ADVANTEST®
ADVANTEST CORPORATION

HANDBOOK
R3261/3361 SERIES
OPTION 15
GUIDE

MANUAL NUMBER OEB00 9112

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

First printing July 17, 1989
Printed in Japan

PREFACE

1. This manual describes Option 15 in a question and answer form for easier understanding.

2. This manual consists of the following contents:

   1. Preparation before application of Option 15

   2. Creation of program

   3. Execution of program

   4. Applied operation

   You can understand the functions and practical operating method of Option 15 if you read this manual from Chapter 1.

   Refer to the separated "R3261/3361 Series Option 15 (Reference)" for details.

3. The following abbreviations are used in this manual:

   HP : Personal computer manufactured by Hewlett-Packard Co.
   VG-920: Terminal manufactured by Victor Data Systems Co.
   VT-220: Terminal manufactured by DEC

4. External terminals connected to the R3261/3361 are VG-920, VT-220 and equivalents. VG-920 is used and referred to in this manual.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

1. Preparation Before Application of Option 15

Q1. What do you need to prepare to create the programs?

A1. You have to prepare a measuring device and external terminal (VG-920) in order to create a BASIC program using the controller function (ate editor+BASIC interpreter). You have to connect the external terminal to the RS-232C connector on the back panel of the measuring device. Arrange the wiring as follows:

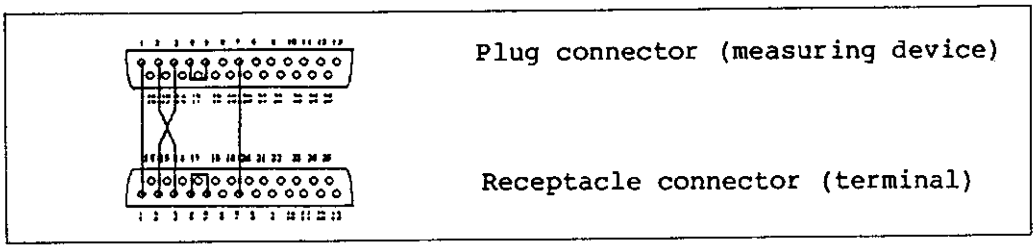
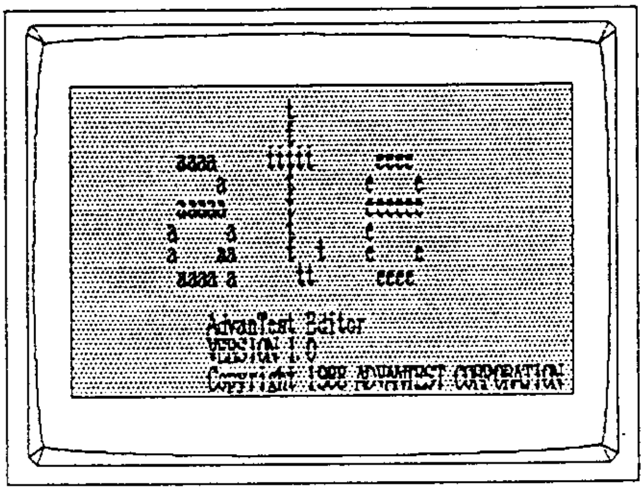

Plug connector (measuring device)
Receptacle connector (terminal)

Figure 1 Wiring of RS-232C Cables

In addition, you have to prepare the memory card for the storing program (standard equipment (32Kbyte)) and a GPIB cable.

Q2. How do you start the controller?

A2. First, turn on the power supply switch of the terminal. When the message "VG-920 OK" is displayed, turn on the power supply switch of the measuring device. When the following initial screen is displayed with buzzer sound, the controller starts the operation.


ate

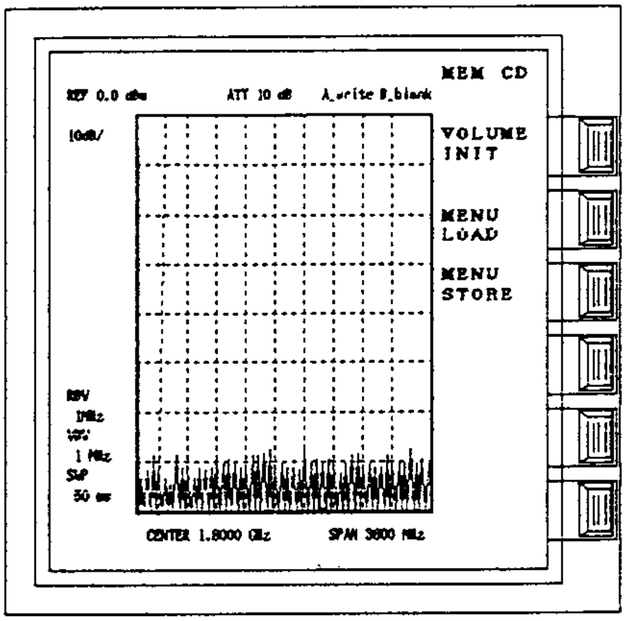
AdvanTest Editor
VERSION 1.0
Copyright 1988 ADVANTEST CORPORATION

Figure 2 Initial Screen of ate Editor

Q3. How do you initialize the memory card?

A3. Insert the memory card into the measuring device and operate it as follows:
However, if you initialize the card saved data, all saved data will be deleted.

Press SHIFT key and 4 key.

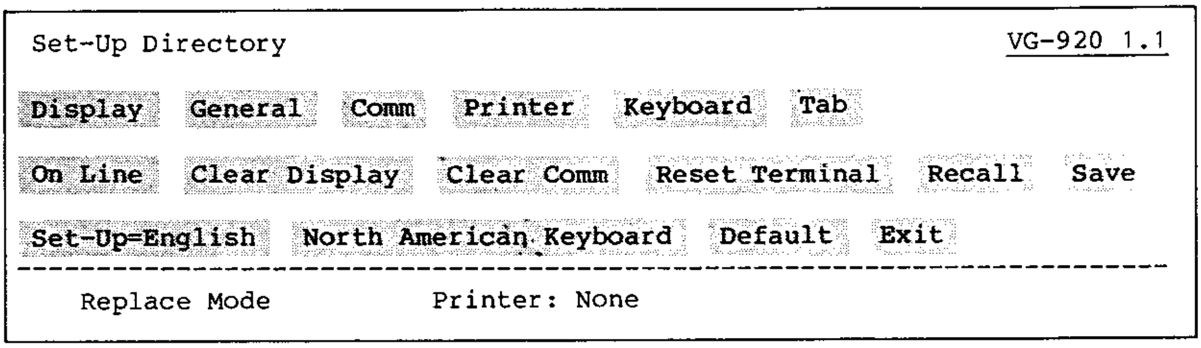


Press VOLUME INIT key.

Figure 3 Initialization Procedure of the Memory Card

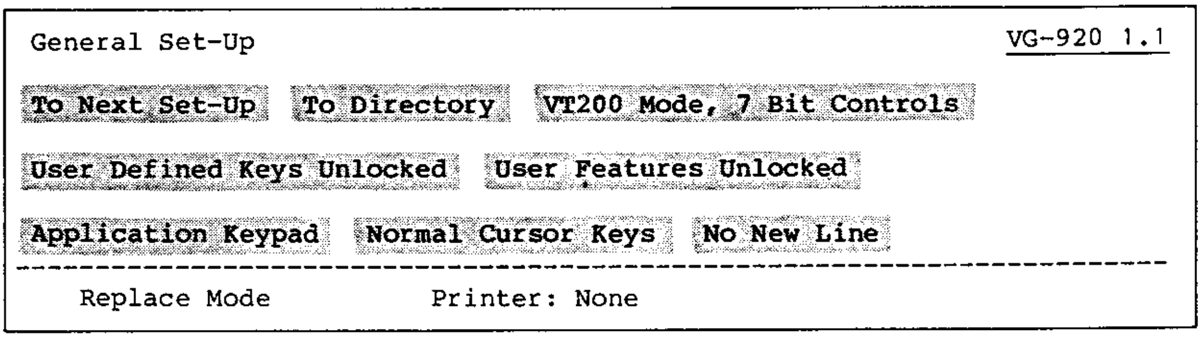
Q4. How do you set the terminal?

A4. The following is a typical setting procedure:
Refer to the instruction manual of VG-920 for details.



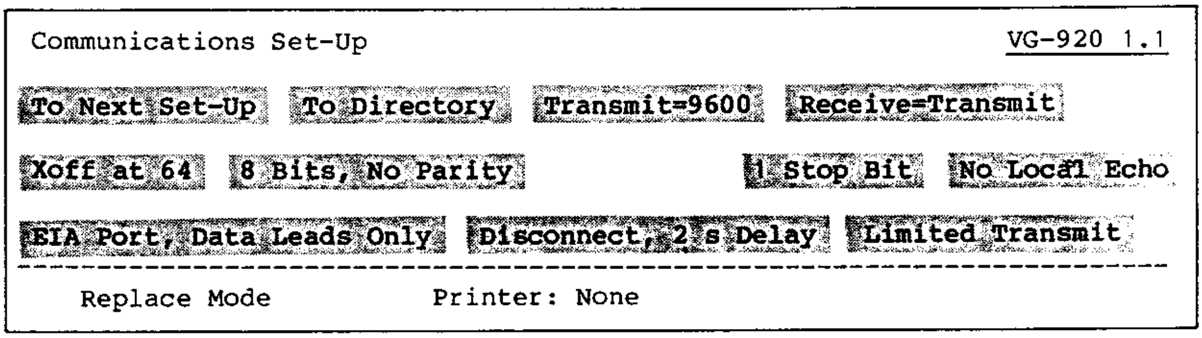
```
Set-Up Directory                                              VG-920 1.1

Display  General  Comm  Printer  Keyboard  Tab

On Line  Clear Display  Clear Comm  Reset Terminal  Recall  Save

Set-Up=English  North American Keyboard  Default  Exit
---------------------------------------------------------------------------
   Replace Mode          Printer: None
```

Figure 4 Set-Up Directory



```
General Set-Up                                                VG-920 1.1

To Next Set-Up  To Directory  VT200 Mode, 7 Bit Controls

User Defined Keys Unlocked  User Features Unlocked

Application Keypad  Normal Cursor Keys  No New Line
---------------------------------------------------------------------------
   Replace Mode          Printer: None
```

Figure 5 General Set-Up



```
Communications Set-Up                                         VG-920 1.1

To Next Set-Up  To Directory  Transmit=9600  Receive=Transmit

Xoff at 64  8 Bits, No Parity            1 Stop Bit  No Local Echo

EIA Port, Data Leads Only  Disconnect, 2 s Delay  Limited Transmit
---------------------------------------------------------------------------
   Replace Mode          Printer: None
```

Figure 6 Communications Set-Up

MEMO

2. Creation of Program

Q5. How do you create a program?

A5. Since the controller starts operation when you turn on the power supply switches of the measuring device and terminal, key-in the data using the terminal.

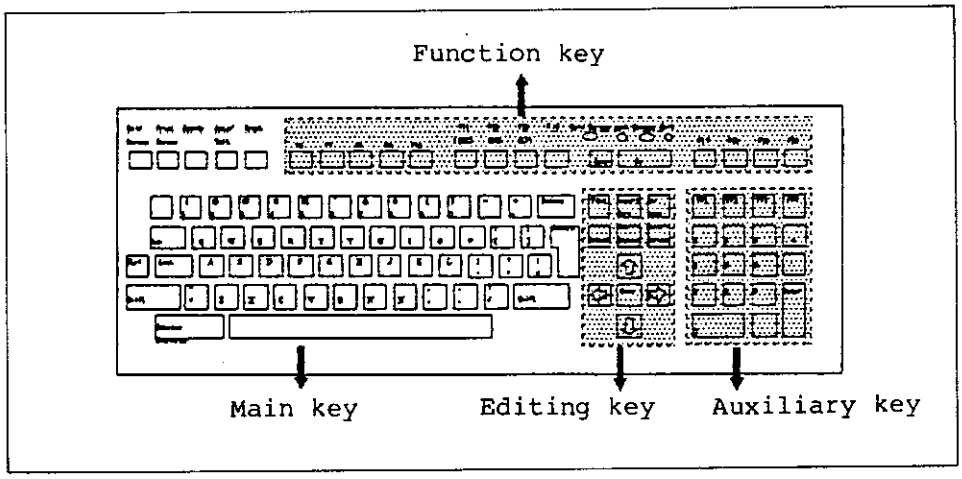


Figure 7 VG-920 Keyboard

The following is an example of the simple program:

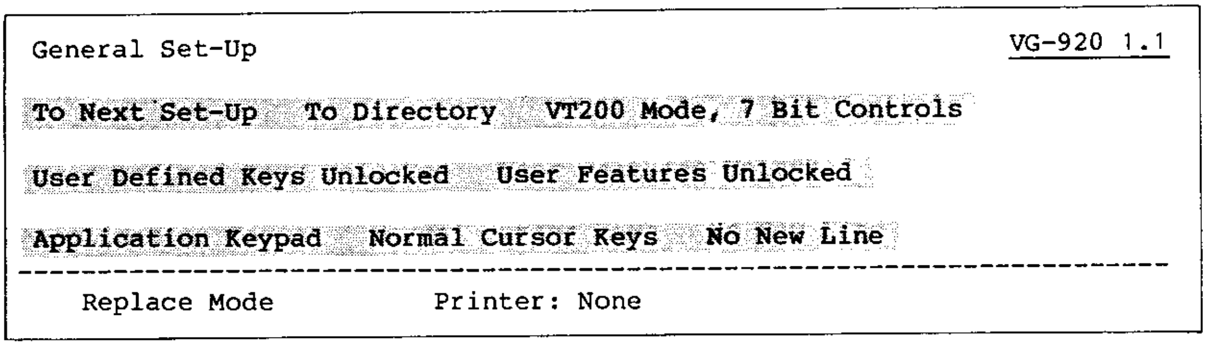
```
TOTAL=0
FOR I=1 TO 100
  TOTAL=TOTAL+I
NEXT I
PRINT "1~100 TOTAL = ",TOTAL
STOP
END
```

Figure 8 Example of Simple Program

Q6. The machine does not execute the auxiliary functions even if you press the auxiliary keypad.

A6. Since the setting of the auxiliary keypad is in a numeral mode instead of editing mode, the editing mode is not recognized. Reset the auxiliary keypad to the application keypad.

See Figure 7 for arrangement of the auxiliary keypad.



```
General Set-Up                                              VG-920 1.1

To Next Set-Up   To Directory   VT200 Mode, 7 Bit Controls

User Defined Keys Unlocked   User Features Unlocked

Application Keypad   Normal Cursor Keys   No New Line
-----------------------------------------------------------------------
   Replace Mode             Printer: None
```

Figure 9 Setting of Auxiliary Keypad

Q7. I cannot key-in the data correctly. How do you key-in in the data correctly?

A7. Check if the terminal setting is "**VT200 Mode, 7 Bit Controls**" or if the transfer rate is correct?

Q8. How do you use the popup menu?

A8. Press "Do" key or "Ctrl-A" key of the function keypad. Then, the popup menu is displayed at the top of the screen. Select the menu with the cursor key and press the "Return" key.

See Figure 7 for arrangement of the function keypad.

BASIC	Region	Window/Other	File	Search/Replace
MOVE and RUN BASIC mode RUN CONT Line No. Renumbering	Set mark Kill region Copy region Yank	Only Split Next Redisplay Help SCRATCH	Visit Save Write	Forward search Backward search Query replace

Figure 10 Popup Menu List

Q9. How many lines can this editor create in a program?

A9. Suppose there are 40 letters in one line with no line numbers, the editor can create a program with about 2500 lines.

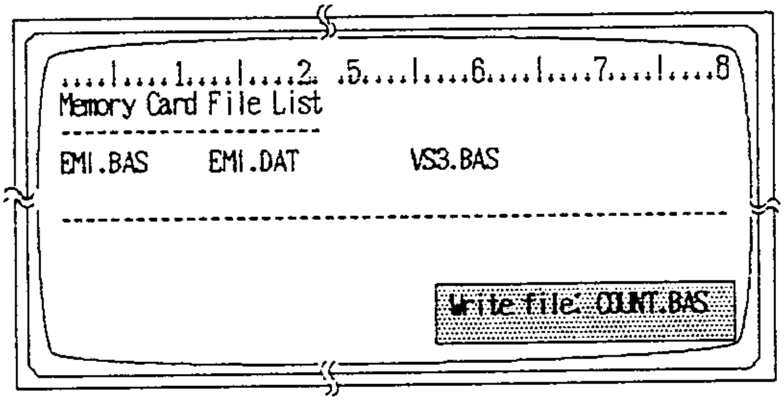
Q10. The execution result of the program created in lowercase letters is incorrect.

A10. Since the lowercase letters in the program are recognized as variables, the machine outputs the data different from what you expected.
Therefore, use uppercase letters for programming. Use lowercase letters for variables is placed.

Q11. How do you save the created program in the memory card?

A11. First, insert the card into the measuring device. Next, select "Write" in the popup menu or press the "9" key of the auxiliary keypad. The mini window is displayed at the lower right-hand portion of the screen. Then, input the file name to be saved. The number of lines written will be displayed if the file is correctly saved.

See Figure 7 for the arrangement of the auxiliary keypad.

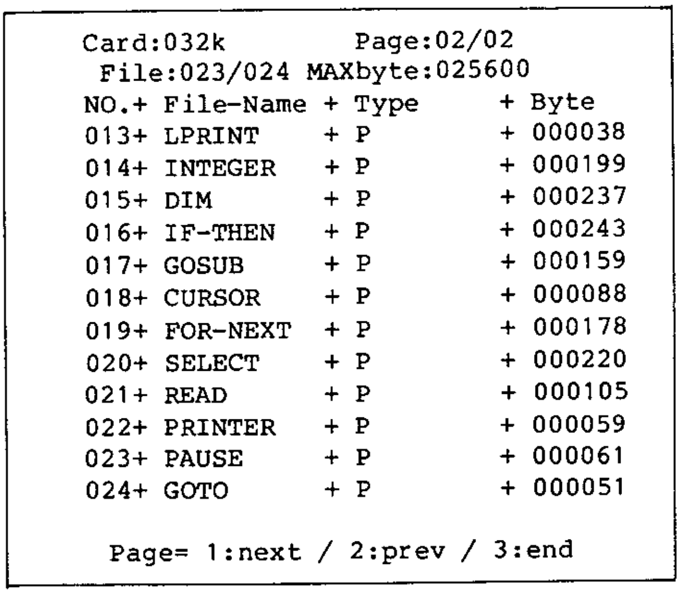


```
....|....1....|....2  .5....|....6....|....7....|....8
Memory Card File List
---------------------
EM1.BAS     EM1.DAT        VS3.BAS
-----------------------------------------------------

                               Write file: COUNT.BAS
```

Figure 11 Saving a File

Q12. I cannot save a file in the memory card.

A12. Is the "Write protect" switch of the memory card turned on? Is the card full? You can check the remaining space in the card with the CAT command in the BASIC mode.



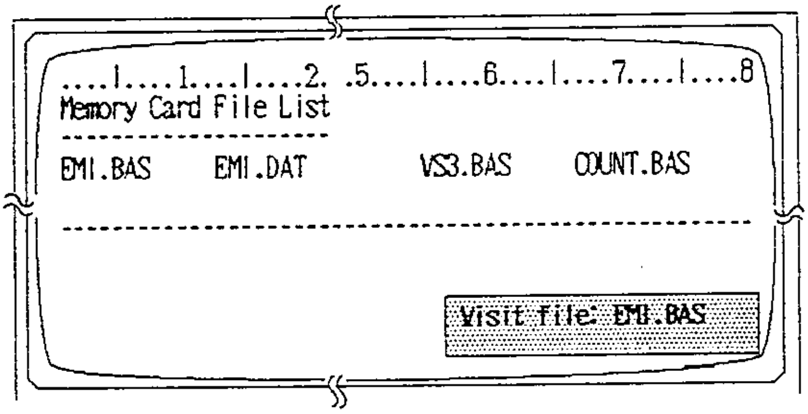
```
Card:032k          Page:02/02
 File:023/024 MAXbyte:025600
NO.+ File-Name + Type      + Byte
013+ LPRINT    + P         + 000038
014+ INTEGER   + P         + 000199
015+ DIM       + P         + 000237
016+ IF-THEN   + P         + 000243
017+ GOSUB     + P         + 000159
018+ CURSOR    + P         + 000088
019+ FOR-NEXT  + P         + 000178
020+ SELECT    + P         + 000220
021+ READ      + P         + 000105
022+ PRINTER   + P         + 000059
023+ PAUSE     + P         + 000061
024+ GOTO      + P         + 000051

  Page= 1:next / 2:prev / 3:end
```

Figure 12 Execution of CAT Command

Q13. How do you load the file from memory card?

A13. First, insert the memory card into the measuring device. Next, select **"Visit"** in the popup menu or press the "7" key of the auxiliary keypad. The screen is divided into two windows. The top window displays a file list and the bottom section displays a mini window. Then, input the file name to be loaded.
If your file is not there, an error message is displayed.

See Figure 7 for the arrangement of the auxiliary keypad.



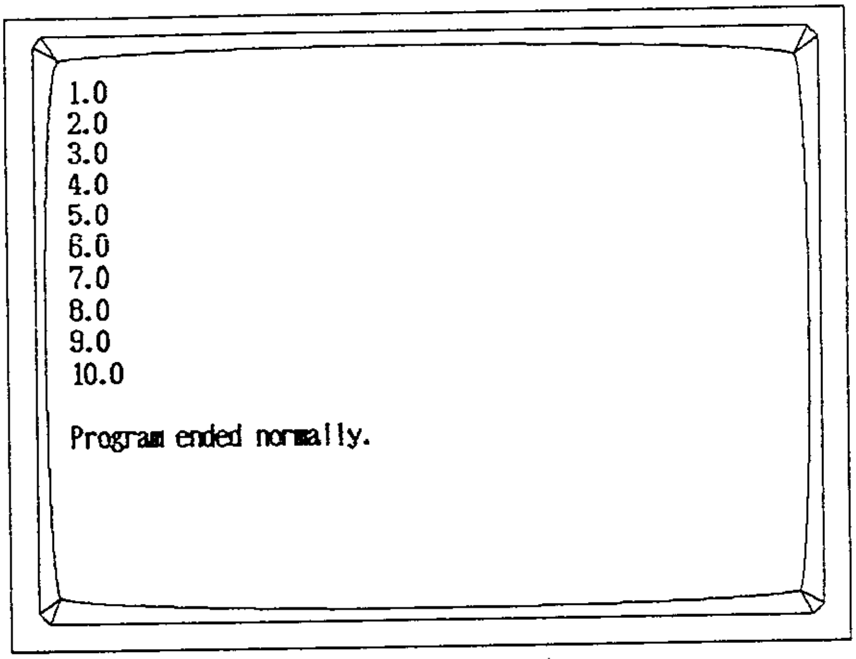
```
....|....1....|....2. .5....|....6....|....7....|....8
Memory Card File List
---------------------
EMI.BAS     EMI.DAT        VS3.BAS     COUNT.BAS
-------------------------------------------------------

                                  Visit file: EMI.BAS
```

Figure 13 Loading a File

3. Execution of Program

Q14. How do you execute the program?

A14. Select "**MOVE and RUN**" in the popup menu or press the "**F17**" key of the function key. Then, the program is executed and the result is displayed.



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

Program ended normally.
```

Figure 14 Execution Result of Program

Q15. How do you display the execution result on the measuring device screen?

A15. If an external terminal is connected to the measuring device, the execution result cannot be displayed on the measuring device screen. If there is only the measuring device, the result can be displayed on its screen. This time, select the following display conditions with the GPIB command:

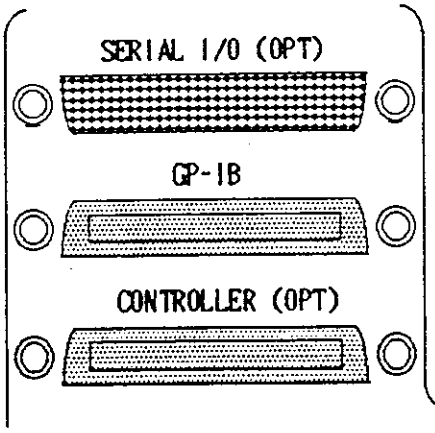
Display condition	Screen setting	Delete the screen	
	GPIB command	BASIC command	
Waveform only	OUTPUT 31;"VS0"		
Waveform + execution result	OUTPUT 31;"VS1"		
Execution result only	OUTPUT 31;"VS2"	CLS	Simultaneous
Graphic	OUTPUT 31;"VS3"	CLS 1	CLS 2

Figure 15 Displaying the Conditions of Measuring Device

However, only graphics are displayed on the measuring device screen if the external terminal is connected to it. But they are not displayed on the external terminal.

Q16. How do you print the program source list with the GPIB printer?

A16. First, connect the printer GPIB port to the measuring device GPIB port (controller side), then turn on the power supply switch.



- Connection of external terminal (VG-920)
- GPIB control from external personal computer
- Block output of waveform
- Printer output of program source list
- Data output to the external device

Figure 16 GPIB Port on Rear Panel of Measuring Device

Next, select **"BASIC mode"** in the popup menu or press the **"F18"** key of the function key. You can input the following command in the mini window displayed at lower right-hand side of the screen. However, **"MOVE and RUN"** must be executed beforehand.

Input procedure	Explanation
PRINTER 3 GLIST	Specify the printer. ("3" is the GPIB address) Output the program source list with the GPIB printer.

Figure 17 Source List Printer Output

Q17. How do you output the execution result with the GPIB printer?

A17. First, connect the printer and measuring device with cables using the same procedure in A16. Next, input the following command in the BASIC mode.

Input procedure	Explanation
PRINTER 3 GPRINT	Specify the printer. ("3" is the GPIB address) Output the execution result by the GPIB printer.

Figure 18 Execution Result Printer Output

Q18. How do you automatically load the multiple programs from the memory card for execution?
Also, how do you reside the program in the internal memory for execution?

A18. Both are not available. The ate editor loads only one program, and the BASIC interpreter executes only the program saved in the ate editor. Therefore, you cannot perform the above executions.

4. Applied Operation

Q19. How do you set the data in the measuring section?

A19. Use the OUTPUT command. The address number is 31. In "31", the communication is done internally without passing through the external GPIB cable. The following shows a sample program.

```
OUTPUT 31;"IP"                ! Preset
OUTPUT 31;"CF200MZ"           ! Set the center frequency 200MHz.
OUTPUT 31;"SP100KZ"           ! Set the frequency span 100kHz.
WAIT 200                      ! Wait approx. 200ms
OUTPUT 31;"PS"                ! Peak search
```

Figure 19 "OUTPUT 31" Sample Program

Q20. How do you read the measurement data from the measuring section?

A20. Use the ENTER command. The address number is 31. In "31", the communication is done internally without passing through the external GPIB cable. Specify the data to be read with the OUTPUT command. Place "?" at the end of the GPIB command like "OUTPUT 31;"MF?". The following shows a sample program:

```
OUTPUT 31;"IP"                ! Preset
OUTPUT 31;"CF200MZ"           ! Set the center frequency 200MHz.
OUTPUT 31;"SP100KZ"           ! Set the frequency span 100kHz.
WAIT 200                      ! Wait approx. 200ms
OUTPUT 31;"PS"                ! Peak search
OUTPUT 31;"MF?"               ! Read the peak frequency.
ENTER 31;MKRF                 ! Read the peak frequency.
PRINT "MARKER FREQ = ",MKRF   !
OUTPUT 31;"ML?"               ! Read the peak level.
ENTER 31;MKRL                 ! Read the peak level.
PRINT "MARKER LEVEL = ",MKRL  !
```

Figure 20 "ENTER 31" Sample Program

Q21. How do you read or write the data in the parallel I/O?

A21. Use the OUTPUT and ENTER commands. The address number is 32. 16 bit data is available in the parallel I/O. The following shows a sample program:

```
A=1
OUTPUT 32;A                     ! Set the data.
!
ENTER 32;B                      ! Read the data.
  SELECT B
    CASE 1:   TOTAL=1
    CASE 16:  TOTAL=16
    CASE 256: TOTAL=256
  END SELECT
STOP
```

Figure 21 Sample Program Using Parallel I/O

Q22. How do you read or write the data in the serial I/O?

A22. You cannot read the data form the serial I/O. Use the LPRINT command to write the data. This time, since you have to set the conditions such as transfer rate, you can set them with the CONTROL command. The data is then output in ASCII code. The following shows a sample program.

```
CONTROL 1;2+12+0+64             ! 4800bps, 8Bit, No Parity, Stop 1 Bit
LPRINT                          ! Printer output
STOP
! SAMPLE PROGRAM                ! Source program
!
!
```

Figure 22 Sample Program Using Serial I/O

Q23. How do you execute the program in conversation form?

A23. Use the ON KEY command or INPUT command. You can specify the range from 1 to 6 in the ON KEY command. Set the INPUT command by pressing the "Return" key or "Unit" key. Key-in of the numerals is available without connecting to the terminal. Input of letters or negative numbers is available only by the keyboard. It is not available if there is only the measuring device. The following shows a sample program:

```
ON KEY 1 GOTO *KEY1           ! Specify the ON KEY command.
ON KEY 2 GOTO *KEY2
ON KEY 3 GOTO *KEY3
*L
  ENABLE INTR                 ! Approve the interruption.
  PRINT" >> key in !"
*LL
  GOTO *LL                    ! Wait for the interruption.
!
*KEY1
  DISABLE INTR: INPUT "---> key 1 OK? (Y=1/N=0)",I
                              ! Interruption is disabled and execute
                                the INPUT command.
  IF I=0 THEN GOTO *KEY1
  GOTO *L
*KEY2
  DISABLE INTR: INPUT "---> key 2 OK? (Y=1/N=0)",I
  IF I=0 THEN GOTO *KEY2
  GOTO *L
*KEY3
  DISABLE INTR: INPUT "---> key 3 OK? (Y=1/N=0)",I
  IF I=0 THEN GOTO *KEY3
  GOTO *L
STOP
```

Figure 23 Sample Program for Executing Conversation Form

Q24. How do you control the other measuring device by the GPIB?

A24. Use the OUTPUT command. The address number is 0 to 30. The following shows a sample program:

```
X$=1500:Y$=600                  ! Draw a real line from (1500,600)
                                  to (2000,600).
OUTPUT 5;"PU;PA"&X$&","&Y$&";"
                                ! The GPIB address is 5.
X$=2000;Y$=600
OUTPUT 5;"PD;PA"&X$&","&Y$&";"
```

Figure 24 Sample Program for Plotter Control

Q25. How do you save the data in the program execution stage in the memory card?

A25. Use the OPEN/CLOSE and OUTPUT commands. If a file is already saved there, delete it, then, save the new file. The following shows a sample program:

See the OPEN/CLOSE command for details.

```
DIM A(100)
FILE$="ABC"                     ! The file name is "ABC."
INPUT "Is ¥"ABC¥" file new ? (Y=1/N=0) ->",NEW
IF NEW=0 THEN PURGE FILE$
OPEN FILE$ FOR OUTPUT AS #FD;ASCII
                                ! Open the "ABC" file.
FOR I=1 TO 100
  A(I)=I
  OUTPUT #FD;A(I)               ! Write the data.
  PRINT "Save = ",A(I)
NEXT I
CLOSE #FD                       ! Close the "ABC" file.
STOP
```

Figure 25 Sample Program for Saving Variables

Q26. How do you load the data saved in the memory card?

A26. Use the OPEN/CLOSE and ENTER commands. The file to be loaded must already exist. Also, it must be written in the same format. The following shows a sample program:

See the OPEN/CLOSE command for details.

```
DIM B(100)
FILE$="ABC"                   ! The file name is "ABC."
OPEN FILE$ FOR INPUT AS #FD;ASCII
                              ! Open the "ABC" file.
FOR I=1 TO 100
  ENTER #FD;B(I)              ! Read the data.
  PRINT "Load = ",B(I)
NEXT I
CLOSE #FD                     ! Close the "ABC" file.
STOP
```

Figure 26 Sample Program for Loading Data

Q27. How do you read the trace data in the variables?

A27. Use the GTA/GTB command and built-in function RTRACE(). The GTA/GTB command reads the trace data in the work area of the measuring device. The GTA/GTB command must be executed before the built-in function RTRACE() is used. The following shows a sample program:

```
INTEGER T1(701)                  ! Specify the storage area.
!
OUTPUT 31;"GTA"                  ! Read the trace A in the work area.
FOR I=1 TO 701
  T1(I)=RTRACE(I-1,0)            ! Read the trace data in the variables.
NEXT I
STOP
```

Figure 27 Sample Program for Reading the Trace Data

Use the PTA/PTB command and built-in function WTRACE() when you want to write the trace data. The PTA/PTB command transfers the data written in the work area by the built-in function WTRACE() to the trace memory in the measuring device. The following shows a sample program:

```
INTEGER T1(701)                  ! Specify the variable.
!
FOR I=1 TO 701
  WTRACE(T1(I),I-1,1)            ! Write the data to the work area.
NEXT I
OUTPUT 31;"PTB"                  ! Write the data, which is in the work
                                 ! area, in trace B.
OUTPUT 31;"BV"                   ! Set B view.
STOP
```

Figure 28 Sample Program for Writing the Trace Data

The following sample program saves the trace A data in the memory card and writes it to trace B.

```
! trace A               ->Memory card (FILE$)
! Memory card (FILE$)   ->trace B
INTEGER T1(701),T2(701)
!
GOSUB *SETUP                     ! Set up
GOSUB *FSAVE                     ! Save the data in the memory card.
GOSUB *FLOAD                     ! Load the data from the memory card.
!
*SETUP
  OUTPUT 31;"VS2":CLS
  PRINT "##### OPEN/CLOSE #####"
  BUZZER 1000,500
  CURSOR 5,5:PRINT ">>Please, write protect switch is off !"
  CURSOR 5,7:INPUT ">>Save file name = ?", FILE$
                                 ! Input the file name.
  CURSOR 5,9:INPUT ">>New file ? (Y=1/N=0) ",NEW
  IF NEW=0 THEN PURGE FILE$      ! Current file is deleted.
  OUTPUT 31;"IP VS1":CLS
  OUTPUT 31;"CLN CF30MZ SP1MZ RE-10DB RB100KZ"
  RETURN
!
*FSAVE
  OUTPUT 31;"VS1":CLS
  BUZZER 500,500
  PRINT "trace A ->Card saving .. (file) = ",FILE$
  OPEN FILE$ FOR OUTPUT AS #FD;ASCII
                                 ! Open the file.
```

Figure 29 Sample Program for Read/Write
           the Trace Data from/to the Memory Card

```
   OUTPUT 31;"GTA"                ! Read trace A.
   FOR I=1 TO 701
     T1(I)=RTRACE(I-1,0)          ! Save the data in the variables.
     OUTPUT #FD,T1(I)             ! Save the data in the memory card.
   NEXT I
   CLOSE #FD                      ! Close the file.
   OUTPUT 31;"AB"                 ! A blank
   CLS:BUZZER 500,500
   RETURN
!
*FLOAD
   OUTPUT 31;"VS1":CLS
   BUZZER 500,500
   OUTPUT 31;"CWB BV"             ! B clear & view
   CLS:PRINT "Card loading .. ->trace B (file) = ",FILE$
   OPEN FILE$ FOR INPUT AS #FD;ASCII
                                  ! Open the file.
   FOR I=1 TO 701
     ENTER #FD;T2(I)              ! Load the data from the memory card.
     WTRACE(T2(I),I-1,1)          ! Write the data in the work area.
   NEXT I
   OUTPUT 31;"PTB"                ! Write the data in trace B.
   OUTPUT 31;"BV"                 ! B view
   CLOSE #FD                      ! Close the file.
   CLS:BUZZER 500,500
   RETURN
```

Figure 29 Sample Program for Read/Write the Trace Data from/to the Memory Card (cont'd)

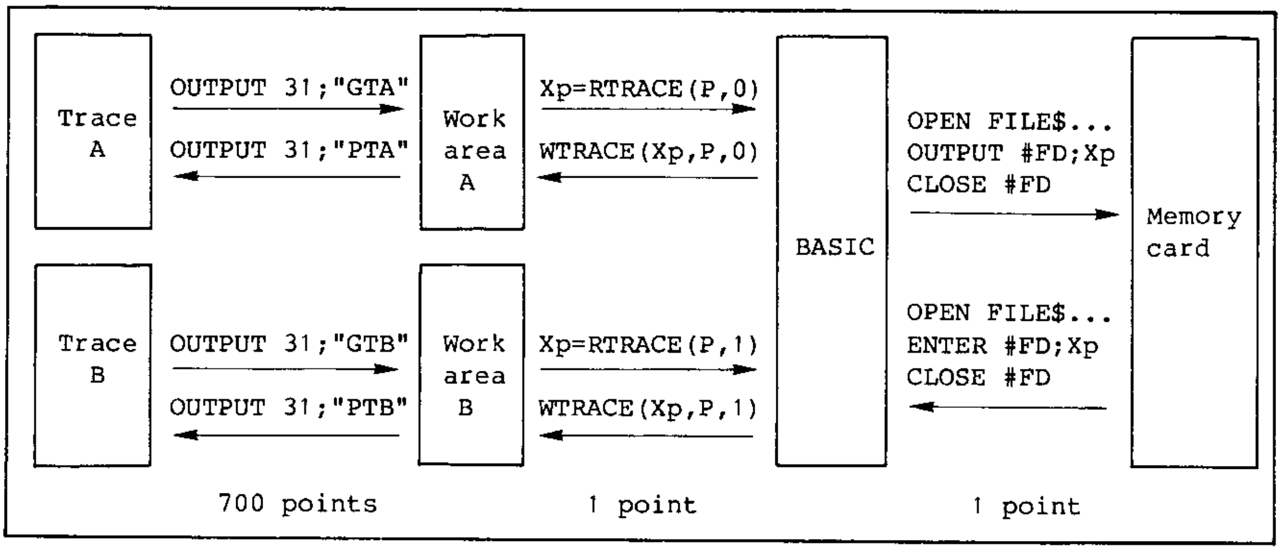


Figure 30 Data Transfer Between the Trace and Memory Card

Q28. How do you use the service request (SRQ)?

A28. Use the SPOLL and bit operator. Interruption can be applied. The following shows a sample program using the interruption function:

```
INTEGER S
SPA=31
ON ISRQ GOSUB *SRQCHK          ! Specify the interruption.
OUTPUT SPA;"IP"
OUTPUT SPA;"VS1 S0"
!----- Calibration -----!
DATA "CLG","IT0","IT1","IT2","IT3","IT4"
FUNC$="CAL"
RESTORE
FOR I=0 TO 5
  READ CAL$
  OUTPUT SPA;CAL$
  GOSUB *SWAIT
  WAIT 1000
NEXT I
!----- Sweep end -----!
FUNC$="SWP"
OUTPUT SPA;"IP CLN CF30MZ SP500KZ SW2SC SI"
FOR I=0 TO 5
  OUTPUT SPA;"SP DN SR"
  GOSUB *SWAIT
NEXT I
STOP
!
!----- SRQ interrupt wait -----!
*SWAIT
  FLAG=0
  ENABLE INTR                  ! Approve the interruption.
*LL
  IF FLAG=1 THEN RETURN        ! Wait for the interruption.
  GOTO *LL
!----- SRQ status check -----!
*SRQCHK
  DISABLE INTR                 ! Interruption is disabled.
  S=SPOLL(SPA)                 ! Status polling
  IF FUNC$="CAL" AND (S BAND 2) <>0 THEN
    GOSUB *CALEND: RETURN
  END IF
```

Figure 31 Sample Program Using SQR Function

```
  IF FUNC$="SWP" AND (S BAND 4)<>0 THEN
    GOSUB *SWPEND: RETURN
  END IF
  ENABLE INTR                    ! Approve the interruption.
  RETURN
!
!----- Calibration end -----!
*CALEND
  BUZZER 500,50
  PRINT "CAL. end -->",CAL$
  FLAG=1
  RETURN
!
!----- Sweep end -----!
*SWPEND
  BUZZER 500,50
  FLAG=1
  FREQ=BND(PMAX(0,700,0),10,0)
  PRINT "10 dB down band = ",FREQ/1000,"kHz"
  RETURN
```

Figure 31   Sample Program Using SQR Function (cont'd)

Q29. How do you transfer the program created by the personal computer to the ate editor?

A29. First, change the branched statement having the line number (example: GOTO 100) in the program to be transferred into the label, and save it as an ASCII file.
Next, create the control program. The control program transfers the program to the measuring device in the following procedure:

(1) Load the program to be transferred.
(2) Delete the line number.
(3) Place '@' at the beginning of the line.
(4) Transfer the line separately to the measuring device.

Figure 32 Transfer Procedure

The transfer of one line needs a waiting time of approx. 1 second. If the transfer cannot be made smoothly, you have to increase the waiting time. Make sure that one line does not exceed 128 bytes.

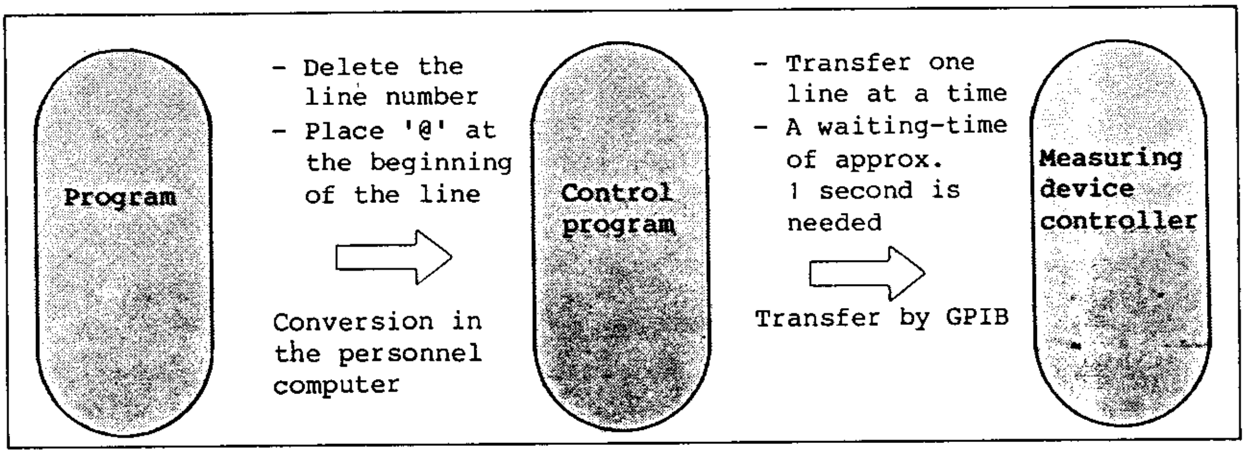
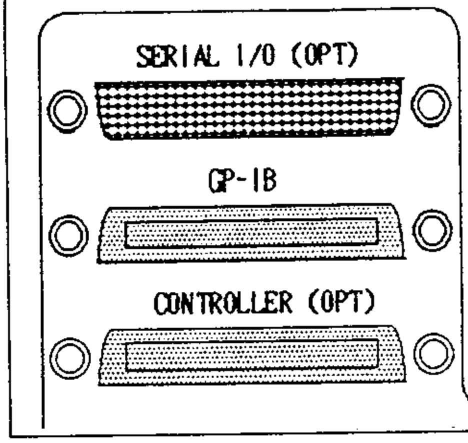

Program
- Delete the line number
- Place '@' at the beginning of the line
Conversion in the personnel computer
Control program
- Transfer one line at a time
- A waiting-time of approx. 1 second is needed
Transfer by GPIB
Measuring device controller

Figure 33 Outline of Program Transfer

When you complete the setting for the transfer on the external personal computer, you have to set the measuring device.
First, connect the GPIB port on the rear panel of measuring device and GPIB port of the external personal computer with a cable.



- Connection of external terminal (VG-920)

- Connection of external personal computer.

Figure 34 GPIB Port of on the Rear Panel of Measuring Device

Next, connect the external terminal to the measuring device. Then, turn on the power supply switch of the measuring device. When the initial screen of the ate editor is displayed on the external terminal, you can execute the control program created on the external personal computer.

Then, the program is transferred from the external personal computer to the measuring device starts.

After completing the transfer, you have to check the existence of the program by the ate editor. If the data is insufficient, the waiting time is not enough. Therefore, you have to reset a longer waiting time and re-transfer the data.

Note that you have to execute "SCRATCH" in the popup menu to initialize the editor before transferring the program.

The following control program sample is of the HP version:

HP version control program

```
10    DIM A$[128],B$[128]
20    Adrs=708                        ! GPIB address
30    !
40    Restart:!
50      INPUT "Down load file name = ? ",Name$
                                      ! Input the file name.
60      MASS STORAGE IS ":HP913X,701"
                                      ! Specify the disk.
70      ASSIGN @Fd TO Name$           ! Open the file.
80      ON END @Fd GOTO End_load
90    D_load:!
100     ENTER @Fd;A$                  ! Load the program to be transferred.
110     Cnt=1                         ! Delete the line number (up to 190)
120     LOOP                          !
130       Chk$=A$[Cnt,Cnt]            !
140     EXIT IF Chk$=" "              !
150       A$[Cnt,Cnt]=""              !
160       Cnt=Cnt+1                   !
170     END LOOP                      !
180     B$=""                         !
190     B$=A$[Cnt,127]                !
200     B$="@"&B$                     ! Place '@' at the beginning of the
                                      ! line.
210     PRINT B$
220     OUTPUT Adrs;B$                ! Transfer one line.
230     WAIT 1                        ! Waiting time of approx. 1 second
240     IF B$<>"65525 END" THEN GOTO D_load
250     GOTO 250
260   !
270   End_load:!
280     ASSIGN @Fd TO *               ! Close the file.
290     DISP "Down load end !!"
300     END
```

MEMO

ADVANTEST®
ADVANTEST CORPORATION

HANDBOOK
R3261/3361 SERIES
OPTION 15
REFERENCE

MANUAL NUMBER OEC00 9307

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

First printing July 25, 1989
Printed in Japan

PREFACE

1. This handbook consists of the following contents:

PART I	ate EDITOR
PART II	SYSTEM CONTROLLER
APPENDIX	Error Message List

2. R3261/3361 SERIES OPTION 15 HANDBOOK GUIDE is provided as a separated volume to describe briefly the functions and the operating method of the option 15 in the questions and answers form.
   Read R3261/3361 SERIES OPTION 15 HANDBOOK GUIDE before reading this handbook.

3. The following abbreviations are used in this handbook:

   HP   : Personal computer manufactured by Hewlett-Packard Co.
   VG-920: Terminal manufactured by Vcitor Data System Co.
   VT-220: Terminal manufactured by DEC

4. External terminals connected to the R3261/3361 are VG-920, VT-220 and equivalents.
   VG-920 is used and referred to in this manual.

TABLE OF CONTENTS

【PART I】

【PARTⅡ】

【APPENDIX】

PART I ate EDITOR

This handbook Part1 separately describes the ate editor using the measuring instrument together with the external terminal VG-920 and that using only the measuring instrument.

(1) Chapers 2 through 5: Description of operation using the measuring instrument together with the external terminal VG-920

Chapter 2: Features
Chapter 3: Starting
Chapter 4: Function list
Chapter 5: description of Functions

(2) Chapers 6 through 9: Description of operation using only the measuring instrument

Chapter 6: Features
Chapter 7: Starting
Chapter 8: Function list
Chapter 9: description of Functions

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) connected to the RS-232C connector on the panel at the back of the measuring instrument, the operability is greatly improved. Editing can also be made only with the measuring instrument. Therefore the editor has portability such as program correction at the site.

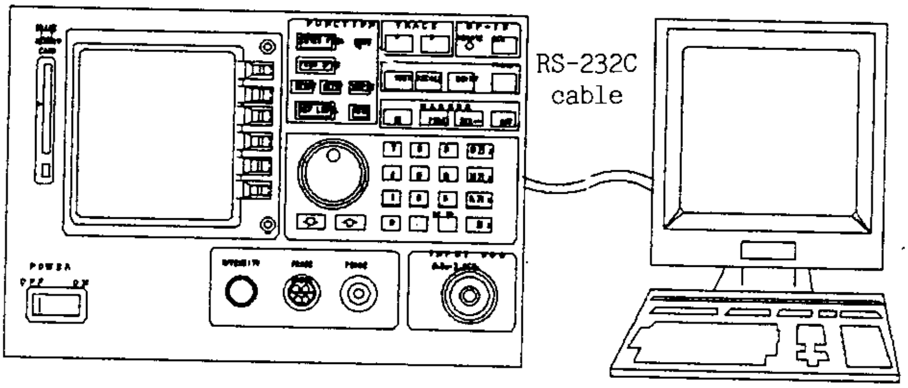


Figure 1 - 1 Connection Between R3261/3361 and VG-920

MEMO

2. FEATURES (When using the measuring instrument together with the external terminal)

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.

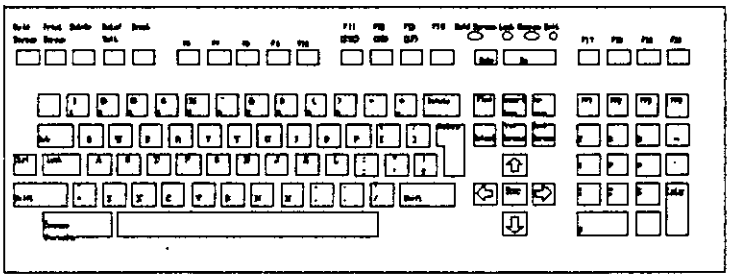


Figure 2 - 1 Keyboard of VG-920

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.

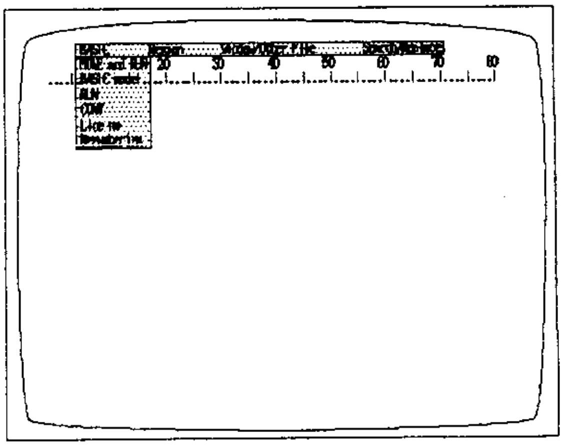


Figure 2 - 2 Pop-up Menu

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.

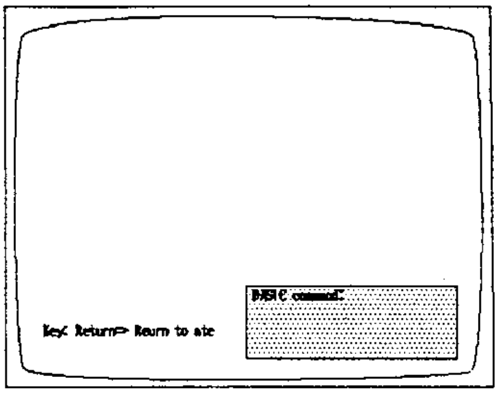
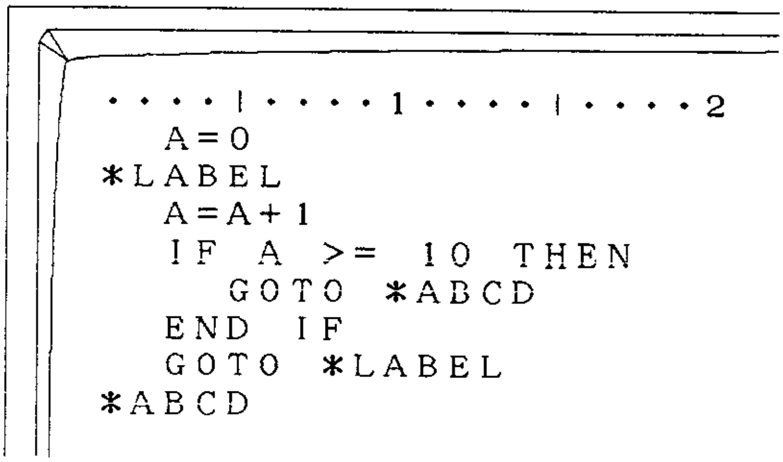


Figure 2 - 3 BASIC Mode

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.

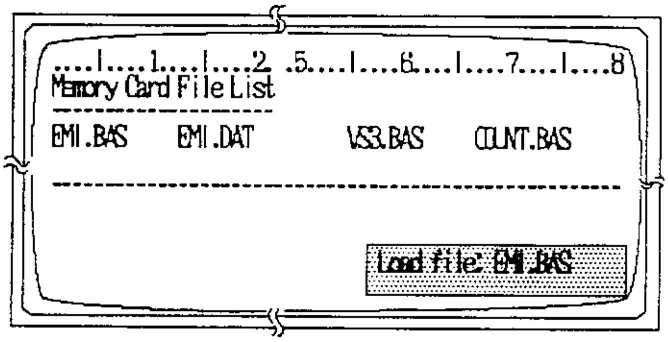


```
....|....1....|....2
  A=0
*LABEL
  A=A+1
  IF A >= 10 THEN
    GOTO *ABCD
  END IF
  GOTO *LABEL
*ABCD
```

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.

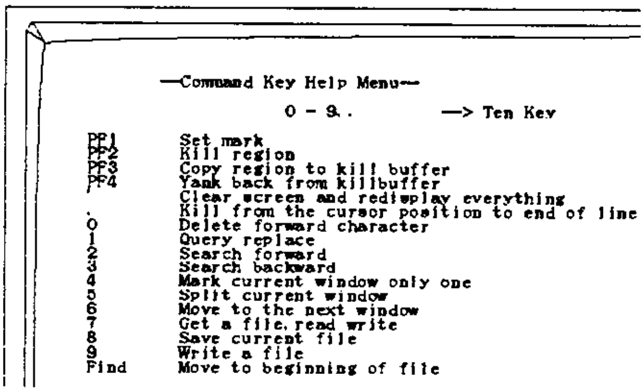


```
....|....1....|....2 .5....|....6....|....7....|....8
Memory Card File List
---------------------
EMI.BAS     EMI.DAT         VS3.BAS     COUNT.BAS
------------------------------------------------------

                                  Load file: EMI.BAS
```

Figure 2 - 5 File Loading

2.6 Help Menu

The functions assigned to the keyboard of the VG-920 are listed on the screen.



```
        —Command Key Help Menu—

                0 - 9. .        —> Ten Key

PF1     Set mark
PF2     Kill region
PF3     Copy region to kill buffer
PF4     Yank back from killbuffer
/       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
```

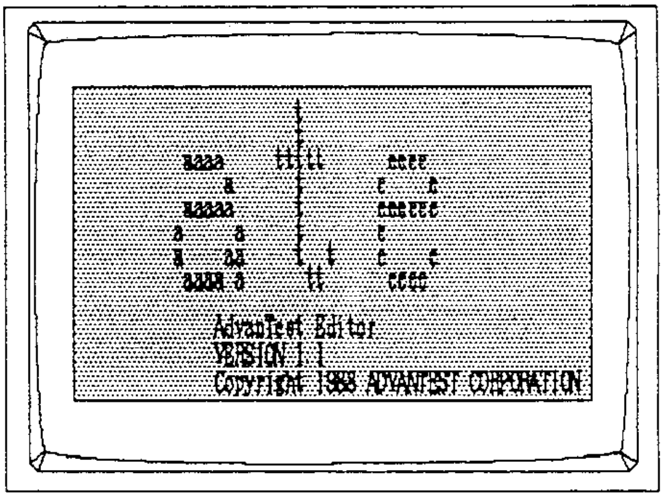
Figure 2 - 6 Help Menu

MEMO

3. STARTING (When using the measuring instrument together with the external terminal)

(1) Operation

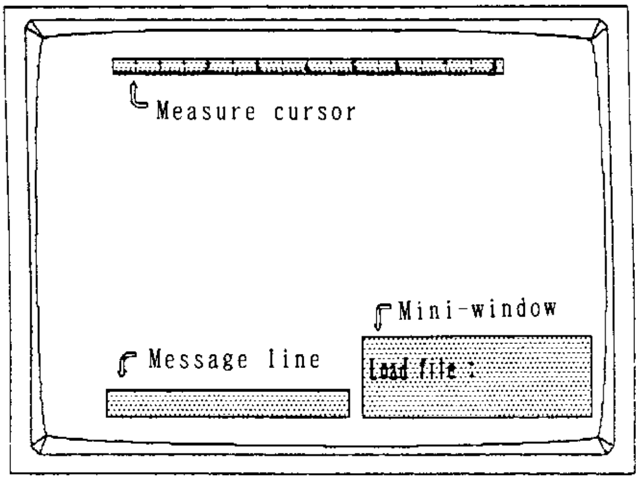
When connecting the terminal (VG-920) to the measuring insturment and turning on the power, the initial screen shown below appears and the ate editor starts.



```
                 t
                 t
 aaaa          ttttt     eeee
     a           t      e    e
 aaaaa           t      eeeeee
a     a          t      e
a    aa          t   t  e    e
 aaaa a           tt     eeee

       AdvanTest Editor
       VERSION 1.1
       Copyright 1988 ADVANTEST CORPORATION
```

Figure 3 - 1 Initial Screen of ate

When you press any key, the following screen appears. (The message line and the mini-window are not displayed on the screen.)



Load file :

Figure 3 - 2 Operation Screen of ate

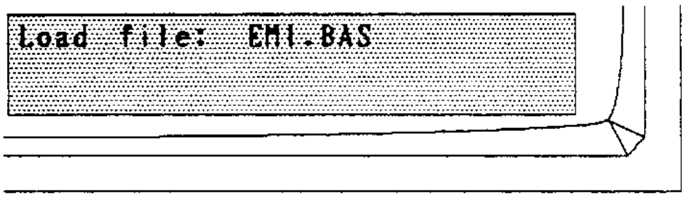
(2) Description of Screen Display

The following describes the screen display function. (See Figure 3-2 Operation Screen of ate.)

- Measure cursor
  This is the measure display to obtain the cursor position for editing.

- Message line
  This displays the command execution procedure, execution results, and error messages.

- Mini-window
  This is used when parameters are needed to execute commands. A buffer name, file name, or character string is displayed according to the command requiring parameters.
  Input parameters according to the displayed prompt. Be sure to press the Return key after inputting a parameter. The area enclosed by a square [] after the prompt is used for default parameters. The default value functions as a parameter when pressing the Return key without inputting any parameter.



Load file: EMI.BAS

Figure 3 - 3 Mini-window

4. FUNCTION LIST (When using the measuring instrument together with the external terminal)

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.

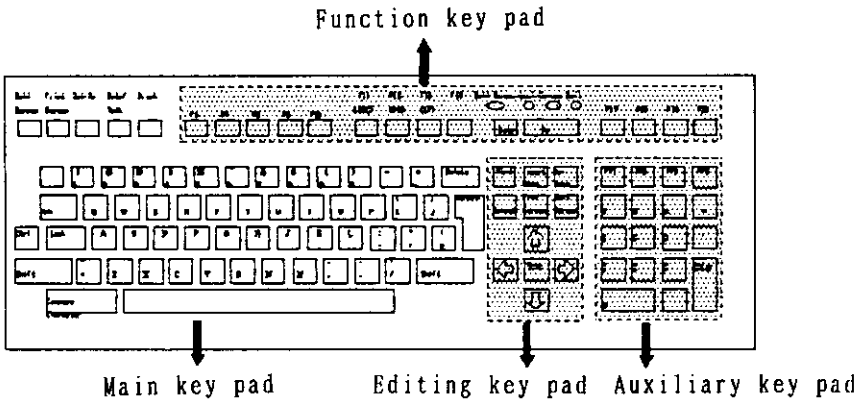


Figure 4 - 1 Key Pad Layout

4. 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)

4. 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.

4. 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).

4.4 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)

4.5 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 displayes data again. (Auxiliary key pad)

4.6 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)

4.7 Search

2 : Searches character strings forward. (Auxiliary key pad)
3 : Searches character strings backward. (Auxiliary key pad)

4.8 Replacement

1 : Replaces character strings. (Auxiliary key pad)

4.9 BASIC Mode

F13: Sets automatic line-number insertion. (Function key pad)
F14: Re-defines line numbers. (Function key pad)
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 progrem. (Function key pad)
F20: Continues the program interrupted by Ctrl-C. (Function key pad)

4.10 Help

Help : Displays the editor functions and key assignment list. (Function key pad)

4.11 Pop-up Menu

Do (Ctrl-A) : Displays the pop-up menu. (Function key pad)

4.12 Cancel

F11 (Ctrl-Z): Cancels the editor functions. (Function key pad)

4.13 Editor Restarting

F12 : Ends the current editor and initializes the buffer for restarting. (Function key pad)

MEMO

5. DESCRIPTION OF FUNCTIONS

(When using the measuring instrument together with the external terminal)

5.1 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.

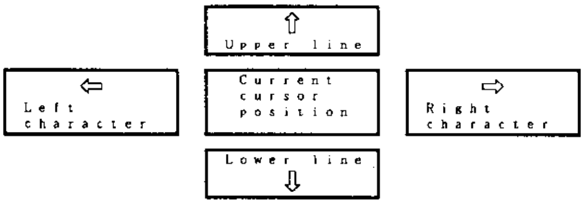


Figure 5 - 1 Cursor Key

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.

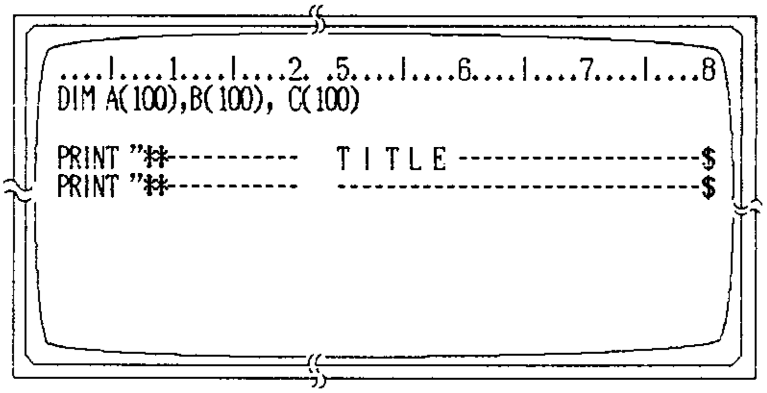
Find	Moves the cursor to the top of the file.
Insert-Here	Moves the cursor to the end of the file.
Remove	Moves the cursor to the top of the line.
Select	Moves the cursor to the end of the line.
Next-Screen	Advances the cursor to the next page.
Prev-Screen	Returns the cursor to the previous page.

Figure 5 - 2 Auxiliary Key Pads To Move Cursor

5. 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 "**----------  T I T L E -------------------$
PRINT "**----------  -----------------------------$
```

Figure 5 - 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 5 - 4 Main Key Pad for Insertion

5.3 Deletion

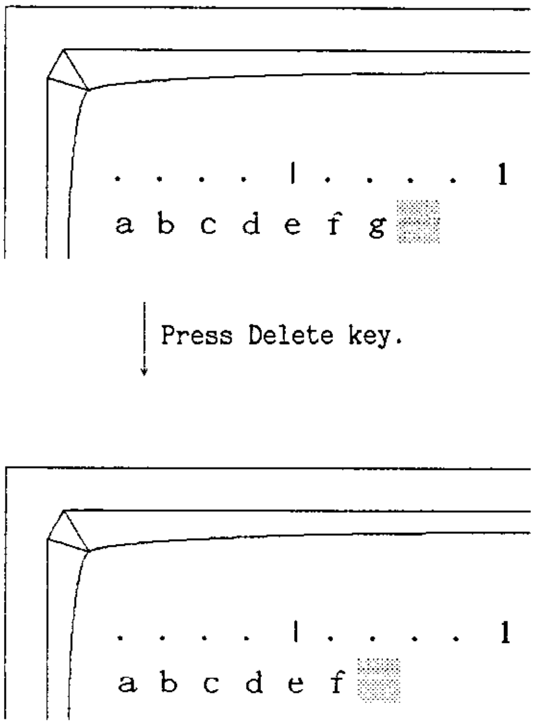
The deleting function includes one-character deletion, one-line deletion, and designated region deletion.

5.3.1 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).



Press Delete key.

Figure 5 - 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).

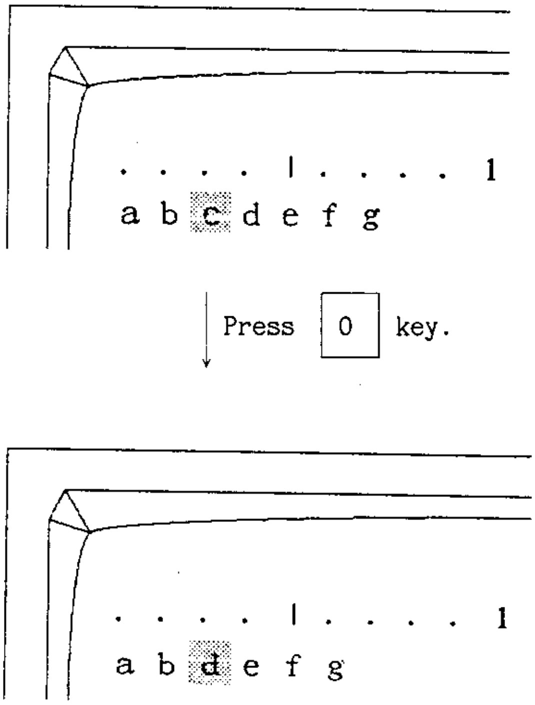


Figure 5 - 6 Deletion of the Character at the Cursor Position

5.3.2 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.

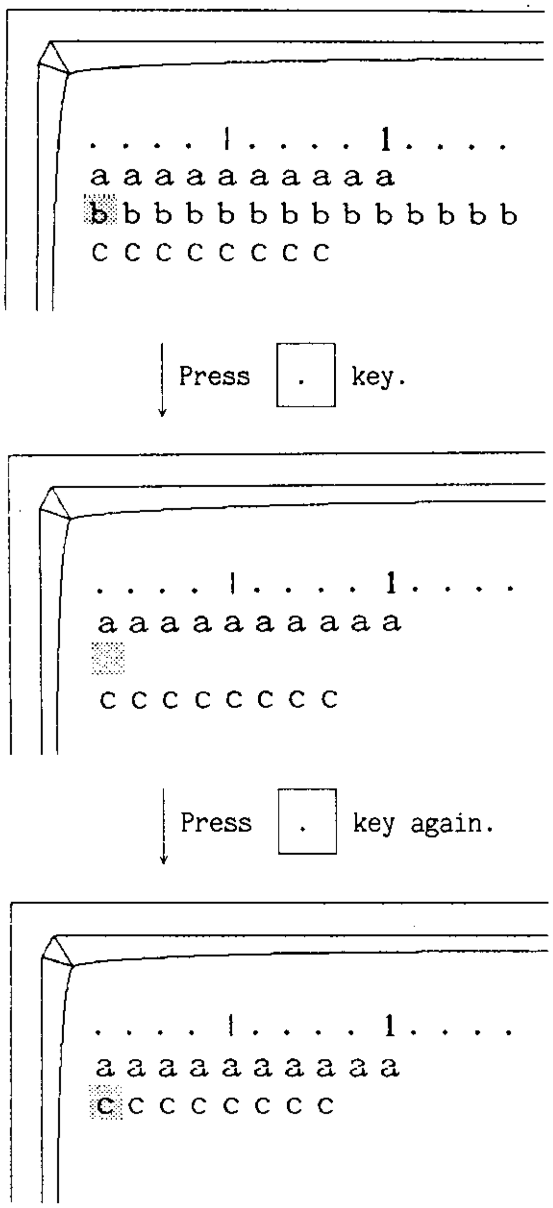


Figure 5 - 7 Deletion of Characters from Cursor Position Through Line End

5.3.3 Designated Region Deletion

The method to delete characters by designating a region is described in Section 5.4 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 5 - 8 Key Pad for Deletion

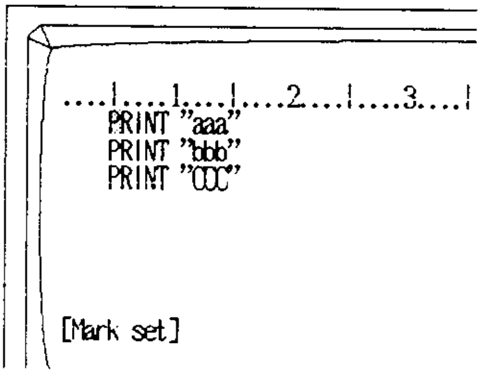
5.4 Copying, Moving, and Deleting (Region designation)

Store a text by designating a region for copying, moving, and deleting.

5.4.1 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 Mark set in the pop-up menu.

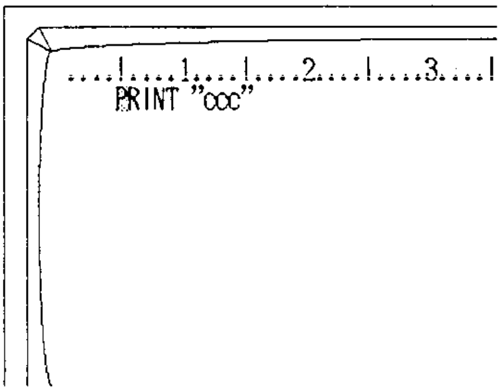


```
....|....1....|....2....|....3....|
    PRINT "aaa"
    PRINT "bbb"
    PRINT "ccc"




[Mark set]
```

Figure 5 - 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.

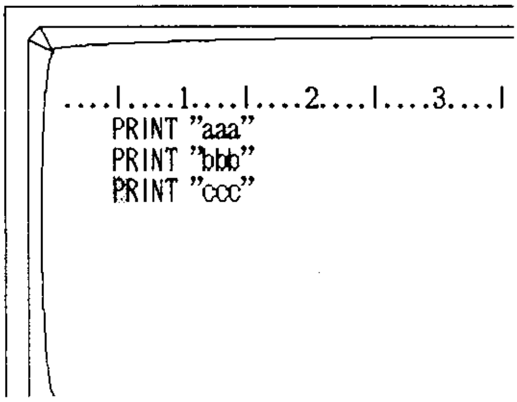


```
....|....1....|....2....|....3....|
    PRINT "ccc"
```

Figure 5 - 10 Storage of Text (Deletion)

5.4.2 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.

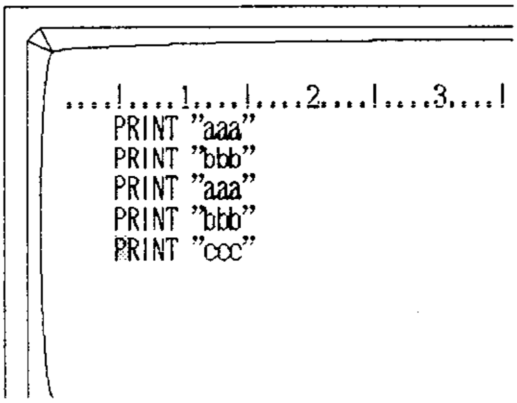


```
....|....1....|....2....|....3....|
    PRINT "aaa"
    PRINT "bbb"
    PRINT "ccc"
```

Figure 5 - 11 Recovery of Text

5.4.3 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, because 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 position 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 5 - 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 text in 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 5 - 13 Auxiliary Key Pad for Region Processing

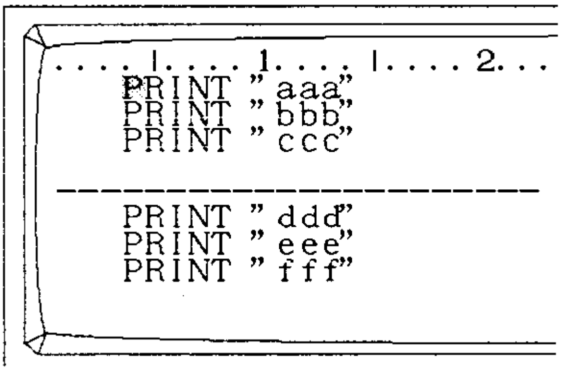
5. 5 Window

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.

5. 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.

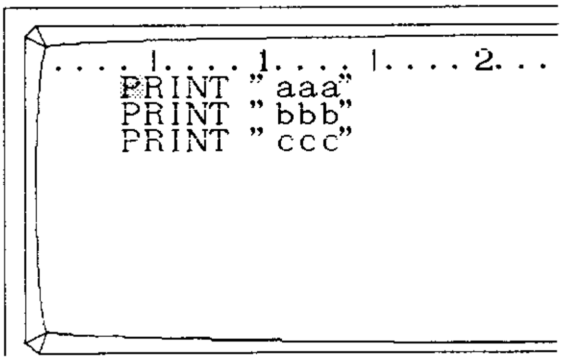


```
....|....1....|....2...
    PRINT "aaa"
    PRINT "bbb"
    PRINT "ccc"
-----------------------
    PRINT "ddd"
    PRINT "eee"
    PRINT "fff"
```

Figure 5 - 14 Window Split into Top and Bottom Ones

5. 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 5 - 15 Merged Window

5.5.3 Re-displaying of Screen

To display the text being edited again, press , (comma) key (Auxiliary Key Pad) or input Redisplayof the pop-up menu to display the text 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 5 - 16 Auxiliary Key Pad for Window

5.6 File

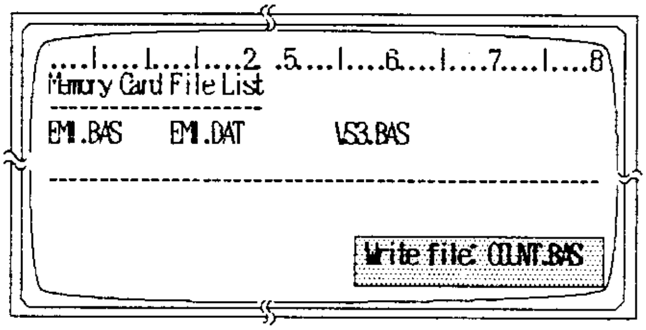
The ate editor saves texts in the memory card of the main unit and loads them from the memory card in files.
Therefore, be sure to save edited texts in the memory card.

Note: If the editor is turned off before saving the edited texts, the texts are lost.

The following describes how to save/load file.

5.6.1 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.

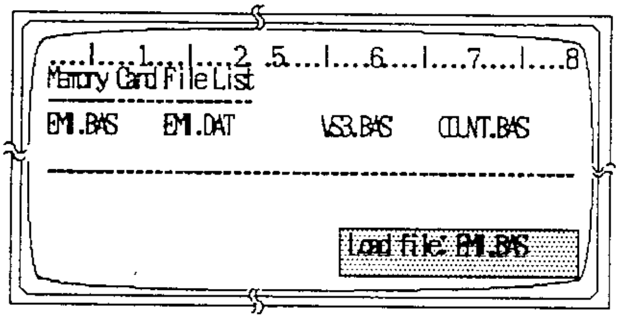


```
....|....L....|....2 .5....|....6....|....7....|....8
Memory Card File List
---------------------
EM1.BAS    EM1.DAT        VS3.BAS

----------------------------------------------------

                               Write file: COUNT.BAS
```

Figure 5 - 17 Saving of Text (Input a file name.)

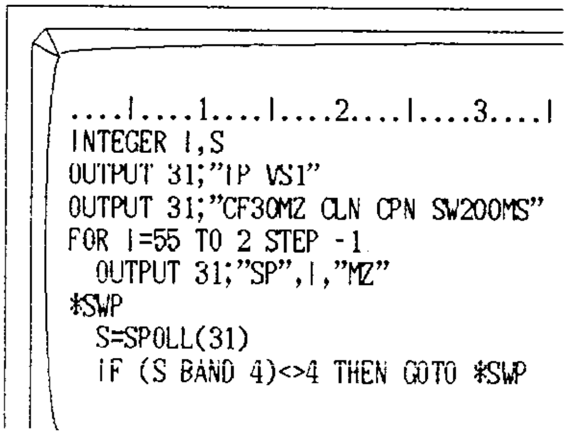
5.6.2 Loading of Files

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
EM1.BAS    EM1.DAT       VS3.BAS    COUNT.BAS

                                   Load file: EM1.BAS
```

Figure 5 - 18 Inputting of File Name



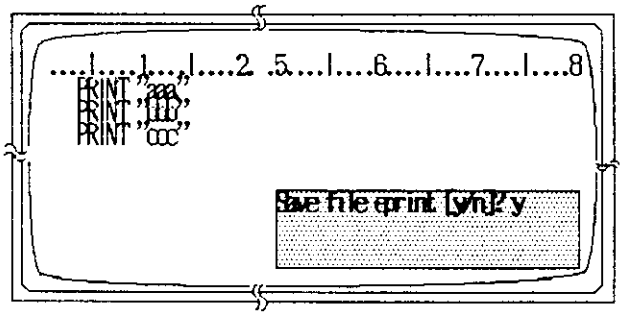
```
....|....1....|....2....|....3....|
INTEGER I,S
OUTPUT 31;"IP VS1"
OUTPUT 31;"CF30MZ CLN CPN SW200MS"
FOR I=55 TO 2 STEP -1
  OUTPUT 31;"SP",I,"MZ"
*SWP
  S=SPOLL(31)
  IF (S BAND 4)<>4 THEN GOTO *SWP
```

Figure 5 - 19 Loading of File

While a file is loaded from the memory card, the LED on the memory card entrance of the main unit lights. When loading is completed, the text is displayed on the screen and the number of read lines is displayed in the message line.
In this case, the automatic line number inserting (AUTO) function is set to the lines with a line number.
To create a new file, input the file name.
Unless the memory card is set, the buzzer sounds or an error message is displayed.

5.6.3 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 change" is displayed.



```
....|....1....|....2 .5....|....6....|....7....|....8
PRINT "aaa"
PRINT "bbb"
PRINT "ccc"

                              Save file eprint [y/n]? y
```

Figure 5 - 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 5 - 21 Auxiliary Key Pad for File

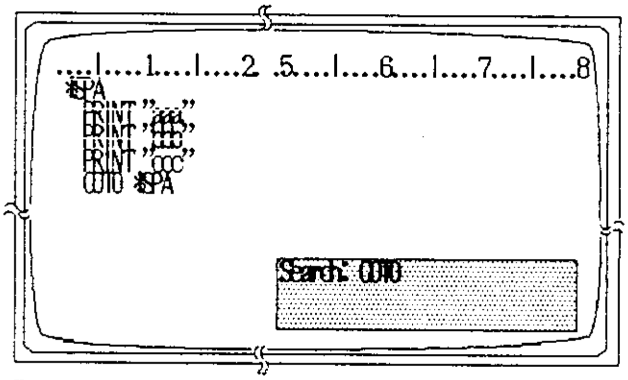
5.6.4 Deleting of Files

To delete a file, refer to "(58) PURGE" of Chapter 2 "GRAMMARS AND DESCRIPTION OF COMMANDS AND STATEMENTS" of 2nd Section "SYSTEM CONTROLLER".

5.7 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.

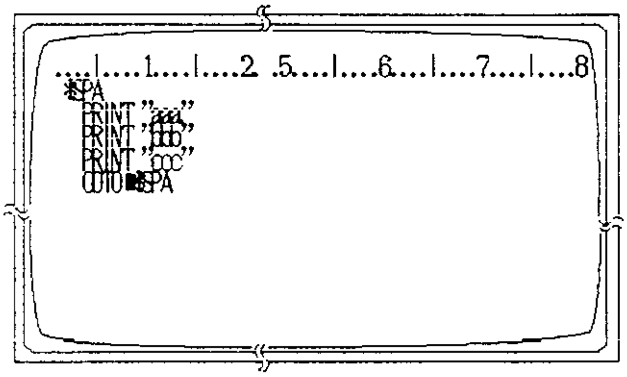
The following is the example to search a character string after the cursor position. Press 2 key (Auxiliary Key Pad) or input Forward search of the pop-up menu.
A mini-window is displayed on the screen and the character string to be searched is queried. Therefore, input the character string to be searched.



```
....|....1....|....2 .5....|....6....|....7....|....8
*SPA
  PRINT "aaa"
  PRINT "bbb"
  PRINT "ccc"
  GOTO *SPA

                          Search: GOTO
```

Figure 5 - 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 5 - 23 Execution Result

If the character string cannot be searched, the message "Not found" is displayed.

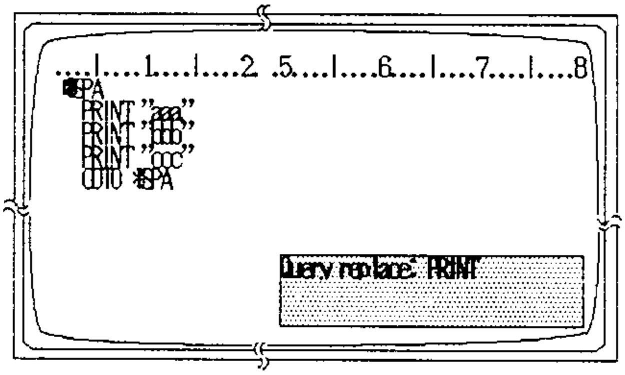
2	Searches a character string backward from the cursor position.
3	Searches a character string forward from the cursor position.

Figure 5 - 24 Auxiliary Key Pad for Search

5.8 Replacement

This is the function to replace any character string after the cursor position in a text.

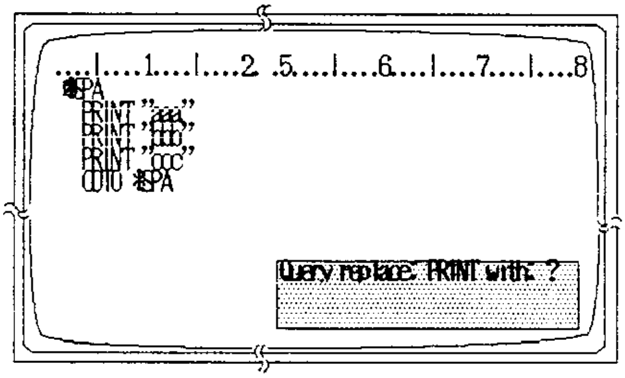
The following is the example to replace a character string. Press 1 key (Auxiliay Key Pad) or input Query replace of the pop-up menu, then a mini-window is displayed on the screen and the character string to be replaced is queried. Therefore, input the character string to be replaced.



```
....|....1....|....2 .5....|....6....|....7....|....8
*SPA
  PRINT "aaa"
  PRINT "bbb"
  PRINT "ccc"
  GOTO *SPA

                              Query replace: PRINT
```

Figure 5 - 25 Inputting of Character String to be Replaced

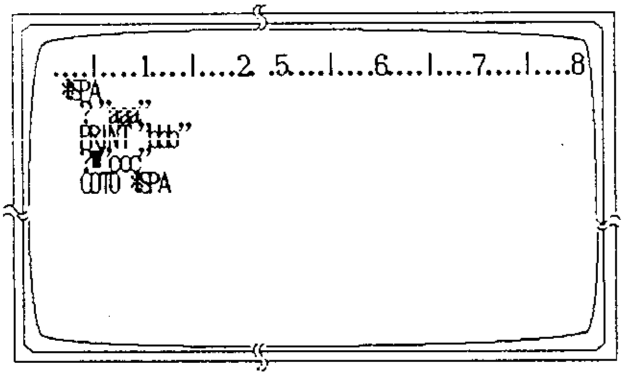
When the character string to be newly changed is queried, input an optional character string.



```
....|....1....|....2 .5....|....6....|....7....|....8
*SPA
  PRINT "aaa"
  PRINT "bbb"
  PRINT "ccc"
  GOTO *SPA

                              Query replace: PRINT with: ?
```

Figure 5 - 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 5 - 27 Execution Result

1	Replaces a character string.
space	Executes replacement.
delete	Executes no replacement.
.	Ends operation after replacement.
F11	Ends operation.

Figure 5 - 28 Auxiliary and Function Key Pad for Replacement

5.9 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.

5.9.1 Setting of Line Number

Though line numbers have been used so far to create a BASIC program, the ate editor uses labels in stead of line numbers to simplify the program.

```
A=1:B=0
*LOOP
        B=B+A^2
        IF B>10000000 THEN GOTO *ENDLOOP
        PRINT B
        GOTO *LOOP
*ENDLOOP
STOP
```

Figure 5 - 29 Program Using Labels

It is also possible to make editing while giving line numbers with the automatic line number inserting (AUTO) function.
When you press F13 key (Function Key Pad) or input Line no. of the pop-up menu, and the mini-window is displayed to designate and the starting number and interval. When pressing Return key for editing under the above condition, line numbers are automatically output.

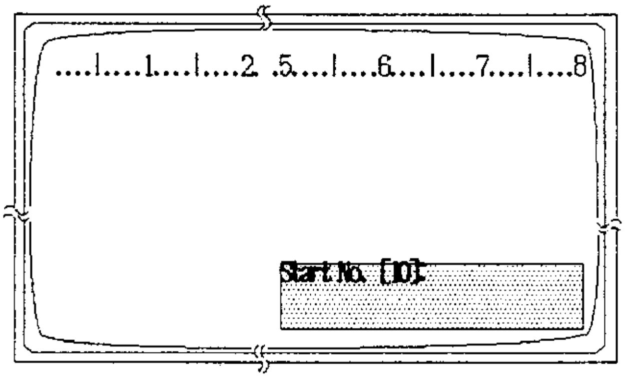
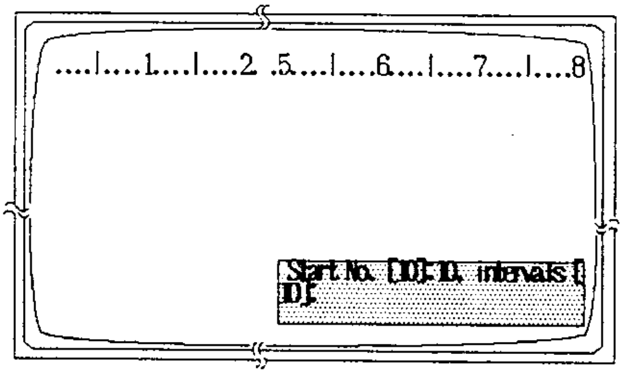
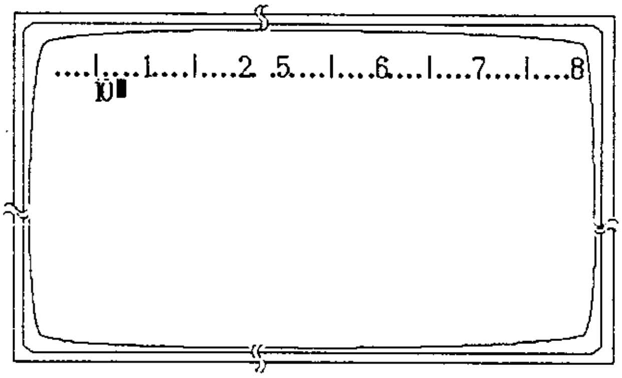


Figure 5 - 30 Designation of Starting Number



```
....|....1....|....2 .5....|....6....|....7....|....8
Start No. [10]:10, intervals [
10]:
```

Figure 5 - 31 Designation of Line Number Interval



```
....|....1....|....2 .5....|....6....|....7....|....8
 10▮
```

Figure 5 - 32 Automatic Line Number Inserting Function

Also it is possible to add line numbers with the function during editing.
If a line is inserted between lines, the number obtained by adding "1" to the upper line number is used for the line number of the inserted line.

Note: Once this function is executed, it is retained until the editor is initi alized (see section 5.13).

This function re-numbers lines according to a certain rule.
When you press F14 key (Function Key Pad) or input Renumbering of the pop-up menu, the starting number and the interval are designated similarly to automatic line-number inserting function.

5.9.2 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.

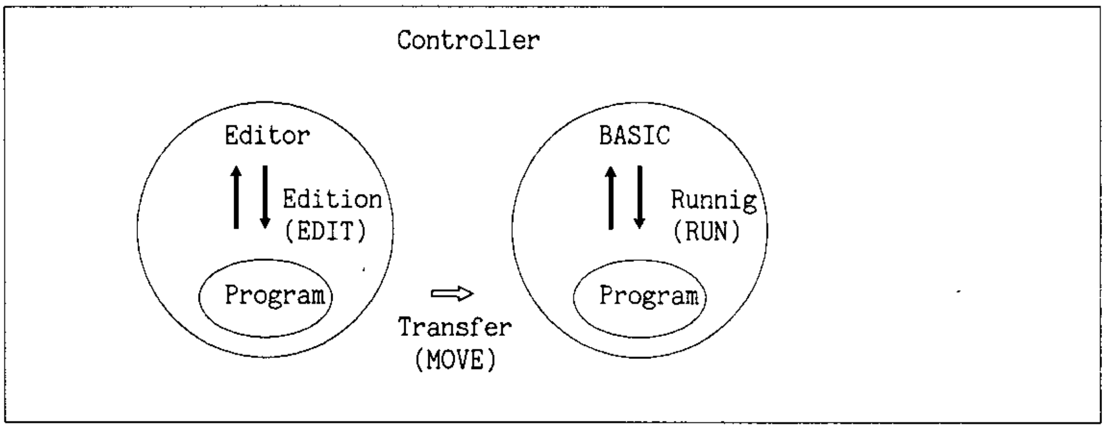


Figure 5 - 33 Relationship Between Editor and Interpreter

Therefore, move the program from the editor to the interpreter before running the program.
When you press F17 key (Function Key Pad) or input MOVE and RUN of the pop-up menu, the program is moved from the editor and run. In this case, the program last run is deleted. If an error occurs between program running, control is transferred to the editor and the cursor moves to the error line.

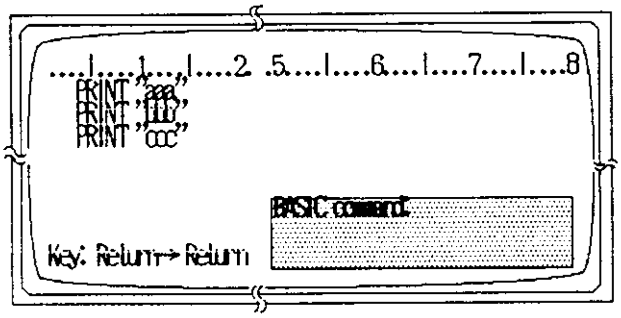
To run a moved program, press F19 key (Function Key Pad) or input RUN of the pop-up menu. If the program is executed with F19 key (Function Key Pad) or RUN of the pop-up menu before it is moved, the error message "Program not exist" is displayed.
When program running ends, the prompt "BASIC command:" is displayed in the mini-window and the debugging environment is ready. To return control to the editor, press Return key.

5.9.3 Stopping of BASIC

To stop running BASIC, press Ctrl-C key (Main Key Pad).

5.9.4 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 5 - 34 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.

For description of commands, see Part 2 "System controller".

5.9.5 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.

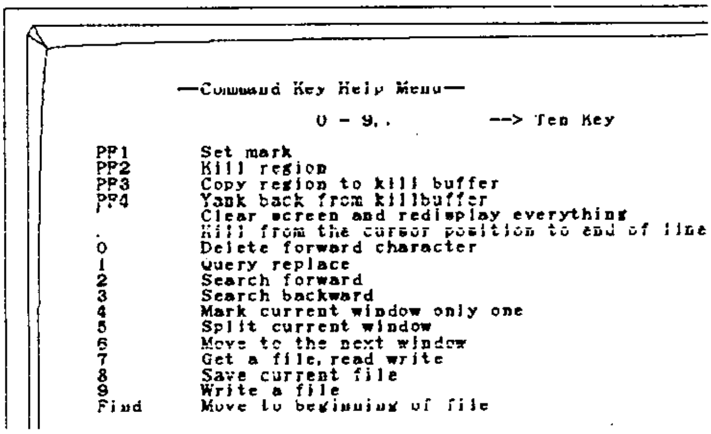
F13	Automatically inserts line numbers.
F14	Numbers lines.
F17	Moves the program before running it.
F18	Changes the mode to the BASIC mode.
F19	Runs the moved program.
F20	Continues the program interrupted by Ctrl-C key or the PAUSE command.

Figure 5 - 35 Function Key Pad for BASIC

5.10 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-Z key (Main Key Pad), and the HELP function stops and the original editing screen appears.



```
            —Command Key Help Menu—

                      0 - 9,.          --> Ten Key

PP1         Set mark
PP2         Kill region
PP3         Copy region to kill buffer
PP4         Yank back from killbuffer
,           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 5 - 36 Help Menu

HELP	Displays the help menu.
F11 (Ctrl-Z)	Cancels the help menu.

Figure 5 - 37 Function and Main Key Pad for Help

5.11 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) or Ctrl-A key (Main Key Pad), the pop-up menu is displayed at the top of the screen.

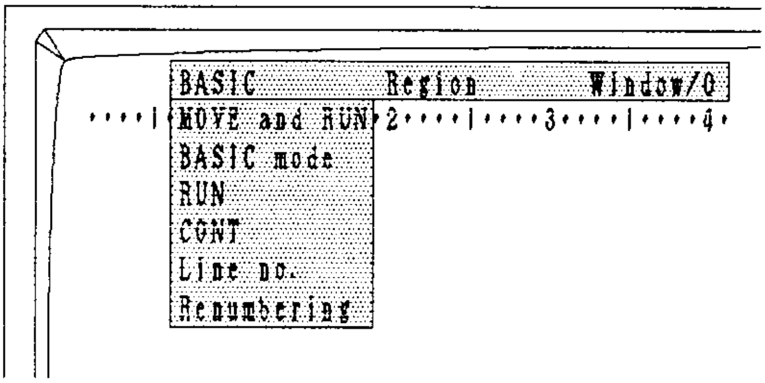


Figure 5 - 38 Pop-up Menu

BASIC	Region	Window/Other	File	Search/Replace
MOVE and 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		
Line No.		Help		
Renumbering		SCRATCH		

Figure 5 - 39 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 5 - 40 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 (Ctrl-A)	Displays the pop-up menu
F11 (Ctrl-Z)	Cancels the pop-up menu.

Figure 5 - 41 Function and Main Key Pad for Pop-up Menu

Menu	Description	Reference
MOVE and RUN	Moves and runs BASIC programs.	5.9.2
BASIC mode	Executes BASIC commands.	5.9.4
RUN	Runs the moved programs.	5.9.2
CONT	Continues the program interrputed by Ctrl-C key.	5.9.5
Line no.	Sets line numbers.	5.9.1
Renumbering	Resets line numbers.	5.9.1
Set mark	Deletes texts and sets the top position for copying.	5.4.1
Kill region	Deletes a text from the text mark through the cursor position.	5.4.1
Copy region	Stores a text from the text mark through the cursor position.	5.4.3
Yank	Copies (Recovers) the stored text.	5.4.2
Only	Merges windows into one.	5.5.2
Split	Splits a window into top and bottom ones.	5.5.1
Next	Moves the cursor to other window.	5.5.2
Redisplay	Re-displays a text.	5.5.3
Help	Displays the editor functions assigned to VG-920.	5.10
SCRATCH	Initializes the editor.	5.13
Load	Loads files from the memory card.	5.6.2
Save	Updates files.	5.6.3
Write	Saves files in the memory card.	5.6.1
Forward search	Searches a character string backward from the cursor position.	5.7
Backward search	Searches a character string forward from the cursor position.	5.7
Query replace	Replaces character strings.	5.8

Figure 5 - 42 Description of Pop-up Menu

5.12 Cancel of Editor Function

To cancel the editor function being executed, press F11 (Function Key Pad) or Ctrl-Z key (Main Key Pad).

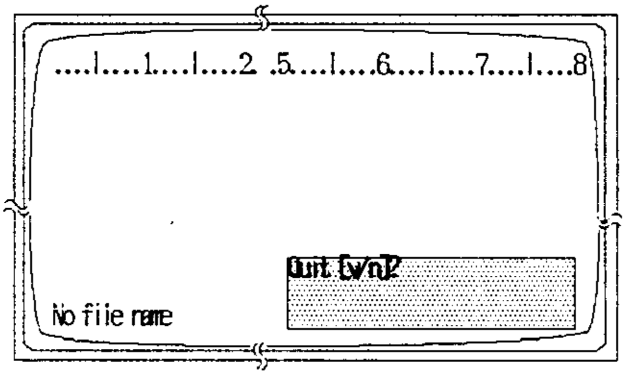
Note: The function cannot be canceled while a text is moved to the BASIC buffer or a file is saved or loaded.

F11 (Ctrl-Z)	Cancels editor functions.

Figure 5 - 43 Function (Main) Key Pad for Cancel

5.13 Initialization of Editor

Initialize the editor being operated. When you press F12 key (Function Key Pad) or input SCRATCH 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 initialize the editor. When you press y and Return keys, the editor is initialized. When you press n and Return keys, the editor is not initialized and original editor screen appears.



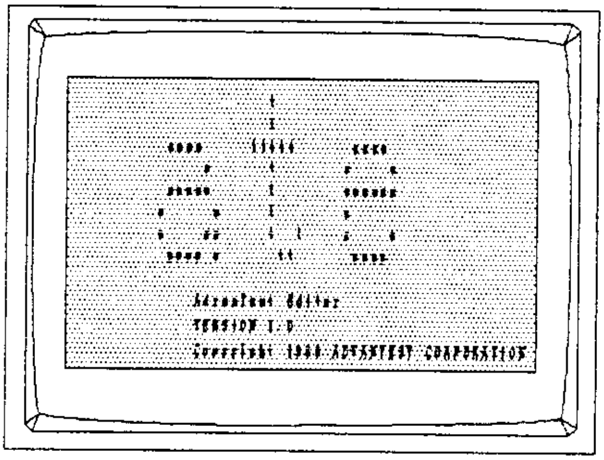
....|....1....|....2 .5....|....6....|....7....|....8

Quit [y/n]?

No file name

Figure 5 - 44 Mini-window To End Editor Operation

Press y and Return keys.



Advantest Editor
Version 1.0
Copyright 1988 ADVANTEST CORPORATION

Figure 5 - 45 Initialization of Editor

F12	Initializes the ate editor.

Figure 5 - 46 Function Key Pad for Initialization of Editor

MEMO

6. FEATURES (When using the measuring instrument alone)

The ate editor is a full screen editor having sophisticated editing functions.
The ate editor function includes cursor control, text insertion, deletion, copying, moving, replacement, search, window, file, and BASIC program running. In addition, the pop-up menu is prepared to smoothly execute these functions.
The following shows the difference between use of only the measuring instrument and use of it together with the external terminal.

	Use of VG-920 and measuring instrument	Use of only measuring instrument
Maximum number of columns	80	54
Maximum number of lines	23	17
Keyboard	Assignment of all functions	Assignment of some functions
Character input	Input with keyboard	Input with label function

Figure 6 - 1 Difference Between Use of only the Measuring Instrument and Use of It Together with external Terminal

6.1 Keyboard

All functions are assigned to the keyboard when the measuring instrument is used together with the external terminal while only some functions are assigned to the front panel when only the measuring instrument is used.

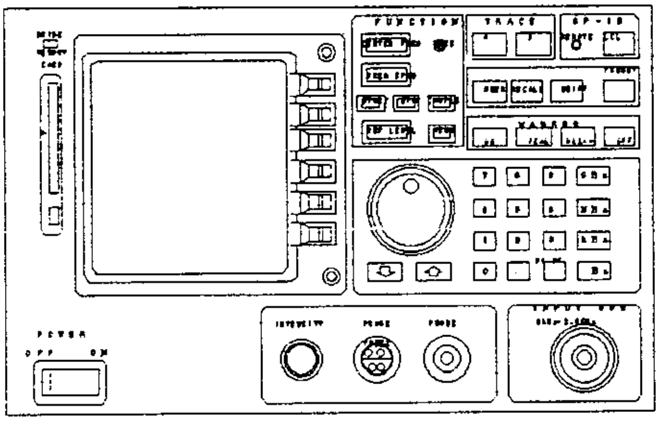


Figure 6 - 2 Front Panel of R3261/3361

6.2 Pop-up Menu

All functions except cursor movement can be executed with the pop-up menu similarly to the case when the external terminal is used. Unless there is a keyboard, editing is mainly made with the pop-up menu.
Select the function to be executed in the pop-up menu and press unit key (GHz, MHz, kHz, Hz), then the function is executed.

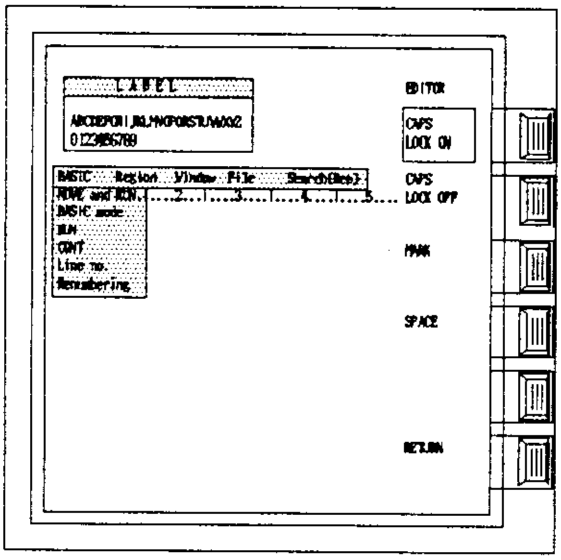


Figure 6 - 3 Pop-up Menu

6.3 BASIC Running Environment

The BASIC program edited with the ate editor can be run on the editor.
The BASIC commands RUN, CONT, and SCRATCH can be executed with the soft key or pop-up menu. Commands other than the above can be executed through the mini-window in the BASIC mode.

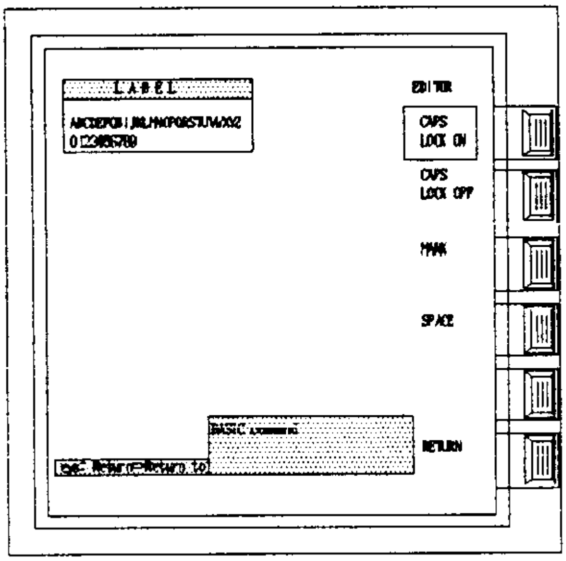


Figure 6 - 4 BASIC Mode

6.4 Definition of Label

The ate editor omits line numbers and defines labels. Then, understandable programs can be created when sub-routine call is used.

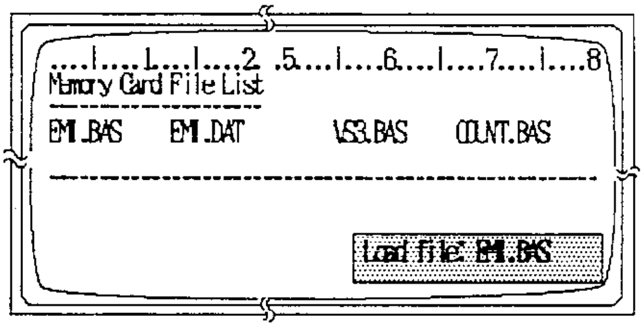
Note: When using labels, be sure to put an asterisk "*" at the top of the character string.

```
A=1:B=0
*LOOP
        B=B+A^2
        IF B>10000000 THEN GOTO *ENDLOOP
        PRINT B
        GOTO *LOOP
*ENDLOOP
STOP
```

Figure 6 - 5 Definition of Label

6.5 File

It is possible to save created programs in the memory card of the main unit in files. It is also possible to load the program from the memory card and edit it.



```
....|....1....|....2 .5...|....6....|....7....|....8
Memory Card File List
---------------------
EM1.BAS    EM1.DAT       VS3.BAS    COUNT.BAS
------------------------------------------------------

                              Load file: EM1.BAS
```

Figure 6 - 6 File Loading

6.6 Character Input

When the measuring instrument is used together with the external terminal, characters can be input with the keyboard. For only the measuring instrument, however, input characters using the label functions. In this case, press SHIFT key (Measuring Instrument Panel) before inputting characters according to the procedure of the label function.

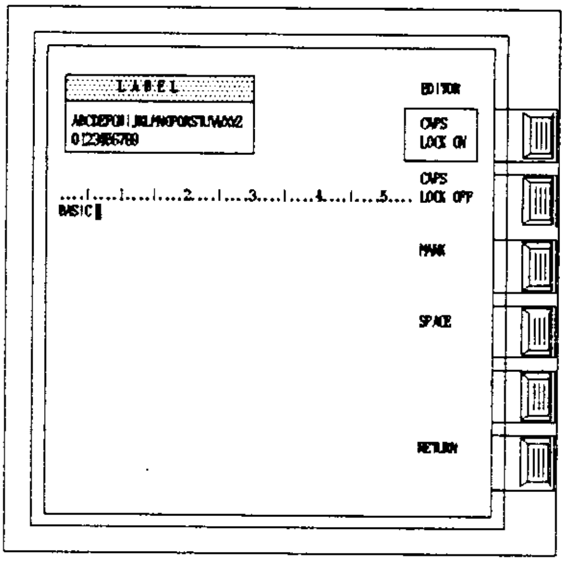


Figure 6 - 7 Character Input

6.7 Help Menu

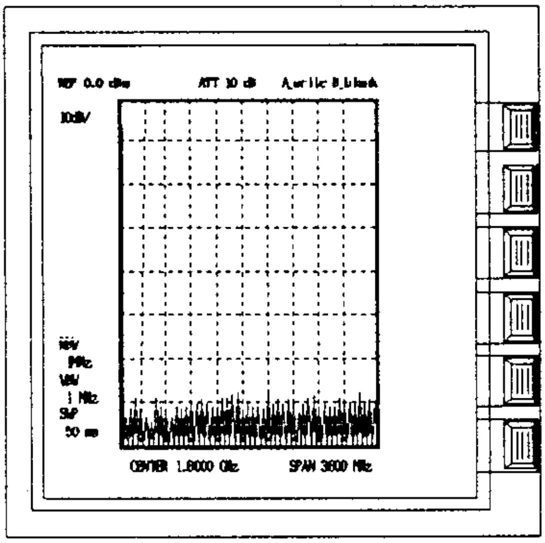
The help menu lists the assigned functions when the measuring instrument is used together with the external teminal (VG-920) but it does not support them when only the measuring instrument is used.

MEMO

7. STARTING (When using only the measuring instrument)

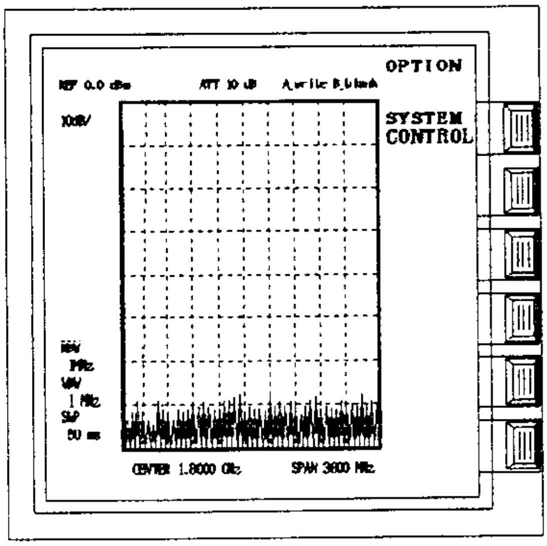
(1) Operation

Turn on the measuring instrument and perform the operations shown below, then the initial screen appears and the ate editor starts.

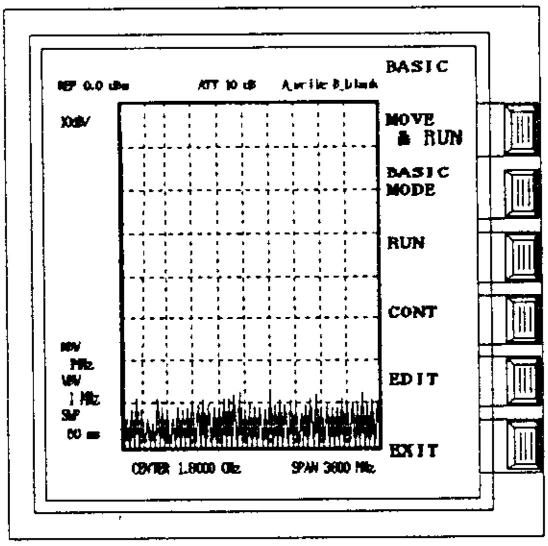


Perform the key operations on the front panel (Measuring Instrument Panel).

↓ Press SHIFT and 6 keys.



↓ Press SYSTEM CONTROL



Press EDIT key.

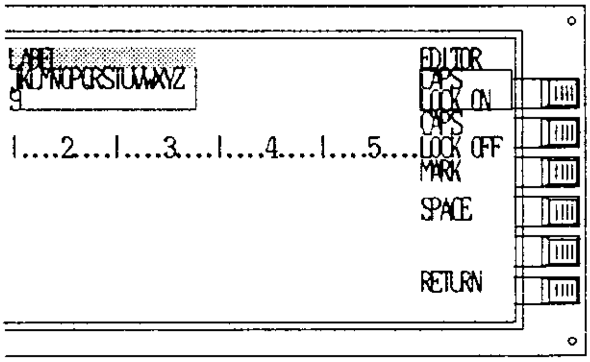


Figure 7 - 1 How To Start ate

(2) Description of Screen Display

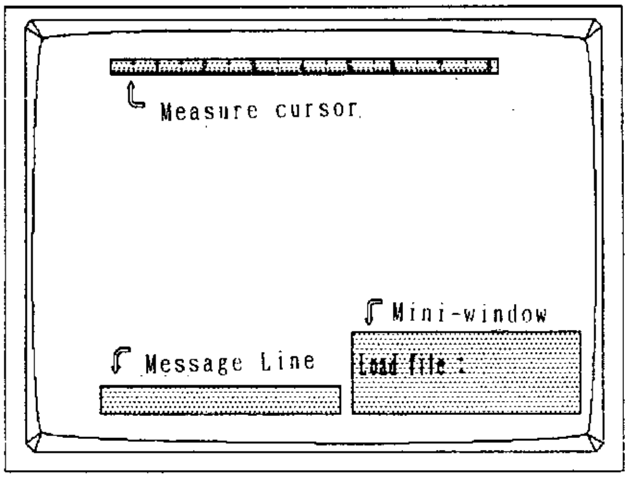


Figure 7 - 2 ate Operation Screen

- Measure cursor
  Displays the measure to obtain the cursor position for editing.

- Message line
  Displays the command execution procedure, execution result, and error messages.

- Mini-window
  This is used when parameters are needed to execute commands. A buffer, name, file name, or character string is displayed according to the command requiring parameters.
  Input parameters according to the displayed prompt. Be sure to press unit key (GHz , MHz , kHz , Hz) after a parameter. The area enclosed by a square [] after the prompt shows the default parameter. The default parameter functions as a parameter when unit key (GHz , MHz , kHz , Hz) is pressed without inputting any parameter.

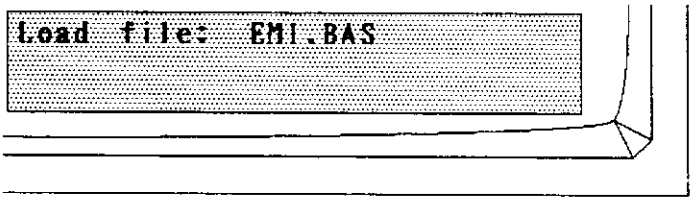

Load file: EMI.BAS

Figure 7 - 3 Mini-window

MEMO

8. FUNCTION LIST (When using only the measuring instrument)

Execute each function of the editor with the pop-up menu because of no keyboard. However, some functions are assigned to the front panel of the main unit as shown below.

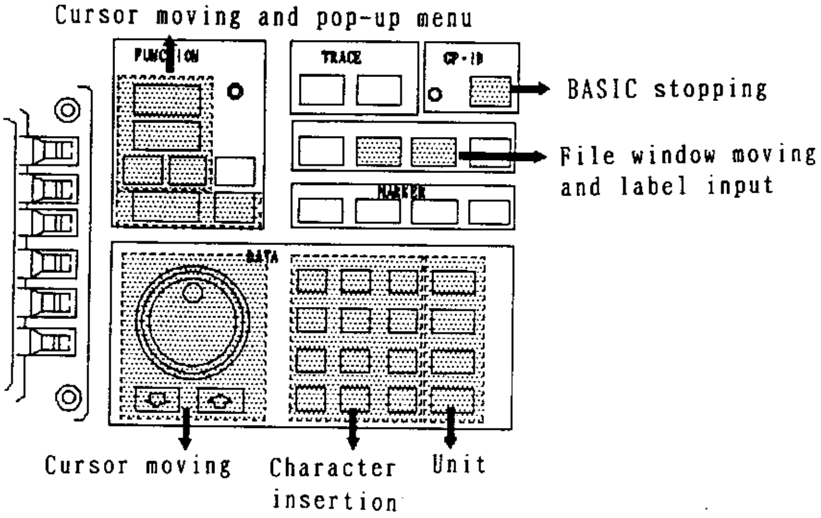


Figure 8 - 1 Measuring Instrument Panel Layout

8.1 Cursor Control

◎ (→ knob)	:	Moves the cursor to the next character.
◎ (← knob)	:	Moves the cursor to the preceding character.
↑	:	Moves the cursor to the preceding line.
↓	:	Moves the cursor to the next line.
START key	:	Moves the cursor to the top of the file.
STOP key	:	Moves the cursor to the end of the file.
SPAN key	:	Moves the cursor to the preceding screen.
CENTER key	:	Moves the cursor to the next screen.

8.2 Insertion

Label function	:	Inserts general characters.
Numeral key	:	Inserts numerals.
Unit key	:	Inserts line feed characters.
SPACE key	:	Inserts spaces.
Period key	:	Inserts periods.

8.3 Deletion

BK SP key	:	Deletes the character before the cursor.

8.4 Copying, Moving, and Deleting (Region designation)

Set Mark	: Sets a mark necessary for processing to the cursor position. (Pop-up)
Kill Region	: Deletes a text from the mark through the cursor position and saves remaining text in the buffer. (Pop-up)
Copy Region	: Saves a text from the mark through the cursor position in the buffer. (Pop-up)
Yank	: Copies the text saved in the buffer with the kill or copy key to the cursor position. (Pop-up)

8.5 Window

Only	: Merges windows into one. (Pop-up)
Split	: Divides a window into two. (Pop-up)
Next	: Moves the cursor to the next window. (Pop-up)
redisplay	: Clears the screen to display data. (Pop-up)
COUPLE key	: Clears the screen to display data.

8.6 File

Load	: Loads the designated file from the memory card. (Pop-up)
Save	: Saves files in the memory card. (Pop-up)
Write	: Saves a file with the designated name in the memory card. (Pop-up)
RECALL key	: Moves the cursor to the file name list window.

8.7 Search

Forward search	: Searches a character string forward. (Pop-up)
Backward search	: Searches a character string backward. (Pop-up)

8.8 Replacement

Query replace : Replaces character strings. (Pop-up)

8.9 BASIC Mode

Line no.   : Sets automatic line-number insertion. (Pop-up)
Renumbering   : Renumbers lines. (Pop-up)
MOVE and RUN   : Moves a program to the BASIC buffer before running it. (Pop-up)
BASIC mode   : Changes the mode to the BASIC mode. (Pop-up)
RUN   : Runs the moved program. (Pop-up)
CONT   : Contines the program interrputed with LOCAL key. (Pop-up)

MOVE & RUN key: Moves a program to the BASIC buffer before running it.
BASIC MODE key: Changes the mode to the BASIC mode.
RUN key   : Runs the moved program.
CONT key   : Continues the program interrupted with LOCAL key.

8.10 Pop-up Menu

REF LEVEL key : Displayes the pop-up menu.

8.11 Cancel

MENU key   : Cancels various functions of the editor.

8.12 Restarting of Edition

SCRATCH   : Ends the current editing and initializes the buffer to restart the editor. (Pop-up)

MEMO

9. DESCRIPTION OF FUNCTIONS
(When using only the measuring instrument)

9.1 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.

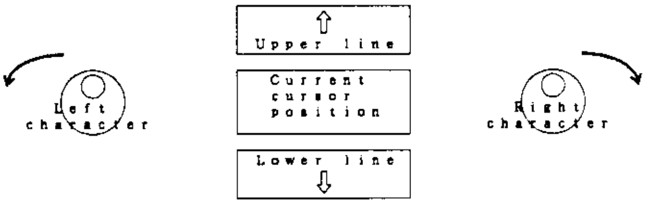


Figure 9 - 1 Cursor Key

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 some functions assigned to the front panel.

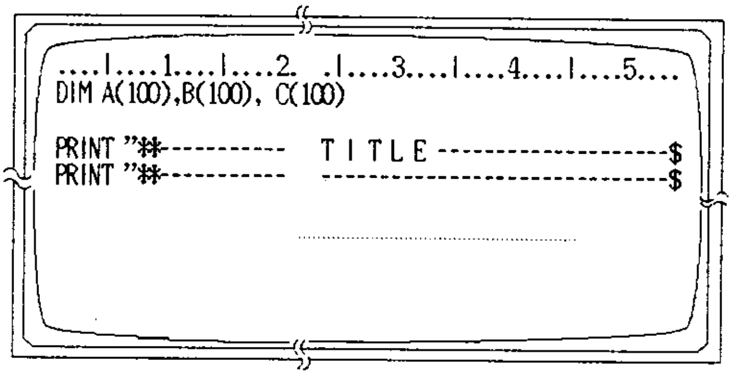
START key	Moves the cursor to the top of the file.
STOP key	Moves the cursor to the end of the file.
CENTER key	Advances the cursor to the next page.
SPAN key	Returns the cursor to the previous page.

Figure 9 - 2 Measuring Instrument Panel for Moving Cursor

9. 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. .|....3....|....4....|....5....
DIM A(100),B(100), C(100)

PRINT "**----------  TITLE ------------------$
PRINT "**----------  ----------------------------$
```

Figure 9 - 3 Sign Showing Continuation of Characters

Press the unit key at the end of a line to input the line feed character.

Label function	Inserts general characters.
Numeral key	Inserts numerals. (Measuring Instrument Panel)
Unit key	Inserts line feed characters. (Measuring Instrument Panel)
SPACE key	Inserts spaces. (Measuring Instrument Soft Menu Key)
Period key	Inserts periods. (Measuring Instrument Panel)

Figure 9 - 4 Key Pad for Insertion

CAUTION

Because the functions of the ate editor are normally assigned to the front panel, it is impossible to insert characters. To insert characters similarly to the keyboard of the terminal, press SHIFT key (Measuring Instrument Panel) before inserting characters with the label function keys.

9.3 Deletion

The deleting function includes one-character deletion, one-line deletion, and designated region deletion. The following describes one-character deletion. Other deleting methods are described later.

9.3.1 One-character Deletion

The one-character deletion represents to delete the character immediately before the cursor.

To delete the character immediately before the cursor, press BK SP key (Measuring Instrument Panel) on the front panel.

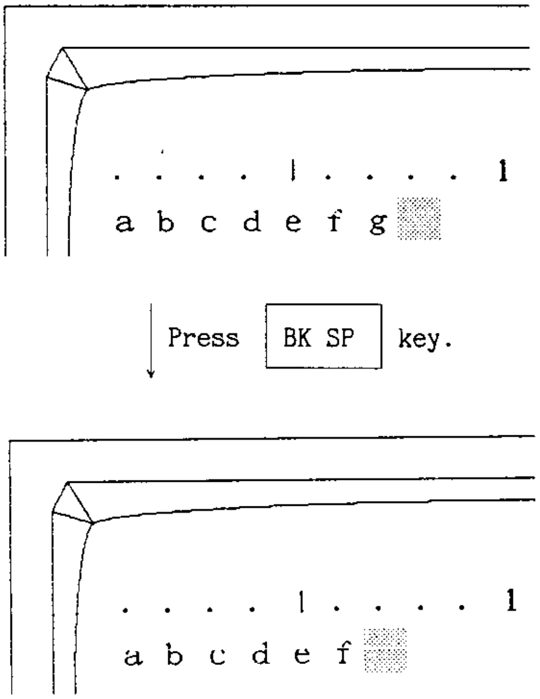


Figure 9 - 5 Deletion of the Character Immediately Before the Cursor

BK SP	Deletes the character immediately before the cursor.

Figure 9 - 6 Measuring Instrumentnel Panel for Deletion

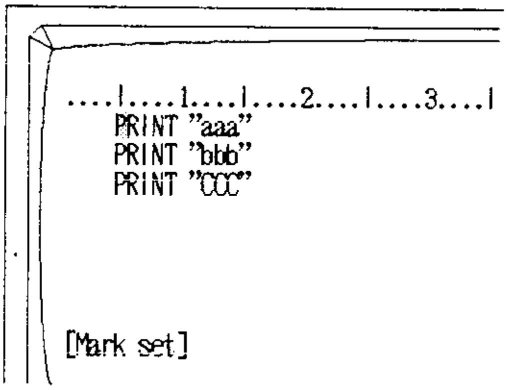
9.4 Copying, Moving, and Deleting (Region designation)

Store a text by designating a region for copying, moving, and deleting.
When you press REF LEVEL key on the measuring instrument panel, the pop-up menu is displayed at the top of the screen.

9.4.1 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 Mark set in the pop-up menu.

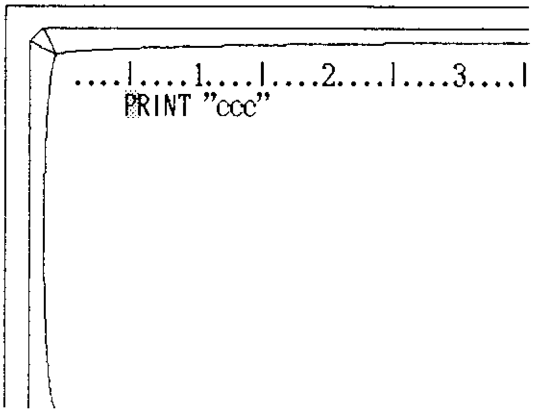


```
....|....1....|....2....|....3....|
    PRINT "aaa"
    PRINT "bbb"
    PRINT "ccc"



[Mark set]
```

Figure 9 - 7 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 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.

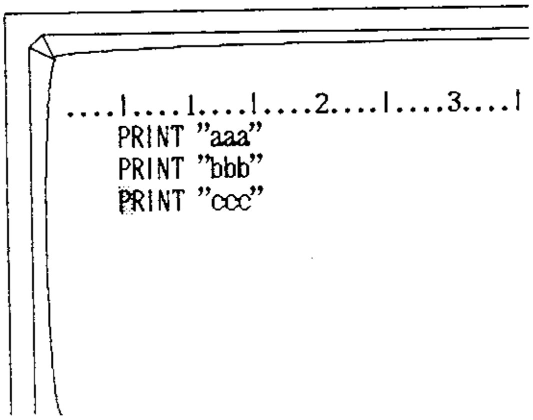


```
....|....1....|....2....|....3....|
    PRINT "ccc"
```

Figure 9 - 8 Storage of Text (Deletion)

9.4.2 Recovering and Moving of Text

To recover the deleted text, input Yank of the pop-up menu. To move the text, move the cursor to the destination and input Yank of the pop-up menu.
As mentioned above, Yank of the pop-up menu is the function to insert the saved text into the current cursor position.

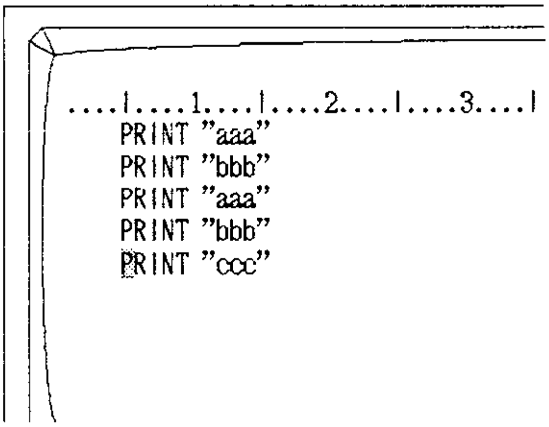


```
....|....1....|....2....|....3....|
    PRINT "aaa"
    PRINT "bbb"
    PRINT "ccc"
```

Figure 9 - 9 Recovery of Text

9.4.3 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 input Copy region of the pop-up menu.
In this case, because 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 position and input Yank of the pop-up menu.



```
....|....1....|....2....|....3....|
     PRINT "aaa"
     PRINT "bbb"
     PRINT "aaa"
     PRINT "bbb"
     PRINT "ccc"
```

Figure 9 - 10 Copying of Text

The text recovered by Yank of the pop-up menu is used as the latest text in the internal buffer.

Set mark	Sets the mark necessary for designating a region to the cursor position.
Kill region	Stores the text deleted from the mark through the cursor position
Copy region	Stores the text from the mark through the cursor position.
Yank	Inserts the stored text into the cursor position.

Figure 9 - 11 Pop-up menu for Region Processing

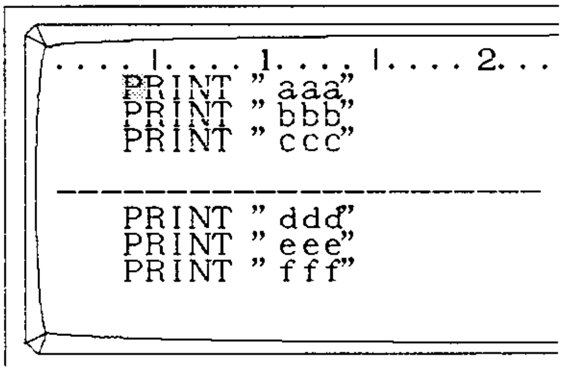
9.5 Window

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.

9.5.1 Splitting of Window

To split a window into two, input Split of the pop-up menu.

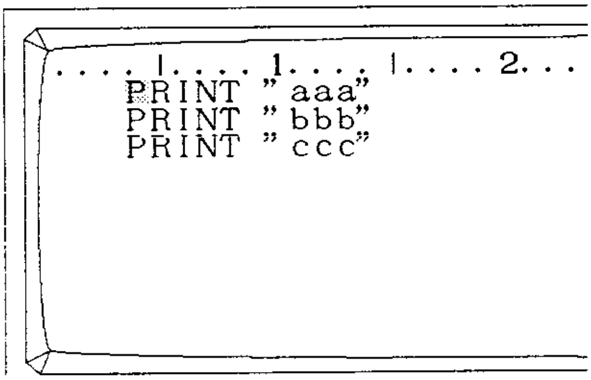


```
. . . . | . . . . 1. . . . | . . . . 2. . .
        PRINT " aaa"
        PRINT " bbb"
        PRINT " ccc"
------------------------------------------
        PRINT " ddd"
        PRINT " eee"
        PRINT " fff"
```

Figure 9 - 12 Window Split into Top and Bottom Ones

9.5.2 Initialization of Window

To merge the top and bottom window into one, input Next of the pop-up menu to move the cursor to the window to be left. Then input Only of the pop-up menu and the two windows are merged into one.



```
. . . . | . . . . 1. . . . | . . . . 2. . .
        PRINT " aaa"
        PRINT " bbb"
        PRINT " ccc"
```

Figure 9 - 13 Merged Window

9.5.3 Re-displaying of Screen

To display the text being edited again, press COUPLE key or input Redisplay of the pop-up menu to display the text again.

Only	Deletes other windows to display only the window in which the cursor is present.
Split	Splits a window into the top and bottom ones and brings the cursor to the top one.
Next	Moves the cursor to the next window.
Redisplay	Clears the screen to display the text again.

Figure 9 - 14 Pop-up menu for Window

COUPLE key	Clears the screen and displays the text again.

Figure 9 - 15 Measuring Instrument Panel for Window

9.6 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.

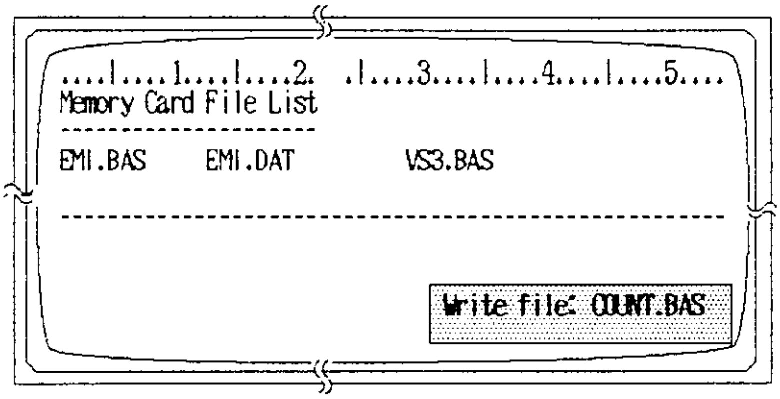
Note: If the editor is turned off before saving the edited texts, the texts are lost.

The following describes how to save/load file.
When you press REF LEVEL key on the measuring instrument panel, the pop-up menu is displayed at the top of the screen.

9.6.1 Saving of Files

To save an edited text in the memory card, input Write of the pop-up menu. In this case, give a name to the file to be saved by using the label function.
The file name is allowed up to 10 characters.

After saving the text, the number of written lines is displayed in the message line.

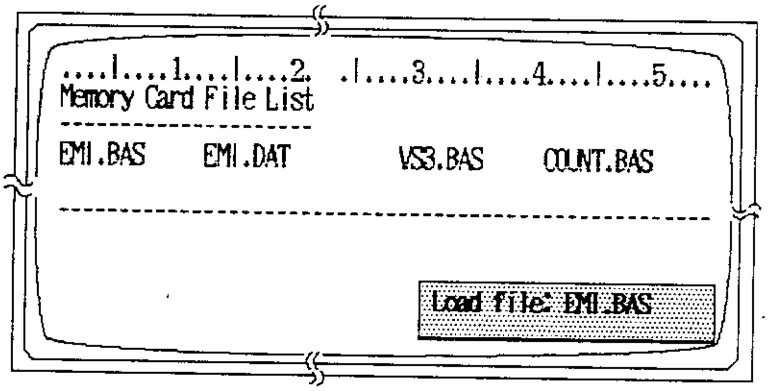


```
....|....1....|....2. .|....3....|....4....|....5....
Memory Card File List
---------------------
EMI.BAS    EMI.DAT        VS3.BAS

--------------------------------------------------

                          Write file: COUNT.BAS
```

Figure 9 - 16 Saving of Text (Input a file name.)

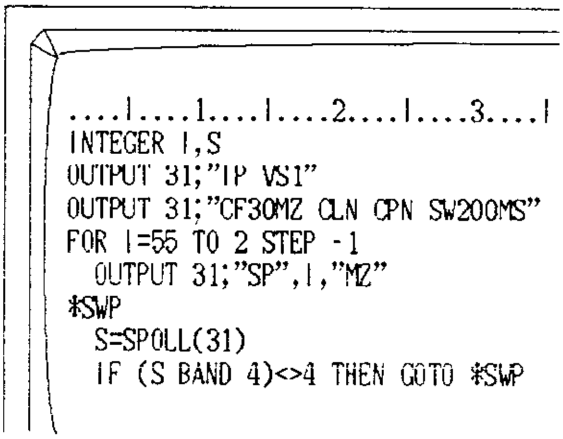
9.6.2 Loading of Files

To load the saved files, 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 RECALL key (Measuring Instrument Panel) before selecting the file name with the cursor or data knob and finally presssing $\boxed{\text{Hz}}$ key (Unit Key of Measuring Instrument Panel).



```
....|....1....|....2. .|....3....|....4....|....5....
Memory Card File List
---------------------
EMI.BAS     EMI.DAT        VS3.BAS     COUNT.BAS
----------------------------------------------------

                                  Load file: EMI.BAS
```

Figure 9 - 17 Inputting of File Name



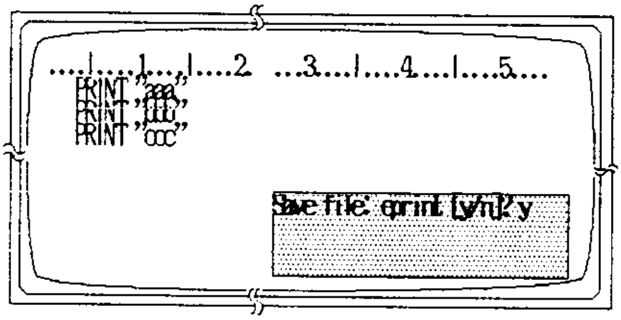
```
....|....1....|....2....|....3....|
INTEGER I,S
OUTPUT 31;"IP VS1"
OUTPUT 31;"CF30MZ CLN CPN SW200MS"
FOR I=55 TO 2 STEP -1
  OUTPUT 31;"SP",I,"MZ"
*SWP
  S=SPOLL(31)
  IF (S BAND 4)<>4 THEN GOTO *SWP
```

Figure 9 - 18 Loading of File

While a file is loaded from the memory card, the LED on the memory card entrance of the main unit lights. When loading is completed, the text is displayed on the screen and the number of read lines is displayed in the message line.
In this case, the automatic line number inserting (AUTO) function is set to the lines with a line number.
To create a new file, input the file name.
Unless the memory card is set, the buzzer sounds or an error message is displayed.

9.6.3 Updating of Files

To save the same file name, 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 change" is displayed.



```
....|....1....|....2 ...3....|....4....|....5...
  PRINT "aaa"
  PRINT "bbb"
  PRINT "ccc"

                         Save file: print [y/n]? y
```

Figure 9 - 19 Updating of Text (Same File Name)

When the y is entered using the label function and the Hz key (Unit Key of Measuring Instrument Panel) is pressed, 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 the n is entered using the label function and the Hz key (Unit Key of Measuring Instrument Panel) is pressed, text saving is not executed.

Load	Loads the designated file from the memory card.
Save	Saves a file with the same name in the memory card.
Write	Saves a file with the designated file name in the memory card.

Figure 9 - 20 Pop-up menu for File

RECALL key	Moves cursor to the file list window.

Figure 9 - 21 Measuring Instrument Panel Front Panel for File

REF LEVEL key	Displays the pop-up menu.
MENU key	Cancels the pop-up menu.

Figure 9 - 22 Measuring Instrument Panel for Pop-up Menu

◎ (knob)	Moves the cursor.
↑ key	
↓ key	

Figure 9 - 23 Measuring Instrument Panel for Cursor

GHz key	Completion of key entering. (ENTER)
MHz key	
kHz key	
Hz key	

Figure 9 - 24 Measuring Instrument Panel for Enter

9.6.4 Deleting of Files

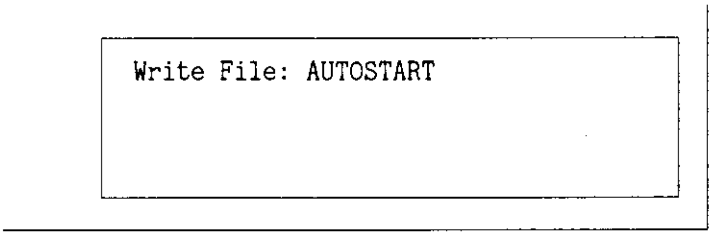
To delete a file, refer to "(47) PURGE" of Chapter 2 "GRAMMARS AND DESCRIPTION OF COMMANDS AND STATEMENTS" of 2nd Section "SYSTEM CONTROLLER".

9.6.5 Program automatic Start Function

When the power is set to on, the program can be automatically loaded (read) from the memory card and moved/run (activated).

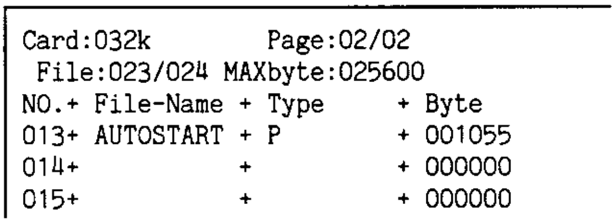
Register (save) the created program with file name "AUTOSTART" into the memory card.

Register the program with the [Write] function on the pop-up menu.



```
Write File: AUTOSTART
```

Confirm it with the CAT command.



```
Card:032k        Page:02/02
 File:023/024 MAXbyte:025600
NO.+ File-Name + Type     + Byte
013+ AUTOSTART + P        + 001055
014+           +          + 000000
015+           +          + 000000
```

From the next power-on, the system searches for file name "AUTOSTART" and loads and executes the file if found.

CAUTION

1. Be sure to register file name AUTOSTART in uppercase characters.
   Only one AUTOSTART file can exist in one memory card.
   The AUTOSTART file can be activated only once after power-on.

2. If a terminal is connected to the RS-232C terminal at the time of power-on, the program is directly loaded and executed. If no terminal is connected to the RS-232C terminal, however, the following operation is needed to active the AUTOSTART file.

Operation to activate AUTOSTART with no terminal connected:

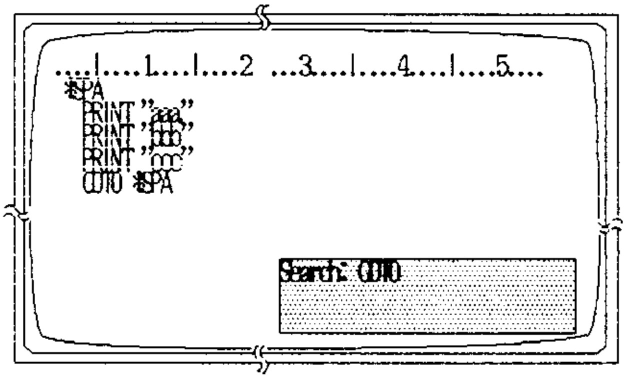
OPTION

SHIFT → 6 → SYSTEM CONTROL → EDIT

9.7 Search

This is the function to search a character string forward or backward in a text. When you press REF LEVEL key on the measuring instrument panel, the pop-up menu is displayed at the top of the screen.
To search the character string after the cursor position, input Forward search of the pop-up menu. To search it before the cursor position, 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. 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.

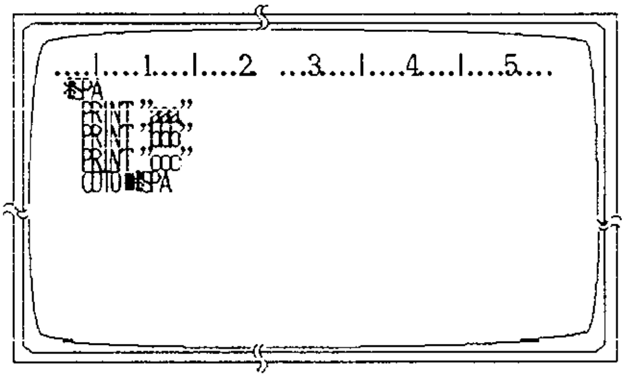


```
....|....1....|....2 ...3....|....4....|....5...
*SPA
  PRINT "aaa"
  PRINT "bbb"
  PRINT "ccc"
  GOTO *SPA
```

```
Search: GOTO
```

Figure 9 - 25 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 ...3....|....4....|....5...
*SPA
  PRINT "aaa"
  PRINT "bbb"
  PRINT "ccc"
  GOTO *SPA
```

Figure 9 - 26 Execution Result

If the character string cannot be searched, the message "Not found" is displayed.

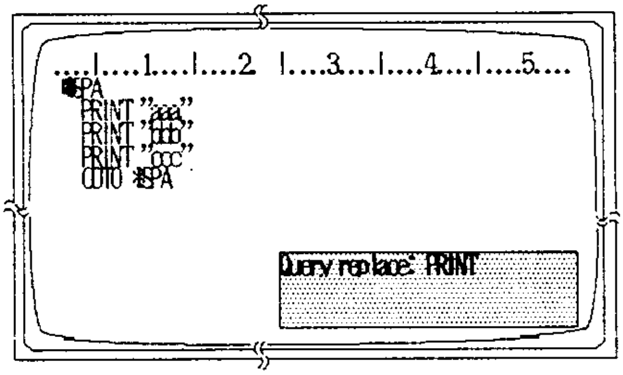
Forward search	Searches a character string backward from the cursor position.
Backward search	Searches a character string forward from the cursor position.

Figure 9 - 27 Pop-up menu for Search

9. 8 Replacement

This is the function to replace any character string after the cursor position in a text.

When you press REF LEVEL key on the measuring instrument panel, the pop-up menu is displayed at the top of the screen.
The following is the example to replace a character string. Input Query replace, then a mimi-window is displayed on the screen and the character string to be replaced is queried. Therefore, input the character string to be replaced.

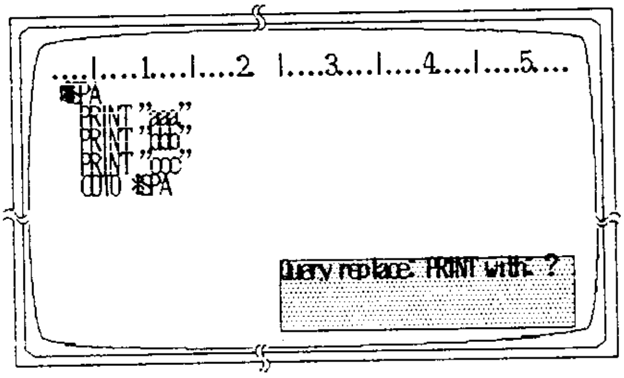


```
....|....1....|....2 |....3....|....4....|....5....
*SPA
  PRINT "aaa"
  PRINT "bbb"
  PRINT "ccc"
  GOTO *SPA
```

Query replace: PRINT

Figure 9 - 28 Inputting of Character String to be Replaced

When the character string to be newly changed is queried, input an optional character string.

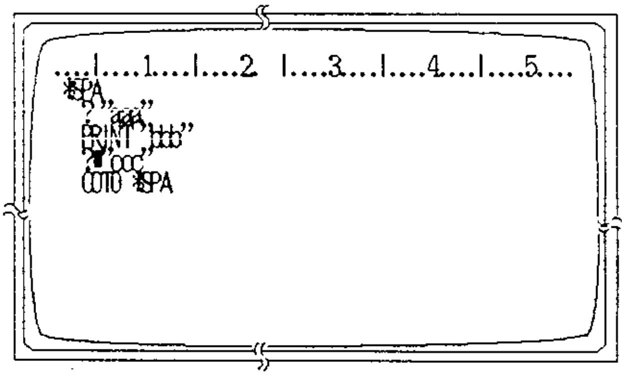


```
....|....1....|....2 |....3....|....4....|....5....
*SPA
  PRINT "aaa"
  PRINT "bbb"
  PRINT "ccc"
  GOTO *SPA
```

Query replace: PRINT with: ?

Figure 9 - 29 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, input soft key SPACE (Measuring Instrument Panel) to replace the character string or press BK SP key (Measuring Instrument Panel) unless replacing it.
To end the operation after replacement, press . key (Measuring Instrument Panel) on the front panel. To end the operation without replacement, press MENU key (Measuring Instrument Panel) onthe front panel.



```
....|....1....|....2 |....3....|....4....|....5....
*SPA
  PRINT"bbb"
  GOTO *SPA
```

Figure 9 - 30 Execution Result

Query replace	Replaces character strings.

Figure 9 - 31 Pop-up menu for Replacement

SPACE key	Executes replacement.
BK SP key	Executes no replacement.
Period key	Ends operation after replacement.
MENU key	Ends operation.

Figure 9 - 32 Measuring Instrument Panel for Replacement

9. 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.
When you press REF LEVEL key on the measuring instrument panel, the pop-up menu is displayed at the top of the screen.

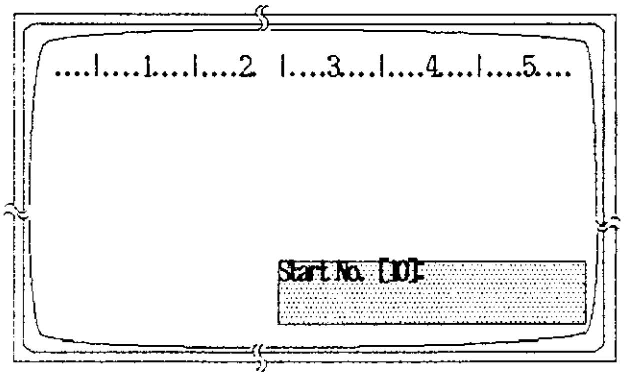
9. 9. 1 Setting of Line Number

Though line numbers have been used so far to create a BASIC program, the ate editor uses labels in stead of line numbers to simplify the program.

```
A=1:B=0
*LOOP
        B=B+A^2
        IF B>10000000 THEN GOTO *ENDLOOP
        PRINT B
        GOTO *LOOP
*ENDLOOP
STOP
```

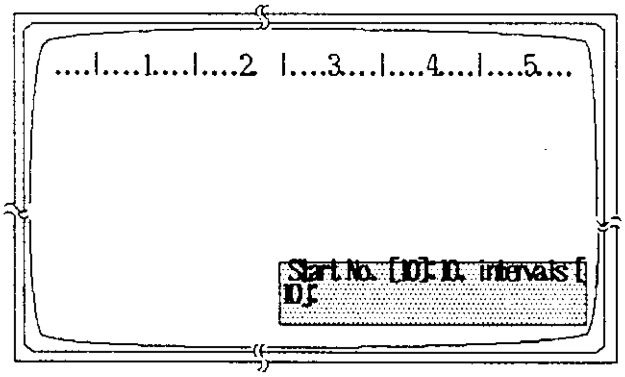
Figure 9 - 33 Program Using Labels

It is also possible to make editing while giving line numbers with the automatic line number inserting (AUTO) function.
When you input Line no. of the pop-up menu, and the mini-window is displayed to designate and the starting number and interval.
When pressing Hz key (Unit Key of Measuring Instrument Panel) for editing under the above condition, line numbers are automatically output.



Start No. [10]

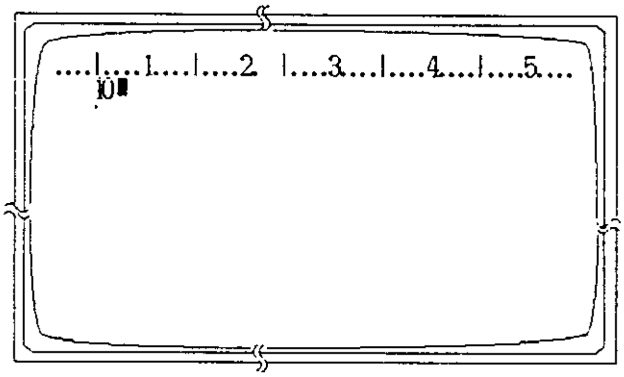
Figure 9 - 34 Designation of Starting Number



....|....1....|....2 |....3....|....4....|....5....

Start No. [10]: 10, intervals [
10]:

Figure 9 - 35 Designation of Line Number Interval



....|....1....|....2 |....3....|....4....|....5....
10

Figure 9 - 36 Automatic Line Number Inserting Function

Also it is possible to add line numbers with the function during editing.
If a line is inserted between lines, the number obtained by adding "1" to the upper line number is used for the line number of the inserted line.

Note: Once this function is executed, it is retained until the editor is initialized (see section 9.13).

This function re-numbers lines according to a certain rule.
When you input Renumbering of the pop-up menu, the starting number and the interval are designated similarly to automatic line-number inserting function.

9.9.2 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.

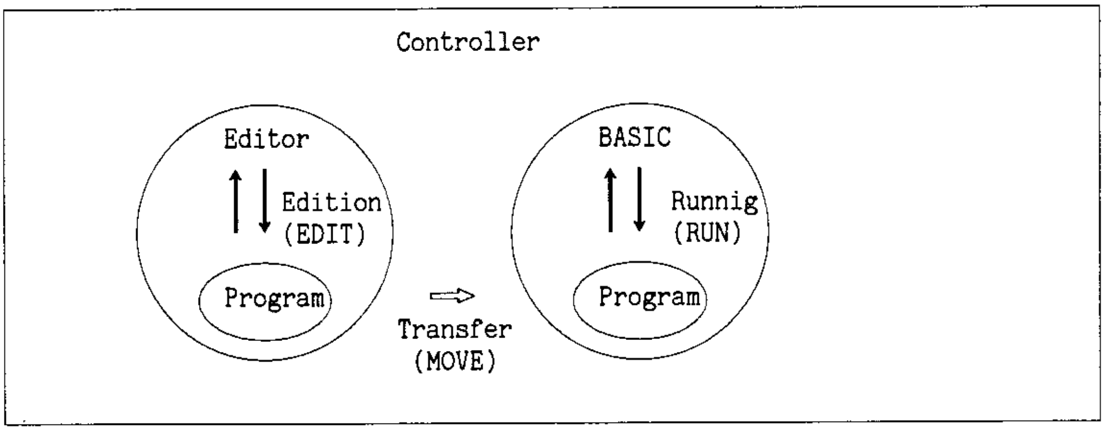


Figure 9 - 37 Relationship Between Editor and Interpreter

Therefore, move the program from the editor to the interpreter before running the program.
When you input soft key MOVE & RUN* or input MOVE and RUN of the pop-up menu, the program is moved from the editor and run. In this case, the program last run is deleted. If an error occurs during program running, control is transferred to the editor and the cursor moves to the error line.

To run a moved program, input soft key RUN or input RUN* of the pop-up menu. If the program is executed with the above procedure before it is moved, the error message "Program not exist" is displayed.

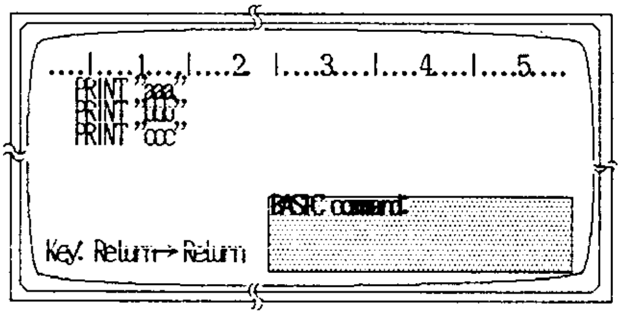
* Press the SHIFT and 6 keys to display MOVE & RUN and RUN in the right side on the screen.

9.9.3 Stopping of BASIC

To stop running BASIC, press LCL key (Measuring Instrument Panel).

9.9.4 BASIC Mode

The BASIC mode is the environment capable of executing commands for BASIC. Input soft key BASIC MODE or input BASIC mode of the pop-up menu, and the mini-window appears.



```
PRINT "aaa"
PRINT "bbb"
PRINT "ccc"
```

Figure 9 - 38 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.

For description of commands, see Part 2 "System controller".

9.9.5 Continuation of BASIC

To continuously run the program which is interruped by LCL key (Measuring Instrument Panel) or PAUSE command, input CONT (Measuring Instrument Soft Menu Key) 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.

Line no.	Automatically inserts line numbers.
Renumbering	Numbers lines.
MOVE and RUN	Moves the program before running it.
BASIC mode	Changes the mode to the BASIC mode.
RUN	Runs the moved program.
CONT	Continues the program interrupted by LOCAL key or the PAUSE command.

Figure 9 - 39 Pop-up Menu for BASIC

MOVE & RUN key	Moves the program before running it.
BASIC MODE key	Changes the mode to the BASIC mode.
RUN key	Runs the moved program.
CONT key	Continues the program interrupted by LOCAL key or the PAUSE command.

Figure 9 - 40 Measuring Instrument Panel for BASIC & Measuring Instrument Soft Menu Key

9.10 Help

This allows the operator to check the relationship between the editor functions and keyboard (VG-920) on the screen but dose not support the help menu for the front panel.

9.11 Pop-up Menu

All functions of the ate editor except the cursor moving function can be executed with the pop-up menu.
Because all editor functions except cursor key are displayed in the pop-up menu, select any function among them to executed it.

When you press REF LEVEL key on the measuring instrument panel, the pop-up menu is displayed at the top of the screen.

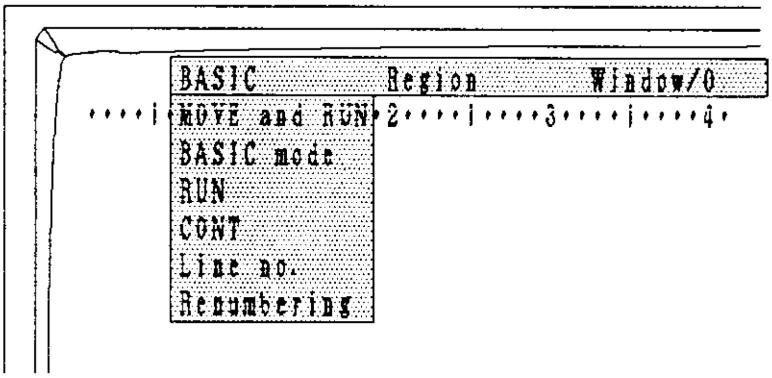


Figure 9 - 41 Pop-up Menu

BASIC	Region	Window/Other	File	Search/Replace
MOVE and RUN BASIC mode RUN CONT Line No. Renumbering	Set mark Kill region Copy region Yank	Only Split Next Redisplay Help SCRATCH	Load Save Write	Forward search Backward search Query replace

Figure 9 - 42 Pop-up Menu List

Keep pressing REF LEVEL key (Measuring Instrument Panel) and the cursor moves between items as a toggle. When the menu of the item concerned is displayed, select the function to be executed with the ↑ and ↓ keys (Measuring Instrument Panel).

REF LEVEL key	Displays the pop-up menu.
MENU key	Cancels the pop-up menu.

Figure 9 - 43 Measuring Instrument Panel for Pop-up Menu

Menu	Description	Reference
MOVE and RUN	Moves and runs BASIC programs.	9.9.2
BASIC mode	Executes BASIC commands.	9.9.4
RUN	Runs the moved programs.	9.9.2
CONT	Continues the program interrputed by LOCAL key.	9.9.5
Line no.	Sets line numbers.	9.9.1
Renumbering	Resets line numbers.	9.9.1
Set mark	Deletes texts and sets the top position for copying.	9.4.1
Kill region	Deletes a text from the text mark through the cursor position.	9.4.1
Copy region	Stores a text from the text mark through the cursor position.	9.4.3
Yank	Copies (Recovers) the stored text.	9.4.2
Only	Merges windows into one.	9.5.2
Split	Splits a window into top and bottom ones.	9.5.1
Next	Moves the cursor to other window.	9.5.2
Redisplay	Re-displays a text.	9.5.3
Help	Supports noting.	9.10
SCRATCH	Initializes the editor.	9.13
Load	Loads files from the memory card.	9.6.2
Save	Updates files.	9.6.3
Write	Saves files in the memory card.	9.6.1
Forward search	Searches a character string backward from the cursor position.	9.7
Backward search	Searches a character string forward from the cursor position.	9.7
Query replace	Replaces character strings.	9.8

Figure 9 - 44 Description of Pop-up Menu

9.12 Cancel of Editor Function

To cancel the editor function being executed, press MENU key on the measuring instrument panel.

Note: The function cannot be canceled while a text is moved to the BASIC buffer or a file is saved or loaded.

MENU key	Cancels editor functions.

Figure 9 - 45 Measuring Instrument Panel for Cancel

9.13 Initialization of Editor

Initialize the editor being operated. When you input SCRATCH 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 initialize the editor. When the y is entered using the label function and the Hz key (Unit Key of Measuring Inst Panel) is pressed, the editor is initialized. When the n is entered using the label function and the Hz key (Unit Key of Measuring Instrument Panel) is pressed, the editor is not initialized and original editor screen appears.

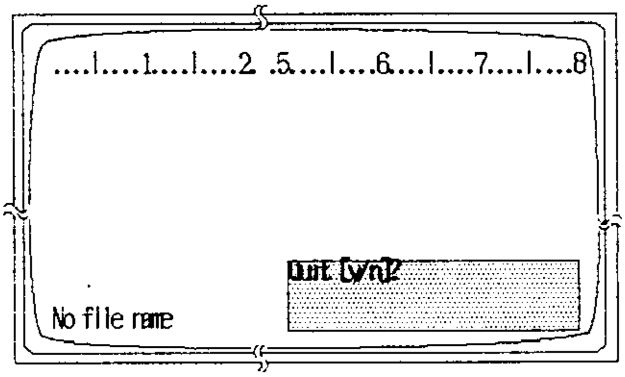


Figure 9 - 46 Mini-window To End Editor Operation

↓ Press y and unit keys.

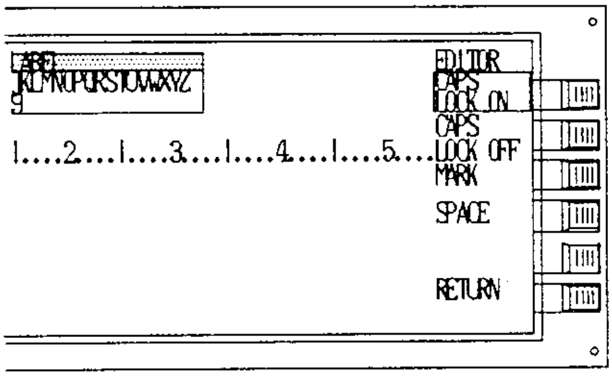


Figure 9 - 47 Initialization of Editor

SCRATCH	Initializes the ate editor.

Figure 9 - 48 Pop-up Menu for Initialization of Editor

MEMO

PART II SYSTEM CONTROLLER

1. BASIC GPIB CONTROLLER

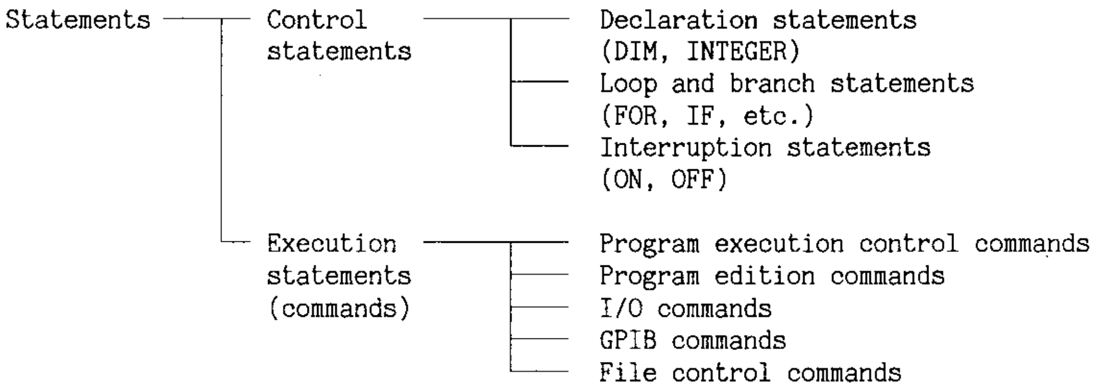
1.1 Outline

The BASIC language, option 15, 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.

1.2 BASIC Programming

1.2.1 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)
Loop and branch statements (FOR, IF, etc.)
Interruption statements (ON, OFF)
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.

1. 2. 2 Keywords

【List of keywords】

AND	AS	ASCII	BAND	BINARY	BNOT
BOR	BREAK	BUZZER	BXOR	CASE	CAT
CLEAR	CLOSE	CLS	CMD	CONT	CONTINUE
CONTROL	CSR	CURSOR	DATA	DELIMITER	DIM
DISABLE	ELSE	ENABLE	END	ENT	ENTER
ERROR	FOR	GLIST	GLISTN	GOSUB	GOTO
GPRINT	IF	INIT	INITIALIZE	INP	INPUT
INTEGER	INTERFACE	INTR	ISRQ	KEY	LISTEN
LISTN	LLIST	LLISTN	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	SELECT
SEND	SPRINTF	SRQ	STEP	STOP	TALK
TEXT	THEN	TO	TRIGGER	UNL	UNT
USE	USING	WAIT	XOR		

Non-executable keywords

APPEND	BASIC	CHKDSK	COPY	COPYFILES	COUNT
DEL	DSTAT	ENTERF	FORMAT	LABEL	LOAD
MERGE	NEWVERSION	REN	SAVE	SYSTEM	TIME
POKE	DUMP				

Note: Non-executable keywords have been registered as keywords but cannot be executed as commands.

1. 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. Whether to display short name or full name (LIST, GLIST, etc.) can be specifie with the CONTROL command (in the BASIC mode of the editor).

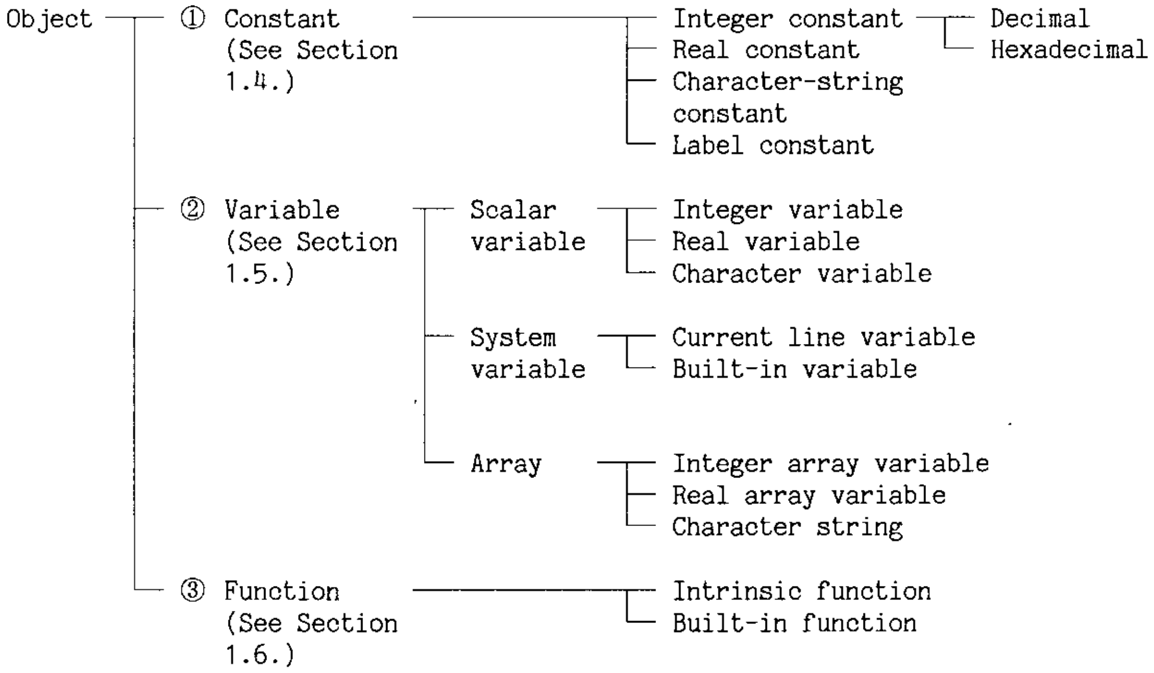
```
CONTROL 3;0  ·········  Full-name display
         ;1  ·········  Short-name display
```

【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

1.3 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.



Object
- ① Constant (See Section 1.4.)
  - Integer constant
    - Decimal
    - Hexadecimal
  - Real constant
  - Character-string constant
  - Label constant
- ② Variable (See Section 1.5.)
  - Scalar variable
    - Integer variable
    - Real variable
    - Character variable
  - System variable
    - Current line variable
    - Built-in variable
  - Array
    - Integer array variable
    - Real array variable
    - Character string
- ③ Function (See Section 1.6.)
  - Intrinsic function
  - Built-in function

1.4 Constant

1.4.1 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=0x10AB
            PRINT 0x0000FFFF
```

1.4.2 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 (IEEE) 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=1.23E6-9.87654
```

Note: Operation between constants

For example, the operation PRINT 3/2 will be evaluated as 1.0 because 3 and 2 are assumed integers.
To obtain a real value, the fractional part should be attached to either or both of the constants, as shown: PRINT 3.0/2.0.

1.4.3 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"
```

1.4.4 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 assigned 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 numbers or branch destinations described in the section 2.3 "Description on commands and statements".

1.5 Variables

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

Note: Any keyword cannot be used as a variable name.

【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)
```

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 code2	:	Integer variable
	INTEGER wk(7)	:	Integer array variable

1.5.1 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 assigned the value explicitly in the program.
The range of values to be assigned 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 do not include arrays.

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.

1.5.2 System Variables

- Current line variable @

A current line variable stores the line number where a program is executed. No value may be assigned.

Example: LIST @ : Displays the line being currently executed.

- 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 assigned another value. A variable may be returned to the initial value if it is assigned the initial value or the program is initialized with MOVE and 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

For details, refer to the ERRM$ in "1.6.1 Built-in Functions."

1.5.3 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 or 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 0.
The subscript specified in declaration will be the maximum number of the elements.

```
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 number of characters to be stored in a character-string variable can be declared with a DIM statement.

Example:

```
DIM A$ [100] :  Up to 100 characters can be assigned.
DIM B$ [50]  :  Up to 50 characters can be assigned.
```

Note: Two dimensional arrays cannot be used as character-string variable.

1.6 Functions

1.6.1 Intrinsic Functions

ERRM$ (Error number):

Returns the error message specified with the parameter.
If 0 is passed as parameter, the function returns the error message that has been displayed last.
The error number has the following structure inside of the program.

Error class * 256 + Error message number

However, even if a number including an error class is specified, only the error message number will be referred to inside of the program. Therefore, ERRN can be set as an error number, as follows:

```
Example:  PRINT ERRM$ (ERRN):  Displays the error message that has
                               been displayed last.
```

NUM (character-string expression):

Returns the ASCII code of the first character of the character-string expression.

```
Example:  NUM ("ABC")   →  65
          A$="XYZ"
          NUM (A$)      →  88
```

CHR$ (arithmetic expression):

Returns the character-string expression determined by the value of the arithmetic expression.

```
Example:  CHR$ (65)     →  "A"
          A=88
          CHR$ (A)      →  "X"
```

LEN (character-string expression):

Returns the length of the character-string expression.

```
Example:  LEN ("ADVANTEST")  →  9
          A$="CORP."
          LEN (A$)           →  5
```

POS (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.

```
Example:  A$="AN"
          POS ("ADVANTEST",A$)  →  4
```

ABS (arithmetic expression):
Returns the absolute value of the arithmetic expression.

Example: ABS (-1.2) → 1.2

ATN (arithmetic expression):
Returns the arc tangent of the arithmetic expression. (The arithmetic expression should be passed in radian.)

Example: ATN (PI) → 1.26262...

COS (arithmetic expression):
Returns the cosine of the arithmetic expression. (The arithmetic expression should be passed in radian.)

Example: COS (PI) → -1.0

FRE (arithmetic expression):
Returns the size (bytes) of the free area in the memory for BASIC.

Example: FRE (0) → 301458
(When the system is turned on, this function is executed as follows in the BASIC mode: PRINT FRE (0).)

LOG (arithmetic expression):
Returns the natural logarithm (base e logarithm) of the arithmetic expression. (The arithmetic expression should be passed in radian.)

Example: LOG (EXP) → 1.0

SIN (arithmetic expression):
Returns the sine of the arithmetic expression. (The arithmetic expression should be passed in radian.)

Example: SIN (PI) → 0.0

SPOLL (arithmetic expression):
Performs serial polling to the GPIB device and then returns the status byte. (The arithmetic expression should be in range between 0 and 31.)

Example: SPOLL (2) : Performs serial polling to address 2 of the external GPIB device connected.
SPOLL (31): Performs serial polling to the measurement section of the main device.

SQR (arithmetic expression):
Returns the square root of the arithmetic expression.

Example: SQR (2) → 1.41421356...

TAN (arithmetic expression):
Returns the tangent of the arithmetic expression. (The arithmetic expression should be passed in radian.)

Example: TAN (PI) → 1.0

1.6.2 Built-in Function

Item	Built-in Function
(1) Obtains the frequency.	FREQ (P) DFREQ (P1, P2) FMAX (P1, P2, M) FMIN (P1, P2, M) BND (P, X, M) BNDL (P, X, M) BNDH (P, X, M) CBND (F, X, M) CBNDL (F, X, M) CBNDH (F, X, M) FRPLHN (N, M) FRPLLN (N, M)
(2) Obtains points.	POINT (F) DPOINT (F1, F2) PMAX (P1, P2, M) PMIN (P1, P2, M) PRPLHN (N, M) PRPLLN (N, M)
(3) Obtains the traced data.	LVPOINT (L) LVDPOINT (L1, L2)
(4) Obtains levels.	VALUE (P, M) DVALUE (P1, P2, M) CVALUE (F1, F2, M) DCVALUE (F1, F2, M) LEVEL (T) DLEVEL (T1, T2) MAX (P1, P2, M) MIN (P1, P2, M) RPL1 (P1, P2, Dx, Dy, M) VRPLHN (N, M) VRPLIN (N, M)
(5) Obtains the total power.	POWER (P1, P2, M)

Item	Built-in Function
(6) Obtains the number of ripples (maximum and minimum waves)	NRPLH (P1, P2, Dx, Dy, M) NRPLL (P1, P2, Dx, Dy, M)
(7) Checks the upper and lower limits.	LMTMD1 (Dd, S, Ds) LMTMD2 (P, S, Ds, M) LMTUL1 (Dd, Up, Lo) LMTUL2 (P, Up, Lo, M)
(8) Inputs and outputs the traced data.	RTRACE (P, M) WTRACE (T, P, M)
(9) Graphics functions	GADRS (P, X, Y) GFLRECT (W, X1, Y1, X2, Y2) GLINE (L, W, X1, Y1, X2, Y2) GMKR (M, W, X, Y) GPOINT (W, X, Y) GRECT (L, W, X1, Y1, X2, Y2) GSTR (W, X, Y, Character-string expression)

1.7 Operation Expression

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

Operators		
	(1) Assignment operator	(See subsection 1.7.1.)
	(2) Unary arithmetic operator	(See subsection 1.7.2.)
	(3) Binary arithmetic operator	(See subsection 1.7.3.)
	(4) Logical operator	(See subsection 1.7.4.)
	(5) Relational operator	(See subsection 1.7.5.)
	(6) Substring operator	(See subsection 1.7.6.)
	(7) Bitwise operator	(See subsection 1.7.7.)

1.7.1 Assignment Operator

The LET statement, which has been used in the conventional BASIC, is not used in this BASIC. Therefore, assignment statements can directly be written. 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$=>"ABCD"	Assign a character string right-justified.
=<	A$=<"ABCD"	Assign 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
```

1.7.2 Unary Arithmetic Operator

-	Minus sign
+	Plus sign
++	Prefix/postfix increment: Add 1 to a variable. • Prefix increment Example: A = 2 Assign 2 to A. B = ++A Add 1 to A before assigning A to B. • Postfix increment Example: A = 2 Assign 2 to A. B = A++ Add 1 to A after assigning A to B.
--	Prefix/postfix decrement: Subtract 1 from a variable. • Prefix decrement Example: A = 2 Assign 2 to A. B = --A Subtract 1 from A before assigning A to B. • Postfix decrement Example: A = 2 Assign 2 to A. B = A-- Subtract 1 from A after assigning 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              : Assign A to B after subtracting 1 from A.
          C = A++              : Assign A to C before adding 1 to A.
          PRINT "3", A
          PRINT "4", B
          PRINT "5", C
          B = A--              : Assign 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
```

1.7.3 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
5.0
100.0
2.0
1
81.0
ABCDEFG
```

1.7.4 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	X	Y	X AND Y
	(Logical AND)	0	0	0
		0	1	0
		1	0	0
		1	1	1
OR	Disjunction	X	Y	X OR Y
	(Logical OR)	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 the value of variable A is zero, a jump to *MI takes place.

IF X<100 OR 199<X THEN GOTO *LA

If the value of variable 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 the value of variable X is greater than or equal to 0 and less than or equal to 100, the value of variable X is printed.

IF A XOR B THEN PRINT A, B

If the value of variable A is true (1) but the value of variable B is false (0) or if the value of variable A is false (0) but the value of variable B is true (1), both values of variable A and B are printed.

1.7.5 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
```

1.7.6 Substring Operator

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

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

  The substring starts with the character that separates from the top by the value of arithmetic expression 1 and ends with the character that separates by the value of arithmetic expression 2.

  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)

  The substring starts with the character that separates from the top by the value of arithmetic expression 1 and its length is determined by the value of arithemtic expression 2.

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

  【Result】

  ```
  CDEF
  TEST
  ADVAN
  ```

1.7.7 Bitwise Operators

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) BNOT 0 → 65535 BNOT 65535 → 0
BAND	(Logical AND) 65535 BAND 255 → 255 255 BAND 1024 → 0
BOR	(Logical OR) 255 BOR 1024 → 1279 1 BOR 2 → 3
BXOR	(Exclusive OR) 255 BXOR 128 → 127 1 BXOR 3 → 2

Example:

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

1.8 Precedence of Operation

Various opeators 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

1.9 Character-string Operation

In BASIC, operations can take place with character strings.

1.9.1 Join Character Strings

Character strings may be joined with "&".

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

       【Result】

          ADVANTEST
          ADVANTEST co.
```

1.9.2 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
```

1.9.3 Type Conversion

Assignments 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
```

2. GRAMMARS AND DESCRIPTION OF COMMANDS AND STATEMENTS

2.1 Outline

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

2.2 Introduction

2.2.1 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 instruciton
【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:
Indicats 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.
	LIST	Displays a program list.
	LLIST	Displays a program list. (RS-232C)
	LISTN	Displays a program list.
	LLISTN	Displays a program list. (RS-232C)
	RUN	Executes a program.
	SCRATCH	Deletes a program in BASIC.
	STEP	Executes a line of 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 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 an SRQ interrupt of the main measurement section.
	ON SRQ	Defines the branch of an SRQ interrupt of GPIB.

Note: See section 2.3 for details of each command and statement.

Function	Command and Statement	Description
④Interrupt controls (cont'd)	ON ERROR OFF END OFF KEY OFF ISRQ OFF SRQ OFF ERROR	Defines the branch of an error occurrence interrupt. Releases the branch of an EOF interrupt. Releases the branch of a key interrupt. Releases the branch of an SRQ interrupt of the main measurement section. Releases the branch of an SRQ interrupt of GPIB. Releases the branch of an error occurrence interrupt.
⑤ Character-string manipulations	NUM CHR$ LEN POS SPRINTF	Converts the first character of a character string to the ASCII code. Converts a numeric character to an ASCII character. Obtains the length of a character string. Positions character string 1 in character string 2. Formats a string and assigns it to a character-string variable.
⑥Memory card controls	CAT CLOSE # ENTER # OPEN # OUTPUT # INITIALIZE (or INIT) PURGE RENAME	Displays the contents of a memory card. Close a file. Reads data from a file. Open a file. Writes data into a file. Initializes a memory card. Deletes the specified file. Rename a file.
⑦Screen controls	CURSOR(or CSR) CLS	Moves the cursor to the specified position. Clears the screen.
⑧Statements	BUZZER DIM FOR TO STEP NEXT BREAK CONTINUE GOSUB RETURN GOTO	Sounds the buzzer. Declares array variables. Sets an iteration. Exits from the current iteration. Returns to the beginning of the current iteration. Branches to a subroutine. Returns from the current subroutine. Branches to the specified statement.

Note: See section 2.3 for details of each command and statement.

Function	Command and Statement	Description
⑧ Statements (cont'd)	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.
	LPRINT	Outputs data on a printer (RS-232C).
	LPRINT USING (or USE	Outputs formatted data on a printer (RS-232C).
	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 scree.
	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 periond.
⑨ 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.
	GLISTN	Ouuputs the program list to a GPIB printer.
	GPRINT	Ouputs data to a GPIB printer.
	GPRINT USING (or USE)	Outputs formatted data to a GPIB printer.
	INTERFACE CLEAR	Transfers IFC.
	LOCAL	Places the specified device in the local status.
	LOCAL LOCKOUT	Places the specified device in the local lokced-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.

Note: See section 2.3 for details of each command and statement.

2.3 Description on Commands and Statements

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

(1) BUZZER
(2) CALL
(3) CAT
(4) CLEAR
(5) CLS
(6) CLOSE #
(7) CONT
(8) CONTROL
(9) CURSOR or CSR
(10) DELIMITER

(11) DIM
(12) DISABLE INTR
(13) ENABLE INTR
(14) ENTER #
(15) ENTER or ENT
(16) FOR··· TO ··· STEP BREAK/CONTINUE - NEXT
(17) GLIST
(18) GLISTN
(19) GOSUB and RETURN
(20) GOTO

(21) GPRINT
(22) GPRINT USING or GPRINT USE
(23) IF THEN ELSE/ELSE IF/END IF
(24) INITIALIZE or INIT
(25) INTEGER
(26) INTERFACE CLEAR
(27) INPUT or INP
(28) LIST
(29) LISTN
(30) LLIST

(31) LLISTN
(32) LOCAL
(33) LOCAL LOCKOUT
(34) LPRINT
(35) LPRINT USING or LPRINT USE

(36) OFF END
(37) OFF ERROR
(38) OFF KEY
(39) OFF SRQ/ISRQ
(40) ON END - GOTO/GOSUB

(41) ON ERROR - GOTO/GOSUB
(42) ON KEY - GOTO/GOSUB
(43) ON SRQ/ISRQ - GOTO/GOSUB
(44) OPEN #
(45) OUTPUT #
(46) OUTPUT or OUT
(47) PURGE
(48) PRINTF or PRF
(49) PRINT or ?
(50) PRINT USING or PRINT USE

(51) PRINTER
(52) READ DATA/RESTORE
(53) REM or !
(54) REMOTE
(55) RENAME
(56) REQUEST
(57) RUM
(58) PAUSE
(59) SCRATCH
(60) SELECT CASE/CASE ELSE/END SELECT

(61) SEND
(62) SPOLL (X)
(63) SPRINTF
(64) STEP
(65) STOP
(66) TRIGGER
(67) WAIT

(1) 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 (standard values)】

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 octavbe. 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.

(2) 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. Up to 10 characters can be obtained for a file name.

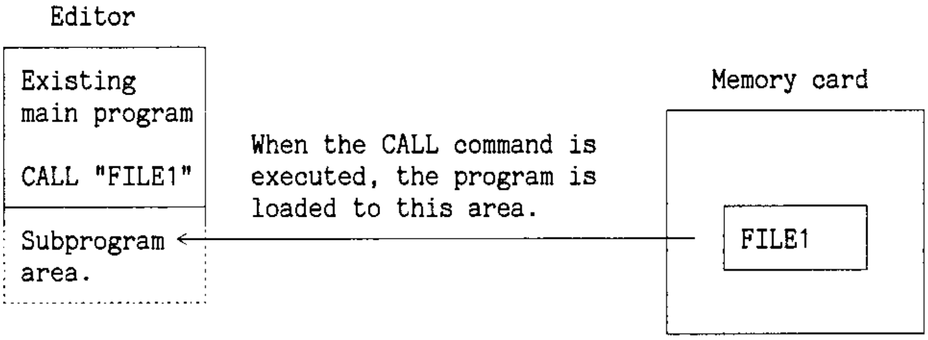
【Format】

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 (19) 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 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.

- If a subprogram contains line numbers, the line numbers are ignored during loading. Even if the line numbers overlap once in the main program, opration is normally performed.

【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.
```

(3) CAT

【Outline】

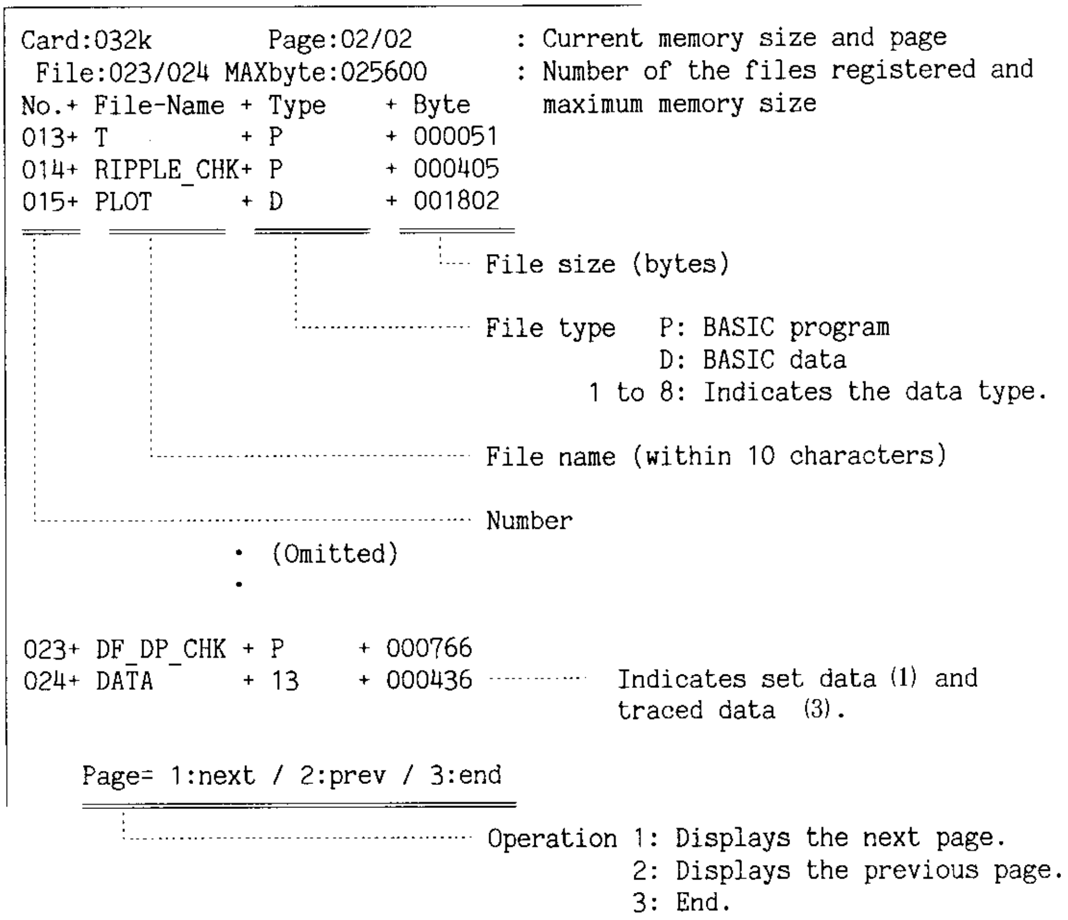
Outputs the list of the files stored in the memory card.

【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 the BASIC mode of the editor.
- If the CAT command is executed, the list is displayed on the screen, as follows:



```
Card:032k        Page:02/02
 File:023/024 MAXbyte:025600
No.+ File-Name + Type    + Byte
013+ T         + P       + 000051
014+ RIPPLE_CHK+ P       + 000405
015+ PLOT      + D       + 001802

        • (Omitted)
        •

023+ DF_DP_CHK + P     + 000766
024+ DATA      + 13    + 000436

    Page= 1:next / 2:prev / 3:end
```

: Current memory size and page

: Number of the files registered and maximum memory size

File size (bytes)

File type P: BASIC program
D: BASIC data
1 to 8: Indicates the data type.

File name (within 10 characters)

Number

Indicates set data (1) and traced data (3).

Operation 1: Displays the next page.
2: Displays the previous page.
3: End.

- Data types includes the following:

  1: Set data
  2: Menu (user-defined)
  3: Traced data A
  4: Traced data B
  5: Normalized data A
  6: Normalized data B
  7: Limit line data
  8: Antenna data

【Example 】

```
CAT
```

【Note】

New cards cannot be used until they are initialized by the (24).

(4) CLEAR

【Outline 】

To initialize 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)

```
【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.
```

(5) CLS

【Outline】

Deletes the screen of the external terminal or the main-body CRT.

【Format】

```
CLS  [1 | 2]
```

【Description】

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

【Example】

```
CLS
CLS 1
CLS 2
```

【Note】

Before use of the graphics screen, the GPIB code "VS3" (Refer to the (2) Graphics function in sub-section 3.2.1.) should be transmitted with the OUTPUT 31 statement (Refer to the (46) OUTPUT.).

【Program Example】

```
No.1
   OUTPUT 31;"VS3"
   GLINE(0,1,0,0,1023,439)
   PRINT "TEST"
   WAIT 2000
   CLS

No.2
   OUTPUT 31;"VS3"
   GLINE(0,1,0,0,1023,439)
   PRINT "TEST"
   WAIT 2000
   CLS 1
```

No.3

```
   OUTPUT 31;"VS3"
   GLINE(0,1,0,0,1023,439)
   PRINT "TEST"
   WAIT 2000
   CLS 2
```

【Result】

No.1
The character string "TEST" is deleted and the graphics objects remain.

No.2
The graphics objects are deleted and the character string "TEST" remain.

No.3
Both graphics objects and the character string "TEST" remain.

(6) 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 (58) 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 (Refer to the (41) ON ERROR.) 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.)

(7) 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 specified label.

- Up to 20 characters can be obtained for the number of label character.

【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 F20 key on the external terminal or 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 the (47) PAUSE.

(8) CONTROL

【Outline】

Sets the values related to BASIC controls.

【Format】

CONTROL <register number>;<numeric expression>

<registser number>: Specify any of 1, 2, and 3.

【Description】

Register number 1: Sets the serial I/O port (RS-232C).
Register number 2: Specifies the left margin for the output on printers.
Register number 3: Specifies whether to output short names to printers.

● Register 1

Register 1 specifies the conditions of the serial I/O ports (RS-232C). Sum the values of the items in the table and give the sum to the command as follows:

CONTROL 1;<X: sum>

Baud rate	1 :	2400 baud	
	2 :	4800 baud	
	3 (0):	9600 baud	◎
Character length	0 :	5 bit	
	4 :	6 bit	
	8 :	7 bit	
	12 :	8 bit	◎
Parity check	0 :	None	◎
	16 :	Odd parity	
	32 :	Even parity	
Stop bit	0 :	None	
	64 :	1 bit	◎
	128 :	1.5 bit	
	192 :	2 bit	

◎: Indicates settings at shipment.

- 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 3

Register 3 specifies whether to display LIST output with short names or full names.

0 : Full name output
1 : Short name output

【Example】

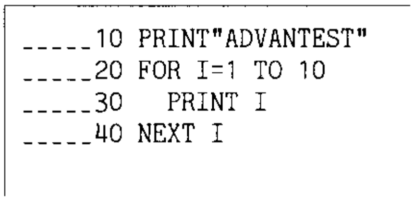
Register 1

Baud rate	:	9600 baud
Character length	:	8 bits
Parity	:	None
Stop bit	:	1 bit

To set the serial I/O port condition as shown above, specify CONTROL 1;0+12+0+64 or CONTROL 1;76.

Register 2

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



```
_____10 PRINT"ADVANTEST"
_____20 FOR I=1 TO 10
_____30   PRINT I
_____40 NEXT I
```

Register 3

To output data on printers with short names, specify CONTROL 3;1.

```
-----10 CSR 10,10
-----20 OUT 31;"CF1.23MZ"
-----30 ENT 31;A
-----40 PRF"%f",A
```

To output data on printers with full names, specify CONTROL 3;0.

```
-----10 CURSOR 10,10
-----20 OUTPUT 31;"CF1.23MZ"
-----30 ENTER 31;A
-----40 PRINTF"%f",A
```

【Note】

Even if the conditions of the serial I/O port are changed, the display of the setting by the main-body panel will not change. When the system is turned on, the serial I/O port conditions are set to the parameter of the RS-232C setting items of the main-body panel.

(9) CURSOR or CRS

【Outline】

Moves the cursor to the specified position on the screen.

【Format】

```
CURSOR <X: column>,<Y: line>
or
CSR <X: column>,<Y: line>
```

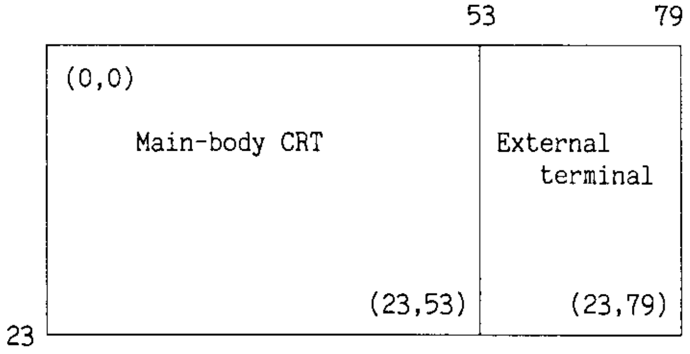
【Description】

CURSOR may be abbreviated to CSR. The CURSOR statement moves the cursor to the position determined by the column and the line.

	Terminal	Non-terminal
<X: column>(integer)	0 ≦X ≦79	0 ≦X ≦53
<Y: line> (integer)	0 ≦Y ≦23	0 ≦Y ≦23

Terminal : Activates the program in the state of external connection.
Non-terminal: Activates the program only in the measuring instrument.

In both CRTs, the position of the top-left corner is (0, 0).



【Example】

```
CURSOR 10,20  : Moves the cursor to column 10, line 20.
CSR 50, 11    : Moves the cursor to column 50, line 11.
```

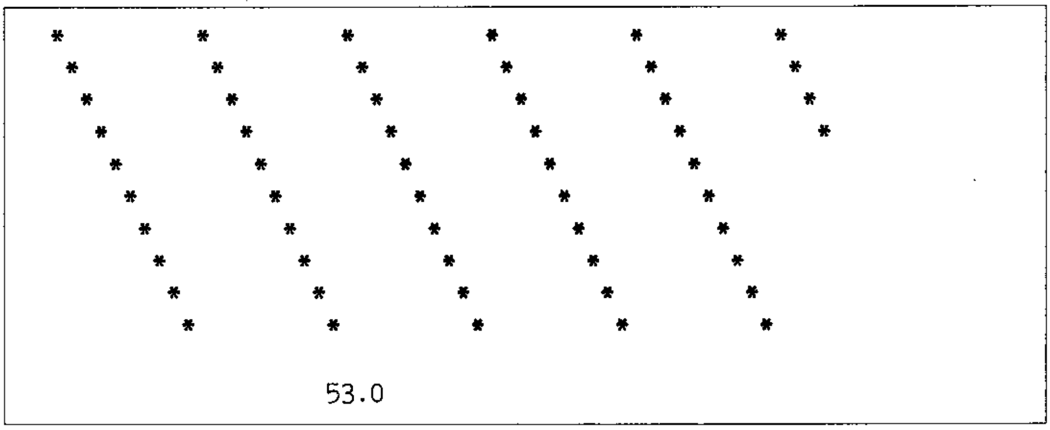
【Note】

In case of non-terminal CRT (Measuring Instrument CRT), the width (X axis: number of columns) of the screen is smaller than that of the terminal CRT screen.

【Program Example】

```
OUTPUT 31;"VS2"
FOR X=0 TO 53
 CURSOR X,X%10
 PRINT "*";
 CSR 20,15:PRINT X
NEXT X
```

【Result】



```
*         *         *         *         *         *
 *         *         *         *         *         *
  *         *         *         *         *         *
   *         *         *         *         *         *
    *         *         *         *         *
     *         *         *         *         *
      *         *         *         *         *
       *         *         *         *         *
        *         *         *         *         *
         *         *         *         *         *

                     53.0
```

(10) DELIMITER

【Outline】

To select 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 (Refer to the (46) OUTPUT.) 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.
```

(11) DIM

【Outline】

Declares the size of array variables or character string variables.

【Format】

```
DIM <X>, {,<X>}
X: variable name (<numeric expression> {,<numeric expression> } ) |
               character string variable name" ["<numeric expression>"] "
```

【Description】

- The DIM statement declares the size of array variables or character string varialbes. The subscript indicates the maximum number of the elements. (If the size is not declared, the maximum number of elements for one dimension is taken to be 10 for the array and 18 for the character string.).

- If tha DIM 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, the following message appears, and the execution of the program is stopped: (memory space full)

- Even if a numeric expression indicating the size of an array variable is of real numbers, its fractional part is truncated.
  The expression is then declared and referenced as an integer expression.

- A numeric expression indicating the size of an array variable declares the length of the character string.

- 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. (As many dimensions as allowed by the memory capacity may be specified).

【Example】

```
DIM A (20)          : Secures 20 elements for the real-type array variable A.
DIM B (30),C(25,5)  : Secures 30 elements for the real-type array variable B
                      and 25 by 5 elements for the real-type two-dimensional
                      array variable C.
DIM S$  [50]        : Secures 50 characters for the character-string
                      variable S$.
DIM D(15),T$  [80]  : Secures 15 elements for the real-type array variable
                      D and 50 characters for the character-string variable
                      T$.
DIM E(3.8,5,2)      : Secures 3 by 5 by 2 elements of the real type 3
                      dimensional array variable E.
```

【Note】

- The minimum number of the subscripts for an array variable is 1.
- The total size of array variables that can be secured differs according to the size of the program.
- Length of character-string variable is a maximum of 128.
- For the declaration of integer-type variables, see the description on the (25) INTEGER statement.
- If the number of the subscripts is out of the specified range or less than or equal to zero, the following error message appears when the array variable is referenced:
  (Array's range error, or Invalid dimension parameter)
- Multi-dimensional arrays cannot be declared as character-string variables.

【Program Example 】

```
DIM A(20),B(5,4);S$ [20]  : Declares real-type array variables and a
                            character-string variable.
FOR I=1 TO 20             : Loops since I is equal to zero until it exceeds 20.
 A(I)=I                   : Assings I the an array variable A.
NEXT I                    : NEXT I
FOR X=1 TO 5              : Loops since X is equal to 1 until it exceeds 5.
 FOR Y=1 TO 4             : Loops since Y is equal to 1 until it exceeds 4.
  B(X,Y)=A((Y-1)*5+X)     : Assigns the array variable A to the array
                            variable B.
 NEXT Y                   : NEXT Y
NEXT X                    : NEXT X
S$="ABCDEFGHIJKLMNOPQRSTUVWXYZ:
                            Assigns a character string of 26 characters to S$.
FOR X=1 TO 5              : Loops since X is equal to 1 until it exceeds 5.
 FOR Y=1 TO 4             : Loops since Y is equal to 1 until it exceeds 4.
  CURSOR X*5,Y:PRINT B(X,Y):Outputs the array variable B.
 NEXT Y                   : NEXT Y
NEXT X                    : NEXT X
PRINT S$                  : Outputs the character string S$.
```

【Result】

```
        1.0  2.0  3.0  4.0  5.0
        6.0  7.0  8.0  9.0  10.0
        11.0 12.0 13.0 14.0 15.0
        16.0 17.0 18.0 19.0 20.0
  ABCDEFGHIJKLMNOPQRST
```

(12) DISABLE INTR

【Outline】

Disables the reception of the interrupts caused by the ON (KEY/SRQ/ISRQ) (Refer to the (42) and (43).) 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 (Refer to the (13).) 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 SO SI"  : Sets IP, a second of sweep, interrupt output,
                               and single sweeping.
*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
 ENABLE INTR                 : Enables interrupts to be received.
 OUTPUT 31;"SI"              : Starts a single sweep.
 F=0                         : F=0
 *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        : Assign 1 to F when bit 3 is on.
 RETURN                      : Return
```

【Result】 Outputs the maximum level of each sweep.

(13) ENABLE INTR

【Outline】

Enables the interrupts caused by the ON (KEY/SRQ/ISRQ) instructions (Refer to the (42) and (43).) to be received.

【Format】

```
ENABLE INTR
```

【Description】

Enables the reception of interrupts that has been disabled with a DISABLE INTR instruction (Refer to the (12).).

【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.

(14) 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 assigned to the entry.

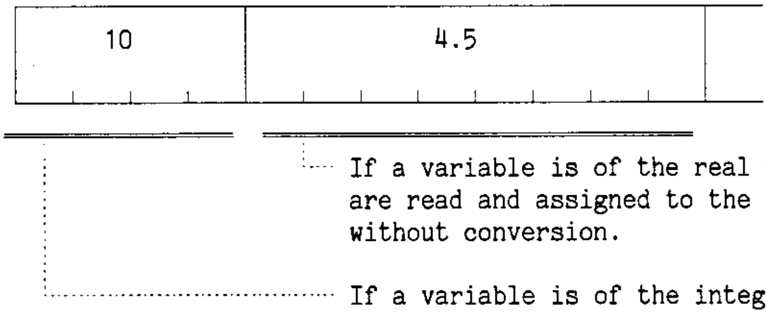
- According to the type specified with the OPEN statement (Refer to the (44).), the data is read in the following format.

  BINARY type

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

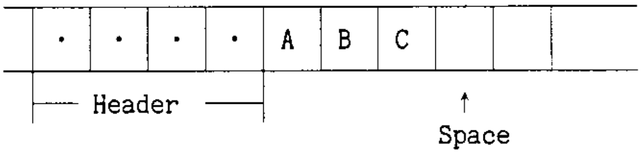
  The data length differs according to the type, as follows:
  Integer type   : 4 bytes
  Real type   : 8 bytes
  Character string: Data size (number of bytes) is indicated by the
                    4-byte header.
  Example)

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



If a variable is of the real type, 8 bytes are read and assigned to the variable without conversion.

If a variable is of the integer type, 4 bytes are read and assigned to the variable without conversion.



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 ssigned to the character string variable.

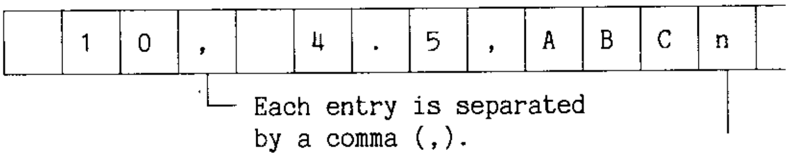
TEXT type

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 assigned 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 (0x0a).

Example)

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



Each entry is separated by a comma (,).

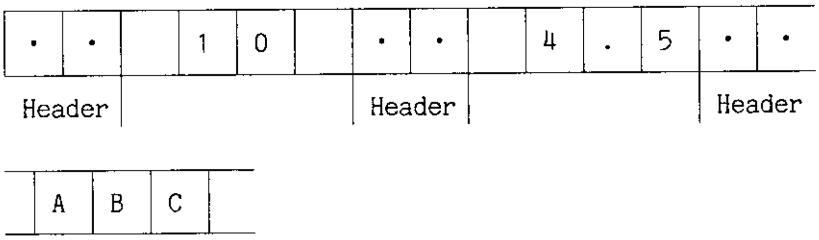
A character string ends with a line feed character (0x0a).

ASCII type

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

```
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
```

(15) ENTER or ENT

【Outline】

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

【Format】

```
ENTER<device address> | 31 | 32;<X>
or
ENT<device address> | 31 | 32;<X>

Device address 0 to 30 : Enter data from the devices on GPIB.
                    31 : Enter data from the measuring device of the main
                         unit.
                    32 : Enter data from the parallel I/O. *1

<X>: [numeric  | character  {, numeric  | character } ]
      variable | variable      variable | 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. To set 32 causes data to enter from the 16 bits parallel I/O into a variable.

- When 32 is specified, only numeric values are handled. Therefore it causes an error to specify in character variable and the program will be interrupted. The following message will be displayed.
  (Invalid type in assume)

- More than one variables can be specified by delimiting a variable with commas(,) or delimiters (except for ENTER 32). In this case, received data will be assigned 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.

*1 Refer to the Parallel I/O (next page).

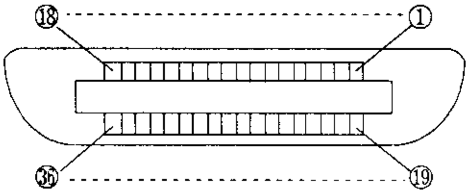
Parallel I/O

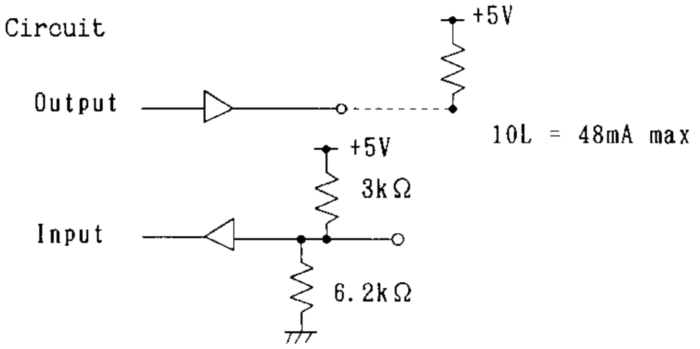
Connector: 36 pins unphenol connector (plug)

【36 pins connector terminal allocation list】

Terminal number	Signal name	Terminal number	Signal name	
1	GND	19	*OE	LSB
2	IN0	20	OUT0	
3	IN1	21	OUT1	
4	IN2	22	OUT2	
5	IN3	23	OUT3	
6	IN4	24	OUT4	
7	IN5	25	OUT5	
8	IN6	26	OUT6	
9	IN7	27	OUT7	
10	IN8	28	OUT8	
11	IN9	29	OUT9	
12	IN10	30	OUT10	
13	IN11	31	OUT11	
14	IN12	32	OUT12	
15	IN13	33	OUT13	
16	IN14	34	OUT14	
17	IN15	35	OUT15	MSB
18		36		

IN0 to IN15 : Input (TTL) positve logic
OUT0 to OUT15 : Output (TTL open corrector) positve logic
*OE : I/O enable negative logic





(cont'd)

```
• When  OUTPUT 32 is used
      OUTPUT 32;1 --- OUTO becomes H level.
      OUTPUT 32;3 --- OUTO and OUT1 become H level.

• When ENTER 32 is used
     when INO is H level         --- ENTER 32;A One is assigned into A.
     when IN1 and IN2 are H level--- ENTER 32;A Six is assigned
                                     into A.
```

【Example 】

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

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

ENTER 32;A
```

【Note】

- To specify numbers other than 0 to 33 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.

【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$
```

③

```
    INTEGER A
    ENTER 32;A
    IF A BAND 4 THEN PRINT "BIT 3"
```

(16) FOR... TO... STEP BREAK/CONTINUE - NEXT

【Outline】

Repeats the execution of the statements between the for 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 1 STEP -.5       : 3-1
  IF K>3 THEN CONTINUE       : 3-2
  PRINT K                    : 3-3
NEXT K                       : 3-4
```

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

2-1 : Assigns 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 : Assigns 10 to the counter J and loops increasing I in increments of -0.5 until it becomes less than 1.
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
```

(17) GLIST

【Outline】

To output a program list to the GPIB printer.

【Format】

```
GLIST [<label> [, ] ]
```

【Description】

- GLIST outputs a program list, whose location is specified by a parameters, to the GPIB printer.
- To specify a label, and comma(,) after the label causes the program list to be printed from the label position to the end of it.
- Up to 20 characters can be obtained for the nember of lebel character.
- To output to a printer, first transfer the BASIC program (MOVE and 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 [<print start line>] [, [<print 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 (51) 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 (MOVE and RUN) to the buffer of the BASIC interpreter side.

【Sample program】

① Transfer a program, whose list is to be output, into the editor, then select "MOVE and 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 "MOVE and RUN".

(Example)

```
PRINTER 3
GLIST
STOP
  ⋮
```

【Result】

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

② To execute "MOVE and RUN" causes the list output to the GPIB address 3 printer.

(18) GLISTN

【Outline】

To output a program list to the GPIB printer.

【Format】

```
GLISTN [<lable>] [,<number of lines>]
```

【Description】

- GLISTN outputs a program list, whose position is specified with parameters, to the GPIB printer.
- If a label is specified, and after it a comma(,) and line numbers are specified, the list is displayed as many lines as specified.
- Up to 20 characters can be obtained for the number of label character.
- If a label is omitted and number of lines is specified, positive number of lines causes as many lines as specified to be displayed from the head of a program, negative number of lines causes to be displayed as many as specified from the end of a program.
- To output follows the same way as the (17) GLIST.

【Example】

```
GLISTN
GLISTN  *ABC, 10
GLISTN  *ABC, -10
GLISTN  , 20
GLISTN  , -20
```

【Note】

Though you may specify with line number, the format differs from the above.

```
GLISTN [<display start line>] [, [<number of lines> ] ]
```

(Example)

```
GLISTN 100
GLISTN 100, 15
GLISTN 100, -15
GLISTN , 20
GLISTN , -20
```

- To specify the GPIB address of a printer, refer to the (51) PRINTER command.
- 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 way to use this command is the same as the (17) GLIST.

(19) 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>.
- Up to 20 characters can be obtained for the number of label character.
- 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             : Assigns 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 n",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
```

(20) 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)
- Up to 20 characters can be obtained for the number of label character.

【Program Example】

```
I=0                 : Assigns 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
```

(21) GPRINT

【Outline】

To output 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 the (51) 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 Assigns "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 a 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 connecs 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 displays from 1 to 10.

(22) GPRINT USING or GPRINT USE

【Outline】

To edit 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 | 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 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 print spaces on the rest of the specified fields.
Z	To insert 0s on the rest of the specified fields.
K	To print 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 print a decimal point.
E	To print with the exponent form (e, sign, exponent).
H	To display numeric or character strings as they are with the European type decimal point.
R	To print the European type decimal point.
*	To print *s on the rest of the specified fields.
A	To print a character.
k	To character strings just as they are.
X	To print spaces.
Listeral	To write a literal in format specifications, put them in\"s.
B	Print numeric with the ASCII cord.
@	To feed lines.
+	To move the printing positions to the top of the same line.
-	To move the printing 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,"\n"
  ```

- 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 a 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
GPRINT USING "DDD.DD,k"; 1.2,"\r"
GPRINT USE "ZZZ.ZZ,k"; 1.2,"\r"
GPRINT USE "***.ZZ,k"; 2.4,"\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
```

(23) IF THEN ELSE/ELSE IF/END IF

【Outline】

Tests a condition and causes a brance 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 than 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
AND
OR
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
```

(24) INITIALIZE or INIT

【Outline】

Initializes a memory card.

【Format】

```
INITIALIZE
or
INIT
```

【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
INIT
```

【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 an memory card that stores files is initialized, all of the files will be deleted.

(25) 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 same as that for integer constant, as follows:
  -2, 147, 483, 648 to +2, 147, 483, 647
- It is recommended that a variable that only 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.
  (As many dimensions as allowed by the memory capacity may to specified).

【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.

- The 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                      : Assigns 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.
                               Assigns 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
```

(26) INTERFACE CLEAR

【Outline】

To initialize 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
```

【Sample program】

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

(27) INPUT or INP

【Outline】

Assigns 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 assigned 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 assigned to a numberic variable does not include any numbers. 0 is assigned 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
```

(28) LIST

【Outline】

Displays the program list on the screen of an external terminal or the main body.

【Format】

```
LIST [<label> [,] ]
```

【Description】

- Displays a program list at the position specified with the parameter on the screen.
- 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 (MOVE and RUN), stop the program and then issue the LIST command in the BASIC mode. A LIST command can be placed in the program.
- Up to 20 characters can be obtained for the number of label character.

【Example】

```
LIST

LIST *ABC,
```

【Note】

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

```
LIST [<start line>]  [, [<end line>] ]

Example:  LIST 100
          LIST 100,
          LIST ,200
          LIST 100,200
```

- If an external terminal is connected to the system when it is turned on, the list is output to the external terminal. If not, the list is displayed on the main-body screen.

【Program example】

Execute MOVE and RUN of the editor.
Stop the program with Control C.
Execute LIST in the BASIC mode of the editor.

【Result】

A progam list is displayed on the screen.

(29) LISTN

【Outline】

Displays a program list on the screen of an external terminal or the main body.

【Format】

```
LISTN [<label>]  [,<number of lines>]
```

【Description】

- Displays the program list at the position specified with the parameter on the screen.
- If a label is followed by a comma and a number of lines is specified, the specified number of the lines from the label will be listed on the screen.
- If a label is omitted and a plus number is specified, the specified number of lines from the top of the program will be displayed. If the line number is a minus number, lines will be counted from the bottom to the top.
- A program list will be output with the same procedure as that for the (28) LIST.
- Up to 20 characters can be obtained for the number of label character.

【Example】

```
LISTN
LISTN  *ABC,10
LISTN  *ABC,-10
LISTN  ,20
LISTN  ,-20
```

【Note】

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

```
  LISTN [<start line>]  [,<number of lines>]

 Example:  LISTN  100
           LISTN  100,15
           LISTN  100,-15
           LISTN  ,20
           LISTN  ,-20
```

- If an external terminal is connected to the system when it is truned on, the list is ouput to the external terminal. If not, the list is displayed on the main-body screen.

(30) LLIST

【Outline】

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

【Format】

```
LLIST [<label> [,] ]
```

【Description】

- Outputs the program list at the position specified with the parameter to the serial I/O port (RS-232C).
- 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 (MOVE and RUN), stop the program and then issue the LLIST command in the BASIC mode. An LLIST command can be placed in the program.
- Up to 20 characters can be obtained for the number of label character.

【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
```

- For the setting of the serial I/O port (RS-232C), see the (8) CONTROL.

【Program example】

Execute MOVE and 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.

(31) LLISTN

【Outline】

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

【Format】

```
LLISTN [<label>]  [,<number of lines>]
```

【Description】

- Outputs the program list at the position specified with the parameter to the serial I/O port (RS-232C).
- If a label followed by a comma and a number of lines is specified, the specified number of the lines from the label will be listed on the screen.
- If a label is omitted and a plus number is specified, the specified number of lines from the top of the program will be displayed. If the line number is a minus number, lines will be counted from the bottom to the top.
- Up to 20 characters can be obtained for the number of label character.
- A program list will be output with the same procedure as that for the (30) LLIST.

【Example】

```
LLISTN
LLISTN  *ABC,10
LLISTN  *ABC,-10
LLISTN  ,20
LLISTN  ,-20
```

【Note】

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

```
  LLISTN [<start line>]  [, [<number of lines>] ]

Example:  LLISTN  100
          LLISTN  100,15
          LLISTN  100,-15
          LLISTN  ,20
          LLISTN  ,-20
```

- For the setting of the serial I/O port (RS232C), see the (8) CONTROL.

(32) LOCAL

【Outline】

To release 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
```

【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.)

(33) LOCAL LOCKOUT

【Outline】

To lock 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.

【Sample progrm】

```
REMOTE 11,12
LOCAL LOCKOUT

LOCAL
```

【Result】

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

(34) LPRINT

【Outline】

Outputs numeric values and character strings to the device (terminal, printer, etc.) connected through the RS-232C 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 ezch 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 Assigns "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 an external terminal is used, the LPRINT statement works in the same way as the (49) PRINT statement does.
- 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】

If an external terminal is connected, the following is output to the terminal:

```
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
```

(35) LPRINT USING or LPRINT USE

【Outline】

Edits numeric values or character strings and outputs them through the RS-232C 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-232C) 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 print spaces on the rest of the specified fields.
Z	To insert 0s on the rest of the specified fields.
K	To print 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 print a decimal point.
E	To print with the exponent form (e, sign, exponent).
H	To display numeric or character strings as they are with the European type decimal point.
R	To print the European type decimal point.
*	To print *s on the rest of the specified fields.
A	To print a character.
k	To character strings just as they are.
X	To print spaces.
Listeral	To write a literal in format specifications, put them in\"s.
B	Print numeric with the ASCII cord.
@	To feed lines.
+	To move the printing positions to the top of the same line.
-	To move the printing 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,"\n"
  ```

- 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"; 2.4,"\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
```

(36) 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 the (40) ON END.

(37) 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 the (41) ON ERROR.

(38) OFF KEY

【Outline】

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

【Format】

```
OFF KEY <key number>

      <key number>:  Numeric expression (any of 1, 2, 3, 4, 5, and 6)
```

【Description】

- The OFF KEY instruction releases the definition of the branch destination (the statement to branch to) for <key number> that has been given by an ON KEY instruction.

- Definitions of the branch destination with key numbers may be released respectively.

【Example】

```
OFF KEY 1
OFF KEY 2
OFF KEY 3
OFF KEY 4
OFF KEY 5
OFF KEY 6
```

【Note】

- After the definition of the branch destination has been released with this instruction, no branch will be taken even if the specified key is pressed.

- The branch destination can be defined with the (42) ON KEY.

【Program example】

```
ON KEY 1 GOSUB *K1 : Defines the branch destination (subroutine) for key
                     number 1.
ON KEY 2 GOTO *K2  : Defines the branch destination (jump) for key number 2.
!
ENABLE INTR        : Enables an interrupts to be received.
*L                 : Loop
GOTO *L            : Loop
!
*K1                : Operations for the interrupt caused by key number 1.
PRINT "KEY 1"      : Outputs "KEY 1".
OFF KEY 1          : Releases the definition of the branch destination for
                     key number 1.
ON KEY 2 GOTO *K2  : Defines the branch destination for key number 2 (jump).
RETURN             : Return
!
*K2                : Operations for the interrupt caused by key number 2.
PRINT "KEY 2"      : Outputs "KEY 2".
OFF KEY 2          : Releases the definition of the branch destination for
                     key number 2.
ON KEY 1 GOSUB *K1 : Defines the branch destination for key number 1
                     (subroutine).
GOTO *L            : Jumps to *L.
```

【Result】

```
KEY 1
KEY 2
KEY 1
KEY 2
   .
   .
   .
```

The same character string will be output only once even if the same key is pressed twice continuously.

(39) 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】

- 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 the (43) ON SRQ or ON ISRQ.

(40) 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)
- Up to 20 characters can be obtained for the number of label character.

【Example】

```
ON END #FD GOTO *ERR

ON END #FD GOSUB *ERR
```

【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 the (36) OFF END.
- For writing and reading data, see the (45) OUTPUT #··· and (14) 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
                               assigns 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
```

(41) 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.
- Up to 20 characters can be obtained for the number of label character.

【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 the (37) 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
30: 0 divide
<<ERROR>>
  >>    30: 0 divide
```

(42) ON KEY - GOTO/GOSUB

【Outline】

Defines the statement to which control will branch to when an interrupt is received from a full keyboard or the main-body panel (ten-key pad).

【Format】

ON KEY <key number> GOTO <label>

ON KEY <key number> GOSUB <label>

<key number>: Numeric expression (any of 1, 2, 3, 4, 5, and 6)

【Description】

- This instruction defines the statement to which control will branch to when an interrupt caused by a full keyboard or the main-body panel (ten-key pad).
- Numeric keys 1, 2, 3, 4, 5, and 6 are available. The key numbers that may be defined with ON KEY <key number> are between 1 and 6.
- To use this instruction on the ten-key pad of the external terminal, change the settings of the external terminal, beforehand.
- Up to 20 characters can be obtained for the number of label character.

【Example】

```
ON KEY 1 GOTO *K1
ON KEY 2 GOSUB *K2
ON KEY 3 GOTO *K3
ON KEY 4 GOSUB *K4
ON KEY 5 GOTO *K5
ON KEY 6 GOSUB *K6
```

【Note】

- 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 the (13) ENABLE INTR and (12) DISABLE INTR.
- To release the statement to branch to, execute the (38) OFF KEY.

【Program example】

```
ON KEY 1 GOTO *SC
ON KEY 2 GOTO *SD
ON KEY 3 GOTO *SE
ON KEY 4 GOSUB *SF
ON KEY 5 GOSUB *SG
ON KEY 6 GOSUB *RA
ENABLE INTR
*LOOP
 GOTO *LOOP
STOP

*SC:BUZZER 261,300:GOTO *LOOP
*SD:BUZZER 294,300:GOTO *LOOP
*SE:BUZZER 330,300:GOTO *LOOP

*SF
 DISABLE INTR
 BUZZER 349,300
 ENABLE INTR
 RETURN

*SG
 DISABLE INTR
 BUZZER 392,300
 ENABLE INTR
 RETURN

*SA
 DISABLE INTR
 BUZZER 440,300
 ENABLE INTR
 RETURN
```

【Result】

Keys from 1 to 6 are associated with do, re, mi, fa, sol, la, respectively. Each sound is output when the associated key is pressed.

(43) 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.
- Up to 20 characters can be obtained for the number of label character.

【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 the (13) ENABLE INTR and (12) DISABLE INTR.
- To release the statement to branch to, execute the (39) 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 levles are obtained.

(44) 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.

- Up to 10 characters can be obtained for a file name.

- 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.
  Up to 20 characters should be writen after immediately # using alphanumeric characters for a file discriptor.

- Type

  There are the following three types: BINARY, TEXT, and ASCII.
  (If the type is not specified, BINARY is assumed.)

  BINARY

  Data is read or written as expressed inside.
  The data size differs according to the type:

Integer	4 bytes
Real	8 bytes
Character string	a 4-bytes header and a character string (in the ASCII notation)

  If the number of the elements in a character string is odd, one-byte space will follow the string.

TEXT

Data is converted into the ASCII codes before it is read or written. If data is numeric, a minus sign (-) or a space will be placed before the data.

If the TEXT type is specified, the USING specification is allowed.

ASCII

Each data object consists of a two-byte header (identifier) and ASCII codes. If the data object is numeric, a minus sign (-) or a space will be placed before it. If the number of the elements in a character string is odd, a one-byte space will follow the string.

【Example】

```
OPEN "FILE0" FOR OUTPUT AS # 10P
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 "FILE0" FOR INPUT AS # 10P
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.

- The same file may not be opened with more than one file descriptor at the same time.

- If the file opened in the OUTPUT (write) mode exists in the memory card, the error occurs and the execution of the program is stopped.
  Necessary files are thus prevented from being deleted by mistake.
  To renew a file, the file should be deleted with a (47) PURGE command.

- The multi files cannot be opened simultaneously at the same time.
  Always open only one file for each of OUTPUT and INPUT mode.

【Program Example 】

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

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

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

【Result】

If the CAT command is entered after the above program has been executed, the result shows as follows:

```
Data size of file A  : 000080
Data size of file AA : 000051
Data size of file AAA: 000062
```

(45) 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 (44) OPEN statement, the data is written in the following format.

BINARY type

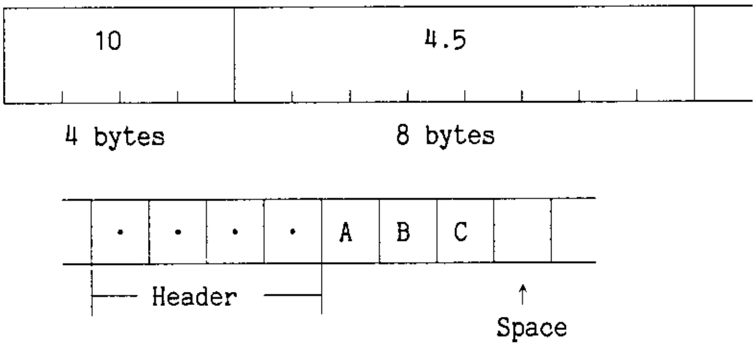
Data is written in the same type as that of the internal expression.

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"
```

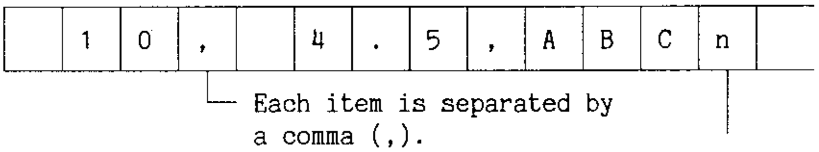


TEXT type

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 (0x0a).

Example)

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



Each item is separated by a comma (,).

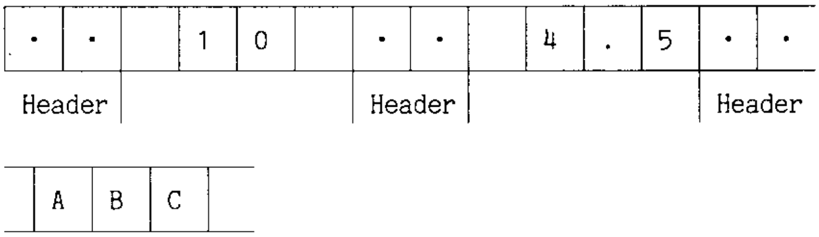
A character string ends with a line feed character (0x0a).

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 the (44) 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
```

【Result】

If the CAT command is entered after the above programs have been used, the results shows the following:

```
Data size of file A   : 000120
Data size of file AA  : 000082
Data size of file AAA : 000104
```

(46) OUTPUT or OUT

【Outline】

To send data to the GPIB, measuring device of the main-unit or parallel I/O.

【Format】

```
OUTPUT <device address> | 31 | 32;<X>
or
OUT <device address> | 31 | 32;<X>

Device address 0 to 30 : Enter data from the device on GPIB.
                    31 : Enter data from the measuring device of the main
                         unit.
                    32 : Enter data from the parallel I/O.    ※1

 <X>:┌numeri   | character  {  ,  numric   | character  }  ┐
     └variable | variable         variable | variable      ┘
```

【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.)

- To set 32 causes outputs data to the 16 bits parallel I/O.   ※2
  The value can be output are between 0 and 65535. Only numeric expression is available in the 32 specification. If character strings are used, an error will result to interrupt a program execution.

※1 : Refer to section 7 of the manuals(separate volumes) of R3261/R3361 series for the list of the GPIB codes.

※2 : Refer to the parallel I/O.

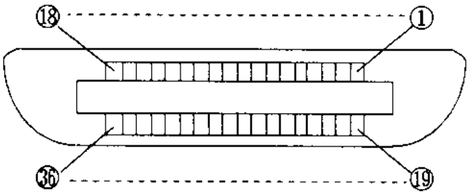
Parallel I/O

Connector: 36 pins unphenol connector (plug)

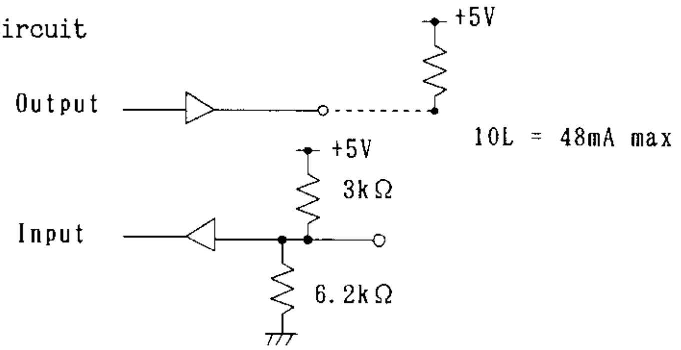
【36 pins connector terminal allocation list】

Terminal number	Signal name	Terminal number	Signal name	
1	GND	19	*OE	LSB
2	IN0	20	OUT0	
3	IN1	21	OUT1	
4	IN2	22	OUT2	
5	IN3	23	OUT3	
6	IN4	24	OUT4	
7	IN5	25	OUT5	
8	IN6	26	OUT6	
9	IN7	27	OUT7	
10	IN8	28	OUT8	
11	IN9	29	OUT9	
12	IN10	30	OUT10	
13	IN11	31	OUT11	
14	IN12	32	OUT12	
15	IN13	33	OUT13	
16	IN14	34	OUT14	
17	IN15	35	OUT15	MSB
18		36		

IN0 to IN15 : Input (TTL) positive logic
OUT0 to OUT15 : Output (TTL open corrector) positive logic
*OE : I/O enable negative logic



Circuit



(cont'd)

- When OUTPUT 32 is used

```
OUTPUT 32;1  ····  OUTO becomes H level.
OUTPUT 32;3  ····  OUTO and OUT1 become H level.
```

- When ENTER 32 is used

```
When INO is H level  ····  ENTER 32;A  One is assigned into A.
When IN1 and IN2 are H level   ····
                           ENTER 32;A  Six is assigned into A.
```

【Example】

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

OUTPUT 31;"CF",CF,"MZ"

OUTPUT 32;128
OUTPUT 32;DT
```

【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.

【Sample program】

```
OUTPUT 5;"CF?"
ENTER 5;C
ENTER 5;S$
PRINT C,S$

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
```

(47) 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.

- Up to 10 characters can be obtaind for a file name.

(48) PRINTF or PRF

【Outline】

To output numeric and character strings to external terminals or main display.

【Format】

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

 <x> : numeric expression  |  character expression
```

【Description】

- If external terminals are used, numeric or character strings will be output to them. If not, the output will be to the main display.
- 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       : Displays the variable C with the decimal numeral.
PRINTF "C=%5d", C      : Displays the variable C in five figures with the
                         decimal numeral. (Right justification)
PRINTF "C=%-5d", C     : Displays the variable C in five figures with the
                         decimal numeral. (Left justification)
PRINTF "C=%05d", C     : Displays the variable C in five figures with the
                         decimal numeral.
                         (Right justification : suppress with zero)

PRINTF "H=%x", H       : Displays the variable H with the hexadecimal
                         numeral.
PRINTF "H=%4x", H      : Displays the variable H in four figures with the
                         hexadecimal unmeral. (Right justification)
PRINTF "H=%-4x", H     : Displays the variable H in four figures with the
                         hexadecimal unmeral. (Left justification)
PRINTF "H=%04x", H     : Displays the variable H in four figures with the
                         hexadecimal unmeral.
                         (Right justification : suppress with zeros)

PRINTF "S$=%s", S$     : Displays character strings
PRINTF "S$=%8s", S$    : Displays character strings justifying to the right
                         most column.
PRINTF "S$=%-8s", S$ : Displays character strings justifying to the left
                         most column.
PRINTF "%20.10s", S$ : Displays character strings justifying to the right
                         most column in a 20 characters length field.
PRINTF "%-20.10s", S$: Displays ten character strings justifying to the
                         left most column in a 20  characters length field.

PRINTF "F=%f",F        : Displays the variable F in a real number with
                         decimal numeral.
PRINTF "F=%8.2f", F    : Displays the variable F in a eight figures to the
                         second decimal including a decimal point.
PRINTF "F=%08.2f", F : Displays the xariable F in eight figures to the
                         second decimal including a decimal point. (Suppress
                         with zeros)
```

【Note】

- Since the output using PRINTF statements does not strat 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 print may be resulted.

【Example 】

```
PRINTF "%d * %4d=%05",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
```

(49) PLINT or ?

【Outline】

To output numeric or character strings to external terminals or the main display.

【Format】

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

<X> : Numeric expresion | Character expession

【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 Assigns "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.

【Note】

If external terminals are used, the numeric or character strings will be output to them. If not used, output will be to the main display.

【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
```

If external terminals connect, the output will be to them.
If not connect, to the main display.

(50) PRINT USING or PRINT USE

【Outline】

To edit numeric or character strings and output to external terminals or to the main display.

【Format】

```
PRINT USING < image specification > ;  {, <x> }
or
PRINT USE   < image specification > ;  {, <x> }
  <x> : numeric expression | character expression
```

【Description】

- Numeric or character strings are edited and output to external terminals, if they connect. If not connect, output is to the mail display.

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

【image specification list】

D	To print spaces on the rest of the specified fields.
Z	To insert 0s on the rest of the specified fields.
K	To print 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 print a decimal point.
E	To print with the exponent form (e, sign, exponent).
H	To display numeric or character strings as they are with the European type decimal point.
R	To print the European type decimal point.
*	To print *s on the rest of the specified fields.
A	To print a character.
k	To character strings just as they are.
X	To print spaces.
Listeral	To write a literal in format specifications, put them in\ "s.
B	Print numeric with the ASCII cord.
@	To feed lines.
+	To move the printing positions to the top of the same line.
-	To move the printing 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.

【Description 】

```
PRINT USING "DDD.DD" ; 1.2

PRINT USE "ZZZ.ZZ" ; 1.2
```

【Note】

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

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

- Image

【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
**1.20
2.34     A  1,20
```

(51) PRINTER

【Outline】

To specify the GPIB addresses for the GPIB printer output.

【Format】

```
PRINTER <device addresses>
<device addresses> : 0 to 30
```

【Description】

- PRINTER specifies the GPIB addresses of devices to which commands of GPRINT, GLIST, GLISTN, etc. are issued. (Note to execute a PRINTER command before executions of GPRINT, GLIST, GLISTN, etc.)
- The device addresses can be specified between 0 and 30. (Since 31 is set as the default, the device addresses, from 0 to 30, should be set with a PRINTER command whenever PRINTER is used.)

【Example】

```
PRINTER 1

PRINTER 5
```

【Note】

- The default, 31, has no relation with the address number 31 of OUTPUT /INPUT commands.
- If no addesses is specified, no output will be executed.

【Sample program】

```
PRINTER 1              : Spcifies the output to device address 1.
GPRINT "ABCD"          : Outputs "ABCD" to address 1.
A=12                   : Assigns 12 to the variable A.
PRINTER A              : Specifies the output to device address 12.
GPRINT "printer=",A    : Outputs character strings and contents of variable
                         to address 12.
```

【Result】

Each of two GPIB printers is assigned to address 1 and address 2.
Address 1 printer outputs
ABCD
and
Address 12 printer outputs
printer=12. 0

(52) READ DATA/RESTORE

【Outline】

To assign a constant in a DATA statements into a variable.

【Format】

```
READ <X> {, <X> }
<X> : numeric variable | character variable

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

RESTORE  [<label> ]
```

【Description】

- READ DATA assigns 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 assigned into a variable. Then the corresponding DATA statement constants will be retrieved and assigned 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 lable 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.)
- Up to 20 characters can be obtained for the number of label character.

【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)

- If the type of the constant specified in a DATA statement and type of the variables to read are different, an error will be resulted. (Invalid type in getdata)

【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
```

(53) REM or !

【Outline】

To write 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
```

(54) REMOTE

【Outline】

To set 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 the (32) 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 following commands causes the remote mode without a execution of a REMOTE command.
  OUTPUT ············ See (46).
  SEND LISTEN ········ See (61).

【Example】

```
REMOTE

REMOTE 6

REMOTE 6,7,10
```

【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.)

(55) RENAME

【Outline】

Renames a file in the memory card.

【Format】

```
RENAME <current file name>,<new file name>
```

【Description】

- The RENAME command changes the file name of a program file or data file in the memory card.
- The RENAME command should be entered in the BASIC mode of the editor.
- If the specified file is not found, the following error message appears: (RENAME (current file name, new file name) error)

【Example】

```
RENAME "BAS1","BASIC1"
```

【Notes】

- Before renaming a file, check the contents of the file with the CAT command or Load of the editor.
- A file name should be within 10 characters. If 11 or more characters are specified, the 11th and the following characters are ignored.
- The RENAME command cannot be executed when the write protect of the memory card is turned on.

(56) REQUEST

【Outline】

To output a SRQ(Service Request) to a external controller on the standard GPIB(upper part of the main-unit rear panel).

【Format】

```
REQUEST
```

【Description】

- REQUEST outputs a SRQ(Service Request) to a external controller on the standard GPIB(upper part of the main-unit rear panel).
- SRQ(Service Request) is as follows.

status byte

REQ	RSV						
7	6	5	4	3	2	1	0 (bit)

To execute this command makes the seventh bit of the status byte on.

【Example】

```
REQUEST
```

【Sample program】

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

【Result】

If A>1 SRQ is transferred to a external controller and data is output.
In other case, A is printed.

(57) RUN

【Outline】

Starts executing a program.

【Format】

```
RUN [<label>]
```

【Description】

- If the label is omitted, the execution starts from the top of the 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 MOVE and RUN.

【Example】

```
RUN

RUN *ABC
```

【Note】

- A RUN command with a label should be executed in the BASIC mode. (To start the program execution, press the F19 key on the external terminal or the RUN key on the main-body CRT. In this case, the program must be transferred to the BASIC buffer beforehand.)
- Variables are not initialized with a CONT command.
- Up to 20 characters can be obtained for the number of label character.

(58) 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 (7) CONT command is entered or the F20 (CONT) key of the terminal or the soft key 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.

(59) SCRATCH

【Outline】

Clears the internal buffer of the BASIC interpreter.

【Format】

```
SCRATCH  [<numeric expression>]
```

<numeric expression>: Either 1 or 2.

【Description】

- Execute this command when a BASIC program that has been transferred to the buffer is not required.
- To initialize only the program data (variable values, etc.) in the buffer, specify 1 as <numeric expression>.
- To initialize only the program (variable values will remain) in the buffer, specify 2 as <numeric expression>.

【Example】

```
SCRATCH

SCRATCH 1

SCRATCH 2
```

【Notes】

A SCRATCH command cannot be executed if the BASIC program has not been transferred to the internal buffer of the BASIC interpreter.

(60) SELECT CASE/CASE ELSE/END SELECT

【Outline】

To branch more than one times, using the value of a expression as a condition

【Format】

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

【Description】

- SELECT CASE executes the CASE statements (multiple statements) whose specified value of a expression corresponds with the value of a expression specified in a SELECT statement.

- 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
```

(61) SEND

【Outline】

To output commands and data to the GPIB.

【Format】

```
SEND  <A> | <B> | <C> {,<A> | <B> }

<A> :  CMD  | DATA | LISTEN [<D> {,<D>} ]
<B> :  UNT  | 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.
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".
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 exection 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
```

【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】

The center frequency of the device address 11 will be reset every second between 1 MHz and 9 MHz.
Finally the device address 11 will be released its listener mode.

(62) SPOLL (X)

【Outline】

To provide 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 byts 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	None
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 "SO", 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.

【Sample program】

```
INTEGER S
OUTPUT 31;"SO"
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.

(63) SPRINTF

【Outline】

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

【Format】

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

<X>: Numeric expression | character-string expression

【Description】

- The SPRINTF instruction is equivalent to the (48) PRINTF instruction, except that SPRINTF can assign 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 assigns 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 assigned to the character-string expression to be output. Therefore, specify %% to assign % to the output.

【Example】

```
SPRINTF S$,"C=%d",C
          Assigns variable C as a decimal number.

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

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

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

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

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

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

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

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

SPRINTF B$,"S$=%8s",S$
          Assigns a character string right-justified.

SPRINTF B$,"S$=%-8s",S$
          Assigns a character string left-justified.

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

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

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

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

SPRINTF B$,"F=%08.2f",F
Assigns 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$,"%d * %4d=%05",12,34,12*34
PRINT S$
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
```

(64) STEP

【Outline】

Execute a line of a BASIC program.

【Format】

```
STEP [<label>]
```

【Description】

- If a label is specified, the labeled line is executed.
- If a label is not specified, the next line of the line that was executed last is executed. (If this command is issued continuously, the program is executed line by line.)
- Even when the program has temporarily stopped with a PAUSE command, the program lines can be executed one by one with STEP commands.

【Example】

```
STEP

STEP *ABC
```

【Note】

- STEP commands can only be executed when the program has been stopped with the stop key (control C) or a PAUSE command. (STEP commands can be used only in the BASIC mode of the pop-up menu.)
- STEP commands cannot be executed in the loop of a GOTO statement or FOR-NEXT statement.
- Up to 20 characters can be obtained for the number of label character.

(65) STOP

【Outline】

To terminate 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       : Assigns 1 into the counter1, 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.
```

(66) TRIGGER

【Outline】

To send 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
```

【Sample program】

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

【Result】

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

(67) WAIT

【Outline】

To terminate 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".

3. BUILT-IN FUNCTION

3.1 Outline

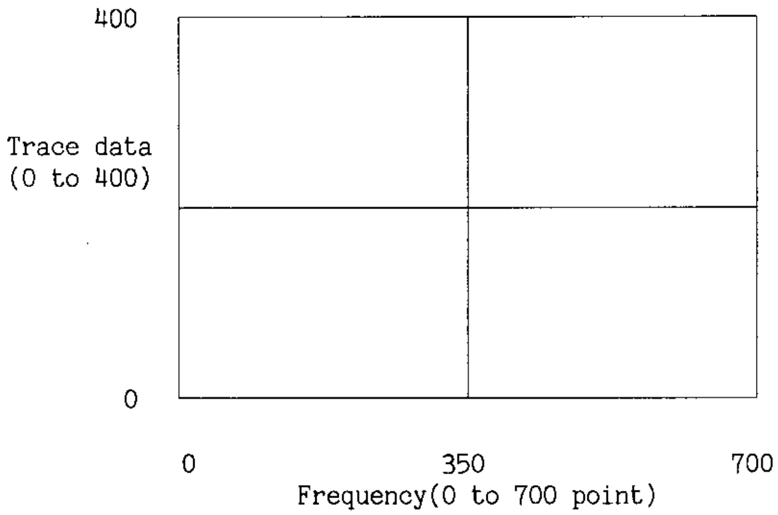
Built-in function makes you to analyze easily and shorten time for software development. In result time for process becomes shorter than former process time by GPIB operation.

3.2 BEFORE USING THE BUILT-in FUNCTION

3.2.1 Point, Trace data and Notes

(1) Point and Trace data

To use the built-in function, knowledge of the point is needed.
The point is used in the frequency (horizontal axis) and level (vertical axis). From now on, the point is used for a frequency (horizontal axis) and level (vertical axis).
From now on, the point is used for a frequency (horizontal axis) and the trace data is used for a level (vertical axis).

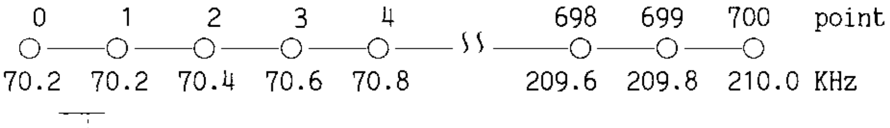


- Point
  Frequency per point : frequency width [span] /700

- Trace data
  Level per point : dynamic range/400

Note: In calculation of the point and trace data, data which cannot be divided by 700 or 400 will be rounded.

(Example)
Center frequency [CENTER] : 140KHz
Frequency width [SPAN] : 140KHz



At this point the calculated points are divided right and left by rounding.

(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.

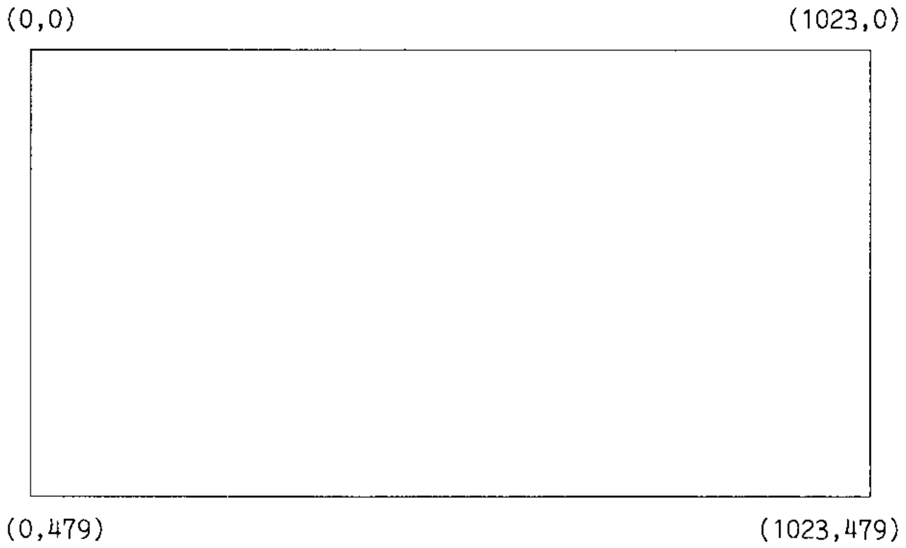
Waveform screen : Appears first after the preset button is depressed.
Character screen(Execution screen):
Displays characters by BASIC PRINT command and so on.
Graphic screen : The BASIC graphic function displays this screen.

Before output either of the above screeens by the BASIC program, execute either of the following GPIB codes.

Display mode	GPIB command
Waveform screen	OUTPUT 31;"VS0"
Waveform screen +Character screen	OUTPUT 31;"VS1"
Character screen	OUTPUT 31;"VS2"
Graphic screen	OUTPUT 31;"VS3"

Before executing the graphic function, execute OUTPUT 31 for "VS3".

The graphic screen is specified as follows.



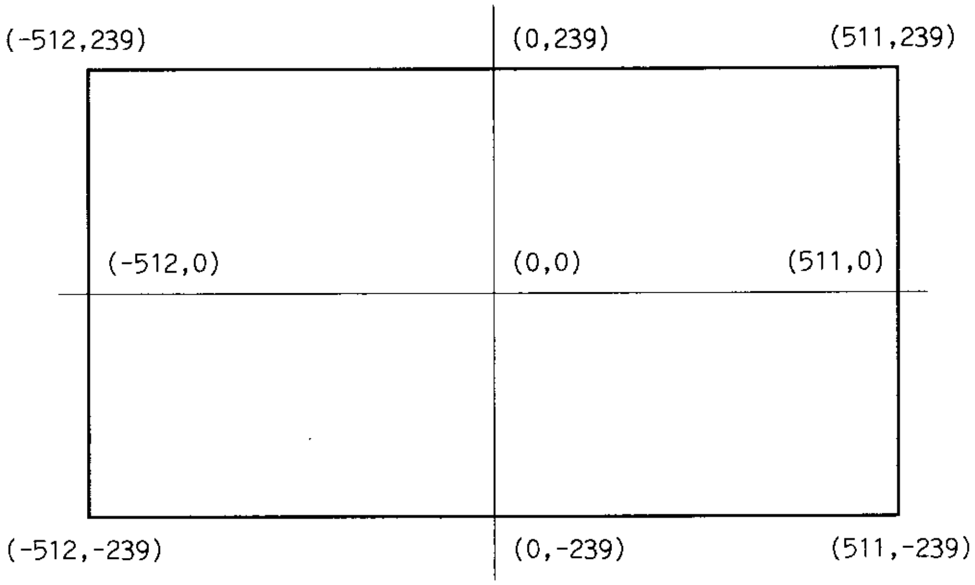
The above figure is called an absolute address. The upper left is the original (0,0).

The original is determined optionally inside the above area by specifying the viewport address through the CADRS function.

(Example)

If GADRS (1, 512, 240) is determined, the display is as the follows.

Note : It causes an error to specify the viewport address over the settled screen area.



Refer to ① to ⑧ of section 3.4.

3.2.2 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 400) and trace A/B(0/1) out of the range causes an error. (function error)

3.2.3 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) Built-in function and Graphic function

function	structure	describe
① Calculating a frequency	FREQ	Calculates a frequency from a point value.
	DFREQ	Calculates a frequecny from a width between points.
	FMAX	Calculates the frequency at the maximum level position between two positions specified with point value.
	FMIN	Calculates the frequency at the minimum level position between two positions specified with point value.
	BND	Calculates the frequency bandwidth of a LOSS level at a position specified with point values.
	BNDL	Calculates the low frequency bandwidth of a LOSS level at a position specified with point values.
	BNDH	Calculates the high frequency bandwidth of a LOSS level at a position specified with point values.

function	structure	describe
① Calculating a frequency (cont'd)	CBND	Calculates the frequency bandwidth of a LOSS level at a position specified with a frequency.
	CBNDL	Calculates the low frequency bandwidth of a LOSS level at a position specified with a frequency.
	CBNDH	Calculates the high frequency bandwidth of a LOSS level at a position specified with a frequency.
	FRPLHN	Calculates the frequency at the maximum point n-th from the left on the frequency axis.
	FRPLLN	Calculates the frequency at the minimum point n-th from the left on the frequency axis.
② Calculating point (Frequency point : Horizontal axis)	POINT	Calculates a point (horizontal axis).
	DPOINT	Calculates a point between specified frequencies.
	PMAX	Calculates the frequency point (horizontal axis) maximum level position between two positions at the specified with point values.
	PMIN	Calculates the frequency point (horizontal axis) at the minimum level position between two positions specified with point values.
	PRPLHN	Calculates the point at the maximum point n-th from the left on the frequency axis.
	PRPLLN	Calculates the point at the minimum point n-th from the left on the frequency axis.
③ Calculating point (Trace data: Vertical axis)	LVPOINT	Calculates the point (vertical axis) of trace data from a level.
	LVDPOINT	Calculates the point (vertical axis) of trace data between levels.
④ Calculating level	VALUE	Calculates a level at a frequency position specified with a point value.
	DVALUE	Calculates a difference of levels at two frequency positions specified with point values.
	CVALUE	Calculates a level at a frequency position.
	DCVALUE	Calculates the level difference between two specified frequency positions.
	LEVEL	Calculates a level of trace data specified with a point value (vertical axis).
	DLEVEL	Calculates a level between points (vertical axis).
	MAX	Calculates the maximum level between two positions specified with point values.
	MIN	Calculates the minimum level between two positions specified with point values.

function	structure	describe
④ Calculating level (cont'd)	RPL1	Calculates the level difference between the maximum value at the miximum point and minimum value at the minimum point.
	VRPLHN	Calculates the Level at the maximum point n-th from the left on the frequency axis.
	VRPLLN	Calculates the Level at the minimum point n-th from the left on the frequency axis.
⑤ Calculating the total power	POWER	Calculates total power between two positions specified with point values.
⑥ Calculating ripple number (maximum and minimum)	NRPLH	Calculates all maximum values on the frequency axis to result the number of the maximum values.
	NRPLL	Calculates all maximum values on the frequency axis to result the number of the maximum values.
⑦ Checking the upper and lower limits	LMTMD1	Checks the upper and lower range limits with a reference value, reference width and specification.
	LMTMD2	Checks a level specified with a point value using a reference value, reference width and specification.
	LMTUL1	Checks the upper and lower range limits.
	LMTUL2	Checks the upper and lower range limits at a frequency position specified with a point value.
⑧ Trace data function (read/write)	RTRACE	Reads trace data at a specified point.
	WTRACE	Writes trace data at a specified point.
⑨ Graphics function	GADRS	Specifies the absolute address or viewport address of a graphic function.
	GSTR	Draws a character strings.
	GLINE	Draws a straight line between two specified points.
	GSTYLE	Specifies the length of a element drawn with a broken line dotted line or chain line.
	GPOINT	Dots a point at a specified position.
	GRECT	Draws a rectangular which has a diagonal between two specified points.
	GFLRECT	Paints over a rectangular which has a diagonal between two specified points.
	GMKR	Draws a marker (normal/delta) at a specified position.

(3) Parameter list

● built-in, trace

F	Frequency	W	watt (power)
P	Point (0 to 700)	Dx	Horizontal axis differential coefficient
L	Level	Dy	Vertical axis differential coefficient
T	Trace data (0 to 400)	S	Reference value
M	Trace A/B (0/1)	Ds	Reference value width
X	Loss level	Dd	Sample data
C	Check value (0,1,2,-1)	Lo	Lower limit value
N	Ripple number	Up	Upper limit value

```
F  = FREQ (P)                      T  = LVPOINT (L)
F  = DFREQ (P1, P2)                T  = LVDPOINT(L1, L2)
F  = FMAX (P1, P2, M)
F  = FMIN (P1, P2, M)              L  = VALUE (P, M)
F  = BND (P, X, M)                 L  = DVALUE (P1, P2, M)
F1 = BNDL (P, X, M)                L  = CVALUE (F1, F2, M)
Fh = BNDH (P, X, M)                L  = DCVALUE (F1, F2, M)
F  = CBND (F, X, M)                L  = LEVEL (T)
F1 = CBNDL (F, X, M)               L  = DLEVEL (T1, T2)
Fh = CBNDH (F, X, M)               L  = MAX (P1, P2, M)
F  = FRPLHN (N, M)                 L  = MIN (P1, P2, M)
F  = FRPLLN (N, M)                 L  = RPL1 (P1, P2, Dx, Dy, M)
                                   L  = VRPLHN (N, M)
P  = POINT (F)                     L  = VRPLLN (N, M)
P  = DPOINT (F1, F2)
P  = PMAX (P1, P2, M)              W  = POWER (P1, P2, M)
P  = PMIN (P1, P2, M)
P  = PRPLHN (N, M)                 N  = NRPLH (P1, P2, Dx, Dy, M)
P  = PRPLLN (N, M)                 N  = NRPLL (P1, P2, Dx, Dy, M)

C  = LMTMD1 (Dd, S, Ds)
C  = LMTMD2 (P, S, Ds, M)
C  = LMTUL1 (Dd, Up, Lo)
C  = LMTUL2 (P, Up, Lo, M)

T  = RTRACE (P, M)
WTRACE (T, P, M)       *Note: This function returns no value.
```

● Graphic

C	Character size	0 - 16×20 dot 1 - 18×24 dot
D	Set/Erase	0 - Erase 1 - Set (Draw)
MK	Marker	0 - Normal marker 1 - Δ marker
Mo	Address mode	0 - Absolute address 1 - Viewport address
S	Line style	0 - Solid line 1 - Broken line 2 - Dotted line 3 - Chain line
STR	Character expression	
X	Coordinate (Horizontal axis)	
Y	Coordinate (Vertical axis)	
dash	Dash part	
dot	Dot part	
space	Space part	

```
GADRS (Mo, X, Y)
GFLRECT (D, X1, Y1, X2, Y2)
GLINE (S, D, X1, Y1, X2, Y2)
GMKR (MK, D, X, Y)
GPOINT (D, X, Y)
GRECT (S, D, X1, Y1, X2, Y2)
GSTR (C, X, Y, STR)
CTYLE (dash, space, dot)
```

3.3 Built-in Functions

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

(1) BND (Frequency)
(2) BNDL (Frequency)
(3) BNDH (Frequency)
(4) CBND (Frequency)
(5) CBNDL (Frequency)
(6) CBNDH (Frequency)
(7) CVALUE (Level)
(8) DCVALUE (Level)
(9) DFREQ (Frequency)
(10) DLEVEL (Level)

(11) DPOINT (Point: Horizontal axis [0 to 700])
(12) DVALUE (Level)
(13) FMAX (Frequency)
(14) FMIN (Frequency)
(15) FREQ (Frequency)
(16) FRPLHN (Frequency)
(17) FRPLLN (Frequency)
(18) LEVEL (Level)
(19) LMTMD1 (Check value [0,1,2])
(20) LMTMD2 (Check value [0,1,2])

(21) LMTUL1 (Check value [0,1,2])
(22) LMTUL2 (Check value [0,1,2])
(23) LVDPOINT (Trace data: [0 to 400])
(24) LVPOINT (Trace data: [0 to 400])
(25) MAX (Level)
(26) MIN (Level)
(27) NRPLH (Maximum points number)
(28) NRPLL (Minimum points number)
(29) PMAX (Point: Horizontal axis [0 to 700])
(30) PMIN (Point: Horizontal axis [0 to 700])

(31) POINT (Point: Horizontal axis [0 to 700])
(32) POWER (Total power)
(33) PRPLHN (Point: Horizontal axis [0 to 700])
(34) PRPLLN (Point: Horizontal axis [0 to 700])
(35) RPL1 (Level)
(36) RTRACE (Trace data: [0 to 400])
(37) VALUE (Level)
(38) VRPLHN (Level)
(39) VRPLLN (Level)
(40) WTRCE (Trace data: [0 to 400])

(1) 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
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)
```

(2) 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
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), 3dB lower than the center frequency point of the trace B.

```
BL=BNDL (350, 3, 1)
```

(3) 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
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), 3dB higher than the center frequency point of the trace B.

```
BH=BNDH (350, 3, 1)
```

(4) 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
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)
```

(5) 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 of reference data
X:  LOSS level
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)
```

(6) 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 of reference data
X:  LOSS level
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 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 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)
```

(7) 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)
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 frequency point is out of range between the start frequency and the 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 level at 3.5MHz of the trace A.

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

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

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

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

【Reference】

(31) POINT
(37) VALUE

(8) 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)
F2:  Specified frequency 2 (Hz)
M :  Trace 0: Trace A
            1: Trace B
```

【Return value】

Normal termination: Level difference between the two specified frequencies (Unit is dB or mV)
Error interruption: Indefinite value

【Error】

- If the specified trace memory is other than the trace A/B or the specified frequencies 1 and 2 are out of range between the frequecy and stop frequecy, 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 level difference between 3MHz and 4MHz of the trace A.

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

L=DVALUE (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.)

【Reference】

(12) DVALUE (31) POINT

(9) 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.
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 betweeen the 300th point and 400th point.

```
FS=DFREQ (300,400)

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

(10) DLEVEL (Level)

【Feature】

Specify a trace data (vertical axis: [0 to 400]) span (T1, T2), and DLEVEL will calculate the corresponding level.

【Format】

DLEVEL (T1, T2)

T1: Specified trace data 1 (vertical axis: [0 to 400])
T2: Specified trace data 2 (vertical axis: [0 to 400])

【Return value】

Normal termination:	Level converted from the trace data (vertical axis: [0 to 400]) (Unit is dB or mV)
Error interruption:	Indefinite value

【Error】

- If the specified trace data are out of the range, 0 to 400, 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, convert T1 and T2.

【Example】

To calculate the level between trace data 200 and 300.

```
L=DLEVEL (200, 300)

L=DLEVEL (300, 200)
```

(11) 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)
F2:  Specified frequency 1 (Hz)
```

【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)
```

(12) 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 mV.)
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, convert P1 and P2.

【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.)

【Reference】

(8) DCVALUE   (31) POINT

(13) 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, convert P1 and P2.

【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.)
```

【Reference】

(31) POINT

(14) 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 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, convert P1 and P2.

【Example 】

To calculate the frequency at the minimum value position between 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.)

【Reference】

(31) POINT

(15) 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.
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)
```

(16) 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. (Before this function, execute the (27) NRPLH.)
- Return value will be indefinite.
- An error causes only an error message and not terminates the program.

【Note】

Before this function, execute the (27) 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)
```

(17) 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
Trace 1: Trace B

【Return value】

Normal termination : The frequency (Hz) 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. (Before this function, execute the (28) NRPLL.)
- Return value will be indefinite.
- An error causes only an error message and not terminates the program.

【Note】

Before this function, execute the (28) 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, 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=FRPLL
```

(18) LEVEL (Level)

【Feature 】

Specify a trace data (vertical axis [0 to 400]), and LEVEL will calculate the corresponding level.

【Format】

LEVEL(T)
T: Specified trace data (Vertical axis: [0 to 400])

【Return value】

Normal termination:	Level converted from the trace data (vertical axis: [0 to 400]) (The same unit as the reference level)
Error interruption:	Indefinite value

【Error 】

- If the specified trace data are out of the range, 0 to 400, 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
```

【Reference】

(36) RTRACE

(19) LMTMD1 (Chec Value [0, 1, 2])

【Feature 】

Specify a raference value, its upper and lower limits and sample data. LMTMD1 will check if the sample data are inside or outside of the range between the upper and lower limits.

【Format】

LMTMD1 (Dd, S, Ds)

Dd: Sample data
S : Reference value
Ds: Upper and lower limits

((S-Ds) ≦ Dd ≦ (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 "STANDARD", 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"
```

(20) LMTMD2 (Check Value [0, 1, 2])

【Feature】

Specify a reference value, its upper and lower limits, the trace A/B data and a point. LMTMD2 will check the point level is inside or outside of the range between the upper and lower limits.

【Format】

LMTMD2 (P, S, Ds, M)

```
P : Point (0 to 700)
S : Reference value
Ds: Upper and lower limits
M : Trace 0: Trace A
          1: Trace B
```

((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 +10 and -10 of the input reference value.

```
INPUT "STANDARD", S
P=POINT (30E6)
T=LMTMD2 (P, S, 20, 0)
IF T=0 THEN PRINT "OK"
IF T=1 THEN PRINT "UPPER"
IF T=2 THEN PRINT "LOWER"
```

(21) 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

(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"
```

(22) LMTUL2 (Check Value [0, 1, 2])

【Feature 】

Specify an upper and lower limits, the trace A/B data and a point. LMTUL2 will check the point level is inside or outside of the range between the upper and lower limits.

【Format】

LMTUL2 (P, Up, Lo, M)

```
P :  Point (0 to 700)
Up:  Upper limit
Lo:  Lower limit
M :  Trace 0:  Trace A
           1:  Trace B
(Lo ≦ level ≦ 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, 0)
  IF T=0 THEN PRINT "OK"
  IF T=1 THEN PRINT "UPPER"
  IF T=2 THEN PRINT "LOWER"
```

(23) LVDPOINT (Trace data: [0 to 400])

【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
(vertical axis [0 to 400])
Error interruption : Indefinite value

【Error】

- Return value will be indefinite.
- An error causes only an error message and not terminates the program.
- The specified levels 1 and 2 should be the optimum value between the scales of the measuring instrument.

【Note】

If L1>L2, convert L1 and L2.

【Example】

To calculate the trace data between -20 and -30dB.

```
TD=LVDPOINT (-20, -30)
```

To calculate the trace data between -20 and -50dB.

```
L=-50
TD=LVDPOINT (-20, L)
```

(24) LVPOINT (Trace data: [0 to 400])

【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
                     (vertical axis [0 to 400] )
Error interruption : Indefinite value
```

【Error 】

- Return value will be indefinite.
- An error causes only an error message and not terminates the program.
- The specified levels 1 and 2 should be the optimum value between the scales of the measuring instrument.

【Example 】

To calculate the trace data at -10dB.

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

To calculate the trace data at -20dB.

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

(25) MAX (Level)

【Feature】

Specify a measuring point range (point 1 and 2) and the trace A/B. MAX will calculate the maximum 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, convert P1 and P2.

【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.)

(26) MIN (Level)

【Feature】

Specify a measuring point range (piont 1 and 2) and the trace A/B.
MIN will calculate the minimum 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, convert P1 and P2.

【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.

```
F1=10E6
P1=POINT (F1)
ML1=MIN (POINT (20E6), P1, 1)
ML2=MIN (POINT (20E6), POINT (F1), 1)

    (ML1 and ML2 are equal.)
```

(27) NRPLH (Number of the Maximum)

【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 (0 to 700)
Dy: Differential coefficient trace data (0 to 400)
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: 256)
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 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)
```

(28) NRPLL (Number of the Minimum)

【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 (0 to 700)
Dy:  Differential coefficient trace data (0 to 400)
M :  Trace 0:  Trace A
           1:  Trace B
```

【Return value】

Normal termination:	Number of the minimum values corresponding with the frequency axis (horizontal axis: from the left) (minimum number: 256)
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 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)
```

(29) 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 (horizonal : 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, convert P1 and P2.

【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.)

(30) 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 (horizonal 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, convert P1 and P2.

【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 of the trace B.

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

(MP1 and MP2 are equal.)

(31) 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)

【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)
```

(32) POWER (Total Power)

【Feature】

Calculate the total power when the REF level is set to 0 dBm.
The frequency range depends on the point width (points 1 to 2).

【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) * $10^{\frac{x}{10}}$ (x dBm)

【Example】

To calculate the total power between point 300 and 400 of the trace A.
(When REF level=0 dBm)

```
AW=POWER (300, 400, 0)
```

To calculate the total power between point 0 and 700 of the trace A.
(When REF level=-20dBm)

```
AW = POWER (0,700,0)/100
```

(33) 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. (Before this function, execute the (27) NRPLH.)
- Return value will be indefinite.
- An error causes only an error message and not terminates the program.

【Note】

Before this function, execute the (27) 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)
```

(34) 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 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 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. (Before this function, execute the (28) NRPLL.)
- Return value will be indefinite.
- An error causes only an error message and not terminates the program.

【Note】

Before this function, execute the (28) 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)
```

(35) RPL1 (Level)

【Feature 】

Specify a measuring point range (point 1 and 2), trace A/B and a differntial 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 (0 to 700)
Dy: Differential coefficient trace data (0 to 400)
M : Trace 0: Trace A
1: Trace B

【Return value】

Normal termination: Level difference between the biggest maximum and smallest minimum values.
Error interruption: Indefinite value

【Error 】

- If the specified point 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, convert P1 and P2.

【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)
```

(36) RTRACE (Trace data: [0 to 400])

【Feature】

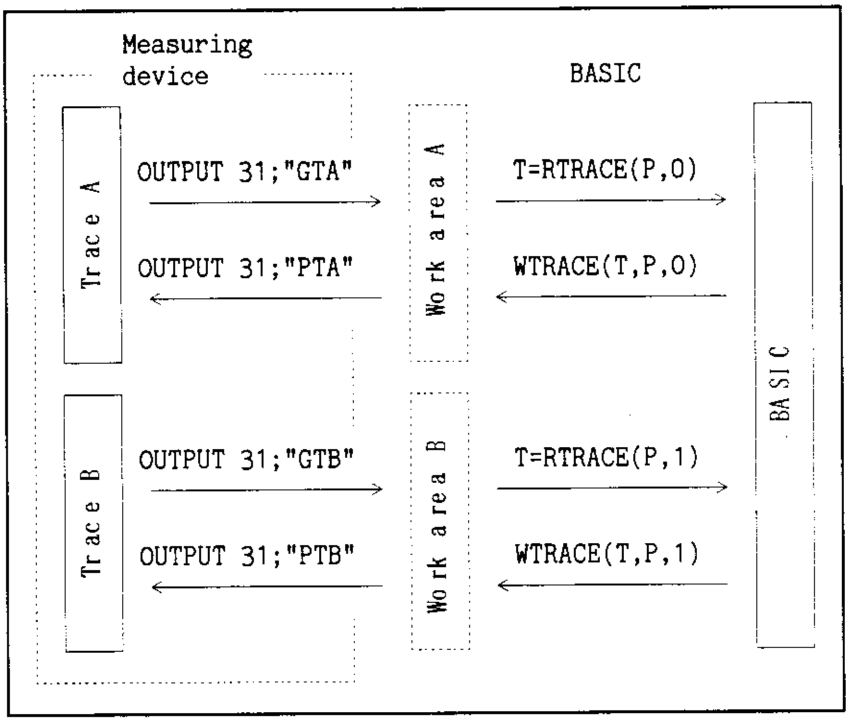
RTRACE returns the trace data of a specified point.

【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 400)
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
```

【Data】

The above sample program executes in about 2.5s.
The running time per data point is about 3.5ms.

(37) 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.)

【Reference】

(7) CVALUE
(31) POINT

(38) 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 maximum position is detected, an error will result. (Before this function, execute the (27) NRPLH.)
- Return value will be indefinite.

- An error causes only an error message and not terminates the program.

【Note】

Before this function, execute the (27) 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=VRPLHN (2, TS)
```

(39) 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. (Before this function, execute the (28) NRPLH.)
- Return value will be indefinite.
- An error causes only an error message and not terminates the program.

【Note】

Before this function, execute the (28) 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)
```

(40) WTRACE

【Feature】

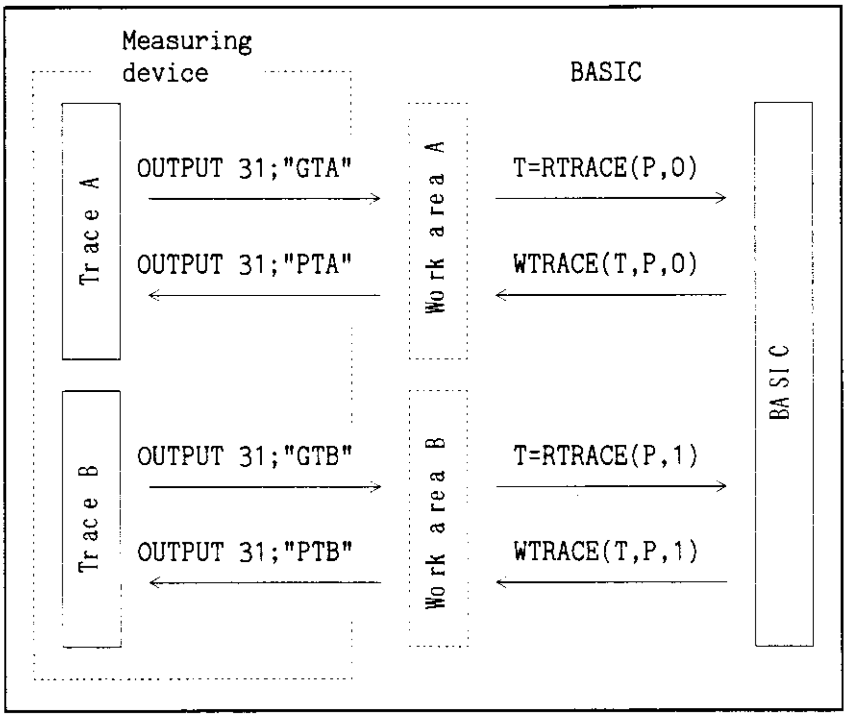
WTRACE writes the trace data into a specified point.

【Format】

WTRACE (T, P, M)

T: Trace data (0 to 400)
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"
```

【Data】

The above sample program executes in about 2.2s.
The running time per data point is about 3.1ms.

3.4 Graphic Function

The graphic function will be described as the following order.

(1) GADRS

(2) GFLRECT

(3) GLINE

(4) GMKR

(5) GPOINT

(6) GRECT

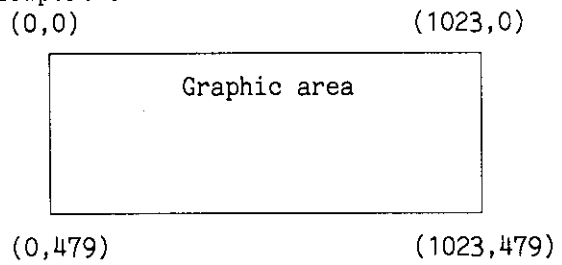
(7) GSTR

(8) GSTYLE

(1) GADRS

【Feature】

GADRS specify the address mode of the graphic display.
Two address modes are available.
0: Absolute address
1: Viewport address



The origin is located at the upper left by specifying the absolute address mode. In this mode, the X and Y coordinates specification will be ignored. The origin is the X and Y when the viewpoint address is specified.

Note: To specify the X and Y by absolute address and not to deviate from the above area. When a viewpoint address is specified, the upper right of theorigin is the first quadrant.

【Format】

```
GADRS(Mo, X, Y)

Mo:  mode 0: Absolute address
          1: View point address
X :  Horizontal axis (0 to 1023)
Y :  Vertical axis (0 to 479)
```

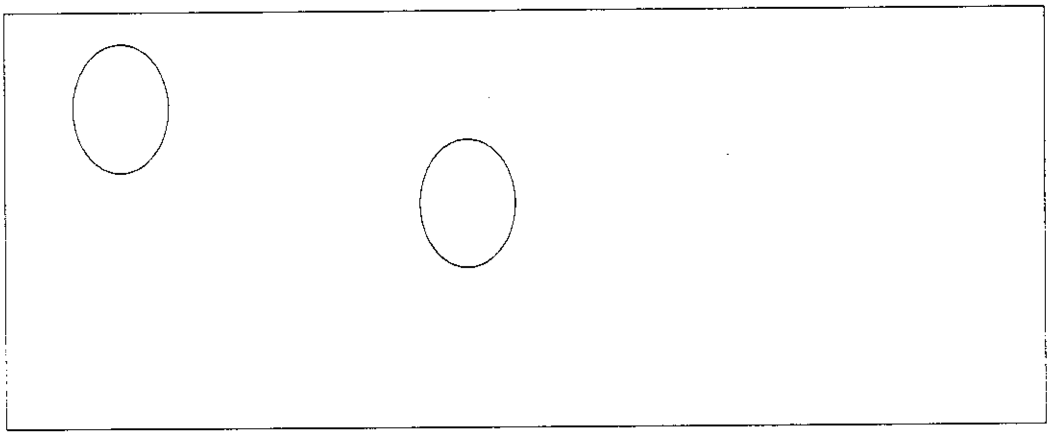
【Error】

- An error will arise when the XY is set out of the graphic display area.
- Though the error outputs a message, does not interrupt a program execution.

【Example】

```
OUTPUT 31;"VS3"
CLS 1
R=100
FOR M=0 TO 1
  GADRS(M,512,240)
  FOR I=0 TO R*2 STEP R/90
    X=SIN(I)*R
    Y=COS(I)*R
    GPOINT(1,X,Y)
  NEXT I
NEXT M
```

【Result】



(2) 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: Erase |      └─ coordinate2
            1: Draw  └──────── coordinate1
```

【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.

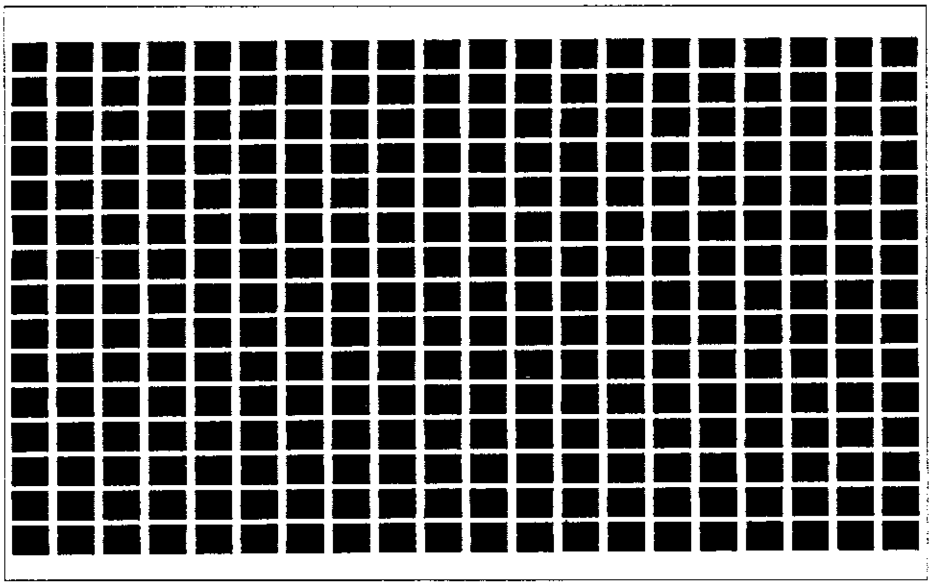
【Note】

Coordinate 1 must be specified at the upperleft, and coodinate 2 at the upper right.

【Example 】

```
OUTPUT 31;"VS3"
CLS 1
INTEGER X,Y
FOR X=10 TO 970 STEP 50
  FOR Y=10 TO 450 STEP 30
    GFLRECT(1,X,Y,X+40,Y+25)
  NEXT Y
NEXT X
STOP
```

【Result】



(3) GLINE

【Feature 】

GLINE draw a line between specified two points (coordinate 1, coordinate 2).

```
【Format】  GLINE (S,   D,   X1,   Y2,   X2,   Y2)
                  |     |    ===========   ===========
                  |     |          |             └─ coordinate 2
                  |     └─ 0: Erase└─────────────── coordinate 1
                  |        1: Draw
                  └──Style 0: Solid line
                           1: Broken line
                           2: Dotted line
                           3: Chain line
```

【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.

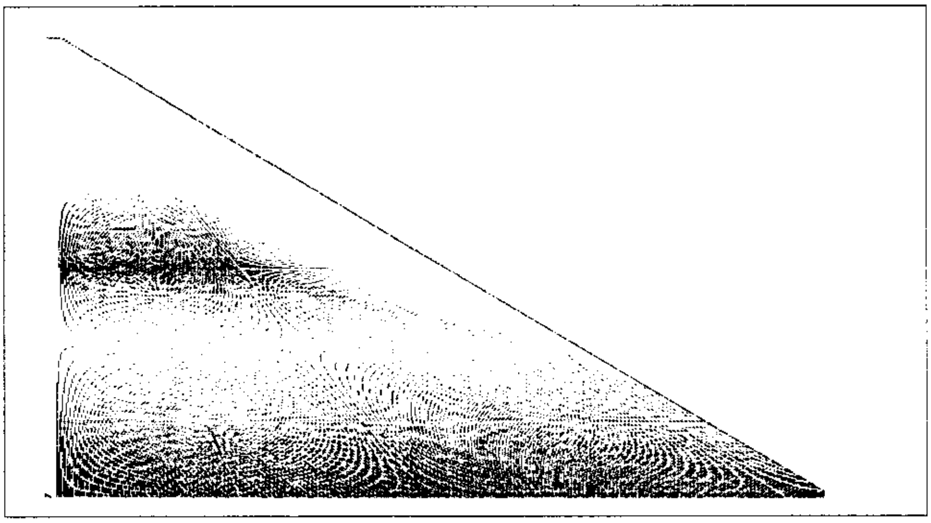
【Note】

The style must be specified previously by the line style (GSTYLE function) except for a solid line drawing. (Refer to the GSTYLE function.)

【Example 】

```
OUTPUT 31;"VS3"
CLS 1
FOR I=2 TO 1023 STEP 3
  GLINE(0,1,2,0,I,479)
  GLINE(0,1,0,0,I-2,479)
NEXT I
STOP
```

【Result】



(4) GMKR

【Feature】

GMKR draw a marker (normal, △) at a specified position.

【Format】

```
GMKR (MK,   D,   X,   Y)
      |     |    ‾‾‾‾‾‾‾
      |     |       └─ coordinate
      |     └───────── 0: Erase
      |                1: Draw
      └─────────────── Marker type 0: Normal marker
                                   1: △ marker
```

【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.

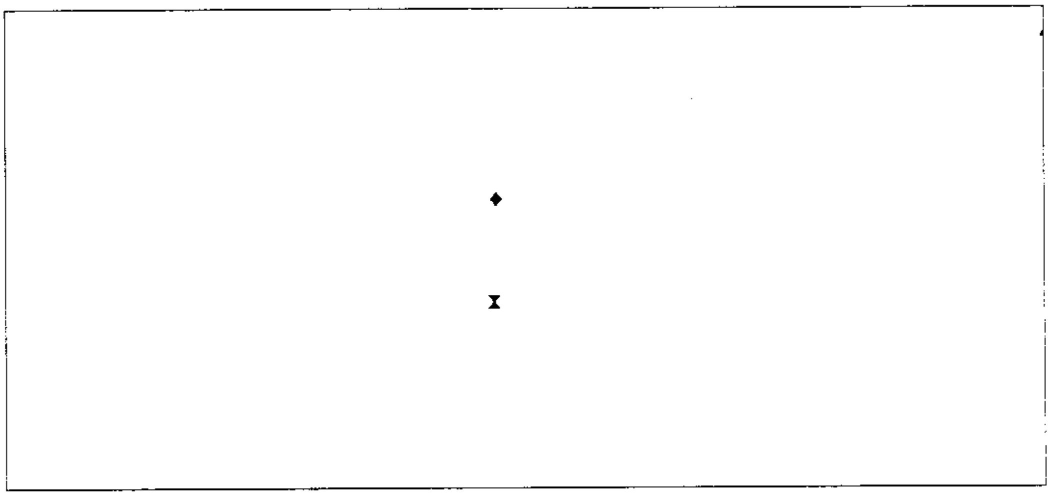
【Note】

- The marker size is 11× 11 dot.
- A marker is drawn being its center at a specified position.

【Example】

```
OUTPUT 31;"VS3"
CLS 1
GMKR(0,1,512,200)
GMKR(1,1,512,280)
```

【Result】



(5) GPOINT

【Feature】

GPOINT dots a point at a specified position.

【Format】

```
GPOINT (D,  X,   Y)
         |   └─ coordinate
         └──────── 0: Erase
                   1: Draw
```

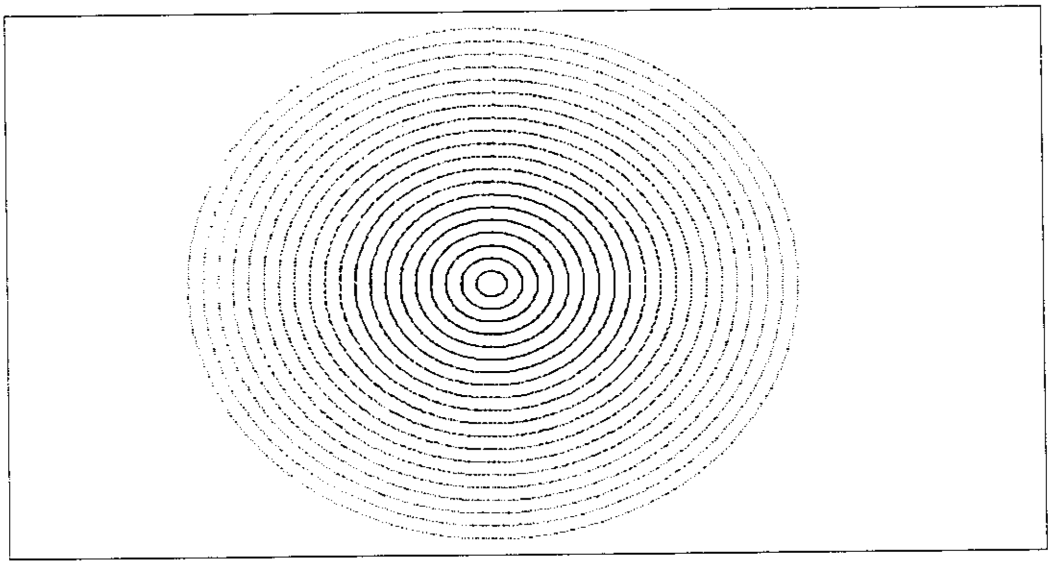
【Error】

- 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】

```
OUTPUT 31;"VS3"
CLS 1
FOR R=200 TO 1 STEP -10
  FOR I=200 TO R*2 STEP R/270
    X=SIN(I)*R*1.5+512
    Y=COS(I)*R+240
    GPOINT(1,X,Y)
  NEXT I
NEXT R
```

【Result】



(6) GRECT

【Feature】

GRECT draws a rectangular whose diagonal is across specified two points (coordinate 1, coordinate 2).

```
【Format】 GRECT (S,   D,   X1,   Y1,   X2,   Y2)

                                            coorcinate 2
           Style 0: Solid line              coordinate 1
                 1: Dash line               0: Erase
                 2: Broken line             1: Draw
                 3: Dotted line
                 4: Chain line
```

【Error】

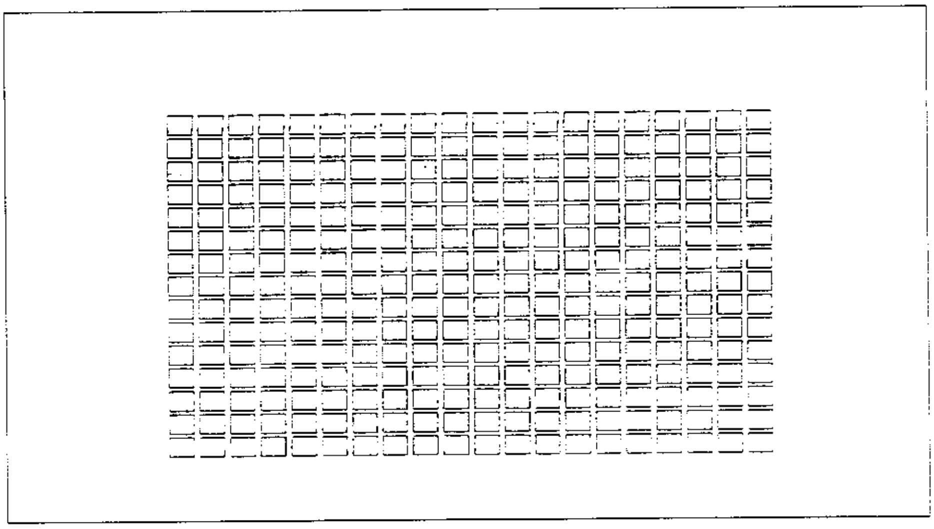
- An error will arise, if the X and Y are specified out of the graphic display area. (Take care when being in the viewport address mode.)
- Though the error outputs a message, does not interrupt a program execution.

【Note】 The style must be specified previously by the line style (GSTYLE function)except for a solid line drawing. (Refer to the GSTYLE function.)

【Example】

```
OUTPUT 31;"VS3"
CLS 1
INTEGER X,Y
FOR X=10 TO 970 STEP 50
  FOR Y=10 TO 450 STEP 30
    GRECT(0,1,X,Y,X+40,Y+25)
  NEXT Y
NEXT X
STOP
```

【Result】



(7) GSTR

【Feature】

GSTR draws a character at a specified position on the graphic display.

【Format】

GSTR (C, X, Y, STR)

C : Character size 0: 16 × 20 dot, 1: 18 × 24 dot
X : Horizontal axis (absolute address: 0 to 1023)
Y : Vertical axis (absolute address: 0 to 479)
STR: Character expression

【Error】

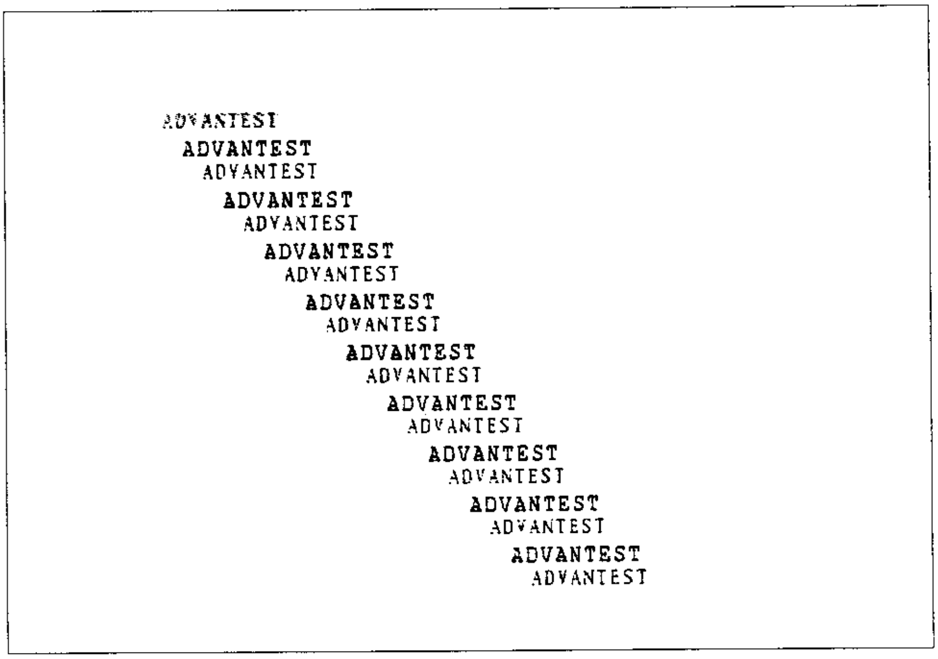
- An error will arise, if the X and Y are specified out of the graphic display area.
  (Take care when being in the viewport address mode.)
- Though the error outputs a message, does not interrupt a program execution.

【Note】 Output characters are available up to 18. Characters after the 18th will be ignored.

【Example】

```
OUTPUT 31;"VS3"
S$="ADVANTEST"
FOR I=0 TO 450 STEP 25
  GSTR(I%2,I,I,S$)
NEXT I
STOP
```

【Result】



```
ADVANTEST
 ADVANTEST
  ADVANTEST
   ADVANTEST
    ADVANTEST
     ADVANTEST
      ADVANTEST
       ADVANTEST
        ADVANTEST
         ADVANTEST
          ADVANTEST
           ADVANTEST
            ADVANTEST
             ADVANTEST
              ADVANTEST
               ADVANTEST
                ADVANTEST
                 ADVANTEST
                  ADVANTEST
```

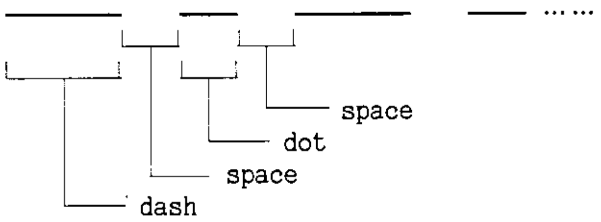
(8) GSTYLE

【Feature】

GSTYLE specifies the component length of a broken line, dotted line and chain line.

【Format】

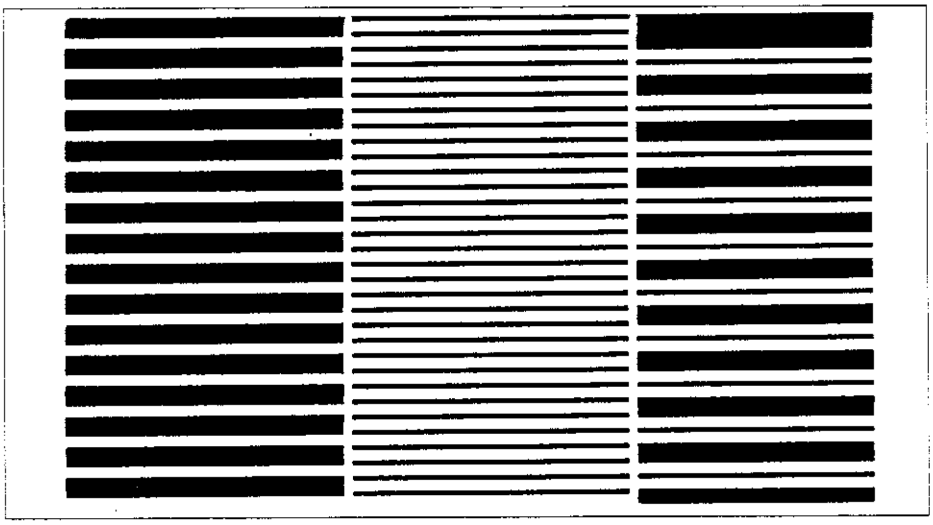
GSTYLE (dash, space, dot)
This function specifies component length of a broken line, dotted line and chain line. Components means a dash, space and dot parts.
These components are combined as the follows.
  Broken line: dash+space
  Dotted line: dot+space
  Chain line : dash+space+dot+space
    (Example) To specify a chain line as follows.



【Example】

```
OUTPUT 31;"VS3"
CLS 1
S=0
GSTYLE(20,10,5)
FOR X=1 to 1000
  IF X % 350 = 1 THEN ++S : X+=10
  GLINE(S,1,X,0,X,479)
NEXT X
STOP
```

【Result】



APPENDIX

A.1 List of commands and statements

Function	Command and Statement	Description
① Commands	CONT CONTROL LIST LLIST LISTN LLISTN RUN SCRATCH STEP	Resumes the execution of the program after it has stopped. Sets values for each control. Displays a program list. Displays a program list. (RS-232C) Displays a program list. Displays a program list. (RS-232C) Executes a program. Deletes a program in BASIC. Executes a line of a program.
② Arithmetic functions	ABS ATN COS LOG SIN SQR TAN	Produces the absolute value of the given value. Produces the arc tangent of the given value. Produces the cosine of the given value. Produces the natural logarithm of the given value. Produces the sine of the given value. Produces the square root of the given value. Produces the tangent of the given value.
③ Bitwise operations	BAND BNOT BOR BXOR	Produces a bit AND. Produces a bit NOT. Produces a bit OR. Produces a bit XOR.
④ Interrupt controls	ENABLE INTR DISABLE INTR ON END ON KEY ON ISRQ ON SRQ ON ERROR OFF END OFF KEY OFF ISRQ OFF SRQ OFF ERROR	Enables interrupts to be received. Disables interrupts to be received. Defines the branch of an EOF interrupt. Defines the branch of a key interrupt. Defines the branch of an SRQ interrupt of the main measurement section. Defines the branch of an SRQ interrupt of GPIB. Defines the branch of an error occurrence interrupt. Releases the branch of an EOF interrupt. Releases the branch of a key interrupt. Releases the branch of an SRQ interrupt of the main measurement section. Releases the branch of an SRQ interrupt of GPIB. Releases the branch of an error occurrence interrupt.

Note: See section 2.3 for details of each command and statement.

(cont'd)

Function	Command and Statement	Description
⑤ Character-string manipulations	NUM CHR$ LEN POS SPRINTF	Converts the first character of a character string to the ASCII code. Converts a numeric character to an ASCII character. Obtains the length of a character string. Positions character string 1 in character string 2. Formats a string and assigns it to a character-string variable.
⑥ Memory card controls	CAT CLOSE # ENTER # OPEN # OUTPUT # INITIALIZE (or INIT) PURGE RENAME	Displays the contents of a memory card. Close a file. Reads data from a file. Open a file. Writes data into a file. Initializes a memory card. Deletes the specified file. Rename a file.
⑦ Screen controls	CURSOR(or CSR) CLS	Moves the cursor to the specified position. Clears the screen.
⑧ Statements	BUZZER DIM FOR TO STEP NEXT BREAK CONTINUE GOSUB RETURN GOTO	Sounds the buzzer. Declares array variables. Sets an iteration. Exits from the current iteration. Returns to the beginning of the current iteration. Branches to a subroutine. Returns from the current subroutine. Branches to the specified statement.

Note: See section 2.3 for details of each command and statement.

(cont'd)

Function	Command and Statement	Description
⑧ Statements (cont'd)	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.
	LPRINT	Outputs data on a printer (RS-232C).
	LPRINT USING (or USE	Outputs formatted data on a printer (RS-232C).
	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 scree.
	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 periond.
⑨ 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.
	GLISTN	Ouuputs the program list to a GPIB printer.
	GPRINT	Ouputs data to a GPIB printer.
	GPRINT USING (or USE)	Outputs formatted data to a GPIB printer.
	INTERFACE CLEAR	Transfers IFC.
	LOCAL	Places the specified device in the local status.
	LOCAL LOCKOUT	Places the specified device in the local lokced-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.

Note: See section 2.3 for details of each command and statement.

A. 2 List of built-in function and graphic function

function	structure	describe
① Calculating a frequency	FREQ	Calculates a frequency from a point value.
	DFREQ	Calculates a frequecny from a width between points.
	FMAX	Calculates the frequency at the maximum level position between two positions specified with point value.
	FMIN	Calculates the frequency at the minimum level position between two positions specified with point value.
	BND	Calculates the frequency bandwidth of a LOSS level at a position specified with point values.
	BNDL	Calculates the low frequency bandwidth of a LOSS level at a position specified with point values.
	BNDH	Calculates the high frequency bandwidth of a LOSS level at a position specified with point values.
	CBND	Calculates the frequency bandwidth of a LOSS level at a position specified with a frequency.
	CBNDL	Calculates the low frequency bandwidth of a LOSS level at a position specified with a frequency.
	CBNDH	Calculates the high frequency bandwidth of a LOSS level at a position specified with a frequency.
	FRPLHN	Calculates the frequency at the maximum point n-th from the left on the frequency axis.
	FRPLLN	Calculates the frequency at the minimum point n-th from the left on the frequency axis.
② Calculating point (Frequency point : Horizontal axis)	POINT	Calculates a point (horizontal axis).
	DPOINT	Calculates a point between specified frequencies.
	PMAX	Calculates the frequency point (horizontal axis) maximum level position between two positions at the specified with point values.
	PMIN	Calculates the frequency point (horizontal axis) at the minimum level position between two positions specified with point values.
	PRPLHN	Calculates the point at the maximum point n-th from the left on the frequency axis.
	PRPLLN	Calculates the point at the minimum point n-th from the left on the frequency axis.

(cont'd)

function	structure	describe
④ Calculating level	VALUE	Calculates a level at a frequency position specified with a point value.
	DVALUE	Calculates a difference of levels at two frequency positions specified with point values.
	CVALUE	Calculates a level at a frequency position.
	DCVALUE	Calculates the level difference between two specified frequency positions.
	LEVEL	Calculates a level of trace data specified with a point value (vertical axis).
	DLEVEL	Calculates a level between points (vertical axis).
	MAX	Calculates the maximum level between two positions specified with point values.
	MIN	Calculates the minimum level between two positions specified with point values.
	RPL1	Calculates the level difference between the maximum value at the miximum point and minimum value at the minimum point.
	VRPLHN	Calculates the Level at the maximum point n-th from the left on the frequency axis.
	VRPLLN	Calculates the Level at the minimum point n-th from the left on the frequency axis.
⑤ Calculating the total power	POWER	Calculates total power between two positions specified with point values.
⑥ Calculating ripple number (maximum and minimum)	NRPLH	Calculates all maximum values on the frequency axis to result the number of the maximum values.
	NRPLL	Calculates all maximum values on the frequency axis to result the number of the maximum values.
⑦ Checking the upper and lower limits	LMTMD1	Checks the upper and lower range limits with a reference value, reference width and specification.
	LMTMD2	Checks a level specified with a point value using a reference value, reference width and specification.
	LMTUL1	Checks the upper and lower range limits.
	LMTUL2	Checks the upper and lower range limits at a frequency position specified with a point value.

(cont'd)

function	structure	describe
⑧ Trace data function (read/write)	RTRACE WTRACE	Reads trace data at a specified point. Writes trace data at a specified point.
⑨ Graphics function	GADRS GSTR GLINE GSTYLE GPOINT GRECT GFLRECT GMKR	Specifies the absolute address or viewport address of a graphic function. Draws a character strings. Draws a straight line between two specified points. Specifies the length of a element drawn with a broken line dotted line or chain line. Dots a point at a specified position. Draws a rectangular which has a diagonal between two specified points. Paints over a rectangular which has a diagonal between two specified points. Draws a marker (normal/delta) at a specified position.

A. 3 List of parameter

(1) built-in, trace

F	Frequency	W	watt (power)
P	Point (0 to 700)	Dx	Horizontal axis differential coefficient
L	Level	Dy	Vertical axis differential coefficient
T	Trace data (0 to 400)	S	Reference value
M	Trace A/B (0/1)	Ds	Reference value width
X	Loss level	Dd	Sample data
C	Check value (0,1,2,-1)	Lo	Lower limit value
N	Ripple number	Up	Upper limit value

```
F  = FREQ (P)                       T  = LVPOINT (L)
F  = DFREQ (P1, P2)                 T  = LVDPOINT(L1, L2)
F  = FMAX (P1, P2, M)
F  = FMIN (P1, P2, M)               L  = VALUE (P, M)
F  = BND (P, X, M)                  L  = DVALUE (P1, P2, M)
F1 = BNDL (P, X, M)                 L  = CVALUE (F1, F2, M)
Fh = BNDH (P, X, M)                 L  = DCVALUE (F1, F2, M)
F  = CBND (F, X, M)                 L  = LEVEL (T)
F1 = CBNDL (F, X, M)                L  = DLEVEL (T1, T2)
Fh = CBNDH (F, X, M)                L  = MAX (P1, P2, M)
F  = FRPLHN (N, M)                  L  = MIN (P1, P2, M)
F  = FRPLLN (N, M)                  L  = RPL1 (P1, P2, Dx, Dy, M)
                                    L  = VRPLHN (N, M)
P  = POINT (F)                      L  = VRPLLN (N, M)
P  = DPOINT (F1, F2)
P  = PMAX (P1, P2, M)               W  = POWER (P1, P2, M)
P  = PMIN (P1, P2, M)
P  = PRPLHN (N, M)                  N  = NRPLH (P1, P2, Dx, Dy, M)
P  = PRPLLN (N, M)                  N  = NRPLL (P1, P2, Dx, Dy, M)

C  = LMTMD1 (Dd, S, Ds)
C  = LMTMD2 (P, S, Ds, M)
C  = LMTUL1 (Dd, Up, Lo)
C  = LMTUL2 (P, Up, Lo, M)

T  = RTRACE (P, M)
WTRACE (T, P, M)      *Note: This function returns no value.
```

(2) Graphic

C	:	Character size	0 - 16× 20 dot 1 - 18× 24 dot
D	:	Set/Erase	0 - Erase 1 - Set (Draw)
MK	:	Marker	0 - Normal marker 1 - Δ marker
Mo	:	Address mode	0 - Absolute address 1 - Viewport address
S	:	Line style	0 - Solid line 1 - Broken line 2 - Dotted line 3 - Chain line
STR	:	Character expression	
X	:	Coordinate (Horizontal axis)	
Y	:	Coordinate (Vertical axis)	
dash	:	Dash part	
dot	:	Dot part	
space	:	Space part	

```
GADRS (Mo, X, Y)
GFLRECT (D, X1, Y1, X2, Y2)
GLINE (S, D, X1, Y1, X2, Y2)
GMKR (MK, D, X, Y)
GPOINT (D, X, Y)
GRECT (S, D, X1, Y1, X2, Y2)
GSTR (C, X, Y, STR)
CTYLE (dash, space, dot)
```

A. 4 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 sprit 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 biiger 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 stataement.
FOR variable does NOT exist.	No counter variable exists in a FOR statement.
FOR's nest 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 decraration. (Up to 128 characters)
Substring error	Invalid substring specification.

(cont'd)

Error message	Description
Unbalanced BREAK	No BREAKE statement between FOR and NEXT statements.
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.

(cont'd)

Error message	Description
xxx: invalid source type in xxx	Source type is invalid to assign into an expression.
xxx: invalid target type in xxx	Variable type is invalid to assign.
xxx: Invalid TERGET 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.

MEMO

INDEX (PARTⅡ)

IMPORTANT INFORMATION FOR ADVANTEST SOFTWARE

PLEASE READ CAREFULLY: This is an important notice for the software defined herein. Computer programs including any additions, modifications and updates thereof, operation manuals, and related materials provided by Advantest (hereafter referred to as "SOFTWARE"), included in or used with hardware produced by Advantest (hereafter referred to as "PRODUCTS").

SOFTWARE License

All rights in and to the SOFTWARE (including, but not limited to, copyright) shall be and remain vested in Advantest. Advantest hereby grants you a license to use the SOFTWARE only on or with Advantest PRODUCTS.

Restrictions

(1) You may not use the SOFTWARE for any purpose other than for the use of the PRODUCTS.

(2) You may not copy, modify, or change, all or any part of, the SOFTWARE without permission from Advantest.

(3) You may not reverse engineer, de-compile, or disassemble, all or any part of, the SOFTWARE.

Liability

Advantest shall have no liability (1) for any PRODUCT failures, which may arise out of any misuse (misuse is deemed to be use of the SOFTWARE for purposes other than it's intended use) of the SOFTWARE. (2) For any dispute between you and any third party for any reason whatsoever including, but not limited to, infringement of intellectual property rights.

LIMITED WARRANTY

1. Unless otherwise specifically agreed by Seller and Purchaser in writing, Advantest will warrant to the Purchaser that during the Warranty Period this Product (other than consumables included in the Product) will be free from defects in material and workmanship and shall conform to the specifications set forth in this Operation Manual.

2. The warranty period for the Product (the "Warranty Period") will be a period of one year commencing on the delivery date of the Product.

3. If the Product is found to be defective during the Warranty Period, Advantest will, at its option and in its sole and absolute discretion, either (a) repair the defective Product or part or component thereof or (b) replace the defective Product or part or component thereof, in either case at Advantest's sole cost and expense.

4. This limited warranty will not apply to defects or damage to the Product or any part or component thereof resulting from any of the following:

   (a) any modifications, maintenance or repairs other than modifications, maintenance or repairs (i) performed by Advantest or (ii) specifically recommended or authorized by Advantest and performed in accordance with Advantest 's instructions;

   (b) any improper or inadequate handling, carriage or storage of the Product by the Purchaser or any third party (other than Advantest or its agents);

   (c) use of the Product under operating conditions or environments different than those specified in the Operation Manual or recommended by Advantest, including, without limitation, (i) instances where the Product has been subjected to physical stress or electrical voltage exceeding the permissible range and (ii) instances where the corrosion of electrical circuits or other deterioration was accelerated by exposure to corrosive gases or dusty environments;

   (d) use of the Product in connection with software, interfaces, products or parts other than software, interfaces, products or parts supplied or recommended by Advantest;

   (e) incorporation in the Product of any parts or components (i) provided by Purchaser or (ii) provided by a third party at the request or direction of Purchaser or due to specifications or designs supplied by Purchaser (including, without limitation, any degradation in performance of such parts or components);

   (f) Advantest's incorporation or use of any specifications or designs supplied by Purchaser;

   (g) the occurrence of an event of force majeure, including, without limitation, fire, explosion, geological change, storm, flood, earthquake, tidal wave, lightning or act of war; or

   (h) any negligent act or omission of the Purchaser or any third party other than Advantest.

5. **EXCEPT TO THE EXTENT EXPRESSLY PROVIDED HEREIN, ADVANTEST HEREBY EXPRESSLY DISCLAIMS, AND THE PURCHASER HEREBY WAIVES, ALL WARRANTIES, WHETHER EXPRESS OR IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, (A) ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE AND (B) ANY WARRANTY OR REPRESENTATION AS TO THE VALIDITY, SCOPE, EFFECTIVENESS OR USEFULNESS OF ANY TECHNOLOGY OR ANY INVENTION.**

6. **THE REMEDY SET FORTH HEREIN SHALL BE THE SOLE AND EXCLUSIVE REMEDY OF THE PURCHASER FOR BREACH OF WARRANTY WITH RESPECT TO THE PRODUCT.**

7. **ADVANTEST WILL NOT HAVE ANY LIABILITY TO THE PURCHASER FOR ANY INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL OR PUNITIVE DAMAGES, INCLUDING, WITHOUT LIMITATION, LOSS OF ANTICIPATED PROFITS OR REVENUES, IN ANY AND ALL CIRCUMSTANCES, EVEN IF ADVANTEST HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES AND WHETHER ARISING OUT OF BREACH OF CONTRACT, WARRANTY, TORT (INCLUDING, WITHOUT LIMITATION, NEGLIGENCE), STRICT LIABILITY, INDEMNITY, CONTRIBUTION OR OTHERWISE. TORT (INCLUDING, WITHOUT LIMITATION, NEGLIGENCE), STRICT LIABILITY, INDEMNITY, CONTRIBUTION OR OTHERWISE.**

8. **OTHER THAN THE REMEDY FOR THE BREACH OF WARRANTY SET FORTH HEREIN, ADVANTEST SHALL NOT BE LIABLE FOR, AND HEREBY DISCLAIMS TO THE FULLEST EXTENT PERMITTED BY LAW ANY LIABILITY FOR, DAMAGES FOR PRODUCT FAILURE OR DEFECT, WHETHER ARISING OUT OF BREACH OF CONTRACT, TORT (INCLUDING, WITHOUT LIMITATION, NEGLEGENCE), STRICT LIABILITY, INDEMNITY, CONTRIBUTION OR OTHERWISE.**

CUSTOMER SERVICE DESCRIPTION

In order to maintain safe and trouble-free operation of the Product and to prevent the incurrence of unnecessary costs and expenses, Advantest recommends a regular preventive maintenance program under its maintenance agreement.

Advantest's maintenance agreement provides the Purchaser on-site and off-site maintenance, parts, maintenance machinery, regular inspections, and telephone support and will last a maximum of ten years from the date the delivery of the Product. For specific details of the services provided under the maintenance agreement, please contact the nearest Advantest office listed at the end of this Operation Manual or Advantest 's sales representatives.

Some of the components and parts of this Product have a limited operating life (such as, electrical and mechanical parts, fan motors, unit power supply, etc.). Accordingly, these components and parts will have to be replaced on a periodic basis. If the operating life of a component or part has expired and such component or part has not been replaced, there is a possibility that the Product will not perform properly. Additionally, if the operating life of a component or part has expired and continued use of such component or part damages the Product, the Product may not be repairable. Please contact the nearest Advantest office listed at the end of this Operation Manual or Advantest's sales representatives to determine the operating life of a specific component or part, as the operating life may vary depending on various factors such as operating condition and usage environment.

SALES & SUPPORT OFFICES

Advantest Korea Co., Ltd.
22BF, Kyobo KangNam Tower,
1303-22, Seocho-Dong, Seocho-Ku, Seoul #137-070, Korea
Phone: +82-2-532-7071
Fax: +82-2-532-7132

Advantest (Suzhou) Co., Ltd.
Shanghai Branch Office:
Bldg. 6D, NO.1188 Gumei Road, Shanghai, China 201102 P.R.C.
Phone: +86-21-6485-2725
Fax: +86-21-6485-2726

Shanghai Branch Office:
406/F, Ying Building, Quantum Plaza, No. 23 Zhi Chun Road,
Hai Dian District, Beijing,
China 100083
Phone: +86-10-8235-3377
Fax: +86-10-8235-6717

Advantest (Singapore) Pte. Ltd.
438A Alexandra Road, #08-03/06
Alexandra Technopark Singapore 119967
Phone: +65-6274-3100
Fax: +65-6274-4055

Advantest America, Inc.
3201 Scott Boulevard, Suite, Santa Clara, CA 95054, U.S.A
Phone: +1-408-988-7700
Fax: +1-408-987-0691

ROHDE & SCHWARZ Europe GmbH
Mühldorfstraße 15 D-81671 München, Germany
(P.O.B. 80 14 60 D-81614 München, Germany)
Phone: +49-89-4129-13711
Fax: +49-89-4129-13723

ADVANTEST

http://www.advantest.co.jp

ADVANTEST CORPORATION
Shin-Marunouchi Center Building, 1-6-2 Marunouchi, Chiyoda-ku, Tokyo 100-0005, Japan
Phone: +81-3-3214-7500