ADVANTEST®

ADVANTEST CORPORATION

# U3641 Series OPT15 Controller Function Operation Manual

MANUAL NUMBER FOE-8311268A01

*Applicable models*
*U3641*
*U3641N*

First printing December 1, 1996
Printed in Japan

# Safety Summary

To ensure thorough understanding of all functions and to ensure efficient use of this instrument, please read the manual carefully before using. Note that Advantest bears absolutely no responsibility for the result of operations caused due to incorrect or inappropriate use of this instrument.

If the equipment is used in a manner not specified by Advantest, the protection provided by the equipment may be impaired.

## • Warning Labels

Warning labels are applied to Advantest products in locations where specific dangers exist. Pay careful attention to these labels during handling. Do not remove or tear these labels. If you have any questions regarding warning labels, please ask your nearest Advantest dealer. Our address and phone number are listed at the end of this manual.

Symbols of those warning labels are shown below together with their meaning.

**DANGER**: Indicates an imminently hazardous situation which will result in death or serious personal injury.

**WARNING**: Indicates a potentially hazardous situation which will result in death or serious personal injury.

**CAUTION**: Indicates a potentially hazardous situation which will result in personal injury or a damage to property including the product.

## • Basic Precautions

Please observe the following precautions to prevent fire, burn, electric shock, and personal injury.

- Use a power cable rated for the voltage in question. Be sure however to use a power cable conforming to safety standards of your nation when using a product overseas.
- When inserting the plug into the electrical outlet, first turn the power switch OFF and then insert the plug as far as it will go.
- When removing the plug from the electrical outlet, first turn the power switch OFF and then pull it out by gripping the plug. Do not pull on the power cable itself. Make sure your hands are dry at this time.
- Before turning on the power, be sure to check that the supply voltage matches the voltage requirements of the instrument.
- Connect the power cable to a power outlet that is connected to a protected ground terminal. Grounding will be defeated if you use an extension cord which does not include a protected ground terminal.
- Be sure to use fuses rated for the voltage in question.
- Do not use this instrument with the case open.
- Do not place anything on the product and do not apply excessive pressure to the product. Also, do not place flower pots or other containers containing liquid such as chemicals near this

product.

- When the product has ventilation outlets, do not stick or drop metal or easily flammable objects into the ventilation outlets.
- When using the product on a cart, fix it with belts to avoid its drop.
- When connecting the product to peripheral equipment, turn the power off.

- **Caution Symbols Used Within this Manual**

Symbols indicating items requiring caution which are used in this manual are shown below together with their meaning.

**DANGER**: Indicates an item where there is a danger of serious personal injury (death or serious injury).

**WARNING**: Indicates an item relating to personal safety or health.

**CAUTION**: Indicates an item relating to possible damage to the product or instrument or relating to a restriction on operation.

- **Safety Marks on the Product**

The following safety marks can be found on Advantest products.

⚠ : ATTENTION - Refer to manual.

⏚ : Protective ground (earth) terminal.

ϟ : DANGER - High voltage.

⚠ : CAUTION - Risk of electric shock.

- **Replacing Parts with Limited Life**

The following parts used in the instrument are main parts with limited life.
Replace the parts listed below before their expected lifespan has expired to maintain the performance and function of the instrument.
Note that the estimated lifespan for the parts listed below may be shortened by factors such as the environment where the instrument is stored or used, and how often the instrument is used.
The parts inside are not user-replaceable. For a part replacement, please contact the Advantest sales office for servicing.

Each product may use parts with limited life.
For more information, refer to the section in this document where the parts with limited life are described.

Main Parts with Limited Life

| Part name | Life |
|---|---|
| Unit power supply | 5 years |
| Fan motor | 5 years |
| Electrolytic capacitor | 5 years |
| LCD display | 6 years |
| LCD backlight | 2.5 years |
| Floppy disk drive | 5 years |
| Memory backup battery | 5 years |

- **Hard Disk Mounted Products**

The operational warnings are listed below.

- Do not move, shock and vibrate the product while the power is turned on.
  Reading or writing data in the hard disk unit is performed with the memory disk turning at a high speed. It is a very delicate process.
- Store and operate the products under the following environmental conditions.
  An area with no sudden temperature changes.
  An area away from shock or vibrations.
  An area free from moisture, dirt, or dust.
  An area away from magnets or an instrument which generates a magnetic field.
- Make back-ups of important data.
  The data stored in the disk may become damaged if the product is mishandled. The hard disc has a limited life span which depends on the operational conditions. Note that there is no guarantee for any loss of data.

- **Precautions when Disposing of this Instrument**

When disposing of harmful substances, be sure dispose of them properly with abiding by the state-provided law.

| | |
|---|---|
| Harmful substances: | (1) PCB (polycarbon biphenyl)<br>(2) Mercury<br>(3) Ni-Cd (nickel cadmium)<br>(4) Other<br>Items possessing cyan, organic phosphorous and hexadic chromium and items which may leak cadmium or arsenic (excluding lead in solder). |
| Example: | fluorescent tubes, batteries |

# Environmental Conditions

This instrument should be only be used in an area which satisfies the following conditions:

- An area free from corrosive gas
- An area away from direct sunlight
- A dust-free area
- An area free from vibrations
- Altitude of up to 2000 m

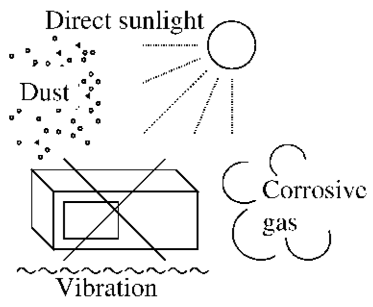
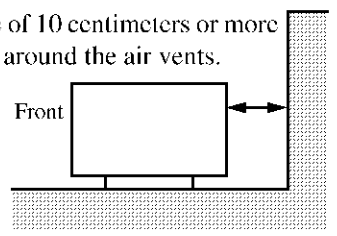


**Figure-1 Environmental Conditions**

- Operating position

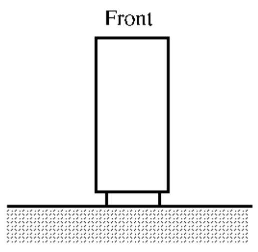
A clear space of 10 centimeters or more must be kept around the air vents.



The instrument must be used in a horizontal position.
A cooling fan, which prevents the internal temperature from rising, is equipped with the instrument.
The air vents on the case must be unblocked.

**Figure-2 Operating Position**

- Storage position



This instrument should be stored in a horizontal position.
When placed in a vertical (upright) position for storage or transportation, ensure the instrument is stable and secure.

-Ensure the instrument is stable.
-Pay special attention not to fall.

**Figure-3 Storage Position**

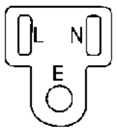
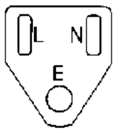
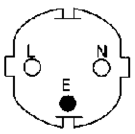
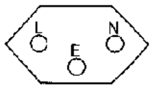
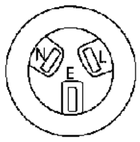
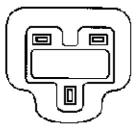
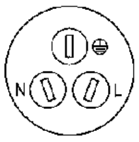
- The classification of the transient over-voltage, which exists typically in the main power supply, and the pollution degree is defined by IEC61010-1 and described below.

  Impulse withstand voltage (over-voltage) category II defined by IEC60364-4-443

  Pollution Degree 2

# Types of Power Cable

Replace any references to the power cable type, according to the following table, with the appropriate power cable type for your country.

| Plug configuration | Standards | Rating, color and length | Model number (Option number) |
|---|---|---|---|
|  | PSE: Japan<br><br>Electrical Appliance and Material Safety Law | 125 V at 7 A<br>Black<br>2 m (6 ft) | Straight: A01402<br><br>Angled: A01412 |
|  | UL: United States of America<br><br>CSA: Canada | 125 V at 7 A<br>Black<br>2 m (6 ft) | Straight: A01403<br>(Option 95)<br>Angled: A01413 |
|  | CEE: Europe<br>DEMKO: Denmark<br>NEMKO: Norway<br>VDE: Germany<br>KEMA: The Netherlands<br>CEBEC: Belgium<br>OVE: Austria<br>FIMKO: Finland<br>SEMKO: Sweden | 250 V at 6 A<br>Gray<br>2 m (6 ft) | Straight: A01404<br>(Option 96)<br>Angled: A01414 |
|  | SEV: Switzerland | 250 V at 6 A<br>Gray<br>2 m (6 ft) | Straight: A01405<br>(Option 97)<br>Angled: A01415 |
|  | SAA: Australia, New Zealand | 250 V at 6 A<br>Gray<br>2 m (6 ft) | Straight: A01406<br>(Option 98)<br>Angled: --------- |
|  | BS: United Kingdom | 250 V at 6 A<br>Black<br>2 m (6 ft) | Straight: A01407<br>(Option 99)<br>Angled: A01417 |
|  | CCC:China | 250 V at 10 A<br>Black<br>2 m (6 ft) | Straight: A114009<br>(Option 94)<br>Angled: A114109 |

# PREFACE

This manual consists of the following contents:

| | |
|---|---|
| PART I | SYSTEM CONTROLLER |
| PART II | ate EDITOR |

# Part I

# SYSTEM CONTROLLER

The contained BASIC in the spectrum analyzer has the general BASIC commands, GPIB control commands and dedicated built in functions and allows easy construction of the GPIB system.

# TABLE OF CONTENTS

# LIST OF ILLUSTRATIONS

# LIST OF TABLES

# 1 BASIC OPERATION

Loading the BASIC Program , execution and existing can be carried out by the panel operation of the spectrum analyzer. Moreover, external terminal (VG-920 or equivalent) is connected then the editing of BASIC program can be carried out.

## 1.1 Panel Operation

### 1.1.1 Load and Execution of the Program

Upon loading of the BASIC program by the panel operation, it runs automatically.

Loading procedure of the BASIC program.

(1) Insert the memory card saved BASIC program to be run to the memory card drive of the spectrum analyzer.

(2) Press RCL key of the panel.

(3) Move the cursor to the program to be loaded.

(4) The program is run automatically by Pressing RECALL EXECUTE soft key.

### 1.1.2 Menu of the BASIC Control Function

(1) Following soft menus are displayed by pressing the CONT key on the panel.

$\boxed{\text{CONT}}$

$\downarrow$

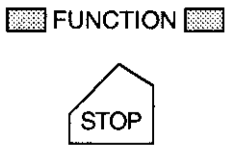
| | | |
|---|---|---|
| 1 | RUN | Executes BASIC program. |
| 2 | CONTINUE | Re-executions the program from paused line. |
| 3 | | |
| 4 | AUTO EXE CARD/MEM | Selects either execution AUTOST .BAS program in the memory card or the program in the internal memory. |
| 5 | MASTER /SLAVE | Selects either master (controller) or slave (under controller) for the spectrum analyzer. *1) |
| 6 | CONT OFF | Exits from BASIC control function to the measurement mode screen. |

*1) When MASTER mode is selected, control of the spectrum analyzer and external connected GPIB devices can be controlled by the contained BASIC program in the spectrum analyzer.

When SLAVE mode is selected, control of the spectrum analyzer and, sending and receiving the data can be carried out between host computer and the spectrum analyzer.

(2) Upon the stop key of the function panel, execution of the BASIC program can be paused.

Pressing this key pauses at all time during BASIC program's execution.
Press CONTINUE key of the soft menu described on the previous page for re-running.



### 1.1.3 Data Input Key

For requirement of the data during program's execution, input the data using by a numeric keypad (0 to 9), a decimal keypad (.), minus keypad (-(BS)) and finally press the unit keypad then data input is completed.

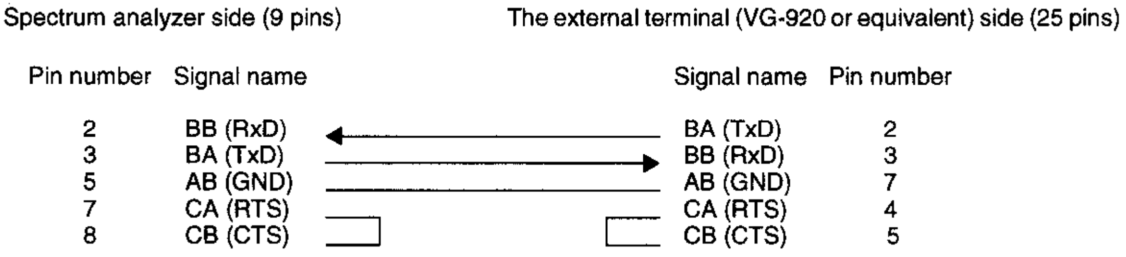
Furthermore, input data can be deleted one character by one character using by -(BS) keypad.

### 1.1.4 Function Key

The soft key (1 to 6) is assigned for the function key after program started.

## 1.2 Interconnection with the External Terminal

For editing BASIC program, it is necessary to interconnect between the external terminal (VG-920 or equivalent) and the spectrum analyzer. The external terminal (VG-920 or equivalent) is connected to RS-232 connector on the rear panel of the spectrum analyzer. Interface connection is as follows.



| Spectrum analyzer side (9 pins) | | | The external terminal (VG-920 or equivalent) side (25 pins) | |
|---|---|---|---|---|
| Pin number | Signal name | | Signal name | Pin number |
| 2 | BB (RxD) | ← | BA (TxD) | 2 |
| 3 | BA (TxD) | → | BB (RxD) | 3 |
| 5 | AB (GND) | — | AB (GND) | 7 |
| 7 | CA (RTS) | ┐ | ┌ CA (RTS) | 4 |
| 8 | CB (CTS) | ┘ | └ CB (CTS) | 5 |

Starting the edit function

(1) Interconnect between the external terminal (VG-920 or equivalent) and the spectrum analyzer using by RS-232 cable.

(2) Turn on the power both of the external terminal (VG-920 or equivalent) and the spectrum analyzer.

(3) Carry out the operation of the panel keys on the spectrum analyzer in the following procedure.

$\boxed{\text{SHIFT}}$ + $\boxed{2}$ → $\boxed{6}$ NEXT → $\boxed{1}$ SETUP RS232 → $\boxed{6}$ NEXT
→ $\boxed{1}$ DEV SW MAIN/CTL

Select CTL in DEV SW then > is displayed on the screen of the external terminal.

***Note:*** ***MAIN is used for the remote control for the spectrum analyzer using by the GPIB command from the external terminal through RS-232 cable.***

(4) Input EDIT from the external terminal then edit screen is appeared and creating for the BASIC program becomes available.

## 1.3 Memory Card

In the memory card, the set condition, the BASIC program and the data files from inside of the BASIC program and so on are able to be recorded and regenerated.
The following memory cards are able to be used in this analyzer.

### 1.3.1 Usable Memory Card

- Adapted to JEIDA Ver.4.0 or higher ( 68 pin two piece connector).

  TYPE1

- Only the following Memory types are permitted.

| | | |
|---|---|---|
| Common memory | : | SRAM |
| Attribute memory | : | Any one of the SRAM, EPROM, MASKROM, EEPROM, OTPROM or flash memory is all right. |

- Formatting

  MS-DOS format.
  Corresponding to the various kinds of memory size.

***CUATION!***
***Only the memory cards that are adapted to the PC card guide line Ver 4.1 of the Japan Electronic Industry Development Association (JEIDA) or to PCMCIA Release 2.0 or higher that is the United States of America standards are permitted. Use the memory cards only after making sure that those are adapted to the standards as above. For details, refer to "U3641 Series OPERATION MANUAL".***

## 1.3.2 Memory Card Specifications

**Table 1-1 Memory Card Specifications**

| Specifications | Memory Card |
|---|---|
| Connector | 68 Pin two Piece Connector |
| | In accordance with JEIDA Ver. 4.1 |
| Dimensions | 54 (Width) × 86 (Length) × 3.3 (Thickness) mm |
| Operating Environment | No condensation<br>Operating environment : 0 to 55°C<br>Storage environment : -20 to 60°C<br>Relative humidity : Less than 95% |
| Write protect | Switching ON and OFF by the switch.<br>It is impossible to write on the side of ON. |

## 1.3.3 Note on Handling the Memory Card

- Keep dust out from the hole of the connector.
  It causes the defective contact or the damage of the connector.
- Do not touch the connector with a material like a metal needle and so on.
  It may causes the static electricity destruction.
- Do not bend it or give a great shock on it.
- Keep it away from water.

## 1.3.4 Insertion and Ejection of Memory Card

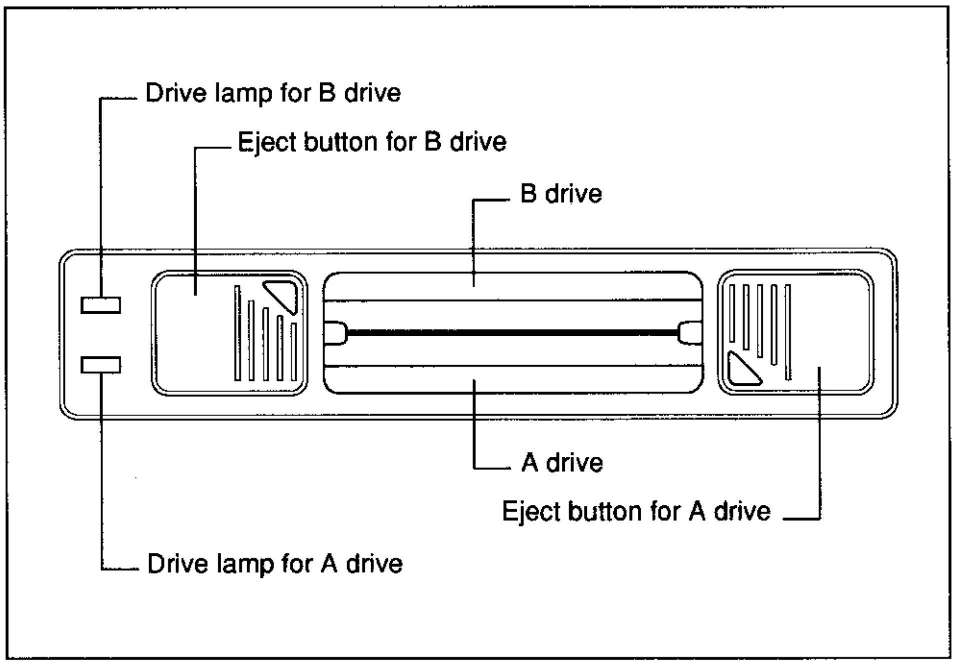


**Figure 1-1 Drive Slot for Memory Card**

The drive slots for the memory card are on the upper of the equipment.

① Insert the memory card with the printed side front.

② The drive lamp is turned on yellow when the memory card is inserted.

③ When the memory card is ejected, press the eject button only after making sure that the drive lamp is turned on yellow.

***CAUTION!***
***The drive lamp is turned on red when the card is given access. Do not press the eject button to eject the memory card when the drive lamp is turned on red.***
***In the case that the memory card is ejected when the drive lamp is turned on red, the data in the memory card is not guaranteed.***

# 2 BASIC GPIB CONTROLLER

## 2.1 Outline

The BASIC language, Controller function, covers the GPIB control commands as well as the general-purpose BASIC commands. It allows small-size GPIB systems to be constructed. Moreover, measurement-dedicated built-in functions enable simple and high-speed measurement.

***Note:*** ***Controller function controllers' GPIB adresses are set by CONTROL command BASIC language.***

## 2.2 BASIC Programming

### 2.2.1 Program Configuration

Statements are the smallest unit that the BASIC manipulates. A group of statements configure a BASIC program.
Statements are roughly divided into the control statements and execution statements (commands).

- Statements
  - Control statements
    - Declaration statements (DIM, INTEGER, etc.)
    - Loop and branch statements (FOR, IF, etc.)
    - Interruption statements (ON-, OFF-, etc.)
  - Execution statements (commands)
    - Program execution control commands
    - Program edition commands
    - I/O commands
    - GPIB commands
    - File control commands

Each statement consists of the keyword and expressions. The structure of each statement is determined by the syntax rule.
In BASIC, keywords have been already assigned specific meanings and uses.
The table in the next paragraph shows the keywords in BASIC. Same name as variables cannot be used.

### 2.2.2 Keywards

[List of keywords]

| | | | | | |
|---|---|---|---|---|---|
| AND | AS | ASCII | BAND | BINARY | BNOT |
| BOR | BREAK | BUZZER | BXOR | CALL | CASE |
| CAT | CLEAR | CLOSE | CLS | CMD | CONT |
| CONTINUE | CONTROL | CSR | CURSOR | DATA | DELIMITER |
| DIM | DISABLE | ELSE | ENABLE | END | ENT |
| ENTER | ERROR | FOR | GLIST | GOSUB | GOTO |
| GPRINT | IF | INIT | INITIALIZE | INP | INPUT |
| INTEGER | INTERFACE | INTR | ISRQ | KEY | LABEL |
| LIST | LISTEN | LLIST | LOCAL | LOCKOUT | LPRINT |
| NEXT | NOT | OFF | ON | OPEN | OR |
| OUT | OUTPUT | PAUSE | PRF | PRINT | PRINTER |
| PRINTF | PURGE | READ | REM | REMOTE | RENAME |
| REQUEST | RESTORE | RETURN | RUN | SCRATCH | SCREEN |
| SELECT | SEND | SPOOL | SPRINTF | SRQ | STEP |
| STOP | TALK | TEXT | THEN | TO | TRIGGER |
| UNL | UNT | USE | USING | WAIT | XOR |

## 2.2.3 Short Name

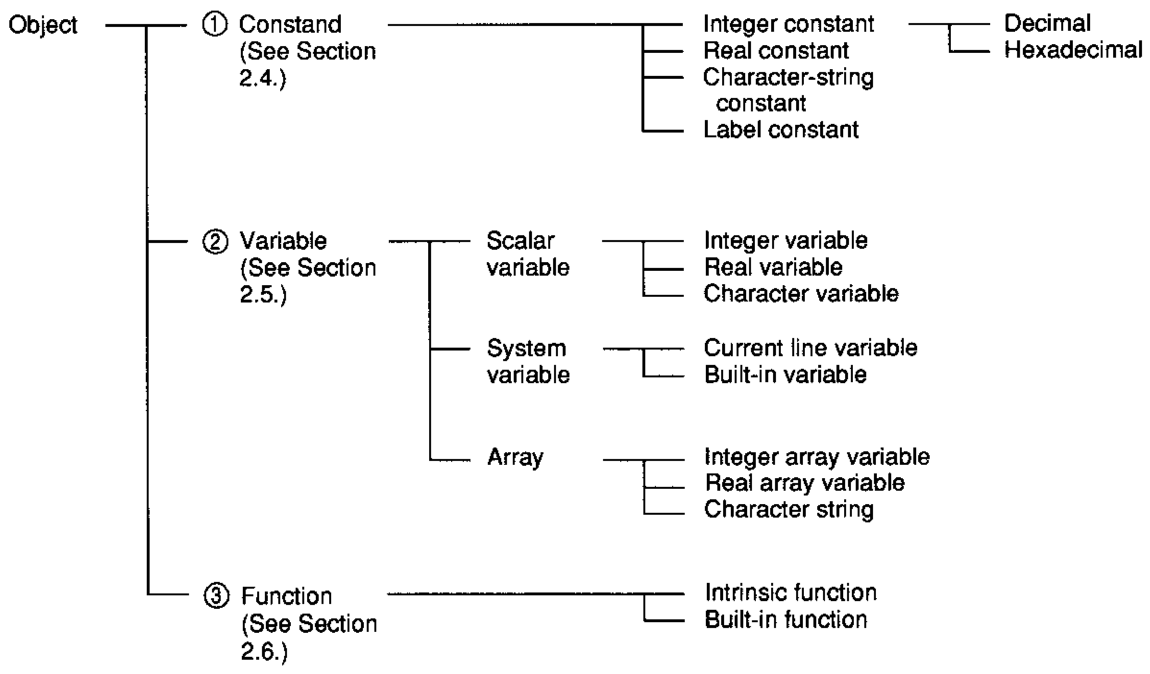
Short names can be used to input those keywords that may be used frequently and have a long name. They are also keywords and cannot be used as variables.

[Relations between full names and short names]

| Full name | Short name |
|---|---|
| CURSOR | CSR |
| ENTER | ENT |
| INITIALIZE | INIT |
| INPUT | INP |
| OUTPUT | OUT |
| PRINTF | PRF |
| USING | USE |

## 2.3 Objects

The objects that BASIC manipulates include variables, constants, and functions.
Each object should has a data type. Data types contain integer, real, and character string.



## 2.4 Constant

### 2.4.1 Integer Constant

A sequence of digits that does not contain decimal points is taken to be an integer constant. Both decimal and hexadecimal numbers are used as integer constants. Since an integer constant is represented with 4 bytes inside, BASIC can take values between -2, 147, 483, 648 and +2, 147, 483, 647.
A sequence of digits preceded by 0x is taken to be a hexadecimal number.

Decimal numbers

```
Example:  A = 123
          PRINT 456
```

Hexadecimal numbers

```
Example:  A = 0 × 10AB
          PRINT 0 × FFFF
```

### 2.4.2 Real Constant

A sequence of digits consisting of a decimal number or represented with a floating point is taken to be a real constant.
Since real constants are represented with 8 bytes inside, BASIC can take values between approximately -1E+308 to 1E+308.
The accuracy is of 15 digits.

```
Example:  A = 123.0
          B = 1.23456789
          C = 1.23E6
```

## 2.4.3 Character Constant

A sequence of within 255 characters enclosed in double quotations is called a character constant. A character constant can contain either a null-character string " "or up to 255 characters.
To represent those characters that are not assigned keyboard, escape sequences may be used. Each escape sequence consists of a character preceded by a backslash (\). Escape sequences may also be used to represent the ASCII control characters.

***Note:*** ***A backslash should be used with an octal number or the following escape sequences:***

| *Escape sequence* | *Octal* | *Decimal* | |
|---|---|---|---|
| *\b* | *010* | *8* | *Backspace* |
| *\t* | *011* | *9* | *Horizontal tab* |
| *\n* | *012* | *10* | *Line feed* |
| *\v* | *013* | *11* | *Vertical tab* |
| *\f* | *014* | *12* | *Formfeed* |
| *\r* | *015* | *13* | *Carriage return* |

***Example:***

```
A$ = "ABCDE"
CR$ = "\r"
NL$ = "\n"
B$ = " "
PRINT "ABCD\014"
PRINT "AB\007"
PRINT "ABC\n"
```

## 2.4.4 Label Constant

A label constant takes the part of a statement number. A label name should be preceded by an asterisk (*) when it is declared.
Those characters used for a label name are the same as those for a variable.
However, a numeric value cannot be entered to a label since it is not a variable. The positions where the labels can be specified are limited by the syntax rule. The labels should be specified in the positions indicated as <label> described in the section 3.3 "Description on commands and statements".

## 2.5 Variables

A variable name consists of up to 20 alphanumeric characters, beginning with an alphabetic character.

[Alphanumeric]

```
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
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,
1, 2, 3, 4, 5, 6, 7, 8, 9, 0
_ (Underscore)
```

***Note:*** ***The following names can not be used.***
- ***Keyword name (See 2.2.2.)***
- ***Function name (See 2.6)***
- ***APPEND, DEL, LOAD, SAVE, TIME, POKE, EDIT***

A variable name terminated by a $ is a character variable. A variable name terminated by a pair of parentheses ( ) is an array variable.
A variable is taken to be a real variable if it is not declared as an INTEGER.

| | | |
|---|---|---|
| Example: | VAL | : Real variable |
| | STRG$ | : Character variable |
| | ARRY1 (4) | : Real array variable |
| | INTEGER code 2 | : Integer variable |
| | INTEGER wk (7) | : Integer array variable |

### 2.5.1 Scalar Variables

- Integer variable
- Real variable
- Character variable

Any numeric variables of the above will be initialized to zero when a BASIC program that includes those variable starts.
Therefore, if a variable should be initialized to a specific value, it must be entered the value explicitly in the program.
The range of values to be entered to an integer variable is the same as that of integer constants used in BASIC (from -2, 147, 483, 648 to +2, 147, 483, 647).
The range of values to be assigned to a real variable is the same as that of real constants used in BASIC (between approximately -1E+308 and 1E+308, and the accuracy is of 15 digits).

Character variables, as well as character strings, have attributes of the length.The length should be declared with a DIM statement.

Example: DIM STRG$ [100]: Declares a variable where the length is 100 characters.

If the length of a variable is not declared, a space for 18 characters will be assigned to the variable.

### 2.5.2 System Variables

- Built-in variable

  When a BASIC program starts, built-in variables are automatically registered and initialized to the specified values. The value of a built-in function can be changed if it is entered another value. A variable may be returned to the initial value if it is assigned the initial value or the program is initialized with **compile & run** .

```
PI   :3.141592.....
EXP  :2.718281.....
```

- Error number variable

  The error number variable holds the error number in BASIC. It is initialized to zero when a BASIC program starts. If an error occurs, the value of the error is assigned to the variable.

  Example: PRINT ERRN

  The error number has the following structure inside of the program:

  Error class * 256 + Error message number

  Error class
  1 : Data I/O errors
  2 : Data operation errors
  3 : Built-in function errors
  4 : BASIC syntax errors

### 2.5.3 Arrays

Arrays should be declared with DIM or INTEGER statements. If two or more subscripts separated by comma are specified with an array variable name, the array variable will have a number of dimensions according to the number of the subscripts. (The maximum number of the dimensions is 10 but limited by the memory capacity.)

- Numeric arrays

  If an array that has not been declared is referred to, the array size, the number of elements, is assumed to be 10. The array could have been declared as in the following examples:

  Subscripts always start with 1.
  The subscript specified in DIM declaration will be the maximum number of the elements. The range which can declare DIM is up to 32767.

```
        DIM         AB (10)
        INTEGER     CD (10)
```

```
Example: DIM         RL (30)       : Declares a real array variable.
         INTEGER     IT (10, 20)   : Declares a integer array variable (two dimensional).
```

- Character string

  The character-string array, and the number of characters to be stored in a character-string can be declared.

```
Example: DIM A$ (100)       : Declares a character-string array.
         DIM B$ [100]       : Up to 100 characters can be entered.
         DIM C$ [50]        : Up to 50 characters can be entered.
         DIM D$ (100) [50]  : Declares a character-string array and available character
                              numbers of up to 50.
```

***Note:*** ***The number of characters to be stored is up to 128.***
***When the number of characters is not declared, the number of characters is 18.***

## 2.6 Function

### 2.6.1 Intrinsic Functions

| Intrinsic Function | Explanation |
|---|---|
| ERRM$<br>(Error nuber) | Returns the error message specified with the parameter.<br>If 0 is passed as parameter, the function returns the error message that has been output last.<br>The error number has the following structure inside of the program.<br><br>Error class * 256 + Error message number<br><br>However, even if a number including an error class is specified, only the error message number will be referred to inside of the program.<br>Therefore, ERRN can be set as an error number, as follows:<br><br>Example: PRINT ERRM$ (ERRN):<br>Displays the error message that has been displayed last.<br>When a parameter except for 0 or the last error number, ERRN is specified, xxx is displayed in the part of an arbitrary character string and yy is displayed in the part of an arbitrary numerical value.<br><br>Example: PRINT ERRM$ (91) →\| Invalid valie in xxx<br>PRINT ERRM$ (90) →\| FOR NEXT yy error(s) appeared. |
| NUM<br>(Character-string expression) | Returns the ASCII code of the first character of the character-string expression.<br><br>Example: NUM ("ABC") → 65<br>A$ = "XYZ"<br>NUM (A$) → 88 |
| CHR$<br>(Arithmetic expression) | Returns the character-string expression determined by the value of the arithmetic expression.<br><br>Example: CHR$ (65) → "A"<br>A = 88<br>CHR$ (A) → "X" |
| LEN<br>(Character-string expression) | Returns the length of the character-string expression.<br><br>Example: LEN ("ADVANTEST") →\| 9<br>A$ = "CORP."<br>LEN (A$) →\| 5 |

(cont'd)

| Intrinsic Function | Explanation |
|---|---|
| POS<br>(Character-string expression 1, character-string expression 2) | Returns the position of the first set of character-string expression 2 in character-string expression 1.<br><br>Example: A$ = "AN"<br>POS ("ADVANTEST", A$) → 4 |
| ABS<br>(Arithmetic expression) | Returns the absolute value of the arithmetic expression.<br><br>Example: ABS (-1.2) → 1.2 |
| ATN<br>(Arithmetic expression) | Returns the arc tangent of the arithmetic expression.<br>(The arithmetic expression should be passed in radian.)<br><br>Example: ATN (PI) → 1.26262... |
| COS<br>(Arithmetic expression) | Returns the consine of the arithmetic expression.<br>(The arithmetic expression should be passed in radian.)<br><br>Example: COS (PI) → -1.0 |
| FRE<br>(0) | Returns the size (bytes) of the free area in the memory for BASIC.<br><br>Example: FRE (0) → 502718<br>When the system is turned on, this function is executed as follows in the BASIC mode: PRINT FRE (0).) |
| LOG<br>(Arithmetic expression) | Returns the natural logarithm (base e logarithm) of the arithmetic expression.<br>(The arithmetic expression should be passed in radian.)<br><br>Example: LOG (EXP) → 1.0 |
| SIN<br>(Arithmetic expression) | Returns the sine of the arithmetic expression.<br>(The arithmetic expression should be passed in radian.)<br><br>Example: SIN (PI) → 0.0 |
| SPOLL<br>(Arithmetic expression) | Performs serial polling to the GPIB device and then returns the status byte.<br>(The arithmetic expression should be in range between 0 and 31.)<br><br>Example: SPOLL (2) : Performs serial polling to address 2 of the external GPIB device connected.<br>SPOLL (31) : Performs serial polling to the measurement section of the main device. |

(cont'd)

| Intrinsic Function | Explanation |
|---|---|
| SQR<br>(Arithmetic expression) | Returns the square root of the arithmetic expression.<br><br>Example: SQR (2) → 1.41421356... |
| TAN<br>(Arithmetic expression) | Returns the tangent of the arithmetic expression.<br>(The arithmetic expression should be passed in radian.)<br><br>Example: TAN (PI) → 1.0 |
| RAND<br>(0) | Generates random numbers from 0 to 1.<br><br>Example: RAND (0) → .375 |
| LGT<br>(Arithmetic expression) | Returns common logarithm (logarithm based on radix of 10) for arithmetic expression.<br><br>Example: LGT (19) → 1.0 |

### 2.6.2 Built-in Function

| Item | Built-in Function |
|---|---|
| Obtains the frequency or point. | F = FREQ (P)<br>F = DFREQ (P1, P2)<br>P = POINT (F)<br>P = DPOINT (F1, F2) |
| Obtains the level or point. | L = LEVEL (T)<br>L = DLEVEL (T1, T2)<br>T = LVPOINT (L)<br>T = LVDPOINT (L1, L2)<br>L = VALUE (P, M)<br>L = DVALUE (P1, P2, M)<br>L = CVALUE (F, M)<br>L = DCVALUE (F1, F2, M) |
| Obtains the maximum and minimum. | F = FMAX (P1, P2, M)<br>F = FMIN (P1, P2, M)<br>P = PMAN (P1, P2, M)<br>P = PMIN (P1, P2, M)<br>L = MAX (P1, P2, M)<br>L = MIN (p1, P2, M) |

(cont'd)

| Item | Built-in Function |
|---|---|
| Obtains the band width. | F = BND (P, X, M)<br>F = BNDL (P, X, M)<br>F = BNDH (P, X, M)<br>F = CBND (F, X, M)<br>F = CBNDL (F, X, M)<br>F = CBNDH (F, X, M) |
| Obtains the maximum and minimum (ripple). | N = NRPLH (P1, P2, Dx, Dy, M)<br>N = NRPLL (P1, P2, Dx, Dy, M)<br>P = PRPLHN (N, M)<br>P = PRPLLN (N, M)<br>F = FRPLHN (N, M)<br>F = FRPLLN (N, M)<br>L = VRPLHN (N, M)<br>L = VRPLLN (N, M)<br>L = RPL1 (P1, P2, Dx, Dy, M) |
| Checks the upper and lower limits. | C = LMTMD1 (Dd, S, Ds)<br>C = LMTMD2 (P, S, Ds, M)<br>C = LMTUL1 (Dd, Up, Lo)<br>C = LMTUL2 (P, Up, Lo, M) |
| Obtains the total power. | W = POWER (P1, P2, M) |
| Input and output the traced data. | T = RTRACE (P, M)<br>T = TRACE (P, M), TRACE (P, M) = T<br>WTRACE (T, P, M)<br>***Note: This function returns no value.*** |
| Graphics functions. | GADRS (Mo, X, Y)<br>GFLRECT (D, X1, Y1, X2, Y2)<br>GLINE (D, X1, Y1, X2, Y2)<br>GPOINT (D, X, Y)<br>GRECT (D, X1, Y1, X2, Y2) |

## 2.7 Operation Expression

Objects are manipulated by operators. An expression consists of a combination of operators and objects.

| | | | |
|---|---|---|---|
| Operators | (1) | Assignment operator | (See subsection 2.7.1.) |
| | (2) | Unary arithmetic operator | (See subsection 2.7.2.) |
| | (3) | Binary arithmetic operator | (See subsection 2.7.3.) |
| | (4) | Logical operator | (See subsection 2.7.4.) |
| | (5) | Relational operation | (See subsection 2.7.5.) |
| | (6) | Substring operator | (See subsection 2.7.6.) |
| | (7) | Bitwise operator | (See subsection 2.7.7.) |

### 2.7.1 Assignment Operator

An assignment expression has its value.

```
Example:  A = 123
          PRINT A
          PRINT B$ = “ADVANTEST”
          PRINT (A = 2) + A

     [Result]

          123.0
          ADVANTEST
          4.0
```

Assignment operators are summarized below.

| Assignment operator | Example | Meaning |
|---|---|---|
| = | A = 123 | Normal assignment |
| += | A += 5 | Equivalent to A = A + 5. |
| -= | A -= 5 | Equivalent to A = A - 5. |
| *= | A *= 5 | Equivalent to A = A * 5. |
| /= | A /= 5 | Equivalent to A = A / 5. |
| %= | A %= 5 | Equivalent to A = A % 5. |
| => | A$ => “ABCD” | Enter a character string right-justified. |
| =< | A$ =< “ABCD” | Enter a character string left-justified. |

***Note:*** ***% is a modulus operator, which produces a remainder.***

Example:
```
DIM S$ [15]
A = 5      : PRINT A
A += 10    : PRINT A
A -= 3     : PRINT A
A *= A     : PRINT A
A /= 2     : PRINT A
A %= 5     : PRINT A
PRINT "123456789012345"
S$ => "TEST"     : PRINT S$
S$ =< "TEST"     : PRINT S$
```

[Result]

```
5.0
15.0
12.0
144.0
72.0
2.0
123456789012345
           TEST
TEST
```

## 2.7.2 Unitary Arithmetic Operator

| | |
|---|---|
| - | Minus sign |
| + | Plus sign |
| ++ | Prefix/postfix increment: Add 1 to a variable.<br><br>• Prefix increment<br>Example: A = 2 Assign 2 to A.<br>B = ++A Add 1 to A before entering A to B.<br><br>• Postfix increment<br>Example: A = 2 Assign 2 to A.<br>B = A++ Add 1 to A after entering A to B. |
| -- | Prefix/postfix decrement: Subtract 1 from a variable.<br><br>• Prefix decrement<br>Example: A = 2 Enter 2 to A.<br>B = --A Subtract 1 from A before entering A to B.<br><br>• Postfix decrement<br>Example: A = 2 Assign 2 to A.<br>B = A-- Subtract 1 from A after entering A to B. |

When an increment operator (++) is placed before a variable, 1 is added to the variable before the its value is used. When an increment operator is placed after a variable, 1 is added to the variable after its value has been used.
In case of a decrement operator (--), the same operations take place except that 1 is subracted from a variable before or after the assignment.

Example:
```
A = 10
A ++                     : Equivalent to A = A + 1.
++A                      : Equivalent to A = A + 1.
A--                      : Equivalent to A = A - 1.
--A                      : Equivalent to A = A - 1.
PRINT "1", A++           : Add 1 to A after displaying the value of A.
PRINT "2", A
B = --A                  : Enter A to B after subtracting 1 from A.
C = A++                  : Enter A to C before addting 1 to A.
PRINT "3", A
PRINT "4", B
PRINT "5", C
B = A--                  : Enter A to B before subtracting 1 from A.
PRINT "6", A
PRINT "7", B
C = -123
D = C + (-23) - (+50)
PRINT "8", C
PRINT "9", D
```

[Result]

```
1       10.0
2       11.0
3       11.0
4       10.0
5       10.0
6       10.0
7       11.0
8       -123.0
9       -196.0
```

### 2.7.3 Binary Arithmetic Operator

| | |
|---|---|
| + | Addition |
| - | Subtraction |
| * | Multiplication |
| / | Division |
| % | Modulus (remainder) |
| ^ | Power |
| & | Connecting character strings |

Example:
```
PRINT 10+2
A = 10
PRINT A-5
PRINT A*A
PRINT 20/A
PRINT A%3
B = 3 ^ 4
PRINT B
S$ = "ABCD"
S1$ = S$& "EFG"
PRINT S1$
```

[Result]

```
12.0
5.0
100.0
2.0
1
81.0
ABCDEFG
```

## 2.7.4 Logical Operator

Logical operators connects two or more relational operators to express compound conditions.

| | | | | |
|---|---|---|---|---|
| NOT | Negation | X | | NOT X |
| | | 0 | | 1 |
| | | 1 | | 0 |
| AND | Conjunction (Logical AND) | X | Y | X AND Y |
| | | 0 | 0 | 0 |
| | | 0 | 1 | 0 |
| | | 1 | 0 | 0 |
| | | 1 | 1 | 1 |
| OR | Disjunction (Logical OR) | X | Y | X OR Y |
| | | 0 | 0 | 0 |
| | | 0 | 1 | 1 |
| | | 1 | 0 | 1 |
| | | 1 | 1 | 1 |
| XOR | Exclusive OR | X | Y | X XOR Y |
| | | 0 | 0 | 0 |
| | | 0 | 1 | 1 |
| | | 1 | 0 | 1 |
| | | 1 | 1 | 0 |

Example: IF NOT A THEN GOTO *MI
If A is zero, a jump to *MI takes place.

IF X < 100 OR 199 < X THEN GOTO *LA
If X is less than 100 or greater than 199, a jump to *LA takes place.

IF 0 <= X AND X <= 100 THEN PRINT X
If X is greater than or equal to 0 and less than or equal to 100, X is printed.

IF A XOR B THEN PRINT A, B
If A is true but B is false or if A is false but B is true, both A and B are printed.

### 2.7.5 Relational Operator

A relational operator is used to compare two numeric values. The result of the comparison will be either true (1) or faluse (0).
It is used to branch the flow of the program in such a statement as a condition judgment statement (IF statement).
In the conditional expression of an IF statement, a logical operation always takes place and an equal sign (=) will always be taken to be a relational operator. Therefore, the conditional expression cannot include any assignment expression.
To express equality out of the conditional expression of IF statement, use "==" and distinguish it from "=" for an assignment operator.

Example: A = (B$ == "ADVAN")
If the character variable B$ is "ADVAN", 1 is assigned to A.

| Symbol | Meaning | Example |
|---|---|---|
| = (or ==) | Equal to | X = Y, X == Y |
| <> | Not equal to | X <> Y |
| < | Less than | x < Y |
| > | Greater than | x > Y |
| <= | Less than or equal to | x <= Y |
| >= | Greater than or equal to | x >= Y |

***Note: "<=" or ">=" cannot be substitute for "=<" or "=>".***

Example:
```
A = 1
B = 2
IF A = 1 AND B > 1 THEN PRINT "A"
IF A <> 1 OR B = 5 THEN PRINT "B"
IF NOT A THEN PRINT "C"
IF A XOR B >= 3 THEN PRINT "D"
IF A == (B-1) THEN PRINT "E"
```

[Result]

```
A
D
E
```

### 2.7.6 Substring Operator

With substring operator, a part of a character-string expression can be referenced.

- Character-string expression (arithmetic expression 1, arithmetic expression 2)

  Pick up characters from the first arithmetic expression character to the second arithmetic expression character of the character string.

```
Example: A$ = "ABCDEFG"
         PRINT A$ [3, 5]
         PRINT "*ADVANTEST*" [2, 6]
         PRINT "*ADVANTEST*" [7, 10]

      [Result]

         CDE
         ADVAN
         TEST
```

- Character-string expression (arithmetic expression 1; arithmetic expression 2)

  Pick up character line of 2-digit arithmetic expression character from the first arithmetic expression character.

```
Example: A$ = "ABCDEFG"
         PRINT A$ [3; 4]
         PRINT "*ADVANTEST*" [7; 4]
         PRINT "*ADVANTEST*" [2; 5]

      [Result]

         CDEF
         TEST
         ADVAN
```

### 2.7.7 Bitwise Operastors

Bitwise operators directly manipulate each of bits that consist of data. They provide logical operations (AND, OR, XOR, etc.) for each bit.
Bit operations may only be applied to integers. The range of bit operations can take place within 16 bits, 0 to 65535. If a minus number is specified, an error occurs. (NO operand in ...)
To use a bit operator with a numeric variable, declare the variable as an integer by using the INTEGER instruction.

| | |
|---|---|
| BNOT | (One's complement)<br>BNOT 0 → 65535<br>BNOT 65535 → 0 |
| BAND | (Logical AND)<br>65535 BAND 255 → 255<br>255 BAND 1024 → 0 |
| BOR | (Logical OR)<br>255 BOR 1024 → 1279<br>1 BOR 2 → 3 |
| BXOR | (Exclusive OR)<br>255 BXOR 128 → 127<br>1 BXOR 3 → 2 |

Example:
```
INTEGER S
*L
S = SPOLL (31)
IF S BAND 4 THEN PRINT "SWEEP END"
GOTO *L
```

## 2.8 Precedence of Operation

Various operators have precedence.
Operators on the same line are evaluated according to the number shown below.

| Precedence | Operator |
|---|---|
| 1 | Expression enclosed in parantheses ( ). |
| 2 | Function |
| 3 | ^ |
| 4 | Sign |
| 5 | ++, -- |
| 6 | *, / |
| 7 | % |
| 8 | +, - |
| 9 | Relational operator |
| 10 | BNOT |
| 11 | BAND |
| 12 | BOR |
| 13 | BXOR |
| 14 | NOT |
| 15 | AND |
| 16 | OR |
| 17 | XOR |

## 2.9 Character-string Operation

In BASIC, operations can take place with character strings.

### 2.9.1 Join Character Strings

Character strings may be joined with "&".

```
Example:  A$ = "ADVANT" & "TEST"
          B$ = A$& "co."
          PRINT A$
          PRINT B$

     [Result]

          ADVANTEST
          ADVANTEST co.
```

### 2.9.2 Compare Character Strings

As well as numbers, characters may be compared with each other with relational operations.

=, <, >, <>, <=, >=

"==" should be used to indicate equality out of IF statements.

```
Example:  PRINT A == B
```

Comparison takes place from the first character serially. If two character strings to compare are of the same length, the character string that has the larger ASCII code will be evaluated to be greater than the other.
If either of the two character strings to compare is shorter, the shorter one will be evalutated to be less than the other.
Note that blank spaces have meanings in character strings.

```
Example:  "AA" = "AA"     →  True
          "AA" < "aa"     →  True
          "AAA" > "AA"    →  True
```

### 2.9.3 Type Conversion

Input may take place from chracter-string expression to numeric variables or from numeric expression to character-string variables, directly.
When a character-string expression include both alphabetic and numeric characters, only the first numeric character string will be assigned to a numeric variable.

Example:
```
A = "123.4"
B = "ABC456.7DEF89"
C$ = 123
D$ = B
PRINT A
PRINT B$
PRINT C$
PRINT D$
```

[Result]

```
123.4
456.7
123
456.7
```

# 3 GRAMMARS AND DESCRIPTION OF COMMANDS AND STATEMENTS

## 3.1 Outline

This chapter provides descriptive expression on the syntax of commands and statements so that it can be understood.

## 3.2 Introduction

### 3.2.1 Structure of Description

| | |
|---|---|
| BUZZER | ---------- Command name (Short name: abbreviation) |
| [Outline] | ---------- Function of the instruction |
| [Format] | ---------- How to describe the instruction (Descriptive expression) |
| [Description] | ---------- Usage and details of the instruction |
| [Example] | ---------- Example of the instruction |
| [Note] | ---------- Note on using the instruction |
| [Program Example] | ---------- Program example where the instruction is used |
| [Result] | ---------- Execution results of the program example |

(1) [Format]

The following symbols are used in descriptive expressions of [Format].

| | |
|---|---|
| < > | : The item enclosed in brackets (< >) should be specified by the user. |
| [ ] | : The item enclosed in square brackets ([ ]) may be omitted. |
| { } | : The item enclosed in braces may be used repeatedly. |
| , | : Two or more parameters may be specified if they are separated by a comma. |
| \| | : A vertical bar means OR. |

Example: <A> | <B> ---------- <A> or <B> is used.

(2) [Description]

The meanings of the terms used in the description are explained below.

Numeric expression:
Indicates any of a numeric constant, numeric variable, or numeric expression.

Character-string expression:
Indicates a character-string expression consisting of character-string constants, character-string variables, character-string function, and substrings.

Device address:
A GPIB address of the device connected to the GPIB.

File descriptor:
It is equivalent to a variable and attached to a data input or output statement.

Label: Label name (including a line number).
A label name consists of alphabetic characters lead by an asterisk (*).

(3) Commands and Statements Classified by Functions

| Function | Command and Statement | Description |
|---|---|---|
| ① Commands | CONT | Resumes the execution of the program after it has stopped. |
| | CONTROL | Sets values for each control. |
| | EDIT | Starts the ate editor function. |
| | LIST | Displays a program list. |
| | LLIST | Displays a program list. (RS-232) |
| | RUN | Executes a program. |
| ② Arithmetic functions | ABS | Produces the absolute value of the given value. |
| | ATN | Produces the arc tangent of the given value. |
| | COS | Produces the cosine of the given value. |
| | LOG | Produces the natural logarithm of the given value. |
| | SIN | Produces the sine of the given value. |
| | SQR | Produces the square root of the given value. |
| | TAN | Produces the tangent of the given value. |
| ③ Bitwise operations | BAND | Produces a bit AND. |
| | BNOT | Produces a bit NOT. |
| | BOR | Produces a bit OR. |
| | BXOR | Produces a bit XOR. |
| ④ Date/Time controls | TIME$ | Returns the present time (hour/minute/second) by character string. |
| | DATE$ | Returns the date (year/month/day) by character string. |
| ⑤ Interrupt controls | ENABLE INTR | Enables interrupts to be received. |
| | DISABLE INTR | Disables interrupts to be received. |
| | ON END | Defines the branch of an EOF interrupt. |
| | ON KEY | Defines the branch of a key interrupt. |
| | ON ISRQ | Defines the branch of a SRQ interrupt of the main measurement section. |
| | ON SRQ | Defines the branch of a SRQ interrupt of GPIB. |
| | ON ERROR | Defines the branch of an error occurrence interrupt. |
| | OFF END | Releases the branch of an EOF interrupt. |
| | OFF KEY | Releases the branch of a key interrupt. |
| | OFF ISRQ | Releases the branch of an SRQ interrupt of the main measurement section. |

| Function | Command and Statement | Description |
|---|---|---|
| ⑤ Interrupt controls (cont'd) | OFF SRQ | Releases the branch of an SRQ interrupt of GPIB. |
| | OFF ERROR | Releases the branch of an error occurrence interrupt. |
| ⑥ Character-string manipulation | NUM | Converts the first character of a character string to the ASCII code. |
| | CHR$ | Converts a numeric character to an ASCII character. |
| | LEN | Obtains the length of a character string. |
| | POS | Positions character string 1 in character string 2. |
| | SPRINTF | Formats a string and enters it to a character-string variable. |
| ⑦ Memory card controls | CAT | Displays the contents of a memory card. |
| | CALL | Loads sub-program. |
| | CLOSE # | Close a file. |
| | COPY | Copies the file in the memory card. |
| | DIR | Outputs the name of stored file in the memory card. |
| | ENTER # | Reads data from a file. |
| | OPEN # | Open a file. |
| | OUTPUT # | Writes data into a file. |
| | INITIALIZE (or INIT) | Initializes a memory card. |
| | PURGE | Deletes the specified file. |
| | RENAME | Rename a file. |
| ⑧ Screen controls | CURSOR (or CSR) | Moves the cursor to the specified position. |
| | CLS | Clears the screen. |
| | SCREEN | Sets up the basic screen. |
| | KEY ON | Displays the user-defined menu. |
| | KEY OFF | Clears (deletes) the user-defined menu. |
| | PANEL | Enables/Disables the panel key control. |
| ⑨ Statements | BUZZER | Sounds the buzzer. |
| | DIM | Declares array variables. |
| | FOR TO STEP NEXT | Sets an iteration. |
| | BREAK | Exits from the current iteration. |
| | CONTINUE | Returns to the beginning of the current iteration. |
| | GOSUB | Branches to a subroutine. |
| | RETURN | Returns from the current subroutine. |
| | GOTO | Branches to the specified statement. |
| | IF THEN ELSE END IF | Executes the statements after evaluating conditions. |
| | INPUT (or INP) | Inputs a value to a variable. |
| | INTEGER | Declares integer-type variables. |

| Function | Command and Statement | Description |
|---|---|---|
| ⑨ Statements (cont'd) | LPRINT | Outputs data on a printer (RS-232). |
| | LPRINT USING (or USE) | Outputs formatted data on a printer (RS-232). |
| | MENU | Displays the arbitrary character string to the sort menu and waits until precessing the soft key. |
| | PAUSE | Stops execution temporarily. |
| | PRINT (or ?) | Displays characters on the screen. |
| | PRINT USING (or USE) | Displays formatted characters on the screen. |
| | PRINTER | Addresses a GPIB printer device. |
| | PRINTF (or PRF) | Displays formatted characters on the screen. |
| | READ DATA | Reads data from a DATA statement and assigns it to a variable. |
| | RESTORE | Specifies the DATA statement to read by the READ DATA statement. |
| | REM (or !) | Provides a comments. |
| | SELECT CASE END SELCT | Executes statements after evaluating condition. |
| | STOP | Stops program execution. |
| | WAIT | Holds the program execution for the specified period. |
| ⑩ GPIB commands | CLEAR | Transfers DCL and SDC. |
| | DELIMITER | Sets a delimiter. |
| | ENTER (or ENT) | Inputs GPIB data. |
| | GLIST | Outputs the program list to a GPIB printer. |
| | GPRINT | Outputs data to a GPIB printer. |
| | GPRINT USING (or USE) | Outputs formatted data to a GPIB printer. |
| | INTERFACE | Transfers IFC. |
| | CLEAR LOCAL | Places the specified device in the local status. |
| | LOCAL LOCKOUT | Places the specified device in the local locked-out status. |
| | OUTPUT (or OUT) | Outputs data to GPIB. |
| | REMOTE | Places the specified device in the remote status. |
| | REQUEST | Outputs SRQ to the standard GPIB. |
| | SEND | Outputs a set of GPIB data. |
| | SPOLL | Provides serial polling to the specified device. |
| | TRIGGER | Outputs GET. |

## 3.3 Description on commands and Statements

This section describes commands and statements in the sequence shown below.

| Command and Statement | Command and Statement |
|---|---|
| BUZZER | LPRINT |
| CALL | LPRINT USING or LPIRNT USE |
| CAT | MENU |
| CLEAR | OFF END |
| CLS | |
| CLOSE # | OFF ERROR |
| CONT | OFF KEY |
| CONTROL | OFF SRQ/ISRQ |
| COPY | ON END - GOTO/GOSUB |
| CURSOR or CSR | ON ERROR - GOTO/GOSUB |
| | ON KEY - GOTO/GOSUB |
| DATE$ | ON SRQ/ISRP - GOTO/GOSUB |
| DELIMITER | OPEN # |
| DIM | OUTPUT # |
| DIR | OUTPUT or OUT |
| DISABLE INTR | |
| EDIT | PANEL |
| ENABLE INTR | PAUSE |
| ENTER or ENT | PRINTF or PRF |
| ENTER # | PRINT or ? |
| FOR ... TO ... STEP BREAK/CONTINUE - | PRINT USING or PRINT USE |
| NEXT | PRINTER |
| | READ DATA/RESTORE |
| GLIST | REM or ! |
| GOSUB - RETURN | REMOTE |
| GOTO | RENAME |
| GPRINT | |
| GPRINT USING or GPRINT USE | REQUEST |
| IF THEN ELSE/ELSE IF/END IF | RUN |
| INITIALIZE or INIT | PURGE |
| INTEGER | SCREEN |
| INTERFACE CLEAR | SELECT CASE/CASE ELSE/END SELECT |
| INPUT or INP | SEND |
| | SPOLL (X) |
| KEY OFF | SPRINTF |
| KEY ON | STOP |
| LIST | TIME$ |
| LLIST | |
| LOCAL | TRIGGER |
| LOCAL LOCKOUT | WAIT |

```
BUZZER
```

[Outline]

Sounds the buzzer that the main device has inside.

[Format]

BUZZER <interval>, <time>

[Description]

The BUZZER statement sounds the specified <interval> for the specified amount of <time>.
<interval> : Integer 0 to 65535 (Hz)
<time> : Integer 0 to 65535 (millisecond)

[Intervals]

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 261 | : | Do | C | 277 | : | Do# | C# |
| 294 | : | Re | D | 311 | : | Re# | D# |
| 339 | : | Mi | E | | | | |
| 349 | : | Fa | F | 370 | : | Fa# | F# |
| 392 | : | Sol | G | 415 | : | Sol# | G# |
| 440 | : | La | A | 466 | : | La# | A# |
| 494 | : | Si | B | | | | |

If the value is doubled, the interval is raised by an octave. If the value is halved, the interval is lowered by an octave.
As the value becomes greater, the sound becomes higher.

[Example]

BUZZER 440, 1000 : Sounds la for one second.

[Note]

Since the BUZZER statement only gives the settings to the circuit that sounds the buzzer, the execution completes immediately while the buzzer continues sounding for the specified amount of time. (If one second is set as the amount of time, the control passes to the next statement immediately after the BUZZER statement gives the settings to the circuit.)
To sound buzzer continuously, place the WAIT statement after the BUZZER statement. The amount of time set in the WAIT statement must be equal to the one in the BUZZER statement.

[Program Example]

```
FOR I = 1 TO 8
  READ S
  BUZZER S, 1000
  WAIT 1000
NEXT I
DATA 261, 294, 330, 349, 392, 440
DATA 494, 523
```

[Result]

The buzzer sounds each interval for one second, changing intervals as follows: do, re, mi, fa, sol, la, si, and do.

```
CALL
```

[Outline]

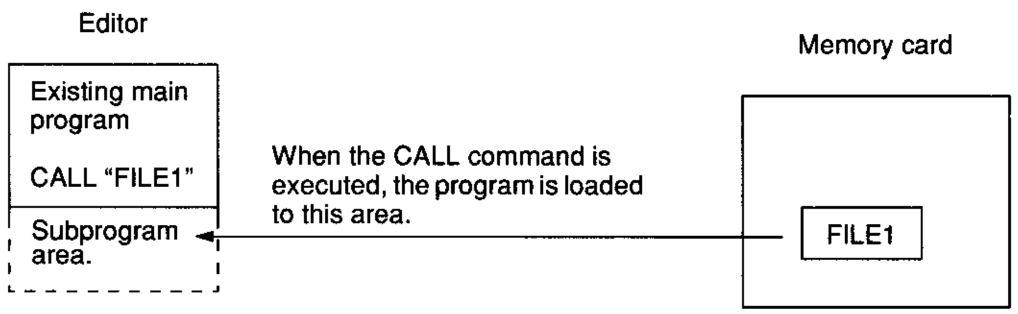
This function reads a file (subprogram) in the memory card and adds it to the end of a program created for executing it as a subroutine.

[Format]

CALL <filename>

[Description]

- The CALL command in the program existing in this system, which is regarded as the main program, reads a program from the memory card to add it to the end of the main program. The added program is regarded as a subprogram. (A subprogram is treated as a subroutine.)
  (Program CANNOT be continued)
- After the subprogram is read and executed as a subroutine by the CALL command in the main program, the control returns and restarts to execute the statement following the CALL command (the same as GOSUB and RETURN).



[Example]

CALL "FILE1" : Program "FILE1" is read and executed.

[Notes]

- The CALL command cannot be used in a subprogram. If used, an error results and execution is stopped.
  (CANNOT next CALL)
- The system deletes the loaded subprogram after execution of the subprogram and returns. Therefore, no program is displayed on the editor but the main program.

- Since variables are used in common with the main program, use of same variables in the loop process such as FOR to NEXT causes malfunction.
  Variables specific to subprograms (local variables) cannot be defined.
- When a subprogram is loaded, the RETURN command is automatically inserted. Even if the RETURN command already exists, operation is not influenced.

[Program Example]

```
• FOR I = 1 TO 10  :  This program is saved as "FILE1".
    PRINT I;       :
  NEXT I           :

• FOR K = 1 TO 5   :  This program is executed as the main program.
    CALL "FILE1"   :
    PRINT          :
  NEXT K           :
```

[Result]

```
loading...

12345678910
loading...

12345678910
loading...

12345678910
loading...

12345678910
loading...

12345678910
Program ended normally.
```

```
CAT
```

[Outline]

Outputs the list of the files stored in the memory card.

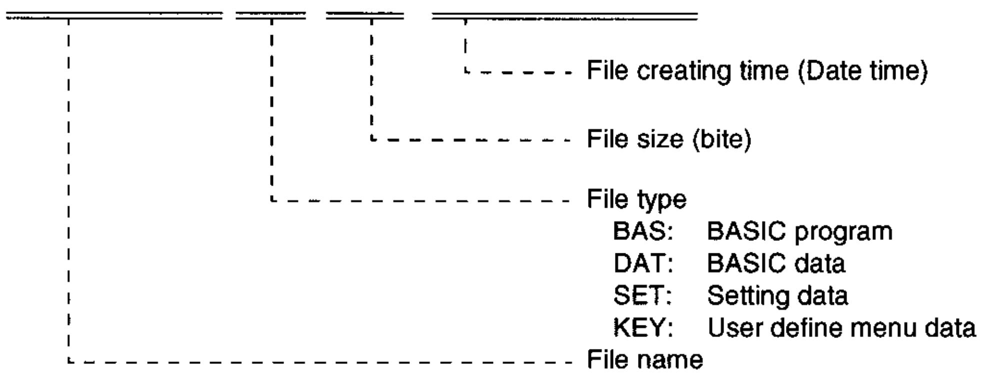
[Format]

CAT

[Description]

- The CAT command outputs the list of the files stored in the memory card.
- The CAT command should be entered in the in the BASIC mode of the editor.
- If the CAT command is executed, the list is displayed onthe screen, as follows:

```
FILE_1        BAS    256  1991-01-01   10:10
DATA_NO1      DAT  19200  1991-02-14   12:10
FILE_2        BAS    256  1991-01-15   10:20
PROGRAM001    BAS    128  1990-10-04   15:35
PROGRAM002    BAS    128  1990-12-24   09:04
PROGRAM003    BAS   1408  1991-05-05   11:22
NOISE         SET   1280  1991-03-03   13:45
FILE_4        BAS   1792  1991-05-05   14:56
FILE_5        BAS   1280  1991-07-14   17:24
```



File creating time (Date time)

File size (bite)

File type

| | |
|---|---|
| BAS: | BASIC program |
| DAT: | BASIC data |
| SET: | Setting data |
| KEY: | User define menu data |

File name

[Example]

CAT

[Note]

New memory cards cannot be used until they are initialized by the INITIALIZE command.

## CLEAR

[Outline]

Initializes every or specified devices connected to GPIB.

[Format]

CLEAR [device address { , device address } ]

[Description]

- If a CLEAR statement is executed without specifying device addresses, an universal command, Device Clear (DCL), will be sent. This command initializes every device on GPIB.
- If a device address (0 to 30) are specified after the CLEAR statement, an universal command, Selector Device Clear (SDC) is sent only to the device specified by the device address. This enables to initialize only the specified deivce.
  More than one device addresses can be specified.

[Example]

```
CLEAR
CLEAR 1
CLEAR 2, 5, 8
```

[Note]

- To specify numbers other than 0 to 30 as device addresses causes an error to interrupt the program execution. The following message will be displayed. (UNIT addr error in CLEAR)
- Not function in slave mode.

| [Sample Program] | [Result] |
|---|---|
| CLEAR 3 | : Device address 3 will be initialized. |
| OUTPUT 3; "CF3MZ" | : Sent data to device address 3. |
| CLEAR 3, 5 | : Device addresses 3 and 5 will be initialized. |
| OUTPUT 3, 5; "CF2MZ" | : Sent data to device addresses 3 and 5. |
| CLEAR | : Every connected device will be initialized. |

```
CLS
```

[Outline]

Deletes the screen of the main-body CRT.

[Format]

CLS [1 I 2]

[Description]

- The CLS statement clears the character and graphic screens of the main-body CRT.
- The following parameters are available:
  Not specified: Clears only the character screen.
  1: Clears only the graphic screen.
  2: Clears both character and graphic screen.

[Example]

```
CLS
CLS1
CLS2
```

[Note]

Before use of the graphic screen, send the Basic command "SCREEN 3".

[Program Example]

Example 1
```
PRINTER CRT
SCREEN 3
GLINE (1, 0, 0, 720, 439)
PRINT "TEST"
WAIT 2000
CLS
```

Example 2
```
PRINTER CRT
SCREEN 3
GLINE (1, 0, 0, 720, 439)
PRINT "TEST"
WAIT 2000
CLS 1
```

Example 3
```
PRINTER CRT
SCREEN 3
GLINE (1, 0, 0, 720, 439)
PRINT "TEST"
WAIT 2000
CLS 2
```

[Result]

Example 1
The character string "TEST" is deleted and the graphics objects remain.

Example 2
The graphics objects are deleted and the character string "TEST" remain.

Example 3
Both graphics objects and the character string "TEST" remain.

```
CLOSE #
```

[Outline]

Closes the file that is assigned to the specified file descriptor.

[Format]

CLOSE <# file descriptor>

[Description]

- All of the opened files should be closed before the memory card is removed or the main body is turned off. Otherwise, the file is broken.
- In the BASIC program, the files are not automatically closed if the execution of a program is stopped by the PAUSE statement or pressing the STOP key. In other cases, all files are automatically closed when a program ends.
  Files are closed when a program is terminated because of errors.
  If there is an ON ERROR definition in the program, the files are not closed.
  Therefore, the opened files should be explicitly closed if the program ends because of erros, as follows:

  ```
  CLOSE *
  ```

  This commands closes all files.

[Example]

```
CLOSE #FD
CLOSE *
```

[Note]

All of the files that has been opened in a program should be closed in the end of the program.

[Program Example]

```
OPEN "FFF" FOR OUTPUT AS # FD
FOR I = 1 TO 100
  OUTPUT # FD; I
NEXT I
CLOSE # FD
OPEN "FFF" FOR INPUT AS # FD
FOR I = 1 TO 100
  ENTER # FD; N
  PRINT N
NEXT I
CLOSE # FD
```

[Result]

The above program first writes real numbers from 1 to 100 into the file "FFF". It then reads the numbers and displays then.
(The numbers 1.0 to 100.0 are thus printed.)

```
CONT
```

[Outline]

Resumes the execution of a BASIC program.

[Format]

CONT [ <label> ]

[Description]

- When a label is omitted, the program execution restarts from the next line of that where it has stopped.
  If this command is executed when the program has ended, however, the following error message appears:
  (Program CANNOT be continued)
- When a label is specified, the program execution starts from the command following the label.

[Example]

```
CONT
CONT  *ABC
```

[Note]

- The CONT command with a label should be executed in the BASIC mode.
  (To execute a CONT command without a label, press the CONT key on the main-body CRT.
- Contents of variables will not be initialized with a CONT command.
- To stop executing a program, use PAUSE.

## CONTROL

[Outline]

Sets the values related to BASIC controls.

[Format]

CONTROL <register number> ; <numeric expression>
<register number> : Specify any of 2, 4 and 5.

[Description]

| | |
|---|---|
| Register number 2: | Specifies the left margin for the output on printers. |
| Register number 4: | Specifies GPIB adress. |
| Register number 5: | Switches Master/Slave Mode |

- Register 2

  The value in register 2 will be used as the offset size from the left edge of the paper, when data is output with a LIST command.
  Each line will be preceded by the number of the spaces which is determined by the offset size.

- Register 4

  Specifies GPIB adress.
  Specifies the numeric in 0 to 30.

  * At initial value at power ON is 0.

- Register 5

  Switches Master/Slave Mode

  0: Slave mode
  1: Master mode

[Example]

Register 2

To put 5 spaces before each line of the data to be printed, specify CONTROL 2; 5.

```
----- PRINT "ADVANTEST"
----- FOR I = 1 TO 10
----- PRINT I
----- NEXT I
```

Register 4

Specifies GPIB address 10.
CONTROL 4; 10.

```
COPY
```

[Outline]

The file in the memory card can be copied.

[Format]

COPY <[A: | B:] FILE1>, <A: | B:] FILE2>

[Description]

- The contents of a file FILE1 is copied to FILE2 in the memory card.
- When copy carried out, the drive (A or B) can be designated. When the designation to the drive is omitted, the active drive becomes default.

[Example]

```
COPY "A:PROG1.BAS", "B:PROG2.BAS"
COPY "DATA1.DAT", "DATA2.DAT"
COPY "A:TEST1.BAS", "TEST2.BAS"
```

[Note]

- Before changing, carry out CAT command or Load of editor and ensure the file to be changed.
- The file name is limited within eight characters and the extension is limited within three characters.
- Copy cannot be carried out when write protect of the memory card is on.

```
CURSOR or CSR
```

[Outline]

The cursor can be moved to the specific position.

[Format]

CURSOR <X : Column>, <Y : Line>
CSR <X : Column>, <Y : Line>

[Description]

The cursor is moved to the position which is designated by the column and line, and is turned on or off. Upper left position becomes (0, 0) on the CRT. Set up range is (0, 0) to (25, 20) when menu on and (0, 0) to (34, 20 ) when menu off.

[Example]

```
CURSOR 10, 12
CSR 2, 3
```

[Note]

- It is not displayed when the screen mode is 0.
- Be sure to carry out PRINTER CRT before using the cursor.

[Program Example]

```
SCREEN 3
PRINTER CRT
CURSOR 10, 15
```

```
DATE$
```

[Outline]

The date (year/month/day) can be returned by the character string.

[Format]

DATE$

[Description]

The date (year/month/day) can be returned by the character string.

[Example]

DATE$

[Program Example]

```
A$ = DATE$
PRINT A$
```

```
DELIMITER
```

[Outline]

Selects delimiters from four types, and set.

[Format]

DELIMITER <X>
<X>: numeric expression either of 0, 1, 2 and 3

[Description]

- OUTPUT 0 to 30 commands set terminators (delimiters) for output.
- Delimiter selection numbers and types are listed as follows.

| Selection number | Delimiter type |
|---|---|
| 0 | Output "CR" or "LF", and two bytes of a single line signal EOI. |
| 1 | Output a byte of a "LF". |
| 2 | Output a single line signal simultaneously with the last byte of data. |
| 3 ◎ | Output two bytes of "CR" or "LF". |

(◎ means default at turning on.)

[Example]

DELIMITER 0
DELIMITER 1
DELIMITER 2
DELIMITER 3

[Note]

- If numbers other than 0 to 3, an error will be resulted to interrupt program execution. The following message will be displayed.
  (Invalid value in DELIMITER)
- Default of the DELIMITER statement is 3.

| [Sample Program] | [Result] |
|---|---|
| DELIMITER 0 | : Set a delimiter at CR and LF + EOI. |
| OUTPUT 3; "CF3MZ" | : Output data to device address 3. |
| DELIMITER 1 | : Set a delimiter at LF. |
| OUTPUT 3; "CF4MZ" | : Output data to device address 3. |
| DELIMITER 2 | : Set a delimiter at the last byte + EOI. |
| OUTPUT 3; "CF5MZ" | : Output data to device address 3. |
| DELIMITER 3 | : Set a delimiter at CR and LF. |
| OUTPUT 3; "CF6MZ" | : Output data to device address 3. |

## DIM

[Outline]

The size of the array variables or the length of character string variables are declared.

[Format]

DIM <X> {, <X>}

X: Variable name ([N:>] <Numerical expression> {, <Numerical expression>} ) | Character string variable name ' ' [ ' ' <Numerical expression> ' ' ] ' ' \ \

N: First number of the numerical subscript of the array variable. (0 or 1)

[Description]

- The size of the array variables or the length of character string variables is declared. Maximum value is designated for the numerical subscript of the array element. (If variables are used without declaration then the number of element for each array becomes maximum value of 10 and the length of the character string becomes 18 characters. Moreover, when the first number of the numerical subscript of the array variable is not designated, it becomes 1.)
- The array is declared by DIM command then the amount of memory is dedicated according to the designated size. If too large number of value is declared for the array then the working memory becomes not enough, therefore, program execution is stopped and the error is occurred. (Memory space full)
- If the numerical expression for the size of the array variable is set in the real number but the result has no decimal fraction. This integers are processed, declared and referred.
- The numerical expression for the length of the character string variable is declared the length of the character string.
- If the multiple number of numerical subscripts are designated then the array variable has the designated number of dimensions. (The number of dimensions can be designated as the memory space allows.)

[Example]

```
DIM A (20)
DIM B (30), C (25, 5)
DIM S$ [50]
DIM D (15), T$ [50]
DIM E (3.8, 5, 2)
DIM F (1 : 10)
```

[Note]

- The minimum value of the numerical subscript of the array variable is determined by the first number of the array.
- The maximum value of the array declaration is 32767 but the amount of memory dedicated to the purpose is changed by the program size.
- The length of the character string is maximum 128.
- When the array is declared for the integer type variable, refer to INTEGER command.
- If the designated numerical subscript of the array variable is out of range or the array variable is referred by the negative number then the error occurred. (Array's range error or Invalid dimension param.)
- The array declaration of the multiple number of dimensions cannot be carried out for the character string variable.

[Program Example]

```
DIM A (0 : 10)
FOR I = 0 TO 10
  A (I) = I
NEXT I
```

```
DIR
```

[Outline]

The name of stored file in the memory card is output.

[Format]

DIR

[Description]

This is the same as CAT command.

[Example]

DIR

[Note]

New memory card cannot be used until they are initialized by INITIALIZE command.

```
DISABLE INTR
```

[Outline]

Disables the reception of the interrupts caused by the ON (KEY/SRQ/ISRQ) instructions.

[Format]

DISABLE INTR

[Description]

Disable the reception of interrupts that has been enabled by an ENABLE INTR instruction.

[Note]

- When a branch is caused by an interrupt, a DISABLE INTR instruction should be executed at the destination of the branch. Otherwise, the interrupts will be nested.
- If you specify the interrupt operation as a subroutine, we recommend that you place an ENABLE INTR instruction immediately before the RETURN instruction of the subroutine. The operation will take place smoothly.

[Program Example]

```
INTEGER S                  : Declares an integer-type variable.
ON ISRQ GOSUB *SWPEND      : Defines the destination of an interrupt branch.
OUTPUT 31; "IP SW1SC S0"   : Sets IP, a second of sweep, interrupt output.
* L                        : ┐ (Loop)
  GOSUB *ONESWP            : │ Subroutine for one sweep.
  M = MAX (0, 700, 0)      : │ Maximum level (built-in function)
  PRINT M                  : │ Maximum level output
  GOTO *L                  : ┘ Causes a jump to the label *L.
!
* ONESWP
  F = 0                    : F = 0
  ENABLE INTR              : Enables interrupts to be received.
  OUTPUT 31; "SI"          : Starts a single sweep.
* LL                       : ┐ (Loop)
  IF F THEN RETURN         : │ Returns if F is true.
  GOTO *LL                 : ┘ Causes a jump to the label *LL.
!
* SWPEND
  DISABLE INTR             : Disables interrupts from being received.
  S = SPOLL (31)           : Serial polling to the main-body measurement section.
  IF S BAND 4 THEN F = 1   : Enter 1 to F when bit 3 is on.
  RETURN                   : Return
```

[Result]

Outputs the maximum level of each sweep.

```
EDIT
```

[Outline]

The ate editor function is started.

[Format]

EDIT

[Description]

The ate editor function is started and the BASIC program can be created.

[Example]

EDIT

[Note]

For starting the ate editor, interconnect the external terminal to the RS-232 on the rear panel.

```
ENABLE INTR
```

[Outline]

Enables the interrupts caused by the ON (KEY/SRQ/ISRQ) instructions to be received.

[Format]

ENABLE INTR

[Description]

Enables the reception of interrupts that has been disabled with a DISABLE INTR instruction.

[Note]

- When a branch is caused by an interrupt, a DISABLE INTR instruction should be executed at the destination of the branch. Otherwise, the interrupts will be nested.
- If you specify the interrupt operation as a subroutine, we recommend that you place an ENABLE INTR instruction immediately before the RETURN instruction of the subroutine. The operation will take place smoothly.
- After running a program, the reception of the interrupts are disabled.
  Execute an ENABLE INTR instruction before using interrupts.

[Program Example]

```
ON KEY 1 GOSUB *K1
ON KEY 2 GOSUB *K2
ENABLE INTR
* L
  GOTO *L
!
* K1
  DISABLE INTR
  PRINT "KEY1"
  ENABLE INTR
  RETURN
* K2
  DISABLE INTR
  PRINT "KEY2"
  ENABLE INTR
  RETURN
```

[Result]

When 1 or 2 on a full keyboard or 1 or 2 of the ten-key pad of the main-body panel is pressed, KEY1 or KEY2 is displayed on the screen.

```
ENTER or ENT
```

[Outline]

Enters data from the measuring device of the main unit or the parallel I/O.

[Format]

ENTER <device address> I 31; <X>
or
ENT <device address> I 31; <X>

Device address 0 to 30: Enter data from the devices on GPIB.
31: Enter data from the measuring device of the main unit.

<X>: [ numeric variable | character variable {, numeric variable | character variable } ]

[Description]

- The device addresses 0 to 30 enter data from the specified devices on GPIB (below: CONTROLLER) of the rear panel of the main unit.
- To set 31 causes data to enter from the measuring device of the main unit through the internal memory into a variable.
- More than one variables can be specified by delimiting a variable with commas (,) or delimiters. In this case, received data will be entered every delimiting by commas. If specified number of the variables are more than the received data, the variables which are not corresponding with the data receives nothing. On the contrary the variables are more than the received data, the remainder of those data will be ignored. 1,024 characters can be received by the ENTER command. Even if more than 1,024 characters are sent, the remainder will be ignord.

[Example]

```
ENTER 5; A
ENTER 5; A, B
ENTER 5; S$

ENTER 31; A
ENTER 31; A, B
ENTER 31; S$
```

[Note]

- To address numbers other than 0 to 31 causes an error to terminate the program execution. The following message will displayed.
  (UNIT addr error in ENTER)
- If the enter data type dose not correspond with the variable type, an error will be caused.
- Refer to "5. MASTER/SLAVE MODE" when using in slave mode.

[Sample Program]

①
```
DIM B$ [80]
OUTPUT 3; "CF?"
ENTER 3; A
OUTPUT 5; "SP?"
ENTER 5; B$
PRINT A
PRINT B$
```

②
```
DIM S$ [80]
OUTPUT 31; "CF1.23MZ"
OUTPUT 31; "CF?"
ENTER 31; A
ENTER 31; S$
PRINT A
PRINT S$
```

## ENTER #

[Outline]

Inputs (reads) data from the file that is assigned to the specified file descriptor.

[Format]

ENTER <# file descriptor> ; <X> [, <X>]
<X>: entry (numeric variable, character string variable)

[Description]

- The ENTER # statement reads data from the file that is assigned to the specified file descriptor. The read data is formatted for the data type of the associated entry and entered to the entry.
- According to the type specified with the OPEN statement, the data is read in the following format.

BINARY type

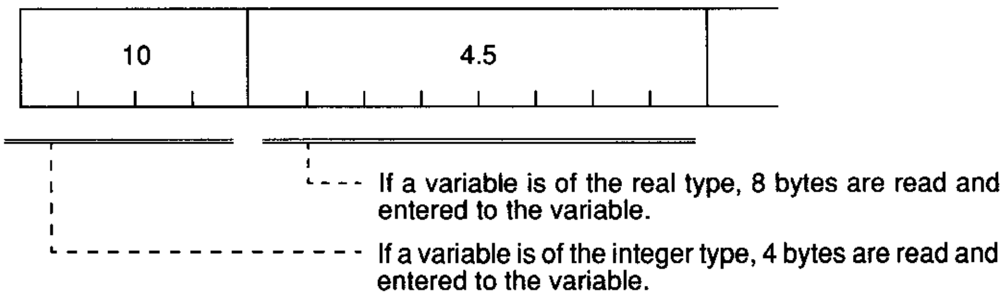
Data is read in the same type that of the internal expression.

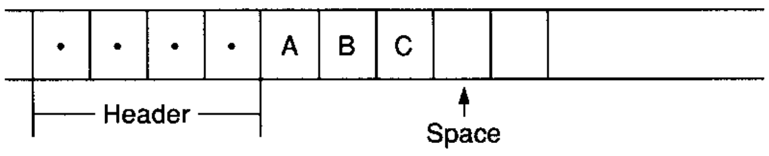
The data length differs according to the type, as follows:

| | | |
|---|---|---|
| Integer type | : | 4 bytes |
| Real type | : | 8 bytes |
| Character string | : | Data size (number of bytes) is indicated by the 4-byte header. |

Example)

```
INTEGER I
OPEN "FILE" FOR INPUT AS # FD
ENTER # FD; I, R, S$
```


10
4.5
If a variable is of the real type, 8 bytes are read and entered to the variable.
If a variable is of the integer type, 4 bytes are read and entered to the variable.



If a variable is a character string, the header is read first.
The header indicates the length of the data to read.
Then the specified length of the data is read and assigned to the character string variable.

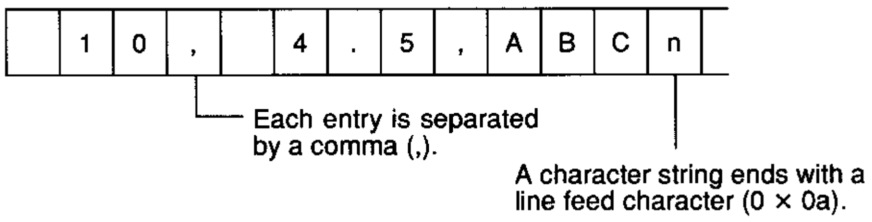
TEXT type

In regardless of the number of the input entries, data is read until a line feed character is detected. Each data object is terminated by a comma. It is converted to the type of the entry and entered to the variable.

Data is converted to the ASCII codes and output. A numeric data object is lead by a space or a sign. A character string ends with a line feed character (0 × 0a).

Example)

```
INTEGER I
OPEN "FILE" FOR INPUT AS # FD; TEXT
ENTER # FD; I, R, S$
```

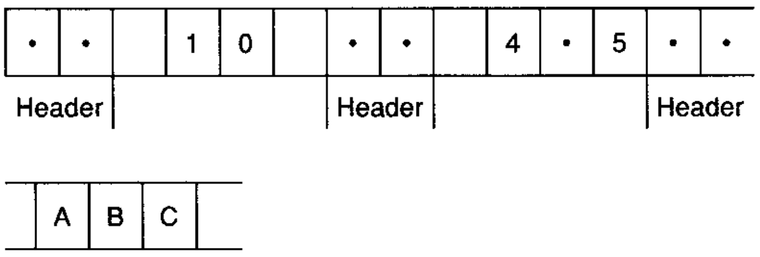

1 0 , 4 . 5 , A B C n
Each entry is separated by a comma (,).
A character string ends with a line feed character (0 × 0a).

ASCII type

The 2-byte header is read first. The header indicates the length of the data to read. According to the length, data is read, converted into the type of the associated variable, and entered to the variable.

Example)

```
INTEGER I
OPEN "FILE" FOR INPUT AS # FD; ASCII
ENTER # FD; I, R, S$
```



[Example]

```
ENTER  # FD; S$
ENTER  # FD; A, B
```

[Note]

- Specify a file descriptor that has been opened with the OPEN statement.
- A header is a separator of entries and has the length of a data object.
- If there are more entries than the data objects, no data is entered to excessive variables. The excessive variables holds the old data.
  If the number of the data objects is greater than that of the entries, however, the excessive objects are abandoned.
- The number of the bytes to read is determined by the type of an entry. The data objects should be input (entered) in the same type that will be used for output. Otherwise, the contents of the output will differ from those of the input.
- The following error message appears if this statement is used for the file that has not been opened with the OPEN statement:
  (file NOT open)

[Program Example]

Execute the program of OUTPUT # before executing the following program.

```
OPEN "A" FOR INPUT AS  # FA; BINARY
INTEGER S
FOR I = 1 TO 10
  ENTER  # FA; R, S
  PRINT R, S
NEXT I
CLOSE  # FA

OPEN "AA" FOR OUTPUT AS  # FB; TEXT
INTEGER S
FOR I = 1 TO 10
  ENTER  # FB; R, S
  PRINT R, S
NEXT I
CLOSE  # FB

OPEN "AAA" FOR OUTPUT AS  # FC; ASCII
INTEGER S
FOR I = 1 TO 10
  ENTER  # FC; R, S
  PRINT R, S
NEXT I
CLOSE  # FC
```

[Result]

All of the left programs print the same contents, as follows:

```
 1.0      1
 2.0      2
 3.0      3
 4.0      4
 5.0      5
 6.0      6
 7.0      7
 8.0      8
 9.0      9
10.0     10
```

```
FOR... TO... STEP
 BREAK/CONTINUE - NEXT
```

[Outline]

Repeats the execution of the statements between for the statement and the NEXT statement.

[Format]

FOR <numeric variable> = <initial value> TO <final value> [STEP <increment>]

BREAK
CONTINUE
NEXT <numeric variable>

<initial value>, <final value>, <increment> : numeric expression

[Description]

- The specified numeric variable is used as a counter. Its value is increased in the specified increments beginning with the initial value to the final value. If the counter value exceeds the final value, the looping is terminated and control moves to the statement following the NEXT statement.
- The NEXT statement increases or decreases the counter value. It adds the increment to the numeric variable and returns control to the FOR statement.
- If STEP <increment> is omitted, the increment is taken to be 1.
- FOR - NEXT statements can be nested.
- The same variables must be used after FOR and NEXT that are paired.
  Otherwise, the following error occurs: (NEXT without FOR).
- The BREAK statement allows the control to exit from a FOR - NET loop.
- The CONTINUE statement adds the increment to <numeric variable> and returns control to the beginning of the loop even before the control reaches the NEXT statement.

[Example]

```
FOR I = 1 TO 5                 : 1-1
  PRINT I                      : 1-2
NEXT I                         : 1-3

FOR J = 1 TO 20 STEP 3         : 2-1
  PRINT J                      : 2-2
  IF J > 12 THEN BREAK         : 2-3
NEXT J                         : 2-4

FOR K = 10 TO 0 STEP - .5      : 3-1
  IF K > 3 THEN CONTINUE       : 3-2
  PRINT K                      : 3-3
NEXT K                         : 3-4
```

1-1 : Enters 1 to the counter I and loops until I exceeds 5.
1-2 : Prints I.
1-3 : NEXT (returns to 1-1).

2-1 : Enters 1 to the counter J and loops increasing I in increments of 3 until it exceeds 20.
2-2 : Prints J.
2-3 : Exits (BREAK) from the loop if J is larger than 12.
2-4 : NEXT (returns to 2-1).

3-1 : Enters 10 to the counter J and loops increasing I in increments of -0.5 until it becomes less than 0.
3-2 : If K is greater than 3, CONTINUE (returns to 3-1).
3-3 : Prints K.
3-4 : NEXT (returns to 3-1).

[Note]

- If the value of the numeric varialbe used as the loop counter is changed in a FOR-NEXT loop, looping may not be performed normally (endless looping, etc.).
- If a GOTO statement or the other jump statement transfers control into a FOR-NEXT loop from the outside or if it passes control outside from the inside the FOR - NEXT loop, the operations of the program may not be ensured.
- BREAK and CONTINUE statements may only be used within FOR - NEXT loops.
- In the following cases, the FOR - NEXT loop cannot be executed and control jumps to the statement following the NEXT statement:
  1: The increment is positive and <initial value> is greater than <final value>.
  2: The increment is negative and <initial value> is less than <final value>.

[Program Example]

See the example shown above.

[Result]

```
Program 1
   1.0
   2.0
   3.0
   4.0
   5.0
Program 2
   1.0
   4.0
   7.0
  10.0
  13.0
Program 3
   3.0
   2.5
   2.0
   1.5
   1.0
   0.5
   0.0
```

```
GLIST
```

[Outline]

Outputs a program list to the GPIB printer.

[Format]

GLIST [ <label> [, ] ]

[Description]

- GLIST outputs a program list, whose location is specified by a labels, to the GPIB printer.
- To specify a label, and comma (,) after the label causes the program list to be output from the label position to the end of it.
- To output to a printer, first transfer the BASIC program (compile & run), next terminate the program, then execute a PRINTER and GLIST command in the BASIC mode. Those commands can be located in a program.

[Example]

```
GLIST
GLIST *ABC,
```

[Note]

- Though to specify by line numbers is possible, the format differs from the above.
  GLIST [ <output start line> ] [, [ <output end line> ] ]

  (Example)
  ```
  GLIST 100
  GLIST 100,
  GLIST ,200
  GLIST 100, 200
  ```
- The connecting GPIB connector should be used the CONTROLLER side of the rear panel.
- Refer to the (56) PRINTER command for specifying the GPIB addresses of a printer.
- Note that if the device addresses are missed or no device connects to the specified address, this command will be ignored and the program will proceeds to the next step.
- The output program list is the last one transferred (compile & run) to the buffer of the BASIC interpreter side.

[Program Example]

① Transfer a program, whose list is to be output, into the editor, then select "**compile & run**" from the popup menu.
After the transfer of the program has been completed and the program starts to execute, terminate it by "control + C". (You may have the program execute completely.)
Then the mini-window of the **BASIC mode** appears at the lower right of the display, execute a PRINTER command and a GLIST command. (Those commands can be selected from the popup menu.)

(Example)

```
PRINTER 3
GLIST
```

② Allocate the following commands at the head of a program whose list is to be output;

PRINTER command
GLIST command
STOP command

Then execute "compile & run".

(Example)

```
PRINTER 3
GLIST
STOP
   :
```

[Result]

① The list is output to the GPIB address 3 printer.

② To execute "**compile & run**" causes the list output to the GPIB address 3 printer.

## GOSUB - RETURN

[Outline]

Branches to the specified subroutine and returns back.

[Format]

```
GOSUB <label>
RETURN
```

[Description]

- The GOSUB statement passes control to the subroutine specified with <label>.
- When the control reaches the RETURN statement, it returns to the statement following the GOSUB statement where it has branched.
- GOSUB - RETURN pairs can be nested. Therefore, control may branch from a subroutine to the other subroutine. If too many levels of nesting is used, the memory capacity may run out and the following error occurs: (GOSUB nest overflow).

[Example]

```
GOSUB *S1

*S1
 A = A*2
 RETURN
```

[Note]

If the specified label does not exist in the program, the following error occurs and execution of the program is stopped when control jumps with the GOSUB statement. (Undefined LABEL)

[Program Example]

```
FOR I = 2 TO 5                      : Enters 2 to I and loops until I exceeds 5.
  A = 2                             : A = 2
  B = A                             : B = A
  GOSUB *SUB1                       : Jumps to the subroutine labeled *SUB1.
  PRINTF "2 ^ %2d = %5d\n\r", I, B
                                    : Outputs the data.
NEXT I                              : Next
STOP                                : Program end
!                                   :
*SUB1                               : Label *SUB1
 FOR C = 1 to I-1                   : Assigns 1 to C and loops until C exceeds I-1.
   GOSUB *SUB2                      : Jumps to the subroutine labeled *SUB2.
 NEXT C                             : NEXT
 RETURN                             : Return
!                                   :
*SUB2                               : Label *SUB2
 B * = A                            : B = B*A
 RETURN                             : Return
```

[Result]

```
2 ^ 2 =  4
2 ^ 3 =  8
2 ^ 4 = 16
2 ^ 5 = 32
```

## GOTO

[Outline]

Branches to the specified label.

[Format]

GOTO <label>

[Description]

A GOTO statement causes a branch to the specified <label> unconditionally.

[Example]

GOTO *LA

[Note]

If the specified label does not exist in the program, the following error occurs and the execution of the program is stopped when control jumps with the GOSUB statement. (Undefined LABEL)

[Program Example]

```
I = O                      : Enters 0 to the variable I.
*L                         : Label name
 I = I + 1                 : Adds 1 to the variable I.
 IF I = 10 THEN STOP       : Terminates when I reaches 10.
 PRINT I                   : Outputs I.
 GOTO *L                   : Jumps to *L.
```

[Result]

```
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
```

## GPRINT

[Outline]

Outputs numeric and character strings to a printer connected with GPIB.

[Format]

```
GPRINT [ <X> { [ , | ; <X> ] } ] [ < ; > ]
<X> :  numeric expression | character expression
```

[Description]

- GPRINT outputs numeric and character strings to the GPIB address device specified with PRINTER command.
- The numeric or character expression can be delimited with a semicolon (;) or comma (,) to specify more than one.
- If a semicolon is put at the end of a PRINT statement, the line will not start a new line. Therefore when the next PRINT statement is executed, the output will follows the previous output.

[Example]

```
S$ = "DEF"                    : ①
GPRINT "ABC" ;                : ②
GPRINT S$                     : ③
GPRINT "A =", A               : ④
GPRINT "CF", CFQ, "KHz"       : ⑤
GPRINT A + 100                : ⑥
```

① Enters "DEF" to the character string variable S$.
② Outputs "ABC" to a printer without no line feed.
③ Outputs the character variable S$.
④ Outputs the character string constants and the numeric.
⑤ Outputs the character string constants and the numeric.
⑥ Adds 100 to the numeric variable and output.

[Note]

- Before an output to the GPIB printer, set the address of the GPIB printer with PRINTER command exactly. (If the address differs from the address of the GPIB printer, the output does not fed to the printer.)
- If the device address are missed or no device connects to the specified address, this command will be ignored and the program will proceed to the next step.

[Sample Program]

```
PRINTER 5
FOR I = 1 TO 10
  GPRINT "I =" ;
  GPRINT I
NEXT I
```

[Result]

The GPIB address 5 printer output from 1 to 10.

## GPRINT USING or GPRINT USE

[Outline]

Edits numeric or character strings and so on, and output to the GPIB.

[Format]

GPRINT USING <image specification> ; { , <X> }
or
GPRINT USE <image specification> ; { , <X> }

<X> : numeric expression I character expression

[Description]

- Numeric or character strings are edited and output to the GPIB port in the ASCII cord.
- Numeric or character strings are output to the GPIB address devices specified with a (56) PRINTER command. (Refer to the PRINTER command.)
- To specify the image specification, represent with character strings and delimit with comma (,). (The line will be fed automatically.)

[Image specification list]

| | |
|---|---|
| D ................................ | To output spaces on the rest of the specified fields. |
| Z ................................ | To insert 0s on the rest of the specified fields. |
| K ................................ | To output numeric just as they are. |
| S ................................ | To add a +/- symbols at all times. |
| M ................................ | To add a - symbols to negative values, and add a space to positive values. |
| . (Decimal point) ........... | To output a decimal point. |
| E ................................ | To output with the exponent form (e, sign, exponent). |
| H ................................ | To output numeric or character strings as they are with the European type decimal point. |
| R ................................ | To output the European type decimal point. |
| * ................................ | To output *s on the rest of the specified fields. |
| A ................................ | To output a character. |
| k ................................ | To character strings just as they are. |
| X ................................ | To output spaces. |
| Listeral .......................... | To write a literal in format specifications, put them in \"s. |
| B ................................ | Output numeric with the ASCII cord. |
| @ ................................ | To feed lines. |
| + ................................ | To move the outputting positions to the top of the same line. |
| - ................................ | To move the outputting positions to the next line. |
| # ................................ | To not feed lines at the last position. |
| n ................................ | To output with accuracy of n decimals. If specify n to character strings, output value will be the length of the specified character strings. |

[Example]

```
GPRINT USING "DDD.DD" ; 1. 2
GPRINT USE "ZZZ.ZZ" ; 1. 2
```

[Note]

- The strings printed with GPRINT USING will not be returned. Insert return symbols in the parameters of GPRINT USING.
  GPRINT USING "K, k" 123, "\r"
- If the number specified with the image specification is more than the number specified with the parameter, an error will be caused.
  (Unmatched IMAGE-spec in USING)
- Before an output to the GPIB printer, set the address of the GPIB printer with PRINTER command exactly. (If the address differs from the address of the GPIB printer, the output does not fed to the printer.)
- If the device address are missed or no device connects to the specified address, this command will be ignored and the program will proceed to the next step.

[Program Example]

```
PRINTER 5
GPRINT USING "DDD.DD,k"; 1.2, "\r"
GPRINT USE "ZZZ.ZZ, k"; 1.2, "\r"
GPRINT USE "***.ZZ, k"; 1.2, "\r"
A$ = "K, 5X, B, DDDRDD, k"
GPRINT USE A$; 2.34, 65, 1.2, "\r"
```

[Result]

The GPIB printer outputs as follows.

```
1.20
001.20
**1.20
2.34      A     1, 20
```

```
IF THEN ELSE/ELSE IF/END IF
```

[Outline]

Tests a condition and causes a branch or executes the specified statement according to the result.

[Format]

```
•  IF <logical expression> THEN <statement>
•  IF <logical expression> THEN
          <compound statement>
   END IF
•  IF <logical expression> THEN
          <compound statement>
   ELSE
          <compound statement>
   END IF
•  IF <logical expression> THEN
          <compound statement>
   ELSE IF <conditional expression> THEN
          <compound statement>
   ELSE IF <conditional expression> THEN
          <compound statement>
              •
              •
   ELSE
          <compound statement>
   END IF
```

[Description]

- According to the condition of <logical expression, the execution of the program is controlled. If the result of <logical expression> is zero, it is false. If it is not zero, the logical expression is true.
- Not only logical expressions but also numeric expressions may be used as<logical expression>. The numeric expression is evaluated first.
  Then, if the result is zero, the expression is false. Otherwise, the expression is true.
- If <logical expression> is true, the THEN statement is executed.
  The THEN statement may be followed by a statement. (In this case, the IF statement should be written in a single line.)
- To continue more than one statement after THEN, terminate the line by THEN first. Then place the statements over the following lines and terminate the continuation by END IF. The ELSE statement may be placed to specify the statements to execute if the condition is false.

- The ELSE IF statement allows two or more logical expressions to be placed.
- Relational operators include the following:

| Symbol | Meaning | Example |
|---|---|---|
| - (or ==) | Equal to | X = Y, X == Y |
| <> | Not equal to | X <> Y |
| < | Less than | X < Y |
| > | Greater than | X > Y |
| <= | Less then or equal to | X <= Y |
| >= | greater than or equal to | X >= Y |

***Note:*** ***'=<' or '=>' cannot be substitute for '<=' or '>='.***

- Relational operators can be joined with logical operators.
  (See the description on the logical operators: operational expression.)

| |
|---|
| NOT<br>AND<br>OR<br>XOR |

[Example]

```
IF A = 1 THEN PRINT "A = 1"

IF B > 10 THEN
  PRINT "B > 10"
END IF

IF B > 2 THEN
  PRINT "B > 2"
ELSE
  PRINT "B <= 2"
END IF

IF 1 < C AND C < 5 THEN
  PRINT "1 < C < 5"
ELSE IF C = 10 THEN
  PRINT "C = 10"
ELSE
  PRINT "C:ELSE"
END IF
```

[Note]

If an IF block extends over more than one line, the block must end with END IF. If the number of IFs does not match the number of END IFs, the following error occurs: (Unbalanced IF block)

[Program Example]

```
FOR I = 1 TO 100                : Assigns 1 to I and loops until I exceeds 100.
  IF 10 <= I AND I <= 20 THEN   : ① If I is greater than or equal to 10 and less than or equal
                                     to 20,
    PRINT "10 <= I <= 20"       :    Outputs the character string as shown.
  END IF                        : Terminates the IF block ①.
  IF I = 50 OR I = 60 THEN      : ② If I = 50 or I = 60,
    IF I = 50 THEN              :      ③ If I = 50,
      PRINT "I = 50"            :         Outputs the character string as shown.
    ELSE                        :      ③ If I is not equal to 50,
      PRINT "I = 60"            :         Outputs the character string as shown.
    END IF                      :    Terminates the IF block ③.
  ELSE IF I = 70 THEN           : ② If I = 70,
    PRINT "I = 70"              :    Outputs the character string as shown.
  ELSE IF I = 90 THEN           : ② If I = 90,
    PRINT "I = 90"              :    Outputs the character string as shown.
  END IF                        : Terminates the IF block ②.
NEXT I                          : Retruns to the beginning of the loop.
```

[Result]

```
10 <= I <= 20
10 <= I <= 20
10 <= I <= 20
10 <= I <= 20
10 <= I <= 20
10 <= I <= 20
10 <= I <= 20
10 <= I <= 20
10 <= I <= 20
10 <= I <= 20
10 <= I <= 20
I = 50
I = 60
I = 70
I = 90
```

```
INITIALIZE or INIT
```

[Outline]

Initializes a memory card.

[Format]

```
INITIALIZE  <drive name>
or
INIT        <drive name>
```

[Description]

- The INITIALIZE command initializes a memory card. INITIALIZE may be abbreviated to INIT.
- The INITIALIZE command should be executed in the **BASIC mode** of the editor.

[Example]

```
INITIALIZE  "A:"
INIT        "B:"
```

[Note]

- Before initialization, check the contents of the file with the CAT Command or **Load** of the editor.
- New memory cards cannot be used until initialized. Note that if a memory card that stores files is initialized, all of the files will be deleted.

```
INTEGER
```

[Outline]

Declares integer-type variables or array variables.

[Format]

INTEGER <X> | <X> (numeric expression) { ,<X> | <X> (numeric expression) }
<X>: numeric variable

[Description]

- Numeric variables or array variables specified with the INTEGER statement will be treated as the integer type.
- The range of numbers that an integer-type variable can contain is the -2, same as that for integer constant, as follows:
  147, 483, 648 to +2, 147, 483, 647
- It is recommended that a variable that contains integers should be declared with the INTEGER statement. Integer-type variables are manipulated a little faster than real variables.
- If the INTEGER statement declares arrays, the specified size of the array variables are secured in the memory. If too large arrays are declared, the memory areas run out and the execution of the program is stopped because of the following error: (memory space full)
- If more than one subscript is specified, the array variable is declared as multi-dimensional. The number of the subscripts will be the number of dimensions.
  (Number of dimensions can be specified up to the allowance of memory capacity.)

[Example]

```
INTEGER A
INTEGER B, C
INTEGER D (20), E (30)
```

[Note]

- If integer-type variables are used in statement such as IF, the operation speed is improved.
- DIM statement should be used to declare real variables.
  (See the description on the DIM statement.)

[Program Example]

```
INTEGER A (20), B (5, 4),  I,  X, Y      : Integer-type declaration
FOR I = 1 TO 20                          : Sets I to 1 and loops until I exceeds 20.
  A (I) = I                              : Enters I to the array variable A.
NEXT I                                   : NEXT I
FOR X = 1 TO 5                           : Sets X to 1 and loops until X exceeds 5.
  FOR Y = 1 TO 4                         : Sets Y to 1 and loops until Y exceeds 4.
                                           Enters the array variable A to the array.
    B (X, Y) = A ((Y-1)*5 + X)           : Variable B.
  NEXT Y                                 : NEXT Y
NEXT X                                   : NEXT X
FOR X = 1 TO 5                           : Sets X to 1 and loops until X exceeds 5.
  FOR Y = 1 TO 4                         : Sets Y to 1 and loops until Y exceeds 4.
    CURSOR X*5, Y:PRINT B (X, Y)         : Outputs the contents of the array variable B.
  NEXT Y                                 : NEXT Y
NEXT X                                   : NEXT X
```

[Result]

```
1    2    3    4    5
6    7    8    9    10
11   12   13   14   15
16   17   18   19   20
```

```
INTERFACE CLEAR
```

[Outline]

Initializes interfaces of all devices connected to the GPIB of the CONTROLLER side of the main-unit rear panel.

[Format]

INTERFACE CLEAR

[Description]

- INTERFACE CLEAR enables the single line signal, IFC, of the GPIB to output (true mode) for 100μs.
- GPIB interfaces of all devices connected to the GPIB of the main-unit release the talker or listener modes.

[Example]

INTERFACE CLEAR

[Note]

Not function in slave mode.

[Sample Program]

```
INTERFACE CLEAR
OUTPUT 3; "CF1MZ"
```

## INPUT or INP

[Outline]

Enters the data entered from the keyboard or the main panel to a numeric variable or character-string variable.

[Format]

```
INPUT     [ <character-string constant>, ] <X> [ , | ; <X> ]
or
INP       [ <character-string constant>, ] <X> [ , | ; <X> ]

          <X>:  <numeric variable> | <character-string variable>
```

[Description]

- When the INPUT statement is executed, the program pauses to wait for key entry. The wait state continues until the Return key is pressed on the terminal or the unit key is pressed on the main panel. If the Return or unit key is pressed, the input is entered to the specified variable.
- If a character-string constant (prompt) is specified, the program displays the prompt on the screen before entering the wait state.
- The INPUT statement can manipulate both numeric and character-string variables. If alphabetic characters or symbols are entered together with numeric ones to be assigned to numeric variable, only the numeric characters are assigned and others omitted. If the character string to be entered to a numberic variable does not include any numbers. 0 is entered to the variable. If only the Return key is pressed in the wait state, assignment does not take place and the value of the variable does not change.

[Example]

```
INPUT A
INPUT "B =", B
INP C, D$
INP "name & No ?", NA$, N
```

[Note]

- A character string needs not be enclosed in quotations when it is entered.
- A character string may not be entered from the main panel.
  (Only numbers can be entered from the ten-key pad of the main panel.)

[Program]

```
INPUT "Center Freq. (MHz)?", CF
INP "Span Freq. (MHz) ?", SF
INP "No. ?", A$
OUTPUT 31; "CF", CF, "MZ"
OUTPUT 31; "SP", SF, "MZ"
PRINT "No.      :", A$
PRINT "CF       :", CF, "MHz"
PRINT "SF       :", SF, "MHz"
```

[Result]

```
Center Freq. (MHz)? 12
Span Freq. (MHz)? 30
No. ? 5
No. :     5
CF  :     12.0    MHz
SF  :     30.0    MHz
```

```
KEY OFF
```

[Outline]

Clears the display of user-defined menu.

[Format]

KEY OFF

[Description]

- Clears the display of user-defined menu.
- Interruption from the softkey cannot be generated after KEY OFF operation.

[Note]

- KEY OFF function can clear the display of user-defined menu only, however, cannot perform the interruption enable/disable.
  Refer to "ENABLE INTR" or "DISABLE INTR".

[Program]

```
ON KEY 1 LABEL "KEY-OFF" GOSUB *KOF
KEY ON
*LOOP
ENABLE INTR
GOTO *LOOP
*KOF
PRINT "KEY-OFF"
KEY OFF
RETURN
```

```
KEY ON
```

[Outline]

Displays the user-defined menu on the softkey menu.

[Format]

KEY ON

[Description]

- Displays the user-defined menu and enables to perform the interruption from softkey using the ON KEY instruction. Also, displays the letters defined by the ON KEY LABEL.

[Example]

KEY ON

[Note]

- KEY ON function can clear the display of user-defined menu only, however, cannot perform the interruption enable/disable.
  Refer to "ENABLE INTR" or "DISABLE INTR".

[Program]

```
ON KEY 1 LABEL "PROGRAM \n A" GOSUB *PRG-A: Defines KEY 1.
ON KEY 2 LABEL "PROGRAM \n B" GOSUB *PRG-B: Defines KEY 2.
KEY ON: ENABLE INTR                       : Displays LABEL on softmenu.
ENABLE INTR                               : Enables interruption.
*LOOP: Loops.
GOTO *LOOP:
*PRG-A                                    : Calls PRG-A if KEY 1 is pressed.
RETURN
*PRG-B                                    : Calls PRG-B if KEY 2 is pressed.
CALL "PRG-B"
RETURN
```

```
LIST
```

[Outline]

Program list is output.

[Format 1]

LIST [ <Label> [,] ]

[Format 2]

LIST [ <Output starting line> ], [, [ <Output ending line>] ]

[Description]

- The list display is carried out to the output (CRT or SIO1) which is designated by the PRINTER command.
- The program list of designated lines by the label is output.
- When the comma is designated following designation label, the list of the designated line by the label to the end of program is output.
- When the list is output in the editor of the BASIC mode, carry out LIST command in the BASIC mode after the BASIC program is transmitted (Compile & execution).
- LIST command is carried out in the BASIC mode then TAB and space are omitted to minimum. Moreover, the line of REM (!) cannot be displayed. (It is the same as GLIST, LLIST and others.)

[Example]

```
LIST
LIST *ABC
LIST 100
LIST 100,
LIST ,200
LIST 100, 200
```

```
LLIST
```

[Outline]

Outputs a program list of the device connected with the serial I/O port (RS-232).

[Format]

LLIST [ <label> [,] ]

[Description]

- Outputs the program list at the position specified with the label to the serial I/O port (RS-232).
- If a label followed by a comma is specified, the program will be listed on the screen from the label to the end of the program.
- To output the list, first transfer a BASIC program ( **compile & run** ), stop the program and then issue the LLIST command in the **BASIC mode**.
  An LLIST command can be placed in the program.

[Example]

LLIST

LLIST *ABC,

[Note]

- Line numbers may be specified instead of a label: a different format is required.

  LLIST [ <start line> ] [, [ <end line> ] ]

  ```
  Example:  LLIST 100
            LLIST 100,
            LLIST ,200
            LLIST 100, 200
  ```

[Program Example]

Execute **compile & run** of the editor.
Stop the program with Control C.
Execute LLIST in the **BASIC mode** of the editor.

[Result]

A program list is displayed on the screen.

```
LOCAL
```

[Outline]

Releases specified devices from the remote mode or to make the Remote Enable (REN) line false.

[Format]

LOCAL [ device address {, device address} ]
  Device address : 0 to 30

[Description]

- To execute the LOCAL command without the device address specification makes the Remote Enable (REN) line false and all devices on the GPIB become local.
  When the REN line is false, to set the devices on the GPIB using OUTPUT command is impossible. To set the REN line true, execute a REMOTE command. (Refer to the REMOTE command.)
- When a device address is specified after a LOCAL command, only the device of the specified address will be released from the remote mode.
  More than one device addresses can be specified, delimiting with commas (,).

[Example]

```
LOCAL

LOCAL 2

LOCAL 3, 4, 5
```

[Note]

Not function in slave mode.

[Sample Program]

```
REMOTE 11
WAIT 2000
LOCAL 11
```

[Result]

The device address 11 will be in the remote mode (The remote lamp will light if available) and two seconds after it will be in the local. (The remote lamp will turn off.)

```
LOCAL LOCKOUT
```

[Outline]

Locks devices connect to the GPIB, and prohibit to localize those devices from their panels.

[Format]

LOCAL LOCKOUT

[Description]

- When the devices on the GPIB is under the remote mode (controlled remotely by the main-unit controller), the panel keys of the devices are locked. Though it is impossible to set data from their panels, their local keys are unlocked. Therefore it is allowed to press each of them and localize the devices themselves to set data. If they were localized, various obstacles may arise and they cannot be controlled accurately.
  To avoid this case, execute LOCAL LOCKOUT command to lock the local keys of every devices on the GPIB, and prohibit to set from the panels of the devices.
- Execution of LOCAL LOCKOUT command send the GPIB universal command, Local Lockout (LLO).

[Example]

LOCAL LOCKOUT

[Note]

- To release the local lockout mode, execute a LOCAL command.
- Not function in slave mode.

[Sample Program]

```
REMOTE 11, 12
LOCAL LOCKOUT
LOCAL
```

[Result]

The device address 11 and 12 become invalid. To release this mode, execute a LOCAL command.

## LPRINT

[Outline]

Outputs numeric values and character strings to the device (printer, etc.) connected through the RS-232 interface.

[Format]

LPRINT [ <X> { [, | ; <X> ] } ]
<X>: numeric expression | character-string expression

[Description]

- Two or more numeric expressions or character-string expressions may be specified with each expression separated by a comma.
- The LPRINT statement always causes line feeding (moves the display or print position to the same column of the next line).

[Example]

```
S$ = "DEF"                  : ①
LPRINT "ABC"                : ②
LPRINT S$                   : ③
LPRINT "A =", A             : ④
LPRINT "CF", CFQ, "KHz"     : ⑤
LPRINT A + 100              : ⑥
```

① Enters "DEF" to the character string variable S$.
② Outputs "ABC" on the display without no line feed.
③ Outputs the character-string variable S$.
④ Outputs the character string and the numeric.
⑤ Outputs the character strings and the numeric variable CFQ as shown.
⑥ Adds 100 to the numeric variable and outputs the contents of A.

[Note]

- If the LPRINT statement ends with a semicolon (;), it causes line feeding.

[Program Example]

```
FOR I = 1 TO 10  :  Sets I to 1 and loops until I exceeds 10.
  LPRINT "I =", I :  Outputs I to the serial I/O port.
NEXT I           :  NEXT I
```

[Result]

```
I =   1.0
I =   2.0
I =   3.0
I =   4.0
I =   5.0
I =   6.0
I =   7.0
I =   8.0
I =   9.0
I =   10.0
```

```
LPRINT USING or LPRINT USE
```

[Outline]

Edits numeric values or character strings and outputs them through the RS-232 interface.

[Format]

LPRINT USING <image specification>; { , <X> }
or
LPRINT USE <image specifications> ; { , <X> }

<X>: numeric expression | character-string expression

[Description]

- The LPRINT USE (USING) statement first edits numeric values or character strings according to the image specifications. It then outputs the edited values to the serial I/O port (RS-232) in the ASCII notation.
- The image specifications are represented by character-string expressions, where each expression is separated by a comma (,).
  (Line feeding is automatically performed at the end of the specifications.)

[Image specification list]

| | |
|---|---|
| D .................................. | To output spaces on the rest of the specified fields. |
| Z .................................. | To insert 0s on the rest of the specified fields. |
| K .................................. | To output numeric just as they are. |
| S .................................. | To add a +/- symbols at all times. |
| M .................................. | To add a - symbols to negative values, and add a space to positive values. |
| . (Decimal point) ........... | To output a decimal point. |
| E .................................. | To output with the exponent form (e, sign, exponent). |
| H .................................. | To output numeric or character strings as they are with the European type decimal point. |
| R .................................. | To output the European type decimal point. |
| * .................................. | To output *s on the rest of the specified fields. |
| A .................................. | To output a character. |
| k .................................. | To character strings just as they are. |
| X .................................. | To output spaces. |
| Listeral .......................... | To write a literal in format specifications, put them in \"s. |
| B .................................. | Output numeric with the ASCII cord. |
| @ .................................. | To feed lines. |
| + .................................. | To move the outputting positions to the top of the same line. |
| - .................................. | To move the outputting positions to the next line. |
| # .................................. | To not feed lines at the last position. |
| n .................................. | To output with accuracy of n decimals. If specify n to character strings, output value will be the length of the specified character strings. |

[Example]

```
LPRINT USING "DDD.DD"; 1.2
LPRINT USE "ZZZ.ZZ"  ; 1.2
```

[Note]

- A carriage return is not automatically caused by the LPRINT USING statement. The display position just moves to the same column of the next line. A new-line character should be included in the statement, as follows:

  LPRINT USING "K, k"; 123, "\r"

- If number of the expressions for the image specifications is greater than that of the parameters, the program is stopped by the following errors: (Unmatched IMAGE-spec in USINT)

[Program Example]

```
LPRINT USING "DDD.DD, k"; 1.2 "\r"
LPRINT USE "ZZZ.ZZ, k"; 1.2, "\r"
LPRINT USE "***.ZZ, k"; 1.2, "\r"
A$ = "K, 5X, B, DDDRDD, k"
LPRINT USE A$; 2.34, 65, 1.2, "\r"
```

[Result]

```
1.20
001.20
**1.20
2.34       A     1, 20
```

## MENU

[Outline]

Arbitrary character string is displayed to the soft menu and it is waited until pressing the soft menu.

[Format]

```
MENU (S1, S2, S3, S4, S5, S6)
```

[Description]

- Arbitrary character string is displayed to the soft menu and it is waited until pressing the soft menu.
- Pressed number of soft key (1 to 6) is returned by the numeric value after pressing the soft menu.

[Example]

```
MENU ("KEY1", "KEY2", "KEY3", "KEY4", "KEY5", "KEY6")
MENU ("KEY1\nSF1", "KEY2", "KEY3", "KEY4", "KEY5", "KEY6")
```

[Note]

MENU is only displayed for the user definition menu and, the interrupt and inhibit cannot be carried out.
Refer to ENABLE INTR and DISABLE INTR.

[Program Example]

```
ENABLE INTR
*LOOP
 A = MENU ("KEY1", "KEY2", "KEY3", "KEY4", "KEY5", "KEY6")
 PRINT A
 *LOOP
```

```
OFF END
```

[Outline]

Releases a definition of the branch destination given by the ON END instruction.

[Format]

OFF END <# file descriptor>

[Description]

The OFF END instruction releases the definition of the destination (statement) of a branch caused by the EOF of <# file descriptor>. The definition has been given by the ON END instruction.

[Example]

OFF END # FD

[Note]

- After the definition of a branch destination has been released with this instruction, an error occurs and the execution of the program stops in the middle if an EOF is read with the ENTER # instruction.
- The branch destination can be defined with ON END.

```
OFF ERROR
```

[Outline]

Releases a definition of the branch destination given by the ON ERROR instruction.

[Format]

OFF ERROR

[Description]

The OFF ERROR instruction releases the definition of the branch destination (the statement to branch to) that has been given by the ON ERROR instruction.

[Example]

OFF ERROR

[Note]

- If the program causes an error after the definition of a branch destination has been released with this instruction, it stops immediately.
- The branch destination can be defined with ON ERROR.

```
OFF KEY
```

[Outline]

Cancels the branch destination defined using the ON KEY.

[Format]

OFF KEY (key numbers)

Key numbers (numeric expression)

| | |
|---|---|
| 1 to 6: | Softkey |
| 10: | Step key ↑ |
| 11: | Step key ↓ |
| 12: | Data knob ⟳ (clockwise) |
| 13: | Data knob ⟲ (counterclockwise) |

[Explanation]

- Cancels the branch destination defined in the key numbers using the ON KEY.
- Cancels each definition individually for the destination by the key numbers.

[Example]

```
OFF KEY 1
OFF KEY 2
OFF KEY 10
OFF KEY 11
OFF KEY KK
OFF KEY BB
```

[Note]

- After canceling the definition using this instruction, does not branch if softkey, step key, or rotary encoder is used for inputting.
- For definition of destination, conduct it by "ON KEY Instruction".

[Program]

```
ON KEY 1 GOSUB *L1
ON KEY 2 GOSUB *L2
ON KEY 3 GOSUB *L3
ENABLE INTR
*LOOP
GOTO *LOOP
*L1
PRINT "OFF KEY 1"
OFF KEY 1
RETURN
*L2
PRINT "OFF KEY 2"
OFF KEY 2
RETURN
*L3
PRINT "OFF KEY 3"
OFF KEY 3
RETURN
```

[Result]

IF the interruption is generated once, 2nd interruption cannot be generated.

```
OFF SRQ/ISRQ
```

[Outline]

Releases a definition of the branch destination given by the ON SRQ or ON ISRQ instruction.

[Format]

OFF SRQ

OFF ISRQ

[Description]

The OFF SRQ or OFF ISRQ instruction releases the definition of the branch destination (the statement to branch to) that has been given by the ON SRQ or ON ISRQ instruction.

[Example]

```
OFF SRQ
OFF ISRQ
```

[Note]

The branch destination can be defined with ON SRQ or ON ISRQ.

## ON END - GOTO/GOSUB

[Outline]

Defines the statement to which control will branch when an EOF occurs.

[Format]

ON END <# file desciptor> GOTO <label>

ON END <# file desciptor> GOSUB <label>

[Description]

- This instruction defines the statement to which control will branch when the end of file (EOF) is detected in a file being read with an ENTER # instruction.
- If data is serially read by the ON END statement where an operation after an EOF detection has not been defined, the following error message appears and the execution stops when an EOF read. (end of <file name> file)

[Example]

```
ON END # FD GOTO *EOF
ON END # FD GOSUB *EOF
```

[Note]

- A branch will be taken when an EOF is read. If control has branched to a subroutine with GOSUB statement, it returns to the instruction following the ENTER instruction by which the EOF was read.
- A ON END - GOTO/GOSUB statement causes control to branch to the specified statement in regardless of ENABLE INTR or DISABLE INTR instructions.
- To release the statement to branch to, execute OFF END.
- For writing and reading data, see OUTPUT # and ENTER.

[Program Example]

```
OPEN "FFF" FOR OUTPUT AS # FD  : Opens "FFF" for the file to which data is to be written.
FOR I = 100 TO 200             : Loops I from 100 to 200.
  OUTPUT # FD; I               : Saves the data of I.
NEXT I                         : NEXT I
CLOSE # FD
!
ON END # FR GOTO *LA          : Defines the destination of the branch with an EOF.
OPEN "FFF" FOR INPUT AS # FR  : Opens "FFF" for the file from which data is to be read.
*LOOP
 ENTER # FR; N                : Loads data from the file to the load and enters it to N.
 PRINT N                      : Displays the contents of N on the screen.
 GOTO *LOOP                   : Jumps to the statement labeled "*LOOP".
!
*LA
 CLOSE # FR                   : Closes the file.
 PRINT "EOF"                  : Displays "EOF" on the screen.
 STOP                         : Program end
```

[Result]

```
100.0
101.0
102.0
  •
  •  (Omitted)
  •
199.0
200.0
EOF
```

## ON ERROR - GOTO/GOSUB

[Outline]

Defines the statement to which control will branch when an error occurs during execution of a BASIC program.

[Format]

ON ERROR GOTO <label>

ON ERROR GOSUB <label>

[Description]

- Defines the destination to branch to when an error occurs during execution of a BASIC program.
- If this instruction has been executed, the execution of the program transfers to the position specified with <label> when an error occurs.
- This instruction can be used for error operations while executing a built-in function.

[Example]

```
ON ERROR GOTO *ERR
ON ERROR GOSUB *ERR
```

[Note]

- A branch will be taken when an error occurs. If control has branched to a subroutine with GOSUB statement, it returns to the instruction following the instruction where the error occurred.
- A ON ERROR - GOTO/GOSUB instruction causes control to branch to the specified statement in regardless of ENABLE INTR or DISABLE INTR instructions.
- To release the statement to branch to, execute OFF ERROR.

[Program Example]

| | |
|---|---|
| ON ERROR GOTO *ERR | : Defines the destination of the branch to be caused by an error input. |
| PRINT "START" | : Displays "START" on the screen. |
| PRINT 1/0 | : Divides 1 by zero. |
| PRINT "END" | : Outputs "END". |
| STOP | : Program end |
| *ERR | : |
| PRINT "<<ERROR>>" | : Outputs "<<ERROR>>" |
| PRINT ">>", ERRM$ (0) | : Outputs an error message. |

[Result]

```
START
<<ERROR>>
  >>        40:  0 divide
```

```
ON KEY - GOTO/GOSUB
```

[Outline]

Defines the statement to which control will branch when receiving the interruption by key of the main-body panel.

[Format]

ON KEY <Key number> GOTO <label>
ON KEY <Key number> GOSUB <label>
ON KEY <Key number> LABEL <string expression> GOTO <label>
ON KEY <Key number> LABEL <string expression> GOSUB <label>

Key numbers (numeric expression)

| | |
|---|---|
| 1 to 6: | Softkey |
| 10: | Step key ↑ |
| 11: | Step key ↓ |
| 12: | Data knob ↻ (clockwise) |
| 13: | Data knob ↺ (counterclockwise) |

[Description]

- The key numbers (1 to 6) define the destination to branch the interruption (SF1 to SF7) corresponding to the softkey of the main-body panel. The interruption from the softkey is available only when the user-defined menu of the softmenu is displayed. (Refer to KEY ON.)
- In case of the key numbers (1 to 6), displays letters specified in the softkey menu using LABEL.
- The key numbers (10 to 13) define the destination to branch the interruption by the data knob or step keys of the main-body panel.
- Refer to ENABLE INTR/DISABLE INTR for the interruption enable/disable.

[Example]

```
ON KEY 1 GOTO *K3
ON KEY 2 GOSUB *K4
ON KEY XX LABEL "ON" GOTO *K1
ON KEY BB LABEL "OFF" GOSUB *K2
ON KEY XX LABEL ST$ GOTO *K1
ON KEY BB LABEL ST$& "ON" GOSUB *K2

ON KEY 10 GOSUB *STUP
ON KEY 11 GOTO *STDWN
ON KEY 12 GOSUB *ENR
ON KEY 13 GOTO *ENL
```

[Note]

- A branch will be taken after the instruction being executed on the occurrence of an interruption is executed. If control has branched to a subroutine with GOSUB statement, it returns to the instruction following the instruction being executed when the interruption generated.
- To cancel the statement to branch to, execute the OFF KEY instruction. (Refer to OFF KEY.)
- The softkey (SF1 to SF7) for interruption cannot be used when the user-defined menu is not displayed. To display the user-defined key, execute the KEY ON instruction.

[Program Example]

```
ON KEY 1 LABEL "PROGRAM \n A" GOSUB *PRG-A: Defines the KEY 1.
ON KEY 2 LABEL "PROGRAM \n B" GOSUB *PRG-B: Defines the KEY 2.
KEY ON: ENABLE INTR                       : Displays LABEL on the softmenu.
ENABLE INTR                               : Enables interruption.
*LOOP: Loops.
GOTO *LOOP:
*PRG-A                                    : Calls PRG-A if KEY 1 is pressed.
RETURN
*PRG-B                                    : Calls PRG-B if KEY 2 is pressed.
CALL "PRG-B"
RETURN
```

```
ON SRQ/ISRQ - GOTO/GOSUB
```

[Outline]

Defines the statement to which control will branch when the GPIB controller receives a service request (SRQ).

[Format]

ON SRQ GOTO <label>
ON SRQ GOSUB <label>
(Defines the destination of the branch to be caused by an SRQ from an external GPIB.)

ON ISRQ GOTO <label>
ON ISRQ GOSUB <label>
(Defines the destination of the branch to be caused by an SRQ from the measurement section of the main body.)

[Description]

- Defines the statement to which control will branch when an interrupts is caused by an SRQ.
- ON SRQ defines the destination of the branch to be caused by an SRQ from an external GPIB device.
- ON ISRQ defines the destination of the branch to be caused by an SRQ from the measurement section of the main body.

[Example]

```
ON SRQ GOTO *SS
ON SRQ GOSUB *SS
ON ISRQ GOTO *IS
ON ISRQ GOSUB *IS
```

[Note]

- When an ON ISRQ instruction is used, an SRQ interrupt should be caused in the measurement section. To cause an SRQ interrupt, execute the OUTPUT 31 instruction with the following codes:

| GPIB code | Function |
|---|---|
| S0 | The measurement section transmits an SRQ interrupt to the controller. |
| S1 | The measurement section does not transmit an SRQ interrupt to the controller. |
| S2 | Clears the status bytes. |

- A branch will be taken after the instruction being executed on the occurrence of an interrupt is executed. If control has branched to a subroutine with GOSUB statement, it returns to the instruction following the instruction being executed when the interrupt occurred.
- For enabling or disabling reception of interrupts, see ENABLE INTR and DISABLE INTR.
- To release the statement to branch to, execute OFF SRQ or OFF ISRQ.

[Program Example]

Example 1:

```
TGT = 7
ON SRQ GOSUB *SS
ENABLE INTR
*MAINLOOP
 GOSUB *BEEP
 GOTO *MAINLOOP

*BEEP
 BUZZER 440, 20
 WAIT 200
 RETURN

*SS
 DISABLE INTR
 S = SPOLL (TGT)
 PRINT "SPOLL", S
 ENABLE INTR
 RETURN
```

Example 2:

```
INTEGER S
ON ISRQ GOTO *SS
*ST
 OUTPUT 31; "SO SI"
 ENABLE INTR
 *MAINLOOP
 GOTO *MAINLOOP

*SS
 S = SPOLL (31)
 IF S BAND 4 THEN
    PRINT "MAX :", MAX (0, 700, 0)
    PRINT "MIN :", MIN (0, 700, 0)
 GOTO *ST
```

[Result]

Example 1: A branch is caused by the SRQ from the external GPIB and the result of serial polling is output.

Example 2: Every time the measurement section of the main body completes a sweep, the maximum and minimum levels are obtained.

```
OPEN #
```

[Outline]

Assings a file to a file descriptor and opens the file in the specified operation mode.

[Format]

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

<operation mode>

| | | |
|---|---|---|
| OUTPUT | : | Write |
| INPUT | : | Read |

<type>

| | | |
|---|---|---|
| BINARY | : | Binary |
| TEXT | : | ASCII code |
| ASCII | : | ASCII code with a header |

[Description]

- The OPEN # statement assigns a file descriptor to a file and opens it in the specified mode, thus allowing the program to identify the file.
- Allowable operation modes are OUTPUT and INPUT.
  The former is used to write data in the file and the latter to read data from the file.
- file descriptor
  Actually, writing and reading data to and from files are performed by the ENTER and OUTPUT statements, respectively.
  These statements identify the object file with the file descriptor.
  Each file descriptor begins with a #, followed by alphanumeric characters.
- Type
  There are the following three types: BINARY, TEXT, and ASCII.
  (If the type is not specified, BINARY is assumed.)
  Refer to OUTPUT #.

[Example]

```
OPEN "FILE1" FOR OUTPUT AS # FD
OPEN "FILE2" FOR OUTPUT AS # FD; BINARY
OPEN "FILE3" FOR OUTPUT AS # FF; TEXT
OPEN "FILE4" FOR OUTPUT AS # FA; ASCII

OPEN "FILE1" FOR INPUT AS # FD
OPEN "FILE2" FOR INPUT AS # FD; BINARY
OPEN "FILE3" FOR INPUT AS # FF; TEXT
OPEN "FILE4" FOR INPUT AS # FA; ASCII
```

[Note]

- If the file descriptor being opened has already been assigned to the other file, the following error message appears and the program execution is stopped.
  ("file name" file is already exist.)
  ("file name" file is already with another path.)
- All of the files opened in a program should be closed in the end of the program.
- One file may not be opened with more than one file descriptor.
- Up to three files can be opened simultaneously.

[Program Example]

```
OPEN "B" FOR OUTPUT AS # FA; BINARY
FOR I = 1 TO 10
  OUTPUT # FA; I
NEXT I
CLOSE # FA

OPEN "BB" FOR OUTPUT AS # FB; TEXT
FOR I = 1 TO 10
  OUTPUT # FB; I
NEXT I
CLOSE # FB

OPEN "BBB" FOR OUTPUT AS # FC; ASCII
FOR I = 1 TO 10
  OUTPUT # FC; I
NEXT I
CLOSE # FC
```

```
OUTPUT #
```

[Outline]

Outputs (writes) data to the file that is assigned to the specified file descriptor.

[Format]

OUTPUT <# file descriptor> ; <X> [ , <X> ]
    <X>: print item (numeric expression, character string expression)

[Description]

According to the type specified with the OPEN statement, the data is written in the following format.

BINARY type

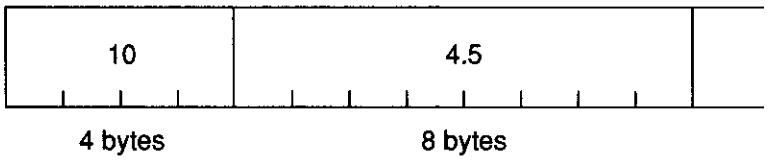
Data is written in the same type as that of the internal expression.

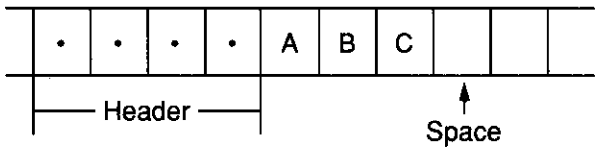
The data length differs according to the type, as follows:

| | | |
|---|---|---|
| Integer type | : | 4 bytes |
| Real type | : | 8 bytes |
| Character string | : | A four-byte header plus a character string of several bytes. (If the number of characters are odd, a space will follow the character string.) |

Example)

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



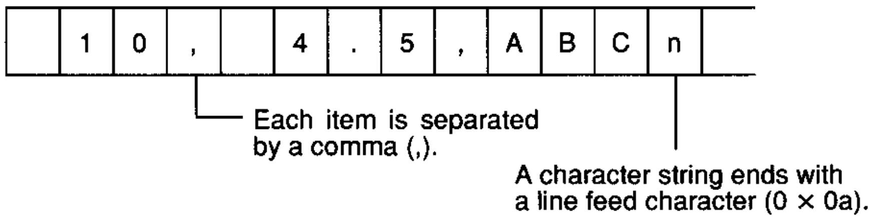


TEXT type

Data is converted to the ASCII codes and output. A numeric data object is preceded by a space or a sign. A character string is terminated by a line feed character (0 × 0a).

Example)

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

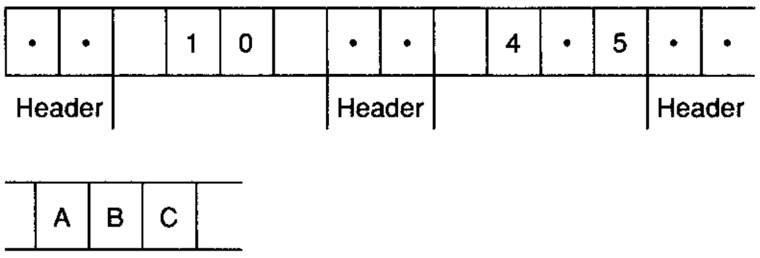

1
0
,
4
.
5
,
A
B
C
n
Each item is separated by a comma (,).
A character string ends with a line feed character (0 × 0a).

ASCII type

Data is converted to the ASCII codes and output. A numeric data object is preceded by a space or a sign.
A 2-byte header leads each item. If the number of characters in a data object is odd, a space will follow the object.

Example)

```
        OPEN "FILE" FOR OUTPUT AS  # FD; ASCII
        OUTPUT  # FD; 10, 4.5 "ABC"
```



[Example]

```
OUTPUT  # FD; "ABC"
OUTPUT  # FD; A, B, C
```

[Note]

- Specify a file descriptor that has been opened with the OPEN statement.
- A file descriptor should be assigned to the file to which data is written, when it is opened by OPEN #.
  The assigned file descriptor will always be used in stead of the name of the file when it is manipulated.
- A header is a separator of items and has the length of a data object.
- The following error occurs if the OUTPUT statement is used for the file that has not been opened with the OPEN statement:
  (file NOT open)

[Program Example]

```
OPEN "A" FOR OUTPUT AS # FA; BINARY
INTEGER S
FOR I = 1 TO 10
  S = I
  OUTPUT # FA; I, S
NEXT I
CLOSE # FA

OPEN "AA" FOR OUTPUT AS # FB; TEXT
INTEGER S
FOR I = 1 TO 10
  S = I
  OUTPUT # FB; I, S
NEXT I
CLOSE # FB

OPEN "AAA" FOR OUTPUT AS # FC; ASCII
INTEGER S
FOR I = 1 TO 10
  S = I
  OUTPUT # FC; I, S
NEXT I
CLOSE # FC
```

```
OUTPUT or OUT
```

[Outline]

Sends data to the GPIB or measuring device of the main-unit.

[Format]

OUTPUT < A {, A} I 31 >; < X {,X} >
or
OUT < A {, A} I 31 >; < X {,X} >

| | | |
|---|---|---|
| A: | Device address 0 to 30 : | Outputs data to the devices on the GPIB |
| | 31 : | Outputs data to the measuring device of the main-unit |
| <X>: | numeric expression I character expression | |

[Description]

- OUTPUT or OUT send numeric or character strings in the ASCII code to the devices specified with the device addresses (0 to 30).*1

  More than one device addresses can be set by delimiting commas (,).
  Mixed setting of numeric and character expressions is available by delimiting with comma (,).
  When the REN (Remote Enable) line is true, to execute the OUTPUT command (0 to 30) causes the devices specified with the device addresses to be automatically in the remote mode.

- To specify 31 causes send data to the measuring devices of the main-unit.*1
  (To send the talker/listener GPIB codes to the measuring devices enables to set them as the external controller.)

*1: Refer to section 7 of the manuals (separate volumes) of ULIS for the list of the GPIB codes.

[Example]

```
OUTPUT 3; "CF3MZ"
OUTPUT 3, 4, 5; "SP", S, "MZ"
OUTPUT 31; "CF", CF, "MZ"
```

[Note]

- No addresses can be specified other than 0 to 30 by delimiting with comma (,). (Multiple specification is available only between 0 and 30.)
- If the device addresses are missed or no device connects to the specified address, this command will be ignored and the program will proceeds to the next step.
- Refer to "5. MASTER/SLAVE MODE" when using in slave mode.

[Sample Program]

```
DIM S$ [26], B$ [20]
OUTPUT 5; "CF?"
ENTER 5; C
ENTER 5; S$
PRINT C, S$

CF = 1.23
A$ = "CF?"
OUTPUT 31; "CF", CF, "MZ"
OUTPUT 31; A$
ENTER 31; B$
PRINT B$

INTEGER S
ENTER 32; S
IF S BAND 4 THEN OUTPUT 32; 128
```

## PANEL

[Outline]

Enables/Disables the operation using the main-body panel key.

[Format]

PANEL <numeric expression>
Numeric expression: 0 to 1

[Description]

| | Panel condition | |
|---|---|---|
| 0 | Panel unlock: | Permits the normal operation using the main-body panel. (Enables the operation of center, span, etc.) |
| 1 | Panel lock: | Inhibits the operation using the main-body panel. (Disables the operation.) |

[Example]

```
PANEL 0
PANEL 1
PANEL X
```

[Program Example]

```
ON KEY 1 LABEL "PANEL \n UNLOCK" GOSUB *PU
ON KEY 2 LABEL "PANEL \n LOCK" GOSUB *PL
KEY ON
ENABLE INTR
SCREEN 4
*LL
GOTO *LL
*PU
PANEL 0
PRINT "PLEASE KEY OPERATION"
RETURN
*PL
PANEL 1
PJRINT "PANEL LOCK"
RETURN
```

```
PAUSE
```

[Outline]

Stops executing the program temporarily.

[Format]

PAUSE

[Description]

The execution of the program pauses where the PAUSE statement is placed.
The execution will resumes when the CONT command is entered or the CONT of the main-body panel is pressed.
(The CONT command should be entered in the BASIC mode.)

[Example]

PAUSE

[Program Example]

```
FOR I = 1 TO 5          : Initializes the counter I to zero.
 PRINT I                : Outputs the variable I.
 PRINT "Hit CONT key"   : Outputs the character constant.
 PAUSE                  : PAUSE
NEXT I                  : NEXT
```

[Result]

The execution always pauses after the content of the variable I have been printed.

```
PRINTF or PRF
```

[Outline]

Outputs numeric and character strings to main display.

[Format]

PRINTF <character expression> [ , <x> ]
or
PRF <character expression> [ , <x> ]

<x>: numeric expression I character expression

[Description]

- If the following format is specified in the character expression, it is allowed to edit and output the data of the arguments. (The editing specification will start with %.)

| | |
|---|---|
| Minus symbol | : To cause the edited parameter to be aligned to the left side of the field (output area). |
| Period | : To delimit the two type of numeric strings, representing the length and accuracy of the fields. |
| 0 | : To fill the rest of fields with zero suppression. |

- Editing characters and meanings

| | |
|---|---|
| d | : To change parameters into the decimal numeral. |
| o | : To change parameters into the octal numeral without symbols. (No 0 at the head) |
| x | : To change parameters into the hexadecimal numeral without symbol. (No 0x at the head) |
| s | : Character strings |
| e | : To accept as real numbers and change the decimal numeral, with the form of [-] m. nnnnnnE [±] xx (The length of n strings determined by the accuracy. The default is six) |
| f | : To accept as real numbers and change the decimal numeral, with the from of [-] mmm. nnnn. (The length of n strings determined by the accuracy. The default is six) |

- The characters after % are output themselves, except when they are editing characters. Therefore to % will be output with the %% form.

[Sample Program]

| | |
|---|---|
| PRINTF "C = %d", C | : Output the variable C with the decimal numeral. |
| PRINTF "C = %5d", C | : Output the variable C in five figures with the decimal numeral. (Right justification) |
| PRINTF "C = %-5d", C | : Output the variable C in five figures with the decimal numeral. (Left justification) |
| PRINTF "C = %05d", C | : Output the variable C in five figures with the decimal numeral. (Right justification: suppress with zero) |
| PRINTF "H = %x", H | : Output the variable H with the hexadecimal numeral. |
| PRINTF "H = %4x", H | : Output the variable H in four figures with the hexadecimal numeral. (Right justification) |
| PRINTF "H = %-4x", H | : Output the variable H in four figures with the hexadecimal numeral. (Left justification) |
| PRINTF "H = %04x", H | : Output the variable H in four figures with the hexadecimal numeral. (Right justification: suppress with zeros) |
| PRINTF "S$ = %s", S$ | : Output character strings. |
| PRINTF "S$ = %8s", S$ | : Output 8 string characters from right-justify. |
| PRINTF "S$ = %-8s", S$ | : Output 8 string characters from left-justify. |
| PRINTF "%20.10s", S$ | : Output character strings justifying to the right most column in a 20 characters length field. |
| PRINTF "%-20.10s", S$ | : Output ten character strings justifying to the left most column in a 20 characters length field. |
| PRINTF "F = %f", F | : Output the variable F in a real number with decimal numeral. |
| PRINTF "F = %8.2f", F | : Output the variable F in a eight figures to the second decimal including a decimal point. |
| PRINTF "F = %08.2f", F | : Output the variable F in eight figures to the second decimal including a decimal point. (Suppress with zeros) |

[Note]

- Since the output using PRINTF statements does not start a new line, insert line feed cords into the PRINTF statements or add PRINT statements after PRINTF statements.

```
PRINTF "A = %d\n\r", A
                ↑
         Line feed cords
```

- If lack of the PRINTF editing characters or parameters, or the type specifications errors are detected, invalid output may be resulted.

[Example]

```
PRINTF "%d * %4d = %05d", 12, 34, 12*34
PRINT
PRINTF ":%-10d:%10d:", 123, 456
PRINT
A$ = "ABCD"
PRINTF ":%8s:%-8s:%8.2s:%-8.2s:", A$, A$, A$, A$
PRINT
PRINTF "%f+%8.3f-%06.3f = %e", 1.23, 4.56, 7.89, 1.23+4.56-7.89
```

[Result]

```
12 *    34 = 00408
:123       :       456:
:    ABCD:ABCD    :      AB:AB      :
1.230000+    4.560 - 07.890 = -2.100000e+00
```

```
PRINT or ?
```

[Outline]

Outputs numeric or character strings to the main display.

[Format]

```
PRINT [ <X> { [, | ; <X> ] } ] [ <, | ; > ]
or
  ?   [ <X> { [, | ; <X> ] } ] [ <, | ; > ]
```

<X>: Numeric expression | Character expression

[Description]

- The numeric or character expression can be delimited with a semicolon (;) or comma (,) to specify more than one. (Semicolons output sequentially, and commas output at a tab's intervals)
- If a semicolon is put at the PRINT statement's end, the line will not start a new line. Therefore when the next PRINT statement is executed, the output will follows the previous output.

[Example]

```
S$ = "DEF"                : ①
PRINT "ABC" ;             : ②
PRINT S$                  : ③
PRINT "A =", A            : ④
PRINT "CF", CFQ, "KHz"    : ⑤
PRINT A + 100             : ⑥
```

① Enters "DEF" to the character string variable S$.
② Outputs "ABC" on the display without no line feed.
③ Outputs the character string variable S$.
④ Outputs the character strings and the numeric.
⑤ Outputs the character strings and the numeric.
⑥ Adds 100 to the numeric variable and output.

[Sample Program]

```
FOR I = 1 TO 10
  PRINT "I =";
  PRINT I
NEXT I
```

[Result]

```
I =  1.0
I =  2.0
I =  3.0
I =  4.0
I =  5.0
I =  6.0
I =  7.0
I =  8.0
I =  9.0
I =  10.0
```

```
PRINT USING or PRINT USE
```

[Outline]

Edits numeric or character strings and output to the main display.

[Format]

PRINT USING <image specification> ; { , <x> }
or
PRINT USE <image specification> ; { , <x> }

<x> : numeric expression I character expression

[Description]

- To specify the image specification, represent with character strings and delimit with comma (,). (The line will be fed automatically.)

[Image specification list]

| | |
|---|---|
| D | To output spaces on the rest of the specified fields. |
| Z | To insert 0s on the rest of the specified fields. |
| K | To output numeric just as they are. |
| S | To add a +/- symbols at all times. |
| M | To add a - symbols to negative values, and add a space to positive values. |
| . (Decimal point) | To output a decimal point. |
| E | To output with the exponent form (e, sign, exponent). |
| H | To output numeric or character strings as they are with the European type decimal point. |
| R | To output the European type decimal point. |
| * | To output *s on the rest of the specified fields. |
| A | To output a character. |
| k | To character strings just as they are. |
| X | To output spaces. |
| Listeral | To write a literal in format specifications, put them in \"s. |
| B | Output numeric with the ASCII cord. |
| @ | To feed lines. |
| + | To move the outputting positions to the top of the same line. |
| - | To move the outputting positions to the next line. |
| # | To not feed lines at the last position. |
| n | To output with accuracy of n decimals. If specify n to character strings, output value will be the length of the specified character strings. |

[Example]

```
PRINT USING "DDD.DD" ; 1.2
PRINT USE "ZZZ.ZZ" ; 1.2
```

[Note]

- The strings outputted with PRINT USING will not be returned. Insert return symbols in the parameters of PRINT USING.

```
    PRINT USING "K, k"; 123, "\ n"
```

[Sample Program]

```
PRINT USING "DDD.DD, k"; 1.2, "\r"
PRINT USE "ZZZ.ZZ, k"; 1.2, "\r"
PRINT USE "***.ZZ, k"; 2.4, "\r"
A$ = "K, 5X, B, DDDRDD, k"
PRINT USE A$; 2.34, 65, 1.2, "\r"
```

[Result]

```
1.20
001.20
**2.40
2.34      A    1, 20
```

```
PRINTER
```

[Outline]

The output destination of the measurement result and calculation result is designated.

[Format]

PRINTER <GPIB address - CRT - SIO1>

- GPIB address 0 to 30.
- CRT (It is output to the screen.)
- SIO1 (It is output to the RS-232 port)

[Description]

- GPIB address
  The output destination of the command of GPRINT, GLIST GLISTN and others is designated to the GPIB address. (Carry out PRINTER command without exception before GPRINT, GLIST, GLISTN and others are carried out.)
- CRT
  The output destination of the PRINT and others command is designated to the screen. (Carry out PRINTER command without exception before PRINT, CAT and others are carried out.)
- SIO1
  The output destination of the PRINT and others command is designated to the RS-232. (Carry out PRINTER command without exception before PRINT, CAT and others are carried out.) Default is RS-232.

[Example]

```
PRINTER 1
PRINTER CRT
PRINTER SIO1
```

## READ DATA/RESTORE

[Outline]

Enters a constant in a DATA statements into a variable.

[Format]

READ <X> {, <X> }
<X>: numeric variable I character variable

DATA <Y> {, <Y> }
<Y>: numeric variable I character variable

RESTORE [ <label> ]

[Description]

- READ DATA enters numeric or character strings, determined in a DATA statement, into variables specified with a READ statement.
- The first READ generally retrieves DATA statement from the start to the end, and the first value of the DATA statement will be entered into a variable. Then the corresponding DATA statement constants will be retrieved and entered one by one.
- The specified character strings in a DATA statement must be put in double quotations (").
- A RESTORE command specifies the start position of a DATA statement to be read by a READ statement. (If a label is specified, the DATA statement after the label will be retrieved. If no label is specified, the DATA statement will be retrieved from the start of a program.)

[Example]

```
READ A
READ A, B$

DATA 1, 2, 3, 4
DATA “ABC”, 5, “DEF”, 6

RESTORE
RESTORE *LA
```

[Note]

- If the number of constants specified in a DATA statement are less than the corresponding variables of a READ statement, an error will be caused. (Unmatched DATA's values and READ variable)
- When unmatched entering variable style and DATA's specified constant variable, error occurs.
  (Invalid type in get data)

[Sample Program]

```
RESTORE *LA : Specifies to read a DATA statement after the label *LA.
READ A, B$  : Reads numeric value A and character value B.
PRINT A, B$ : Outputs the contents of the variable to the display.
RESTORE *LB : Specifies to read a DATA statement after the label *LB.
READ A, B$  : Reads numeric value A and character value B.
PRINT A, B$ : Outputs the contents of the variable to the display.
STOP        : Termination of the program
!
*LB         : Label *LB
 DATA 2, "B" : Data
*LA         : Label *LA
 DATA 1, "A" : Data
```

[Result]

```
1.0     A
2.0     B
```

```
REM or !
```

[Outline]

Writes comments into a program.

[Format]

```
REM  [ <character strings> ]
or
!    [ <character strings> ]
```

[Description]

- REM or ! are used to add comments to a program.
- Since REM is a nonexcutable command, all the characters after a REM statement will be ignored. (Therefore all the characters, numbers and symbols are available.)

[Example]

```
REM  <<<remark>>>
!
!       **** ADVANTEST ****
```

[Note]

Since all the statements after a REM statement are assumed as comments, multi-statements with a colon (:) can not function after a REM statement.

[Sample Program]

```
! *************************  : Comment
! ** <<PROGRAM>> **          : Comment
! *************************  : Comment
REM                          : Comment
REM        ADVANTEST         : Comment
!                            : Comment
PRINT "TEST"                 : Outputs character strings.
STOP                         : Termination of the program
```

[Result]

```
TEST
```

## REMOTE

[Outline]

Sets a specified in the remote mode or to make the Remote Enable (REN) line true.

[Format]

```
REMOTE [ device address { , device address } ]
    Device address:  0 to 30
```

[Description]

- To execute the REMOTE command without the device address specification makes the Remote Enable (REN) line true and all devices on the GPIB remote-controllable.
  To make the REN line false, execute LOCAL.
- When a device address is specified after a REMOTE command, only the device of the specified address will be in remote mode.
  More than one device addresses can be specified, delimiting with commas (,).
- To execute the OUTPUT or SEND command causes the remote mode without a execution of a REMOTE command.

[Example]

```
REMOTE

REMOTE 6

REMOTE 6, 7, 10
```

[Note]

Not function in slave mode.

[Sample Program]

```
REMOTE 11
WAIT 2000
LOCAL 11
```

[Result]

The device address 11 will be in the remote mode (The remote lamp will light if available) and two seconds after it will be in the local. (The remote lamp will turn off.)

```
RENAME
```

[Outline]

The file in the memory card is changed.

[Format]

RENAME < [A: | B:] FILE1 >, < [A: | B:] FILE2 >

[Description]

- The file name FILE1 in the memory card is changed to FILE2.
- The drive (A or B) can be designated but designate the same drive. If designation drive is omitted then the default becomes an active drive.

[Example]

```
COPY "A:PROG1.BAS", "A:PROG2.BAS"
COPY "DATA1.DAT", "DATA2.DAT"
```

[Note]

- Before changing, carry out CAT command or Load of editor and ensure the file to be changed.
- The file name is limited within eight characters and the extension is limited within three characters.
- Copy cannot be carried out when write protect of the memory card is on.

```
REQUEST
```

[Outline]

The REQUEST command outputs SRQ (service request) to an external controller connected to the GPIB port (in the upper section of the rear panel).

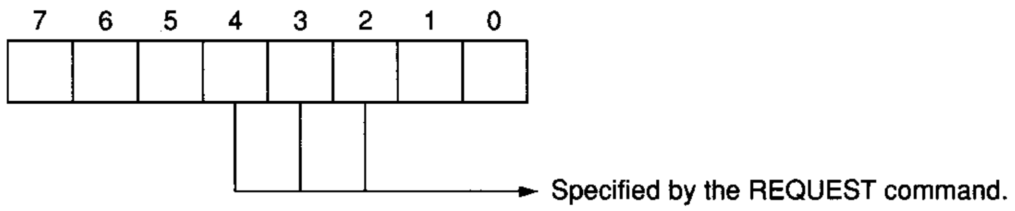
[Format]

REQUEST n (n = 1 to 7)

[Description]

- This command is used to output SRQ (Service Request) from the U3641 Series controller to an external controller connected to the GPIB port (In the upper section of the rear panel).
- SRQ has the following structure.

Status byte



[Example]

REQUEST 1

[Program Example]

```
IF A > 1 THEN
REQUEST 1
ELSE
PRINT "A = 1"
END IF
```

[Result]

If A is greater than 1, SRQ is sent to the external controller and data output is started. In the other cases, A is printed.

```
RUN
```

[Outline]

Starts executing a program.

[Format]

RUN [ <label> ]

[Description]

- If the label is omitted, the execution starts from the top of the program.
- If the label is specified, the execution starts from the position of the label.
- This command is used to reexecute (RUN only) the program that has been transferred to the BASIC buffer with **compile & run**.

[Example]

RUN

RUN *ABC

```
PURGE
```

[Outline]

Deletes a file in the memory card.

[Format]

PURGE <file name>

[Description]

- The PURGE command deletes a program file or data file in the memory card.
- The PURGE command should be used in the **BASIC mode** of the editor.
- If the specified file is not found, the following error message appears:
  (PURGE (file name) error)

[Example]

PURGE "BAS1"

[Note]

- Before entering the PURGE command, check the contents of the file to delete with the CAT command or Load of the editor.
- The PURGE command cannot be executed when the write protect of the memory card is turned on.

```
SCREEN
```

[Outline]

Specifies the display condition of the main-body screen.

[Format]

SCREEN <numeric expression>

Numeric expression: 0 to 4

[Description]

- Specifies the display condition of the main-body screen at the execution of the program.
- The following numbers from 0 to 4 correspond to the main-body GPIB codes from VS0 to VS4.

| | Display condition | |
|---|---|---|
| 0 | SPA waveform screen | (Measuring) |
| 1 | Coupled-screen of SPA waveform + BASIC | (Measuring) |
| 2 | BASIC screen | (Measuring) |
| 3 | BASIC screen | (No measuring) |
| 4 | Split-screen (2 sections) of SPA waveform + BASIC | (Measuring) |

0 : SPA waveform screen
Displays the normal waveform screen only, not display the PRINT and graphic instructions.

1 : Coupled-screen of SPA waveform screen + BASIC screen
Displays the PRINT and graphic instructions on the waveform screen with coupling the screens.

2 : BASIC screen (Measuring)
Displays the PRINT and graphic instructions on the BASIC screen with performing the measurement.

3 : BASIC screen (No measuring)
Displays the PRINT and graphic instructions speedy on the BASIC screen with no measurement.

4 : SPA waveform screen + BASIC screen (multi-screen)
Displays the SPA waveform on the top screen, and the BASIC screen on the bottom screen.

These screen settings are initialized by the IP command. After executing the IP command, therefore, reset them.

[Example]

```
SCREEN 0
SCREEN 1
SCREEN M
```

[Note]

The specification of display position is different in the display condition (1 to 3, and 4) by the CURSOR instruction.

```
SELECT CASE/CASE ELSE/END SELECT
```

[Outline]

Branches more than one times, using the value of a expression as a condition.

[Format]

```
SELECT  <numeric expression>
CASE    <numeric expression>
END SELECT
```

[Description]

- Execute sentence following to the CASE sentence matched specified value by CASE sentence and specified value by SELECT sentence.
- If each of the values is the same, the CASE statements will be executed. If not, a CASE ELSE statements will be executed.
- SELECT statements can be nested.

[Example]

```
INPUT “A =”, A
SELECT A
  CASE 1
    PRINT “1”
  CASE 3
    PRINT “3”
  CASE 5
    PRINT “5”
  CASE ELSE
    PRINT “ELSE”
  END SELECT
```

[Note]

- END SELECT must be put, when a SELECT statement is specified.
- The SELECT statement allows only numeric expressions.

[Sample Program]

```
INPUT "A =", A                   : Inputs to the variable A.
INPUT "B =", B                   : Inputs to the variable B.
SELECT A                         : ┐ SELECT statement of A
  CASE 1                         : │    If A = 1, goes to the following processes.
    SELECT B                     : │ ┐ SELECT statement of B
      CASE 10                    : │ │    If B = 10, goes to the following processes.
        PRINT "A = 1, B = 10"    : │ │      Prints character strings
      CASE ELSE                  : │ │    If B <> 10, goes to the following processes.
        PRINT "A = 1, B = ELSE"  : │ │      Prints character strings
    END SELECT                   : │ ┘ Termination of the SELECT statement of B
  CASE 2                         : │    If A = 2, goes to the following processes.
    PRINT "A = 2"                : │      Prints character strings
  CASE 10                        : │    If A = 10, goes to the following processes.
    PRINT "A = 10"               : │      Prints character strings
END SELECT                       : ┘ Termination of the SELECT statement of A
```

[Result]

```
A = 1
B = 10
A = 1, B = 10

A = 1
B = 20
A = 1, B = ELSE

A = 10
B = 10
A = 10
```

## SEND

[Outline]

Outputs commands and data to the GPIB.

[Format]

```
SEND   <A> I <B> I <C> { ,<A> I <B> }
<A>:   CMD I DATA I LISTEN [<D> {, <D> } ]
<B>:   UNT I UNL
<C>:   TALK [ <D> ]
<D>:   numeric expression
```

[Description]

- SEND sends the universal command, address command and data independently to the GPIB.

| | |
|---|---|
| CMD | : Makes attention line true and sends provided numeric in eight bits binary data to the GPIB. Therefore the numeric must be between 0 and 255 and expression using decimal will be converted into the integer type.<br>To execute without numeric specification makes the Attention line (ATN) true. |
| DATA | : Makes the Attention (ATN) line false and sends provided numeric in eight bits binary data to the GPIB. The numeric is the same one as handled in “CMD”.<br>To execute without numeric specification makes the Attention line (ATN) false. |
| LISTEN | : Sends provided numeric to the GPIB as the Listener Address Group (LAG). It is allowed to specify the numeric between 0 and 30, and more than one. |
| TALK | : Sends provided numeric to the GPIB as the Talker Address Group (TAG). The numeric is between 0 and 30. (Multiple specification is not allowed.) |
| UNL | : Sends an Unlisten (UNL) command to the GPIB. Devices specified as a listener before this command execution will be released its listener mode. |
| UNT | : Sends an Untalk (UNT) commands to the GPIB. Devices specified as a talker before this command execution will be released its talker mode. |

[Example]

```
SEND UNT UNL TALK 1 LISTEN 2, 3 DATA 0x43 0x46 0x35 0x4d 0x5a 0xd 0xa
SEND CMD
SEND DATA
```

[Note]

Not function in slave mode.

[Sample Program]

```
FOR I = 1 TO 9
  A = I+0x30
  SEND UNT UNL TALK 30 LISTEN 11 DATA 0x43 0x46 A 0x4d 0x5a 0xd 0xa
  WAIT 1000
NEXT I
SEND UNL UNT
```

[Result]

Change specification for center frequency of address 11 from 1MHz to 9MHz by each one second.
Release listener condition of address 11 and talker condition of address 30.

```
SPOLL (X)
```

[Outline]

Provides a serial pole onto a specified device on the GPIB and read its status byte.

[Format]

SPOLL (X)

| | | |
|---|---|---|
| X: | Device address 0 to 30: | Devices on the GPIB |
| | 31: | measuring devices of the main unit |

[Description]

- SPOLL provides a serial pole onto a specified device on the GPIB (CONTROLLER side) and read its status byte.
- The status byte of the measuring devices of the main unit is as follows.

| Bit | Description |
|---|---|
| 0 | Sets when UNCAL arises. |
| 1 | Sets when a calibration is completed. |
| 2 | Sets when a scanning is completed. |
| 3 | Sets when an average reaches a set up times. |
| 4 | Sets when a plot output is completed. |
| 5 | Sets when an error is detected in the GPIB code. |
| 6 | Sets when either of bits, 0 to 5 and 7, is set during the mode is set in “S0”, in which an SRQ interruption may occur. |
| 7 | Sets when a REQUEST command is executed. |

[Example]

| | |
|---|---|
| SPOLL (11) : | Provides a serial pole to the device address 11. |
| SPOLL (31) : | Provides a serial pole to the measuring device of the main unit. |

[Note]

Not function in slave mode. (Return “0” in slave mode.)

[Sample Program]

```
INTEGER S
OUTPUT 31; "S0"
ON ISRQ GOSUB *SS
ENABLE INTR
*L
 GOTO *L
*SS
 S = SPOLL (31)
 IF S BAND 4 THEN PRINT "SWEEP END"
 RETURN
```

[Result]

"SWEEP END" will be printed whenever the measuring device of the main unit finishes its scanning.

```
SPRINTF
```

[Outline]

Edits numeric values and character strings and enters them to a character-string variable.

[Format]

SPRINTF <character-string variable>, <character-string expression> [ , <X> ]

X : Numeric expression | character-string expression

[Description]

- The SPRINTF instruction is equivalent to the PRINTF instruction, except that SPRINTF can enter a edited character string to the specified character-string variable.
- If the character-string expression includes the following conversion-specification characters, SPRINTF converts its arguments and enters them to a character-string expression. (Each conversion specification starts with a %.)

| | |
|---|---|
| Minus sign : | Light-justifies the converted argument in its output field. |
| Period : | Separates the numeric string for the field width from that for the number of characters (precision). |
| 0 (zero) : | Zero-suppresses the excessive character positions of the field. |

- Conversion characters and their meanings

| | |
|---|---|
| d : | Converts the argument to decimal number. |
| o : | Converts the argument to unsigned octal number (without a leading zero). |
| x : | Converts the argument to unsigned hexadecimal number (without a leading zero.) |
| s : | Prints the character string. |
| e : | Accepts the argument as a real number and converts it into a decimal number in the form of [-] m.nnnnnnE [±] xx. (The size of the n-character string is determined by the precision, normally 6.) |
| f : | Accepts the argument as a real number and converts it into a decimal number in the form of [-] mmm.nnnn (The size of the n-character string is determined by the precision, normally 6.) |

- If a non-conversion character follows a %, the character is entered to the character-string expression to be output. Therefore, specify %% to enter % to the output.

[Example]

SPRINTF S$, "C = %d", C
: Enters variable C as a decimal number.

SPRINTF S$, "C = %5d", C
: Enters variable C as a decimal number of 5 digits (right-justified).

SPRINTF S$, "C = %-5d", C
: Enters variable C as a decimal number of 5 digits (left-justified).

SPRINTF S$, "C = %05d", C
: Enters variable C as a decimal number of 5 digits (right-justified and zero-suppressed).

SPRINTF S$, "H = %x", H
: Enters variable H as a hexadecimal number.

SPRINTF S$, "H = %4x", H
: Enters variable H as a hexadecimal number. (right-justified).

SPRINTF S$, "H = %-4x", H
: Enters variable H as a hexadecimal number. (left-justified).

SPRINTF S$, "H = %04x", H
: Enters variable H as a hexadecimal number. (right-justified and zero-suppressed).

SPRINTF B$, "S$ = %s", S$
: Enters a character string.

SPRINTF B$, "S$ = %8s", S$
: Enter 8 string characters from right-justify.

SPRINTF B$, "S$ = %-8s", S$
: Enter 8 string characters from left-justify.

SPRINTF B$, "%20.10s", S$
: Enters 10 characters right-justified in the field of 20-character width.

SPRINTF B$, "%-20.10s", S$
: Enters 10 characters left-justified in the field of 20-character width.

SPRINTF B$, "F = %f", F
: Enters variable F as a real decimal number.

SPRINTF B$, "F = %8.2f", F
: Enters variable F as a 8-digit number, including a decimal point and two fractional digits.

SPRINTF B$, "F = %08.2f", F
: Enters variable F as a zero-suppressed number of 8 digits, including a decimal point and two fractional digits.

[Note]

If there are not enough conversion characters or arguments or they are the wrong type, the assignment may turn out meaningless.

[Program Example]

```
DIM S$ [80]
SPRINTF S$, “%d * %4d = %05d”, 12, 34, 12*34
PRINT S$
SPRINTF S$, “:%-10d:%10d:”, 123, 456
PRINT S$
A$ = “ABCD”
SPRINTF S$, “:%8s:%-8s:%8.2s:%-8.2s:”, A$, A$, A$, A$
PRINT S$
SPRINTF S$, “%f+%8.3f - %06.3f = %e”, 1.23, 4.56, 7.89, 1.23+4.56-7.89
PRINT S$
```

[Result]

```
12 *     34 = 00408
:123        :       456:
:     ABCD:ABCD     :      AB:AB      :
1.230000 +      4.560 - 07.890 = -2.100000e + 00
```

```
STOP
```

[Outline]

Terminates the execution of a program.

[Format]

STOP

[Description]

STOP terminates the execution of a program. The following message will be displayed with STOP termination.
(Program ended normally.)

[Example]

STOP

[Note]

If a program was stopped with STOP, it cannot be restarted.

[Sample Program]

```
FOR I = 1 TO 10        : Enters 1 into the counter 1, then loops until I > 10.
  IF I = 5 THEN STOP : If I = 5, terminates the program.
  PRINT I              : Outputs variable I.
NEXT I                 : Returns to the loop.
```

[Result]

```
1.0
2.0
3.0
4.0
Program ended normally.
```

```
TIME$
```

[Outline]

The present time (hour, minute, second) is returned by the character string.

[Format]

TIME$

[Description]

The present time (hour, minute, second) is returned by the character string.

[Example]

TIME$

[Program Example]

```
A$ = TIME$
PRINT A$
```

## TRIGGER

[Outline]

Sends Group Execute Triggers (GET) of Address Command Group (ACG) to all or specified devices on GPIB.

[Format]

```
TRIGGER  [ device address { , device address } ]
   device address:   0 to 30
```

[Description]

- If a TRIGGER command is executed without a device address specification, only a Group Execute Trigger (GET) is send. In this case devices to be triggered must be set as a listener previously.
- If a device address is specified after a TRIGGER command, a GET command is sent only to a specified device.

[Example]

```
TRIGGER

TRIGGER 2

TRIGGER 3, 4, 5
```

[Note]

Not function in slave mode.

[Sample Program]

```
TRIGGER 5
ENTER 5; A$
PRINT A$
```

[Result]

The device address 5 is triggered and input data from it.

## WAIT

[Outline]

Terminates the execution of a program for specified time.

[Format]

```
WAIT <numeric expression>
```

[Description]

- The numeric expression specifies time in milliseconds.
- It allows to specify time between 0 and 63999 milliseconds.

[Example]

```
WAIT 1000
WAIT 60000
A = 200
WAIT A
```

[Note]

- Time must be specified in milliseconds.
- Even if a interruption specified by ON (SRQ/ISRQ/KEY) arises, no branching will be executed by the interruption during a execution of this command.

[Sample Program]

```
FOR I = 1 TO 8
 READ S
 BUZZER S, 1000
 WAIT 1000
NEXT I
DATA 261, 294, 330, 349, 392, 440
DATA 494, 523
```

[Result]

Sounds in a music scale, changing every second just as, “do, re, mi, fa, sol, la, si, do”.

# 4 BUILT-IN FUNCTION

## 4.1 Outline

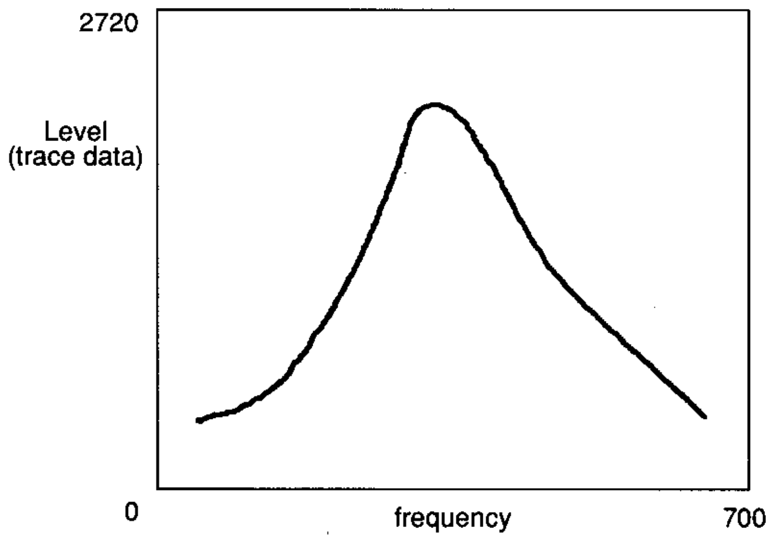
The built-in function can perform analytical process with easy operation, and shorter the time for program development and handle high throughput.

## 4.2 Before Using The Built-in Function

### 4.2.1 Point, Trace Data and Notes

(1) Point processing

The point includes both frequency (horizontal axis) and level (vertical axis), and indicates the precious as shown in the following figure.



- Frequency point
  Horizontal axis 701 points The frequency per one point shows frequency span/700.
- Level point (trace data)
  Vertical axis 2721 points The level per one point shows dynamic range/2720.

(2) Graphic function

This BASIC has a graphic function.

Before you go to the graphic description, you should know the function of the display.

There are three types of the screen.

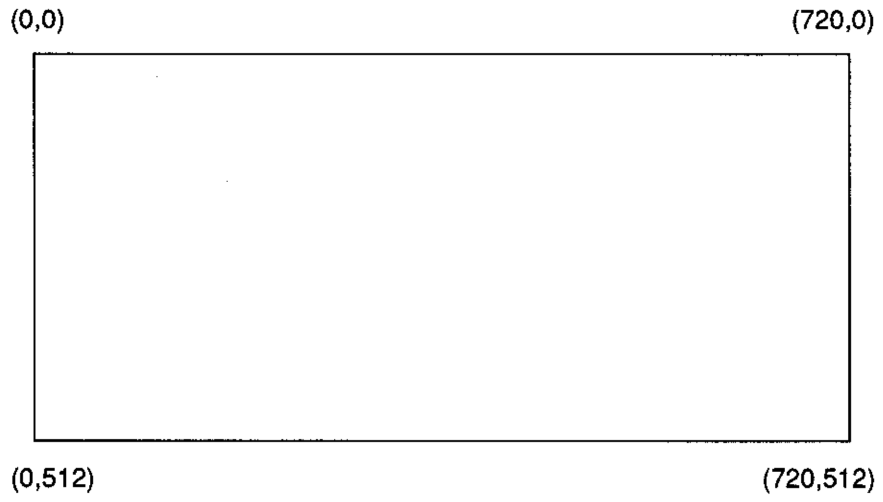
| | |
|---|---|
| Waveform screen : | Appears first after the preset button is depressed. |
| Character screen (Execution screen) : | Displays characters by BASIC PRINT command and so on. |
| Graphic screen : | The BASIC graphic function displays this screen. |

Before output either of the above screens by the BASIC program, execute either of the following BASIC command.
(The SCREEN command also allows the specification.)

| Display mode | BASIC command |
|---|---|
| Waveform screen | SCREEN 0 |
| Waveform screen + Character screen | SCREEN 1 |
| Character screen | SCREEN 2 |
| Graphic screen | SCREEN 3 |
| Waveform screen + Character screen (Split-screen (2 sections)) | SCREEN 4 |

Before executing the graphic function, execute SCREEN 3.

The graphic screen is specified as follows.



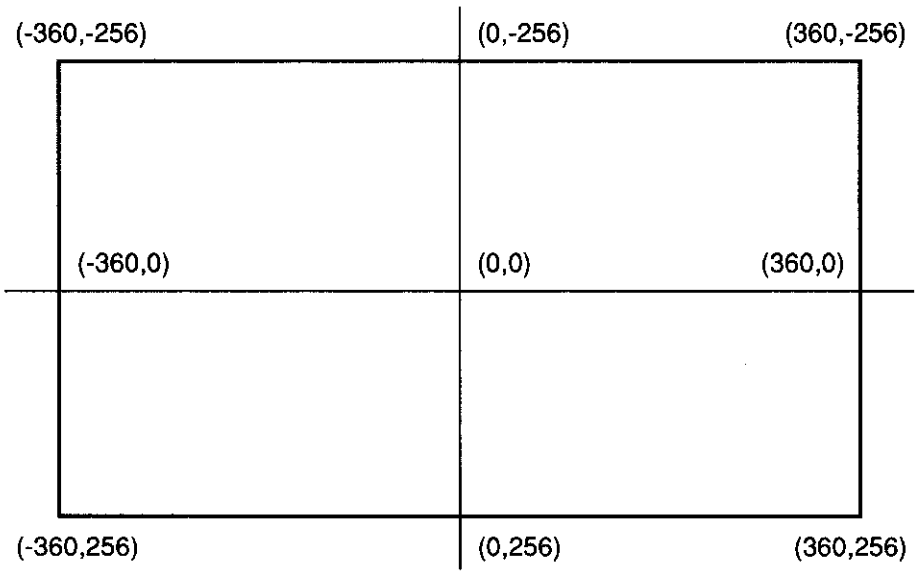
The above figure is called an absolute address. The upper left is the original (0, 0).

When view-port address is set in GADRS function, the starling point (0, 0) can be specified accordingly within the range at the figure of the previous page.

(Example)

If GADRS (1, 360, 256) is determined, the display is as the follows.

***Note:*** ***In the view port address specification, specification which exceeds the picture range in setting will occur on error.***



### 4.2.2 Note for Use

① To calculate data related the ripple, the maximum and minimum number must be calculated first by the following built-in functions. Without this calculation result in a error. (function error)

| | | |
|---|---|---|
| NRPLH | -- | for the maximum data |
| NRPLL | -- | for the minimum data |

***Note:*** ***In the case both data (maximum and minimum) is needed, execute both calculation.***

After the above calculation, the following built-in functions are available.

PRPLHN
PRPLLN
FRPLHN
FRPLLN
VRPLHN
VRPLLN

② If an error occurs in function parameter and the like, only an error message will be displayed without a termination of the program. (function error)
In this case, the calculated value will be indefinite.

③ The built-in function calculates faster than the set of OUTPUT 31 and ENTER 31. (Both the set of OUTPUT 31 and ENTER 31 and the built-in function calculate the same value.)

④ To specify the point (0 to 700), trace data (0 to 2720) and trace A/B (0/1) out of the range causes an error. (function error)

### 4.2.3 Structure of Description

The function will be described as follows.

| | |
|---|---|
| FREQ (Frequency) ----- | Frequency number (return value) |
| [Feature] ------------- | Feature of functin |
| [Format] ------------- | Syntax (format) in the BASIC |
| [Return Value] ------------- | Returned value from execution of a function |
| [Error] ------------- | Specified parameter error, etc. |
| [Note] ------------- | Note for specification of a parameter |
| [Example] ------------- | Sample BASIC program |

(1) Error

If an error arises in built-in function, the BASIC program will not terminate.
To identify an error in a program, use the ON ERROR statement.

(Example)

```
ON ERROR GOTO *ERR
```

(2) List of Built in Function

| Function | Structure | Description |
|---|---|---|
| Frequency/ point operation | F = FREQ (P) | Calculates a frequency from a point value. |
| | F = DFREQ (P1, P2) | Calculates a frequecny from a width between points. |
| | P = POINT (F) | Calculates a point (horizontal axis). |
| | P = DPOINT (F1, F2) | Calculates a point between specified frequencies. |
| Level/point operation | L = LEVEL (T) | Calculates a level of trace data specified with a point value (vertical axis). |
| | L = DLEVEL (T1, T2) | Calculates a level between points (vertical axis). |
| | T = LVPOINT (L) | Calculates the point (vertical axis) of trace data from a level. |
| | T = LVDPOINT (L1, L2) | Calculates the point (vertical axis) of trace data between levels. |
| | L = VALUE (P, M) | Calculates a level at a frequency position specified with a point value. |
| | L = DVALUE (P1, P2, M) | Calculates a difference of levels at two frequency positions specified with point values. |
| | L = CVALUE (F, M) | Calculates a level at a frequency position. |
| | L = DCVALUE (F1, F2, M) | Calculates the level difference between two specified frequency positions. |
| Maximum/ minimum operation | F = FMAX (P1, P2, M) | Calculates the frequency at the maximum level position between two positions specified with point value. |
| | F = FMIN (P1, P2, M) | Calculates the frequency at the minimum level position between two positions specified with point value. |
| | P = PMAX (P1, P2, M) | Calculates the frequency point (horizontal axis) maximum level position between two positions at the specified with point values. |
| | P = PMIN (P1, P2, M) | Calculates the frequency point (horizontal axis) at the minimum level position between two positions specified with point values. |
| | L = MAX (P1, P2, M) | Calculates the maximum level between two positions specified with point values. |
| | L = MIN (P1, P2, M) | Calculates the minimum level between two positions specified with point values. |

| Function | Structure | Description |
|---|---|---|
| Bandwidth operation | F = BND (P, X, M) | Calculates the frequency bandwidth of a LOSS level at a position specified with point values. |
| | F = BNDL (P, X, M) | Calculates the low frequency bandwidth of a LOSS level at a position specified with point values. |
| | F = BNDH (P, X, M) | Calculates the high frequency bandwidth of a LOSS level at a position specified with point values. |
| | F = CBND (F, X, M) | Calculates the frequency bandwidth of a LOSS level at a position specified with a frequency. |
| | F = CBNDL (F, X, M) | Calculates the low frequency bandwidth of a LOSS level at a position specified with a frequency. |
| | F = CBNDH (F, X, M) | Calculates the high frequency bandwidth of a LOSS level at a position specified with a frequency. |
| Maximum/ minimum operation | N = NRPLH (P1, P2, Dx, Dy, M) | Set the number of every maximum point. |
| | N = NRPLL (P1, P2, Dx, Dy, M) | Set the number of every minimum point. |
| | P = PRPLHN (N, M) | Set horizontal axis point for maximum point of N's turn from the left. |
| | P = PRPLLN (N, M) | Set horizontal axis point for minimum point of N's turn from the left. |
| | F = FRPLHN (N, M) | Set frequency for maximum point of N's turn from the left. |
| | F = FRPLLN (N, M) | Set frequency for minimum point of N's turn from the left. |
| | L = VRPLHN (N, M) | Set level for maximum point of N's turn from the left. |
| | L = VRPLLN (N, M) | Set level for minimum point of N's turn from the left. |
| | L = RPL1 (P1, P2, Dx, Dy, M) | Set level difference between maximum value of maximum point and minimum value of minimum point. |

| Function | Structure | Description |
|---|---|---|
| Decision for upper and lower limit | C = LMTMD1 (Dd, S, Ds)<br>C = LMTMD2 (P, S, Ds, M)<br>C = LMTUL1 (Dd, Up, Lo)<br>C = LMTUL2 (P, Up, Lo, M) | Decide specified data with standard value and width of upper and lower.<br>Decide waveform data for horizontal axis-point position with standard value and width of upper and lower.<br>Decide specified data with upper limit value and lower limit value.<br>Decide waveform data for horizontal axis-point position with upper limit value and lower limit value. |
| Electric power operation | W = POWER (P1, P2, M) | Set total electric power between horizontal axis points. |
| Read/write of trace data | TRACE (M, P)<br>T = RTRACE (P, M)<br>WTRACE (T, P, M)<br>***Note: This function returns no value.*** | Read or Write trace data for appointed point.<br>Read trace data for appointed point.<br>Write trace data for appointed point. |
| Graphic | GADRS (Mo, X, Y)<br>GFLRECT (D, X1, Y1, X2, Y2)<br>GLINE (D, X1, Y1, X2, Y2)<br>GPOINT (D, X, Y)<br>GRECT (D, X1, Y1, X2, Y2) | Specify absolute address/view-point address for graphic point.<br>Paint out rectangle which includes diagonal between appointed 2 poinsts.<br>Draw line between appointed 2 points.<br>Draw dot at appointed position.<br>Draw rectangle which includes diagonal between appointed 2 points. |

| | | | |
|---|---|---|---|
| F | : Frequency (Hz) | Dx | : Differential coefficient in horizontal axis |
| P | : Point (0 to 700) | Dy | : Differential coefficient in vertial axis |
| L | : Level | S | : Reference value |
| T | : Trace data (0 to 2720) | Ds | : Upper and lower limit width |
| M | : Kind of Trace | Dd | : Sample data |
| | 0;Trace A, 1;Trace B | Lo | : Lower limit |
| X | : Loss level (dB) | Up | : Upper limit |
| C | : Check value [0, 1, 2]<br>0;Inside, 1;Above the upper limit, 2;Under the lower limit | | |
| N | : Ripple number | | |
| W | : Power | | |

For Graphic:

| | | |
|---|---|---|
| D | : Set/Erase | 0;Erase, 1;Set (Draw) |
| Mo | : Address mode | 0;Absolute address, 1;Viewport address |
| X | : Coordinate (Horizontal axis) | |
| Y | : Coordinate (Vertical axis) | |

## 4.3 Built-in Functions

The built-in functions are described as the following order.

| Function |
|---|
| FREQ (Frequency) |
| DFREQ (Frequency) |
| POINT (Point: Horizontal axis [0 to 700]) |
| DPOINT (Point: Horizontal axis [0 to 700]) |
| LEVEL (Level) |
| DLEVEL (Level) |
| LVPOINT (Trace data: [0 to 2720]) |
| LVDPOINT (Trace data: [0 to 2720]) |
| VALUE (Level) |
| DVALUE (Level) |
| CVALUE (Level) |
| DCVALUE (Level) |
| FMAX (Frequency) |
| FMIN (Frequency) |
| PMAX (Point: Horizontal axis [0 to 700]) |
| PMIN (Point: Horizontal axis [0 to 700]) |
| MAX (Level) |
| MIN (Level) |
| BND (Frequency) |
| BNDL (Frequency) |
| BNDH (Frequency) |
| CBND (Frequency) |
| CBNDL (Frequency) |
| CBNDH (Frequency) |
| NRPLH (Maximum points number) |
| NRPLL (Minimum points number) |
| PRPLHN (Point: Horizontal axis [0 to 700]) |
| PRPLLN (Point: Horizontal axis [0 to 700]) |
| FRPLHN (Frequency) |
| FRPLLN (Frequency) |
| VRPLHN (Level) |
| VRPLLN (Level) |
| RPL1 (Level) |
| LMTMD1 (Check value [0, 1, 2]) |
| LMTMD2 (Check value [0, 1, 2]) |
| LMTUL1 (Check value [0, 1, 2]) |
| LMTUL2 (Check value [0, 1, 2]) |
| POWER (Total power) |
| TRACE (Trace data: [0 to 2720]) |
| RTRACE (Trace data: [0 to 2720]) |
| WTRACE (Trace data: [0 to 2720]) |

```
FREQ (Frequency)
```

[Feature]

Specify a point, and FREQ will calculate the corresponding frequency.

[Format]

FREQ (P)
P: Specified Point (0 to 700)

[Return Value]

| | | |
|---|---|---|
| Normal termination | : | Frequency (Hz) converted from the point value. At zero frequency span, the time (sec). |
| Error interruption | : | Indefinite value. |

[Error]

- If the specified value is out of the range, 0 to 700, an error will result.
- Return value will be indefinite.
- An error causes only an error message and not terminates the program.

[Example]

To calculate the frequency at the 350th point.

```
F = FREQ (350)
```

To calculate the frequency at the 400th point.

```
I = 200
F = FREQ (I*2)
```

## DFREQ (Frequency)

[Feature]

Specify a point width (point 1 and 2), and DFREQ will calculate the corresponding frequency bandwidth.

[Format]

DFREQ (P1, P2)

| | |
|---|---|
| P1: | Specified point 1 (0 to 700) |
| P2: | Specified point 2 (0 to 700) |

[Return Value]

| | |
|---|---|
| Normal termination: | Frequency (Hz) between the point 1 and point 2. At zero frequency span, the time (sec). |
| Error interruption : | Indefinite value. |

[Error]

- If the specified value is out of the range, 0 to 700, an error will result.
- Return value will be indefinite.
- An error causes only an error message and not terminates the program.

[Note]

If P1 > P2, convert P1 and P2.

[Example]

To calculate the frequency between the 300th point and 400th point.

```
FS = DFREQ (300, 400)

I = 400
FS = DFREQ (I, 300)
```

```
POINT (Point: horizontal axis [0 to 700] )
```

[Feature]

Specify a frequency, and POINT will calculate a point number (0 to 700) in a measuring device. (This function is important for the built-in function to operate rapidly.)

[Format]

POINT (F)
F: Specified frequency (Hz). At zero frequency span, the time (sec).

[Return Value]

| | |
|---|---|
| Normal termination: | Point number (0 to 700) converted from the frequency. |
| Error interruption : | Indefinite value. |

[Error]

- If the specified value is out of the range between a start frequency and stop frequency, an error will result.
- Return value will be indefinite.
- An error causes only an error message and not terminates the program.

[Example]

To calculate the measuring point at 3MHz.

```
PO = POINT (3E6)
```

To calculate the measuring point at 10MHz.

```
F = 10E6
PO = POINT (F)
```

```
DPOINT (Point: horizontal axis [0 to 700] )
```

[Feature]

Specify a frequency width (frequency 1 and 2), and DPOINT will calculate a point number (0 to 700) in a measuring device.

[Format]

DPOINT (F1, F2)

F1: Specified frequency 2 (Hz). At zero frequency span, the time should set 1 (sec).
F2: Specified frequency 1 (Hz). At zero frequency span, the time should set 2 (sec).

[Return Value]

Normal termination: Point number (0 to 700) between the specified frequencies F1 and F2.
Error interruption : Indefinite value.

[Error]

- If the specified value is out of the range between a start frequency and stop frequency, an error will result.
- Return value will be indefinite.
- An error causes only an error message and not terminates the program.

[Note]

If F1 > F2, convert F1 and F2.

[Example]

To calculate the measuring point between 3MHz and 4MHz.

```
PSPAN = DPOINT (3E6, 4E6)
PSPAN = DPOINT (4E6, 3E6)
```

To calculate the measuring point between 3MHz and 3.5MHz.

```
FA = 3E6
PSPAN = DPOINT (FA, 3.5E6)
```

## LEVEL (Level)

[Feature]

Specify a trace data (vertical axis [0 to 2720]), and LEVEL will calculate the corresponding level.

[Format]

LEVEL (T)
T: Specified trace data.

[Return Value]

| | |
|---|---|
| Normal termination: | Level converted from the trace data. (The same unit as the reference level.) |
| Error interruption : | Indefinite value. |

[Error]

- If the specified trace data are out of the range, 0 to 2720, an error will result.
- Return value will be indefinite.
- An error causes only an error message and not terminates the program.

[Example]

To calculate the level at trace data 200.

```
L = LEVEL (200)
```

To convert all the trace data of trace A into level values.

```
DIM L [701]
OUTPUT 31; "GTA"
FOR I = 0 TO 700
  L [I+1] = RTRACE (I, 0)
NEXT I
```

```
DLEVEL (Level)
```

[Feature]

Specify a trace data (vertical axis: [0 to 2720]) span (T1, T2), and DLEVEL will calculate the corresponding level.

[Format]

DLEVEL (T1, T2)

T1: Specified trace data 1
T2: Specified trace data 2

[Return Value]

| | |
|---|---|
| Normal termination: | Level converted from the trace data. (Unit is dB or V, W.) |
| Error interruption : | Indefinite value. |

[Error]

- If the specified trace data are out of the range, 0 to 2720, an error will result.
- Return value will be indefinite.
- An error causes only an error message and not terminates the program.

[Note]

If T1 < T2, T1 and T2 are exchanged.

[Example]

To calculate the level between trace data 200 and 300.

```
L = LEVEL (200, 300)
L = LEVEL (300, 200)
```

```
LVPOINT (Trace data: [0 to 2720] )
```

[Feature]

Specify a level, and LVPOINT will calculate a corresponding trace data value (vertical axis).

[Format]

LVPOINT (L)
L: Level (The same unit as the reference level).

[Return Value]

Normal termination: Trace data converted from the level.
Error interruption : Indefinite value.

[Error]

- When specification level shows external range of lower level to REF level, error occurs.
- Return value will be indefinite.
- An error causes only an error message and not terminates the program.

[Example]

To calculate the trace data at -10dBm.

```
TD = LVPOINT (-10)
```

To calculate the trace data at -20dBm.

```
L = -20
TD = LVPOINT (L)
```

```
LVDPOINT (Trace data: [0 to 2720] )
```

[Feature]

Specify a level range (L1, L2), and LVDPOINT will calculate a trace data value (vertical axis) between the levels.

[Format]

LVDPOINT (L1, L2)

L1: Level 1 (The same unit as the reference level).
L2: Level 2 (The same unit as the reference level).

[Return Value]

Normal termination: Trace data converted from the level.
Error interruption : Indefinite value.

[Error]

- When each specification level 1, 2 shows external range of lower level to REF level, error occurs.
- Return value will be indefinite.
- An error causes only an error message and not terminates the program.

[Note]

If L1 < L2, L1 and L2 are exchanged.

[Example]

To calculate the trace data between -5 and -10dBm.

```
TD = LVDPOINT (-5, -10)
```

To calculate the trace data between -5 and -20dBm.

```
L = -20
TD = LVDPOINT (-5, L)
```

```
VALUE (Level)
```

[Feature]

Specify a point value and the trace (A/B), and VALUE will calculate the level of the point in the specified trace side.

[Format]

VALUE (P, M)

P : Specified point value (0 to 700)
M : Trace 0 : Trace A
Trace 1 : Trace B

[Return Value]

| | |
|---|---|
| Normal termination: | Level of the specified value. (The same unit as the reference level.) |
| Error interruption : | Indefinite value. |

[Error]

- If the specified trace memory is other than the trace A/B or the specified point is out of the range, 0 to 700, an error will result.
- Return value will be indefinite.
- An error causes only an error message and not terminates the program.

[Example]

To calculate the level of the 300th point of the trace A.

```
L = VALUE (300, 0)
```

To calculate the level at 3MHz of the trace B frequency.

```
F = 3E6            |   |
P = POINT (F)      | = |  L = CVALUE (3E6, 1)
L = VALUE (P, 1)   |   |
```

(both the right and left programs calculate the same answer.)

```
DVALUE (Level)
```

[Feature]

Specify a point width (point 1 and 2) and the trace (A/B). DVALUE will calculate the level difference between the two points in the specified trace side.

[Format]

DVALUE (P1, P2, M)

| | | |
|---|---|---|
| P1 : | Specified point 1 (0 to 700) | |
| P2 : | Specified point 2 (0 to 700) | |
| M : | Trace 0: | Trace A |
| | Trace 1: | Trace B |

[Return Value]

| | |
|---|---|
| Normal termination: | Level difference between the two specified points. (Unit is dB or V, W.) |
| Error interruption : | Indefinite value. |

[Error]

- If the specified trace memory is other than the trace A/B or the specified point is out of the range, 0 to 700, an error will result.
- Return value will be indefinite.
- An error causes only an error message and not terminates the program.

[Note]

If P1 > P2, P1 and P2 are exchanged.

[Example]

To calculate the level difference between 30th points and 40th points of the trace A.

```
L = DVALUE (30, 40, 0)
L = DVALUE (40, 30, 0)
```

To calculate the level difference between 2MHz and 5MHz of the trace B frequency.

```
F = 2E6                  |   |
P1 = POINT (F)           |   |
P2 = POINT (5E6)         | = |  L = DCVALUE (2E6, 5E6, 1)
L = DVALUE (P1, P2, 1)   |   |
```

(both the right and left programs calculate the same answer.)

```
CVALUE (Level)
```

[Feature]

Specify a point value and the trace (A/B). CVALUE will calculate the level of the frequency position in the specified trace side.

[Format]

CVALUE (F, M)

F : Specified frequecny position (Hz). At zero frequency span, the time (sec).
M : Trace 0 : Trace A
    Trace 1 : Trace B

[Return Value]

| | |
|---|---|
| Normal termination: | Level of the specified frequency. (The same unit as the reference level.) |
| Error interruption : | Indefinite value. |

[Error]

- If the specified trace memory is other than the trace A/B or the specified point is out of the range, 0 to 700, an error will result.
- Return value will be indefinite.
- An error causes only an error message and not terminates the program.

[Example]

To calculate the level at 3.5MHz of the trace A.

```
L = VALUE (3.5E6, 0)
```

To calculate the level at 3MHz of the trace B frequency.

```
F = 3E6          |   |
P = POINT (F)    | = |  L = CVALUE (3E6, 1)
L = VALUE (P, 1) |   |
```

(both the right and left programs calculate the same answer.)

```
DCVALUE (Level)
```

[Feature]

Specify a frequency width (point 1 and 2) and the trace (A/B). CVALUE will calculate the level difference between the two frequencies in the specified trace side.

[Format]

DCVALUE (F1, F2, M)

| | | |
|---|---|---|
| F1 : | Specified frequency 1 (Hz). At zero frequency span, the time should set 1 (sec). | |
| F2 : | Specified frequency 2 (Hz). At zero frequency span, the time should set 2 (sec). | |
| M : | Trace 0: | Trace A |
| | 1: | Trace B |

[Return Value]

| | |
|---|---|
| Normal termination: | Level difference between the two specified frequencies. (Unit is dB or V, W.) |
| Error interruption : | Indefinite value. |

[Error]

- If the specified trace memory is other than the trace A/B or the specified point is out of the range, 0 to 700, an error will result.
- Return value will be indefinite.
- An error causes only an error message and not terminates the program.

[Note]

If F1 > F2, F1 and F2 are exchanged.

[Example]

To calculate the level difference between 3MHz and 4MHz of the trace A.

```
L = DCVALUE (3E6, 4E6, 0)
L = DCVALUE (4E6, 3E6, 0)
```

To calculate the level difference between 2MHz and 5MHz of the trace B.

```
F = 2E6                 |   |
P1 = POINT (F)          |   |
P2 = POINT (5E6)        | = |   L = DCVALUE (2E6, 5E6, 1)
L = DVALUE (P1, P2, 1)  |   |
```

(both the right and left programs calculate the same answer.)

## FMAX (Frequency)

[Feature]

Specify a measuring point range (point 1 and 2) and the trace A/B, and FMAX will calculate the frequency at the maximum position on the vertical axis of the specified trace side.

[Format]

FMAX (P1, P2, M)

| | |
|---|---|
| P1 : | Specified point 1 (0 to 700) |
| P2 : | Specified point 2 (0 to 700) |
| M : | Trace 0 : Trace A |
| | Trace 1 : Trace B |

[Return Value]

| | |
|---|---|
| Normal termination: | Frequency (Hz) at the maximum value position on the vertical axis between the range of specified points. |
| Error interruption : | Indefinite value. |

[Error]

- If the specified value is out of the range, 0 to 700, an error will result.
- Return value will be indefinite.
- An error causes only an error message and not terminates the program.

[Note]

If P1 > P2, P1 and P2 are exchanged.

[Example]

To calculate the frequency at the maximum value position between the 0th and 700th points of the trace A.

```
MF = FMAX (0, 700, 0)
```

To calculte the frequency at the maximum value position between the 10 and 20MHz of the trace B frequency.

```
F1 = 10E6
P1 = POINT (F1)
MF1 = FMAX (P1, POINT (20E6), 1)
MF2 = FMAX (POINT (F1), POINT (20E6), 1)
```

(MF1 and MF2 are equal.)

```
FMIN (Frequency)
```

[Feature]

Specify a measuring point range (point 1 and 2) and the trace A/B, and FMIN will calculate the frequency at the minimum position on the vertical axis of the specified trace side.

[Format]

FMIN (P1, P2, M)

| | | |
|---|---|---|
| P1 : | Specified point 1 (0 to 700) | |
| P2 : | Specified point 2 (0 to 700) | |
| M : | Trace 0 : | Trace A |
| | Trace 1 : | Trace B |

[Return Value]

| | |
|---|---|
| Normal termination: | Frequency (Hz) at the minimum value position between the range of specified points. |
| Error interruption : | Indefinite value. |

[Error]

- If the specified value is out of the range, 0 to 700, an error will result.
- Return value will be indefinite.
- An error causes only an error message and not terminates the program.

[Note]

If P1 > P2, P1 and P2 are exchanged.

[Example]

To calculate the frequency at the minimum value position between the 0th and 700th points of the trace A.

```
MF = FMIN (0, 700, 0)
```

To calculte the frequency at the minimum value position between the 10 and 20MHz of the trace B frequency.

```
F1 = 10E6
P1 = POINT (F1)
MF1 = FMIN (POINT (20E6), P1, 1)
MF2 = FMIN (POINT (20E6), POINT (F1), 1)
```

(MF1 and MF2 are equal.)

```
PMAX (Point: horizontal axis [0 to 700] )
```

[Feature]

Specify a measuring point range (point 1 and 2) and the trace A/B. PMAX will calculate a point number (horizontal: 0 to 700) on the vertical axis of the specified trace side.

[Format]

PMAX (P1, P2, M)

Trace 0 : Trace A
Trace 1 : Trace B
Specified point 2 (0 to 700)
Specified point 1 (0 to 700)

[Return Value]

| | |
|---|---|
| Normal termination: | Point at the maximum value position on the vertical axis between the range of specified points. (horizontal axis: 0 to 700) |
| Error interruption : | Indefinite value. |

[Error]

- If the specified value is out of the range, 0 to 700, an error will result.
- Return value will be indefinite.
- An error causes only an error message and not terminates the program.

[Note]

If P1 > P2, P1 and P2 are exchanged.

[Example]

To calculate the point (horizontal axis: 0 to 700) at the maximum value position between the 0th and 700th points of the trace A.

```
MP = PMAX (0, 700, 0)
```

To calculate the point (horizontal axis: 0 to 700) at the maximum value position between 10 and 20MHz points of the trace B.

```
F1 = 10E6
P1 = POINT (F1)
MP1 = PMAX (P1, POINT (20E6), 1)
MP2 = PMAX (POINT (F1), POINT (20E6), 1)
```

(MP1 and MP2 are equal.)

```
PMIN (Point: horizontal axis [0 to 700] )
```

[Feature]

Specify a measuring point range (point 1 and 2) and the trace A/B. PMIN will calculate a point (horizontal axis: 0 to 700) at the minimum position on the vertical axis of the specified trace side.

[Format]

PMIN (P1, P2, M)

| | | | |
|---|---|---|---|
| P1 : | Specified point 1 (0 to 700) | | |
| P2 : | Specified point 2 (0 to 700) | | |
| M : | Trace 0: | Trace A | |
| | Trace 1: | Trace B | |

[Return Value]

| | |
|---|---|
| Normal termination: | Point (horizontal axis: 0 to 700) at the minimum value position between a range of specified points. |
| Error interruption : | Indefinite value. |

[Error]

- If the specified value is out of the range, 0 to 700, an error will result.
- Return value will be indefinite.
- An error causes only an error message and not terminates the program.

[Note]

If P1 > P2, P1 and P2 are exchanged.

[Example]

To calculate the point (horizontal axis: 0 to 700) at the minimum value position between the 0th and 700th points of the trace A.

```
MP = PMIN (0, 700, 0)
```

To calculate the point (horizontal axis: 0 to 700) at the minimum value position between 10 and 20MHz points of the trace B frequency.

```
F1 = 10E6
P1 = POINT (F1)
MP1 = PMIN (P1, POINT (20E6), 1)
MP2 = PMIN (POINT (F1), POINT (20E6), 1)
```

(MP1 and MP2 are equal.)

```
MAX (Level)
```

[Feature]

Specify a measuring point range (point 1 and 2) and the trace A/B.
MAX will calculate the maximum level value level on the vertical axis of the specified trace side.

[Format]

MAX (P1, P2, M)

| | | | |
|---|---|---|---|
| P1 : | Specified point 1 (0 to 700) | | |
| P2 : | Specified point 2 (0 to 700) | | |
| M : | Trace | 0: | Trace A |
| | | 1: | Trace B |

[Return Value]

| | |
|---|---|
| Normal termination: | Maximum level on the vertical axis between the range of specified points. (The same unit as the reference level.) |
| Error interruption : | Indefinite value. |

[Error]

- If the specified value is out of the range, 0 to 700, an error will result.
- Return value will be indefinite.
- An error causes only an error message and not terminates the program.

[Note]

If P1 > P2, P1 and P2 are exchanged.

[Example]

To calculate the maximum level between the 0th and 700th points of the trace A.

```
ML = MAX (0, 700, 0)
```

To calculate the maximum level betweeen 10MHz and 20MHz of the trace B frequency.

```
F1 = 10E6
P1 = POINT (F1)
ML1 = MAX (P1, POINT (20E6), 1)
ML2 = MAX (POINT (F1), POINT (20E6), 1)
```

(ML1 and ML2 are equal.)

## MIN (Level)

[Feature]

Specify a measuring point range (piont 1 and 2) and the trace A/B.
MIN will calculate the minimum level value level on the vertical axis of the specified trace side.

[Format]

MIN (P1, P2, M)

| | |
|---|---|
| P1 : | Specified point 1 (0 to 700) |
| P2 : | Specified point 2 (0 to 700) |
| M : | Trace 0: Trace A |
| | 1: Trace B |

[Return Value]

| | |
|---|---|
| Normal termination: | Minimum level on the vertical axis between the range of specified points. (The same unit as the reference level.) |
| Error interruption : | Indefinite value. |

[Error]

- If the specified value is out of the range, 0 to 700, an error will result.
- Return value will be indefinite.
- An error causes only an error message and not terminates the program.

[Note]

If P1 > P2, P1 and P2 are exchanged.

[Example]

To calculate the minimum level between the 0th and 700th points of the trace A.

```
ML = MIN (0, 700, 0)
```

To calculate the minimum level betweeen 10MHz and 20MHz of the trace B frequency.

```
F1 = 10E6
P1 = POINT (F1)
ML1 = MIN (P1, POINT (20E6), 1)
ML2 = MIN (POINT (F1), POINT (20E6), 1)
```

(ML1 and ML2 are equal.)

```
BND (Frequency)
```

[Feature]

Specify a measuring point of reference data, a LOSS level and the trace A/B. BND will calculate a frequency bandwidth in the specified trace side.

[Format]

BND (P, X, M)

|  |  |
|---|---|
| P : | Point of reference data (0 to 700) |
| X : | LOSS level (dB) |
| M : | Trace 0 : Trace A |
|  | Trace 1 : Trace B |

[Return Value]

|  |  |
|---|---|
| Normal termination: | A width (Hz) of a LOSS level from reference data in a specified trace side. |
| Error interruption : | Indefinite value. |

[Error]

- If the specified value is out of the range, 0 to 700, an error will result.
- Return value will be indefinite.
- An error causes only an error message and not terminates the program.

[Example]

To calculate a bandwidth 3dB lower than the maximum value position of the trace A.

```
MP = PMAX (0, 700, 0)
BW1 = BND (MP, 3, 0)

BW2 = BND (PMAX (0, 700, 0), 3, 0)
```

(BW1 and BW2 are equal.)

To calculate a bandwidth 5dB lower than the center frequency point of the trace B.

```
BW = BND (350, 5, 1)
```

```
BNDL (Frequency)
```

[Feature]

Specify a measuring point of reference data, a LOSS level and the trace A/B. BNDL will calculate a frequency of the lower frequency band side (left side) in the specified trace side.

[Format]

BNDL (P, X, M)

P : Point of reference data (0 to 700)
X : LOSS level (dB)
M : Trace 0 : Trace A
    Trace 1 : Trace B

[Return Value]

| | |
|---|---|
| Normal termination: | Frequency (Hz) of the lower frequency band side (left) of a LOSS level from reference data of a specified trace side. |
| Error interruption : | Indefinite value. |

[Error]

- If the specified value is out of the range, 0 to 700, an error will result.
- Return value will be indefinite.
- An error causes only an error message and not terminates the program.

[Example]

To calculate a frequency of the lower frequency band side (left side), 5dB lower than the maximum value position of the trace A.

```
MP = PMAX (0, 700, 0)
BL1 = BNDL (MP, 5, 0)

BL2 = BNDL (PMAX (0, 700, 0), 5, 0)
```

(BL1 and BL2 are equal.)

To calculate a frequency of the lower frequency band side (left side), 5dB lower than the center frequency point of the trace B.

```
BL = BNDL (350, 5, 1)
```

```
BNDH (Frequency)
```

[Feature]

Specify a measuring point of reference data, a LOSS level and the trace A/B. BNDH will calculate a frequency of the higher frequency band side (right side) in the specified trace side.

[Format]

BNDH (P, X, M)

P : Point of reference data (0 to 700)
X : LOSS level (dB)
M : Trace 0 : Trace A
    Trace 1 : Trace B

[Return Value]

| | |
|---|---|
| Normal termination: | Frequency (Hz) of the higher frequency band side (right) of a LOSS level from reference data of a specified trace side. |
| Error interruption : | Indefinite value. |

[Error]

- If the specified value is out of the range, 0 to 700, an error will result.
- Return value will be indefinite.
- An error causes only an error message and not terminates the program.

[Example]

To calculate a frequency of the higher frequency band side (right side), 5dB higher than the maximum value position of the trace A.

```
MP = PMAX (0, 700, 0)
BH1 = BNDH (MP, 5, 0)
BH2 = BNDH (PMAX (0, 700, 0), 5, 0)
```

(BH1 and BH2 are equal.)

To calculate a frequency of the higher frequency band side (right side), 5dB higher than the center frequency point of the trace B.

```
BH = BNDH (350, 3, 1)
```

```
CBND (Frequency)
```

[Feature]

Specify a frequency position of reference data, a LOSS level and the trace A/B. CBND will calculate a frequency bandwidth in the specified trace side.

[Format]

CBND (F, X, M)

| | | |
|---|---|---|
| F : | Frequency of reference data | |
| X : | LOSS level (dB) | |
| M : | Trace 0 : | Trace A |
| | Trace 1 : | Trace B |

[Return Value]

| | |
|---|---|
| Normal termination: | The bandwidth (Hz) of a LOSS level from reference data in a specified trace side. |
| Error interruption : | Indefinite value. |

[Error]

- If the specified value is out of the range between a start frequency and stop frequency, an error will result.
- Return value will be indefinite.
- An error causes only an error message and not terminates the program.

[Example]

To calculate a bandwidth 3dB lower than 3MHz of the trace A frequency.

```
BW = CBND (3E6, 3, 0)
```

To calculate a bandwidth 5dB lower than 10MHz of the trace B frequency.

```
F = 10E6
L = 5
BW = CBND (F, L, 1)
```

```
CBNDL (Frequency)
```

[Feature]

Specify a frequency position of reference data, a LOSS level and the trace A/B. CBNDL will calculate the frequency of the lower frequency band side (left side) in the specified trace side.

[Format]

CBNDL (F, X, M)

|  |  |
|---|---|
| F : | Frequency position of reference data |
| X : | LOSS level (dB) |
| M : | Trace 0 : Trace A |
|  | Trace 1 : Trace B |

[Return Value]

|  |  |
|---|---|
| Normal termination : | The frequency (Hz) of the lower frequency bandwidth (left side) of a LOSS level from reference data in a specified trace side. |
| Error interruption : | Indefinite value. |

[Error]

- If the specified value is out of the range between a start frequency and stop frequency, an error will result.
- Return value will be indefinite.
- An error causes only an error message and not terminates the program.

[Example]

To calculate the left side of a bandwidth 3dB lower than 3MHz of the trace A frequency.

```
BW = CBNDL (3E6, 3, 0)
```

To calculate the left side of a bandwidth 5dB lower than 10MHz of the trace B frequency.

```
F = 10E6
L = 5
BW = CBNDL (F, L, 1)
```

```
CBNDH (Frequency)
```

[Feature]

Specify a frequency position of reference data, a LOSS level and the trace A/B. CBNDH will calculate the frequency of the higher frequency band side (right side) in the specified trace side.

[Format]

CBNDH (F, X, M)

| | | |
|---|---|---|
| F : | Frequency position of reference data | |
| X : | LOSS level (dB) | |
| M : | Trace 0 : | Trace A |
| | Trace 1 : | Trace B |

[Return Value]

| | |
|---|---|
| Normal termination: | The frequency (Hz) of the higher frequency bandwidth (right side) of a HIGH level from reference data in a specified trace side. |
| Error interruption : | Indefinite value. |

[Error]

- If the specified value is out of the range between a start frequency and stop frequency, an error will result.
- Return value will be indefinite.
- An error causes only an error message and not terminates the program.

[Example]

To calculate the right side of a bandwidth 3dB higher than 3MHz of the trace A frequency.

```
BW = CBNDH (3E6, 3, 0)
```

To calculate the right side of a bandwidth 5dB higher than 10MHz of the trace B frequency.

```
F = 10E6
L = 5
BW = CBNDH (F, L, 1)
```

```
NRPLH (Maximum points number)
```

[Feature]

Specify a measuring point range (point 1 and 2), trace A/B and a differential coefficient. NRPLH will search the maximum value corresponding with the frequency axis (horizontal axis: from the left) in the specified trace side to calculate the number of them.

[Format]

NRPLH (P1, P2, Dx, Dy, M)

| | |
|---|---|
| P1 : | Specified point 1 (0 to 700) |
| P2 : | Specified point 2 (0 to 700) |
| Dx : | Differential coefficient point (1 to 700) |
| Dy : | Differential coefficient point (1 to 2720) |
| M : | Trace 0: Trace A |
| | 1: Trace B |

[Return Value]

| | |
|---|---|
| Normal termination: | Number of the maximum values corresponding with the frequency axis. (horizontal axis: from the left) (maximum number: 255) |
| Error interruption : | Indefinite value. |

[Error]

- If the specified point is out of the range, 0 to 700, an error will result.
- If no maximum is searched, an error will result.
  (Change the Dx and Dy numeric.)
- Return value will be indefinite.
- An error causes only an error message and not terminates the program.

[Note]

If P1 > P2, convert P1 and P2.

Before PRPLHN, FRPLHN and VRPLHN are executed this function (NPRLH) must be executed exactly.

[Example]

To calculate the number of maximum value of coefficient Dx = 5 and Dy = 3 in the range 0 to 700 point of the trace A.

```
RH = NRPLH (0, 700, 5, 3, 0)
```

To calculate the number of maximum value of coefficient Dx = 3 and Dy = 5 in the range 10 to 20MHz of the trace B frequency.

```
STF = 10E6
SPF = 20E6
X = 3
Y = 5
STP = POINT (STF)
SPP = POINT (SPF)
TS = 1
RH = NRPLH (STP, SPP, X, Y, TS)
```

```
NRPLL (Minimum points number)
```

[Feature]

Specify a measuring point range (point 1 and 2), trace A/B and a differential coefficient. NRPLL will search the minimum value corresponding with the frequency axis (horizontal axis: from the left) in the specified trace side to calculate the number of them.

[Format]

NRPLL (P1, P2, Dx, Dy, M)

| | | |
|---|---|---|
| P1 | : | Specified point 1 (0 to 700) |
| P2 | : | Specified point 2 (0 to 700) |
| Dx | : | Differential coefficient point (1 to 700) |
| Dy | : | Differential coefficient point (1 to 2720) |
| M | : | Trace 0: Trace A<br>1: Trace B |

[Return Value]

| | | |
|---|---|---|
| Normal termination | : | Number of the minimum values corresponding with the frequency axis. (horizontal axis: from the left) (minimum number: 255) |
| Error interruption | : | Indefinite value. |

[Error]

- If the specified point is out of the range, 0 to 700, an error will result.
- If no minimum is searched, an error will result.
  (Change the Dx and Dy numeric.)
- Return value will be indefinite.
- An error causes only an error message and not terminates the program.

[Note]

If P1 > P2, convert P1 and P2.

Before PRPLLN, FRPLLN and VRPLLN are executed this function (NPRLL) must be executed exactly.

[Example]

To calculate the number of minimum value of coefficient Dx = 5 and Dy = 3 in the range 0 to 700 point of the trace A.

```
RL = NRPLL (0, 700, 5, 3, 0)
```

To calculate the number of minimum value of coefficient Dx = 3 and Dy = 5 in the range 10 to 20MHz of the trace B frequency.

```
STF = 10E6
SPF = 20E6
X = 3
Y = 5
STP = POINT (STF)
SPP = POINT (SPF)
TS = 1
RL = NRPLL (STP, SPP, X, Y, TS)
```

```
PRPLHN (Point: horizontal axis [0 to 700] )
```

[Feature]

Specify the trace A/B and n-th minimum position number from the left on the frequency axis. PRPLHN will calculate the point (0 to 700) at the specified minimum position.

[Format]

PRPLHN (N, M)

N : Number of the n-th maximum position from the left on the frequency axis. (horizontal axis: from the left)
M : Trace 0 : Trace A
Trace 1 : Trace B

[Return Value]

| | |
|---|---|
| Normal termination: | Point number (0 to 700) of the n-th maximum position from the left on the frequency axis. |
| Error interruption : | Indefinite value. |

[Error]

- If no n-th maximum position is detected, an error will result.
- Return value will be indefinite.
- An error causes only an error message and not terminates the program.

[Note]

Before this function, execute the NRPLH.

[Example]

To calculate all the maximum of the differential coefficient Dx = 5 and Dy = 3 in the range 0 to 700 point of the trace A, and next the point number at the maximum position third from the left.

```
RH = NRPLH (0, 700, 5, 3, 0)
P = PRPLHN (3, 0)
```

To calculate all the maximum of the differential coefficient Dx = 3 and Dy = 5 in the range 10 to 20MHz of the trace B frequency, and next the point number at the maximum position second from the left.

```
STF = 10E6
SPF = 20E6
X = 3
Y = 5
STP = POINT (STF)
SPP = POINT (SPF)
TS = 1
RH = NRPLH (STP, SPP, X, Y, TS)
P = PRPLHN (2, TS)
```

```
PRPLLN (Point: horizontal axis [0 to 700] )
```

[Feature]

Specify the trace A/B and n-th minimum position number from the left on the frequency axis. PRPLLN will calculate the point (0 to 700) at the specified minimum position.

[Format]

PRPLLN (N, M)

N : Number of the n-th minimum position from the left on the frequency axis. (horizontal axis: from the left)
M : Trace 0 : Trace A
Trace 1 : Trace B

[Return Value]

| | |
|---|---|
| Normal termination: | Point number (0 to 700) of the n-th minimum position from the left on the frequency axis. |
| Error interruption : | Indefinite value. |

[Error]

- If no n-th minimum position is detected, an error will result.
- Return value will be indefinite.
- An error causes only an error message and not terminates the program.

[Note]

Before this function, execute the NRPLL.

[Example]

To calculate all the minimum of the differential coefficient Dx = 5 and Dy = 3 in the range 0 to 700 point of the trace A frequnecy, and next the point number at the minimum position third from the left.

```
RH = NRPLL (0, 700, 5, 3, 0)
P = PRPLLN (3, 0)
```

To calculate all the minimum of the differential coefficient Dx = 3 and Dy = 5 in the range 10 to 20MHz of the trace B, and next the point number at the minimum position second from the left.

```
STF = 10E6
SPF = 20E6
X = 3
Y = 5
STP = POINT (STF)
SPP = POINT (SPF)
TS = 1
RH = NRPLL (STP, SPP, X, Y, TS)
P = PRPLLN (2, TS)
```

```
FRPLHN (Frequency)
```

[Feature]

Specify the trace A/B and n-th maximum position number from the left on the frequency axis. FRPLHN willl calculate the frequency at the specified maximum position.

[Format]

FRPLHN (N, M)

| | |
|---|---|
| N : | Number of the n-th maximum position from the left on the frequency axis. |
| M : | Trace 0 : Trace A |
| | Trace 1 : Trace B |

[Return Value]

| | |
|---|---|
| Normal termination: | The frequency (Hz) of the n-th maximum position from the left on the frequency axis. |
| Error interruption : | Indefinite value. |

[Error]

- If no n-th maximum position is detected, an error will result.
- Return value will be indefinite.
- An error causes only an error message and not terminates the program.

[Note]

Before this function, execute the NRPLH.

[Example]

To calculate all the maximum of the differential coefficient Dx = 5 and Dy = 3 in the range 0 to 700 point of the trace A, and next the frequency at the maximum position third from the left.

```
RH = NRPLH (0, 700, 5, 3, 0)
F = FRPLHN (3, 0)
```

To calculate all the maximum of the differential coefficient Dx = 3 and Dy = 5 in the range 10 to 20MHz of the trace B frequency, and next the frequency at the maximum position second from the left.

```
STF = 10E6
SPF = 20E6
X = 3
Y = 5
STP = POINT (STF)
SPP = POINT (SPF)
TS = 1
RH = NRPLH (STP, SPP, X, Y, TS)
F = FRPLHN (2, TS)
```

```
FRPLLN (Frequency)
```

[Feature]

Specify the trace A/B and n-th minimum position number from the left on the frequency axis. FRPLLN willl calculate the frequency at the specified minimum position.

[Format]

FRPLLN (N, M)

|  |  |
|---|---|
| N : | Number of the n-th minimum position from the left on the frequency axis. (horizontal axis: from the left) |
| M : | Trace 0 : Trace A<br>Trace 1 : Trace B |

[Return Value]

|  |  |
|---|---|
| Normal termination: | The frequency (Hz) of the n-th maximum position from the left on the frequency axis. |
| Error interruption : | Indefinite value. |

[Error]

- If no n-th minimum position is detected, an error will result.
- Return value will be indefinite.
- An error causes only an error message and not terminates the program.

[Note]

Before this function, execute the NRPLH.

[Example]

To calculate all the minimum of the differential coefficient Dx = 5 and Dy = 3 in the range 0 to 700 point of the trace A, and next the frequency at the minimum position third from the left.

```
RH = NRPLL (0, 700, 5, 3, 0)
F = FRPLLN (3, 0)
```

To calculate all the minimum of the differential coefficient Dx = 3 and Dy = 5 in the range 10 to 20MHz of the trace B frequency, and next the frequency at the minimum position second from the left.

```
STF = 10E6
SPF = 20E6
X = 3
Y = 5
STP = POINT (STF)
SPP = POINT (SPF)
TS = 1
RH = NRPLL (STP, SPP, X, Y, TS)
F = FRPLLN (2, TS)
```

```
VRPLHN (Level)
```

[Feature]

Specify the trace A/B and n-th maximum position number from the left on the frequency axis. VRPLHN will calculate the level at the specified maximum position.

[Format]

VRPLHN (N, M)

|  |  |
|---|---|
| N : | Number of the n-th maximum position from the left on the frequency axis. |
| M : | Trace 0: Trace A |
|  | 1: Trace B |

[Return Value]

|  |  |
|---|---|
| Normal termination: | Level of the n-th maximum position from the left on the frequency axis. (The same unit as the reference level.) |
| Error interruption : | Indefinite value. |

[Error]

- If no n-th minimum position is detected, an error will result.
- Return value will be indefinite.
- An error causes only an error message and not terminates the program.

[Note]

Before this function, execute the NRPLH.

[Example]

To calculate all the maximum of the differential coefficient Dx = 5 and Dy = 3 in the range 0 to 700 point of the trace A, and next the level at the maximum position third from the left.

```
RH = NRPLH (0, 700, 5, 3, 0)
L = VRPLHN (3, 0)
```

To calculate all the maximum of the differential coefficient Dx = 3 and Dy = 5 in the range 10 to 20MHz of the trace B frequency, and next the level at the maximum position second from the left.

```
STF = 10E6
SPF = 20E6
X = 3
Y = 5
STP = POINT (STF)
SPP = POINT (SPF)
TS = 1
RH = NRPLH (STP, SPP, X, Y, TS)
L = LRPLHN (2, TS)
```

```
VRPLLN (Level)
```

[Feature]

Specify the trace A/B and n-th minimum position number from the left on the frequency axis. VRPLLN will calculate the level at the specified minimum position.

[Format]

VRPLLN (N, M)

N : Number of the n-th minimum position from the left on the frequency axis.
M : Trace 0: Trace A
          1: Trace B

[Return Value]

| | |
|---|---|
| Normal termination: | Level of the n-th minimum position from the left on the frequency axis. (The same unit as the reference level.) |
| Error interruption : | Indefinite value. |

[Error]

- If no n-th minimum position is detected, an error will result.
- Return value will be indefinite.
- An error causes only an error message and not terminates the program.

[Note]

Before this function, execute the NRPLH.

[Example]

To calculate all the minimum of the differential coefficient Dx = 5 and Dy = 3 in the range 0 to 700 point of the trace A, and next the level at the minimum position third from the left.

```
RH = NRPLL (0, 700, 5, 3, 0)
L = VRPLLN (3, 0)
```

To calculate all the minimum of the differential coefficient Dx = 3 and Dy = 5 in the range 10 to 20MHz of the trace B frequency, and next the level at the minimum position second from the left.

```
STF = 10E6
SPF = 20E6
X = 3
Y = 5
STP = POINT (STF)
SPP = POINT (SPF)
TS = 1
RH = NRPLL (STP, SPP, X, Y, TS)
L = VRPLLN (2, TS)
```

```
RPL1 (Level)
```

[Feature]

Specify a measuring point range (point 1 and 2), trace A/B and a differential coefficient. RPL1 will search the maximum and minimum values in the point area of the specified trace side to calculate the level difference between the biggest maximum and smallest minimum values.

[Format]

RPL1 (P1, P2, Dx, Dy, M)

| | | |
|---|---|---|
| P1 : | Specified point 1 (0 to 700) | |
| P2 : | Specified point 2 (0 to 700) | |
| Dx : | Differential coefficient point (1 to 700) | |
| Dy : | Differential coefficient point (1 to 2720) | |
| M : | Trace 0: | Trace A |
| | 1: | Trace B |

[Return Value]

| | |
|---|---|
| Normal termination: | Level difference between the biggest maximum and smallest minimum values. (dB or V, W.) |
| Error interruption : | Indefinite value. |

[Error]

- If the specified point P1, P2 is out of the range, 0 to 700, an error will result.
- If no maximum and minimum are searched, an error will result.
  (Change the Dx and Dy numeric.)
- Return value will be indefinite.
- An error causes only an error message and not terminates the program.

[Note]

If P1 > P2, P1 and P2 are exchanged.

[Example]

To calculate level difference between the biggest maximum and smallest minimum values of coefficient Dx = 5 and Dy = 3, in the range 0 to 700 point of the trace A.

```
RP = RPL1 (0, 700, 5, 3, 0)
```

To calculate level difference between the biggest maximum and smallest minimum values of coefficient Dx = 3 and Dy = 5 in the range 10 to 20 MHz of the trace B frequency.

```
STF = 10E6
SPF = 20E6
X = 3
Y = 5
STP = POINT (STF)
SPP = POINT (SPF)
TS = 1
RP = RPL1 (STP, SPP, X, Y, TS)
```

```
LMTMD1 (Check Value [0, 1, 2] )
```

[Feature]

When standard value, upper and lower limit width from the standard value and tested data are given, it is checked whether or not to stay inside of up-down range width.

[Format]

LMTMD1 (Dd, S, Ds)

| | |
|---|---|
| Dd : | Sample data |
| S : | Reference value |
| Ds : | Upper and lower limit width |

Check whether it is ((S-Ds) $\leq$ Dd $\leq$ (S+Ds)).

[Return Value]

| | | |
|---|---|---|
| Normal termination: | Inside | .......... 0 |
| | Above the upper limit | .......... 1 |
| | Under the lower limit | .......... 2 |

[Example]

To check whether the input value is between 40 and 60.

```
INPUT D
T = LMTMD1 (D, 50, 10)
IF T = 0 THEN PRINT "OK"
IF T = 1 THEN PRINT "UPPER"
IF T = 2 THEN PRINT "LOWER"
```

To determine whether the 30MHz level of the trace A is between +10 and -10 of the input reference value.

```
INPUT S
LV = CVALUE (30E6, 0)
T = LMTMD1 (LV, S, 20)
IF T = 0 THEN PRINT "OK"
IF T = 1 THEN PRINT "UPPER"
IF T = 2 THEN PRINT "LOWER"
```

```
LMTMD2 (Check Value [0, 1, 2] )
```

[Feature]

When each point of standard value, upper and lower limit width from the standard value and trace A/B are given, it is checked whether the level of trace side point stay inside of up-down range width or not.

[Format]

LMTMD2 (P, S, Ds, M)

| | | |
|---|---|---|
| P | : | Point (0 to 700) |
| S | : | Reference value (Unit is the same as REF level.) |
| Ds | : | Up-down range width (dB or V, W) |
| M | : | Trace 0: Trace A |
| | | 1: Trace B |

Check whether it is ((S-Ds) ≤ level ≤ (S+Ds)).

[Return Value]

| | | |
|---|---|---|
| Normal termination: | Inside .......... | 0 |
| | Above the upper limit .......... | 1 |
| | Under the lower limit .......... | 2 |

[Example]

To determine whether the 30MHz level of the trace A is between ±10dB of the input reference value.

```
INPUT S
P = POINT (30E6)
T = LMTMD2 (P, S, 10, 0)
IF T = 0 THEN PRINT "OK"
IF T = 1 THEN PRINT "UPPER"
IF T = 2 THEN PRINT "LOWER"
```

```
LMTUL1 (Check Value [0, 1, 2] )
```

[Feature]

Specify an upper and lower limits and sample data. LMTUL1 will check if the sample data are inside or outside of the range between the upper and lower limits.

[Format]

LMTUL1 (Dd, Up, Lo)

| | |
|---|---|
| Dd : | Sample data |
| Up : | Upper limit |
| Lo : | Lower limit |

Check whether it is (Lo ≤ Dd ≤ Up)

[Return Value]

| | | |
|---|---|---|
| Normal termination: | Inside .......... | 0 |
| | Above the upper limit .......... | 1 |
| | Under the lower limit .......... | 2 |

[Example]

To check whether the input value is between 30 and 40.

```
INPUT D
T = LMTUL1 (D, 40, 30)
IF T = 0 THEN PRINT "OK"
IF T = 1 THEN PRINT "UPPER"
IF T = 2 THEN PRINT "LOWER"
```

To determine whether the 30MHz level of the trace A is between the input upper and lower limits.

```
INPUT "UPPER", U
INPUT "LOWER", L
LV = CVALUE (30E6, 0)
T = LMTUL1 (LV, U, L)
IF T = 0 THEN PRINT "OK"
IF T = 1 THEN PRINT "UPPER"
IF T = 2 THEN PRINT "LOWER"
```

```
LMTUL2 (Check Value [0, 1, 2] )
```

[Feature]

When each point of upper and lower limit value and trace A/B are given, it is checked whether the point level of trace side level stay inside range of upper and lower limit value.

[Format]

LMTUL2 (P, Up, Lo, M)

| | | |
|---|---|---|
| P | : | Point (0 to 700) |
| Up | : | Upper limit (Unit is the same as REF level.) |
| Lo | : | Lower limit (Unit is the same as REF level.) |
| M | : | Trace 0: Trace A |
| | | 1: Trace B |

Check whether it is (Lo $\leq$ level $\leq$ Up).

[Return Value]

| | | |
|---|---|---|
| Normal termination: | Inside .......... | 0 |
| | Above the upper limit .......... | 1 |
| | Under the lower limit .......... | 2 |

[Example]

To determine whether the 30MHz level of the trace A is between the input upper and lower limits.

```
INPUT "UPPER", U
INPUT "LOWER", L
P = POINT (30E6)
T = LMTUL2 (P, U, L, O)
IF T = 0 THEN PRINT "OK"
IF T = 1 THEN PRINT "UPPER"
IF T = 2 THEN PRINT "LOWER"
```

## POWER (Total Power)

[Feature]

Find out table electric power for REF level at 0d Bm. Frequency range is based on point width (point 1, point 2). Specification of vertical axis should be 10dB/div.

[Format]

POWER (P1, P2, M)

P1 : Specified point 1 (0 to 700)
P2 : Specified point 2 (0 to 700)
M : Trace 0: Trace A
1: Trace B

[Return Value]

Normal termination: Total power (W) between the specified points, P1 and P2.
Error interruption : Indefinite value.

[Error]

- If the specified value is out of the range, 0 to 700, an error will result.
- Return value will be indefinite.
- An error causes only an error message and not terminates the program.

[Note]

If P1 > P2, convert P1 and P2.

Calculate of the total power with the REF level set to other than 0 dBm is possible only when the REF level is set in the unit of the 10 dBm step. (that is, it is set to -20, -10, 0, 10, 20 dBm, etc.)

The equation for calculation is as follows:

Total power = power (P1, P2, M) * $10^{x/10}$ (x dBm)

[Example]

To calculate the total power between point 300 and 400 of the trace A frequency. (When REF level = 0 dBm)

AW = POWER (300, 400, 0)

To calculate the total power between point 0 and 700 of the trace A frequency. (When REF level = -20dBm)

AW = POWER (0,700, 0)/100

```
RTRACE (Trace data: [0 to 2720] )
```

[Feature]

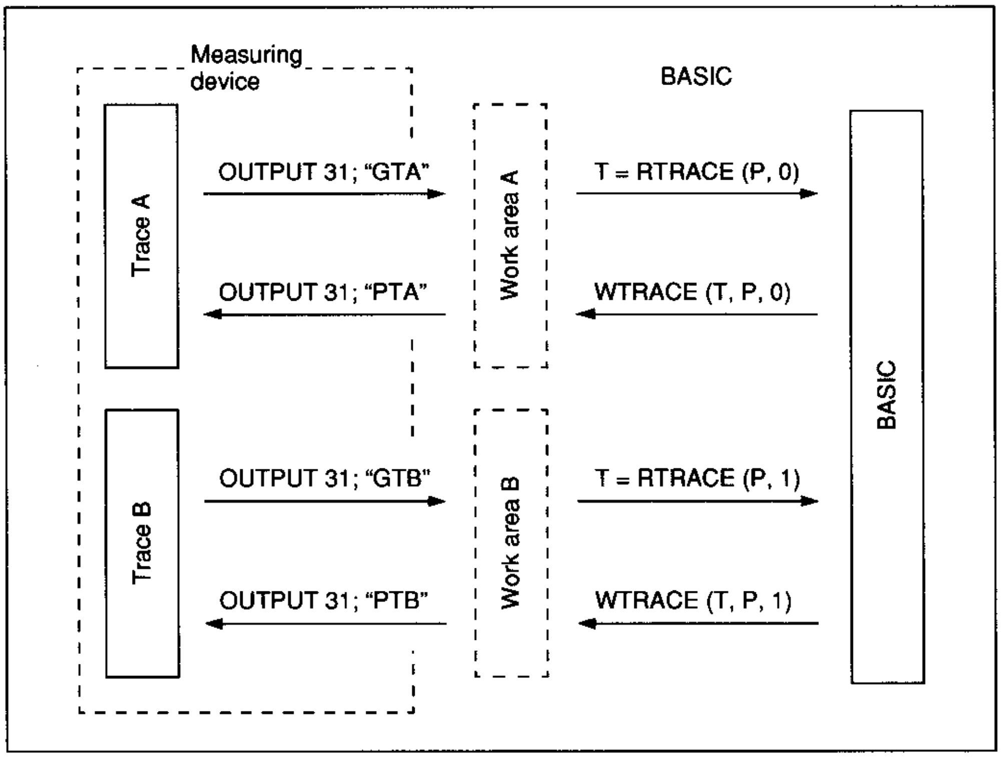
RTRACE returns the trace data of a specified point.

[Format]

RTRACE (P, M)

P : Point (0 to 700)
M : Trace 0: Trace A
         1: Trace B

Before this function is executed, execute "GTA" or "GTB" with OUTPUT 31 command. This makes trace data in a measuring device to be transferred to a work area.
To transfer trace A data to a work area, execute OUTPUT 31 for "GTA".
To transfer trace B data to a work area, execute OUTPUT 31 for "GTB".
The above both operations transfer all the 701 points.
Note that this RTRACE function is for reading data from a work area one by one point.



Trace data and BASIC data transfer

[Return Value]

| | |
|---|---|
| Normal termination: | Trace data (0 to 2720) |
| Error interruption : | Indefinite value. |

[Error]

Though to specify a point other than 0 to 700 causes no error, an indefinite value will be returned.

[Example]

To store the trace A data (0 to 700) into the array variable A.

```
INTEGER I, A (701)
OUTPUT 31; "GTA"
FOR I = 0 TO 700
  A (I+1) = RTRACE (I, 0)
NEXT I
```

## TRACE

[Feature]

The trace data is written to the designated point or the trace data is read from the designated point.

[Format]

TRACE (P, M)

```
P : Point (0 to 700)
M : Trace 0: Trace A
          1: Trace B
```

[Error]

If the numeric value except 0 to 700 is written to the designated point then the error is occurred.

[Note]

There are 0 to 2720 points for the trace data.

[Example]

```
OUTPUT 31; "GTA"       The data of the Trace A is copied to the work area.
FOR I = 0 TO 700
  A = TRACE (0, I)     The trace data is read from the work area.
NEXT I

FOR I = 0 TO 700
  TRACE (1, I) = 100   The data is copied to the work area.
NEXT I
OUTPUT 31; "PTB"       The data of the work area is copied in the data of the Trace B.
OUTPUT 31; "BV"        The trace B is designated for the VIEW.
```

```
WTRACE
```

[Feature]

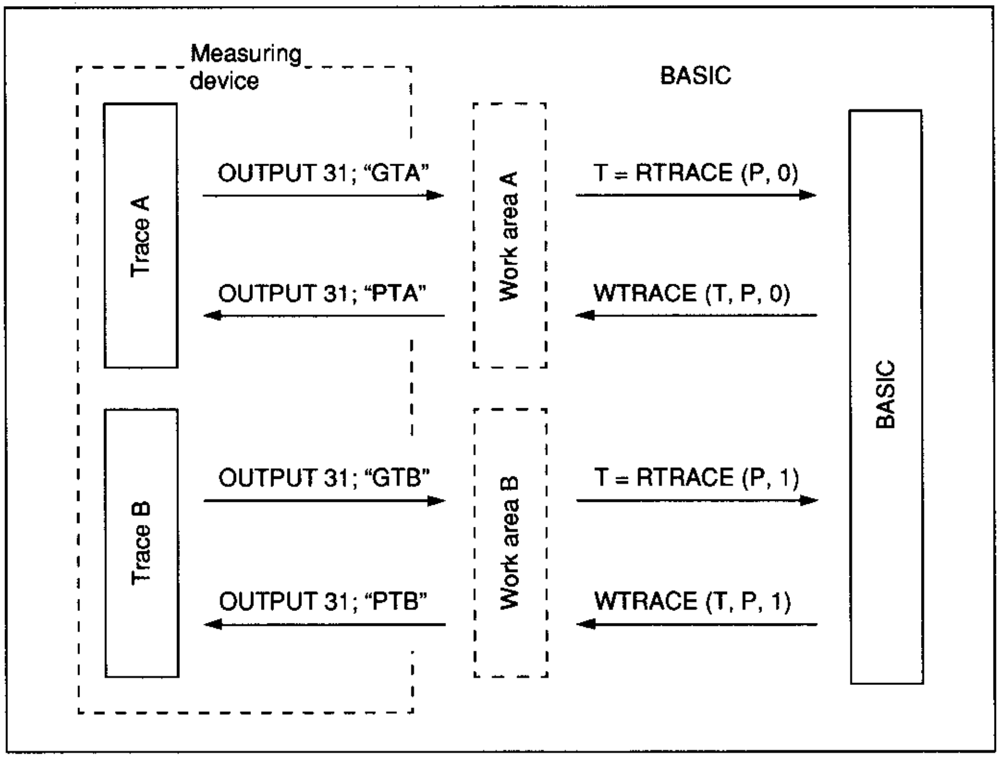
WTRACE writes the trace data into a specified point.

[Format]

WTRACE (T, P, M)

| | | |
|---|---|---|
| T : | Trace data (0 to 2720) | |
| P : | Point (0 to 700) | |
| M : | Trace 0: | Trace A |
| | 1: | Trace B |

After this function is executed, execute "PTA" or "PTB" with OUTPUT 31 command. This makes trace data in a work area to be transferred to a measuring device.
To write work area A data into the trace A, execute OUTPUT 31 for "PTA".
To write work area B data into the trace B, execute OUTPUT 31 for "PTB".
The above both operations write all the 701 points.
Note that this WTRACE function is for writing data into a work area one by one point.



Trace data and BASIC data transfer

[Error]

Though to specify a point other than 0 to 700 causes no error, no data will be written.

[Example]

To transfer the array variable B data to the trace B data (measuring device).
(The array variable B is 0.)

```
INTEGER I, B (701)
FOR I = 0 TO 700
  WTRACE (B (I+1), I, 1)
NEXT I
OUTPUT 31; "PTB"
```

## 4.4 Graphic Function

The graphic function will be described as the following order.

| Function |
|---|
| GADRS |
| GFLRECT |
| GLINE |
| GPOINT |
| GRECT |

GADRS

[Feature]

Address mode of the graphic-drawing screen is designated. There are absolute address and viewport address for the address mode.

[Format]

GADRS (Mo, X, Y)

- Mo : Mode
  0: Absolute address, 1: Viewport address
- X : Horizontal axis (0 to 720)
- Y : Vertical axis (0 to 512)

[Error]

- If X and Y is designated to the out of the graphic-drawing screen then the error is occurred. (Be careful when the viewport address is designated.)
- When the error is occurred, the function is not stopped but only message is output.

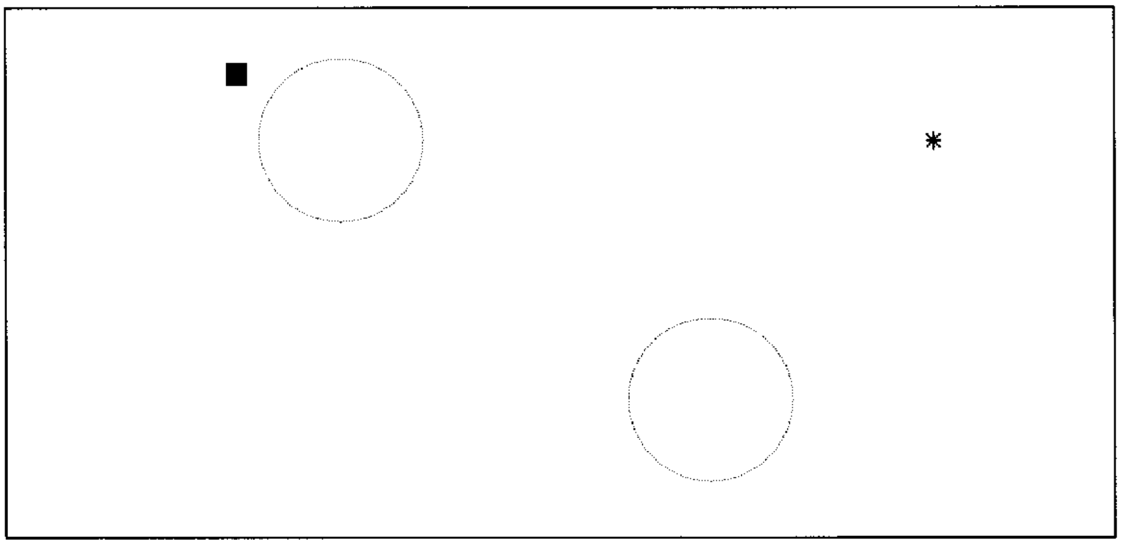
[Note]

- When absolute address mode is designated, origin becomes upper left and designation X and Y is omitted. When the viewport address is designated, the origin of X and Y becomes available.
- Designate absolute address for X and Y and do not exceed the X and Y range.
- When viewport address is designated, the upper right quadrant is used.

[Example]

```
SCREEN 3
PRINTER CRT
CLS 1
R = 100
OFFSET = 100
FOR M = 0 TO 1
  GADRS (M, 360, 256)
  FOR I = 0 TO PT*2 STEP PI/90
    X = SIN (I)*R+OFFSET
    Y = COS (I)*R+OFFSET
    GPOINT (1, X, Y)
  NEXT I
NEXT M
```

[Result]



GFLRECT

[Feature]

GFLRECT draw painted rectangular which has a diagonal between specified two points (coordinate 1, coordinate 2).

[Format]

```
GFLRECT (D,   X1,    Y1,     X2,    Y2)
          |   ============    ==========
          |              |     |
          └─0 : Erace    |     └──coordinate 2
            1 : Draw     └────────coordinate 1
```

[Error]

- An error will arise, if the X and Y are specified out of the graphic display area. (Take care when being in the viewport address mode.)
- Though the error outputs a message, does not interrupt a program execution.

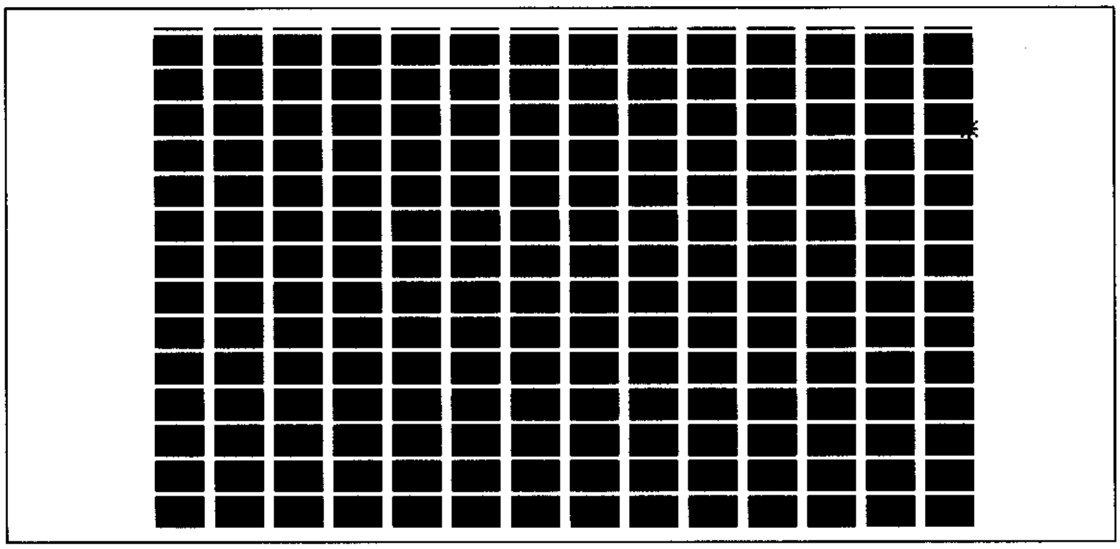
[Note]

Coordinate 1 must be specified at the upperleft, and coodinate 2 at the upper right.

[Example]

```
SCREEN 3
PRINTER CRT
  CLS 1
  INTEGER X, Y
  FOR X = 10 TO 700 STEP 50
    FOR Y = 10 TO 450 STEP 30
      GFLRECT (1, X, Y, X+40, Y+25)
    NEXT Y
  NEXT X
  STOP
```

[Result]



```
GLINE
```

[Feature]

A line is drawn between designated point and point (coordinate 1 and coordinate 2).

[Format]

GLINE (D, X1, Y1, X2, Y2)

- D: 0; Delete, 1; Draw
- X1, Y1: Coordinate 1
- X2, Y2: Coordinate 2

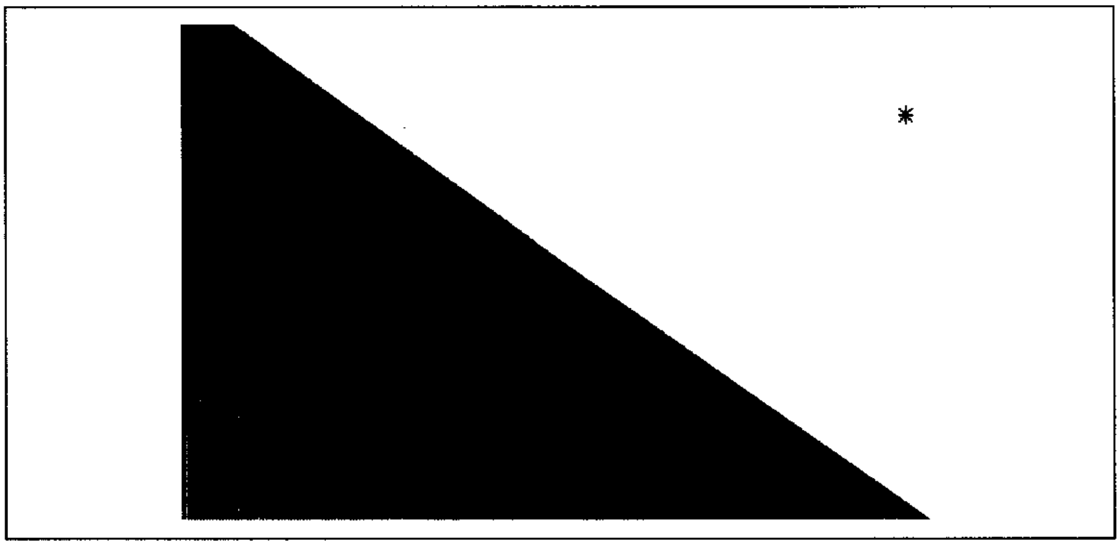
[Error]

- If X and Y are designated to the out of the graphic-drawing screen then the error is occurred. (Be careful when the viewport address is designated.)
- When the error is occurred, the function is not stopped but only message is output.

[Example]

```
SCREEN 3
PRINTER CRT
CLS 1
FOR I = 2 TO 720 STEP 3
  GLINE (1, 2, 0, I, 512)
  GLINE (1, 0, 0, I-2, 512)
  NEXT I
  STOP
```

[Result]



GPOINT

[Feature]

GPOINT dots a point at a specified position.

[Format]

```
GPOINT  (D,    X,    Y)
         |       |
         |       └── coordinate
         └────────── 0 : Erase
                     1 : Draw
```

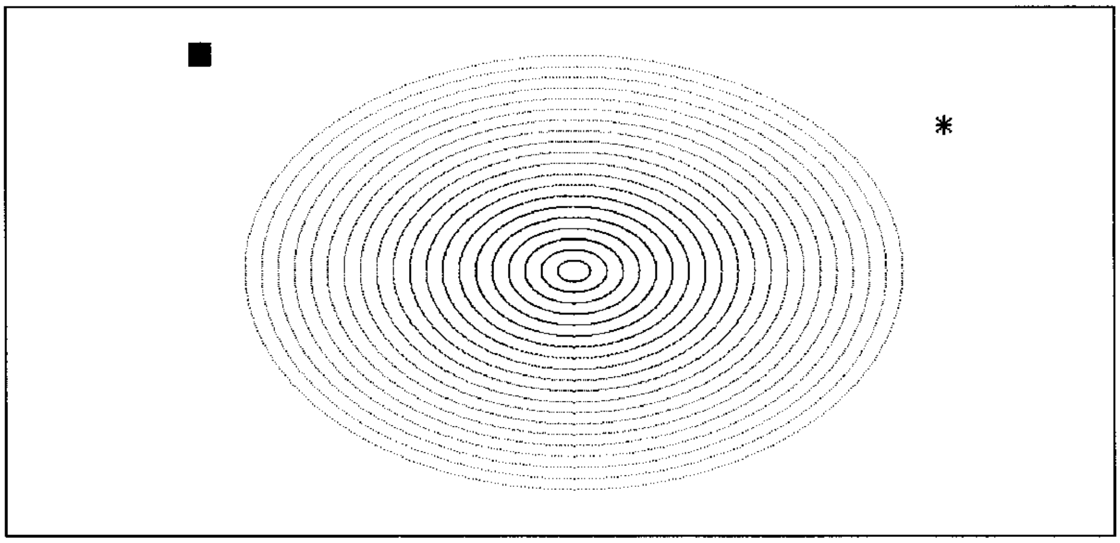
[Error]

- An error will arise, if the X and Y are specified out of the graphic display area. (Take care when being in the viewport address mode.)
- Though the error outputs a message, does not interrupt a program execution.

[Example]

```
SCREEN 3
PRINTER CRT
CLS 1
FOR R = 200 TO 1 STEP -10
  FOR I = 0 TO PI*2 STEP PI/270
    X = SIN (I) *R*1.5+360
    Y = COS (I) *R+256
    GPOINT (1, X, Y)
  NEXT I
NEXT R
```

[Result]



GRECT

[Feature]

The rectangle which diagonal line is designated by the point and point is drawn.

[Format]

GRECT (D, X1, Y1, X2, Y2)

- D:
  0 : Delete, 1 : Draw.
- X1, Y1 : Coordinate 1.
- X2, Y2: Coordinate 2.

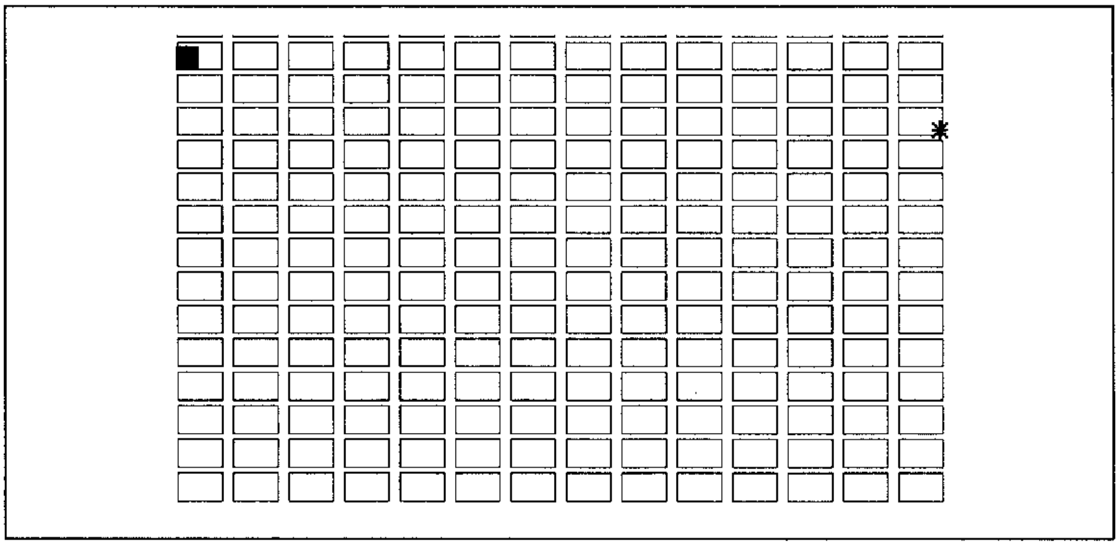
[Error]

- If X and Y are designated to the out of the graphic-drawing screen then the error is occurred. (Be careful when the viewport address is designated.)
- When the error is occurred, the function is not stopped but only message is output.

[Example]

```
SCREEN 3
PRINTER CRT
CLS 1
INTEGER X, Y
FOR X = 10 TO 700 STEP 50
  FOR Y = 10 TO 300 STEP 30
    GRECT (1, X, Y, X+40, Y+25)
  NEXT Y
NEXT X
STOP
```

[Result]



# 5 MASTER/SLAVE MODE

## 5.1 Outline

(1) Master Mode

ULIS is placed in a controller mode and can control itself and external GPIB units.

(2) Slave Mode

ULIS is not in the controller mode but placed in the mode to be controlled by an external controller. In this mode, ULIS controls itself but can not control external GPIB units. The ULIS basic program, however, enables data transfer to an external controller.
In addition to this, the external controller can start and stop the ULIS basic program.

## 5.2 BASIC Command of the Master/Slave Mode

The operation of the BASIC command varies depending on each of the master/slave modes.

| BASIC command | Master mode | Slave mode |
|---|---|---|
| CLEAR | Device clear | Not function *2 |
| DELIMITER | Setting of delimiter<br>Sets the delimiter of the OUTPUT, GPRINT, and GLIST. | |
| ENTER | Sets the specified unit to the talker and inputs the ASCII data from the GPIB port. | Inputs the ASCII data from the GPIB port when the external controller specifies the system as the listener. *1 |
| INTERFACE CLAR | Interface clear | Not function *2 |
| LOCAL | Local | Not function *2 |
| LOCAL LOCKOUT | Local lockout | Not function *2 |
| OUTPUT | Sets the specified unit to the listener and outputs the ASCII data from the GPIB port. | Outputs the ASCII data from the GPIB port when the external controller specifies the system as the talker. *1 |
| REMOTE | Remote | Not function *2 |
| REQUEST | Not function *2 | Outputs SRQ to the external controller. |
| SEND-DATA-CMD-TALK-LISTEN-UNT-UNL | Operation of ATN line | Not function *2 |
| SPOLL | Serial poll | Not function<br>Always returns "0". |
| TRIGGER | Trigger | Not function *2 |
| CONTROL | Switches over between the GPIB address and the master/slave in the BASIC program of the Controller function. | |

*1 : Ignores the addresses of the OUTPUT and ENTER commands in the slave mode.

*2 : "Not function" means that the program jumps to the next command without executing the indicated command.

## 5.3 GPIB Command of the Slave Mode

(1) OUTPUT

Changes the numeric representation to the ASCII data and sends it to the GPIB port when the external controller specifies the system to the listener.
The OUTPUT command format of the slave mode is identical with that of the master mode. However, the address of the GPIB is ignored.

(2) ENTER

Stores the input ASCII data into the specified variable when the external controller specifies the system to the talker.
The ENTER command format of the slave mode is identical with that of the master mode. However, the address of the GPIB is ignored.

***CAUTION!***
***After the OUTPUT and ENTER commands in the slave mode terminate their processing, they go to the next command. (The system stays in the command waiting condition until the external controller specifies the next command after the completion of command processing.)***

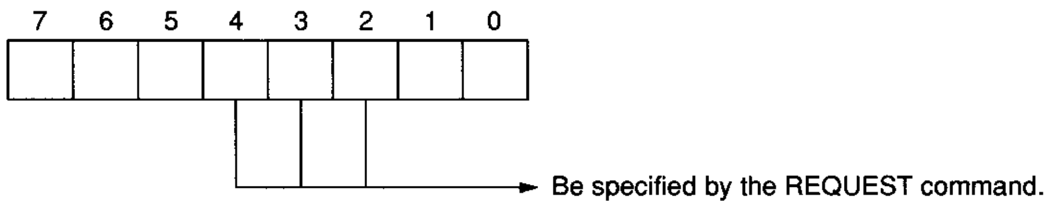
(3) REQUEST

Outputs the SRQ to the external controller.

REQUEST <value 0-7>

When sending RSQ interruption to the host.

Example: REQUEST 1



(4) Controller mode switchover command

The factory setting default values are:

| | | | |
|---|---|---|---|
| Controller function | GPIB address | : | 30 |
| | Master/slave mode | : | master mode |

(After the setting is changed, the value is kept effective.)

Switchover by BASIC command (CONTROL command is used.)

| | | |
|---|---|---|
| Change of GPIB address | : | CONTROL 4; [0 to 30] |
| Change of master/slave mode | : | CONTROL 5; [0, 1] |
| | | 0 : slave mode |
| | | 1 : master mode |

Example:

```
CONTROL 4; 5 .............. Changes to GPIB address 5.
CONTROL 5; 0 .............. Slave mode

CONTROL 4; 20 ............ Changes to GPIB address 20.
CONTROL 5; 1 .............. Master mode
```

## 5.4 Control From the External Controller (Slave Mode)

The following commands can be used to start the BASIC program of the U3641 Series controller from an external controller. (However, the OUTPUT and ENTER instructions must not be executing.)

| Command | Function | Remarks |
|---|---|---|
| @LOAD △ file name | Loads the program. | |
| @RUN | Executes the program. | |
| @STOP | Stops the program. | |
| @CONT | Resumes the program. | |
| @EXIT | Exits from the BASIC (editor) mode. | |
| @INFO? | Returns the U3641 Series controller status. | Recommended to use with SRQ |
| @INFOR △ CLR | Clears the U3641 Series controller status. | |
| @variable-name? | Outputs the variable value to the host in ASCII mode. | |
| @variable-name △ BIN? | Outputs the variable value to the host in binary mode. | Conforms to IEEE 64-bit floating point. |
| @variable-name △ MANY? | Outputs the array variable consisting of multiple lines to the host. | |

△ : Indicates a space.

Each command is placed without space after the @ mark. If the command is sent without the @ mark, the GPIB bus will be inoperable. (It recovers when the STOP key of the U3641 Series controller is pressed.)

***Note1: The above commands are not available while the OUTPUT or ENTER instruction is executing.***

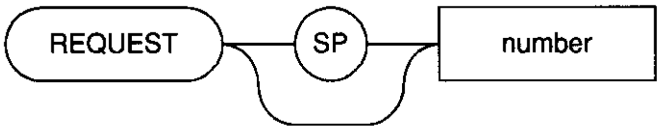
***Note2: For a program intended to communicate with the external controller, first start the program on the U3641 Series controller side manually or by using the above commands.***

## 5.5 ULIS Controller Control Commands of the Slave Mode

The BASIC program in this material are written by N88-BASIC, excluding some portions.

```
REQUEST
```

[Syntax]



[Description]

The REQUEST command is used together with the @INFO? command to notify the host computer of occurrence or completion of a user event. This command is executed on the ULIS controller. The following is an example of synchronization procedure.

1. When the REQUEST command is executed, the U3641 Series controller sends SRQ (Service Request) to the host computer.
2. The host computer issues the @INFO? command to request the bit information of controller status from the ULIS.
3. The ULIS sends the current controller status, i.e., the value indicated by the REQUEST command, back to the host computer.

[Parameters]

| | |
|---|---|
| number | Specifies a user event with a number from 0 to 7. This value is reflected to bit Nos. 2 to 4 of the ULIS controller status shown in Figure 5-1, and read out by the host computer using the @INFO? command. |

[Sample Program]

- Host computer side

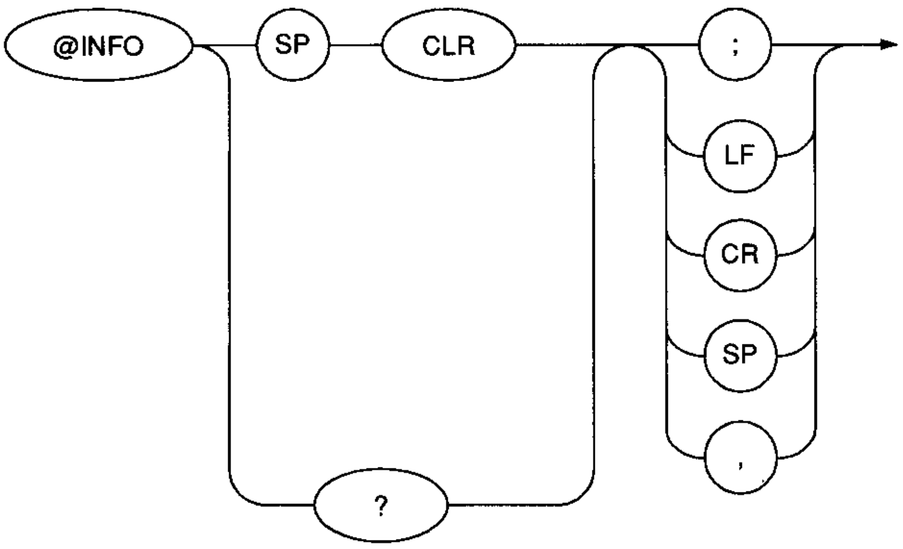
```
1000   ISET IFC:ISET REN
1010   SPA=8
1020   PRINT @SPA;"S0;RQS128"           ' Declares use of SRQ from controller.
1030   PRINT @SPA;"S2"
1040   POLL SPA,S
1050   ON SRQ GOSUB *READSRQ
1060   SRQ ON                           ' Enables SRQ interrupt.
1070   PRINT @SPA;"@RUN"                ' Activates ULIS controller.
1080   EVENT=0
1090 *READXMAX
1100   IF EVENT=0 THEN GOTO *READXMAX   ' Waits for end of processing of ULIS controller.
1110   PRINT @SPA;"@XMAX?"              ' Requests read-out of variable XMAX.
1120   INPUT @SPA;XMAX                  ' Reads out variable XMAX.
1130   PRINT XMAX
1140   PRINT @SPA;"@INFO CLR"           ' Clears ULIS controller status.
1150   PRINT @SPA;"@CR"                 ' Informs ULIS controller of end of read-out.
1160   EVENT=0
1170   GOTO *READXMAX
1180 '
1190 *READSRQ
1200   POLL SPA,S
1210   PRINT @SPA;"@INFO?"              ' Reads out ULIS controller status and confirms
                                          end of processing.
1220   INPUT @SPA;INFO
1230   IF (INFO AND 4)=4 THEN EVENT=1
1240   SRQ ON:RETURN
1250   END
```

- ULIS controller side

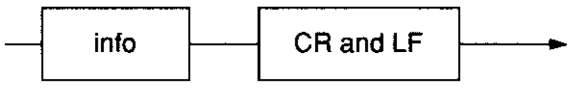
```
SCREEN 1
*MEAS_START
  OUTPUT 31;"TS"                 ! take sweep
  XMAX=MAX(0,700,0)              ! Searches for maximum level position.
  REQUEST 1                      ! Informs end of MAX function execution.
  INPUT TIMING                   ! Waits for end of read-out from host.
  CURSOR 0,0
  GOTO *MEAS_START
```

```
@INFO?(@INFO CLR)
```

[Syntax]



[Query Response]



[Description]

The @INFO? command returns the internal status of the ULIS controller to the host computer. The bit information of the internal status is as follows:

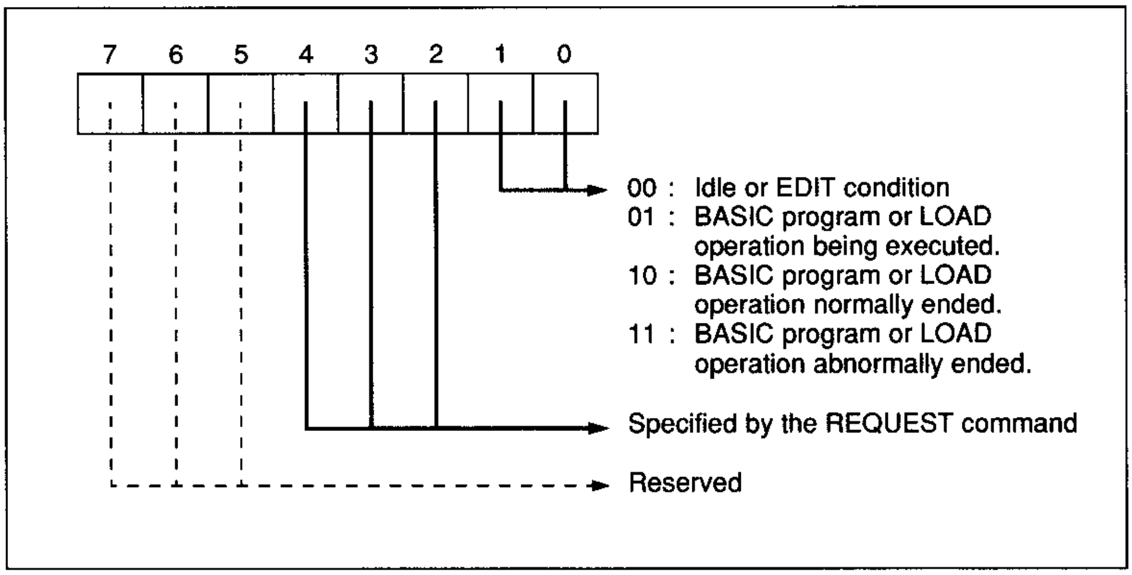

7 6 5 4 3 2 1 0
00 : Idle or EDIT condition
01 : BASIC program or LOAD operation being executed.
10 : BASIC program or LOAD operation normally ended.
11 : BASIC program or LOAD operation abnormally ended.
Specified by the REQUEST command
Reserved

**Figure 5-1 Bit Assignment of the ULIS Controller Status**

The @INFO CLR command clears the bit specified by the REQUEST command.
The bits Nos 0 and 1 which indicate BASIC program execution status are not affected.

[Parameters]

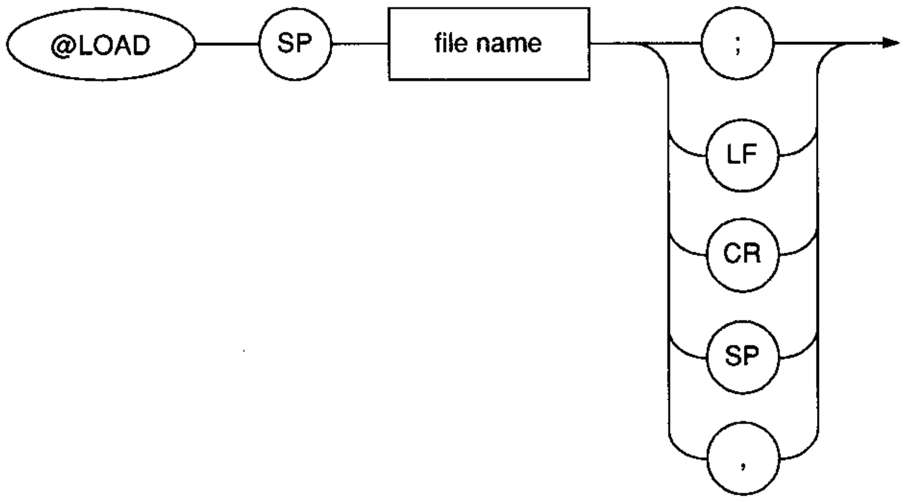
info Returns the internal status of the ULIS controller.

[Sample Program]

See the REQUEST command.

```
@LOAD
```

[Syntax]



[Description]

The @LOAD command loads the specified BASIC program from an IC card.

[Parameters]

| | |
|---|---|
| file name | Specifies the name of the BASIC program to be loaded. |

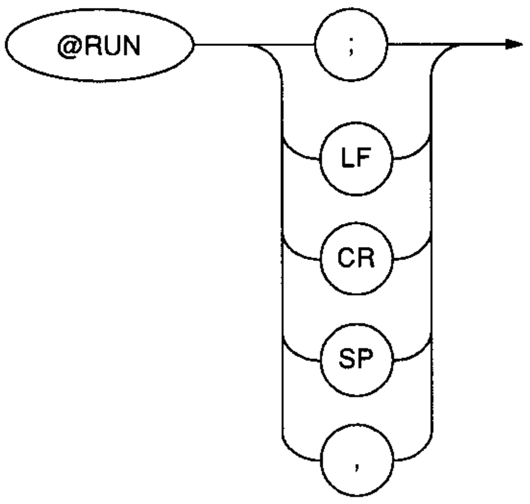
[Sample Program]

- Host computer side

```
1000   ISET IFC:ISET REN
1010   SPA=8:LOADEND=0
1020   PRINT @SPA;"S0;RQS128"          ' Declares use of SRQ from controller.
1030   PRINT @SPA;"S2"
1040   PRINT @SPA;"@INFO CLR;@INFO?"
1050   PRINT @SPA;"@LOAD XMAX"         ' Request loading of file XMAX from card.
1060   ON SRQ GOSUB *READSRQ
1070   SRQ ON
1080  *WAITEND
1090   IF LOADEND=0 THEN GOTO *WAITEND
                                       ' Waits for end of loading.
1100  *READINFO
1110   INPUT @SPA;INFO                 ' When loading is completed, returns 2; if an error occurs,
                                         returns 3.
1120   IF (INFO AND 3)=3 THEN PRINT "Load Error !!":STOP
1130   IF (INFO AND 3)<>2 THEN GOTO *READINFO
1140   PRINT @SPA;"@RUN"               ' When loading ended, requests execution.
1150   STOP
1160   '
1170  *READSRQ
1180   POLL SPA,S
1190   LOADEND=1
1200   RETURN
1210   END
```

```
@RUN
(@CNTRLR/@CONT/@CR/@EXIT/@STOP/@SF[1-7]/@numeric)
```

[Syntax]

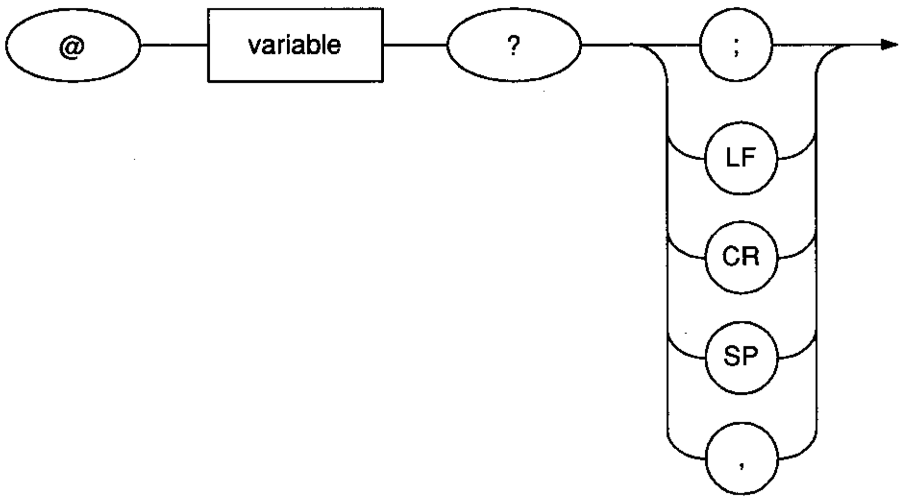


[Description]

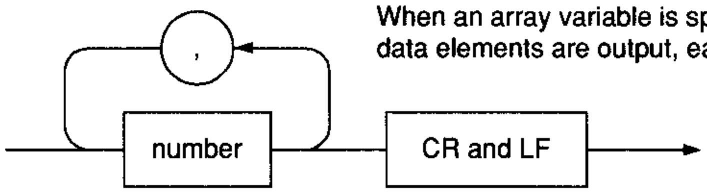
The @RUN command executes the BASIC program loaded on the ULIS controller.
The @CNTRLR command displays the user-defined menu only during execution of the BASIC program.
The @CONT command continues execution of the stopped BASIC program.
The @EXIT command returns the ULIS from the controller mode to the spectrum mode.
The @STOP command stops the BASIC program currently being executed.
The @SF [1-7] command executes the user-defined menu only during execution of the BASIC program.

```
@variable?
```

[Syntax]



[Query Response]



When an array variable is specified, values of all data elements are output, each delimited by a comma.

[Description]

The @variable? command returns the value of the specified variable in the BASIC program executed on the ULIS controller. For integer-type variables, an 11-digit value is returned; for real-type variables, a 23-digit value with a 15-digit mantissa is returned.
When an array variable is specified, values of all data elements are output in succession, each delimited by a comma.

[Parameters]

| | |
|---|---|
| variable | Specifies the name of the variable to be read out. For array variables, supply "(*)" to the end of the variable name. |
| | example) PRINT @8; "@ABC (*)?" |

[Sample Program]

- Host computer side

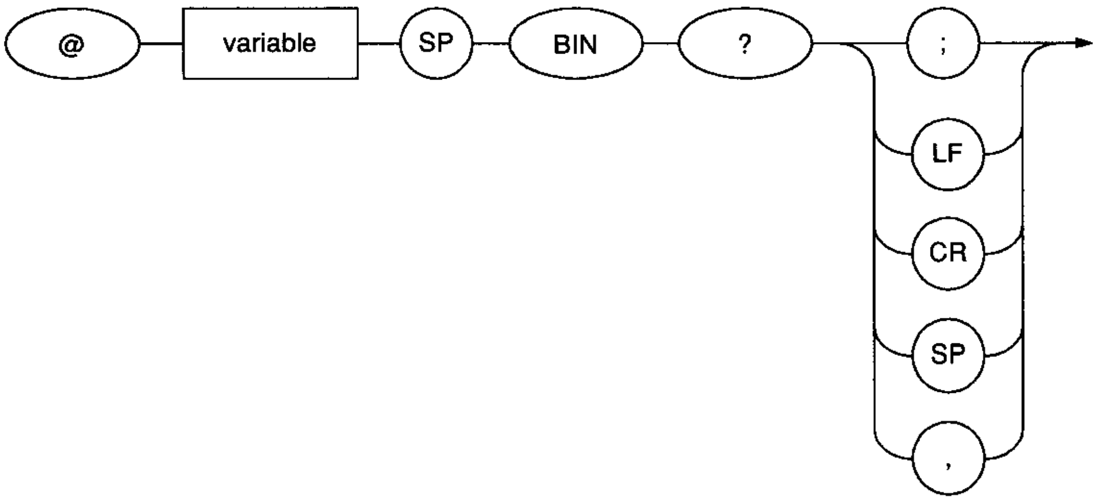
```
0000   ISET IFC:ISET REN
1010   DIM ABC(8)
1020   SPA=8
1030   PRINT @SPA;"S0";RQS128
1040   PRINT @SPA;"S2"
1050   PRINT @SPA;"@RUN"
1060   POLL SPA,S
1070   ON SRQ GOSUB *READSRQ
1080   PRINT @SPA;"@INFO CLR"
1090   EVENT=0
1100   SRQ ON
1110 *READXMAX
1120   IF EVENT=0 THEN GOTO *READXMAX ' Waits for end of measurement.
1130   PRINT @SPA;"@ABC(*)?"          ' Requests read-out of array variable ABC.
                                      ' Reads out CSV-format data.
1140   INPUT @SPA;ABC(1),ABC(2),ABC(3),ABC(4),ABC(5),ABC(6),ABC(7),ABC(8)
1150   FOR I=1 TO 8
1160     PRINT ABC(I)
1170   NEXT I
1180   STOP
1190 '
1200 *READSRQ
1210   POLL SPA,S
1220   PRINT @SPA;"@INFO?"
1230   INPUT @SPA;INFO
1240   IF (INFO AND 8)=8 THEN EVENT=1
1250   SRQ ON:RETURN
1260 '
1270 END
```

- ULIS controller side

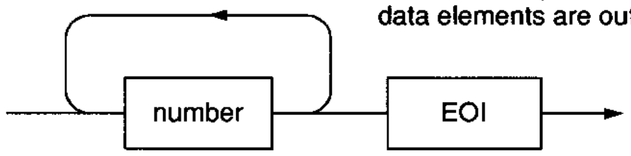
```
DIM ABC(8)
!
SCREEN 1
FOR I=1 TO 8
  OUTPUT 31;"TS"                ! take sweep
  ABC(I)=MAX(0,700,0)           ! Searches for the maximum level position.
NEXT I
REQUEST 2                     ! Informs end of execution.
END
```

```
@variable BIN?
```

[Syntax]



[Query Response]



When an array variable is specified, values of all data elements are output, each delimited by a comma.

[Description]

The @variable BIN? command returns the value of the specified variable in the BASIC program executed on the ULIS controller in binary form.
For integer-type variables, a 32-bit value is returned; for real-type variables, a 64-bit value with the IEEE floating point format is returned.
The EOI signal is output following the last byte.
When an array variable is specified, values of all data elements are output in succession, each delimited by a comma.
For host computers not recognizing the IEEE 64-bit floating point format, it is necessary to convert the format into an internal format after read-out of data.

[Parameters]

| | |
|---|---|
| variable | Specifies the name of the variable to be read out. For array variables, supply "(*)" to the end of the variable name. |
| | example) PRINT @8; "@ABC (*)?" |

[Sample Program]

The following sample program is written by HP-BASIC and therefore cannot be executed by N88-BASIC.

- Host computer side

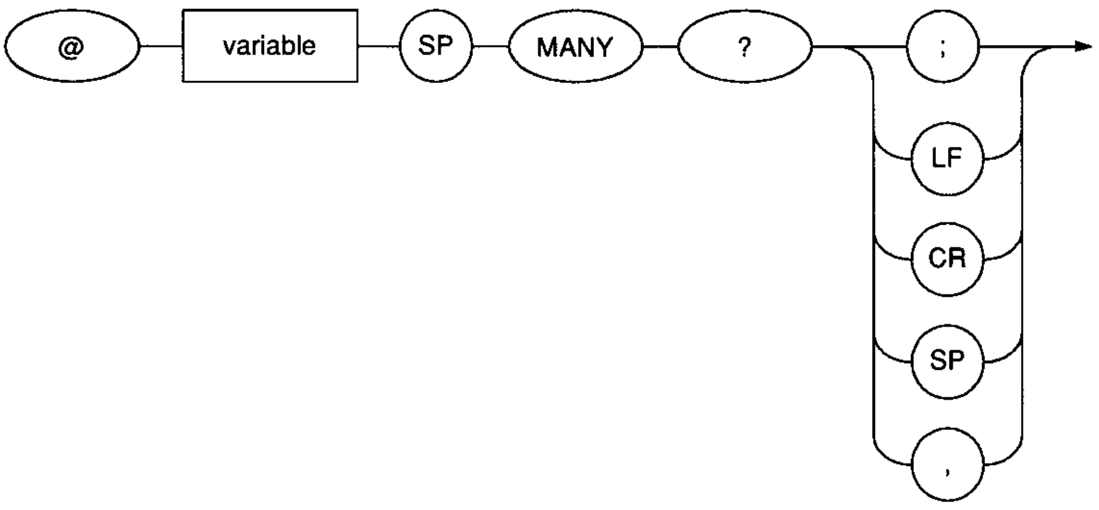
```
1000   OPTION BASE 1
1010   Spa=708
1020   ASSIGN @Device TO Spa;FORMAT OFF   ! Specifies IEEE 64-bit floating point format.
1030   DIM Abc(8)
1040 !
1050   ON INTR 7 GOSUB Srq_intr
1060   OUTPUT Spa;"S0;S2;RQS128"
1070   S=SPOLL(Spa)
1080   OUTPUT Spa;"@RUN"
1090   Event=0
1100   OUTPUT Spa;"@INFO CLR"
1110   ENABLE INTR 7;2
1120 !
1130 Wait_end:!
1140   IF Event=0 THEN GOTO Wait_end      ! Waits for end of measurement.
1150   OUTPUT Spa;"DL2"                   ! Requests EOI as delimiter.
1160   OUTPUT Spa;"@ABC(*) BIN?"          ! Requests read-out of array variable ABC in
                                            binary form.
1170   ENTER @Device;Abc(*)               ! Reads out all data of array variable at one time.
1180   PRINT Abc(*)
1190   STOP
1200 !
1210 Srq_intr:!
1220   S=SPOLL(Spa)
1230   OUTPUT Spa;"@INFO?"
1240   ENTER Spa;Info
1250   IF BINAND(Info,8)=8 THEN Event=1
1260   ENABLE INTR 7;2
1270   RETURN
1280 !!
1290   END
```

- ULIS controller side

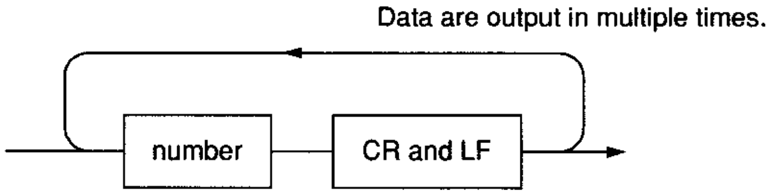
```
DIM ABC(8)
!
SCREEN 1
FOR I=1 TO 8
  OUTPUT 31;"TS"                        ! take sweep
  ABC(I)=MAX(0,700,0)                   ! Searches for the maximum level position.
NEXT I
REQUEST 2                               ! Informs end of execution.
END
```

```
@variable MANY?
```

[Syntax]



[Query Response]



[Description]

The @variable MANY? command returns the value of the specified variable in the BASIC program executed on the ULIS controller in multiple times. For integer-type variables, an 11-digit value is returned; for real-type variables, a 23-digit value with a 15-digit mantissa is returned.
A terminator is appended at the end of data.
The host computer reads out data elements of such an array variable using the FOR statement.
This command is effective only for output of data elements of an array variable.

[Parameters]

| | |
|---|---|
| variable | Specifies the name of the variable to be read out. Since this command is effective only for array variables, supply "(*)" to the end of the variable name.<br>example) PRINT @8; "@ABC (*)?" |

[Sample Program]

- Host computer side

```
1000   ISET IFC:ISET REN
1010   DIM ABC (8)
1020   SPA = 8
1030   PRINT @SPA;"S0;RQS128"
1040   PRINT @SPA;"S2"
1050   PRINT @SPA;"@RUN"
1060   POLL SPA,S
1070   ON SRQ GOSUB *READSRQ
1080   PRINT @SPA;"@INFO CLR"
1090   EVENT=0
1100   SRQ ON
1110 *READXMAX
1120   IF EVENT=0 THEN GOTO *READXMAX   ' Waits for end of measurement.
1130   PRINT @SPA;"@ABC(*) MANY?"       ' Requests read-out of array variable  ABC in
                                          multiple times.
                                        ' Reads out in units of data element.
1140   FOR I=1 TO 8
1150     INPUT @SPA;ABC(I)
1160     PRINT ABC(I)
1170   NEXT I
1180   STOP
1190 '
1200 *READSRQ
1210   POLL SPA,S
1220   PRINT @SPA;"@INFO?"
1230   INPUT @SPA;INFO
1240   IF (INFO AND 8)=8 THEN EVENT=1
1250   SRQ ON:RETURN
1260 '
1270 END
```

- ULIS controller side

```
DIM ABC(8)
!
SCREEN 1
FOR I=1 TO 8
  OUTPUT 31;"TS"               ! take sweep
  ABC(I)=MAX(0,700,0)          ! Searches for the maximum level position.
NEXT I
REQUEST 2                      ! Informs end of execution.
END
```

# APPENDIX

## A.1 List of Commands and Statements

| Function | Command and Statement | Description |
|---|---|---|
| ① Commands | CONT | Resumes the execution of the program after it has stopped. |
| | CONTROL | Sets values for each control. |
| | EDIT | Stards the ate editor function. |
| | LIST | Displays a program list. |
| | LLIST | Displays a program list. (RS-232) |
| | RUN | Executes a program. |
| ② Arithmetic functions | ABS | Produces the absolute value of the given value. |
| | ATN | Produces the arc tangent of the given value. |
| | COS | Produces the cosine of the given value. |
| | LOG | Produces the natural logarithm of the given value. |
| | SIN | Produces the sine of the given value. |
| | SQR | Produces the square root of the given value. |
| | TAN | Produces the tangent of the given value. |
| ③ Bitwise operations | BAND | Produces a bit AND. |
| | BNOT | Produces a bit NOT. |
| | BOR | Produces a bit OR. |
| | BXOR | Produces a bit XOR. |
| ④ Date/Time controls | TIME$ | Returns the present time (hour/minute/second) by character string. |
| | DATE$ | Returns the date (year/month/day) by character string. |
| ⑤ Interrupt controls | ENABLE INTR | Enables interrupts to be received. |
| | DISABLE INTR | Disables interrupts to be received. |
| | ON END | Defines the branch of an EOF interrupt. |
| | ON KEY | Defines the branch of a key interrupt. |
| | ON ISRQ | Defines the branch of a SRQ interrupt of the main measurement section. |
| | ON SRQ | Defines the branch of a SRQ interrupt of GPIB. |
| | ON ERROR | Defines the branch of an error occurrence interrupt. |
| | OFF END | Releases the branch of an EOF interrupt. |
| | OFF KEY | Releases the branch of a key interrupt. |
| | OFF ISRQ | Releases the branch of an SRQ interrupt of the main measurement section. |

(cont'd)

| Function | Command and Statement | Description |
|---|---|---|
| ⑤ Interrupt controls (cont'd) | OFF SRQ<br>OFF ERROR | Releases the branch of an SRQ interrupt of GPIB.<br>Releases the branch of an error occurrence interrupt. |
| ⑥ Character-string manipulation | NUM<br><br>CHR$<br>LEN<br>POS<br>SPRINTF | Converts the first character of a character string to the ASCII code.<br>Converts a numeric character to an ASCII character.<br>Obtains the length of a character string.<br>Positions character string 1 in character string 2.<br>Formats a string and enters it to a character-string variable. |
| ⑦ Memory card controls | CAT<br>CALL<br>CLOSE #<br>COPY<br>DIR<br>ENTER #<br>OPEN #<br>OUTPUT #<br>INITIALIZE (or INIT)<br>PURGE<br>RENAME | Displays the contents of a memory card.<br>Loads sub-program.<br>Close a file.<br>Copies the file in the memory card.<br>Outputs the name of stored file in the memory card.<br>Reads data from a file.<br>Open a file.<br>Writes data into a file.<br>Initializes a memory card.<br>Deletes the specified file.<br>Rename a file. |
| ⑧ Screen controls | CURSOR (or CSR)<br>CLS<br>SCREEN<br>KEY ON<br>KEY OFF<br>PANEL | Moves the cursor to the specified position.<br>Clears the screen.<br>Sets up the basic screen.<br>Displays the user-defined menu.<br>Clears (deletes) the user-defined menu.<br>Enables/Disables the panel key control. |
| ⑨ Statements | BUZZER<br>DIM<br>FOR TO STEP<br>NEXT<br>BREAK<br>CONTINUE<br>GOSUB<br>RETURN<br>GOTO<br>IF THEN ELSE<br>END IF<br>INPUT (or INP)<br>INTEGER | Sounds the buzzer.<br>Declares array variables.<br>Sets an iteration.<br><br>Exits from the current iteration.<br>Returns to the beginning of the current iteration.<br>Branches to a subroutine.<br>Returns from the current subroutine.<br>Branches to the specified statement.<br>Executes the statements after evaluating conditions.<br><br>Inputs a value to a variable.<br>Declares integer-type variables. |

(cont'd)

| Function | Command and Statement | Description |
|---|---|---|
| ⑨ Statements (cont'd) | LPRINT | Outputs data on a printer (RS-232). |
| | LPRINT USING (or USE) | Outputs formatted data on a printer (RS-232). |
| | MENU | Displays the arbitrary character string to the sort menu and waits until precessing the soft key. |
| | PAUSE | Stops execution temporarily. |
| | PRINT (or ?) | Displays characters on the screen. |
| | PRINT USING (or USE) | Displays formatted characters on the screen. |
| | PRINTER | Addresses a GPIB printer device. |
| | PRINTF (or PRF) | Displays formatted characters on the screen. |
| | READ DATA | Reads data from a DATA statement and assigns it to a variable. |
| | RESTORE | Specifies the DATA statement to read by the READ DATA statement. |
| | REM (or !) | Provides a comments. |
| | SELECT CASE END SELCT | Executes statements after evaluating condition. |
| | STOP | Stops program execution. |
| | WAIT | Holds the program execution for the specified period. |
| ⑩ GPIB commands | CLEAR | Transfers DCL and SDC. |
| | DELIMITER | Sets a delimiter. |
| | ENTER (or ENT) | Inputs GPIB data. |
| | GLIST | Outputs the program list to a GPIB printer. |
| | GPRINT | Outputs data to a GPIB printer. |
| | GPRINT USING (or USE) | Outputs formatted data to a GPIB printer. |
| | INTERFACE | Transfers IFC. |
| | CLEAR LOCAL | Places the specified device in the local status. |
| | LOCAL LOCKOUT | Places the specified device in the local locked-out status. |
| | OUTPUT (or OUT) | Outputs data to GPIB. |
| | REMOTE | Places the specified device in the remote status. |
| | REQUEST | Outputs SRQ to the standard GPIB. |
| | SEND | Outputs a set of GPIB data. |
| | SPOLL | Provides serial polling to the specified device. |
| | TRIGGER | Outputs GET. |

## A.2 List of Built-in Function

| Function | Structure | Description |
|---|---|---|
| Frequency/ point operation | F = FREQ (P) | Calculates a frequency from a point value. |
| | F = DFREQ (P1, P2) | Calculates a frequecny from a width between points. |
| | P = POINT (F) | Calculates a point (horizontal axis). |
| | P = DPOINT (F1, F2) | Calculates a point between specified frequencies. |
| Level/point operation | L = LEVEL (T) | Calculates a level of trace data specified with a point value (vertical axis). |
| | L = DLEVEL (T1, T2) | Calculates a level between points (vertical axis). |
| | T = LVPOINT (L) | Calculates the point (vertical axis) of trace data from a level. |
| | T = LVDPOINT (L1, L2) | Calculates the point (vertical axis) of trace data between levels. |
| | L = VALUE (P, M) | Calculates a level at a frequency position specified with a point value. |
| | L = DVALUE (P1, P2, M) | Calculates a difference of levels at two frequency positions specified with point values. |
| | L = CVALUE (F, M) | Calculates a level at a frequency position. |
| | L = DCVALUE (F1, F2, M) | Calculates the level difference between two specified frequency positions. |
| Maximum/ minimum operation | F = FMAX (P1, P2, M) | Calculates the frequency at the maximum level position between two positions specified with point value. |
| | F = FMIN (P1, P2, M) | Calculates the frequency at the minimum level position between two positions specified with point value. |
| | P = PMAX (P1, P2, M) | Calculates the frequency point (horizontal axis) maximum level position between two positions at the specified with point values. |
| | P = PMIN (P1, P2, M) | Calculates the frequency point (horizontal axis) at the minimum level position between two positions specified with point values. |
| | L = MAX (P1, P2, M) | Calculates the maximum level between two positions specified with point values. |
| | L = MIN (P1, P2, M) | Calculates the minimum level between two positions specified with point values. |

| Function | Structure | Description |
|---|---|---|
| Bandwidth operation | F = BND (P, X, M) | Calculates the frequency bandwidth of a LOSS level at a position specified with point values. |
| | F = BNDL (P, X, M) | Calculates the low frequency bandwidth of a LOSS level at a position specified with point values. |
| | F = BNDH (P, X, M) | Calculates the high frequency bandwidth of a LOSS level at a position specified with point values. |
| | F = CBND (F, X, M) | Calculates the frequency bandwidth of a LOSS level at a position specified with a frequency. |
| | F = CBNDL (F, X, M) | Calculates the low frequency bandwidth of a LOSS level at a position specified with a frequency. |
| | F = CBNDH (F, X, M) | Calculates the high frequency bandwidth of a LOSS level at a position specified with a frequency. |
| Maximum/ minimum operation | N = NRPLH (P1, P2, Dx, Dy, M) | Set the number of every maximum point. |
| | N = NRPLL (P1, P2, Dx, Dy, M) | Set the number of every minimum point. |
| | P = PRPLHN (N, M) | Set horizontal axis point for maximum point of N's turn from the left. |
| | P = PRPLLN (N, M) | Set horizontal axis point for minimum point of N's turn from the left. |
| | F = FRPLHN (N, M) | Set frequency for maximum point of N's turn from the left. |
| | F = FRPLLN (N, M) | Set frequency for minimum point of N's turn from the left. |
| | L = VRPLHN (N, M) | Set level for maximum point of N's turn from the left. |
| | L = VRPLLN (N, M) | Set level for minimum point of N's turn from the left. |
| | L = RPL1 (P1, P2, Dx, Dy, M) | Set level difference between maximum value of maximum point and minimum value of minimum point. |

| Function | Structure | Description |
|---|---|---|
| Decision for upper and lower limit | C = LMTMD1 (Dd, S, Ds)<br>C = LMTMD2 (P, S, Ds, M)<br>C = LMTUL1 (Dd, Up, Lo)<br>C = LMTUL2 (P, Up, Lo, M) | Decide specified data with standard value and width of upper and lower.<br>Decide waveform data for horizontal axis-point position with standard value and width of upper and lower.<br>Decide specified data with upper limit value and lower limit value.<br>Decide waveform data for horizontal axis-point position with upper limit value and lower limit value. |
| Electric power operation | W = POWER (P1, P2, M) | Set total electric power between horizontal axis points. |
| Read/write of trace data | TRACE (M, P)<br>T = RTRACE (P, M)<br>WTRACE (T, P, M)<br>***Note: This function returns no value.*** | Read or Write trace data for appointed point.<br>Read trace data for appointed point.<br>Write trace data for appointed point. |
| Graphic | GADRS (Mo, X, Y)<br>GFLRECT (D, X1, Y1, X2, Y2)<br>GLINE (D, X1, Y1, X2, Y2)<br>GPOINT (D, X, Y)<br>GRECT (D, X1, Y1, X2, Y2) | Specify absolute address/view-point address for graphic point.<br>Paint out rectangle which includes diagonal between appointed 2 poinsts.<br>Draw line between appointed 2 points.<br>Draw dot at appointed position.<br>Draw rectangle which includes diagonal between appointed 2 points. |

| | | | | | |
|---|---|---|---|---|---|
| F | : | Frequency (Hz) | Dx | : | Differential coefficient in horizontal axis |
| P | : | Point (0 to 700) | Dy | : | Differential coefficient in vertial axis |
| L | : | Level | S | : | Reference value |
| T | : | Trace data (0 to 2720) | Ds | : | Upper and lower limit width |
| M | : | Kind of Trace<br>0;Trace A, 1;Trace B | Dd | : | Sample data |
| | | | Lo | : | Lower limit |
| X | : | Loss level (dB) | Up | : | Upper limit |
| C | : | Check value [0, 1, 2]<br>0;Inside, 1;Above the upper limit, 2;Under the lower limit | | | |
| N | : | Ripple number | | | |
| W | : | Power | | | |

For Graphic:

| | | | |
|---|---|---|---|
| D | : | Set/Erase | 0;Erase, 1;Set (Draw) |
| Mo | : | Address mode | 0;Absolute address, 1;Viewport address |
| X | : | Coordinate (Horizontal axis) | |
| Y | : | Coordinate (Vertical axis) | |

## A.3 List of Error Message

(1) Ate editor error message list

***Note:*** ○○ : ***Byte number***
△△ : ***Line number***
***xxx*** : ***Character strings***
***yy*** : ***Numeric***

| Error Message | Description |
|---|---|
| Already auto line no. mode | Try to duplicate setting for the auto line numbering mode. |
| Cannot allocate ○○ bytes | No memory exists in the editor. |
| Cannot allocate memory | Cannot renumber for no memory remains in the editor. |
| Cannot allocate WINDOW block | Cannot open a new window for no memory remains in the editor. |
| Cannot create buffer | No memory is allocated space in the editor. |
| Cannot find error line | No error line when BASIC is executed is in the program in the ate editor. |
| Cannot open file for writing | Cannot open a file for no memory card or WRITE PROTECT is on. |
| Cannot sprit a △△ line window | Cannot spirit for window lines are insufficient. |
| Line no. is out of range | Line numbers are over 65535. |
| No file name | File has no name when saved into a memory card. |
| No mark set in this window | No region specification mark for delete or copy is specified. |
| Not found | No character strings for retrieval is specified. |
| Not line no. mode | Renumbering was executed without no auto line numbering mode. |
| Too large region | Too large region for delete or copy. |
| Write I/O error | Cannot access a memory card for no memory card or out of battery. |

(2) System controller error message list

| Error Message | Description |
|---|---|
| yy error(s) appeared | Error for label or line number. |
| "xxx" file cannot be opened. | No file to open exist. |
| "xxx" file is already opened with another PATH. | Try duplicate file opening. |
| "xxx" file is already exist. | Try a different instruction from the mode used at opening. |
| "xxx" read error. | Reading error. |
| 0 divide | Divided by 0. |
| Array's range error | The subscript of an array variable is bigger than a declaration. |
| Bad free call | Error on memory management. |
| CANNOT assigned into this token | Cannot assign into character variable. |
| cannot read data from "xxx" file. | No file exists to read. |
| cannot specify "USING" | Read byte number is different from write byte number. |
| cannot write data into "xxx" file. | No file exists to write. |
| end of "xxx" file | Read up to the EOF (end of file). |
| expression format error | Invalid format. |
| file format error | No terminator exists in 256 characters. |
| file is NOT open. | Read and write are executed without opening. |
| FOR <init value> does NOT exist | No initial value exists in a FOR statement. |
| FOR variable does NOT exist. | No counter variable exists in a FOR statement. |
| FOR's next is abnormal. | Cannot nest a FOR statement. |
| Invalid dimension parameter | Invalid parameter exists in an array variable. |

(cont'd)

| Error Message | Description |
|---|---|
| Invalid string constant | Double quotation marks do not match each other. |
| Invalid type in xxx | Invalid type are detected in xxx. |
| label not found | No specified label exists. |
| Label xxx is already exists | Try a duplicate specification of label xxx. |
| Line No. yy is out of range. | Line number assignment is out of range. |
| memory space full | No memory space is allocated. |
| NO operand in xxx | Expression xxx is missed. |
| NOT available ASCII char (xx) | Invalid ASCII code. |
| Not found DATA statement | No DATA statement to where RESTORE is executed. |
| Not found THEN in xxx | No THEN statement exists after a IF statement. |
| Only one INPUT file can be opened. | Try to open more than one files at one time in read mode. |
| Only one OUTPUT file can be opened. | Try to open more than one files at one time in write mode. |
| Overflow value | Numeric is out of range to handle. |
| parameter error | Invalid parameter. |
| Program CANNOT be continued. | Try to rerun a terminated program. |
| Program NOT exist | Try to execute without a program. |
| SELECT nesting overflow | Too much nests for a SELECT statement exist. |
| string declaration error | Double quotation marks do not match each other. |
| string length is too long | Too long a character string declaration.<br>(Up to 128 characters) |
| Substring error | Invalid substring specification. |
| Unbalanced BREAK | No BREAKE statement between FOR and NEXT statements. |

(cont'd)

| Error Message | Description |
|---|---|
| Unbalanced FOR variable in NEXT | For statement and NEXT statement do not match correctly each other. |
| Unbalanced line No. | No line specified by LIST statement exists. |
| Unbalanced NEXT statement | No NEXT statement even though FOR statement exists. |
| Unbalanced xxx | Unmatched expression (parentheses, bracket). |
| Unbalanced xxx block | Unmatched xxx block (FOR, IF statements) |
| Undefined LABEL | No label exists. |
| Undefined ON condition | On condition arises without no determination for ON condition. |
| Uninstalled type (xxx) | Invalid variable format. |
| Unknown line No. | No specified line exists. |
| Unmatched DATA's values and READ variable | No DATA statement for READ exists. |
| Unmatched IMAGE-spec in USING | Invalid image use of USING. |
| xxx function error | Error in built-in function. |
| xxx nest overflow | Too much nests exist. |
| xxx (xxx) error | No PURGE command specified file exists. |
| xxx (xxx, xxx) error | No RENAME command specified file exists. |
| xxx: "xxx" file was opened with xxx mode. | File descriptor mode (read/write) is different from specified one. |
| xxx: CANNOT convert into string | Cannot convert into character strings. |
| xxx: invalid first type in xxx | Invalid first type in command is detected. |
| xxx: invalid second type in xxx | Invalid second type in command is detected. |
| xxx: invalid source type in xxx | Source type is invalid to assign into an expression. |

(cont'd)

| Error Message | Description |
|---|---|
| xxx: invalid target type in xxx | Variable type is invalid to assign. |
| xxx: Invalid TARGET operand in XXX | Invalid format is detected in xxx. |
| xxx: Syntax error | Syntax is missed. |
| You cannot use POKE command | Try to execute POKE command. |
| yy is invalid value in xxx | yy in xxx is invalid. |
| yy: Undefined Control Register | The register number in a CONTROL command is missed. |
| yy: UNIT addr error in xxx | GPIB address specification is invalid. |

# ALPHABETICAL INDEX

# Part II

# ate editor

# TABLE OF CONTENTS

# LIST OF ILLUSTRATIONS

# 1 OUTLINE

The ate editor is a full screen editor developed so that BASIC programming can easily be made on the measuring instrument.
Because editing can be made by using the external terminal (VG-920 or equivalent) connected to the RS-232 connector on the panel at the back of the measuring instrument, the operability is greatly improved.

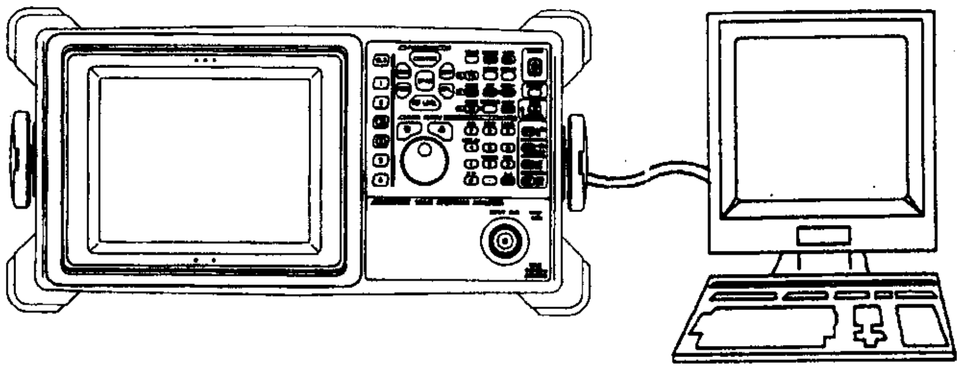


**Figure 1-1 Connection Between U3641/3641N and VG-920**

# 2 FEATURES

The ate editor is a full screen editor having sophisticated editing functions.
The function includes cursor control, text insertion/deletion, copying, moving, replacement, search, window, file, and BASIC program running. In addition to the above, the pop-up and help menus are prepared to smoothly execute these functions.

## 2.1 Keyboard

The basic functions of the ate editor is assigned to the terminal keyboard.
Complicated editing can easily be made through keyboard operations.

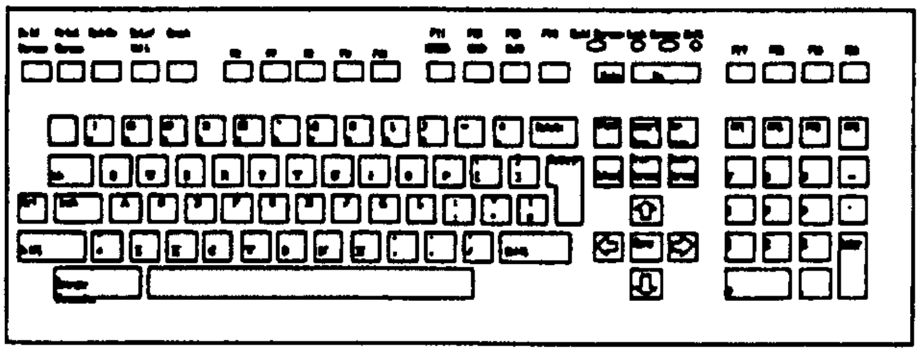


**Figure 2-1 Keyboard of VG-920**

## 2.2 Pop-up Menu

All functions of the ate editor can be executed with the pop-up menu (excluding cursor movement). Select a function to be executed in the pop-up menu with the cursor and press the Return key, then the function is executed.

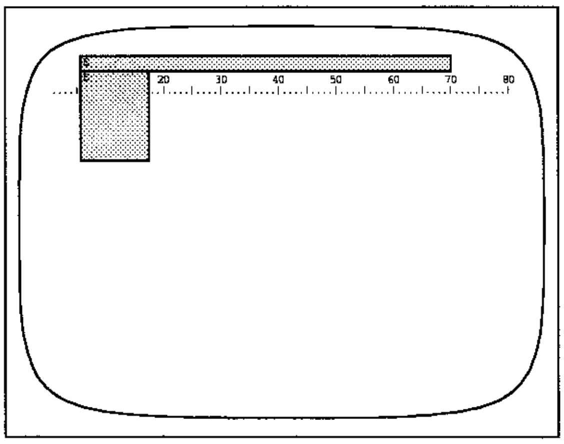


**Figure 2-2 Pop-up Menu**

## 2.3 BASIC Running Condition

A BASIC program edited by the ate editor can directly be run on the editor.
The RUN, CONT, SCRATCH commands of BASIC can be executed with the keyboard or pop-up menu. Commands other than the above ones can be executed in the mini-window of the BASIC mode.

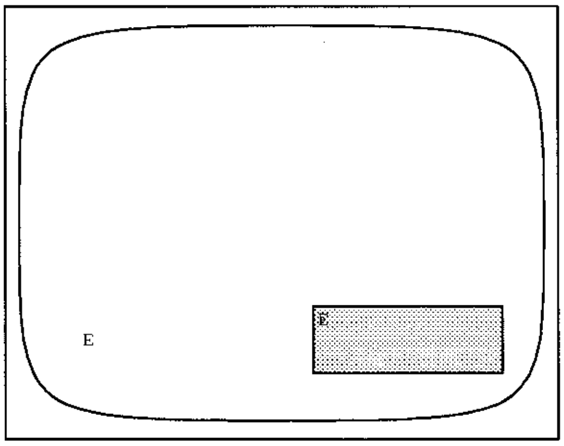
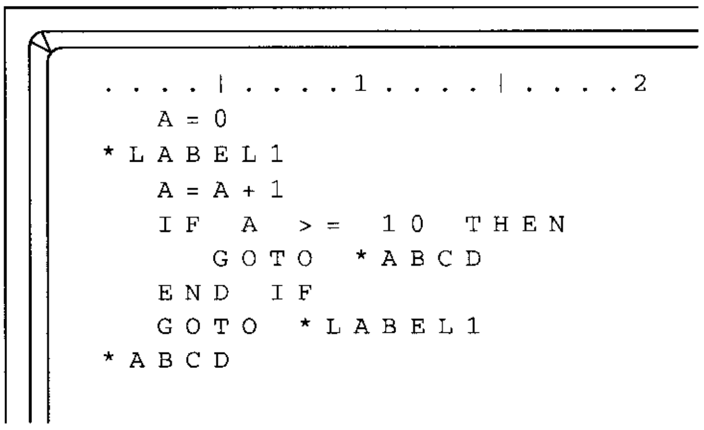


**Figure 2-3 BAISC Mode**

## 2.4 Definition of Label

For the ate editor, line numbers are omitted and labels are defined. By defining labels, understandable programs can be created when using sub-routine call.

***Note:*** ***For the label, be sure to put an asterisk "*" at the top of the character string.***

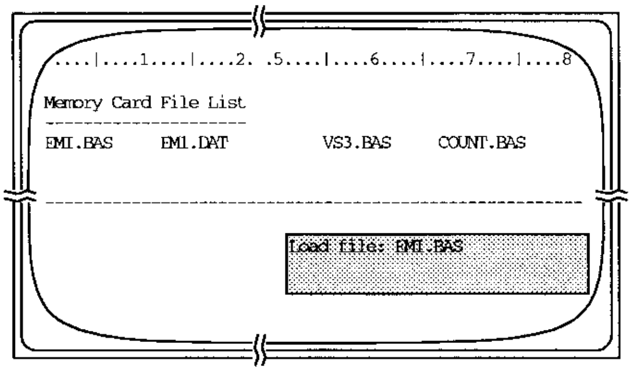


```
....|....1....|....2
  A=0
*LABEL1
  A=A+1
  IF A >= 10 THEN
    GOTO *ABCD
  END IF
  GOTO *LABEL1
*ABCD
```

**Figure 2-4 Definition of Label**

## 2.5 File

It is possible to save the created programs in the memory card of the main unit in files. It is also possible to load a file from the memory card to edit the file.

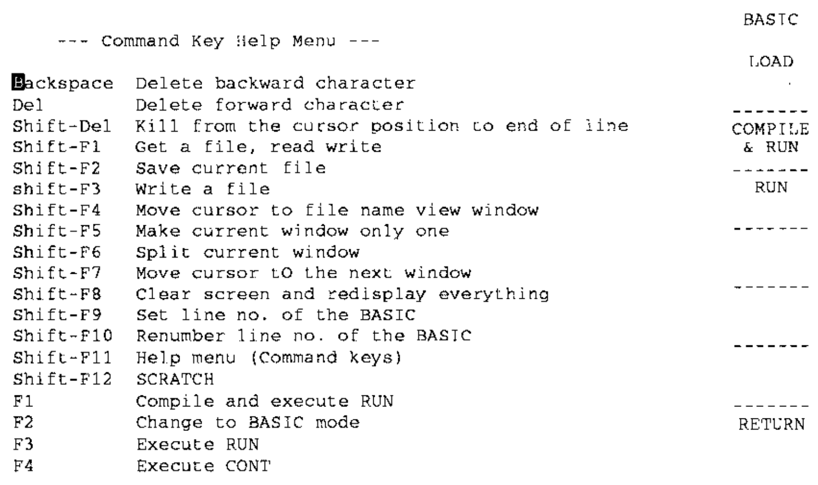


```
....|....1....|....2. .5....|....6....|....7....|....8

Memory Card File List
---------------------
EMI.BAS     EM1.DAT          VS3.BAS     COUNT.BAS

--------------------------------------------------------

                              Load file: EMI.BAS
```

**Figure 2-5 File Loading**

## 2.6 Help Menu

The functions assigned to the external keyboard are listed on the screen.



```
                                                                      BASIC
        --- Command Key Help Menu ---
                                                                      LOAD
Backspace  Delete backward character
Del        Delete forward character                                  -------
Shift-Del  Kill from the cursor position to end of line              COMPILE
Shift-F1   Get a file, read write                                     & RUN
Shift-F2   Save current file                                         -------
shift-F3   Write a file                                                RUN
Shift-F4   Move cursor to file name view window
Shift-F5   Make current window only one                              -------
Shift-F6   Split current window
Shift-F7   Move cursor tO the next window                            -------
Shift-F8   Clear screen and redisplay everything
Shift-F9   Set line no. of the BASIC                                 -------
Shift-F10  Renumber line no. of the BASIC
Shift-F11  Help menu (Command keys)
Shift-F12  SCRATCH                                                   -------
F1         Compile and execute RUN                                   RETURN
F2         Change to BASIC mode
F3         Execute RUN
F4         Execute CONT
```

Figure 2-6 Help Menu

## 2.7 User-Defined Menu (ON KEY x LABEL)

The characters in the menu can be specified by using the ON KEY instruction.

User-Defined Menu is defined on the menu screen corresponding to ON KEY 1 to 6.

In the interruption from the User-Defined Menu, the permission/prohibition can be set using ENABLE INTR/DISABLE INTR instructions.

The User-Defined Menu is sometimes changed when PANEL 0 instruction is executed, however, the User-Defined Menu can be displayed normally by pressing the CONT key.

SAMPLE)

```
    ON KEY 1 LABEL "ABC" GOTO *ABC
    ON KEY 3 LABEL "DEF" GOTO *DEF
    ON KEY 6 LABEL "XYZ" GOTO *XYZ
    KEY ON
```

The user-Defined Menu is displayed as follows:

```
       ABC

    ----------



    ----------
       DEF

    ----------



    ----------



    ----------
       XYZ
```

# 3 FUNCTION LIST

Each function of the ate editor is assigned to the keyboard as the main key pad, editing key pad, auxiliary key pad, and function key pad excluding some functions.

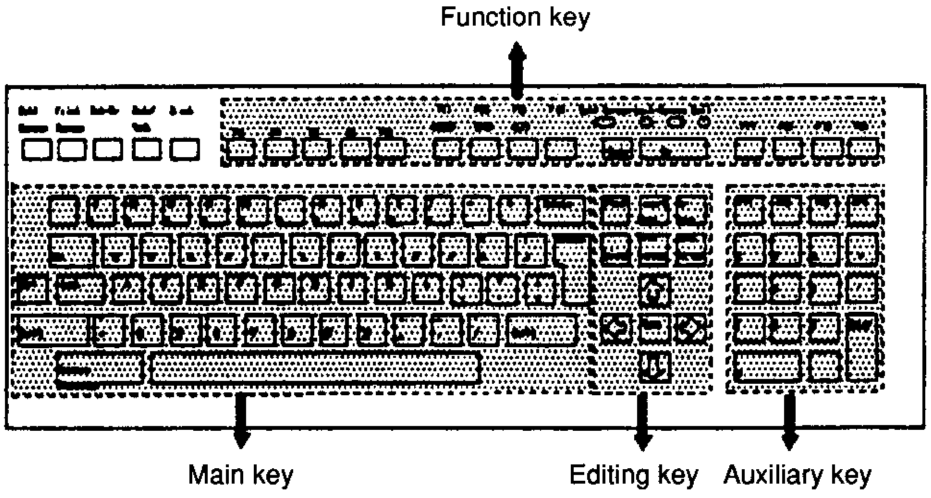


**Figure 3-1 Key Pad Layout**

## 3.1 Cursor Control

| | | |
|---|---|---|
| → | : Moves the cursor to the next character. | (Editing key pad) |
| ← | : Moves the cursor to the preceding character. | (Editing key pad) |
| ↑ | : Moves the cursor to the preceding line. | (Editing key pad) |
| ↓ | : Moves the cursor to the next line. | (Editing key pad) |
| Find | : Moves the cursor to the top of the file. | (Editing key pad) |
| Insert-Here | : Moves the cursor to the end of the file. | (Editing key pad) |
| Remove | : Moves the cursor to the top of the line. | (Editing key pad) |
| Select | : Moves the cursor to the end of the line. | (Editing key pad) |
| Prev-screen | : Moves the cursor to the preceding screen. | (Editing key pad) |
| Next-screen | : Moves the cursor to the next screen. | (Editing key pad) |

## 3.2 Insertion

| | | |
|---|---|---|
| Return | : Inserts line-feed characters. | (Main key pad) |
| Tab | : Inserts tab instruction. | (Main key pad) |

Main key pad other than above
: Inserts general characters.

## 3.3 Deletion

| | |
|---|---|
| Delete | : Deletes the character before the cursor. |
| 0 | : Deletes the character at the cursor. (Auxiliary key pad) |
| . | : Deletes characters from the cursor position through the line end. (Auxiliary key pad) The deleted characters using the "." can be returned to the original state by pressing the PF4 key. (Auxiliary Key Pad) |

## 3.4 Copying, Moving, and Deleting (Region designation)

| | |
|---|---|
| PF1 | : Sets a mark necessary for region processing to the cursor position. (Auxiliary key pad) |
| PF2 | : Deletes characters from the mark through the cursor position to save them in the buffer. (Auxiliary key pad) |
| PF3 | : Stores characters from the mark through the cursor position. (Auxiliary key pad) |
| PF4 | : Copies the characters saved in the buffer with the PF2 or PF3 key to the cursor position. (Auxiliary key pad) |

## 3.5 Window

| | |
|---|---|
| 4 | : Merges windows into one. (Auxiliary key pad) |
| 5 | : Splits a window into two. (Auxiliary key pad) |
| 6 | : Moves the cursor to the next window. (Auxiliary key pad) |
| , | : Clears the screen and displays data again. (Auxiliary key pad) |

## 3.6 File

| | |
|---|---|
| 7 | : Loads the designated file from the memory card. (Auxiliary key pad) |
| 8 | : Saves files in the memory card. (Auxiliary key pad) |
| 9 | : Saves a file with the designated file name in the memory card. (Auxiliary key pad) |
| - | : Moves the cursor to the file name list window. (Auxiliary key pad) |

## 3.7 Search

| | |
|---|---|
| 2 | : Searches character strings forward. (Auxiliary key pad) |
| 3 | : Searches character strings backward. (Auxiliary key pad) |

## 3.8 Replacement

| | |
|---|---|
| 1 | : Replaces character strings. (Auxiliary key pad) |

## 3.9 BASIC Mode

| | |
|---|---|
| F17 | : Moves a program to the BASIC buffer before running the program. (Function key pad) |
| F18 | : Changes the mode to the BASIC mode. (Function key pad) |
| F19 | : Runs the transferred program. (Function key pad) |
| F20 | : Continues the program interrupted by Ctrl-C. (Function key pad) |

## 3.10 Help

| | |
|---|---|
| Help | : Displays the editor functions and key assignment list. (Function key pad) |

## 3.11 Pop-up Menu

| | |
|---|---|
| Do | : Displays the pop-up menu. (Function key pad) |

## 3.12 Cancel

| | |
|---|---|
| F11 | : Cancels the editor functions. (Function key pad) |

## 3.13 Editor Restarting

| | |
|---|---|
| F12 | : Ends the current editor and initializes the buffer for restarting. (Function key pad) |

# 4 DESCRIPTION OF FUNCTIONS

## 4.1 Cursor Control

The cursor indicates the editing position and editing is made according to the cursor.
The cursor control is defined as movement of the cursor to a certain position on the screen.
There are the functions to move the cursor upward, downward, rightward, or leftward by one character. These functions are assigned to the cursor keys respectively to move the cursor from the current position to a destination.

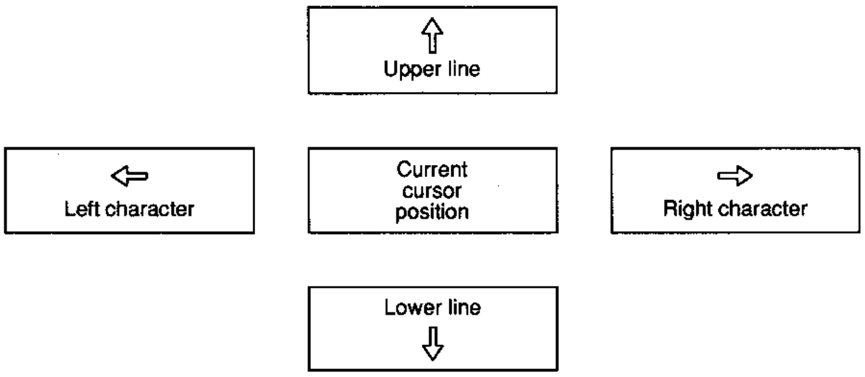


**Figure 4-1 Cursor Key**

These are the basic cursor moving function frequently used. When it is attempted to move the cursor exceeding the upper or lower limit of the screen, the text in the direction is scrolled and the cursor always remains in the screen.

In addition to the above, there are the functions assigned to the editing key pad.

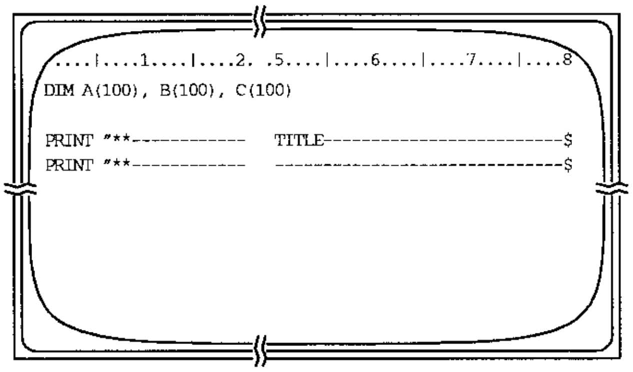
| | |
|---|---|
| `Find` | Moves the cursor to the top of the file. |
| `Insert-Here` | Moves the cursor to the end of the file. |
| `Remove` | Moves the cursor to the top of the line. |
| `Select` | Moves the cursor to the end of the line. |
| `Next-Screen` | Advances the cursor to the next page. |
| `Prev-Screen` | Returns the cursor to the previous page. |

**Figure 4-2 Auxiliary Key Pads to Move Cursor**

## 4.2 Insertion

Because the ate editor is always ready for inserting texts, all characters other than control ones can be inserted with the keyboard. The character includes the general character and the control character.
The general character includes visible characters such as "A" and "1" while the control character includes invisible characters such as "ESC" and "CTRL".

When the length of a line exceeds the screen width, the sign "$" is displayed at the right end of the screen to show that the line is continued.



```
....|....1....|....2. .5....|....6....|....7....|....8
DIM A(100), B(100), C(100)

PRINT "**-------------   TITLE-------------------------$
PRINT "**-------------   ------------------------------$
```

**Figure 4-3 Sign Showing Continuation of Characters**

Press the Return key at the end of a line to input the line feed character.
Press Tab key for the tab function to adjust character position. The tab size is provided with the length of eight characters.

| | |
|---|---|
| `Return` | Inserts line feed characters. |
| `Tab` | Inserts tab. |
| `Main key pad other than above` | Inserts general characters. |

**Figure 4-4 Main Key Pad for Insertion**

## 4.3 Deletion

The deleting function includes one-character deletion, one-line deletion, and designated regin deletion.

### 4.3.1 One-character Deletion

The one-character deleting function includes two methods; one to delete the character immediately before the cursor and the other to delete the character at the cursor position.

(1) Deletion of the character immediately before the cursor

To delete the character immediately before the cursor, press Delete key (Main Key Pad).

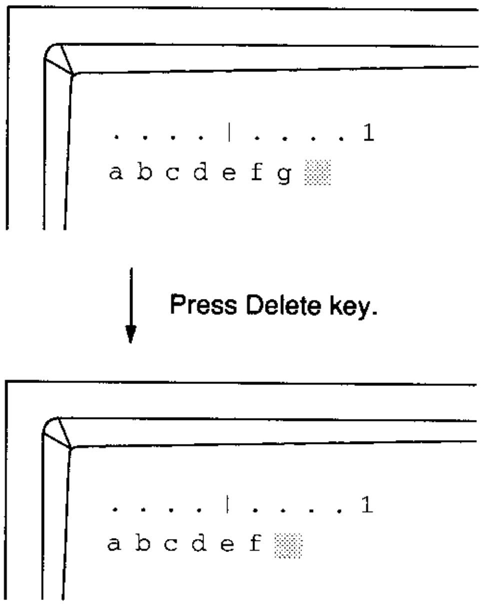


**Figure 4-5 Deletion of the Character Immediately Before the Cursor**

(2) Deletion of the character at the cursor position

To delete the character at the cursor position, press 0 key (Auxiliary Key Pad).

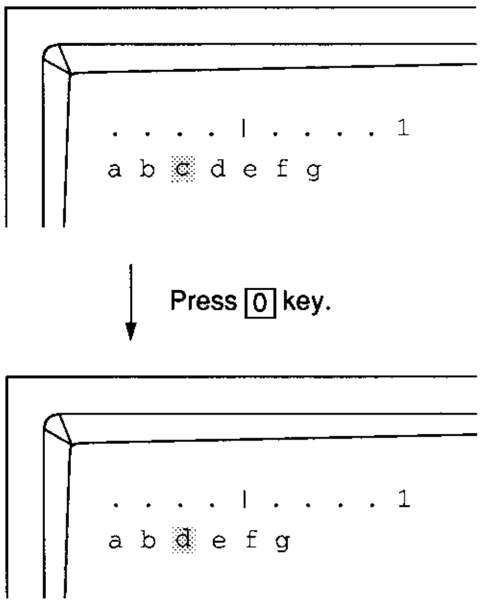
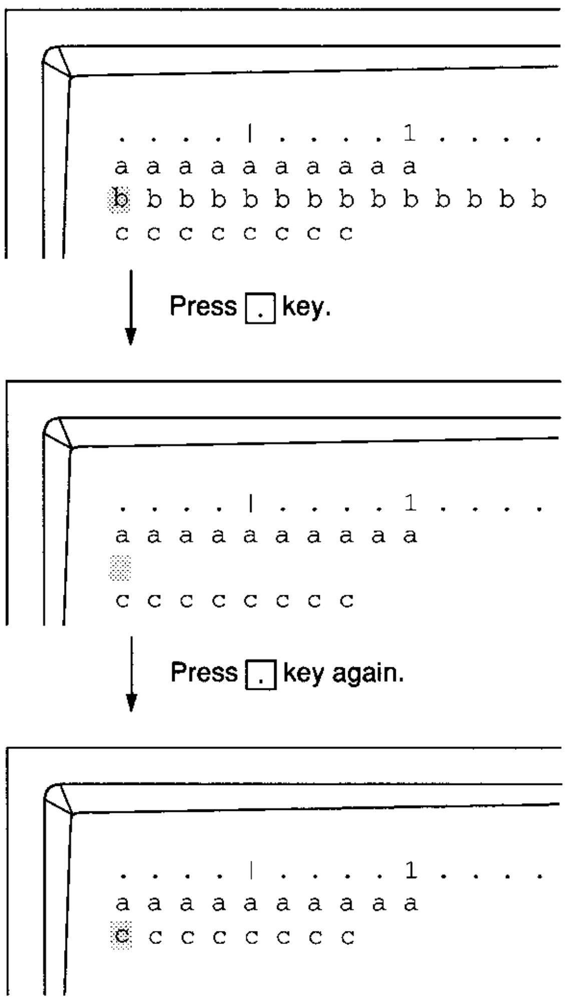


**Figure 4-6 Deletion of the Character at the Cursor Position**

### 4.3.2 One-line Deletion

To delete characters from the cursor position through the line end, press . key (Auxiliary Key Pad). The contents of the line are deleted by pressing . key first time and the line is deleted by pressing . key again.



```
. . . . | . . . . 1 . . . .
a a a a a a a a a a
b b b b b b b b b b b b b
c c c c c c c c
```

```
. . . . | . . . . 1 . . . .
a a a a a a a a a a

c c c c c c c c
```

```
. . . . | . . . . 1 . . . .
a a a a a a a a a a
c c c c c c c c
```

**Figure 4-7 Deletion of the Characters from Cursor Position Through Line End**

### 4.3.3 Designated Region Deletion

The method to delete characters by designating a region is described in Section 4.4 because the method is related to storage of text.

| | |
|---|---|
| `Delete` | Deletes the character immediately before the cursor. |
| `0` | Deletes the character at the cursor position. (Auxiliary Key Pad) |
| `.` | Deletes the characters from the cursor through the line end. (Auxiliary Key Pad) |

**Figure 4-8 Key Pad for Deletion**

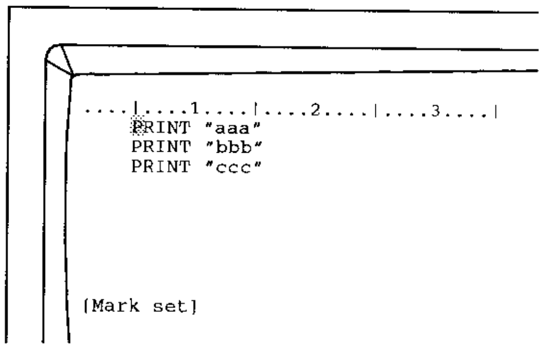
## 4.4 Copying, Moving, and Deleting (Region designation)

Store a text by designating a region for copying, moving, and deleting.

### 4.4.1 Deletion of Text

(1) Mark Setting

For deleting and moving, move the cursor to the top of the region to be deleted and set a mark with PF1 key (Auxiliary Key Pad) or "Set Mark" in the pop-up menu.

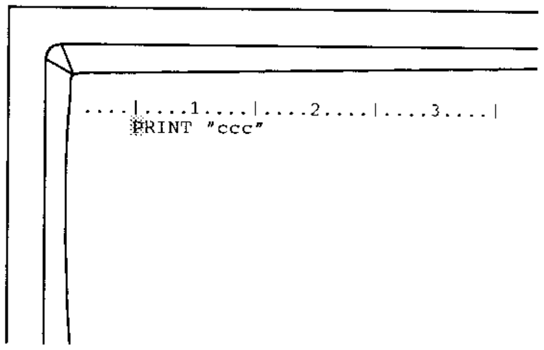


```
....|....1....|....2....|....3....|
    PRINT "aaa"
    PRINT "bbb"
    PRINT "ccc"



[Mark set]
```

**Figure 4-9 Mark Setting**

(2) Deletion of Text

Move the cursor to the position one character ahead of the final position of the region to be deleted and press PF2 key (Auxiliary Key Pad) or input Kill region of the pop-up menu. Then the text from the position where the mark is set in Item (1) through the position before the current cursor position is deleted. Because the deleted text is saved in the internal buffer, perform the operation in Items (1) and (2) before moving, recovering, or copying the text to be mentioned later.

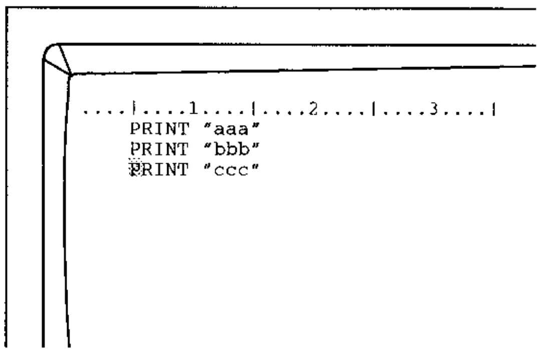


```
....|....1....|....2....|....3....|
    PRINT "ccc"
```

**Figure 4-10 Storage of Text (Deletion)**

## 4.4.2 Recovering and Moving of Text

To recover the deleted text, press PF4 key (Auxiliary Key Pad) at the current cursor position or input Yank of the pop-up menu. To move the text, move the cursor to the destination and press PF4 key (Auxiliary Key Pad) or input Yank of the pop-up menu. As mentioned above, PF4 key (Auxiliary Key Pad) and Yank in the pop-up menu are the function to insert the saved text into the current cursor position.

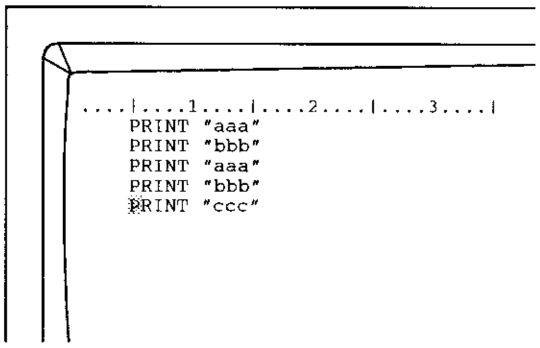


```
....|....1....|....2....|....3....|
    PRINT "aaa"
    PRINT "bbb"
    PRINT "ccc"
```

**Figure 4-11 Recovery of Text**

### 4.4.3 Copying of Text

To copy a text, set the top of the region similarly to the case of deletion, move the cursor to the final position, and press PF3 key (Auxiliary Key Pad) or input Copy region of the pop-up menu. In this case, becuase the designated text is saved in the internal buffer though it seems that no data is changed in the region, move the cursor to the copying positoin and press PF4 key (Auxiliary Key Pad) or input Yank of the pop-up menu.



```
....|....1....|....2....|....3....|
    PRINT "aaa"
    PRINT "bbb"
    PRINT "aaa"
    PRINT "bbb"
    PRINT "ccc"
```

**Figure 4-12 Copying of Text**

The text recovered by PF4 key (Auxiliary Key Pad) or Yank in the pop-up menu is used as the latest textin the internal buffer.

| | |
|---|---|
| PF1 | Sets the mark necessary for designating a region to the cursor position. |
| PF2 | Stores the text deleted from the mark through the cursor position. |
| PF3 | Stores the text from the mark through the cursor position. |
| PF4 | Inserts the stored text into the cursor position. |

**Figure 4-13 Auxiliary Key Pad for Region Processing**

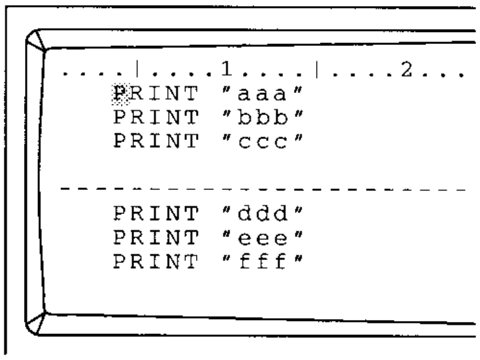
## 4.5 Window

The normal window includes the overlapping type and the tile type. The ate editor window uses the tile type which splits the screen into the top and bottom ones.

Generally, the number of windows is one, the ate editor, however, can have two windows in each of which the text is displayed.
For example, it is possible to simultaneously reference two areas each of which is larger than the screen.
However, it is impossible to open different files in two windows respectively because only one file can be opened.

### 4.5.1 Splitting of Window

To split a window into two, press 5 key (Auxiliary Key Pad) or input Split of the pop-up menu.

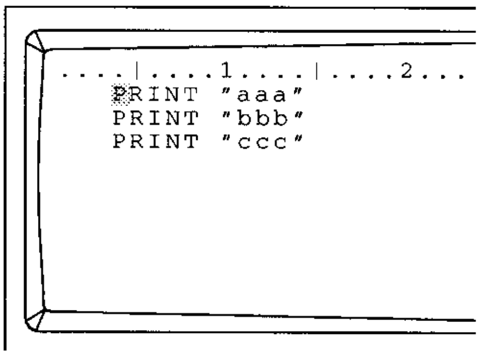


```
....|....1....|....2...
    PRINT "aaa"
    PRINT "bbb"
    PRINT "ccc"

-----------------------
    PRINT "ddd"
    PRINT "eee"
    PRINT "fff"
```

**Figure 4-14 Window Split into Top and Bottom Ones**

### 4.5.2 Initialization of Window

To merge the top and bottom window into one, press 6 key (Auxiliary Key Pad) or input Next of the pop-up menu to move the cursor to the window to be left. Then press 4 key (Auxiliary Key Pad) or input Only of the pop-up menu and the two windows are merged into one.



```
....|....1....|....2...
    PRINT "aaa"
    PRINT "bbb"
    PRINT "ccc"
```

**Figure 4-15 Merged Window**

### 4.5.3 Re-displaying of Screen

To display the text being edited again, press , (comma) key (Auxiliary Key Pad) or input Redisplay of the pop-up menu to display the test again.

| | |
|---|---|
| 4 | Deletes other windows to display only the window in which the cursor is present. |
| 5 | Splits a window into the top and bottom ones and brings the cursor to the top one. |
| 6 | Moves the cursor to the next window. |
| , | Clears the screen to display the text again. |

**Figure 4-16 Auxiliary Key Pad for Window**

## 4.6 File

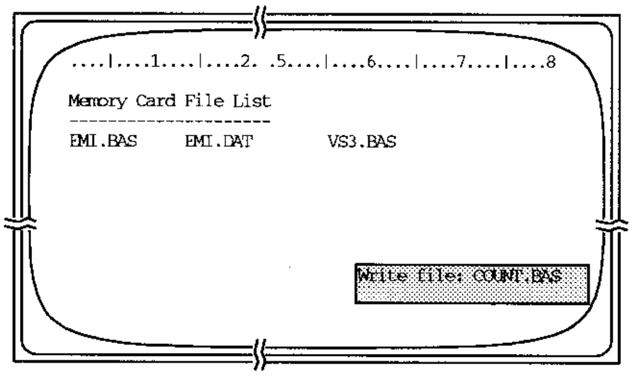
The ate editor saves texts in the memory card of the main unit and loads them from the memory card in files.
Therefore, be sure to save edited texts in the memory card.

***Note: If the editor is turned off before saving the edited texts, the texts are lost.***

The following describes how to save/load file.

### 4.6.1 Saving of Files

To save an edited text in the memory card, press 9 key (Auxiliary Key Pad) on the auxiliary key pad or input Write of the pop-up menu. In this case, give a name to the file to be saved. The file name is allowed up to 10 characters. After saving the text, the number of written lines is displayed in the message line.

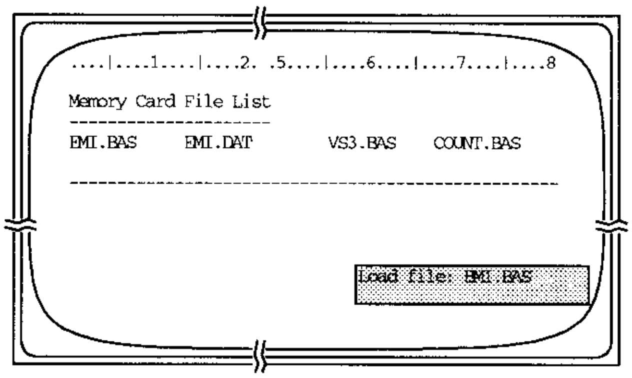


```
....|....1....|....2. .5....|....6....|....7....|....8

Memory Card File List
----------------------
EMI.BAS     EMI.DAT       VS3.BAS



                                      Write file: COUNT.BAS
```

**Figure 4-17 Saving of Text (Input a file name)**

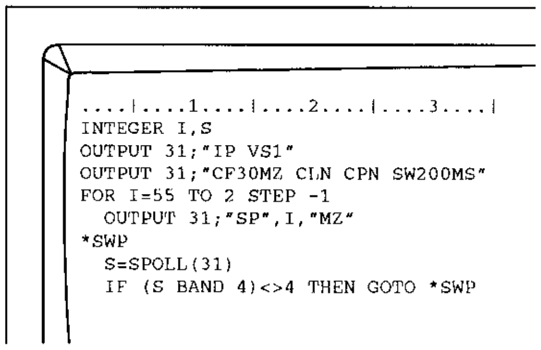
### 4.6.2 Loading of Files

To load the saved files, press 7 key (Auxiliary Key Pad) or input Load of the pop-up menu. Then the files currently saved in the memory card are listed on the screen and the file to be loaded is queried.
In this case, input the file name or move the cursor to the window showing the list by pressing - key (Auxiliary Key Pad) before selecting the file name with the cursor and finally pressing Return key.



```
....|....1....|....2. .5....|....6....|....7....|....8
Memory Card File List
---------------------
EMI.BAS     EMI.DAT       VS3.BAS    COUNT.BAS
------------------------------------------------------
                                   Load file: EMI.BAS
```

**Figure 4-18 Inputting of File Name**



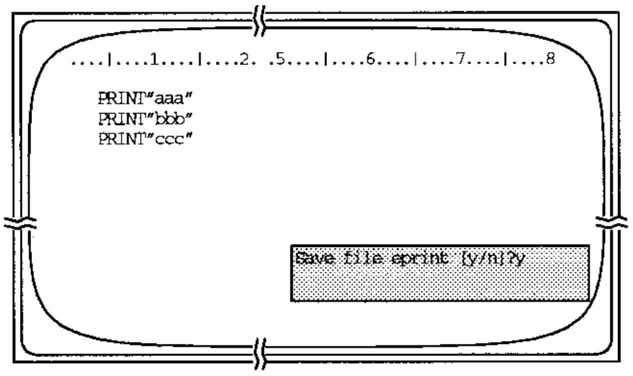
```
....|....1....|....2....|....3....|
INTEGER I,S
OUTPUT 31;"IP VS1"
OUTPUT 31;"CF30MZ CLN CPN SW200MS"
FOR I=55 TO 2 STEP -1
  OUTPUT 31;"SP",I,"MZ"
*SWP
  S=SPOLL(31)
  IF (S BAND 4)<>4 THEN GOTO *SWP
```

**Figure 4-19 Loading of File**

While a file is loaded from the memory card, the lamp on the left side of the memory card drive stays lit up in red color. When loading is completed, the text is displayed on the screen and the number of read lines is displayed in the message line.
To create a new file, input the file name.
Unless the memory card is set, the buzzer sounds or an error message is displayed.

## 4.6.3 Updating of Files

To save the same file name, press 8 key (Auxiliary Key Pad) or input Save of the pop-up menu. Then whether or not to save it is queried. If no data in the text is changed, the message "No changes" is displayed.



```
....|....1....|....2. .5....|....6....|....7....|....8

    PRINT"aaa"
    PRINT"bbb"
    PRINT"ccc"

                         Save file eprint {y/n}?y
```

Figure 4-20 Updating of Text (Same File Name)

When you press y and Return keys, the text is written in the file. When saving is completed, the number of lines of the file is displayed in the message line.
When you press n and Return keys, text saving is not executed.

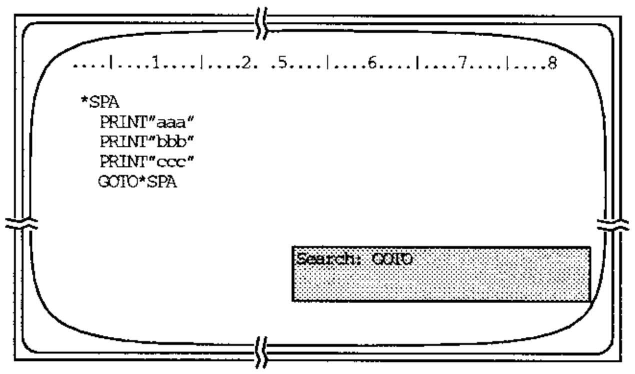
| | |
|---|---|
| 7 | Loads the designated file from the memory card. |
| 8 | Saves a file with the same name in the memory card. |
| 9 | Saves a file with the designated file name in the memory card. |
| - | Moves cursor to the file list window. |

Figure 4-21 Auxiliary Key Pad for File

## 4.7 Search

This is the function to search a character string forward or backward in a text.
To search the character string after the cursor position, press 2 key (Auxiliary Key Pad) or input Forward search of the pop-up menu. To search it before the cursor position, press 3 key (Auxiliary Key Pad) or input Backward search of the pop-up menu.
The both types of search execute the same function except searching direction.

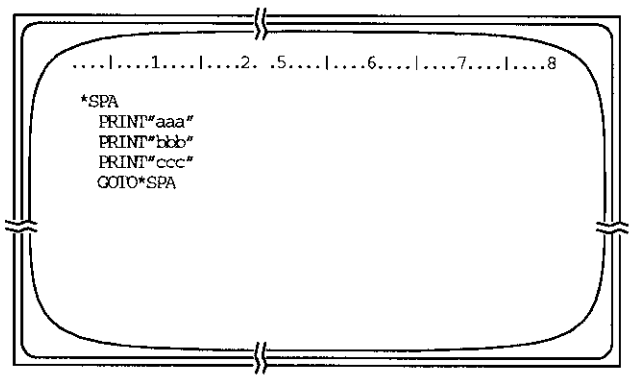
The following is the example to search a character string after the cursor position. Press 2 key (Auxiliary Key Pad) or input Forward search of the pop-up menu.
A mini-window is displayed on the screen and the character string to be searched is queried. Therefore, input the character string to be searched.



```
....|....1....|....2. .5....|....6....|....7....|....8
*SPA
  PRINT"aaa"
  PRINT"bbb"
  PRINT"ccc"
  GOTO*SPA

                              Search: GOTO
```

Figure 4-22 Search of Character String After Cursor Position

The cursor moves to the top of the character string to be searched to end search. To search the same character string consecutively, perform the same operation. However, character string input can be omitted.



```
....|....1....|....2. .5....|....6....|....7....|....8
*SPA
  PRINT"aaa"
  PRINT"bbb"
  PRINT"ccc"
  GOTO*SPA
```

Figure 4-23 Execution Result

If the character string cannot be searched, the message "Not found" is displayed.

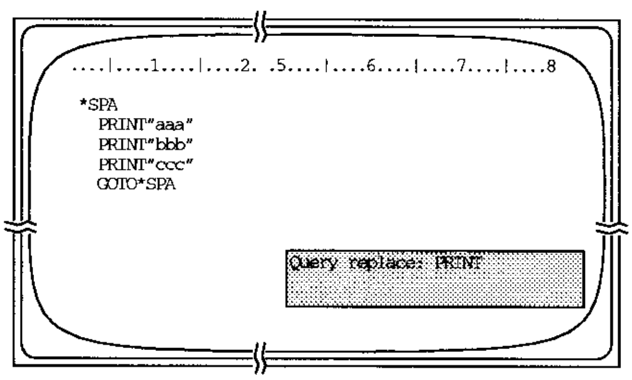
| | |
|---|---|
| 2 | Searches a character string backward from the cursor position. |
| 3 | Searches a character string forward from the cursor position. |

Figure 4-24 Auxiliary Key Pad for Search

## 4.8 Replacement

This is the function to replace any character string after the cursor position in a text.

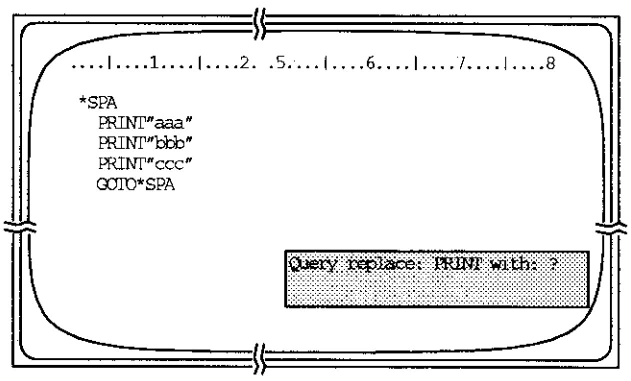
The following is the example to replace a character string. Press 1 key (Auxiliary Key Pad) or input Query replace of the pop-up menu, then a mini-window is displayed on the screen and the character string to be replaced is queried. Therefore, input the character string to be replaced.



```
....|....1....|....2. .5....|....6....|....7....|....8
*SPA
  PRINT"aaa"
  PRINT"bbb"
  PRINT"ccc"
  GOTO*SPA

                      Query replace: PRINT
```

**Figure 4-25 Inputting of Character String to be Replaced**

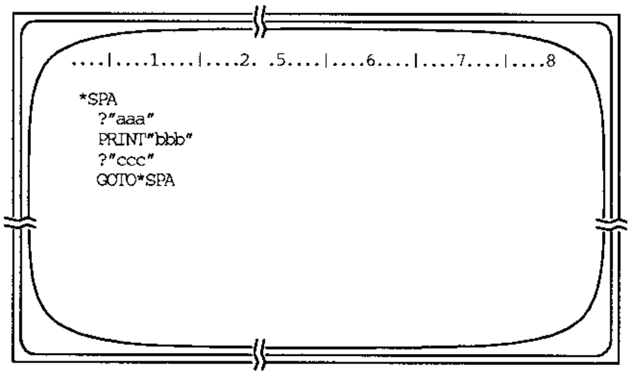
When the character string to be newly changed is queried, input an optional character string.



```
....|....1....|....2. .5....|....6....|....7....|....8
*SPA
  PRINT"aaa"
  PRINT"bbb"
  PRINT"ccc"
  GOTO*SPA

                      Query replace: PRINT with: ?
```

**Figure 4-26 Inputting of New Character String**

When the character string concerned is searched, the cursor moves to the top of the character string and the character string is ready for replacement.
In this case, press space key (Main Key Pad) to replace the character string or delete key (Main Key Pad) unless replacing it.
To end the operation after replacement, press . key (Auxiliary Key Pad). To end the operation without replacement, press F11 (Function Key Pad).



```
....|....1....|....2. .5....|....6....|....7....|....8

*SPA
  ?"aaa"
  PRINT"bbb"
  ?"ccc"
  GOTO*SPA
```

**Figure 4-27 Execution Result**

| | |
|---|---|
| `1` | Replaces a character string. |
| `space` | Executes replacement. |
| `delete` | Executes no replacement. |
| `.` | Ends operation after replacement. |
| `F11` | Ends operation. |

**Figure 4-28 Auxiliary and Function Key Pad for Replacement**

## 4.9 BASIC

The ate editor is a portable editor developed to make BASIC programming on the measuring instrument. Therefore, it allows the operator to easily run BASIC and debugging while editing. In addition, it is possible to execute various BASIC commands in the BASIC mode. The following describes the BASIC commands which can be executed on the editor.

### 4.9.1 Running of BASIC

Before the program is run, the relationship between the ate editor and BASIC interpreter is roughly described below.
The program has been edited by the ate editor.
Therefore, completed programs are saved in the internal buffer. Meanwhile, because the BASIC interpreter also has an internal buffer, the BASIC program must exist in the buffer to run the program.

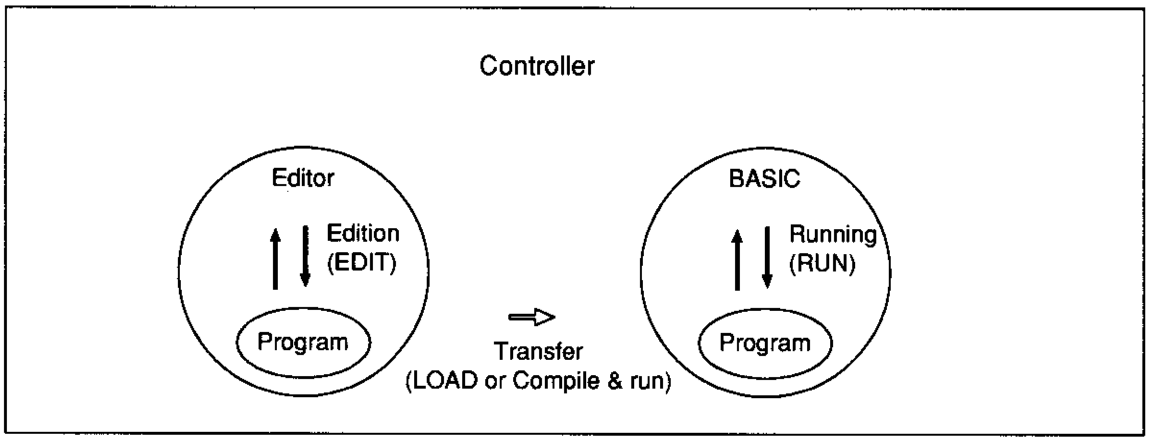


**Figure 4-29 Relationship Between Editor and Interpreter**

Therefore, move the program from the editor to the interpreter before running the program.
When you press F17 key (Function Key Pad) or input Compile & run of the pop-up menu, the program is moved from the editor and run. In this case, the program last run is deleted. Displays the program statement where the error arises and the error message.

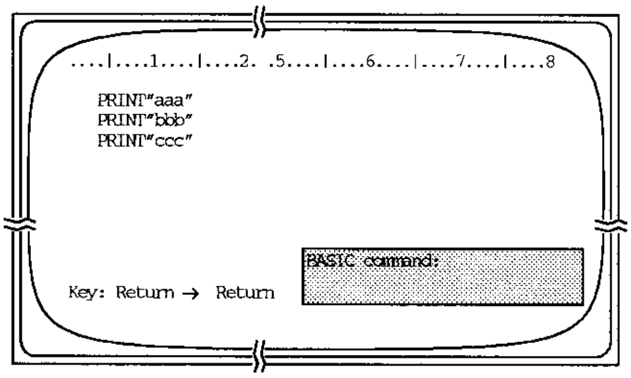
To run a moved program, press F19 key (Function Key Pad) or input Run of the pop-up menu. If the program is executed with F19 key (Function Key Pad) or Run of the pop-up menu before it is moved, the error message "Program not exist" is displayed.
When program running ends, the prompt "BASIC command:" is displayed in the mini-window and the debugging environment is ready. To return control to the editor, press Return key.

### 4.9.2 Stopping of BASIC

To stop running BASIC, press Ctrl-C key (Main Key Pad).

### 4.9.3 BASIC Mode

The BASIC mode is the environment capable of executing commands for BASIC, which includes the debugging environment previously mentioned.
Press F18 key (Function Key Pad) or input Basic mode of the pop-up menu, and the mini-window appears.



```
....|....1....|....2. .5....|....6....|....7....|....8

PRINT"aaa"
PRINT"bbb"
PRINT"ccc"

                                BASIC command:
Key: Return → Return
```

**Figure 4-30 BASIC Mode**

The commands RUN, LIST, and CONT can be executed.

***Note:*** ***To change the program contents, be sure to return the control to the editor.***

### 4.9.4 Continuation of BASIC

To continuously run the program which is interruped by Ctrl-C key (Main Key Pad) or PAUSE command, press F20 key (Function Key Pad) or input Cont of the pop-up menu. This is the same with the BASIC command CONT. The program is run from the line next to the interrupted line. If there is no program to be run, the message "Program cannot be continued" is displayed.

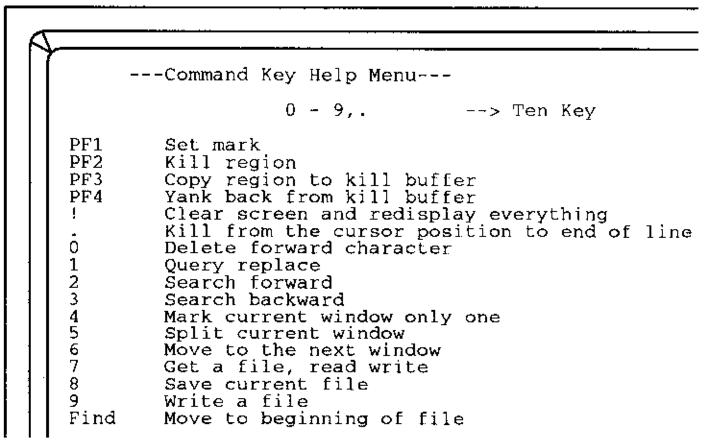
| | |
|---|---|
| F17 | Moves the program before running it. |
| F18 | Changes the mode to the BASIC mode. |
| F19 | Runs the moved program. |
| F20 | Continues the program interrupted by Ctrl-C key or the PUASE command. |

**Figure 4-31 Function Key Pad for BASIC**

## 4.10 Help

This allows the operator to check the relationship between the editor functions and keyboard (VG-920) on the screen.

When you press HELP key (Function Key Pad) or input Help of the pop-up menu, brief description of each key function is displayed.
The function to move the cursor including scrolling of menu screen is the same as the editing function. However, there is not the function to move the cursor right and left.
Press F11 (Function Key Pad) or Ctrl-G key (Main Key Pad), and the HELP function stops and the original editing screen appears.



```
      ---Command Key Help Menu---
                  0 - 9,.       --> Ten Key
PF1     Set mark
PF2     Kill region
PF3     Copy region to kill buffer
PF4     Yank back from kill buffer
!       Clear screen and redisplay everything
.       Kill from the cursor position to end of line
0       Delete forward character
1       Query replace
2       Search forward
3       Search backward
4       Mark current window only one
5       Split current window
6       Move to the next window
7       Get a file, read write
8       Save current file
9       Write a file
Find    Move to beginning of file
```

**Figure 4-32 Help Menu**

| | |
|---|---|
| HELP | Displays the help menu. |
| F11 (CTRL-G) | Cancels the ehip menu. |

**Figure 4-33 Function and Main Key Pad for Help**

## 4.11 Pop-up Menu

Though every function of the ate editor previously mentioned is executed by operating the keyboard, the same operation is realized with the pop-up menu to be described below.
All editor functions except the cursor key function are displayed in the pop-up menu. Therefore, select any function among them for execution.
This menu is very effective when you cannot understand key assignment or you operate this equipment independently.

When you press Do (Function Key Pad), the pop-up menu is displayed at the top of the screen.

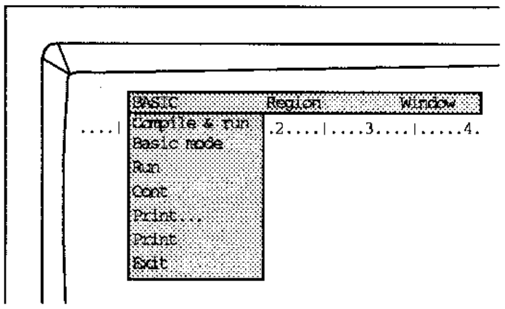


**Figure 4-34 Pop-up Menu**

| BASIC | Region | Window | File | Search/Replace |
|---|---|---|---|---|
| Compile & run | Set mark | Only | Load | Forward search |
| Basic mode | Kill region | Split | Save | Backward search |
| Run | Copy region | Next | Write | Query replace |
| Cont | Yank | Redisplay | | Go to line |
| Print ... | | Help | | Show line No. |
| Print | | | | |
| Exit | | | | |

**Figure 4-35 Pop-up Menu List**

To move the cursor between items, use ←, → (Editing Key Pad), or F6 through F10 keys (Function Key Pad).
The ← and → keys (Editing Key Pad) move the cursor right and left and F6 through F10 keys (Function Key Pad) are assigned to each item.

| | |
|---|---|
| F6 | BASIC menu |
| F7 | Region menu |
| F8 | Window/Other menu |
| F9 | File menu |
| F10 | Search/Replace menu |

**Figure 4-36 Function Key Pad for Pop-up Menu**

When the menu of the item concerned is displayed, select the function to be executed with the ↑ and ↓ keys (Editing Key Pad).

| | |
|---|---|
| Do | Displays the pop-up menu. |
| F11 | Cancels the pop-up menu. |

**Figure 4-37 Function and Main Key Pad for Pop-up Menu**

| Menu | Description | Reference |
|---|---|---|
| Compile & run | Moves and runs BASIC programs. | 4.9.1 |
| Basic mode | Executes BASIC commands. | 4.9.3 |
| Run | Runs the moved programs. | 4.9.1 |
| Cont | Continues the program interrupted by Ctrl-C key. | 4.9.4 |
| Print ... | Set GPIB printer's address to output program list. | |
| Print | Output program list to GPIB printer. | |
| Exit | Leaves from editor function. | 4.13 |
| Set mark | Deletes texts and sets the top position for copying. | 4.4.1 |
| Kill retion | Deletes a text from the text mark through the cursor position. | 4.4.1 |
| Copy region | Stores a text from the text mark through the cursor position. | 4.4.3 |
| Yank | Copies (Recovers) the stored text. | 4.4.2 |
| Only | Merges windows into one. | 4.5.2 |
| split | Splits a window into top and bottom ones. | 4.5.1 |
| Next | Moves the cursor to other window. | 4.5.2 |
| Redisplay | Re-displays a text. | 4.5.3 |
| Help | Displays the editor functions assigned to VG-920. | 4.10 |
| Load | Loads files from the memory card. | 4.6.2 |
| Save | Updates files. | 4.6.3 |
| Write | Saves files in the memory card. | 4.6.1 |
| Forward search | Searches a character string backward from the cursor position. | 4.7 |
| Backward search | Searches a character string forward from the cursor position. | 4.7 |
| Query replace | Replaces character strings. | 4.8 |
| Go to line | Moves cursor to the input line. | |
| Show line No. | Displays the number of lines from the first line to the putting line on the cursor and the number of all line of the program. | |

**Figure 4-38 Description of Pop-up Menu**

## 4.12 Cancel of Pop-up Menu

To cancel the pop-up menu, press F11 (Function Key Pad) or Ctrl-G key (Main Key Pad).

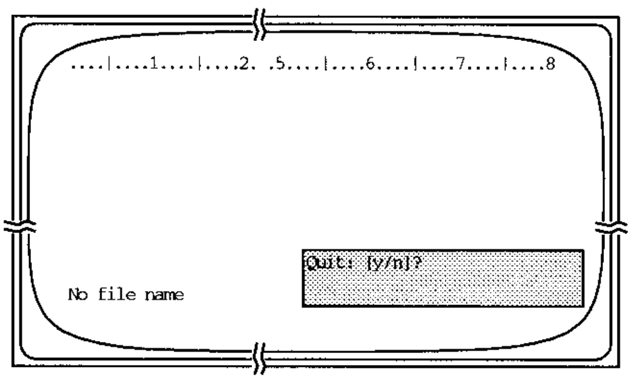
***Note:*** ***The function cannot be canceled while a text is moved to the BASIC buffer or a file is saved or loaded.***

| | |
|---|---|
| `F11 (Ctrl-G)` | Cancels pop-up menu. |

**Figure 4-39 Function (Main) Key Pad for Cancel**

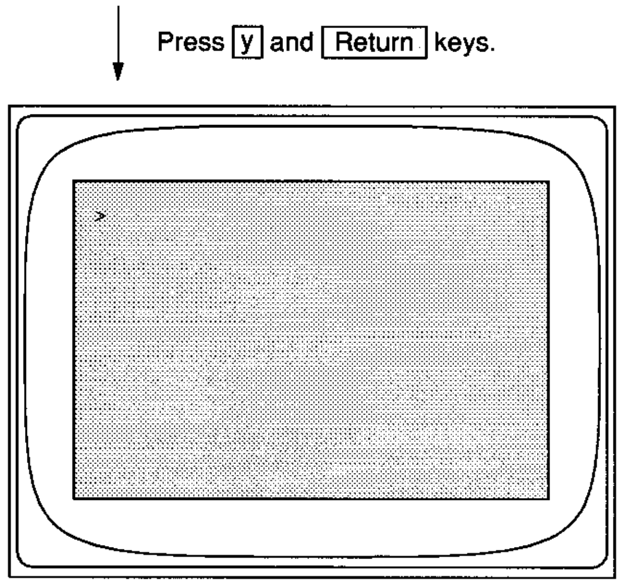
## 4.13 Cancel of Editor Function

To cancel the editor function being operated, press F12 (Function Key Pad) or input Exit of the pop-up menu, the mini-window is displayed and it is queried whether or not to save the text being edited. (Only when changing the text)
After you respond the query, it is queried whether or not to cancel the editor function. When you press y and Return keys, the editor function is canceled. When you press n and Return keys, the original editor screen appears.



```
....|....1....|....2. .5....|....6....|....7....|....8

                                    Quit: [y/n]?
No file name
```

Figure 4-40 Mini-window to End Editor Operation



Press y and Return keys.

```
>
```

Figure 4-41 Initialization of Editor

| | |
|---|---|
| F12 | Cancels the editor function. |

Figure 4-42 Function Key Pad for Initialization of Editor

# 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