGet

R3762/63 Commands	Corresponding R3764/66, R3765/67 commands
UFREQ<real>	[SOURce:]PSWeep[<chno>]:FREQuency[n] <real>
ULEVEL<real>	[SOURce:]PSWeep[<chno>]:POWer[n] <real>
UNWARP	CALCulate[<chno>]:FORMat UPHase
UPOINT<int>	[SOURce:]PSWeep[<chno>]:POINts[n] <int>
URBW<int>	[SOURce:]PSWeep[<chno>]:BANDwidth[n] <int>
USEG<int>	[SOURce:]PSWeep[<chno>]:FREQuency[n] <real>[,<real>]
USEGCL	[SOURce:]PSWeep[<chno>]:CLEar[n]:ALL
USETLT<real>	[SOURce:]PSWeep[<chno>]:SETTling[n] <real>
USPLEV	[SOURce:]PSWeep[<chno>]:POWer[n] <real>[,<real>]
USRASWP	[SOURce:]PSWeep[<chno>]:MODE ALL
USRFSWP	[SOURce:]PSWeep[<chno>]:MODE FREQuency
USRSWP	[SOURce:]PSWeep[<chno>]:MODE FREQuency
USTART<real>	[SOURce:]PSWeep[<chno>]:FREQuency[n] <real>[,<real>]
USTLEV	[SOURce:]PSWeep[<chno>]:POWer[n] <real>[,<real>]
USTOP<real>	[SOURce:]PSWeep[<chno>]:FREQuency[n] <real>[,<real>]
VELOFACT<real>	[SENSe:]CORRection[<chno>]:RVELocity:COAX <real>
WAIT	*WAI
WIDT<bool>	MARKer[<chno>]:FANnalsis[:STATe] <bool>
WIDV<real>	MARKer[<chno>]:FANalysis:WIDTh <real>
YEAR<int>	SYSTem:DATE <year>,<month>,<day>
ZRPSRCH	MARKer[<chno>]:SEARch:TARGet[:MODE] ZERO
ZYMKDFLT	MARKer[<chno>]:CONVert[:MODE] DEFault
ZYMKLIN	MARKer[<chno>]:CONVert[:MODE] LINear
ZYMKRI	MARKer[<chno>]:CONVert[:MODE] RIMaginary

MEMO

A.2 GPIB Command List Corresponding to Panel Key / Softkey

Shows the GPIB command corresponding to the panel key or the softkey.

- Describes depending on the item in the following panel.

    1. ACTIVE CHANNEL block
    2. STIMULUS block
    3. RESPONSE block
    4. INSTRUMENT STATE block
    5. GPIB block

- Explanation of "O"and "N"

    O: IEEE488.1-1987 command mode

    N: IEEE488.2-1987 command mode

A2.1 ACTIVE CHANNEL Block

(1) CH1

CH1

O: `CH1`
N: `DISPlay:ACTive {1|3}`

(2) CH2

CH2

O: `CH2`
N: `DISPlay:ACTive {2|4}`

A2.2 STIMULUS Block

(1) MENU
Signal source menu

POWER
Calls the power menu. (See step (1-1).)

SWEEP TIME
O: `STIME <real>`
`STIMEAUTO`
N: `[SOURce:]SWEep[<chno>]:TIME <real>`
`[SOURce:]SWEep[<chno>]:TIME:AUTO <bool>`

SWEEP TYPE [ ]
Calls the sweep type menu. (See step (1-3).)

TRIGGER [ ]
Calls the trigger menu. (See step (1-2).)

POINTS
O: `M{1201|601|301|201|101|51|21|11|6|3}P/ POIN <int>`
`POIN <int>`
N: `[SOURce:]SWEep[<chno>]:POINts <int>`

COUPLED CH ON/OFF
O: `COUPLE <bool>`
N: `[SOURce:]COUPle <bool>`

CW FREQ
O: `CWFREQ <real>`
N: `[SOURce:]FREQuency[<chno>]:CW <real>`

RESTART
O: `MEAS`
N: `ABORt;INITiate[:IMMediate]`

(1-1) Power menu

POWER	O: `OUTLEV <real>` N: `[SOURce:]POWer[<chno>][:LEVel][:AMPLitude] <real>`
Return	Returns to the signal source menu. (See step (1).)

(1-2) Trigger menu

CONTINUOUS	O: `CONT` N: `INITiate:CONTinuous ON`
SINGLE	O: `SINGLE` N: `INITiate:CONTinuous OFF;:ABORt;INITiate`
HOLD	O: `SWPHLD` N: `INITiate:CONTinuous OFF;:ABORt`
INT TRIG	O: `FREE` N: `TRIGger[:SEQuence]:SOURce IMMediate`
EXT TRIG	O: `EXTERN` N: `TRIGger[:SEQuence]:SOURce EXTernal`
TRIGGER DELAY	O: `SETLTIME <real>` N: `TRIGger[:SEQuence]:DELay <real>`
Return	Returns to the signal source menu. (See step (1).)

(1-3) Sweep type menu

LIN FREQ	O: LINFREQ N: [SOURce:]FREQuency[<chno>]:MODE SWEep [SOURce:]SWEep[<chno>]:SPACing LINear } Use these commands together.
LOG FREQ	O: LOGFREQ N: [SOURce:]FREQuency[<chno>]:MODE SWEep [SOURce:]SWEep[<chno>]:SPACing LOGarithmic } Use these commands together.
USER SWEEP	O: USRFSWP N: [SOURce:]PSWeep[<chno>]:MODE FREQuency
PROGRAM SWEEP	O: USRARWP N: [SOURce:]PSWeep[<chno>]:MODE ALL
POW SWEEP	O: LEVEL N: [SOURce:]POWer[<chno>]:MODE SWEep
EDIT USER SWEEP	Calls the user frequency sweep segment editing menu. (See step (1-3-1).)
EDIT PROG SWEEP	Calls the program sweep segment editing menu. (See step (1-3-2).)
Return	Returns to the signal source menu. (See step (1).)

(1-3-1) User frequency sweep segment editing menu

SEGMENT: NUMBER	O: USEG <n> N: See Note.
START	O: USTART<start> N: [SOURce:]PSWeep[<chno>]:FREQuency[<n>] <start>[,<stop>]
STOP	O: USTOP<stop> N: [SOURce:]PSWeep[<chno>]:FREQuency[<n>] <start>[,<stop>]
FREQ	O: UFREQ<real> N: [SOURce:]PSWeep[<chno>]:FREQuency[<n>] <start>
POINT	O: UPOINT <int> N: [SOURce:]PSWeep[<chno>]:POINts[<n>] <int>
CLEAR SEG	O: There is no GPIB command to be applied. N: [SOURce:]PSWeep[<chno>]:CLEar[<n>]
CLEAR ALL SEG	O: USEGCL N: [SOURce:]PSWeep[<chno>]:CLEar[<n>]:ALL
Return	

<start> and <stop> are <real>.

Note: In IEEE488.2-1987 command mode, the segment number is specified by the parameter <n> in each GPIB command.

(1-3-2) Program sweep segment editing menu (1 of 2)

SEGMENT: NUMBER	O: USEG <n> N: See Note 1.
START	O: USTART<start> / UFREQ<real> N: [SOURce:]PSWeep[<chno>]:FREQuency[<n>] <start>[,<stop>]
STOP	O: USTOP<stop> N: [SOURce:]PSWeep[<chno>]:FREQuency[<n>] <start>[,<stop>]
POINT	O: UPOINT <int> N: [SOURce:]PSWeep[<chno>]:POINts[<n>] <int>
CLEAR SEG	O: There is no GPIB command to be applied. N: [SOURce:]PSWeep[<chno>]:CLEar[<n>]
CLEAR ALL SEG	O: USEGCL N: [SOURce:]PSWeep[<chno>]:CLEar[<n>]:ALL
Return	Returns to the sweep type menu. (See step (1-3).)
More 1/2	Calls the program sweep segment editing menu (2 of 2).

<start> and <stop> are real.

Note: In IEEE488.2-1987 command mode, the segment number is specified by the parameter <n> in each GPIB command.

Program sweep segment editing menu (2 of 2)

SEGMENT: POWER	O: ULEVEL <real> N: [SOURce:]PSWeep[<chno>]:POWer[<n>] <real>
IF RBW	O: URBW <int> N: [SOURce:]PSWeep[<chno>]:BANDwidth[<n>] <int>
SETTLING TIME	O: USETLT <real> N: [SOURce:]PSWeep[<chno>]:SETTling[<n>] <real>
Return	Returns to the sweep type menu. (See step (1-3).)
More 2/2	Calls the program sweep segment editing menu (1 of 2).

(2) START

START

O: STARTF <real>
STLEVEL <real>

N: [SOURce:]FREQuency[<chno>]:STARt <real>
[SOURce:]POWer[<chno>]:STARt <real>

(3) STOP

STOP

O: STOPF <real>
STLEVEL <real>

N: [SOURce:]FREQuency[<chno>]:STOP <real>
[SOURce:]POWer[<chno>]:STOP <real>

(4) CENTER

CENTER

O: CENTERF <real>

N: [SOURce:]FREQuency[<chno>]:CENTer <real>

(5) SPAN

SPAN

O: SPANF <real>

N: [SOURce:]FREQuency[<chno>]:SPAN <real>

A2.3 RESPONSE Block

(1) MEAS
Measurement menu

① R3765A/67A + S parameter, R3765C/67C

S11(A/R) REFL FWD	O: `S11` N: `[SENSe:]FUNCtion[<chno>][:ON] 'POWer:S11'` `[SENSe:]FUNCtion[<chno>]:POWer S11`
S21(B/R) TRANS FWD	O: `S21` N: `[SENSe:]FUNCtion[<chno>][:ON] 'POWer:S21'` `[SENSe:]FUNCtion[<chno>]:POWer S21`
S12(A/R) TRANS REV	O: `S12` N: `[SENSe:]FUNCtion[<chno>][:ON] 'POWer:S12'` `[SENSe:]FUNCtion[<chno>]:POWer S12`
S22(B/R) REFL REV	O: `S22` N: `[SENSe:]FUNCtion[<chno>][:ON] 'POWer:S22'` `[SENSe:]FUNCtion[<chno>]:POWer S22`
S11&S21 FWD	O: There is no GPIB command to be applied. N: `[SENSe:]FUNCtion[<chno>][:ON] 'POWer:SFWD'` `[SENSe:]FUNCtion[<chno>]:POWer SFWD`
S22&S12 REV	O: There is no GPIB command to be applied. N: `[SENSe:]FUNCtion[<chno>][:ON] 'POWer:SREV'` `[SENSe:]FUNCtion[<chno>]:POWer SREV`
SUB MEAS ON/OFF	Calls the parameter conversion menu. (See step (1-1).)
CONVERSION [ ]	Calls the parameter conversion menu. (See step (1-1).)

② 3R3765A/67A

A/R	O: ARIN N: [SENSe:]FUNCtion[<chno>][:ON] 'POWer:AC:RATio2,1' [SENSe:]FUNCtion[<chno>]:POWer AR
B/R	O: BRIN N: [SENSe:]FUNCtion[<chno>][:ON] 'POWer:AC:RATio 3,1' [SENSe:]FUNCtion[<chno>]:POWer BR
R	O: RIN N: [SENSe:]FUNCtion[<chno>][:ON] 'POWer:AC1' [SENSe:]FUNCtion[<chno>]:POWer R
A	O: AIN N: [SENSe:]FUNCtion[<chno>][:ON] 'POWer:AC2' [SENSe:]FUNCtion[<chno>]:POWer A
B	O: BIN N: [SENSe:]FUNCtion[<chno>][:ON] 'POWer:AC3' [SENSe:]FUNCtion[<chno>]:POWer B
SUB MEAS ON/OFF	
CONVERSION [ ]	Calls the parameter conversion menu. (See step (1-1).)

③ R3765B/67B

Softkey	Command
REFLECTION	O: `S11` N: `[SENSe:]FUNCtion[<chno>][:ON] 'POWer:S11'` `[SENSe:]FUNCtion[<chno>]:POWer S11`
TRANS MISSION	O: `S21` N: `[SENSe:]FUNCtion[<chno>][:ON] 'POWer:S21'` `[SENSe:]FUNCtion[<chno>]:POWer S21`
TRANS & REFL	O: There is no GPIB command to be applied. N: `[SENSe:]FUNCtion[<chno>][:ON] 'POWer:SFWD'` `[SENSe:]FUNCtion[<chno>]:POWer SFWD`
SUB MEAS ON/OFF	
CONVERSION [ ]	Calls the parameter conversion menu. (See step (1-1).)

(1-1) Parameter conversion menu

Softkey	Command
Z(REFL)	O: `CONVRZ` N: `CALCulate[<chno>]:TRANsform:IMPedance:TYPE ZREFlection`
Z(TRANS)	O: `CONVTZ` N: `CALCulate[<chno>]:TRANsform:IMPedance:TYPE ZTRansmit`
Y(REFL)	O: `CONVRY` N: `CALCulate[<chno>]:TRANsform:IMPedance:TYPE YREFlection`
Y(TRANS)	O: `CONVTY` N: `CALCulate[<chno>]:TRANsform:IMPedance:TYPE YTRansmit`
1/S	O: `CONV1DS` N: `CALCulate[<chno>]:TRANsform:IMPedance:TYPE INVersion`
OFF	O: `CONVOFF` N: `CALCulate[<chno>]:TRANsform:IMPedance:TYPE NONE`
Z0 VALUE	O: `SETZ0 <real> / MKRZ0{50l75}` N: `CALCulate[<chno>]:TRANsform:IMPedance:CIMPedance <real>`
Return	Returns to the measurement menu. (See step (1).)

(2) FORMAT

Format menu (1 of 2)

LOG MAG	O: `LOGMAG` N: `CALCulate[<chno>]:FORMat MLOGarithmic`
PHASE	O: `PHASE` N: `CALCulate[<chno>]:FORMat PHASe`
DELAY	O: `DELAY` N: `CALCulate[<chno>]:FORMat GDELay`
SMITH (R+jX)	O: `SRJX` N: `CALCulate[<chno>]:FORMat SCHart`
SMITH (G+jB)	O: `SGJB` N: `CALCulate[<chno>]:FORMat ISCHart`
POLAR	O: `POLAR` N: `CALCulate[<chno>]:FORMat POLar`
LIN MAG	O: `LINMAG` N: `CALCulate[<chno>]:FORMat MLINear`
More 1/2	Calls the format menu (2 of 2).

Format menu (2 of 2)

SWR	O: `SWR` N: `CALCulate[<chno>]:FORMat SWR`
REAL	O: `REAL` N: `CALCulate[<chno>]:FORMat REAL`
IMAG	O: `IMAG` N: `CALCulate[<chno>]:FORMat IMAGinary`
PHASE -∞, +∞	O: `UNWRAP` N: `CALCulate[<chno>]:FORMat UPHase`
LOG MAG & PHASE	O: `LOGMP` N: `CALCulate[<chno>]:FORMat MLOPhase`
LOG MAG & DELAY	O: `LOGMD` N: `CALCulate[<chno>]:FORMat MLODelay`
LIN MAG & PHASE	O: `LINMP` N: `CALCulate[<chno>]:FORMat MLIPhase`
More 2/2	Calls the format menu (1 of 2).

(3) SCALE

Scale menu

AUTO SCALE
O: AUTO
N: DISPlay[:WINDow[<chno>]]:Y[<trace>][:SCALe]:AUTO ONCE

/DIV
O: SDIV <real>
N: DISPlay[:WINDow[<chno>]]:Y[<trace>][:SCALe]:PDIVision <real>

REF VALUE
O: REFV <real>
N: DISPlay[:WINDow[<chno>]]:Y[<trace>][:SCALe]:RLEVel <real>

REF POS
O: REFP <real>
N: DISPlay[:WINDow[<chno>]]:Y[<trace>][:SCALe]:RPOSition <real>

REF LINE
O: REFL <bool>
N: DISPlay[:WINDow[<chno>]]:Y[<trace>]RLINe <bool>

SCALE FOR 2nd / 1st
O: SCALF{1ST|2ND}
N: See Note.

Note: In IEEE488.2-1987 command mode, TRACE is selected by the parameter <trace> in each GPIB command.

<trace> = 0,1,4,5,8,9,12,13
(0:CH1 TRACE 1st,
1:CH2 TRACE 1st,
4:CH3 TRACE 1st,
5:CH4 TRACE 1st,
8:CH1 TRACE 2nd,
9:CH2 TRACE 2nd,
12:CH3 TRACE 2nd,
13:CH4 TRACE 2nd)

(4) DISPLAY

Display menu (1 of 2)

DUAL CH ON/OFF	O: DUAL <bool> N: DISPlay:DUAL <bool>
SPLIT CH ON/OFF	O: SPLIT <bool> N: DISPlay:FORMat {ULOWer|FBACk} (See Note.)
DISPLAY DATA	O: DISPDATA N: DISPlay[:WINDow[<chno>]]:TRACe:ASSign DATA
DISPLAY MEMORY	O: DISPMEM N: DISPlay[:WINDow[<chno>]]:TRACe:ASSign MEMory
DISPLAY DATA & MEM	O: DISPDM N: DISPlay[:WINDow[<chno>]]:TRACe:ASSign DMEMory
DEFINE TRACE [ ]	Calls the trace operation menu. (See step (4-2).)
DATA→ MEMORY	O: DTOM N: TRACe[<chno>]:COPY DATA
More 1/2	Calls the display menu (2 of 2).

Note: SPLIT CH:
ULOWer; Split display
FBACk; Over-wrap display

Display menu (2 of 2)

GRATICULE ON/OFF	O: GRAT <bool> N: DISPlay[:WINDow[<chno>]]:TRACe:GRATicule[:STATe] <bool>
LABEL	Calls the label menu. (See step (4-1).)
More 2/2	Calls the display menu (1 of 2).

(4-1) Label menu

DONE	O: `LABEL <str>` N: `DISPlay[:WINDow[<chno>]]:TEXT[:DATA] {<str>|<block>}`
CURSOR →	There is no GPIB command to be applied.
CURSOR ←	There is no GPIB command to be applied.
BACKSPACE	There is no GPIB command to be applied.
DELETE CHAR	There is no GPIB command to be applied.
CLEAR LINE	There is no GPIB command to be applied.
CANCEL	Calls the display menu (2 of 2). (See step (4).)

(4-2) Trace operation menu

DATA/MEM	O: `DISPDDM ON` N: `CALCulate[<chno>]:MATH[:EXPRession]:NAME DDM`
DATA-MEM	O: There is no GPIB command to be applied. N: `CALCulate[<chno>]:MATH[:EXPRession]:NAME DSM`
DATA*MEM	O: There is no GPIB command to be applied. N: `CALCulate[<chno>]:MATH[:EXPRession]:NAME DMM`
DATA + MEM	O: There is no GPIB command to be applied. N: `CALCulate[<chno>]:MATH[:EXPRession]:NAME DAM`
OFF	O: `DISPDDM OFF` N: `CALCulate[<chno>]:MATH[:EXPRession]:NAME NONE`
Return	Returns to the display menu (1 of 2). (See step (4).)

(5) AVG

Average menu

<table>
<tr><td>AVG STATE
ON/OFF</td><td>O: AVER <bool>
N: [SENSe:]AVERage[<chno>][:STATe] <bool></td></tr>
<tr><td>AVG COUNT</td><td>O: AVERFACT <int>/ AVR{2|4|8|16|32|64|128}
N: [SENSe:]AVERage[<chno>]:COUNt <int></td></tr>
<tr><td>AVG
RESTART</td><td>O: AVERREST
N: [SENSe:]AVERage[<chno>]:RESTart</td></tr>
<tr><td>GROUP DELAY
APERTURE</td><td>O: APERTP <real>
N: CALCulate[<chno>]:GDAPerture:APERture <real></td></tr>
<tr><td>SMOOTHING
ON/OFF</td><td>O: SMOO <bool>
N: CALCulate[<chno>]:SMOothing:STATe <bool></td></tr>
<tr><td>SMOOTHING
APERTURE</td><td>O: SMOOAPER <REAL>
N: CALCulate[<chno>]:SMOothing:APERture <real></td></tr>
<tr><td>IF RBW
[ ]</td><td>O: RBW <int> / RBW{1K|300|100|30|10}HZ / RBWAUTO
N: [SENSe:]BANDwidth[<chno>][:RESolution] <real>
[SENSe:]BANDwidth[<chno>][:RESolution]:AUTO <bool></td></tr>
</table>

(6) CAL

Calibration menu (1 of 2)

NORMALIZE (THRU)
O: NORM ON
N: [SENSe:]CORRection[<chno>]:COLLect[:ACQuire] NORMalize

NORMALIZE (SHORT)
O: NORMS ON
N: [SENSe:]CORRection[<chno>]:COLLect[:ACQuire] SNORmalize

CAL MENU
Calls the full calibration selection menu. (See step (6-1).)

CORRECT ON/OFF
O: CORRECT <bool>
N: [SENSe:]CORRection[<chno>]:CSET:STATe <bool>

INTERPOLATE ON/OFF
O: INTERPOL
N: [SENSe:]CORRection[<chno>]:CSET:INTerpolate <bool>

Z0 VALUE
O: SETZ0 <real> / MKRZ0{50|75}
N: CALCulate[<chno>]:TRANsform:IMPedance:CIMPedance <real>

More 1/2
Calls the calibration menu (2 of 2).

Calibration menu (2 of 2)

ELEC DELAY ON/OFF
O: LENGTH <bool>
N: [SENSe:]CORRection[<chno>]:EDELay:STATe <bool>

ELECTRICAL DELAY
O: ELED <real>
N: [SENSe:]CORRection[<chno>]:EDELay[:TIME] <real>

ELECTRICAL LENGTH
O: LENGVAL <real>
N: [SENSe:]CORRection[<chno>]:EDELay:DISTance <real>

VELOCITY FACTOR
O: VELOFACT <real>
N: [SENSe:]CORRection[<chno>]:RVELocity:COAX <real>

PHASE OFFSET VALUE
O: PHAO
N: [SENSe:]CORRection[<chno>]:OFFSet:PHASe <real>

PORT EXTENSION
Calls the port extension menu. (See step (7-2).)

More 2/2
Calls the calibration menu (1 of 2).

(6-1) Full calibration selection menu

1PORT FULL CAL	Calls the 1 port full calibration menu. (See step (6-1-1).)
2PORT FULL CAL	Calls the 2 port full calibration menu. (See step (6-2-1).)
CAL KIT [ ]	Calls the calibration kit menu. (See step (6-3-1).)
CLEAR CAL DATA	O: `CLEAR` N: `[SENSe:]CORRection[<chno>]:COLLect:DELete`
Return	Returns to the calibration menu (1 of 2). (See step (6).)

(6-1-1) 1 port full calibration menu

OPEN	O: `OPEN` N: `[SENSe:]CORRection[<chno>]:COLLect[:ACQuire] OPEN`
SHORT	O: `SHORT` N: `[SENSe:]CORRection[<chno>]:COLLect[:ACQuire] SHORt`
LOAD	O: `LOAD` N: `[SENSe:]CORRection[<chno>]:COLLect[:ACQuire] LOAD`
DONE 1-PORT	O: `DONE / DONE1PORT` N: `[SENSe:]CORRection[<chno>]:COLLect:SAVE`

(6-2-1) 2 port full calibration menu

REFLECT'N	Calls the reflection menu. (See step (6-2-2).)
TRANS- MISSION	Calls the transmission menu. (See step (6-2-3).)
ISOLATION	Calls the isolation menu. (See step (6-2-4).)
DONE 2-PORT	O: DONE N: [SENSe:]CORRection[<chno>]:COLLect:SAVE

(6-2-2) Reflection menu

S11: OPEN	O: S11OPEN N: [SENSe:]CORRection[<chno>]:COLLect[:ACQuire] S11Open
S11: SHORT	O: S11SHORT N: [SENSe:]CORRection[<chno>]:COLLect[:ACQuire] S11Short
S11: LOAD	O: S11LOAD N: [SENSe:]CORRection[<chno>]:COLLect[:ACQuire] S11Load
S22: OPEN	O: S22OPEN N: [SENSe:]CORRection[<chno>]:COLLect[:ACQuire] S22Open
S22: SHORT	O: S22SHORT N: [SENSe:]CORRection[<chno>]:COLLect[:ACQuire] S22Short
S22: LOAD	O: S22LOAD N: [SENSe:]CORRection[<chno>]:COLLect[:ACQuire] S22Load
DONE REFLECT'N	O: DONEREFL N: [SENSe:]CORRection[<chno>]:COLLect:SAVE

(6-2-3) Transmission menu

FWD.TRANS THRU	O: FWDTRNS N: [SENSe:]CORRection[<chno>]:COLLect[:ACQuire] FTRansmit
FWD.MATCH THRU	O: FWDMATCH N: [SENSe:]CORRection[<chno>]:COLLect[:ACQuire] FMATch
REV.TRANS THRU	O: REVTRNS N: [SENSe:]CORRection[<chno>]:COLLect[:ACQuire] RTRansmit
REV.MATCH THRU	O: REVMATCH N: [SENSe:]CORRection[<chno>]:COLLect[:ACQuire] RMATch
GROUP THRU	O: There is no GPIB COMMAND to be applied. N: [SENSe:]CORRection[<chno>]:COLLect[:ACQuire] GTHRU
DONE TRANS	O: DONE N: [SENSe:]CORRection[<chno>]:COLLect:SAVE

(6-2-4) Isolation menu

OMIT ISOLATION	O: OMITISO N: [SENSe:]CORRection[<chno>]:COLLect[:ACQuire] OISolation
FWD.ISOL'N	O: FWDISO N: [SENSe:]CORRection[<chno>]:COLLect[:ACQuire] FISolation
REV.ISOL'N	O: REVISO N: [SENSe:]CORRection[<chno>]:COLLect[:ACQuire] RISolation
DONE ISOLATION	O: DONEISO N: [SENSe:]CORRection[<chno>]:COLLect:SAVE

(6-3-1) Calibration kit menu

N(50Ω)	O: `CKIT1` N: `[SENSe:]CORRection[<chno>]:CKIT[:TYPE] 1`
N(75Ω)	O: `CKIT2` N: `[SENSe:]CORRection[<chno>]:CKIT[:TYPE] 2`
3.5mm	O: `CKIT3` N: `[SENSe:]CORRection[<chno>]:CKIT[:TYPE] 3`
7mm	O: `CKIT4` N: `[SENSe:]CORRection[<chno>]:CKIT[:TYPE] 4`
DONT CARE	O: `CKIT0` N: `[SENSe:]CORRection[<chno>]:CKIT[:TYPE] 0`
Return	Returns to the calibration menu. (See step (6).)

(6-3-2) FEMAL/MAL selection menu

PORT 1 FEMAL/MAL	O: `PORT1 FEM/PORT1 MAL` N: `[SENSe:]CORRection[<chno>]:CKIT:TERMinal1 FEMale` N: `[SENSe:]CORRection[<chno>]:CKIT:TERMinal1 MALe`
PORT 2 FEMAL/MAL	O: `PORT2 FEM/PORT2 MAL` N: `[SENSe:]CORRection[<chno>]:CKIT:TERMinal2 FEMale` N: `[SENSe:]CORRection[<chno>]:CKIT:TERMinal2 MALe`
Return	Calls the calibration kit menu. (See step (6-3-1).)

(6-4) Port extension menu

EXTENSION ON/OFF	O: `PORE <bool>` N: `[SENSe:]CORRection[<chno>]:PEXTension:STATe <bool>`
EXTENSION INPUT R	O: `EPORTR <real>` N: `[SENSe:]CORRection[<chno>]:PEXTension:TIME1 <real>`
EXTENSION INPUT A	O: `EPORTA <real>` N: `[SENSe:]CORRection[<chno>]:PEXTension:TIME2 <real>`
EXTENSION INPUT B	O: `EPORTB <real>` N: `[SENSe:]CORRection[<chno>]:PEXTension:TIME3 <real>`
EXTENSION PORT 1 (Note)	O: `EPORT1 <real>` N: `[SENSe:]CORRection[<chno>]:PEXTension:TIME4 <real>`
EXTENSION PORT 2 (Note)	O: `EPORT2 <real>` N: `[SENSe:]CORRection[<chno>]:PEXTension:TIME5 <real>`
Return	Returns to the calibration menu (2 of 2).

Note : This can be set in case of R3765A/67A + S parameter, R3765C/67C and R3765B/67B.

(7) MKR

Marker menu

ACTIVATE MARKER [ ]	Calls the active marker menu (1 of 2). (See step (7-1).)
MARKER ALL OFF	O: `MKRAOFF` N: `MARKer[<chno>]:AOFF`
ΔMODE MENU	Calls the delta mode menu. (See step (7-2).)
MKR LIST ON/OFF	O: There is no GPIB command to be applied. N: `MARKer[<chno>]:LIST <bool>`
MARKER MODE MENU	Calls the marker mode menu. (See step (7-3).)

For acquiring the marker data, the following commands can be used.

O: `MKR{1|2|3|4|5|6|7|8|9|10}A?`

N: `FETch[<chno>][:MARKer][:ACTivate]?`
`FETch[<chno>][:MARKer]:NUMBer<n>?`

(7-1) Active marker menu (1 of 2)

MARKER 1	O: MKR1A <real> N: MARKer[<chno>]:ACTivate[:NUMBer] 1[,<real>]
MARKER 2	O: MKR2A <real> N: MARKer[<chno>]:ACTivate[:NUMBer] 2[,<real>]
MARKER 3	O: MKR3A <real> N: MARKer[<chno>]:ACTivate[:NUMBer] 3[,<real>]
MARKER 4	O: MKR4A <real> N: MARKer[<chno>]:ACTivate[:NUMBer] 4[,<real>]
MARKER 5	O: MKR5A <real> N: MARKer[<chno>]:ACTivate[:NUMBer] 5[,<real>]
ACTIVATE MKR OFF	O: MKROFF N: MARKer[<chno>]:ACTivate:STATe <bool>
Return	Returns to the marker menu. (See step (7).)
More 1/2	Calls the active marker menu (2 of 2).

Active marker menu (2 of 2)

MARKER 6	O: MKR6A <real> N: MARKer[<chno>]:ACTivate[:NUMBer] 6[,<real>]
MARKER 7	O: MKR7A <real> N: MARKer[<chno>]:ACTivate[:NUMBer] 7[,<real>]
MARKER 8	O: MKR8A <real> N: MARKer[<chno>]:ACTivate[:NUMBer] 8[,<real>]
MARKER 9	O: MKR9A <real> N: MARKer[<chno>]:ACTivate[:NUMBer] 9[,<real>]
MARKER 10	O: MKR10A <real> N: MARKer[<chno>]:ACTivate[:NUMBer] 10[,<real>]
ACTIVATE MKR OFF	O: MKROFF N: MARKer[<chno>]:ACTivate:STATe <bool>
Return	Returns to the marker menu. (See step (7).)
More 2/2	Calls the active marker menu (1 of 2).

(7-2) Delta mode menu

ΔMODE OFF	O: DMKROF N: MARKer[<chno>]:DELTa[:MODE] OFF
ΔREF = ΔMKR	O: DMKRC N: MARKer[<chno>]:DELTa[:MODE] CHILd
ΔREF = ACT MKR	Calls the ACT MKR menu. (See step (7-2-1).) O: DMKRA N: MARKer[<chno>]:DELTa[:MODE] COMPare
ΔREF = FIXED MKR	O: DMKRF N: MARKer[<chno>]:DELTa[:MODE] FIXed
FIXED MKR POSITION	Calls FIXED MKR setting menu. (See step (7-2-2).)
Return	Returns to the marker menu. (See step (7).)

Select the compare marker before setting the delta mode to ΔREF = ACT MKR. (See ACT MKR menu.)

(7-2-1) ACT MKR menu (1 of 2)

COMPARE MARKER 1	O: DMKR10 <real> N: MARKer[<chno>]:DELTa:COMPare 1[,<real>]
COMPARE MARKER 2	O: DMKR20 <real> N: MARKer[<chno>]:DELTa:COMPare 2[,<real>]
COMPARE MARKER 3	O: DMKR30 <real> N: MARKer[<chno>]:DELTa:COMPare 3[,<real>]
COMPARE MARKER 4	O: DMKR40 <real> N: MARKer[<chno>]:DELTa:COMPare 4[,<real>]
COMPARE MARKER 5	O: DMKR50 <real> N: MARKer[<chno>]:DELTa:COMPare 5[,<real>]
ACTIVATE MARKER [ ]	Calls the active marker menu (1 of 2). (See step (7-1).)
Return	Returns to the delta mode menu. (See step (7-2).)
More 1/2	Calls ACT MKR menu (2 of 2).

ACT MKR menu (2 of 2)

COMPARE MARKER 6	O: DMKR60 <real> N: MARKer[<chno>]:DELTa:COMPare 6[,<real>]
COMPARE MARKER 7	O: DMKR70 <real> N: MARKer[<chno>]:DELTa:COMPare 7[,<real>]
COMPARE MARKER 8	O: DMKR80 <real> N: MARKer[<chno>]:DELTa:COMPare 8[,<real>]
COMPARE MARKER 9	O: DMKR90 <real> N: MARKer[<chno>]:DELTa:COMPare 9[,<real>]
COMPARE MARKER 10	O: DMKR100 <real> N: MARKer[<chno>]:DELTa:COMPare 10[,<real>]
ACTIVATE MARKER [ ]	Calls the active marker menu (1 of 2). (See step (7-1).)
Return	Returns to the delta mode menu. (See step (7-2).)
More 2/2	Calls ACT MKR menu (1 of 2).

(7-2-2) FIXED MKR setting menu (1 of 2)

FIXED MKR STIMULUS	O: FMKRS <real> N: MARKer[<chno>]:FIXed:STIMulus <real>
FIXED MKR VALUE	O: FMKRV <real> N: MARKer[<chno>]:FIXed:VALue <real>
FIXED MKR AUX VALUE	O: There is no GPIB command to be applied. N: MARKer[<chno>]:FIXed:AVALue <real>
FIXED MKR→ ACTIVE MKR	O: MKRFIX N: MARKer[<chno>]:LET FIXed
Return	Returns to the delta mode menu. (See step (7-2).)

(7-3) Marker mode menu

MKR CMP/UNCMP	O: MKRCMP/ MKRUCMP N: MARKer[<chno>]:COMPensate <bool>
MKR CPL/UNCPL	O: MKRCOUP/ MKRUCOUP N: MARKer[<chno>]:COUPle <bool>
CONVERSION MKR MENU [ ]	Calls the conversion marker menu. (See step (7-3-1).)
SMITH MKR MENU [ ]	Calls the smith marker menu. (See step (7-3-2).)
POLAR MKR MENU [ ]	Calls the polar marker menu . . (See step (7-3-3).)
Return	Returns to the marker menu. (See step (7).)

(7-3-1) Conversion marker menu

DEFAULT	O: ZYMKDFLT N: MARKer[<chno>]:CONVert[:MODE] DEFault
LIN MKR	O: ZYMKLIN N: MARKer[<chno>]:CONVert[:MODE] LINear
Re/Im	O: ZYMKRI N: MARKer[<chno>]:CONVert[:MODE] RIMaginary
Return	Returns to the marker mode menu. (See step (7-3).)

(7-3-2) Smith marker menu

LIN MKR	O: `SMKRLIN` N: `MARKer[<chno>]:SMITh MLINear`
LOG MKR	O: `SMKRLOG` N: `MARKer[<chno>]:SMITh MLOGarithmic`
Re/Im MKR	O: `SMKRRI` N: `MARKer[<chno>]:SMITh RIMaginary`
R+jX MKR	O: `SMKRRX` N: `MARKer[<chno>]:SMITh IMPedance`
G+jB MKR	O: `SMKRGB` N: `MARKer[<chno>]:SMITh ADMittance`
Z0 VALUE	O: `SETZO <real> / MKRZO{50|75}` N: `CALCulate[<chno>]:TRANsform:IMPedance:CIMPedance <real>`
Return	Returns to the marker mode menu. (See step (7-3).)

(7-3-3) Polar marker menu

LIN MKR	O: `PMKRLIN` N: `MARKer[<chno>]:POLar MLINear`
LOG MKR	O: `PMKRLOG` N: `MARKer[<chno>]:POLar MLOGarithmic`
Re/Im MKR	O: `PMKRRI` N: `MARKer[<chno>]:POLar RIMaginary`
Z0 VALUE	O: `SETZO <real> / MKRZO{50|75}` N: `CALCulate[<chno>]:TRANsform:IMPedance:CIMPedance <real>`
Return	Returns to the marker mode menu. (See step (7-3).)

(8) MKR→

Marker search menu

MARKER→ START	O: MKRSTAR N: MARKer[<chno>]:LET STARt
MARKER→ STOP	O: MKRSTOP N: MARKer[<chno>]:LET STOP
MARKER→ CENTER	O: MKRCENT N: MARKer[<chno>]:LET CENTer
MARKER→ SPAN	O: MKRSPAN N: MARKer[<chno>]:LET SPAN
MARKER→ REF.VALUE	O: MKRREF N: MARKer[<chno>]:LET RLEVel
PART SRCH [ ]	Calls the partial search menu. (See step (8-1).)
MKR SEARCH [ ]	Calls the search menu. (See step (8-2).)

(8-1) Partial search menu

ΔMODE MENU	Calls the delta mode menu. (See step (7-2).)
SET RANGE	O: There is no GPIB command to be applied. N: MARKer[<chno>]:SEARch:PARTial:SRANge
PART SRCH ON/OFF	O: MKRPART <bool> N: MARKer[<chno>]:SEARch:PARTial[:STATe] <bool>
Return	Returns to the marker search menu. (See step (8).)

(8-2) Search menu

MKR SEARCH OFF	O: SRCHOFF N: MARKer[<chno>]:SEARch[:MODE] OFF
MAX	O: MAXSRCH N: MARKer[<chno>]:SEARch[:MODE] MAX
MIN	O: MINSRCH N: MARKer[<chno>]:SEARch[:MODE] MIN
TARGET	Calls the target menu. (See step (8-2-1).) O: ZRPSRCH (0° SEARCH) N: MARKer[<chno>]:SEARch[:MODE] TARGet
RIPPLE	Calls the ripple menu. (See step (8-2-2).) O: DRIPPL1 N: MARKer[<chno>]:SEARch[:MODE] RIPPle
FLTR ANAL	Calls the filter analysis menu. (See step (8-2-3).)
TRACKING ON/OFF	O: MKRTRAC <bool> N: MARKer[<chno>]:SEARch:TRACking <bool>
Return	Returns to the marker search menu. (See step (8).)

(8-2-1) Target menu

TARGET VALUE	O: There is no command to be applied. N: MARKer[<chno>]:SEARch:TARGet[:MODE] VALue MARKer[<chno>]:SEARch:TARGet:VALue <real>
0°	O: ZRPSRCH N: MARKer[<chno>]:SEARch:TARGet[:MODE] ZERO
±180°	O: There is no command to be applied. N: MARKer[<chno>]:SEARch:TARGet[:MODE] PI
LEFT SEARCH	O: There is no command to be applied. N: MARKer[<chno>]:SEARch:TARGet:LEFT
RIGHT SEARCH	O: There is no command to be applied. N: MARKer[<chno>]:SEARch:TARGet:RIGHt
Return	Returns to the search menu. (See step (8-2).)

(8-2-2) Ripple menu

Softkey	Command
MAX∩	O: There is no command to be applied. N: MARKer[<chno>]:SEARch:RIPPle[:MODE] MAX
MIN∪	O: There is no command to be applied. N: MARKer[<chno>]:SEARch:RIPPle[:MODE] MIN
ΔMAX∩ -MIN∪	O: DRIPPL1 N: MARKer[<chno>]:SEARch:RIPPle[:MODE] BOTH
MAX-MIN	O: DMAXMIN N: MARKer[<chno>]:SEARch:RIPPle[:MODE]PPEak
ΔX	O: DLTX <real> N: MARKer[<chno>]:SEARch:RIPPle:DX <real>
ΔY	O: DLTY <real> N: MARKer[<chno>]:SEARch:RIPPle:DY <real>
Return	Returns to the search menu. (See step (8-2).)

(8-2-3) Filter analysis menu

Softkey	Command
WIDTH VALUE	O: T{3|6|60}DB/ T{3|6}DEG/ TXDB <real>/ TXDEG <real> N: MARKer[<chno>]:FANalysis:WIDTh <real>
SEARCH IN/OUT	O: TIN/ TOUT N: MARKer[<chno>]:FANalysis:DIRection {IN|OUT}
FILTER ANAL ON/OFF	O: FLTANA <bool> N: MARKer[<chno>]:FANalysis[:STATe] <bool>
Return	Returns to the search menu. (See step (8-2).)

The data of filter analysis can be acquired by the following command.

O: TXDB?/ TXDEG?

N: FETch[<chno>][:MARKer]:FANalysis?

A2.4 INSTRUMENT STATE Block

(1) SAVE
Save menu

SAVE REGISTER	Calls the save register menu (1 of 2). (See step (1-1).)
CLEAR REGISTER	Calls the clear register menu (1 of 2). (See step (1-2).)
STORE FILE	Calls the store file menu. (See step (1-3).)
PURGE FILE	Calls the purge file menu. (See step (1-4).)
FORMAT DISK	There is no GPIB command to be applied.

(1-1) Save register menu (1 of 2)

SAVE REG-1	O: `SAVEREG1` N: `*SAV 1/ REGister:SAVE 1`
SAVE REG-2	O: `SAVEREG2` N: `*SAV 2/ REGister:SAVE 2`
SAVE REG-3	O: `SAVEREG3` N: `*SAV 3/ REGister:SAVE 3`
SAVE REG-4	O: `SAVEREG4` N: `*SAV 4/ REGister:SAVE 4`
SAVE REG-5	O: `SAVEREG5` N: `*SAV 5/ REGister:SAVE 5`
RENAME REG	There is no GPIB command to be applied.
Return	Returns to the save menu. (See step (1).)
More 1/2	Calls the save register menu (2 of 2).

Save register menu (2 of 2)

Softkey	Command
SAVE REG-6	O: SAVEREG6 N: *SAV 6/ REGister:SAVE 6
SAVE REG-7	O: SAVEREG7 N: *SAV 7/ REGister:SAVE 7
SAVE REG-8	O: SAVEREG8 N: *SAV 8/ REGister:SAVE 8
SAVE REG-9	O: SAVEREG9 N: *SAV 9/ REGister:SAVE 9
SAVE REG-10	O: SAVEREG10 N: *SAV 10/ REGister:SAVE 10
RENAME REG	There is no GPIB command to be applied.
Return	Returns to the save menu. (See step (1).)
More 2/2	Calls the save register menu (1 of 2).

(1-2) Clear register menu (1 of 2)

Softkey	Command
CLEAR REG-1	O: CLRREG1 N: REGister:CLEar 1
CLEAR REG-2	O: CLRREG2 N: REGister:CLEar 2
CLEAR REG-3	O: CLRREG3 N: REGister:CLEar 3
CLEAR REG-4	O: CLRREG4 N: REGister:CLEar 4
CLEAR REG-5	O: CLRREG5 N: REGister:CLEar 5
Return	Returns to the save menu. (See step (1).)
More 1/2	Calls the clear register menu (2 of 2).

Clear register menu (2 of 2)

CLEAR REG-6	O: `CLRREG6` N: `REGister:CLEar 6`
CLEAR REG-7	O: `CLRREG7` N: `REGister:CLEar 7`
CLEAR REG-8	O: `CLRREG8` N: `REGister:CLEar 8`
CLEAR REG-9	O: `CLRREG9` N: `REGister:CLEar 9`
CLEAR REG-10	O: `CLRREG10` N: `REGister:CLEar 10`
Return	Returns to the save menu. (See step (1).)
More 2/2	Calls the clear register menu (1 of 2).

(1-3) Store file menu

STORE	O: `STFILE <str>` N: `FILE:STORe <str>`
ROLL ↑	There is no GPIB command to be applied.
ROLL ↓	There is no GPIB command to be applied.
DEFINE STORE	Calls the file data menu. (See step (1-3-1).)
EDIT NAME	There is no GPIB command to be applied.
NAME ↑	There is no GPIB command to be applied.
NAME ↓	There is no GPIB command to be applied.
CANCEL	There is no GPIB command to be applied.

<str> in "STORE" is file name.

(1-3-1) File data menu

STATE ON/OFF	O: DSSTATE <bool> N: FILE:STATe:CONDition <bool>
RAY ARRAY ON/OFF	O: RAWARY <bool> N: FILE:STATe:RAW <bool>
CORR COEF ON/OFF	O: CORARY <bool> N: FILE:STATe:CORRection <bool>
DATA ARRAY ON/OFF	O: DATAARY <bool> N: FILE:STATe:DATA <bool>
MEM ARRY ON/OFF	O: MEMARY <bool> N: FILE:STATe:MEMory <bool>
Return	Returns to the save menu. (See step (1).)

(1-4) Purge file menu

PURGE	O: PURGE <str> N: FILE:DELete <str>
CURSOR ↑	There is no GPIB command to be applied.
CURSOR ↓	There is no GPIB command to be applied.
Return	Returns to the save menu. (See step (1).)

<str> in "PURGE" is file name.

(2) RECALL

Recall menu (1 of 2)

RECALL REG-1	O: RECLREG1 N: *RCL 1/ REGister:RECall 1
RECALL REG-2	O: RECLREG2 N: *RCL 2/ REGister:RECall 2
RECALL REG-3	O: RECLREG3 N: *RCL 3/ REGister:RECall 3
RECALL REG-4	O: RECLREG4 N: *RCL 4/ REGister:RECall 4
RECALL REG-5	O: RECLREG5 N: *RCL 5/ REGister:RECall 5
RECALL POWER OFF	O: RECLPOFF N: *RCL POFF/ REGister:RECall POFF
LOAD FILE	O: LDFILE <str> N: FILE:LOAD <str>
More 1/2	Calls the recall menu (2 of 2).

<str> in "LOAD FILE" is file name.

Recall menu (2 of 2)

RECALL REG-6	O: RECLREG6 N: *RCL 6/ REGister:RECall 6
RECALL REG-7	O: RECLREG7 N: *RCL 7/ REGister:RECall 7
RECALL REG-8	O: RECLREG8 N: *RCL 8/ REGister:RECall 8
RECALL REG-9	O: RECLREG9 N: *RCL 9/ REGister:RECall 9
RECALL REG-10	O: RECLREG10 N: *RCL 10/ REGister:RECall 10
RECALL POWER OFF	O: RECLPOFF N: *RCL POFF/ REGister:RECall POFF
LOAD FILE	O: LDFILE <str> N: FILE:LOAD <str>
More 2/2	Calls the recall menu (1 of 2).

<str> in"LOAD FILE" is file name.

(3) SYSTEM
System menu

SYSTEM DRIVE	There is no GPIB command to be applied. See Note.
SET CLOCK	Calls the real time clock menu. (See step (3-1).)
LIMIT MENU	Calls the limit menu. (See step (3-2-1).)

Note: Specify the drive name with the file name as follows:
"[drive name:]<file name>"

(3-1) Real time clock menu

YEAR	O: YEAR <int> N: SYSTem:DATE <year>,<month>,<day>
MONTH	O: MONTH <int> N: SYSTem:DATE <year>,<month>,<day>
DAY	O: DAY <int> N: SYSTem:DATE <year>,<month>,<day>
HOUR	O: HOUR <int> N: SYSTem:TIME <hour>,<minute>,<second>
MINUTE	O: MINUTE <int> N: SYSTem:TIME <hour>,<minute>,<second>
SECOND	O: SECOND <int> N: SYSTem:TIME <hour>,<minute>,<second>
Return	Returns to the system menu. (See step (3).)

(3-2-1) Limit menu

LIMIT LINE ON/OFF	O: `LIMILINE` N: `DISPlay[:WINDow[<chno>]]:LIMit[pn]:LINE <bool>`
LIMIT TEST ON/OFF	O: `LIMITEST` N: `DISPlay[:WINDow[<chno>]]:LIMit[pn][:STATe] <bool>`
BEEP FAIL ON/OFF	O: `BEEPFAIL` N: `DISPlay[:WINDow[<chno>]]:LIMit[pn]:BEEP <bool>`
LIMIT MODE MENU	Calls the limit mode menu. (See step (3-2-2).)
EDIT LIMIT LINE	Calls the edit limits menu. (See step (3-2-4).)
SELECT DATA 1ST/2ND	O: There is no GPIB command to be applied. N: There is no GPIB command to be applied.
LIMIT LINE OFFSETS	Calls the offset limits menu. (See step (3-2-8).)
Return	Calls the system menu. (See step (3).)

(3-2-2) Limit mode menu

1ST DATA ON/OFF	O: There is no GPIB command to be applied. N: `DISPlay[:WINDow[<chno>]]:LIMit[<pn>]:PARameter[:STATe]`
2ND DATA ON/OFF	O: There is no GPIB command to be applied. N: `DISPlay[:WINDow[<chno>]]:LIMit[<pn>]:PARameter[:STATe]`
SMITH LIMIT MENU	Calls the limit parameter menu. (See step (3-2-3).)
POLAR LIMIT MENU	Calls the limit parameter menu. (See step (3-2-3.)
Return	Calls the limit menu. (See step (3-2-1).)

(3-2-3) Limit parameter menu

RF/IM LIMIT

O: There is no GPIB command to be applied.

N: Smith display
DISPlay[:WINDow[<chno>]]:LIMit[<pn>]:PARameter:Smith LIMit {RIMaginary | RhoTHeta}
Polar display
DISPlay[:WINDow[<chno>]]:LIMit[<pn>]:PARameter:Polar LIMit {RIMaginary | RhoTHeta}

MAG/PHASE LIMIT

O: There is no GPIB command to be applied.

N: Smith display
DISPlay[:WINDow[<chno>]]:LIMit[<pn>]:PARameter:Smith LIMit {RIMaginary | RhoTHeta}
Polar display
DISPlay[:WINDow[<chno>]]:LIMit[<pn>]:PARameter:Polar LIMit {RIMaginary | RhoTHeta}

Return

Calls the limit mode menu. (See step (3-2-2).)

(3-2-4) Edit limits menu

SEGMENT

O: LSEG

N: There is no GPIB command to be applied.

EDIT SEGMENT

O: There is no GPIB command to be applied.

N: There is no GPIB command to be applied.

DELETE

O: There is no GPIB command to be applied.

N: DISPlay[:WINDow[<chno>]]:LIMit[<pn>]:SEGMent<n>:DELete

ADD SEGMENT

O: There is no GPIB command to be applied.

N: There is no GPIB command to be applied.

CLEAR

Calls the clear limit menu. (See step (3-2-6).)

LINE TYPE

Calls the limit type menu. (See step (3-2-7).)

DONE

O: There is no GPIB command to be applied.

N: There is no GPIB command to be applied.

(3-2-5) Edit segment menu

STIMULUS VALUE	O: LIMS N: DISPlay[:WINDow[<chno>]]:LIMit[pn]:SEGMent<n>:STIMulus <real>
MARKER TO STIMULUS	O: There is no GPIB command to be applied. N: There is no GPIB command to be applied.
UPPER LIMIT	O: LIMU N: DISPlay[:WINDow[<chno>]]:LIMit[pn]:SEGMent<n>:UPPer <real>
LOWER LIMIT	O: LIML N: DISPlay[:WINDow[<chno>]]:LIMit[pn]:SEGMent<n>:LOWer <real>
DELTA LIMIT	O: There is no GPIB command to be applied. N: There is no GPIB command to be applied.
MIDDLE VALUE	O: There is no GPIB command to be applied. N: There is no GPIB command to be applied.
MARKER TO MIDDLE	O: There is no GPIB command to be applied. N: There is no GPIB command to be applied.
Return	O: There is no GPIB command to be applied. N: There is no GPIB command to be applied.

(3-2-6) Clear limit menu

YES	O: LSEGCL N: DISPlay[:WINDow[<chno>]]:LIMit[pn]:CLEar
NO	O: There is no GPIB command to be applied. N: There is no GPIB command to be applied.

(3-2-7) Limit type menu

SLOPING LINE	O: `LIMTSLP` N: `DISPlay[:WINDow[<chno>]]:LIMit[pn]:SEGMent<n>:TYPE Slope LINe`
FLAT LINE	O: `LIMTFLT` N: `DISPlay[:WINDow[<chno>]]:LIMit[pn]:SEGMent<n>:TYPE Flat LINe`
SINGLE POINT	O: `LIMTSP` N: `DISPlay[:WINDow[<chno>]]:LIMit[pn]:SEGMent<n>:TYPE Single Point`
LIMIT COLOR	O: `LIMC` N: `LIMC:COLor`
WAVE COLOR	O: `LIMWC` N: `LIMWC:COLor`
Return	Calls the edit limits menu. (See step (3-2-4).)

(3-2-8) Offset limits menu

STIMULUS OFFSET	O: There is no GPIB command to be applied. N: `DISPlay[:WINDow[<chno>]]:LIMit[pn]:OFFSet:STIMulus <real>`
AMPLITUDE OFFSET	O: There is no GPIB command to be applied. N: `DISPlay[:WINDow[<chno>]]:LIMit[pn]:OFFSet:AMPLitude <real>`
MARKER TO AMP. OFS	O: There is no GPIB command to be applied. N: There is no GPIB command to be applied.
Return	Calls the limits menu. (See step (3-2-1).)

(4) PRESET

PRESET	O: `IP` N: `SYSTem:PRESet`

A2.5 GPIB Block

(1) PROGRAM

PROGRAM	There is no GPIB command to be applied to the following menus which are called by this key. • Controller menu • Load menu • Drive menu

(2) REMOTE/LCL
GPIB menu

SYSTEM CONTROLLER	There is no GPIB command to be applied.
TALKER LISTENER	There is no GPIB command to be applied.
SET ADDRESSES	Calls the address menu. (See step (2-1).)

(2-1) Address menu

ADDRESS R3765 (Note)	There is no GPIB command to be applied. Note: In the case of R3767, the address menu is displayed with R3767.
ADDRESS PLOTTER	There is no GPIB command to be applied.
ADDRESS PRINTER	There is no GPIB command to be applied.
Return	Returns to the GPIB menu. (See step (2).)

MEMO

A3. Initial Settings

Table A3-1 Initial Settings (1 of 3)

Function	Initialization Method	
	Power ON or Preset	*RST
Stimulus		
Sweeping type	Linear frequency sweeping	Same as left column
Continuous sweeping	ON	OFF
Trigger source	Internal (FREE RUN)	Same as left column
Trigger delay	OFF (0sec)	Same as left column
Sweeping time	190.95msec (AUTO) (R3764/65 series)	240.2msec (Auto) (R3764/65 series)
	402.0msec (AUTO) (R3766/67 series)	420.35msec (AUTO) (R3766/67 series)
Number of measurement point	201	1201
Start frequency	5Hz	Same as left column
Stop frequency	3.8GHz (R3764/65 series)	Same as left column
	8.0GHz (R3766/67 series)	Same as left column
Center frequency	1.92GHz (R3764/65 series)	Same as left column
	4.02Hz (R3766/67 series)	Same as left column
Frequency span	3.76GHz (R3764/65 series)	Same as left column
	7.96GHz (R3766/67 series)	Same as left column
Frequency display	Start/Stop	Same as left column
Fixed frequency of level sweeping	1GHz	Same as left column
Output level	*1	Same as left column
Start level	*2	Same as left column
Stop level	*2	Same as left column
2-channel interlocking	ON	Same as left column
Program sweeping segment	All clear	Same as left column
Response		
Dual channel	OFF	Same as left column
Active channel	CH1	Same as left column
Resolution bandwidth	10kHz	Same as left column
Input port selection condition	*3	Same as left column
Averaging	OFF (number of times: 16)	Same as left column
Trace operation	NONE	Same as left column
Conversion	NONE	Same as left column
Characteristic impedance ZO	50Ω	Same as left column
Measurement format	*4	Same as left column
Group delay aperture	10%	0.01%
Smoothing	OFF (Aperture 10%)	OFF (Aperture 0.01%)
Display	Data	Same as left column
Split/Overlap	Overlap	Same as left column
Label	Non	Same as left column

*1: Output level

Type	Power ON or Preset	*RST
A	0dBm	Same as left column
B	0dBm	Same as left column
C A + S parameter	10dBm	Same as left column

*2: Start/Stop level

Type	Power ON or Preset		*RST	
	Start	Stop	Start	Stop
A	-13dBm	0dBm	Same as left column	22dBm
B	-15dBm	0dBm	Same as left column	20dBm
C A + S parameter	-20dBm	0dBm	Same as left column	10dBm

*3: Input port selection condition

Type \ Channel	CH1	CH2	CH3	CH4
A	A/R	B/R	A/R	B/R
B	REFLECTION	TRANSMISSION	REFLECTION	TRANSMISSION
C A + S parameter	S11	S21	S11	S21

*4: Measurement format

Type \ Channel	CH1	CH2	CH3	CH4
A	LOGMAG	LOGMAG	LOGMAG	LOGMAG
B	LOGMAG	LOGMAG	POLAR	LOGMAG
C A + S parameter	LOGMAG	LOGMAG	POLAR	LOGMAG

Table A3-1 Initial Settings (2 of 3)

Function	Initialization Method	
	Power ON or Preset	*RST
Reference value		
Logarithm amplitude	0dB	Same as left column
Phase	0°	Same as left column
Group delay	0sec	Same as left column
Smith chart	1	Same as left column
Polar coordinate	1	Same as left column
Linear amplitude	0	Same as left column
SWR	1	Same as left column
Real part	10	Same as left column
Imaginary part	10	Same as left column
Continuous phase	0°	Same as left column
The value per division of Y-axis		
Logarithm amplitude	*5	Same as left column
Phase	45°	Same as left column
Group delay	100nsec	Same as left column
Smith chart	-	Same as left column
Polar coordinate	-	Same as left column
Linear amplitude	100m	Same as left column
SWR	1	Same as left column
Real part	1	Same as left column
Imaginary part	1	Same as left column
Continuous phase	360°	Same as left column
Reference position		
Logarithm amplitude	*6	Same as left column
Phase	50%	Same as left column
Group delay	50%	Same as left column
Smith chart	-	Same as left column
Polar coordinate	-	Same as left column
Linear amplitude	0%	Same as left column
SWR	0%	Same as left column
Real part	100%	Same as left column
Imaginary part	100%	Same as left column
Continuous phase	50%	Same as left column

*5: Logarithm amplitude (The value per division of Y-axis)

Type \ Channel	CH1	CH2	CH3	CH4
A	10dB	10dB	1dB	1dB
B	5dB	10dB	1 UNIT	1dB
C A + S parameter	5dB	10dB	1 UNIT	1dB

*6: Logarithm amplitude (Reference position)

Type \ Channel	CH1	CH2	CH3	CH4
A	90%	90%	90%	90%
B	90%	90%	—	90%
C A + S parameter	90%	90%	—	90%

Table A3-1 Initial Settings (3 of 3)

Function	Initialization Method	
	Power ON or Preset	*RST
Calibration		
Correction measurement	OFF	Same as left column
Calibration data	Clear	Same as left column
Electrical length correction	OFF(0sec)	Same as left column
Phase offset	OFF(0°)	Same as left column
Measurement end extension correction	OFF	Same as left column
R Input	0sec	Same as left column
A Input	0sec	Same as left column
B Input	0sec	Same as left column
Port 1	0sec	Same as left column
Port 2	0sec	Same as left column
Propagation constant	1	Same as left column

Table A3-2 Backup Memory Settings (factory default settings)

Item	Initial Setting
Analyzer GPIB address	11
System controller/addressable	Addressable
Printer GPIB address	18
Plotter GPIB address	5
Save register	All clear

MEMO

A4. Multi-Line Interface Message

<table>
<tr><td rowspan="4"></td><td colspan="12">PCG</td><td colspan="4" rowspan="2">SCG</td></tr>
<tr><td colspan="2">ACG</td><td colspan="2">UCG</td><td colspan="4">LAG</td><td colspan="4">TAG</td></tr>
<tr><td colspan="2">0</td><td colspan="2">1</td><td colspan="2">2</td><td colspan="2">3</td><td colspan="2">4</td><td colspan="2">5</td><td colspan="2">6</td><td colspan="2">7</td></tr>
<tr><td>ascii</td><td>msg</td><td>ascii</td><td>msg</td><td>ascii</td><td>msg</td><td>ascii</td><td>msg</td><td>ascii</td><td>msg</td><td>ascii</td><td>msg</td><td>ascii</td><td>msg</td><td>ascii</td><td>msg</td></tr>
<tr><td>0</td><td>NUL</td><td></td><td>DEL</td><td></td><td>SP</td><td rowspan="16">(1)</td><td>0</td><td rowspan="15">(1)</td><td>@</td><td rowspan="16">(2)</td><td>P</td><td rowspan="15">(2)</td><td>`</td><td></td><td>p</td><td></td></tr>
<tr><td>1</td><td>SOH</td><td>GTL</td><td>DC1</td><td>LLO</td><td>!</td><td>1</td><td>A</td><td>Q</td><td>a</td><td></td><td>q</td><td></td></tr>
<tr><td>2</td><td>STX</td><td></td><td>DC2</td><td></td><td>"</td><td>2</td><td>B</td><td>R</td><td>b</td><td></td><td>r</td><td></td></tr>
<tr><td>3</td><td>ETX</td><td></td><td>DC3</td><td></td><td>#</td><td>3</td><td>C</td><td>S</td><td>c</td><td></td><td>s</td><td></td></tr>
<tr><td>4</td><td>EOT</td><td>SDC</td><td>DC4</td><td>DCL</td><td>$</td><td>4</td><td>D</td><td>T</td><td>d</td><td></td><td>t</td><td></td></tr>
<tr><td>5</td><td>ENQ</td><td>PPC</td><td>NAK</td><td>PPU</td><td>%</td><td>5</td><td>E</td><td>U</td><td>e</td><td></td><td>u</td><td></td></tr>
<tr><td>6</td><td>ACK</td><td></td><td>SYN</td><td></td><td>&</td><td>6</td><td>F</td><td>V</td><td>f</td><td></td><td>v</td><td></td></tr>
<tr><td>7</td><td>BEL</td><td></td><td>ETB</td><td></td><td>'</td><td>7</td><td>G</td><td>W</td><td>g</td><td></td><td>w</td><td></td></tr>
<tr><td>8</td><td>BS</td><td>GET</td><td>CAN</td><td>SPE</td><td>(</td><td>8</td><td>H</td><td>X</td><td>h</td><td></td><td>x</td><td></td></tr>
<tr><td>9</td><td>HT</td><td>TCT</td><td>EM</td><td>SPD</td><td>)</td><td>9</td><td>I</td><td>Y</td><td>i</td><td></td><td>y</td><td></td></tr>
<tr><td>10</td><td>LF</td><td></td><td>SUB</td><td></td><td>*</td><td>:</td><td>J</td><td>Z</td><td>j</td><td></td><td>z</td><td></td></tr>
<tr><td>11</td><td>VT</td><td></td><td>ESC</td><td></td><td>+</td><td>;</td><td>K</td><td>[</td><td>k</td><td></td><td>{</td><td></td></tr>
<tr><td>12</td><td>FF</td><td></td><td>FS</td><td></td><td>,</td><td><</td><td>L</td><td>\</td><td>l</td><td></td><td>|</td><td></td></tr>
<tr><td>13</td><td>CR</td><td></td><td>GS</td><td></td><td>-</td><td>=</td><td>M</td><td>]</td><td>m</td><td></td><td>}</td><td></td></tr>
<tr><td>14</td><td>SO</td><td></td><td>RS</td><td></td><td>.</td><td>></td><td>N</td><td>^</td><td>n</td><td></td><td>-</td><td></td></tr>
<tr><td>15</td><td>SI</td><td></td><td>US</td><td></td><td>/</td><td>?</td><td>UNL</td><td>O</td><td>_</td><td>UNT</td><td>o</td><td></td><td>DEL</td><td></td></tr>
</table>

Note:
PCG: Primary command group
ACG: Address command group
UCG: Universal command group
LAG: Listener address group
TAG: Talker address group
SCG: Second command group (defined by PCG)
(1): Listener address to be allocated for devices
(2): Talker address to be allocated for devices

MEMO

ALPHABETICAL INDEX

IMPORTANT INFORMATION FOR ADVANTEST SOFTWARE

PLEASE READ CAREFULLY: This is an important notice for the software defined herein. Computer programs including any additions, modifications and updates thereof, operation manuals, and related materials provided by Advantest (hereafter referred to as "SOFTWARE"), included in or used with hardware produced by Advantest (hereafter referred to as "PRODUCTS").

SOFTWARE License

All rights in and to the SOFTWARE (including, but not limited to, copyright) shall be and remain vested in Advantest. Advantest hereby grants you a license to use the SOFTWARE only on or with Advantest PRODUCTS.

Restrictions

(1) You may not use the SOFTWARE for any purpose other than for the use of the PRODUCTS.
(2) You may not copy, modify, or change, all or any part of, the SOFTWARE without permission from Advantest.
(3) You may not reverse engineer, de-compile, or disassemble, all or any part of, the SOFTWARE.

Liability

Advantest shall have no liability (1) for any PRODUCT failures, which may arise out of any misuse (misuse is deemed to be use of the SOFTWARE for purposes other than it's intended use) of the SOFTWARE. (2) For any dispute between you and any third party for any reason whatsoever including, but not limited to, infringement of intellectual property rights.

LIMITED WARRANTY

1. Unless otherwise specifically agreed by Seller and Purchaser in writing, Advantest will warrant to the Purchaser that during the Warranty Period this Product (other than consumables included in the Product) will be free from defects in material and workmanship and shall conform to the specifications set forth in this Operation Manual.

2. The warranty period for the Product (the "Warranty Period") will be a period of one year commencing on the delivery date of the Product.

3. If the Product is found to be defective during the Warranty Period, Advantest will, at its option and in its sole and absolute discretion, either (a) repair the defective Product or part or component thereof or (b) replace the defective Product or part or component thereof, in either case at Advantest's sole cost and expense.

4. This limited warranty will not apply to defects or damage to the Product or any part or component thereof resulting from any of the following:

   (a) any modifications, maintenance or repairs other than modifications, maintenance or repairs (i) performed by Advantest or (ii) specifically recommended or authorized by Advantest and performed in accordance with Advantest 's instructions;

   (b) any improper or inadequate handling, carriage or storage of the Product by the Purchaser or any third party (other than Advantest or its agents);

   (c) use of the Product under operating conditions or environments different than those specified in the Operation Manual or recommended by Advantest, including, without limitation, (i) instances where the Product has been subjected to physical stress or electrical voltage exceeding the permissible range and (ii) instances where the corrosion of electrical circuits or other deterioration was accelerated by exposure to corrosive gases or dusty environments;

   (d) use of the Product in connection with software, interfaces, products or parts other than software, interfaces, products or parts supplied or recommended by Advantest;

   (e) incorporation in the Product of any parts or components (i) provided by Purchaser or (ii) provided by a third party at the request or direction of Purchaser or due to specifications or designs supplied by Purchaser (including, without limitation, any degradation in performance of such parts or components);

   (f) Advantest's incorporation or use of any specifications or designs supplied by Purchaser;

   (g) the occurrence of an event of force majeure, including, without limitation, fire, explosion, geological change, storm, flood, earthquake, tidal wave, lightning or act of war; or

   (h) any negligent act or omission of the Purchaser or any third party other than Advantest.

5. **EXCEPT TO THE EXTENT EXPRESSLY PROVIDED HEREIN, ADVANTEST HEREBY EXPRESSLY DISCLAIMS, AND THE PURCHASER HEREBY WAIVES, ALL WARRANTIES, WHETHER EXPRESS OR IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, (A) ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE AND (B) ANY WARRANTY OR REPRESENTATION AS TO THE VALIDITY, SCOPE, EFFECTIVENESS OR USEFULNESS OF ANY TECHNOLOGY OR ANY INVENTION.**

6. **THE REMEDY SET FORTH HEREIN SHALL BE THE SOLE AND EXCLUSIVE REMEDY OF THE PURCHASER FOR BREACH OF WARRANTY WITH RESPECT TO THE PRODUCT.**

7. **ADVANTEST WILL NOT HAVE ANY LIABILITY TO THE PURCHASER FOR ANY INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL OR PUNITIVE DAMAGES, INCLUDING, WITHOUT LIMITATION, LOSS OF ANTICIPATED PROFITS OR REVENUES, IN ANY AND ALL CIRCUMSTANCES, EVEN IF ADVANTEST HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES AND WHETHER ARISING OUT OF BREACH OF CONTRACT, WARRANTY, TORT (INCLUDING, WITHOUT LIMITATION, NEGLIGENCE), STRICT LIABILITY, INDEMNITY, CONTRIBUTION OR OTHERWISE. TORT (INCLUDING, WITHOUT LIMITATION, NEGLIGENCE), STRICT LIABILITY, INDEMNITY, CONTRIBUTION OR OTHERWISE.**

8. **OTHER THAN THE REMEDY FOR THE BREACH OF WARRANTY SET FORTH HEREIN, ADVANTEST SHALL NOT BE LIABLE FOR, AND HEREBY DISCLAIMS TO THE FULLEST EXTENT PERMITTED BY LAW ANY LIABILITY FOR, DAMAGES FOR PRODUCT FAILURE OR DEFECT, WHETHER ARISING OUT OF BREACH OF CONTRACT, TORT (INCLUDING, WITHOUT LIMITATION, NEGLEGENCE), STRICT LIABILITY, INDEMNITY, CONTRIBUTION OR OTHERWISE.**

CUSTOMER SERVICE DESCRIPTION

In order to maintain safe and trouble-free operation of the Product and to prevent the incurrence of unnecessary costs and expenses, Advantest recommends a regular preventive maintenance program under its maintenance agreement.

Advantest's maintenance agreement provides the Purchaser on-site and off-site maintenance, parts, maintenance machinery, regular inspections, and telephone support and will last a maximum of ten years from the date the delivery of the Product. For specific details of the services provided under the maintenance agreement, please contact the nearest Advantest office listed at the end of this Operation Manual or Advantest 's sales representatives.

Some of the components and parts of this Product have a limited operating life (such as, electrical and mechanical parts, fan motors, unit power supply, etc.). Accordingly, these components and parts will have to be replaced on a periodic basis. If the operating life of a component or part has expired and such component or part has not been replaced, there is a possibility that the Product will not perform properly. Additionally, if the operating life of a component or part has expired and continued use of such component or part damages the Product, the Product may not be repairable. Please contact the nearest Advantest office listed at the end of this Operation Manual or Advantest's sales representatives to determine the operating life of a specific component or part, as the operating life may vary depending on various factors such as operating condition and usage environment.

SALES & SUPPORT OFFICES

Advantest Korea Co., Ltd.
22BF, Kyobo KangNam Tower,
1303-22, Seocho-Dong, Seocho-Ku, Seoul #137-070, Korea
Phone: +82-2-532-7071
Fax: +82-2-532-7132

Advantest (Suzhou) Co., Ltd.
Shanghai Branch Office:
Bldg. 6D, NO.1188 Gumei Road, Shanghai, China 201102 P.R.C.
Phone: +86-21-6485-2725
Fax: +86-21-6485-2726

Shanghai Branch Office:
406/F, Ying Building, Quantum Plaza, No. 23 Zhi Chun Road,
Hai Dian District, Beijing,
China 100083
Phone: +86-10-8235-3377
Fax: +86-10-8235-6717

Advantest (Singapore) Pte. Ltd.
438A Alexandra Road, #08-03/06
Alexandra Technopark Singapore 119967
Phone: +65-6274-3100
Fax: +65-6274-4055

Advantest America, Inc.
3201 Scott Boulevard, Suite, Santa Clara, CA 95054, U.S.A
Phone: +1-408-988-7700
Fax: +1-408-987-0691

ROHDE & SCHWARZ Europe GmbH
Mühldorfstraße 15 D-81671 München, Germany
(P.O.B. 80 14 60 D-81614 München, Germany)
Phone: +49-89-4129-13711
Fax: +49-89-4129-13723

ADVANTEST®

http://www.advantest.co.jp

ADVANTEST CORPORATION
Shin-Marunouchi Center Building, 1-6-2 Marunouchi, Chiyoda-ku, Tokyo 100-0005, Japan
Phone: +81-3-3214-7500