ADVANTEST®

ADVANTEST CORPORATION

R3752/53/64/65/66/67 series
Network Analyzer
PROGRAMMING GUIDE

MANUAL NUMBER GEA00 9603

Applicable Instruments
R3752A/B/E
R3753A/B/E
R3764A/B/C
R3765A/B/C
R3766A/B/C
R3767A/B/C

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 March 1, 1996
Printed in Japan

PREFACE

Using This Manual

This manual describes how to create and execute the BASIC program by the use of editor in R3752/53, R3764/66 and R3765/67 Series analyzers.

Related Manuals:

The following four manuals describe the names of the part, functions and key operations of this network analyzer.

- R3752 Network Analyzer Operation Manual
- R3753 Network Analyzer Operation Manual
- R3764/66 Network Analyzer Operation Manual
- R3765/67 Network Analyzer Operation Manual

The following two manuals describe built-in BASIC and GPIB.

- R3752/53 Programming Manual
- R3764/66, R3765/67 Programming Manual

Convention of the keys in this manual:

• Panel keys of this analyzer:	Be shown with a bracket. (Example) [ENT], [BS], [0] to [9]
• Soft keys of this analyzer:	Be shown with a curly bracket. (Example) {EDIT}
• IBM-PC keyboard:	Be shown with a style of bold Italic Helvetica. (Example) ***Enter, Backspace. 0*** to ***9***

Convention of the commands in this manual:

- The commands input by the keyboard: Be shown with a style of Italic Helvetica.
  (Example) *EDIT* command, *PRINT*

- The commands selecting from the menu of editor:
  Be shown with a style of bold Helvetica.
  (Example) **EDIT** command, **Print**

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

LIST OF EXAMPLES

List of Examples

1 Introduction

In R3752/53, R3764/66 and R3765/67 Series (analyzers) , since the execution environment of BASIC-language programs is provided, the BASIC programs created on external personal computers etc. can be executed. In addition, the development of program with the editor described in this manual can be performed.

(Note) First install the editor, because it has not been installed before delivery.

1.1 Features of the BASIC Language Provided

The BASIC language can load and execute the BASIC programs which have no line numbers. When a BASIC program without line numbers is loaded, it is numbered automatically from the first line. Since the line number is not necessary, the editing of program can be performed easily.

To create a BASIC program with this instrument, the editor should be installed. When the editor is used, the program file can be loaded and executed in this editor. When the program is executed in this editor, the error occurred during the execution (runtime error) is traced.

When an error occurs, the cursor moves automatically to the line where the error generated. Almost every command prepared in this editor can be executed from the menu, and so the commands can be simply carried out without referring to the manual.

1.2 Editor Package

The editor package consists of the following items.

- Editor install disk (1)
  (PR37670001-FK, HD 1.44M bytes)
- Sample program disk (1)
  (PR37670003-FJ , DD 720k bytes)
- Programming Guide (This manual)

1.3 Equipment Hardware Environment

The following hardware environment is required for use of BASIC and the editor.

- Network analyzer R3764/66 series or R3765/67 series
  Or network analyzers with the version of system software is B00 or higher, R3752/53 series.
- IBM-PC compatible keyboard 101 or 106
- External CRT display (for R3752, R3764/66)

1.4 Installation

Install the editor before using.

1.4.1 Installation Method

Installing the editor is done by simply inserting the install disk provided in the editor package into the disk drive of the analyzer and restarting the analyzer.

- Installing procedure

  ① Switch the power off.

② Insert the install-disk of the editor into the drive.

③ Switch on the power.

When the installation gets started, the message "INSTALLING EDITOR..." appears on the screen.

④ Wait until the message " COMPLETE" appears on the screen.

⑤ Eject the install-disk of the editor from the drive.

⑥ Switch off the power of this instrument and then restart.

The editor is completed to be installed in the memory disk (C: BIN) of the analyzer.

(Note) The operation of the editor is not ensured if it has been copied or modified. If installation is attempted when many registers are in use, installation might not succeed. If this occurs, perform installation after first deleting registers.

1.4.2 Setting the System

- Using 101 keyboard

  It is necessary to set the system.
  Enter 101 in the service menu of the analyzer.
  If not, the key code may change.

- Using 106 keyboard

  It is not necessary to set the system.

1.5 Disk Format Conversion Program

The program (floppy disk) created with the old type network analyzer made by ADVANTEST (R4611/R3751 series / R3762 series) can not be used without file exchanging. It is necessary to convert it to the disk format that can be processed with MS-DOS by using the disk conversion program (PR37670002-FK), that is optionally available.

The whole program operation is done on the front panel.

(Caution) Set the disk formatted by the old type to the write protection mode, in order not to initialize it to MS-DOS format.

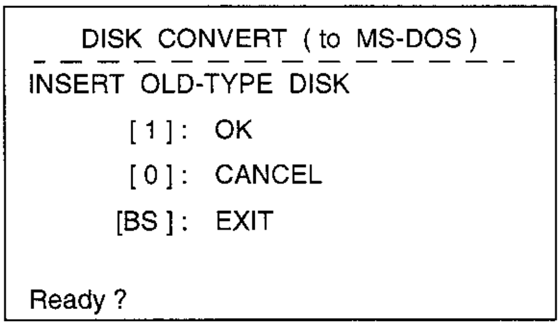
- Converting disk format

① Switch off the power (R3764/66, R3765/67 series).

② Insert the format conversion program disk into the drive.

③ Switch on the power.

④ When the following message appears on the screen, eject the floppy-disk from the drive.

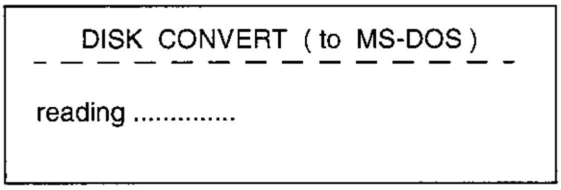


```
   DISK  CONVERT  ( to  MS-DOS )
- - - - - - - - - - - - - - - - -
INSERT  OLD-TYPE  DISK
          [ 1 ] :   OK
          [ 0 ] :   CANCEL
         [BS ] :   EXIT

Ready ?
```

⑤ Insert the floppy disk into the drive to convert the format.

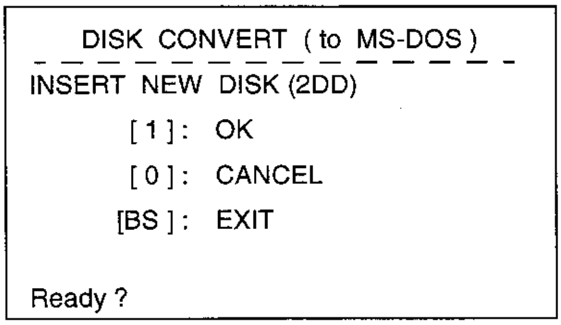
⑥ Press the ten key [1] of the front panel, then the screen changes to as follows.



```
   DISK  CONVERT  ( to  MS-DOS )
- - - - - - - - - - - - - - - - -

 reading ..............
```

After the floppy disk has been read out successfully, the screen changes to as follows.



```
   DISK  CONVERT  ( to  MS-DOS )
- - - - - - - - - - - - - - - - -
INSERT  NEW  DISK (2DD)
          [ 1 ] :   OK
          [ 0 ] :   CANCEL
         [BS ] :   EXIT

Ready ?
```

(Note) ***The use of a floppy disk that is not properly formatted causes error message. In such case, press any one in "Ten Key".***
The screen returns to the original one, then insert another floppy disk and repeat the operation of ⑥

⑦ Eject the floppy disk from the drive.

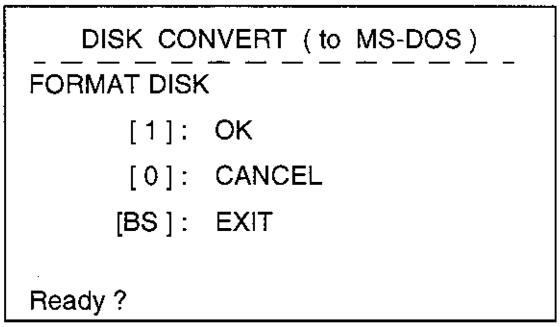
⑧ Insert a new 2DD floppy disk into the drive.

⑨ • In the case of inserting formatted floppy disk

Press the ten key [1] of the front panel , then the contents of the old type network analyzer disk is copied into the new disk. When the copying of the disk ended, the screen returns to the original one.

• In the case of inserting unformatted floppy disk

Press the ten key [1] of the front panel, then the screen changes to as follows.



```
    DISK  CONVERT  ( to  MS-DOS )
FORMAT DISK
          [ 1 ] :   OK
          [ 0 ] :   CANCEL
         [BS ] :   EXIT

Ready ?
```

Press the ten key [1] of the front panel, then the contents of the old type network analyzer disk are copied into the new disk after the floppy disk has been formatted. When the copying of the disk ended, the screen returns to the original one.

⑩ To convert other disks continually , repeat the same operations mentioned above. When the disk format conversion is completed, press the [BS] on the front panel.

2 Fundamentals of Programming

2.1 Program Mode

2.1.1 Starting the Program Mode

(1) Starting the program mode

To create the program of BASIC by using the key board connected with the network analyzer, the system must be turned to the program mode from the measurement mode.

- R3764/66 series

  Since the program mode is started automatically when the power is switched on , no special operation is necessary.

- R3765/67 Series

  When the power is switched on, the state of R3765/67 gets into the measurement mode, in which the frequency of filter etc. can be measured. It is necessary to perform the following operations.

① To convert the mode to the program mode, press [RUN] of the front panel key.

The program menu is displayed and the cursor appears on the screen. (Refer to Figure2-1.)

② To delete the program mode, press any key of the front panel except [CH1] and [CH2].

When the program mode is deleted, the cursor disappears from the screen.

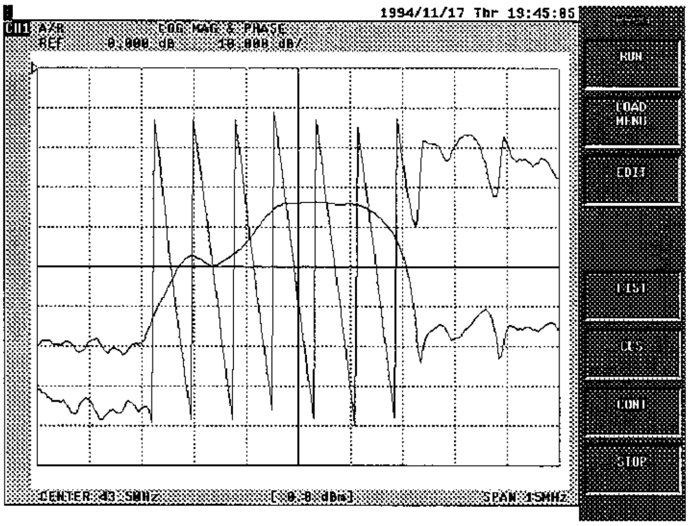


Figure 2-1 Screen of Program (Direct) Mode

(2) Program mode state

The BASIC commands and programs can be entered from the external keyboard in the program mode.
The program mode consists of "direct mode" and "edit mode". In the initial state, the direct mode is set.
In the direct mode, the entered character string is executed after it is sent to the BASIC directly. (Refer to 2.1.2.)
In the edit mode, the entered character string is not executed. (Refer to 2.6.)

The measurement can also be performed in the program mode. In the direct mode, the measured waveform is left as the waveform displayed. The character string produced through the use of entered text and PRINT syntax of the BASIC is shown overlapped with the measurement waveform.
The part in which the character string is displayed is called "text screen", and the part in which the measurement waveform is displayed is called "measurement screen". These two screens are displayed synthetically.
As a result, the characters of the text on the screen are not clear, depending on the displaying position and the measurement waveform. In this case, the following method enables text character being clear. (Refer to (3).)

(3) Method for enabling the text screen clear

There are two methods as follows:

- Disables the measurement screen with DISP statement and enables the text screen only (the waveform data can not be seen .).
- Divides the full screen into two parts: the upper and the lower (the upper is the measurement screen and the lower is the text screen.).

① Press [DISP] key of the front panel.

② Set the display menu to DUAL CH OFF and SPLIT CH ON.

③ Set the scroll area to the lower part of the screen by using the CONSOLE statement in the program mode.

(Note) ***Only the active channel is available in this method. This method is not available for the dual channels being ON (DUAL CH ON), the method can not be used.***

2.1.2 Direct Mode

In the state of direct mode, the cursor move key is not to enable the cursor moving freely. The characters entered from the keyboard are displayed at the position of the cursor. When a character is input, the cursor moves one position to the right.

When the entered character is to be corrected, press the ***Backspace*** to erase one character on the left.

When ***Enter*** is pressed, the entered character string is executed as a command.

There are the following two ways to input from the keyboard.

- Executing by inputting a command and outputs the result soon. (direct format).
- Creating a processing procedure by using multi-commands (program format).

When the direct mode is used, each statement of the BASIC is executed directly and the operation can be ensured. If the statement is executed in direct mode, the operation of the statement can be ensured even when the program is not executed totally,

- Executing the command in direct mode

① Enter CLS and press ***Enter***, then the full text screen can be deleted.

② Input the following statement correctly.

```
PRINT 2+5
```

③ Press ***Enter*** and execute the input code.

When the PRINT statement is executed, 7 is displayed in the text screen.

2.2 BASIC Instructions by Command

When a command is input in the direct mode, it is executed at once and the result is output immediately. It is used only for calculations or commands not necessary to store, such as the list of programs having been input so far.

2.2.1 PRINT Command

PRINT command is used to evaluate the equations following to PRINT and displays the results on the screen. Moreover, it can display character strings instead of the equations on the screen.

- Calculating equations following the PRINT and displaying the results on the screen.

  The four fundamental operations of arithmetic (addition,substraction, multiplication and division) in BASIC are expressed with plus sign (+), minus sign (-), asterisk mark (*) and slash mark (/). The power of number is expressed with the caret (^).

  Input the following statements in the direct mode, then press ***Enter*** and check the results of the operation.

Operation	Result
PRINT 2.0+3.0	5.0
PRINT 2.0-3.0	-1.0
PRINT 2.0*3.0	6.0
PRINT 2.0/3.0	0.66666666666
PRINT 2.0^3.0	8.0

- Displaying character string on the screen.

  Character string is a sequence of characters enclosed in double quotation marks “ ”.

  ① Input the following statement correctly.

  ```
  PRINT  “Hello, world”
  ```

  ② Press ***Enter***.

  Execution result: Hello, world

  The calculation result can be assigned to a variable and then output by using it.

2.2.2 Assignment Command

The assignment command is used to assign the <expression> on the right of the equal mark (=) to the <variable> on the left.

The content of the <expression> is available for numerics or character strings. However, the type of <expression> must be the same as that of<variable>. When the type is mismatched, it is assigned in accordance with the type of the left <variable>.

- Assigning calculation result

  ① Input the following statement correctly.

  ```
  A=(5+3)^2
  B=2*5+6/2
  ```

  ② Press ***Enter***.

  ③ The following statement is input and the value of A is displayed.

  ```
  PRINT A
  ```

  Execution result: 64.0

  ④ The following statement is input and the value of B is displayed.

  ```
  PRINT  B
  ```

  Execution result: 13.0

In these example, equations $(5+2)^2$ and 2×5+6÷2 are calculated and the answers are assigned into A and B, respectively. The contents of A and B are displayed by using the PRINT command.

2.3 Creating and Executing Programs

In the program mode, the program is stored in the memory, then a command is executed. Therefore, after the program execution is completed, the program still remains in the memory, even when the command, having been executed, disappears. The program can be executed for several times.

2.3.1 Program Input and Execution (LIST , RUN)

The input method of program is adding a line number in front of the command input statement that has been described above.

A line which the statement follows is called "program line" and a line without a specified line number is called "command line".

(Example) 10 PRINT 2+5: Program line with line number 10
PRINT 2+5: Command line

Each program line of the BASIC is proceeded by line number.The line number indicates the sequence with which the program lines are stored into the memory, and the program execution is performed from lower number to higher one.

- Program example

```
10 A=(5+3)^2
20 B=2*5+6/2
30 PRINT A
40 PRINT B
50 STOP
```

(1) Displaying program list (LIST).

The LIST command is used to display the program list in the scroll area of the screen.

① Input the following command correctly.

```
LIST
```

② Press ***Enter***.

(2) Displaying a part of the program lines (LIST starting line, ending line).

LIST starting line, or ending line is a command to display the program list of the specified line range.

(Note) If the starting line is omitted , the program from the first line through the ending line will be displayed. If the ending line is omitted, lists from the starting line through the last line of the program. If both of the two lines are omitted, lists from the first line through the last line.

① Input the following statement correctly.

```
LIST 10,40
```

② Press ***Enter***.

Execution result: Displaying the program lines from line number 10 to line number 40.

(3) Executing a program after the input is completed. (RUN)

The RUN command is used to execute the entered program.

① Input the following command

```
RUN
```

② Press ***Enter***.

Execution result: If there is no input mistake, “64.0” and “13.0” will be displayed on the screen.

2.3.2 Scratching Programs (SCRATCH)

SCRATCH command is used to erase the programs and variables, that have been input. When inputting a new program , be sure to erase the programs entered so far.

- Scratching programs

① Input the following command.

```
SCRATCH
```

② Press ***Enter***.

Execution result: The following message is displayed on the text screen.

```
BASIC Ready
```

2.3.3 Input data during program execution

In the programs mentioned above, the data of calculation has been integrated into the programs in advance. Here is a program to enable the calculation data to be entered during execution and then output the result.

- Program example for calculating the area of a triangle

① Input CLS command to clear the screen.

② Input the following program correctly.

```
10  INPUT “TEIHEN ?=” ,A
20  INPUT “TAKASA ?=” ,B
30  C=A*B/2
40  PRINT “KOTAE”, C
50 STOP
```

③ Input Run command to execute the program.

④ Then,the state is ready for the key input.

Since “ TEIHEN?=” is displayed to ask the value of base, inputs the value by the keyboard.
“TAKASA?=” is displayed to ask the value of height, inputs the value by the keyboard.
The area of triangle is calculated and displayed as “KOTAE”, thus the value of answer is output.

2.4 Saving Programs (INITIALIZE, SAVE)

Save the program to be stored in a floppy disk. The object of this procedure is as follows.

A program line proceeded by a line number is input and the ***Enter*** is pressed. Then the program is stored in the memory. However it will be deleted when the power is switched off.That is the same in SCRATCH command.

Prepare a floppy disk and initialize it with INITIALIZE command. The initialization is writing the necessary information on the disk in order to enable it to be used on the network analyzer. INITIALIZE can be abbreviated to INIT.

(Note) 2DD floppy disk can not be initialized with the writing format of 2HD.
Conversely, the 2HD can not be initialized with the writing format of 2DD.

The software structure of floppy disk in R3764/66 and R3765/67 is the same as that of MS-DOS. Therefore, the floppy disks initialized at MS-DOS can be used just as they are.

When the floppy disk initialized at personal computer is used, it is not necessary to use the INITIALIZE command. However, the 2HD floppy disk initialized with 720k bytes and the 2DD floppy disk initialized with 1.2M bytes and 1.4M bytes cannot to be used.

- Initializing floppy disk (INITIALIZE)

  For the 2DD floppy disk:

  ① Close all the opened files according to the CLOSE * command.

  ② Insert the floppy disk into the drive.

  ③ Input the following command.

  ```
  INITIALIZE  "MYDISK" 0
  ```

  If it is executed, the floppy disk is initialized and its name is "MYDISK". For the volume name, up to 12 characters are available, and it can be specified freely (also can be abbreviated). The memory size of floppy disk after initialization becomes 720K bytes.

- For the 2HD floppy disk

  Specifications as follows.

Writing format	Sector Size	Sector count of each drag	Total capacity
INITIALIZE "MYDISK" 1	1024 bytes	8 sector	1.2 M bytes
INITIALIZE "MYDISK" 2	512 bytes	18 sector	1.4 M bytes
INITIALIZE "MYDISK" 3	512 bytes	15 sector	1.2 M bytes

- Saving a program (SAVE)

  To save a program in a floppy disk, use the SAVE command.

  ① Insert the initialized floppy disk into the drive.

  ② Input SAVE command.

  ```
  SAVE  "A:MYFILE.BAS"
  ```

(Note) The file name can be specified with up to 8 characters and an extension of 3 characters at the most. When a file is saved with the same name as that of the file which has been saved previously, the old program will be overwritten and lost.

Although the program has been saved,it still remains in the memory so long as the power is on.
When creating new program, input it after deleting the old with the SCRATCH command.
When a sub-directory is created after the floppy disk is formatted with personal computer, the saving can be performed in this sub-directory.
When saving is performed in a sub-directory, specify the directory name in front of the file name.

```
SAVE  "A:MYSUB1/MYFILE.BAS"
```

In this example, the program is saved under the directory "MYSUB1".
However, there is no function to create the sub-directory in the BASIC.

2.5 Loading Programs (LOAD)

To read out the stored program from the memory (LOAD), use the LOAD command.

① Insert the floppy disk into the drive.

② Input the LOAD command.

```
LOAD "A:MYFILE.BAS"
```

(Note)
- ***Input the file name specified when it is saved.***
- ***When the program is loaded, the program entered previously is removed. (The LOAD command is executed with the SCRATCH together.)***

2.5.1 Searching File Names (CAT)

When you forget loading a file name or want to know which files are saved in a floppy disk, use the CAT command.

① Insert the floppy disk into the drive.

② Input CAT command.

```
CAT
```

Execution result: The following is displayed in the scroll area of the text screen.

```
NO    :File Name      Bytes  At
1     :MYFILE.BAS     418    0
```

The contents shown on the screen represent register number, file name, character counts of file and file attribute from left to right in turn .

2.5.2 Outputting to Printer (GLIST, LLIST)

As one of the methods for storing programs, the program also can be printed out on the printer by using a printer device.

The printer output can be performed in the following two ways.

- Using GLIST command, output to a printer via GPIB.
- Using LLIST command, output to a printer which is connected with a serial port.

The writing methods of these two commands are the same. However, when GPIB is used, the mode of GPIB should be set to SYSTEM CONTROLLER in advance and the GPIB address of printer must be set.

- Printing on a GPIB printer of address 18.

  ① Input “CONTROL 7;1” then GPIB mode turns to SYSTEM CONTROLLER.

  ② Input “PRINTER 18” and set GPIB address of printer.

  ③ Input GLIST command.

  ```
  GLIST  10,100
  ```

  Execution result: Printing out lines from line number 10 to 100.

2.6 Editing Mode

The edit mode is used to edit programs.

When program editing is performed in the direct mode, the cursor can not move freely. Therefore, to correct a program line that has been input previously, it is necessary to input the whole line. This is very inconvenient.

In the edit mode, all the operations necessary for the programming, from program editing to program execution, can be performed by operating the pull-down menu. This program environment is called “BASIC editor “.

Here, editing simple program with this editor is performed. For the detail of editor, see “3. Function of BASIC Editor”.

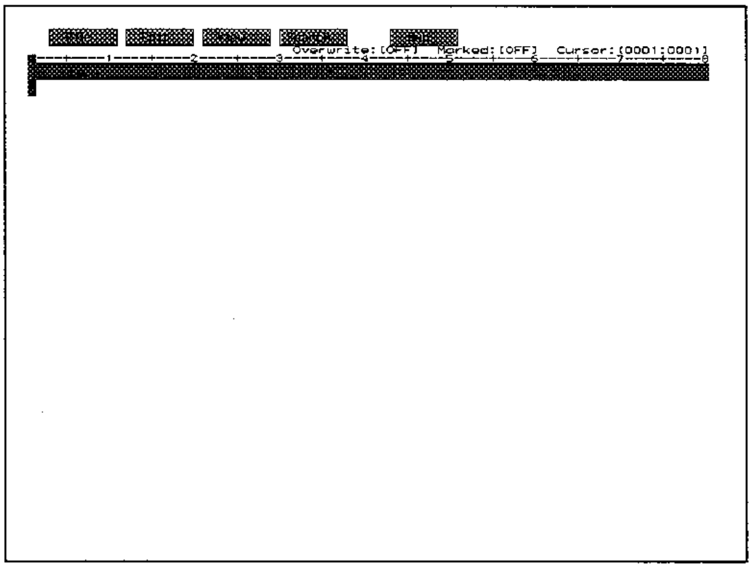
2.6.1 Starting Editor

Install the BASIC editor with the attached install-disk before starting the editor (refer to 1.4.1) . If the editor is started without installation, the following message appears and the execution can not be performed.

Editor not installed

- Turning to edit mode

  Press ***F12*** of the external keyboard to turn to the edit mode.
  When the environment is turned to the edit mode, the screen becomes as shown in Figure 2-2.
  The measurement waveform and the text are not shown on the editor screen.



Overwrite:[OFF] Marked:[OFF] Cursor:(0001:0001]

Figure 2-2 Screen of edit Mode

2.6.2 Programming Environment of Editor

There are many programming tools in the editor.

The program auditing, file management and printing etc. can be performed.

There are several menus in the menu bar at the uppermost place of the editor screen. The commands used to control the programming environment are in the menu. By using these commands, creating and revising program can be carried out in the view window.

2.6.3 Opening Menu

Almost every command of the editor is executed by choosing it from the menu. The menu bar is laid out in the order from left to right corresponding to the function keys (F1, F2 ...) of the keyboard.

To open a menu , press the corresponding function key.

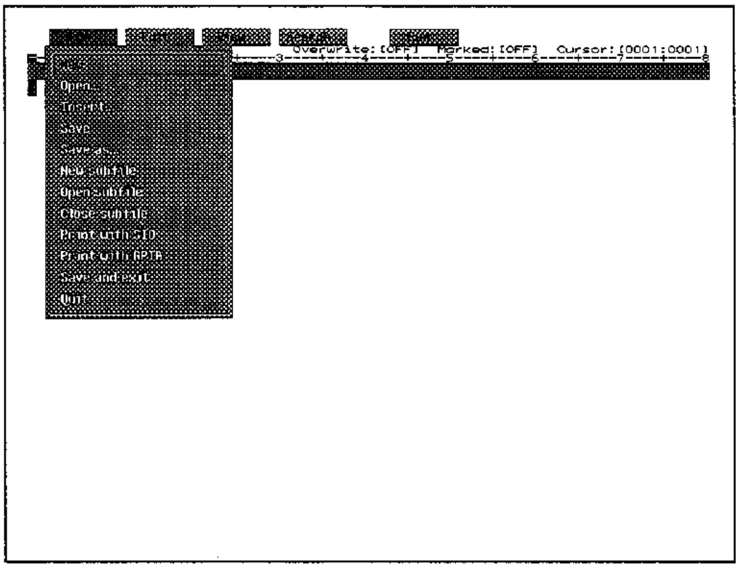
If the menu has been opened, selecting menu can be performed with the direction keys (↑, ↓ , ←, →)

- Procedure for opening menu

① Press ***F1*** to select F1: File menu.

Using ↑ and ↓, you can choose any command in the opened menu.
Using ← and →, you can open any menu in the opened menu.

② To close the menu without executing command, repress the ***F1*** or press the ***Esc*** key.



```
Overwrite:[OFF]  Marked:[OFF]  Cursor:[0001:0001]
```

Figure 2-3 F1: File Menu

2.6.4 Choosing Edit Command

To choose and execute edit commands, choose the target command from menu by using ↑ and ↓, then press ***Enter***.

If a command name is followed by ellipsis dots (...) , execute the command, then a dialog box appears as shown in Figure 2-4. If a command name is not followed by ellipsis, the command is executed immediately.

2.6.5 Using a Dialog Box

When a command, whose name followed by ellipsis(...) is executed, a dialog box appears. The editor presents necessary information before executing a command by using the dialog box.

(1) Components of a dialog box

Dialog box is used to provide editor information.

Components	Explanation
Text field	Used to input text such as file name, etc.
List box	Used to select one item from several items.
Command button	Showing usable keys

(2) Operation of a dialog box

If you open a dialog box for the first time , the default setting is displayed.
After that, if you open the dialog box, the previously selected setting is displayed.
To choose any items in a dialog box, use ***Tab***, ← or →.

- Opening dialog box of the ***Open ...*** command in F1: File menu

① Press ***F1*** to open the F1: File menu.

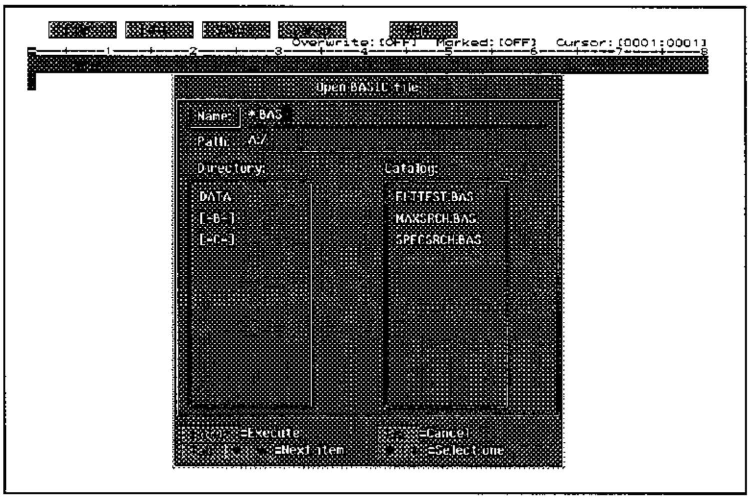
② Press ↓ until the ***open ...*** command is displayed in highlight.

③ Press ***Enter*** to open the dialog box.

④ Press ***Tab*** or → continually to move the cursor to the "Catalog :" list box.

When there exist a lot of items in a list box, changing selection item in the list box can be carried out by pressing ↓.

⑤ Press ***Esc*** to close the dialog box, without reading file.



```
Overwrite: [OFF]  Marked: [OFF]  Cursor: [0001:0001]

Open BASIC file
Name: *.BAS
Path: A:/
Directory:            Catalog:
DATA                  FLTTEST.BAS
[-B-]                 MAXSRCH.BAS
[-C-]                 SPECSRCH.BAS
```

Figure 2-4 Dialog Box of *Open* ... Command

2.6.6 Executing Edit Command Directly

The edit command can be executed directly when used in combination with the shortcut keys.

It is not necessary to open the menu and choose a menu command when the edit command is executed directly.

The shortcut keys can be used with the main edit commands.

(1) Shortcut keys used for choosing command

For the commands being to use shortcut keys,the indications of the short cut keys are listed to the right of the command name.

(Example) When cutting and pasting

The ***cut*** command of F2:Edit menu can be executed by pressing the ***Delete***, holding ***Shift*** key down. The ***Paste*** command, by pressing the ***Insert***, holding ***Shift*** key down.

(2) Other keys used in combination

Almost every key is used in combination of ***Shift***, ***Alt***, ***Ctrl*** and function keys: ***Insert***, ***Delete*** , ↑, ↓, ← , →, etc.

(Example) When pressing ← or →, the cursor of the screen can be moved to the left or right. To move the cursor to the left or the right from the word where it currently located, press ← or →, holding the ***Ctrl*** down.

2.6.7 Closing Editor

- Closing editor

① Execute ***Quit*** command of the F1: File menu.

② When there is a file being not saved, a message appears to make sure whether the closing may be executed or not.

When pressing ***Y***, the environment becomes direct mode without saving the file.
When pressing ***N***, the screen returns to the original edit and executes program once again after saving the file.

2.7 Editing Programs

This section explains how to edit a program by using the editor.

Editing a program means the operations that perform revising, adding and correcting for a written program. When a program line is input in the direct mode, it must be added by a line number. However, it is not necessary in the edit mode.

When a program file without a line number is specified, the BASIC adds the line number automatically.

(Caution) The program examples to be explained later are described without the line number.

2.7.1 Inserting Characters

When you want to insert characters into the line which has been programmed, move the cursor to the character behind the location where you want to insert, then input the characters as they are.

When entered, the characters are inserted into the cursor position. All the characters, from the cursor position to the end of the line, move to the right by one position.

- Inserting characters

① Input the following program line.

```
PRNT “ NUMBER”▨
```

② Press the ← key to move the cursor to the N of PRNT.

```
PR▨T “NUMBER”
```

③ Input I. I is inserted into the location of N and the characters following N is moved to the right.

```
PRI▨T “NUMBER”
```

When pressing ***Insert***, a space is inserted at the cursor location as it is.
All the characters following the cursor move to the right by one position and the space is displayed at the cursor location.
In a program, up to 511 characters are available for each line.
However, when the characters are more than 80, the screen displays only 80 characters at most.
If you press → key to move the cursor to the last character of the displayed line, the remained characters are displayed in the screen.
To display the first character of a line, press ← key and move the cursor to the first character of the displayed line.

2.7.2 Inserting Lines

When you want to insert a new line between two lines of a program, press ***Insert*** holding ***Ctrl*** down.

- Inserting lines

① Input the following program line.

```
INPUT A
C=A*B▨
```

② Add a command line called “ INPUT B” between “ INPUT A” and “C=A*B” of this program as follows.

Press ***Home*** key holding ***Ctrl*** down. The cursor moves to the head of the line.

```
INPUT A
▨=A*B
```

③ After the cursor has been moved to the head of line, press ***Insert*** holding ***Ctrl*** down.

```
INPUT  A
▨
C=A*B
```

The blank line also can be inserted by pressing ***Enter***. In this case, the cursor is moved to the head of the next line.
To split a line into two parts, press ***Insert*** key holding ***Ctrl*** down or execute ***Enter*** at the middle of the line, then the characters from the cursor position to the end of the line move to the second line.

2.7.3 Deleting Characters and Lines

To delete characters , press ***Backspace*** or ***Delete***.

When ***Backspace*** is used, one character to the left of the cursor is deleted, and the cursor and all the characters from the current cursor position to the end of the line are moved one position to the left.
When ***Delete*** is used, the character of the current cursor position is deleted and the characters from the right of the cursor to the end of the line are moved one position to the left.

(1) Deleting characters by the use of ***Backspace***

① Input the following line.

```
PRIINT "A"▨
```

② Move the cursor to the position of the second I by using the cursor move keys.

```
PRI▨NT  "A"
```

③ Press ***Backspace***.

The first I is deleted, and the characters INT "A" are moved to the left.

```
PR▨NT "A"
```

If the ***Backspace*** is performed at the head of the line , the lines can be connected.
The cursor and the cursor resident line will move up to the end of the previous line.

(2) Deleting characters by the use of ***Delete***

① Input the following line.

```
PRIINT  "A"▨
```

② Move the cursor to the position of the second I by using the cursor move keys.

```
PRI▨NT  "A"
```

③ Press Delete.

The I at the cursor position is deleted and the characters NT "A" are moved to the left.

```
PRI▨T  "A"
```

If ***Delete*** is performed at the end of statement , the lines can be connected.

The next line will move to the cursor position wholly.

(3) Deleting from the cursor position to the end of line.

To delete a line from the cursor position to the end of the line, use **Delete line** command of the F2 : Edit menu.

① Input the following program line.

```
A=1+3
B=5*4
PRINT  A▨
```

② Move the cursor to the head of the line B=5*4 by using the cursor move keys.

```
A=1+3
▨=5*4
PRINT  A
```

③ Press ***F2*** to open the F2 : Edit menu, then execute **Delete line** command. The line B=5*4 is deleted and the next line is moved up.

```
A=1+3
▨RINT  A
```

When the **Delete line** command is executed in the middle of the line, all the characters from the cursor position to the end of the line are deleted (The next line does not move).
If you press ***Delete*** holding ***Shift*** down, this command can be executed at one stroke of the key operation.
When **Delete line** command is executed for a wrong position, perform the **Paste** command at once, then the deleted content can be recovered.
If you press ***Insert*** holding ***Shift*** down, this command can be executed at one stroke of the key operation.

2.7.4 Block Editing

The block editing means the operation that specifies the edit lines in a integrated way.

Using editor, the editing can be performed for the characters and the lines. When the lines are specified in a integrated way, the deleting and moving can be performed in a wide range in integrated way. These integrated lines are called “block”. To select a block, move the cursor to the head of the target block and set the mark, then execute the edit command.

(1) Setting mark

The mark is set at the head of block editing line.

① Input the following program line.

```
INPUT  A
INPUT  B
C=A+B
PRINT  A▨
```

② Move the cursor to the head of the line C=A+B, using the cursor move keys.

```
INPUT  A
INPUT  B
▨=A+B
PRINT  A
```

③ Press ***Space*** holding ***Ctrl*** down.

When the mark is set," Mark set" is displayed in the message line.
The mark is also used for memory of cursor position in addition to the block editing. If press ***Space*** holding ***Alt*** down, then the mark and cursor exchange. The cursor moves to the mark position, and the mark is set at the current cursor position.

(2) Cut and copy

The cutting and copying can be performed when the block is selected with the set mark.

(Note) When the mark is not set, the cutting and copying can not be performed.
In this case " No mark in this window " is displayed in the message line.

Cutting a block of lines selected for block editing means that the selected lines are delete in an integrated way.
To copy a block means that the copying is performed for the selected block.
The cut block and copied block are stored in a place of the memory. This place is known as "clip board".
The content of clip board can not be viewed directly. It is displayed when pasting is performed.
Whenever the cutting , copying and line deleting are performed, the content of clip board is renewed. The contents of cut , copied and deleted line are stored at last.
The content of clip board can be inserted to the cursor position by pasting operation to be described later.

- Executing **Cut** command

① Input the following program lines.

```
INPUT  A
INPUT  B
C=A+B
PRINT  A▨
```

② Move the cursor to the end of line after the mark is set at the head of C=A+B.

③ Press ***F2*** to open the F2: Edit menu.

④ Execute **cut** command.

```
INPUT  A
INPUT  B
▨
PRINT  A
```

The selected text is sent to the clip board.
Only one text block can be sent at a time.
If you press ***Backspace*** holding ***Shift*** down, the command can be executed at one

stroke of the key operation.
If you press ***Backspace*** only, merely one character to the left of the cursor position is deleted.

(3) Pasting text block

The text sent to the clip board is remained till the new text is sent to the clip board or the editor is closed.

- Pasting program line

① Input the following program line

```
INPUT   A
INPUT   B
C=A+B
PRINT  A
▨
```

② Press ***Space*** holding ***Ctrl*** down, then the mark is set at the cursor position.

③ Use the arrow key to move the cursor to the head of C=A+B.

④ Press ***Space*** holding ***Alt*** down, then the mark and cursor exchange.

⑤ Press ***F2*** to open the F2: Edit menu.

⑥ Execute **copy** command.

⑦ Press ***F2*** to open the F2: Edit menu.

⑧ Execute **Paste** command.

```
INPUT  A
INPUT  B
C=A+B
PRINT  A
C=A+B
PRINT A
▨
```

If you press ***Insert*** holding ***Shift*** down, this command can be executed at one stroke of the key operation.
If you Press ***Insert*** only, merely one space is inserted to the cursor position.

3 Functions of BASIC Editor

In this chapter, Various BASIC editor functions described in " 2.6 Edit Mode" are explained in detail. As mentioned above, all the operations necessary for the programming such as editing, debugging till program executing can be performed using the pull-down menu *holded* by editor.

The following items are described.

- Method for starting BASIC Editor
- Method for executing menu command
- Method for selecting options of dialog box
- Scrolling list box
- Method for selecting dialog box and text in window
- Method for changing window size
- Method for using direct screen
- Method for using watching window

3.1 Starting BASIC Editor

Editor is started by external keyboard. (Refer to 3.1.1.)

The BASIC editor must be installed in this instrument before starting the editor by the use of attached install-disk. (Refer to 1.4.1.) When you start the editor without installing it, the following message appears and it can not be started.

Editor not installed

3.1.1 Starting Editor

- Turning to the edit mode

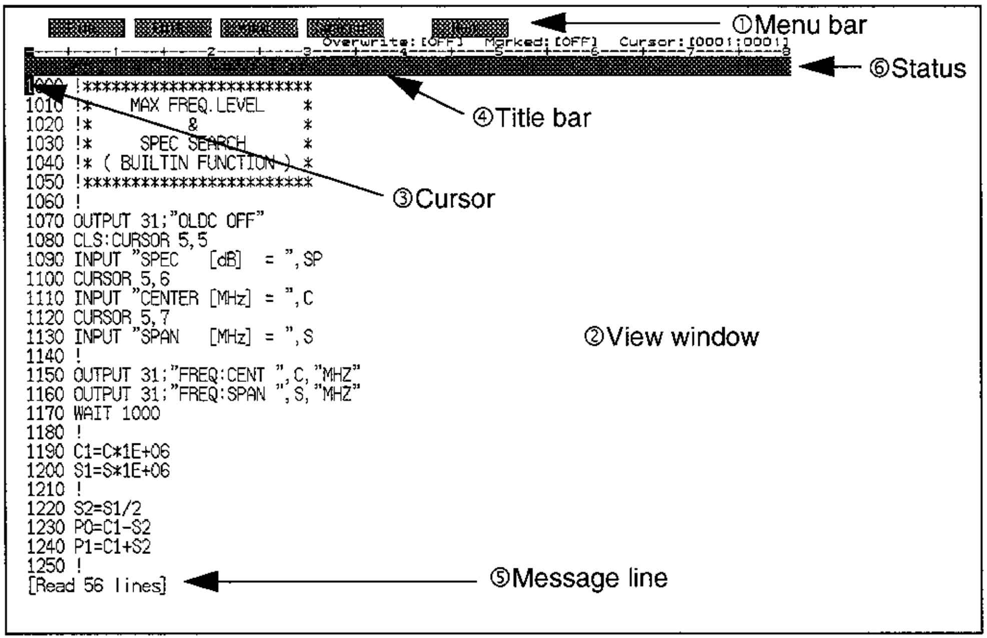
  Press ***F12*** of the external keyboard.

3.1.2 BASIC Editor Screen

When the editor is started, the editor screen is displayed.

In this editor screen, the measurement wave form and BASIC text screen are not displayed.

The component parts of a editor screen are shown in Figure 3-1.



```
Overwrite:[OFF]  Marked:[OFF]  Cursor:[0001:0001]
1000 !************************
1010 !*    MAX FREQ.LEVEL    *
1020 !*          &           *
1030 !*     SPEC SEARCH      *
1040 !* ( BUILTIN FUNCTION ) *
1050 !************************
1060 !
1070 OUTPUT 31;"OLDC OFF"
1080 CLS:CURSOR 5,5
1090 INPUT "SPEC   [dB]  = ",SP
1100 CURSOR 5,6
1110 INPUT "CENTER [MHz] = ",C
1120 CURSOR 5,7
1130 INPUT "SPAN   [MHz] = ",S
1140 !
1150 OUTPUT 31;"FREQ:CENT ",C,"MHZ"
1160 OUTPUT 31;"FREQ:SPAN ",S,"MHZ"
1170 WAIT 1000
1180 !
1190 C1=C*1E+06
1200 S1=S*1E+06
1210 !
1220 S2=S1/2
1230 P0=C1-S2
1240 P1=C1+S2
1250 !
[Read 56 lines]
```

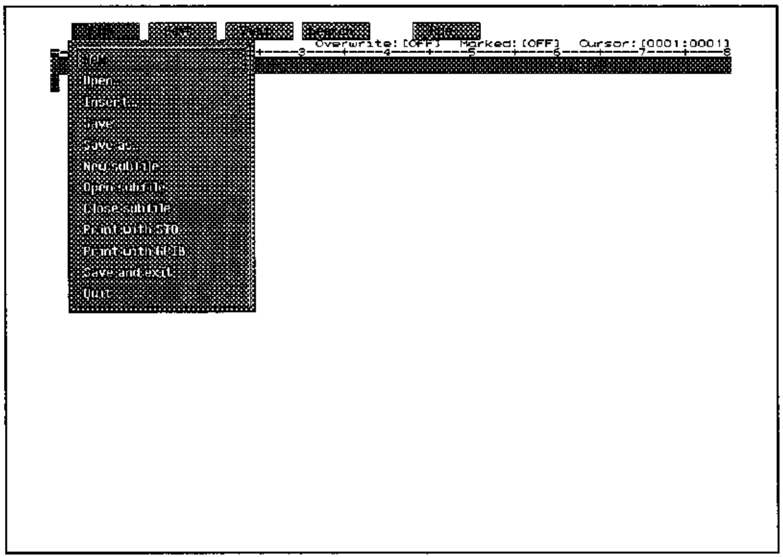
Figure 3-1 Component parts of a Editor Screen

- Component parts of editor Screen

Components	Explanation
①Menu bar	Shows the name of each menu.
②View window	Shows the text of program.
③Cursor	Indicates the position where the text to be input and edited. The cursor appears in the active window.
④Title bar	Shows the buffer name·file name of program and subfile. For BASIC program, the buffer name is represented by main. For other subfiles except that represented as the same as file.
⑤Message line	For the files added by editing, an asterisk (*) is displayed at the head of title bar. Shows the error occurred in the course of editing and the related information.
⑥Status	Shows the status (Overwrite / Marked / Cursor) of editor. • Overwrite: shows Insert / Overwrite mode. • Marker: shows the status of text block.(Specifies text range) • Cursor: shows the position of cursor on the screen with x/y.

3.2 Opening Menu and Executing Command

Each command of editor exists in the pull-down menu of menu bar. Figure 3-2 shows one of the pull-down menu, i.e. F1 : File menu.



Overwrite:[OFF] Marked:[OFF] Cursor:[0001:0001]

Figure 3-2 F1 : File Menu

The programming environment is designed to enable the operation to be performed as simple as possible.

After the menu is opened by pressing function keys,the command can be executed.(Refer to 3.2.1.)

However, some commands can be executed directly by using the shortcut keys without opening the menu. (Refer to 3.2.2.)

For the command followed by ellipsis (...) after the name, it is necessary to show some information that should be specified in more detail before it is executed.

When the command of this kind is executed, a dialog box appears in the view window so as to specify the items that is necessary for the command execution.

3.2.1 Executing Commands through key operation

This section describes a method about opening a menu by using external keyboard and executing the editor commands.

To open a menu, press the function key (***F1*** to ***F5***) that is corresponding to the menu name.

If you press ← or → when a menu has been opened, then the menu to the left or right of this menu can be opened.

When a menu is open, commands of this menu are displayed, and the executable commands are indicated with the command name in high-light.

Press ↑ and ↓, then the high-light indication of command name moves up and down.

Press ***Enter***, then the command displayed in high-light can be executed.

To cancel the execution of a command, press ***Esc.*** Then, the opened menu closes and returns to the original status.

When a command followed by ellipsis (...) after the name is executed, a dialog box appears on the screen. Input the items necessary for command execution.

To cancel the dialog box, press ***Esc***. Then, the execution of this command is cancelled.

3.2.2 Executing Command by using Shortcut keys

This section describes the method of command execution by using the shortcut keys.

When using the shortcut keys, the menu command can be executed at one key stroke. The shortcut keys of menu commands are displayed to the right of the command name on the menu.

Table 3-1 shows the functions of BASIC editor shortcut keys and the catalog of the corresponding commands.

Table 3-1 Shortcut Key Operation

Key operation	Explanation
Shift + Backspace	Cuts the selected area.
Shift + Insert	Pastes the content of clip board.
Ctrl + F2	Cuts from the cursor position to the end of line.
Alt + F2	Pastes the content of clip board.
Ctrl + F3	Loads the next buffer.
Alt + F3	Shows the **Buffer list ...** dialog box.
Ctrl + F4	Searches the next searching target.
Alt + F4	Searches the next searching target backward.
Ctrl+ F5	Restarts the program from the interrupting position.
Alt + F5	Starts the program from the beginning.
F11	Activates the next window.
Shift + F11	Activates the previous window.
F12	Converts between editing screen and output screen.
Shift + F12	Converts the split window alternately.

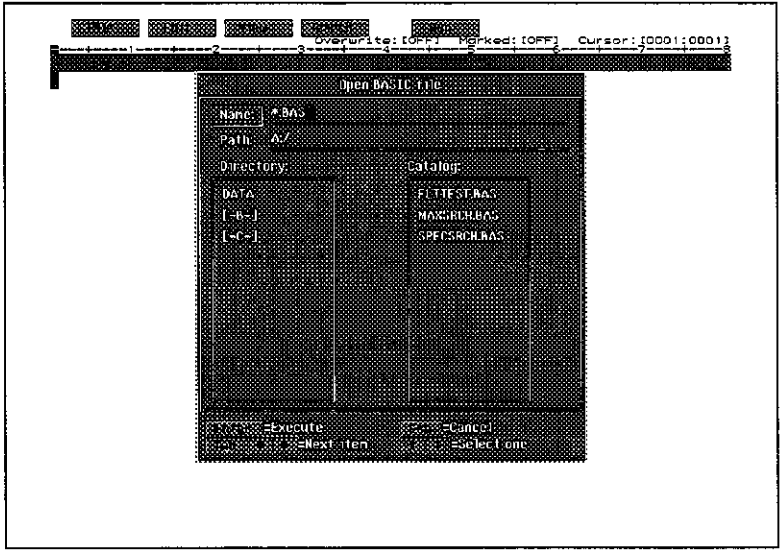
(Note) The said key operation Shift + Backspace means that press the Backspace key holding the shift key down. (The following are same.)

3.3 Using a Dialog Box

(1) Function of dialog box

When it is necessary to specify options before a command is executed, the editor provides a dialog box. Figure 3-3 and Figure 3-4 show the component parts of a dialog box.
The dialog box performs the following functions.

- Acquiring the input of file name
- Acquiring the setting of option



Overwrite:[OFF] Marked:[OFF] Cursor:[0001:0001]

Open BASIC file

Name: *.BAS

Path: A:/

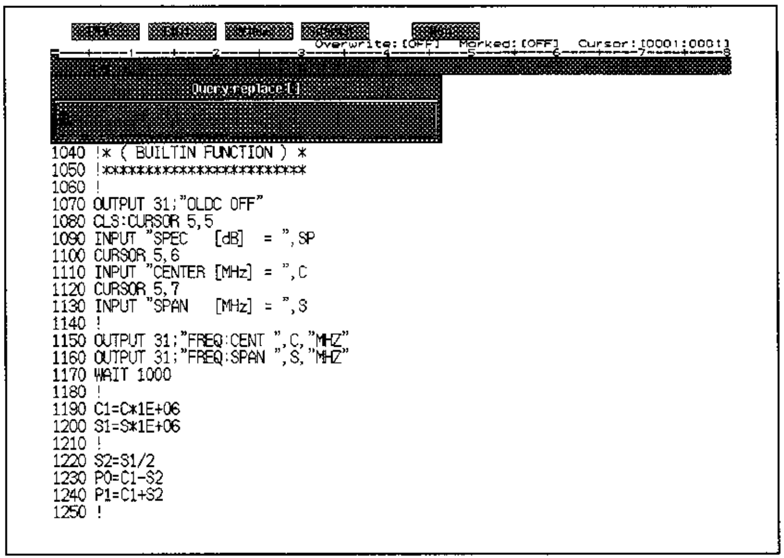
Directory:

DATA
[-B-]
[-C-]

Catalog:

FLTTEST.BAS
MAXSRCH.BAS
SPECSRCH.BAS

Figure 3-3 Dialog Box of Open ... Command



Overwrite:[OFF] Marked:[OFF] Cursor:[0001:0001]

Query replace[]

```
1040 !* ( BUILTIN FUNCTION ) *
1050 !**********************
1060 !
1070 OUTPUT 31;"OLDC OFF"
1080 CLS:CURSOR 5,5
1090 INPUT "SPEC   [dB]  = ",SP
1100 CURSOR 5,6
1110 INPUT "CENTER [MHz] = ",C
1120 CURSOR 5,7
1130 INPUT "SPAN   [MHz] = ",S
1140 !
1150 OUTPUT 31;"FREQ:CENT ",C,"MHZ"
1160 OUTPUT 31;"FREQ:SPAN ",S,"MHZ"
1170 WAIT 1000
1180 !
1190 C1=C*1E+06
1200 S1=S*1E+06
1210 !
1220 S2=S1/2
1230 P0=C1-S2
1240 P1=C1+S2
1250 !
```

Figure 3-4 Dialog Box of Replace ... Command

(2) Moving cursor

To move the cursor to an item of a dialog box, there are the following two ways :

- Press ***Tab*** to move the cursor to the next item.
- Press ***Tab*** holding ***Shift*** down to move the cursor to the previous items.

(3) Function of each item in a dialog box

Path name specifying field :
Shows the path of current directory.
Inputs a new path name with the Name field or selects a suitable directory with the Directory box or changes the display of path.
Text input field:
Shows the entered text.
List box:
Shows the catalog of directory and files etc.
Command button :
Shows the catalog of keys necessary for the command execution.
Performs by pressing the corresponding keys.
To execute command, press ***Enter***.

3.4 Using Message Line

The message line is displayed at the lowest part of the screen.

It is used to make sure the command execution or cancelling, and notify errors.

The message line is erased when performing next operation.

3.5 Using Window

The part showing the loaded program is called " View window ". The view window can perform the split screen editing.

For the editing file and program, a buffer name is added in addition to the file name.
Among the files that in the course of editing, those not displayed in the view window can be referred according to the buffer name.
For BASIC programs, the buffer name is called main.
Among the files being edited, the buffer name:main can be executed in fact.

3.5.1 Function of Each Window

View window can be splitted into two parts : the upper and the lower.
When the window is splitted into two parts, the two parts of program can be viewed and edited at the same time.

To split the view window, execute **Split window** command of F3 : View menu.
Execute this command once again, then the screen returns to the original.

3.5.2 Changing Active Window

The window holding the cursor is called "active window " .
To activate other windows, perform either of the following operations.

- Press ***F11,*** and activate the windows in the lower part of the screen in sequence.
- Press ***F1*** holding ***Shift*** down to activate the windows in the upper part of the screen in sequence.

3.5.3 Changing Window Size

The window size can be enlarged or reduced in line unit.
Also, a window can be displayed on the full screen.

To change the window size, active the window whose size to be changed, and perform the key operations shown below..

Key operation	Explanation
Alt + PageUp	Enlarges the active window by one line.
Alt + PageDown	Reduces the active window by one line.
Shift + F12	Enables the active window to fill the full screen.

3.5.4 Scrolling Active Window

Scroll the view window, when you want to see the upper and/or lower parts of a file that are not displayed in view window.

Press the direction keys to move the cursor to the end of the screen, then the scrolling starts. However, scrolling to left or right is performed only by the current cursor line.

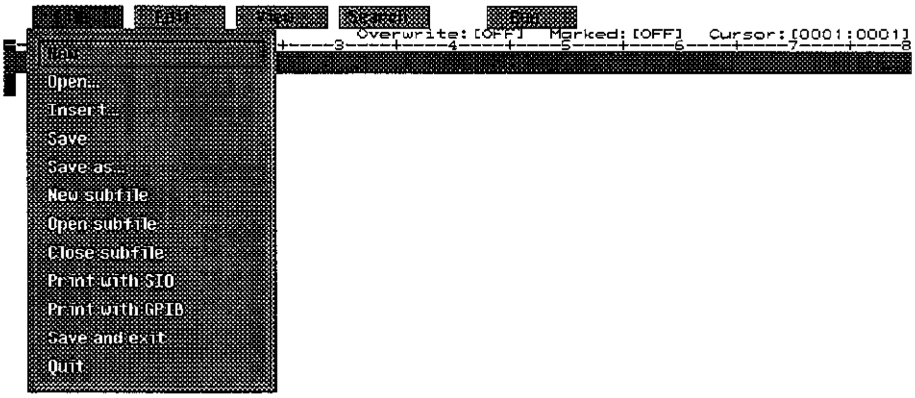
Key operation	Explanation
Home	Moves to the beginning of a file.
End	Moves to the end of a file.
PageUp	Scrolls one page up.
PageDown	Scrolls on page down.
Ctrl + Home	Moves to the beginning of a line.
Ctrl + End	Moves to the end of a line.
Ctrl + ↑	Scrolls one line up.
Ctrl + ↓	Scrolls one line down.

MEMO

4 F1: File Menu

F1: File menu is prepared for editing files used in BASIC.

Executing the commands of F1: File menu, you can create a new file, load files from a floppy disk or revise a file.
Using the commands of F1: File menu, you can also print files on a line printer or end the operations of the editor.



- **New** command

  Removes the BASIC program which has been loaded before.
  Use it when you start to write a new program.

- **Open ...** command

  Loads a program which has been saved in a floppy disk.
  Select the file from files of dialog box and catalog of directory.

- **Insert ...** command

  Merges the contents of two files into one file.

- **Save** command

  Writes the content of a file displayed in an active window into a floppy disk file.

- **Save as ...** command

  Writes a file in the course of operation into a floppy disk file with a specified name.

- **New subfile** command

  Creates a usual text file. The subfile can not be executed.

- **Open subfile** command

  Loads the program stored in a floppy disk and the ordinary file as a subfile.

- **Close subfile** command

  Releases the editing text file from the memory.

- **Print with SIO** command

  Outputs the file contents displayed in an active window from the SIO board.

- **Print with GPIB** command

  Outputs the file contents displayed in an active window from the GPIB board.
- **Save and exit** command

  Writes all the files being iodated into a floppy disk and closes the editor.
- **Quit** command

  Closes the editor.

4.1 New command

The **New** command of F1: File menu deletes all the BASIC program that have been loaded up to now, so that a entirely new program can be entered.
When a program in the memory is not saved, the following message is displayed in the message line.

Discard changes [y / n] ?

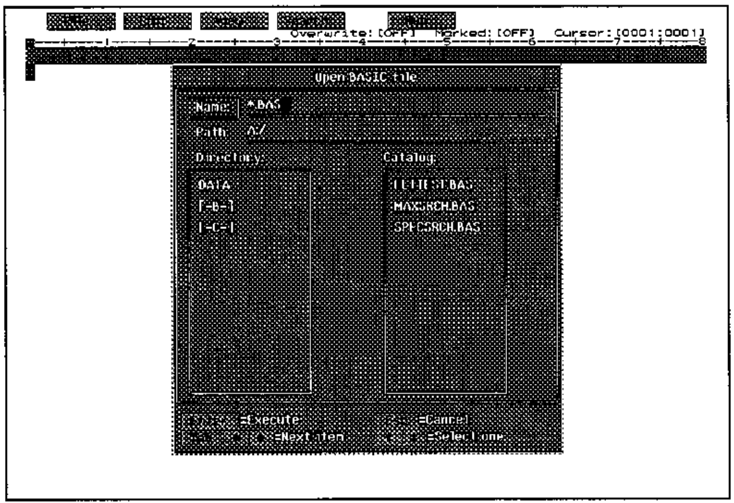
When pressing ***Y***, the program is released from the memory without it saved in a floppy disk, and other files being edited are displayed in the active window.

When pressing ***N***, the program is moved to another buffer editing.

Execute the **New** command after saving in a floppy disk with **Save** command or **Save as ...** command.

4.2 Open ... command

The **Open ...** command is used to load a program which has been saved in a floppy disk.
When this command is executed, a dialog box appears and displays the catalog of files whose extension is.BAS. in the current directory.
The catalog of other directories and the files in other floppy disk also can be displayed with this dialog box.
When the **Open ...** command is executed, the dialog box as shown Figure 4 - 1 appears.



Name: *.BAS
Path: A:/
Directory:
DATA
[-B-]
[-C-]
Catalog:
FLTTEST.BAS
MAXSRCH.BAS
SPFCSRCH.BAS

Figure 4-1 Dialog Box of Open ... Command

4.2.1 Specifying Files

The catalogs of directories and the accessible drives are displayed in Directory box.
The file catalog is displayed in the Catalog box of this dialog box.
When it can not be displayed in one screen, scroll it up and down with ↑ and ↓, separately.
There are the following two ways to specify the file to be loaded.

- Entering file name directly

  Input the file name of program into the Name field, and press ***Enter***.
  To delete the content displayed in the Name field, press ***Backspace***.
  Then, the character to the left of the cursor position is deleted, and the cursor is moved one character position to the left.

- Selecting from Catalog box

  Keep pressing the ***Tab*** to move the cursor to the Catalog box.
  Use ↑ and ↓, to move the high-light indication over to the file to be loaded, then press ***Enter***.

4.2.2 Catalog display of Directory Content

Keep pressing the ***Tab*** to move the cursor to the Directory box. Use ↑ and ↓ to move the high-light indication over to the accessing directory, then press ***Enter***.

All sub-directories on the selected directory and the catalog of files with the extension. BAS in that directory are displayed.

When ***Enter*** is pressed, the file is loaded in the memory.

The following describes how to display the catalog of directory content.

- Displays all the files in current directory.

  Input ∗ in the Name field and then press ***Enter***.
- Displays root directory file of floppy disk.

  Input A:/ in the Path field and then press ***Enter***.
- Displays all the files in subdirectory that is called SUB.

  In the Directory box, display the name called SUB in high-light and press ***Enter***. Then, input ∗ in the Name field and press ***Enter***.
- Displays files of the previous directory of current directory.

  In Directory box, display .. in high-light and then press ***Enter***.
- Displays files whose name is composed of six characters.

  Input ?????? in the Name field, then press ***Enter***.
- Displays files whose name begins with B.

  Input B∗ in the Name field, then press ***Enter***.
- Displays files with the extension DAT.

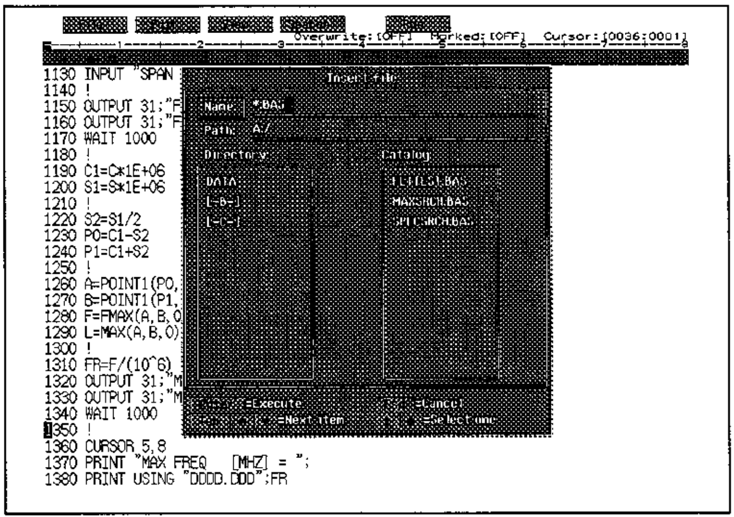
  Input ∗.DAT in the Name field, then press ***Enter***.
- Displays files with the extension composed of two characters.

  Input ∗.?? in the Name field, then press ***Enter***.
- Displays files with the last character of extension from B to FEEEE.

  Input ∗.∗[B-F] in the Name field, then press ***Enter***.

4.3 Insert ... Command

This command is used to insert the content of other files into the cursor position of a working file. When the **Insert ...** command is executed, a dialog box as shown in Figure 4-2 appears. The method of using this dialog box is completely the same as that in the **Open ...** command box.



```
                                  Overwrite:[OFF]  Marked:[OFF]  Cursor:[0036:0001]
1130 INPUT "SPAN
1140 !
1150 OUTPUT 31;"F
1160 OUTPUT 31;"F
1170 WAIT 1000
1180 !
1190 C1=C*1E+06
1200 S1=S*1E+06
1210 !
1220 S2=S1/2
1230 P0=C1-S2
1240 P1=C1+S2
1250 !
1260 A=POINT1(P0,
1270 B=POINT1(P1,
1280 F=FMAX(A,B,0
1290 L=MAX(A,B,0)
1300 !
1310 FR=F/(10^6)
1320 OUTPUT 31;"M
1330 OUTPUT 31;"M
1340 WAIT 1000
1350 !
1360 CURSOR 5,8
1370 PRINT "MAX FREQ   [MHZ] = ";
1380 PRINT USING "DDDD.DDD";FR
```

Insert file

Name: *.BAS

Path: A:/

Directory:

DATA
[-B-]
[-C-]

Catalog:

FLTTEST.BAS
MAXSRCH.BAS
SPTCSRCH.BAS

Figure 4-2 Dialog Box of Insert ... Command

4.4 Save Command

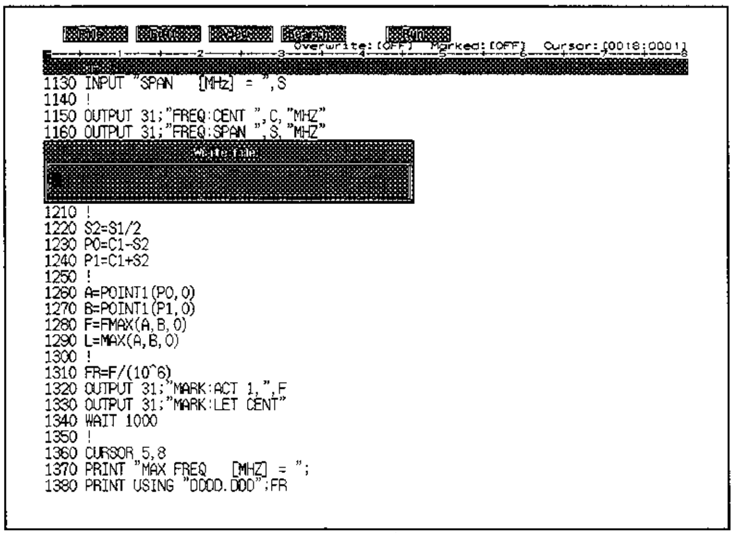
The **Save** command is used to save the content of the operating file to a floppy disk file.

When the file to be saved has been named, the **Save** command writes it into the file which has the same name in the floppy disk.
When the file is not named, the message " No file name" appears in the message line and the file can not be saved. Specify the file name and then save it with the **Save as ...** command.

4.5 Save as ... Command

The **Save as ...** command is used to save a file being operated with the specified name.
This command is used when you specify a new name to a file or do not change a file that has not been revised. When a new name is specified to a file, the old file remains in the floppy disk with the original name.

When the **Save as ...** command is executed, a dialog box as shown in Figure 4-3 appears.
When a new name is entered and saved, the file name in the title bar of window is changed.



Overwrite:[OFF] Marked:[OFF] Cursor:[0018:0001]

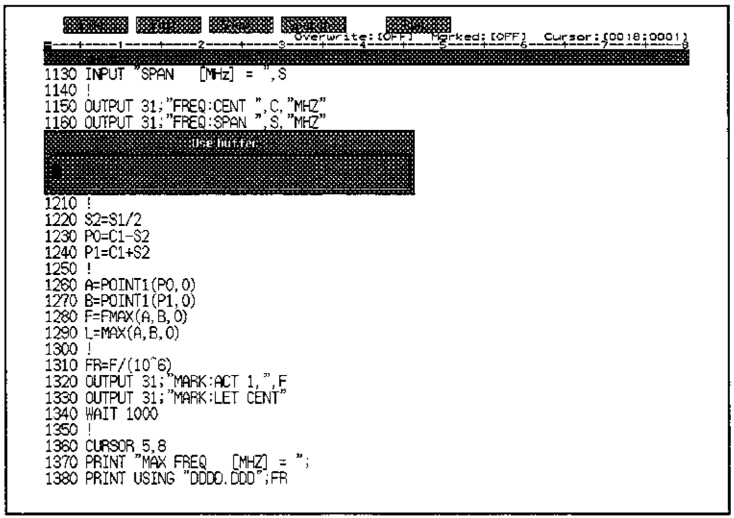
```
1130 INPUT "SPAN   [MHz] = ",S
1140 !
1150 OUTPUT 31;"FREQ:CENT ",C,"MHZ"
1160 OUTPUT 31;"FREQ:SPAN ",S,"MHZ"
1210 !
1220 S2=S1/2
1230 P0=C1-S2
1240 P1=C1+S2
1250 !
1260 A=POINT1(P0,0)
1270 B=POINT1(P1,0)
1280 F=FMAX(A,B,0)
1290 L=MAX(A,B,0)
1300 !
1310 FR=F/(10^6)
1320 OUTPUT 31;"MARK:ACT 1,",F
1330 OUTPUT 31;"MARK:LET CENT"
1340 WAIT 1000
1350 !
1360 CURSOR 5,8
1370 PRINT "MAX FREQ   [MHZ] = ";
1380 PRINT USING "DDDD.DDD";FR
```

Figure 4-3 Dialog Box of Save as ... Command

4.6 New subfile Command

The **New subfile** command is used to create text file except for the BASIC program. The file that is created as a subfile can be executed by using F5 : Run menu.

When the **New subfile** command is executed, a dialog box as shown in Figure 4-4 appears.



```
Overwrite:[OFF]  Marked:[OFF]  Cursor:[0018:0001]
1130 INPUT "SPAN   [MHz] = ",S
1140 !
1150 OUTPUT 31;"FREQ:CENT ",C,"MHZ"
1160 OUTPUT 31;"FREQ:SPAN ",S,"MHZ"
```

```
1210 !
1220 S2=S1/2
1230 P0=C1-S2
1240 P1=C1+S2
1250 !
1260 A=POINT1(P0,0)
1270 B=POINT1(P1,0)
1280 F=FMAX(A,B,0)
1290 L=MAX(A,B,0)
1300 !
1310 FR=F/(10^6)
1320 OUTPUT 31;"MARK:ACT 1,",F
1330 OUTPUT 31;"MARK:LET CENT"
1340 WAIT 1000
1350 !
1360 CURSOR 5,8
1370 PRINT "MAX FREQ   [MHZ] = ";
1380 PRINT USING "DDDD.DDD";FR
```

Figure 4-4 Dialog Box of New subfile Command

Input Buffer Name with this dialog box. The buffer name is used by the editor to control the editing of file internally. When you want to save a file edited with this, do that with **Save** command or specify the file name first and then save with the **Save as ...** command.

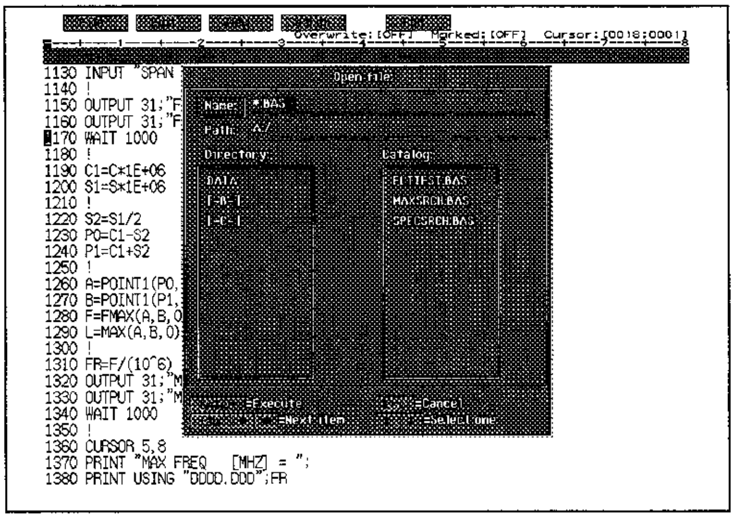
For the program files, only one file can be loaded. But for the subfile, more than one are enabled to be loaded. The subfile can display the file in the course of editing through the buffer name with the **Buffer list ...** of F3 :View window, or display the file information.

For the BASIC program, the executable buffer name is **main**. The **main** must be allocated to the file that is edited with **New** and **Open ...** command.

4.7 Open subfile Command

The **Open subfile** command is used to load the text file from a floppy disk. The file loaded with this command can not be executed by using F5 : Run menu.

When the **Open subfile** command is executed, the dialog box as shown in Figure 4-5 appears.



```
Overwrite:[OFF]  Marked:[OFF]  Cursor:[0018:0001]
1130 INPUT "SPAN
1140 !
1150 OUTPUT 31;"F
1160 OUTPUT 31;"F
1170 WAIT 1000
1180 !
1190 C1=C*1E+06
1200 S1=S*1E+06
1210 !
1220 S2=S1/2
1230 P0=C1-S2
1240 P1=C1+S2
1250 !
1260 A=POINT1(P0,
1270 B=POINT1(P1,
1280 F=FMAX(A,B,0
1290 L=MAX(A,B,0)
1300 !
1310 FR=F/(10^6)
1320 OUTPUT 31;"M
1330 OUTPUT 31;"M
1340 WAIT 1000
1350 !
1360 CURSOR 5,8
1370 PRINT "MAX FREQ   [MHZ] = ";
1380 PRINT USING "DDDD.DDD";FR
```

Open file

Name: *.BAS

Path: A:/

Directory: DATA, [-B-], [-C-]

Catalog: FLTTEST.BAS, MAXSRCH.BAS, SPECSRCH.BAS

Figure 4-5 Dialog Box of Open subfile Command

The method of using this dialog box is completely the same as that in the **Open ...** command.

For the program files, only one file can be loaded. But for the subfile, more than one are enabled to be loaded. The subfile can display the file in the course of editing through the buffer name with the **Buffer list ...** of F3 :View window, or display the file information.

For the BASIC program, the executable buffer name is **main**. The **main** must be allocated to the file that is edited with **New** and **Open ...** command.

4.8 Close subfile Command

The **Close subfile** command is used to delete the subfile being edited in active window from the memory.
When a changed file is not saved in a floppy disk during operation, the following message is displayed in the message line.

Discard changes [y / n] ?

When you press ***Y***, it is released from the memory without saved in a floppy disk, and other files in the course of editing are turned to be displayed in the active window.

When you press ***N***, it is moved to the editing of other buffers.

Execute the **Close subfile** command after saving in a floppy disk with **Save** command or **Save as ...** command.

4.9 Print with SIO Command

The **Print with SIO** command is used to output the content of file displayed in an active window with the SIO board (RS-232C board).

When using the **Print with SIO** command, connect the printer with RS-232C board of this analyzer.

4.10 Print with GPIB Command

The **Print with GPIB** command is used to output the content of file displayed in an active window with the GPIB board.

When using the **Print with GPIB** command, connect the printer with GPIB board of this analyzer.

4.11 Save and exit Command

The **Save and exit** command is used to save all those edited during operation from among the files that currently loaded, and close the editor.
The names of files loaded currently are displayed in the dialog box of **Buffer list ...** in F3 : View menu.

When there exists a file whose name is not specified, " No file name " is displayed and the screen returns to the original. Specify a name to the file with **Save as ...** command.

4.12 Quit Command

The **Quit** command is used to release the editor from the memory and close the editor.

When closing the editor, if there are new files that have not been saved, or programs that have been edited, the following message will be displayed in the message line.

Modified buffers exist, Save all [y / n] ?

When you press ***Y***, the editor closes without saving the modified file.

When you press ***N***, the screen returns to the original.

To close the editor after all files have been saved, execute the **Save and exit** command.

5 Basic Operation of Editor

This chapter describes the fundamental methods for using editor and entering program text.

The following items are explained.

- Entering text and moving cursor
- Deleting and inserting text
- Moving and copying block of text
- Searching and replacing characters, words and statements
- Method for copying text from other files

5.1 Entering Text

There are two ways to write characters in text : " Insert" and " Overwrite "
In the insert mode, the editor inserts the entered characters to the left of cursor.
In the overwrite mode, the character at cursor position is replaced by the entered character.

To convert the insert mode and the overwrite mode, press the ***Alt*** key and hold it down, then press ***Insert*** key.

The setting state of mode can be checked on the status line.
The following message will be shown :

For the insert mode, Overwrite : [OFF]
For the overwrite mode, Overwrite : [ON]

5.2 Selecting Text

When you operate a text program with the editing function, select the range of text first, then specify the part to be edited.

① Move the cursor to the beginning of target text, press the ***Ctrl*** and hold it down, then press ***Space*** key to set the mark.
When the mark is set, the Marked : [ ] in the status line becomes ON.

② Move to the end of text, then execute the edit command.

Once the Mark is set, the texts including from the text where the mark is set to the text where the cursor resident currently turn to the select state all the time.

5.3 Indenting Text

When a text is indented, the structure of program becomes easy to read.
To indent a text, enter ***Space*** or ***Tab*** at the beginning of a line. The indent is set at the column to which the space skipped from the beginning of the line. After this, when the ***Enter*** is pressed, the cursor moves to the same position of the next line.

5.4 Outline of Edit Command

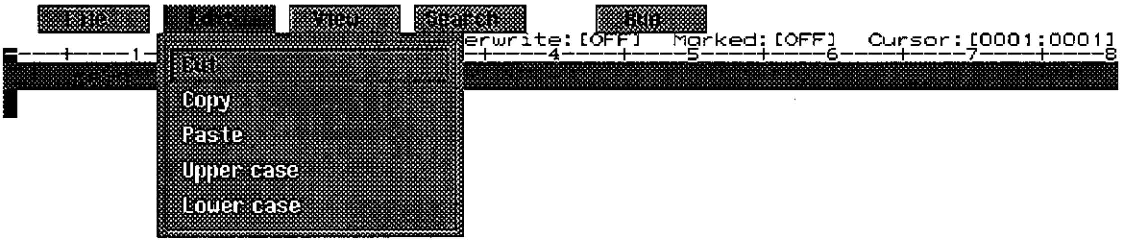
Moving the cursor or an active window can be performed by the simple combined operations of keys. The following is the catalog of edit command.

Table 5-1 Catalog of Edit Command

	Key operation	Explanation
(1)	←	Moves one character left.
	→	Moves one character right.
Moving	***Ctrl + ←***	Moves one word left.
Cursor	***Ctrl + →***	Moves one word right.
	↑	Moves one line up.
	↓	Moves one line down.
	Home	Moves to the beginning of a file.
	End	Moves to the end of a file.
	Ctrl + Home	Moves to the beginning of the cursor line.
	Ctrl + End	Moves to the end of the cursor line.
	Alt + Home	Moves to another window.
	Alt + End	Moves to another window.
	Ctrl + Space	Sets mark at the cursor position.
	Alt + Space	Exchanges between the positions of mark and cursor.
(2)	***Enter***	Inserts change line.
	Insert	Inserts one space.
Insert	***Ctrl + Insert***	Inserts a blank line.
	Shift + Insert	Inserts the content of clip board.
	Alt + Insert	Converts Insert / Overwrite modes.
(3)	***Backspace***	Deletes the character to the left of cursor.
	Delete	Deletes the character at the cursor position.
Delete	***Ctrl + Backspace***	Deletes the word to the left of cursor position.
	Ctrl + Delete	Deletes the word to the right of cursor position.
	Shift + Backspace	Deletes the selected text.
	Shift + Delete	Deletes till the end of a line.
(4)	***Ctrl + ↑***	Scrolls one line upward.
	Ctrl + ↓	Scrolls one line downward.
Scroll	***PageUp***	Scrolls one page upward.
	PageDown	Scrolls one page downward.
	Ctrl + Pageup	Enlarges a window for one line.
	Ctrl + Pagedown	Reduces a window for one line.

6 F2 : Edit menu

The commands, used to write or interchange programs and texts, are prepared in F2 : Edit menu. Using the commands of F2 : Edit menu, you can perform cutting, copying and pasting for a text. It is necessary to select the text to be edited in advance. For the selecting procedure, refer to " 5.2 Selecting Text ".



erwrite: [OFF] Marked: [OFF] Cursor: [0001:0001]

- **Cut** command

  Used to send a selected text to the clip board after it has been cut.

- **Copy** command

  Used to send a copy of a selected text to the clip board.

- **Paste** command

  Used to insert the content of clip board to the cursor position.

- **Upper case** command

  Used to change a selected text to the upper case letters.

- **Lower case** command

  Used to change a selected text to the lower case letters.

6.1 Clip Board

The clip board is a place used to store, cut or copy texts, temporary.
After cutting or copying a text is performed with **Cut** or **Copy** of F2 : Edit menu, the text is stored in the clip board.

The content of clip board can be insert into the cursor position by using the **Paste** command. Only one block can be stored in the clip board at the same time.
The text of clip board can be pasted regardless of how many times.

6.2 Cut Command

This command is used to cut a selected text from the screen and send it to the clip board. To execute the **Cut** command, be sure to perform it after the text has been selected.
When **Cut** and **Paste** commands are used, the movement of line and text block can be performed.

- Moving line and text block

  ① Select a text to be moved. (Refer to 5.2.)

  ② Execute the **Cut** command of F2 : Edit menu.

  ③ Move the cursor to the position where the cut text to be inserted.

  ④ Execute the **Paste** command of F2 : Edit menu.

- Shortcut key : ***Shift*** + ***Backspace***

6.3 Copy Command

This command is used to copy a text just as it is, and send it to the clip board. To execute the **Copy** command, be sure to perform it after the text has been selected.
When **copy** and **Paste** commands are used, a part of program or the whole program can be copied.

- Copying a part of program or the whole program

  ① Select the text to be copied. (Refer to 5. 2.)

  ② Execute the **Copy** command of F2 : Edit menu.

  ③ Move the cursor to the position where the copied text to be inserted.

  ④ Execute the **Paste** command of F2 : Edit menu.

  ⑤ Correct the copied text according to the requirement.

6.4 Paste Command

This command is used to insert the copy of the content of clip board into the cursor position.
The command can be used only when a text is stored in the clip board.

- shortcut key : ***Shift*** + ***Insert***

6.5 Upper case Command

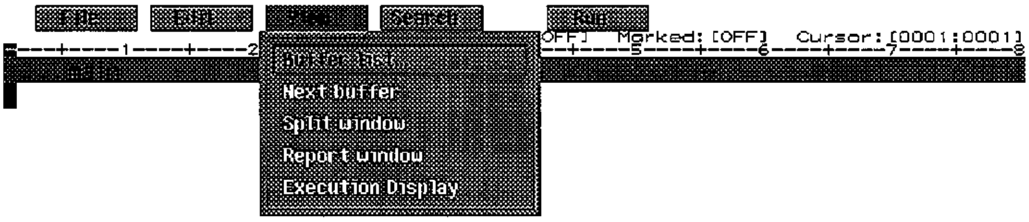
This command is used to change the whole alphabetic characters of the selected text to the upper case letters.
To execute the **Upper case** command, be sure to perform it after the text is selected.

6.6 Lower case Command

This command is used to change the whole alphabetic characters of the selected text to the lower case letters.
To execute the **Lower case** command, be sure to perform it after the text is selected.

7 F3 : View menu

Using F3 : View menu, you can split the view window or edit the content of a loaded file by displaying it in the view window.



- **Buffer list ...** command

  Used to watch programs and buffer catalog of subfiles that have been loaded from the floppy disk.
  In addition, a file can be selected from the buffer catalog and displayed in the view window.

- **Next buffer** command

  Used to display the next file of the buffer in an active window.

- **Split window** command

  Used to split the view window into two parts: the upper and the lower.

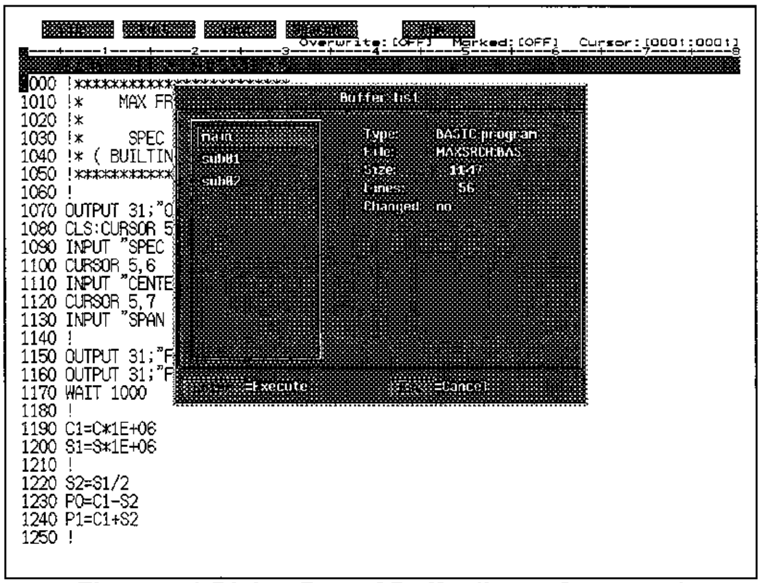
- **Execution display** command

  Used to convert editor screen and measurement screen.

7.1 Buffer list ... Command

When using the **Buffer list ...** command, you can see the catalog of files that have been loaded and select the target file. The selected file is displayed in the view window.

When the **Buffer list ...** is executed, a dialog box as shown in Figure 7-1 appears. Select a file to be edited.



```
                                  Overwrite: [OFF]   Marked: [OFF]   Cursor: [0001:0001]
1000 !************
1010 !*     MAX FR
1020 !*
1030 !*      SPEC
1040 !* ( BUILTIN
1050 !***********
1060 !
1070 OUTPUT 31;"O
1080 CLS:CURSOR 5
1090 INPUT "SPEC
1100 CURSOR 5,6
1110 INPUT "CENTE
1120 CURSOR 5,7
1130 INPUT "SPAN
1140 !
1150 OUTPUT 31;"F
1160 OUTPUT 31;"F
1170 WAIT 1000
1180 !
1190 C1=C*1E+06
1200 S1=S*1E+06
1210 !
1220 S2=S1/2
1230 P0=C1-S2
1240 P1=C1+S2
1250 !
```

Buffer list

main
sub01
sub02

Type: BASIC program
File: MAXSRCH.BAS
Size: 1147
Lines: 56
Changed: no

Figure 7-1 Dialog Box of Buffer list ... Command

- Displaying file

  ① Execute the **Buffer list ...** command of F3 : View menu, a dialog box as shown in Figure 7-1 appears.

  ② Execute one of the following operations after display the target buffer name in high - light with ←, →, ↑ or ↓ keys.

- To display the item shown in high - light in an active window, press ***Enter*** key.
- To cancel the processing, press ***Esc*** key.
- Short cut key : ***Alt*** + ***F3***

7.2 Next Buffer Command

When more than one file have been loaded in the edit memory, if you execute the **Next buffer** command, the next buffer is displayed in active window in alphabetic sequence.
When there is no file is loaded, this command does not function.

- Shortcut key : ***Ctrl*** + ***F3***

7.3 Split Window Command

This command is used to split the view window into two parts : the upper and the lower. When the view window is splitted, you can operate with the two parts of a file viewed at the same time.

- Operating with more than one files seen at the same time

  ① Split the view window with **Split window** command.

  ② A file is displayed in an active window when the **Open subfile** command in F1 :File menu or the **Buffer list ...** command in F3 : View menu is executed. (Active window means a window where the cursor resident.)

- Operating with the view window splitted by **Split window** command

  ① Execute the **Split window** command of F3 : View menu.

  ② When you press ***Home*** holding ***Alt*** down (or press ***End*** holding ***Alt*** down), the cursor moves to the other window. Thus, the cursor resident window becomes an active window.

  ③ If you execute the **Split window** command of F3 : View menu once again, the active window will be enlarged to fill the entire view window and the other which is not activated is closed.

- Changing window size

  To change a window size, activates the window whose size is to be changed, then perform the following operations.

Key operation	Explanation
Ctrl + PageUp	Enlarges active window for one line.
Ctrl + PageDown	Reduces active window for one line.
Shift + F12	Active window is enlarged to fill the entire screen.

7.4 Execution display Command

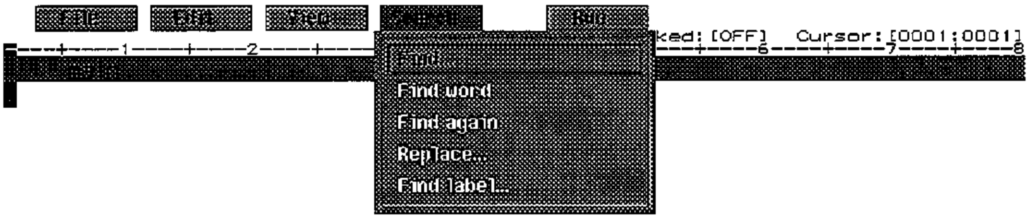
This command is used to display the editor screen and measurement screen of the BASIC alternately. This command can be used at any time in the course of program editing.
Especially, it is convenient to use this command to ensure the result after the program measurement condition has been converted.

- Shortcut key : ***F12***

MEMO

8 F4: Search Menu

Searching a certain character string and replacing the searched character string can be performed by using the command of F4 : Search menu. When a large program is edited, much time is apt to be required to scroll it. In this case, the cursor can jump to the target character string after being searched. In addition, when you want to change a variable name, that can be performed simply by using the replace function.



- **Find ...** command

  Used to search the specified character string and move the cursor to the position behind the searched character string.

- **Find word** command

  Used to search the character string that is the same as the word at the cursor position, then move the cursor to the position behind the searched character string.

- **Find again** command

  Used to search the character string which has been searched just now from the current cursor position.

- **Replace ...** command

  Used to search the specified character string and replace it with a new text.

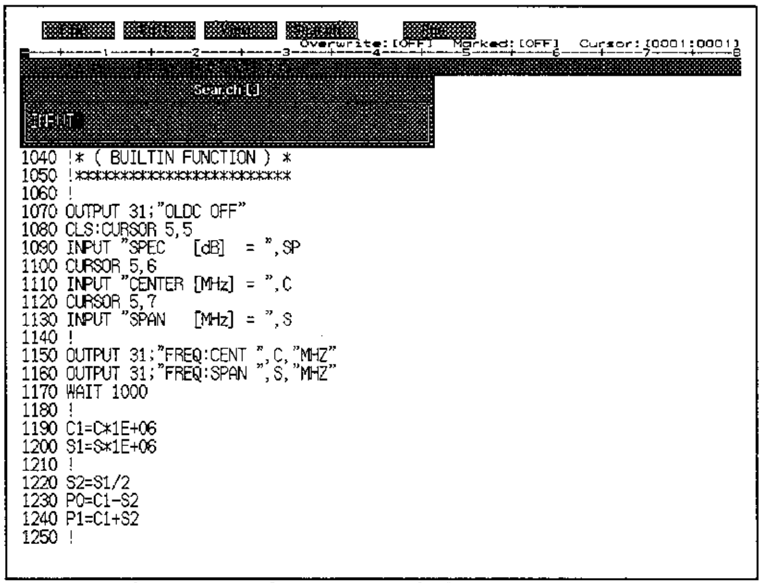
- **Find label ...** command

  Used to search the specified line label.

8.1 Find ... command

This command is used to search a specified character string, starting from the cursor position. When the target character string is found, the cursor is moved to the position behind it .
The character strings that can be searched may be composed of any character (including space etc.), character string and word.

When **Find ...** command is executed, a dialog box as shown in Figure 8-1 appears. Input the character string being searched to the text box.



Overwrite:[OFF] Marked:[OFF] Cursor:[0001:0001]

INPUT

```
1040 !* ( BUILTIN FUNCTION ) *
1050 !************************
1060 !
1070 OUTPUT 31;"OLDC OFF"
1080 CLS:CURSOR 5,5
1090 INPUT "SPEC   [dB]  = ",SP
1100 CURSOR 5,6
1110 INPUT "CENTER [MHz] = ",C
1120 CURSOR 5,7
1130 INPUT "SPAN   [MHz] = ",S
1140 !
1150 OUTPUT 31;"FREQ:CENT ",C,"MHZ"
1160 OUTPUT 31;"FREQ:SPAN ",S,"MHZ"
1170 WAIT 1000
1180 !
1190 C1=C*1E+06
1200 S1=S*1E+06
1210 !
1220 S2=S1/2
1230 P0=C1-S2
1240 P1=C1+S2
1250 !
```

Figure 8-1 Dialog Box of Find ... command

The searching is performed from the cursor position to the whole file displayed in the active window.

- Searching text

  ① Execute the **Find ...** command of F4: Search menu.

  ② Input the text to be searched with a text input file Find text.

When the specified character string is found , the cursor is moved to the position behind the searched character string.

When the specified character string is not found, the message "Not found" is displayed on the lower part of the screen and the cursor is not moved. This message can be cancelled by the next operation.

8.2 Find word Command

This command is used to search a word (sequential alphabetic characters) at the cursor position of an active window.

- Using **Find word** command

  ① Move the cursor to the word being selected. The length of the selected word should be less than one line.

  ② Execute the **Find word** command of F4: Search menu.

8.3 Find again Command

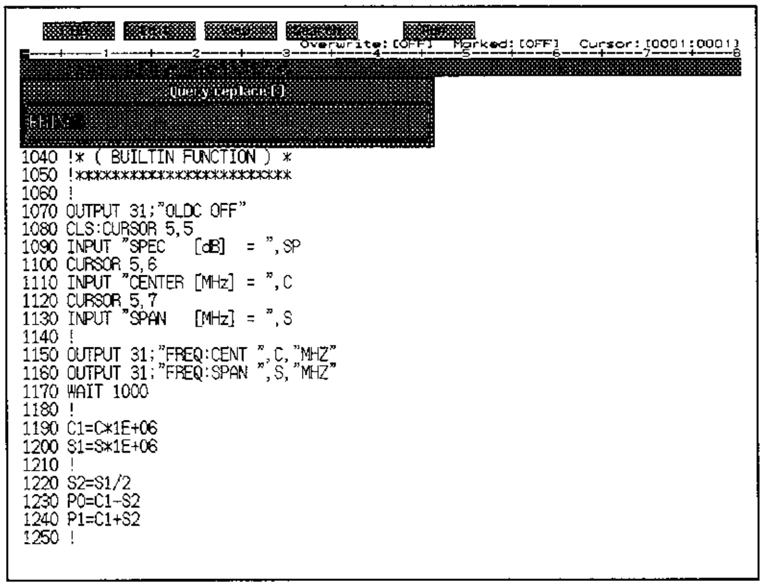
This command is used to search the text once again which has been searched just now.
That is , the searching character string specified by using **Find ...** , **Find word** and **Replace ...** commands becomes symmetrical.
When the searching character string is not entered, the **Find again** command does not work.

- Shortcut key: ***Ctrl*** + ***F4*** (Backward from the cursor position.)
  Alt + ***F4*** (Forward from the cursor position.)

8.4 Replace ... Command

This command is used to search a specified character string and replace it with other character string. The searching character string may be characters, words or the combination of characters and words.

When the **Replace...** command is executed, a dialog box appears as shown in Figure 8-2. Input the searching character string and the replacing character string in this dialog box.



```
1040 !* ( BUILTIN FUNCTION ) *
1050 !************************
1060 !
1070 OUTPUT 31;"OLDC OFF"
1080 CLS:CURSOR 5,5
1090 INPUT "SPEC   [dB]  = ",SP
1100 CURSOR 5,6
1110 INPUT "CENTER [MHz] = ",C
1120 CURSOR 5,7
1130 INPUT "SPAN   [MHz] = ",S
1140 !
1150 OUTPUT 31;"FREQ:CENT ",C,"MHZ"
1160 OUTPUT 31;"FREQ:SPAN ",S,"MHZ"
1170 WAIT 1000
1180 !
1190 C1=C*1E+06
1200 S1=S*1E+06
1210 !
1220 S2=S1/2
1230 P0=C1-S2
1240 P1=C1+S2
1250 !
```

Figure 8-2 Dialog Box of Replace ... Command

- Replacing the specified text

① Execute the **Replace ...** command of F4: Search menu.

② Input the searching character string with the text input file Find-text.

③ Input the replacing character string with the text input file Replace-text.

④ Press ***Enter***, then the replacement of text starts.

⑤ On the character string is found, the following message is displayed in the message line.

Replace ‘ find-text’ with ‘ replace-text’?

When *Y* is pressed, the next searching is performed after the replacement of character string .
When *N* is pressed, the next searching is carried out without performing the replacement.

The following is the list of functions that can be selected in this state.
.

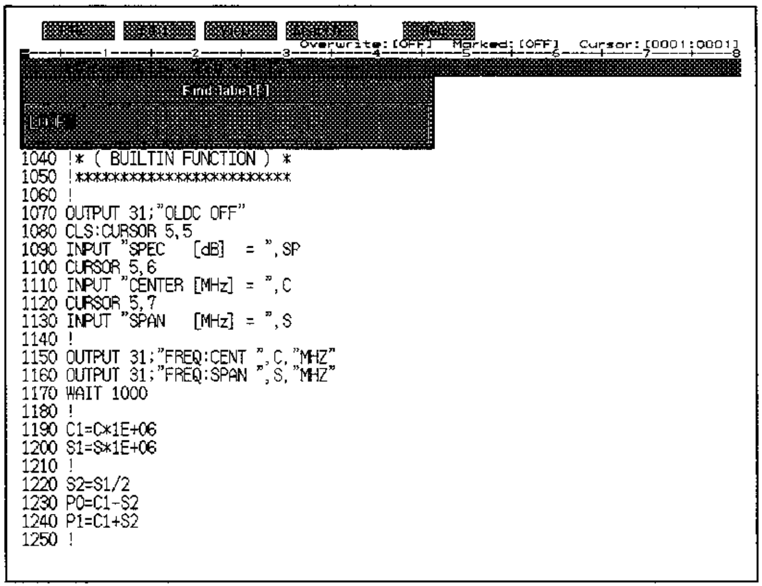
Key operation	Explanation
Esc	Stops searching with the cursor moved to the searched character string.
!	Replaces the remainder in a batch.
?	Displays the list of key operation.
.	Stops searching and returns the cursor to the position where the searching is started.
Y	Replaces the text and searches the next one.
N	Perform the next searching without the text replaced.

When the searching character string is not found. a message “Not found” is displayed in the message line. The message is cancelled by performing the next operation.

8.5 Find label ... Command

This command is used to search a line label of BASIC program.
The line label should be proceeded by an asterisk (*). Therefore, only those texts that entered by using the text input file Find text and an asterisk is added to, can be searched.

When the **Find label ...** command is executed, a dialog box appears as shown in Figure 8-3.

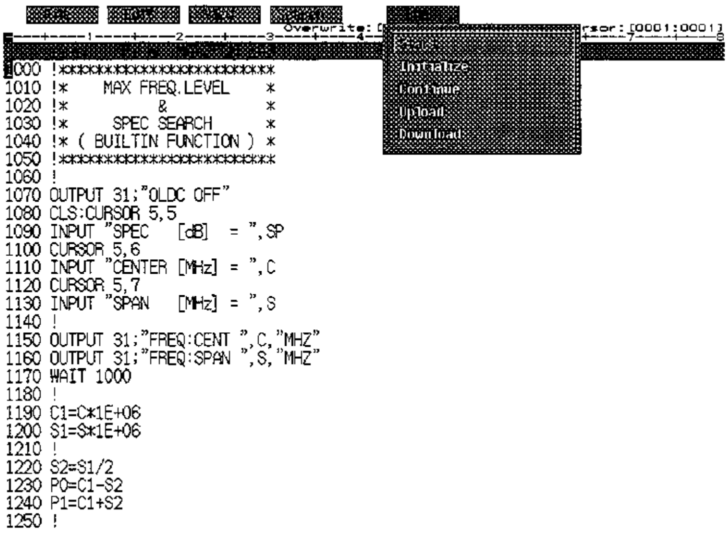


```
1040 !* ( BUILTIN FUNCTION ) *
1050 !************************
1060 !
1070 OUTPUT 31;"OLDC OFF"
1080 CLS:CURSOR 5,5
1090 INPUT "SPEC   [dB]  = ",SP
1100 CURSOR 5,6
1110 INPUT "CENTER [MHz] = ",C
1120 CURSOR 5,7
1130 INPUT "SPAN   [MHz] = ",S
1140 !
1150 OUTPUT 31;"FREQ:CENT ",C,"MHZ"
1160 OUTPUT 31;"FREQ:SPAN ",S,"MHZ"
1170 WAIT 1000
1180 !
1190 C1=C*1E+06
1200 S1=S*1E+06
1210 !
1220 S2=S1/2
1230 P0=C1-S2
1240 P1=C1+S2
1250 !
```

Figure 8-3 Dialog Box of Find label ... Command

9 F5: Run Menu

Several available commands are listed in F5: Run menu to execute a program in the programming environment of BASIC, and perform the Upload from BASIC memory or the Download to BASIC memory and so on.



```
1000 !************************
1010 !*    MAX FREQ.LEVEL     *
1020 !*         &             *
1030 !*     SPEC SEARCH       *
1040 !* ( BUILTIN FUNCTION )  *
1050 !************************
1060 !
1070 OUTPUT 31;"OLDC OFF"
1080 CLS:CURSOR 5,5
1090 INPUT "SPEC   [dB]  = ",SP
1100 CURSOR 5,6
1110 INPUT "CENTER [MHz] = ",C
1120 CURSOR 5,7
1130 INPUT "SPAN   [MHz] = ",S
1140 !
1150 OUTPUT 31;"FREQ:CENT ",C,"MHZ"
1160 OUTPUT 31;"FREQ:SPAN ",S,"MHZ"
1170 WAIT 1000
1180 !
1190 C1=C*1E+06
1200 S1=S*1E+06
1210 !
1220 S2=S1/2
1230 P0=C1-S2
1240 P1=C1+S2
1250 !
```

- **Start** command

  Executes a program.

- **Initialize** command

  Resets all the numeric variables to 0 so as to turn the BASIC to the initial state.

- **Continue** command

  Restarts the program execution from the statement where the execution is interrupted. In this case, the values of variables are not reset.

- **Upload** command

  Reads a program that has been loaded in BASIC memory into the editor.

- **Download** command

  Loads a program in the cause of editing to BASIC memory.

9.1 Start Command

The **Start** command of F5: Run menu is used to run a program.
Loads the program into the BASIC memory first, when a program on the way of editing has not been loaded in BASIC memory or a editing is added.
When there is no line number at the beginning line of a program, the line number is added automatically.

To pause the execution of a program, press ***Pause***.

To restart the program, execute the **Continue** command of F5: Run menu.

To execute the program from the beginning, execute **Start** command.

The screen changes to the measurement screen when the program execution is started , and returns to the editor screen when the program is paused.

- Shortcut key : ***Alt*** + ***F5***

9.2 Initialize Command

The **Initialize** command is used when a program is debugged.
This command resets all the values of variables in a program to 0.
This command is used for preparing program execution, not for running a program.

9.3 Continue Command

The **Continue** command is used for debugging.
When this command is used, a program can be executed from break point and watch point to the next ones.
When a program execution is paused, using this command enables the execution to be restarted from the last executed statement or the program to be executed from the beginning.

- Shortcut key : ***Ctrl*** + ***F5***

9.4 Upload Command

The **upload** command is used to read a program loaded in BASIC memory into the editor.

9.5 Download Command

The **Download** command is used to load a program being edited to the BASIC memory without BASIC program being executed.
The programs being edited with no line number are loaded to BASIC memory with the line number added automatically (The line number can not be directly added to the program on the way of editing.)

10 Automatic Measurement on Network Analyzer

This chapter descries how to create a program to be measured actually with this network analyzer.

(Note) The program presented in this chapter is an example used for R3752/53. When it is used for R3765/67 and R3764/66, the program is required to be changed according to the initial setting and frequency range, etc.

10.1 Program with OUTPUT and ENTER Commands

10.1.1 Executing Program

This program is used to specify a frequency and show a mark at the position, then read the data, displaying the frequency, level and phase. (This program can not be executed with R3752.)

Example 10-1 Program with OUTPUT and ENTER Command

```
100    ! *****************************
110    ! *       OUTPUT / ENTER      *
120    ! *****************************
130    !
140   OUTPUT 31;"OLDC OFF"
150   OUTPUT 31;"MARK:ACT 1, 380E+6"
160   OUTPUT 31;"FETC?"
170.  ENTER 31;F,L,P
180   PRINT "FREQ  [Hz]  = ", F
190   PRINT "LEVEL [dB]  = ", L
200   PRINT "PHASE [deg] = ", P
210   STOP
```

When the program of Example 10-1 is executed with RUN (BASIC command), a mark is shown. The frequency and level of that position are displayed.

```
FREQ  [Hz] = 3.8e + 08
LEVEL [dB] = 0.7818921033
PHASE [deg] = 109.241912841
```

This program shows the example using ceramic band pass filter as DUT (Device Under Test), its center frequency 380MHz.
In R3752, there is no GPIB command to deal with the mark.Therefor, built-in function is used when a wave form analysis is performed with R3752.
The program of Example 10-1 is as follows (This program also can be used for R3752.).
For the details of the built-in functions, refer to 10.2.

Example 10-2 Program with OUTPUT and ENTER Command (using the built- in functions)

```
100  PRINT "FREQ  [Hz] = ",3.8e+8
110  PRINT "LEVEL [dB] = ",CVALUE(3.8e+8,0)
120  PRINT "PHASE [deg]= ",CVALUE(3.8e+8,8)
130  STOP
```

Explanation of Program:

The program flow of " Example 10-1 " is explained below.

Explanation of Example 10-1	
100 to 130	Comment line.
140	Sets GPIB command mode of this instrument.
150	Sets the first mark at 380MHz.
160	The first mark position displayed at line 150 is sent.
170	Receives the data sent from line 160 and assigns necessary data to variables. (Since the frequency and level is necessary here, assigns the frequency to F, the level to L and the phase to P.)
180	Displays the data-entered variable F on the screen with the PRINT statement.
190	Displays the data-entered variable L on the screen with the PRINT statement.
200	Displays the data-entered variable P on the screen with the PRINT statement.
210	Ends the program.

This program is carried out is provided in accordance with the settings after the power is switched on.

Explanation of program command:

The OUTPUT, ENTER and RUN (or !) commands used for the program of Example 10-1 are explained below.

(1) OUTPUT command

OUTPUT device address:	Numeric representation character string representation

The OUTPUT command is used to send data and commands written in numeric or character string to the device specified with the device address number, so as to be executed in the specified device.
The "31" of OUTPUT 31 written in line 140 of this program is the address number . That

shows the sending content to the instrument. OLDC OFF (Setting of IEEE488.2-1987 command mode) is executed in this instrument. The control of this instrument can be performed by using the OUTPUT command and GPIB command.
Furthermore, the other external instrument also can be controlled by changing the device address.

(2) Enter command

ENTER device address : Numeric representation
Character string representation

The ENTER command is used to receive the data through GPIB from the device specified with device address and then assign it to BASIC variables that may be numeric or character string.
In Example 10-1 , the ENTER is used as a response. Here, it is used in the combination with OUTPUT.

```
150 OUTPUT 31;"MARK:ACT 1,380 E+6"
160 OUTPUT 31;"FETC?"
170 ENTER 31;F,L,P
```

- Using OUTPUT command of line 150, sets the first mark at the position of frequency 380 MHz in device address 31 (this instrument) so as to show it .
- Using OUTPUT command of line 160, specifies " FETC ?" to this instrument in the same way. The question mark follows GPIB command closely and it is used when you want to know the values of setting and measurement value (query of data). Here, the data of mark (frequency and level) is asked.
- Using ENTER command of line 170, receives the data of mark and assigns the necessary data to the variables F, L and P . The frequency is assigned to F; the level, to L and the phase, to P.

The contents of sent data are different according to the used GPIB command . For the details , refer to another book " Programming Manual ".

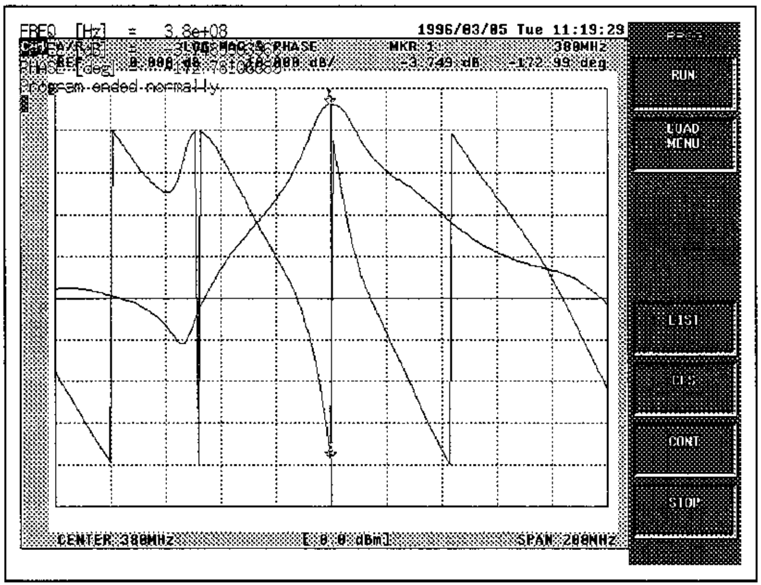
(3) REM command

The REM command is used when a comment line is added to program . All the characters following the REM are considered as a comment statement. REM command can be substituted with ! (exclamation mark).

```
10 REM  PROGRAM1 } Same meaning
10 ! PROGRAM1    }
```

When this program is executed with R3753, it is displayed as shown in Figure10-1.

Execution result:



```
FREQ  [Hz]  =   3.8e+08                          1996/03/05 Tue 11:19:29
            LOG MAG & PHASE                   MKR 1                380MHz
        REF    0.000 dB     10.000 dB/           -3.749 dB    -172.99 deg
PHASE [deg]  =  172.78100000
Program ended normally
  CENTER 380MHz                  [ 0.0 dBm]                  SPAN 200MHz
```

Figure 10-1 Screen Display of Execution Result

Two wave forms are displayed.
The reason is that the measurement FORMAT after the power is switched on adopts the LOG MAG & PHASE traces in R3757.
The ▼ mark is seen at a slightly higher position from the centre, showing the frequency 380MHz at this position.
The uppermost line of graph is taken as reference line.
REF 20.000dB is written on the upper left part of the screen, meaning that the reference value at present is 20dB. However, the interval between each division in graph is 10dB as shown on the right side of the reference value REF in this Figure.
The following is shown on the upper right part of the screen.

```
MKR 1;            380MHZ
  0.781 dB    109.24 deg
```

This means that the level of 380MHz position shown by the mark is 0.781dB.
Since the level value is changed in each scan, it is not limited to the same value for each time. The execution result of this program is also displayed in the figure as follows :

```
FREQ  [Hz] = 3.8e +8
LEVEL [dB] = 0.7818921033
PHASE [deg]  = 109.241912841
```

The value of FREQ (frequency) written as 3.8e+8 is 380,000,000 Hz (Hertz).
The LEVEL is 0.781 dB and the PHASE is 109.24 deg.

10.1.2 Program with USING

In the program of Example 10-1 , FREQ is shown in MHz (megahertz) unit for an easy understand and the values of LEVEL and PHASE are shown with the number that is rounded to three decimal places so as to be seen clearly.

Example 10-3 Display Program with USING

```
100    ! ******************************
110    ! *        OUTPUT / ENTER       *
120    ! *       (GPRINT & USING)      *
130    ! ******************************
140    !
150    OUTPUT 31;"OLDC OFF"
160    OUTPUT 31;"MARK:ACT  1,380E+6"
170    OUTPUT 31;"FETC?"
180    ENTER 31;F,L,P
190    !
200    FR=F/(10^6)
210    !
220    GPRINT " FREQ  [MHz] = ";
230    GPRINT USING "DDDD.DDD";FR
240    GPRINT "LOGMAG  [dB] = ";
250    GPRINT USInG "MDDD.DDD";L
260    GPRINT "PHASE  [deg] = ";
270    GPRINT USING "MDDD.DDD";P
280    STOP
```

When this program is executed, the result becomes as follows.

Execution result:

```
FREQ  [MHz]= 380.000
LEVEL  [dB]=   0.781
PHASE [deg]= 109.241
```

Explanation of Program (GPRINT USING):

The program flow of Example 10-3 is explained below.

Explanation of example 10-3	
100 to 140	Command lines
150	Prompts this instrument to take a new command mode.
160	Sets the first mark of this instrument at 380 MHz
170	Prompts to send the first mark data of this instrument.
180	Assigns the data sent by line 170 to the variables F, L, P.
190	Command line.
200	In order to represent F (frequency) in MHz unit, divides it by 10^6 and assigns the quotient to FR.
210	Command line.
220	Outputs FREQ [MHz] = to a printer. (Do not change the line.)
230	Outputs the value assigned to FR to the printer, positioning it immediately after the last value that is output from line 220 till to the third decimal place. (Changes the line.)
240	Outputs LOGMAG [dB] = to the printer. (Do not change the line.)
250	Outputs the value assigned to L to the printer, positioning it immediately after the last value that is output from line 240 till to the third decimal pace. (Changes the line.)
260	Outputs PHASE [deg]= to the printer. (Do not change the line.)
270	Outputs the value assigned to P to the printer, positioning it immediately after the last value that is output from line 260 till to the third decimal place. (Changes the line.)
280	Ends the program.

A new command is used in program of Example 10-3 that is not used in the program of Example 10-1 . That is the GPRINT USING command used in 220 to 270 lines.
This command can be used in a program not only as the GPRINT USING but also as a GPRINT singly.
GPRINT is almost the same as PRINT command, but the data is output to GPIB board without it displayed on the screen.
That is, the variable and character string enclosed by double quotation " " after the GPRINT can be output from GPIB board. When a printer is connected with this GPIB board, the data also can be output on the printer.
The semicolon ; added at the end of a line means the line does not change. The next output follows the last output without changing the line.

`PRINT formatting command (PRINT USING / GPRINT USING)`

PRINT USING command outputs characters and values according to the image specification determined by the setting of print format . See the line 230 of Example 10-3 program. The "DDDD.DDD" ; FR means that the value assigned to FR is to be printed up to the third decimal place and if the integer part to be printed is only three digits, the remaining position should be filled with a blank (space).
The " MDDD.DDD" in line 250 and line 270 is the same meaning as in line 230. When the value assigned to the variable L or P is negative, adds the minus sign-in front of the value, and when the value is positive, print a blank (space) in stead.
In this example, the execution result is output to printer using the GPRINT USING command.
When the USING is used, the line changement code is added automatically.

10.2 Built-in Functions

The built-in functions are used to compute and analyze the captured measurement data at a high speed with the CPU incorporated.
For these functions, it is not necessary to transfer the data using OUTPUT and ENTER of GPIB program code as it is done previously. Since the operation can be performed with the built-in CPU of this instrument at a high speed directly, so that the time used for the processing can be cut down greatly.

The mark analysis function is not prepared for R3752. Therefore, to analyze the wave form data with R3752, a program that used the built-in functions must be created.
The built-in functions can be used for all types of R3752/53, R3764/66 and R3765/67.

10.2.1 Using Built-in Functions

The built-in functions assign the necessary value to the variables just as other variables used so far have done.

For instance, the format of CVALUE function (specifies frequency value and then evaluates the measurement response value (level) of the frequency.) of built-in functions is shown as follows.

```
CVALUE ( Specifying frequency , Specifying CH  being measured )
```

As an example , a program used to evaluate the level of frequency 380MHz of DUT (device under text) connected with CH1 is shown as follows.

Example 10-4 Program with CVALUE Function

```
100 A=3.8e+8
110 L=CVALUE(A,0)
120 PRINT L
```

In this program, the frequency 380MHz is assigned into variable A first.
Then , the level is evaluated with line 110 in which the above mentioned A is entered as the specifying frequency, and the CH1 connected with DUT is specified to 0 as the CH being measured.
Then, when the level value is evaluated, assigns this value into the variable L .
After that , the value is displayed on the screen by the PRINT Command of line 120.

Thus it can be seen, the built-in functions can be incorporated into an expression, so that it can be used as the same as the normal variable computation.
For the detail of built-in functions, refer to " 4.4 Built-in functions" of R3764/66, R3765/67 programming manual.

10.2.2 Program with Built-in Functions

Here, a program is created using more than one built-in functions.
When the program of Example 10-3 is rewritten with the built-in functions, it becomes as shown below. Where the lines from line 150 to line 190 are changed and others are not.
The execution result is also the same as the original.

Example 10-5 Program with Built-in Functions

```
100  ! ******************************
110  ! *        OUTPUT / ENTER       *
120  ! *       (BUILTIN)             *
130  ! ******************************
140  !
150  OUTPUT 31;"OLDC OFF"
160  AP=POINT1(3.8e+8,0)
170  F=FREQ(AP,0)
180  L=VALUE(AP,0)                            ! 1st data (CH1)
190  P=VALUE(AP,8)                            ! 2nd data (CH1)
200  FR=F/(10^6)
210  !
220  PRINT "FREQ   [MHz] = ";
230  PRINT USING "DDDD.DDD";FR
240  PRINT "LOGMAG  [dB] = ";
250  PRINT USING "MDDD.DDD";L
260  PRINT "PHASE  [deg] = ";
270  PRINT USING "MDDD.DDD";P
280  STOP
```

This program employs the ceramic BPF of 380 MHz as the DUT, just as the program of Example 10-3 did.
The POINT1 function is used in line 160. This function specifies frequency , and calculates the address point where the measurement frequency that is nearest to the specified frequency exists. (Address point is used to specify the analysis range of the measurement data and the position where the data is being measured.It is specified with the value of 0 to 1200 .) Here , the frequency is changed o address point with POINT1 function at the start.
The writing format of POINT function is as follows.

```
POINT1 ( Specifying frequency, Analysis channel )
```

Almost all the built-in functions are used with the similar format of CVALUE and POINT1.
In line 170, using the obtained address, the frequency is evaluated with FREQ function point, and the value is assigned to variable F. Since it has been known that the frequency is 380MHz, it should not have been presented. However, it is used here in order to describe how to obtain the

frequency value from the address point. The writing format o FREQ function is as follows.

```
FREQ  ( Address point, Analysis channel )
```

In line 180, using the assigned variable AP of address point, the amplitude is evaluated and then assigned to variable L with the VALUE function. The format of VALUE function is as follows.

```
VALUE  ( Address point , Analysis channel )
```

In line 190, the phase is determined and the value is assigned to P.

The other lines display the frequency, amplitude and phase on the screen as the same as the program of Example 10-3. (PRINT is used in stead of GPRINT).

The difference in processing speed is not made in such a short program like this. However, when a long and complicated program is created by using the built-in functions, the processing can be performed at high speed.
For the detail of built-in functions and analysis channel, refer to "4.4 Built-in Functions" of R3764/66, R3765/67 programming manual.

10.2.3 Program to Judge Measurement Value

The programs described till now is used to evaluate the amplitude and phase by assigning the specified frequency value into the programs directly.
Here, a new program is introduced. In this program, the center frequency (CENTER) and span value (SPAN) are entered by using INPUT command, then the frequency and amplitude at the maximum amplitude point in the range of the center frequent are evaluated.Furthermore, a decision processing is included in this program to judge whether the amplitude value at the maximum amplitude point has reached the standard value.

Explanation of decision program:

As an initial setting, the following standard values are assigned to the variables with INPUT command.The standard values are used to judge which levels are necessary for CENTER, SPAN and amplitude value.

```
INPUT "SPEC     [dB] = ",SP
INPUT "CENTER [MHz] = ",C
INPUT "SPAN   [MHz] = ",S
```

When executing this , input the SPEC value just as it is . However, since the unit of CENTER and SPAN are MHz, input the data without MHz. For example, input 150 when its value is 150MHz. After the necessary values are entered, input the CENTER and SPAN values in measuring mode.

```
OUTPUT 31;"OLDC OFF"
OUTPUT 31;"FREQ:CENT ";C;"MAHZ"
OUTPUT 31;"FREQ:SPAN ";S;"MAHZ"
```

OLDC OFF is used to set the command mode of GPIB to a new mode. FREQ: CENT and FREQ: SPAN of GPIB program code are used for the frequency setting of CENTER and SPAN.

In the state of OLDC ON , the command names used in R3751 and R3762/63 can be used. However, the new command mode enables the program to be read more easily.

The next is to change the value of CENTER and SPAN which are assigned to the variable C and S from MHz unit to Hz unit.

```
C1=C*1.0e+6
S1=S*1.0e+6
```

The CENTER value in Hz unit is assigned to variable C1 and the SPAN value, to variable S1. The START and STOP values in measuring mode are evaluated before using the built-in functions. After dividing the SPAN value by two, and adding this value to the CENTER value get the STOP value, while subtracting this value from the CENTER value get the START value.

```
S2= S1/2
P0=C1-S2
P1=C1+S2
```

After the START and STOP values have been obtained, the maximum frequency and its amplitude between START and STOP are evaluated with the built-in functions.

```
A=POINT1(P0,0)
B=POINT1(P1,0)
F=FMAX(A,B,0)
L=MAX(A,B,0)
```

First, P0 and P1 are converted to address point that can be used for built-in functions. The starting address point is assigned to A and the stop address point, to B.
Then, the frequency of the maximum amplitude point is evaluated using the FMAX function and the maximum amplitude value is searched using the MAX function.

```
FMAX (Start address point , Stop address point, Analysis channel)
MAX  (Start address point , Stop address point, Analysis channel)
```

The obtained frequency of the maximum amplitude point is assigned to variable F and the amplitude value, to L .
The evaluated values are displayed on the screen using the PRINT command after the processing of built-in functions is ended.

```
FR=F/(10^6)
!
```

```
PRINT "MAX  FREQ [MHz] = ";
PRINT USING "DDDD.DDD";FR
PRINT "MAX  LEVEL [dB] = ";
PRINT USING "MDDD.DDD";L
PRINT "SPEC LEVEL      = " ;
PRINT USING "MDDD.DDD";SP
```

The unit of maximum frequency assigned to variable F is converted from Hz to MHz.
Then, the values are displayed in the sequence of maximum frequency, level, SPEC (standard value). The values of variables are displayed up to the third decimal places, and the PRINT USING command is used to fill up the digits.
Finally, the level of maximum frequency is compared with the input standard value (SPEC). When the maximum frequency is satisfied for SPEC, SPEC OK !! is displayed by the PRINT command, and when it is not satisfied, SPEC NG !! is displayed.

```
IF  L<SP  THEN GOTO  *NG
!
PRINT "*** SPEC OK!! ***"
STOP
!
!  'NG' DISPLAY
!
*NG
PRINT "*** SPEC NG!! ***"
STOP
```

The full decision program is shown as follows.

Example 10-6 Program for Deciding Measurement Value

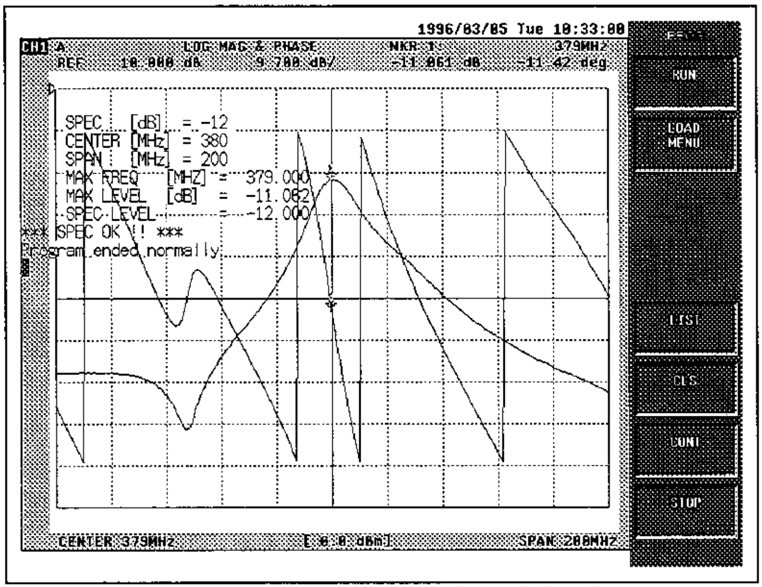
```
100  ! ****************************************
110  !*      MAX FREQ. AND LEVEL SEARCH       *
120  !*               &                       *
130  !*             JUDGE  SPEC               *
140  !*            (BY BUILTIN)               *
150  !*****************************************
160  !
170  OUTPUT 31;“OLDC  OFF”
180  CLS
190  INPUT “SPEC    [dB] = ”,SP
200  INPUT “CENTER [MHz] = ”,C
210  INPUT “SPAN   [MHz] = ”,S
220  !
230  OUTPUT 31;“FREQ:CENT”;C;“MAHZ”
240  OUTPUT 31;“FREQ:SPAN”;C;“MAHZ”
250  !
260  !
270  C1=C*1E+6
280  S1=S*1E+6
290  !
300  S2=S1/2.0
310  P0=C1-S2
320  P1=C1+S2
330  !
340  A=POINT1(P0,0)
350  B=POINT1(P1,0)
360  F=FMAX(A,B,0)
370  L=MAX(A,B,0)
380  !
390  FR=F/(10.0^6)
400  OUTPUT 31;“MARK:ACT 1,”;FR
410  OUTPUT 31;“MARK:LET CENT”
420  !
430  !
```

```
440  PRINT "MAX FREQ  [MHz] = ";
450  PRINT USING "DDDD.DDD";FR
460  PRINT "MAX LEVEL  [dB] = ";
470  PRINT USING "MDDD.DDD";L
480  PRINT "SPEC LEVEL [dB] = ";
490  PRINT USING "MDDD.DDD";SP
500  !
510  IF L<SP THEN GOTO *NG
520  !
530  PRINT "*** SPEC OK !!***"
540  STOP
550  !
560  !  'NG' DISPLAY
570  !
580  *NG
590  PRINT "***  SPEC NG !!*** "
600  STOP
```

When performing Example 10-6 , the SPEC value is inquired first.
Since the ceramic filter of 380MHz is used as DUT here, the measurement is performed with SPEC level (SPEC) -10dB, CENTER 380MHz and SPAN 200MHz.

When Example 10-6 is executed, the result is as follows.

Execution result:



```
SPEC   [dB]  = -12
CENTER [MHz] = 380
SPAN   [MHz] = 200
MAX FREQ   [MHZ] =  379.000
MAX LEVEL  [dB]  =  -11.062
SPEC LEVEL       =  -12.000
*** SPEC OK !! ***
Program ended normally
```

Figure 10-2 Execution Result of Decision Program

10.2.4 Output to Parallel I/O Port

In the program of Example 10-6 , the decided result is displayed as " OK !! " or " NG !! " by using PRINT command , but here it is output through the use of parallel I/O port. Using INPUT 1 (External input) of parallel I/O port, the program is created which starts the measurement with the trigger switch.
As a circuit example, the circuit diagram of the guide of this instrument" Operating Program According to Trigger Switch of Parallel I/O " is employed.
(Refer to the "Function Explanation and Communication with Peripheral Apparatus" in the manuals of various machines.

Creating Program:

① The setting of port mode is added between line 170 and line 180 of the program in Example 10-6.

```
171 OUTPUT 36;16
172 OUTPUT 35;80
173 OUTPUT 35;112
```

- In line 171, since the parallel I/O port is used, the setting of port mode is performed.
  Here , all of A , B, C and D port are set to output port.
- In line 172 and line 173, resets OUTPUT1 and OUTPUT2 . That is , switch off the LED (OUTPUT1) being measured and LED (OUTPUT2) to be measured.

② A program is added between line 260 and line 270 to wait the starting of measurement until the switch is pressed.

```
261 OUTPUT 35;48
262 ENTER 34;A
263 WAIT 500
264 IF A< >1 THEN GOTO 262
265 OUTPUT 35;112
```

- In line 261, OUTPUT2 is turned to the state of set and the LED is lighted up. This means waiting measurement (READY), that is, showing the waiting of switch input with the LED.
- In line 262 and line 264 , the loop is continued till the switch input is done so as to stop the program sequence. When the switch is not pressed, nothing is assigned to the numeric variable A of line 262, and proceeded to line 273 just as 0 it is .
- The WAIT of line 263 is used to keep a short interval for performing the switch input properly (WAIT time : msec ; 0 to 65535)
- In line 264 , when the switch is not pressed, the variable A is 0. If the conditional expression holds, the operation is repeated till the switch is pressed.
  When the switch is pressed, 1 is assigned to the numeric variable A according to the specification of ENTER 34; A (External input) in line 262. In this case, the conditional expression does not hold and the operation proceeds to line 265.
- In line 265, LED of OUTPUT2 is shut off. This means the switch is input and the system is not in the state of waiting measurement.
  If the switch is pressed , the LEDs of OUTPUT1 and OUTPUT2 are lighted up.

The LED of OUTPUT2 is shut off in line 265, while OUTPUT1 is lighted up all the time by reason that no specification is made for it . This indicates the measurement is in progress.

③ The program used to shut off the LED, when the measurement is stopped, is added between line 430 and line 440.

```
431 OUTPUT 35;80
```

- In line 431 , LED of OUTPUT1 is turned to the reset state and the LED is shut off.

④ The I/O port output that is produced when the decision is performed with SPEC value is shown simply with OK or NG. The OK means output to port A and the NG, to port B. OK output is added between line 530 and line 540 , and NG output, between line 590 and line 600.

```
531 OUTPUT 33;1

591 OUTPUT 34;1
```

- When the judgement is OK, outputs 1 to port A. When NG, outputs 1 to port B. Add the LED, to No.5 (1 of port A) and No.13 (1 of port B) of the parallel I/O connector (refer to " Function Explanation-Communication with peripheral apparatus " of various machine manual). The result of output , i.e., OK or NG can be seen according to which LED has been lighted up.

The program list is shown in Example 10-7.

Example 10-7 Program for Deciding Measurement Value (Using parallel I/O port)

```
100  !*****************************************
110  !*      MAX FREQ. AND LEVEL SEARCH        *
120  !*                &                       *
130  !*            JUDGE SPEC                  *
140  !*            (BY BUILTIN)                *
150  !*****************************************
160  !
170  OUTPUT 31;"OLDC OFF"
171  OUTPUT 36;16              ! A,B,C,D -> OUTPUT
172  OUTPUT 35;80              ! RESET OUTPUT1
713  OUTPUT 35;112             ! RESET OUTPUT2
180  CLS
190  INPUT "SPEC     [dB] = ",SP
200  INPUT "CENTER [MHz] = ",C
210  INPUT "SPAN   [MHz] = ",S
220  !
230  OUTPUT 31;"FREQ:CENT";C;"MAHZ"
```

```
240  OUTPUT 31;“FREQ:SPAN”;C;“MAHZ”
250  !
260  !
261  OUTPUT 35;48                 ! SET OUTPUT1 AND OUTPUT2
262  ENTER 34;A                   ! READ OUTPUT1 (INPUT1)
263  WAIT 500                     ! WAIT 500MSEC
264  IF A< >1 THEN GOTO 262       ! CHECK TRIGGER SWITCH INPUT
265  OUTPUT 35;112                ! RESET OUTPUT2
270  C1=C*1E+6
280  S1=S*1E+6
290  !
300  S2=S1/2.0
310  P0=C1-S2
320  P1=C1+S2
330  !
340  A=POINT1(P0,0)
350  B=POINT1(P1,0)
360  F=FMAX(A,B,0)
370  L=MAX(A,B,0)
380  !
390  FR=F/(10.0^6)
400  OUTPUT 31;“MARK:ACT 1, ”;FR
410  OUTPUT 31;“MARK:LET CENT”
420  !
430  !
431  OUTPUT 35;80                  ! RESET OUTPUT1
440  PRINT “MAX PREQ  [MHz] = ”;
450  PRINT USING “DDDD.DDD”;FR
460  PRINT “MAX LEVEL  [dB] = ”;
470  PRINT USING “MDDD.DDD”;L
480  PRINT “SPEC LEVEL [dB] = ”;
490  PRINT USING “MDDD.DDD”;SP
500  !
510  IF L<SP THEN GOTO *NG
520  !
530  PRINT “*** SPEC OK !! ***”
```

```
531  OUTPUT 33;1                      ! WRITE 1 TO PORT-A
540  STOP
550  !
560  !  'NG' DISPLAY
570  !
580  *NG
590  PRINT "*** SPEC NG !! ***
591  OUTPUT 34;1                      ! WRITE 1 TO PORT-B
600  STOP
```

MEMO

11 Program Examples for Wave Form Analysis

This chapter describes program examples for wave form analysis and the using method of built-in functions by turns.

(Note) ***The programs presented in this chapter are the examples used for R3752/53 .***
To use them on R3765/67 and R3764/66 , it is required to be changed according to the difference of initial setting state and the frequency range.

11.1 MAX and MIN Level Automatical Measurement Program

The example of program used to measure automatically the maximum and minimum level value by the use of built-in function MAX and MIN is shown in Example 11-1.

Example 11-1 Automatical Measurement Program of MAX and MIN level

```
1000  !*******************************************
1010  !
1020  !         MAX-MIN  LEVEL  MEASUREMENT
1030  !
1040  !*******************************************
1050  *MAIN
1060      GOSUB  *SETUP
1070      GOSUB  *CAL
1080      CLS
1090      *MEAS_LOOP
1100          GOSUB *MEAS
1110          GOSUB *RESULTS
1120          GOTO  *MEAS_LOOP
1130  !
1140  !----------------------------
1150  *SETUP
1160      OUTPUT 31;"LDC OFF"
1170      OUTPUT 31;" DISP:ACT 1;:FUNC1:POW AR;:CALC:FORM MLOP"
1180      OUTPUT 31;"DISP:Y:PDIV 10"
1190      OUTPUT 31;"DISP:Y:RPOS 10"
1200      OUTPUT 31;"DISP:Y:RLEV 0"
1210      OUTPUT 31;"POW ODBM"
1220      !
1230      OUTPUT 31;"SWE:POIN 201"
```

```
1240        OUTPUT 31;“FREQ:STAR 100MAHZ”
1250        OUTPUT 31;“FREQ:STOP 200MAHZ”
1260        RETURN
1270    !
1280    !--------------------------------
1290    *CAL
1300        CURSOR 6,9
1310        PRINT “CONNECT [THROUGH]”
1320        CURSOR 6,10
1330        INPUT “IF OK THEN PRESS ‘ENT’ or ‘X1’”,D$
1340        OUTPUT 31;“CORR:COLL NORM;*OPC?”
1350        ENTER 31;A
1360        RETURN
1370    !
1380    !---------------------------------
1390    *MEAS
1400        CURSOR 5,10
1410        PRNT “CONNECT DUT”
1420        CURSOR 5,11
1430        INPUT “IF OK THEN PRESS ‘ENT’ or ‘X1’”,D$
1440        !
1450        MAX_DT=MAX(0,1200,0)
1460        MIN _DT=MIN(0,1200,0)
1470        RETURN
1480    !
1490    !----------------------------------
1500    *RESULTS
1510        CLS
1520        CURSOR 5,15
1530        PRINT “MAX VALUE [dB] = ”;
1540        PRINT USING “3D.3D”;MAX_DT
1550        CURSOR 5,16
1560        PRINT “MIN VALUE [dB] = ”;
1570        PRINT USING “3D.3D”;MIN_DT
1580        RETURN
```

The MAX function used in this program searches the maximum response value, while the MIN function searches, the minimum response value.
These functions are also used to evaluate the response value of resonance point and anti-resonance point. The program of Example 11-1 is explained below.

Program Explanation of Example 11-1	
1000 to 1040	Comment lines
1050	Level MAIN of main routine.
1060	Calls out initial setting routine SETUP.
1070	Calls out calibration routine CAL.
1080	Clears the screen.
1090	Level MEAS_LOOP of measurement repetition loop.
1100	Calls out measurement routine MEAS.
1110	Calls out display routine RESULT.
1120	Loop of measurement.
1130 to 1140	Comment lines.
1150	Level SETUP of initial setting routine.
1160	Releases IEEE 488.1-1987 command mode.
1170	Sets active channel to CH1, input port to A/R and format to LOGMAG & PHASE.
1180	Sets the scan resolution to 10dB.
1190	Sets the reference position to 10%.
1200	Sets the reference level to 0dB.
1210	Sets the output level to 0dBm.
1220	Comment line.
1230	Sets the point count to 201 points.
1240	Sets the scan start frequency to 100MHz.
1250	Sets the scan stop frequency to 200MHz.
1260	Gets rid of initial setting routine.
1270 to 1280	Comment lines.
1290	Level CAL of calibration routine.
1300	Moves the cursor.
1310	Displays the message “ CONNECT [THROUGH] ” .

1320	Moves the cursor.
1330	Displays the message “ IF OK THEN PRESS ‘ENT’ or ‘X1’ ” and waits the input.
1340	Executes calibration.
1350	Waiting till the calibration is ended.
1360	Gets rid of the calibration routine.
1370 to 1380	Comment lines.
1390	Level MEAS of measurement routine.
1400	Moves the cursor.
1410	Displays the message “ CONNECT DUT ”.
1420	Moves the cursor.
1430	Displays the message “ IF OK THEN PRESS ‘ ENT ’ or ‘ X1 ’ and waits the input.
1440	Comment line.
1450	Gets the maximum value.
1460	Gets the minimum value.
1470-	Gets rid of the measurement routine.
1480 to 1490	Command lines.
1500	Level RESULTS of display routine.
1510	Clears the screen.
1520 to 1540	Comment line.
1550 to 1570	Moves the cursor and displays the minimum value.
1580	Gets rid of the display routine.

11.2 Ceramic Filter Automatic Measurement Program

The program examples used to evaluate the insertion loss and the frequency of 3dB bandwidth are shown in Example 11-2.

Example 11-2 Ceramic Filter Automatic Measurement Program

```
1000  !*******************************************
1010  !
1020  !         CERAMIC FILTER MEASUREMENT
1030  !
1040  !*******************************************
1050  *MAIN
1060      GOSUB  *SETUP
1070      GOSUB  *CAL
1080      CLS
1090      *MEAS_LOOP
1100          GOSUB *MEAS
1110          GOSUB *RESULTS
1120          GOTO  *MEAS_LOOP
1130  !
1140  *SETUP
1150      NA=31
1160      OUTPUT NA;"OLDC OFF"
1170       OUTPUT NA;"SYST:PRES;:INIT:CONT OFF;:STAT:OPER:ENAB 8;*ESB
                                                              128;*OPC?"
1180      ENTER NA;A
1190      OUTPUT NA;"FREQ:SPAN 10KHZ"
1200      OUTPUT NA;"FREQ:CENT 10.7MAHZ"
1210      SPOLL(NA)
1220      RETURN
1230  !
1240  *CAL
1250      CURSOR 6,9
1260      PRINT "CONNECT [THROUGH]"
1270      CURSOR 6,10
1280      INPUT "IF OK THEN PRESS 'ENT' or 'X1'",D$
1290      OUTPUT NA; "CORR:COLL NORM;*OPC?"
1300      ENTER NA;A
1310      RETURN
1320  !
```

```
1330  *MEAS
1340      CURSOR 6,25
1350      PRINT "CONNECT DUT"
1360      CURSOR 6,26
1370      INPUT "IF OK THEN PRESS 'ENT' or 'X1'",D$
1380  !
1390      ON ISRQ GOTO *LPOUT
1400      ENABLE INTR
1410      OUTPUT NA;"INIT"
1420      *LP
1430          GOTO  *LP
1440  !
1450  *LPOUT
1460      SPOLL(NA)
1470      DISABLE INTR
1480      I_LOSS=MAX(0,1200,0)
1490      MAX_F=FMAX(0,1200,0)
1500      BW3DB=CBND(MAX_F,3,0)
1510      RETURN
1520  !
1530  *RESULTS
1540      CURSOR 5,4
1550      PRINT "I LOSS [dB]          = ";
1560      PRINT USING "3D.3D";I_LOSS
1570      CURSOR 5,5
1580      PRINT "3 DB BAND WIDTH [MHz] = ";
1590      PRINT USING "3D.3D";BW3DB/1E+6
1600      RETURN
```

The CBND function employed in the program of Example 11-2 is used to evaluate the band width. It searches those points that attenuated only for the specified attenuation level at the specified frequency, and then evaluates the band width .
The searching is performed from the specified address point towards outside.
In this program, the frequency of maximum response is evaluated with the FMAX function, and the band width of 3dB down is evaluated from this frequency with the CBND function.

Program Explanation of Example 11-2	
1000 to 1040	Comment lines
1050	Level MAIN of main routine.
1060	Calls out the initial setting routine SETUP.
1070	Calls out calibration routine CAL.
1080	Clears the screen.
1090	Level MEAS_LOOP of measurement repetition loop.
1100	Calls out measurement routine MEAS.
1110	Calls out display routine RESULT.
1120	Loop of measurement.
1130 to 1140	Comment lines.
1150	Level SETUP of the initial setting routine.
1160	Releases IEEE 488.1-1987 command mode.
1170	After pre-sets this instrument, turns it to the single scan mode, and sets it to such a state that enables to produce SRQ requirement when the scan is ended.
1180	Waiting until the end of setting.
1190	Sets the frequency of scan center to 10KHz.
1200	Sets the frequency SPAN of scan to 10.7MHz.
1210	Performs serial poll and drops RSV bit.
1220	Gets rid of initial setting routine.
1230	Comment line.
1240	Level CAL of calibration routine.
1250	Moves the cursor
1260	Displays message " CONNECT [THROUGH] " .
1270	Moves the cursor.
1280	Displays the message " IF OK THEN PRESS ' ENT ' or 'X1' " and waits the input.
1290	Executes calibration.
1300	Waiting until the end of calibration.
1310	Gets rid of calibration routine.

1320	Comment line.
1330	Level MEAS of measuring routine.
1340	Moves the cursor.
1350	Displays message " CONNECT DUT" .
1360	Moves the cursor.
1370	Displays the message " IF OK THEN PRESS 'ENT' or 'X1' " and waits the input.
1380	Comment line.
1390	Specifies the branch destination of service request interrupting .
1400	Enables interrupting.
1410	Executes the scan for one times.
1420 to 1430	Repeats interruption waiting loop.
1440	Comment lines.
1450	Branch destination level LPOUT of service request interrupt.
1460	Performs serial poll and drops RSV bit.
1470	Disables interrupting.
1480	Evaluates maximum value of level with MAX function and assigns it to variable I_LOSS.
1490	Evaluates the measurement frequency of the maximum level with FMAX function and assigns it to variable MAX_F.
1500	Evaluates the band width of 3dB with CBAND function and assigns it to variable BW3DB.
1510	Gets rid of the interrupt processing routine.
1520	Comment line.
1530	Level RESULTS of display routine.
1540 to 1560	Moves the cursor and displays the insertion loss value by moving.
1570 to 1590	Moves the cursor and displays the frequency of 3dB band width.
1600	Gets rid of the display routine.

11.3 Ripple Analysis Program

An example of program using the ripple function is shown in Example 11-3.

Example 11-3 Ripple Analysis Program

```
1000  !********************************************
1010  !
1020  !             RIPPLE MEASUREMENT
1030  !              (NO USED SRQ)
1040  !
1050  !********************************************
1060  DIM  PR1$[25],PR2$[25],PR3$[25]
1070  !
1080  *MAIN
1090      GOSUB *SETUP
1100      CLS
1110      *MEAS_LOOP
1120          GOSUB *MEAS
1130          GOSUB *RESULTS
1140          GOTO  *MEAS_LOOP
1150  !
1160  *SETUP
1170      NA=31
1180      OUTPUT  NA;“OLDC OFF”
1190      OUTPUT  NA;“SYST:PRES;:INIT:CONT OFF”;
1200      OUTPUT  NA;“DISP:FORM ULOW”
1210      OUTPUT  NA;“CALC:FORM MLOD”
1220      OUTPUT  NA;“FREQ:CENT 17.9MAHZ;SPAN 30KHZ“
1230      OUTPUT  NA;“BAND 1KHZ“
1240      OUTPUT  NA;“SWE:TIME 1SEC”
1250      RETURN
1260  !
1270  *MEAS
1280      CURSOR 6,25
1290      PRINT “CONNECT DUT”
1300      CURSOR 6,26
1310      INPUT “IF OK THEN PRESS ‘ENT‘ or ‘X1’”,D$
1320  !
1330      OUTPUT NA;“INIT;*OPC?”
```

```
1340        ENTER NA;DUMMY$
1350        OUTPUT NA;“DISP:Y8 AUTO”
1360    !
1370        A1=PMAX(0,1200,0)
1380        A2=BNDL(A1,3,0)
1390        A3=BNDH(A1,3,0)
1400        A4=POINT2(A2,0)
1410        A5=POINT2(A3,0)
1420        B1=RPL2(A4,A5,1,0.001,0)                ! LOGMAG RIPPLE
1430        B2=RPL4(A4,A5,1,0.001,0)
1440        IF B1<B2 THEN
1450            B3=B2
1460        ELSE
1470            B3=B1
1480        END  IF
1490        C1=RPL2(A4,A5,1,1e-08.8                 ! DELAY RIPPLE
1500        C2=RPL4(A4,A5,1;1e-08.8)
1510        IF C1<C2 THEN
1520            C3=C2
1530        ELSE
1540            C3=C1
1550        END  IF
1560        RETURN
1570    !
1580    *RESULTS
1590        PR1$=“LOGMAG RIPPLE [dB] =”
1600        CURSOR 0,16:PRINT USING“k,M2D.5D”;PR1$,B3
1610        PR2$=”DELAY RIPPLE [us] =”
1620        CURSOR 0,17:PRINT USING “k,M2D.5D”;PR2$,C3*10^6
1630        RETURN
```

First, the frequency range is evaluated with PMAX, BNDL and BNDH in this program. The PMAX function evaluates the measurement point of the maximum response and calculates the band width from the measurement point. The BNDL function evaluates the frequency on the side of low frequency band and the BNDH function evaluates the frequency on the side of high frequency band. then , the ripple analysis is specified with this frequency range.

Next , the ripple analysis is performed after the frequency has been converted to address point with POINT2 function. There are various ripple analysis functions. RPL2 and RPL4 are used in this program. Both of the functions are used to evaluate the maximum values of the neighboring highest value and lowest value, but the ways with which they get a pair of highest value and lowest value are different. The highest value is on the side of low frequency in RPL2 function while it is on the side of high frequency in RPL4 function.

The program of Example 11-3 is explained below.

Program Explanation of Example 11-2	
1000 to 1050	Comment lines
1060	Defines character string array.
1070	Comment line.
1080	Level MAIN of main routine
1090	Calls out initial setting routine SETUP.
1100	Clears the screen.
1110	Level MEAS_LOOP of measurement repetition loop.
1120	Calls out measurement routine MEAS.
1130	Calls out display routine RESULT.
1140	Loop of measurement.
1150	Comment line.
1160	Level SETUP of initial setting routine.
1170	Assigns address 31 to variable NA .
1180	Releases IEEE488. 1-1987 command mode.
1190	Presets this instrument and turns it to single scan mode.
1200	Sets split screen mode.
1210	Sets calculation format to LOGMAG & DELAY
1220	Sets the frequency of scan centre to 17.9MHz, and SPAN to 30kHz.
1230	Sets the resolution band width to 1kHz.
1240	Sets the scan time to one second.
1250	Gets rid of initial setting routine.
1260	Comment line.
1270	Level MEAS of measurement routine.
1280	Moves the cursor.
1290	Displays the message " CONNECT DUT"
1300	Moves the cursor.

1310	Displays the message " IF OK THEN PRESS 'ENT' or 'X1' " and waits the input.
1320	Comment line.
1330	Executes the scan for one times and sends OPC quarry.
1340	Waiting until the end of scan.
1350	Performs the automatic setting of Y axis.
1360	Comment line.
1370	Evaluates the measurement point of maximum level and assigns it to variable A1.
1380	Evaluates the frequency on the low frequency side of 3dB band width , and assigns it to variable A2.
1390	Evaluates the frequency on the high frequency side of 3dB band width , and assigns it to variable A3.
1400	Converts frequency A2 to address point. and assigns it to A4.
1410	Converts frequency A3 to address point, and assigns it to A5.
1420	Evaluates the maximum value of the neighboring highest value and lowest value from the amplitude data with RPL2 function.
1430	Evaluates the maximum values of the neighboring highest value and lowest value from the amplitude data with RPL4 function.
1440 to 1480	Assigns the large value from among them to variable B3
1490	Evaluates the maximum values of the neighboring highest value and lowest value from the delay data with RPL2 function.
1500	Evaluates the maximum values of the neighboring highest value and lowest value from the delay data with RPL4 function.
1510 to 1550	Assigns the large value from among them to variable C3.
1560	Gets rid of the measurement routine.
1570	Comment line.
1580	Level RESULTS of display routine.
1590 to 1600	Displays the ripple analysis value of amplitude data.
1610 to 1620	Displays the ripple analysis value of delay data.
1630	Gets rid of display routine.

11.4 Example of Band-pass Filter Measurement

In this section , the measurement of band-pass filter of central frequency : 10.7MHz is taken as an example to explain the filter analysis program. The program example is shown in Example 11-4.

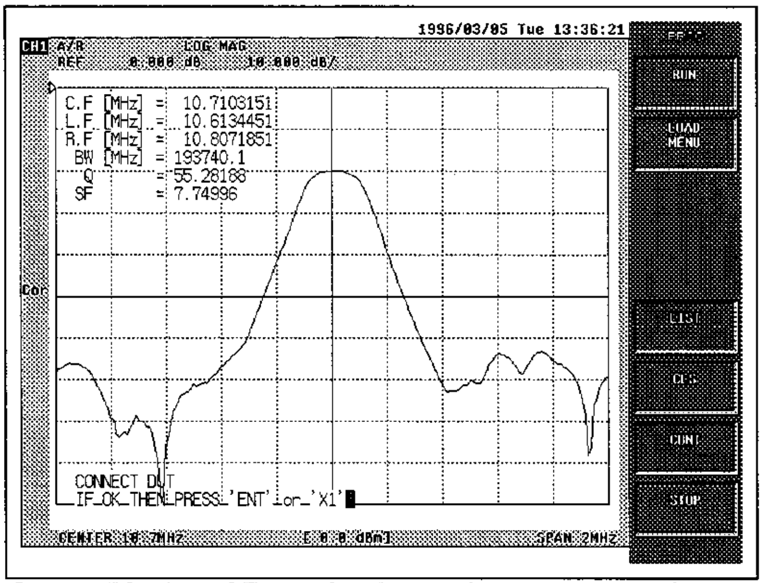
Example 11-4 Measurement of Band-pass Filter

```
1000  !********************************************
1010  !
1020  !          BAND  PASS  FILTER  ANALYSIS
1030  !          f=10.7MHz
1040  !
1050  ! FILE : BPF.BAS
1060  !********************************************
1070  *MAIN
1080      GOSUB *SETUP
1090      GOSUB *CAL
1100      CLS
1110      *MEAS_LOOP
1120          GOSUB *MEAS
1130          GOSUB *RESULTS
1140          GOTO  *MEAS_LOOP
1150  !
1160  *SETUP
1170      INTEGER EV
1180      DIM L(2),F(2,4)
1190      NA=31 :EV=1 :L(1)=3.0 :L(2)=60.0
1200      OUTPUT NA;"OLDC OFF"
1210      OUTPUT NA;"SYST:PRES;:INIT:CONT OFF;:STAT:OPER:ENAB 8;*SRE
                                                              128;*OPC?"
1220      ENTER NA;A
1230      OUTPUT NA;"CALC:FORM MLOG"
1240      OUTPUT NA;"FREQ:SPAN 2MHZ;CENT 10.7MAHZ"
1250      RETURN
1260  !
1270  *CAL
1280      CURSOR 6,9  :PRINT "CONNECT [THROUGH]"
1290      CURSOR 6,10 :INPUT "IF OK THEN PRESS 'ENT' or 'X1'",D$
```

```
1300      OUTPUT NA;“CORR:COLL NORM;*OPC?″ :ENTER NA;A
1310      RETURN
1320  !
1330  *MEAS
1340      CURSOR 6,25 :PRINT “CONNECT DUT″
1350      CURSOR 6,26 :INPUT “IF OK THEN PRESS ‘ENT′ or ‘X1′″,D$
1360      OUTPUT NA;“INIT″ :WAIT EVENT EV
1370      AP=PMAX(0,1200,0)
1380      NP=MBNDI(0,1200,AP,2,L(1),F(1,1),0)
1390      QF=F(1,3)/F(1,4)                       ! QF = CF(3dB)   / BW (3dB)
1400      SF=F(2,4)/F(1,4)                       ! SF = BW′(60dB) / BW (3dB)
1410      RETURN
1420  !
1430  *RESULTS
1440      CURSOR 5,4 :PRINT “C.F [MHz]=″; :PRINT USING “3D.7D″;F(1,3)
                                                                 /1.0E+6
1450      CURSOR 5,5 :PRINT “L.F [MHz]=″; :PRINT USING “3D.7D″;F(1,1)
                                                                 /1.0E+6
1460      CURSOR 5,6 :PRINT “R.F [MHz]=″; :PRINT USING “3D.7D″;F(1,2)
                                                                 /1.0E+6
1470      CURSOR 5,7 :PRINT “ BW [ Hz]=″; :PRINT USING “5D.1D″;F(1,4)
1480      CURSOR 5,8 :PRINT “  Q      =″; :PRINT USING “.5D″;QF
1490      CURSOR 5,9 :PRINT “ SF      =″; :PRINT USING “.5D“;SF
1500      RETURN
```

When this program is performed, the screen display of this instrument becomes as shown in Figure 11-1.

Execution result:



```
C.F [MHz] =  10.7103151
L.F [MHz] =  10.6134451
R.F [MHz] =  10.8071851
 BW [MHz] = 193740.1
  Q       = 55.28188
 SF       = 7.74996

CONNECT DUT
IF_OK_THEN_PRESS_'ENT'_or_'X1'
```

Figure 11-1 Screen Display of Execution Result (Measurement of Band-pass Filter)

After the measurement point of the maximum response value has been found with PMAX function, evaluate the band width and frequency of the two measurement points with the MBND function whose attenuation is 3dB and 60dB, respectively.

Using MBND function, the analysis of more than one attenuations level can be performed at a time, so that the frequency of low frequency band, frequency of high frequency band, the center frequency and the band width can be obtained, respectively.

11.5 Example of Crystal Resonant Point Measurement

In this section, the program seeking from the measurement of transferring resonant point and anti-resonant point of ceramic oscillator (f = 45.1 Mhz) is explained .

The example of program is shown in Example 11-5 .

Example 11-5 Measurement of Crystal Resonant Point

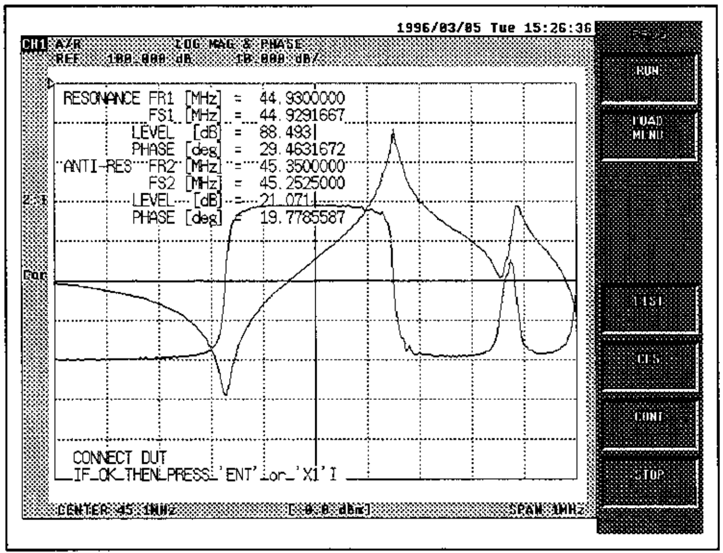
```
1000  !*******************************************
1010  !
1020  !            SEARCH  RESONANCE  POINT
1030  !            f=45.1MHz
1040  !
1050  ! FILE:RESONA.BAS
1060  !*******************************************
1070  *MAIN
1080      GOSUB *SETUP
1090      GOSUB *CAL
```

```
1100        CLS
1110        *MEAS_LOOP
1120            GOSUB *MEAS
1130            GOSUB *RESULTS
1140            GOTO  *MEAS_LOOP
1150    !
1160    *SETUP
1170        INTEGER EV
1180        NA=31 :EV=1
1190        OUTPUT NA;“OLDC OFF”
1200        OUTPUT NA;“SYST:PRES;:INIT:CONT OFF;:STAT:OPER:ENAB 8;*SRE
                                                              128;*OPC?”
1210        ENTER NA;A
1220        OUTPUT NA;“FREQ:SPAN 1MAHZ;CENT 45.1MAHZ”
1230        OUTPUT NA;“BAND 1KHZ”
1240        OUTPUT NA;“CALC:TRAN:IMP:CIMP 12.5;TYPE ZTR”
1250        RETURN
1260    !
1270    *CAL
1280        CURSOR 6,9  :PRINT “CONNECT [THROUGH]”
1290        CURSOR 6,10 :INPUT “IF OK THEN PRESS ‘ENT’ or ‘X1’”,D$
1300        OUTPUT NA;“CORR:COLL NORM;*OPC?” :ENTER NA;A
1310        RETURN
1320    !
1330    *MEAS
1340        CURSOR 6,25 :PRINT “CONNECT DUT”
1350        CURSOR 6,26 :INPUT “IF OK THEN PRESS ‘ENT’ or ‘X1’”,D$
1360        OUTPUT NA;“INIT” :WAIT EVENT EV
1370        FR1=FMAX(0,1200,0) :AP1=POINT1(FR1,0)
1380        FR2=FMIN(0,1200,0) :AP2=POINT1(FR2,0)
1390        FS1=ZEROPHS(AP1-60,AP1+60,8):LV1=VALUE(AP1,0):PH1=VALUE
                                                               (AP1,8)
1400        FS2=ZEROPHS(AP2-60,AP2+60,8):LV2=VALUE(AP2,0):PH2=VALUE
                                                               (AP2,8)
1410        RETURN
1420    !
1430    *RESULTS
```

```
1440      CURSOR 5,4 :PRINT "RESONANCE FR1 [MHz]="; :PRINT USING "3D.7D"
                                                           ;FR1/1.0E+6
1450      CURSOR 5,5 :PRINT "          FS1 [MHz]="; :PRINT USING "3D.7D"
                                                           ;FS1/1.0E+6
1460      CURSOR 5,6 :PRINT "         LEVEL [dB]="; :PRINT USING "3D.7D"
                                                                  ;LV1
1470      CURSOR 5,7 :PRINT "        PHASE [deg]="; :PRINT USING "3D.7D"
                                                                  ;PH1
1480      CURSOR 5,8 :PRINT "ANTI-RES   FR2[MHz]="; :PRINT USING "3D.7D"
                                                           ;FR2/1.0E+6
1490      CURSOR 5,9 :PRINT "           FS2[MHz]="; :PRINT USING "3D.7D"
                                                           ;FS2/1.0E+6
1500      CURSOR 5,10 :PRINT "        LEVEL [dB]="; :PRINT USING "3D.3D"
                                                                  ;LV2
1510      CURSOR 5,11 :PRINT "       PHASE [deg]="; :PRINT USING "3D.7D"
                                                                  ;PH2
1520      RETURN
```

When this program is executed, the screen display of this instrument is turned as shown in "figure 11-2 ". π Circuit jig (PIC-001) is used for setup here.

Execution result:



```
1996/03/05 Tue 15:26:36
CH1 A/R              LOG MAG & PHASE
    REF  100.000 dB    10.000 dB/

RESONANCE FR1 [MHz] =  44.9300000
          FS1 [MHz] =  44.9291667
         LEVEL  [dB] =  88.493
         PHASE [deg] =  29.4631672
ANTI-RES  FR2 [MHz] =  45.3500000
          FS2 [MHz] =  45.2525000
         LEVEL  [dB] =  21.071
         PHASE [deg] =  19.7785587

CONNECT DUT
IF_OK_THEN_PRESS_'ENT'_or_'X1'I

CENTER 45.1MHz          [ 0.0 dBm]          SPAN 1MHz
```

Figure 11-2 Screen Display of Execution Result (Measurement of crystal Resonant Point)

The resonant point and anti-resonant point. can be sought through the following two ways.

- Searching with maximum level value and minimum level value.
- Searching with zero phase.

Here, the measurement points of maximum level value and minimum level value are searched first with FMAX and FMIN functions.
Then , search the zero phase point being near to the measurement point is searched with the use of ZEROPHS function.

11.6 Using Built-in Functions

This instrument is provided with the built-in functions related to wave form analysis such as band width analysis and ripple analysis.
In R3753, R3764/66 and R3765/67, mark function is used to perform the wave form analysis. However, with the built-in functions, you can perform all the operations by calling up one function.

For the function explanation of built-in functions, refer to "4.4 Built-in Functions" of R3752/53, R3764/66, R3765/67 Program Manual.
In this section, the example using built-in functions is explained.

11.6.1 Basic Function

The built-in functions listed below are used to perform the basic conversions such as calculating necessary parameters etc. with the real analysis function.

POINT1	Gets the measurement point nearest the specified frequency.
POINT1L	Gets the maximum measurement point lower than the specified frequency.
POINT1H	Gets the minimum measurement point higher than the specified frequency.
POINT2	Gets the address point nearest the specified frequency.
POINT2L	Gets the address point lower than the specified frequency.
POINT2H	Gets the minimum address point higher than the specified frequency.
DPOINT	Gets the address point band width corresponding to the specified frequency band width.
SWPOINT	Gets the last measurement point.
FREQ	Gets the frequency corresponding to the specified address point.
DFREQ	Gets the frequency band width corresponding to the interval between the specified addresses.
SWFREQ	Gets the last scan frequency.
VALUE	Gets the response value of the specified address point.
DVALUE	Gets the difference of response values between the specified addresses.
CVALUE	Gets the response value of specified frequency.
DCVALUE	Gets the difference of response values between the specified frequencies.
SWVALUE	Gets the last measurement response value.

Almost all the built-in functions treat the address point as argument. To use other built-in functions, convert the frequency to the measurement point by using these functions. The absolute range of address point is 0 to 1200. The measurement point is the value actually existing in this range. The measurement point varies with the measurement point count set by the measurement condition.

The data of address point except for measurement point uses the value interpolated from the measurement point.

The following is a program example.

```
100 P  = POINT1(250.0E6,0)   ! Measurement point nearest to 250MHz.
110 V  = VALUE(P,0)          ! Gets measurement value.
120 P0 = POINT1L(100.0E6,0)  ! The maximum value by address point lower
                               than 100MHz.
130 P1 = POINT1H(200.0E6,0)  ! The minimum value of address point upper
                               than 200MHz.
140 Va = MAX(P0,P1,0)        ! Gets the maximum value.
```

11.6.2 Using Example of Maximum and Minimum Value Analysis Functions

The built-in functions shown as follows are used to analyze the maximum and minimum values in the specified range.

MAX	Gets the maximum response value.
MIN	Gets the minimum response value.
FMAX	Gets the maximum response frequency.
FMIN	Gets the minimum response frequency.
PMAX	Gets the maximum response measurement point.
PMIN	Gets the minimum response measurement point.

These functions are used to search the measurement point of maximum response or minimum response between the address points of specified channel. Then , the analysis value of the measurement point is transferred as a function value.

The function MAX and MIN return the response values, FMAX and FMIN functions return the stimulus values (frequency value) and PMAX and PMIN return the measurement point values.

When these functions are used in combination, the resonant point and anti-resonant point can be analyzed.

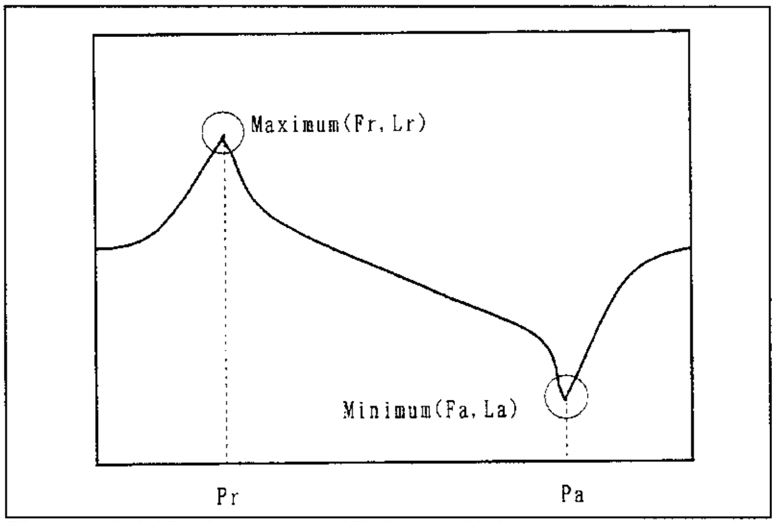


Figure 11-3 Maximum and Minimum Value Analysis Function

The following shows an example of program for analyzing the maximum and minimum values.

```
100 Vr = MAX(0,1200,0)   ! Gets the maximum response value.
110 Fr = FMAX(0,1200,0)  ! Gets the stimulus value of maximum response.
120 Pr = PMAX(0,1200)    ! Gets the measurement point of maximum
                           response.
130 Va = MIN(0,1200,0)   ! Gets the minimum response value.
140 Fa = FMIN(0,1200,0)  ! Gets the stimulus value of minimum response.
150 Pa = PMIN(0,1200,0)  ! Gets the measurement point of minimum
                           response.
```

To get all analysis values of maximum or minimum response, it is not necessary to call out all these functions.

First, the measurement point is got with PMAX or PMIN, then it is taken as a parameter to call out FREQ and VALUE functions.

In this way, the analysis value can be got at a high speed higher than that when MAX and FMAX or MIN and FMIN are used.

```
100 Pr = PMAX(0,1200,0)    ! Gets the measurement point of maximum
                             response.
110 Vr = VALUE(Pr,0)       ! Gets the maximum response value.
120 Fr = FREQ(Pr,0)        ! Gets the stimulus value of maximum
                             response.
140 Pa = PMIN(0,1200,0)    ! Gets the measurement point of minimum
                             response.
150 Va = VALUE(Pa,0)       ! Gets the minimum response value.
160 Fa = FREQ(Pa,0)        ! Gets the stimulus value of minimum
                             response.
```

11.6.3 Using Example of Attenuation Level Analysis Functions

The following built-in functions are used to analyze the typical parameters in filter etc.

BND Gets the band width from the specified address point.

BNDL Gets the low frequency of band width from the specified address point.

BNDH Gets the high frequency of band width from the specified address point.

CBND Gets the band width from the specified frequency.

CBNDL Gets the low frequency of band width from the specified frequency.

CBNDH Gets the high frequency of band width from the specified frequency.

MBNDI Performs the multiple band width analysis outwards.

MBNDO Performs the multiple band width analysis inwards.

(1) BND, BNDL, BNDH, CBND, CBNDL, CBNDH

These functions are used to analyze the attenuation point and band width from the specified attenuation level. For those functions whose name are leaded by C , the standard point of the function is specified with address pointer, and for those whose name are not leaded by C , specified with frequency.
When the special parameter of filter is to be calculated, it can be achieved by combination of these functions.

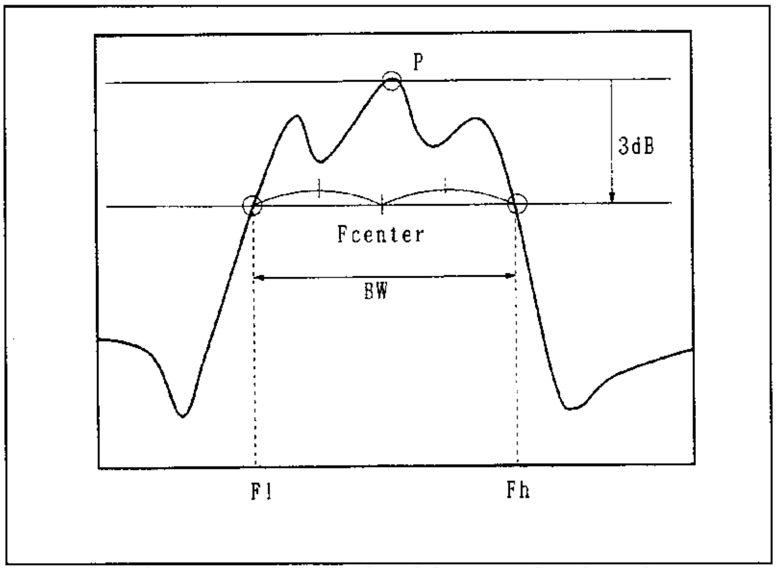


Figure 11-4 Analysis of Attenuation Level

The following is an example of attenuation level analysis program.

```
100 P  = PMAX(0,1200,0)    ! Gets the measurement point of maximum
                             response.
```

```
110 BW = BND(P,3,0)        ! Gets the band width of attenuation
                             level 3dB.
120 F1 = BNDL(P,3,0)       ! Gets the low frequency of attenuation
                             level 3dB of band width.
130 Fh = BNDH(P,3,0)       ! Gets the high frequency of attenuation
                             level 3dB of band width.
140 FC = (F1+Fh)*0.5       ! Calculates the center frequency.
150 Q  = SQR(F1*Fh)/ BW    ! Calculates Q.
```

(2) MBNDI, MBNDO

MBNDI or MBNDO function is used when the analysis of multiple attenuation level is performed.
These functions enable the multiple attenuation points to be analyzed at a time and the low frequency, high frequency, center frequency and band width can be obtained for one attenuation level.
When the analysis of attenuation level is performed outward from the standard pointer, the MBNDI function is used.

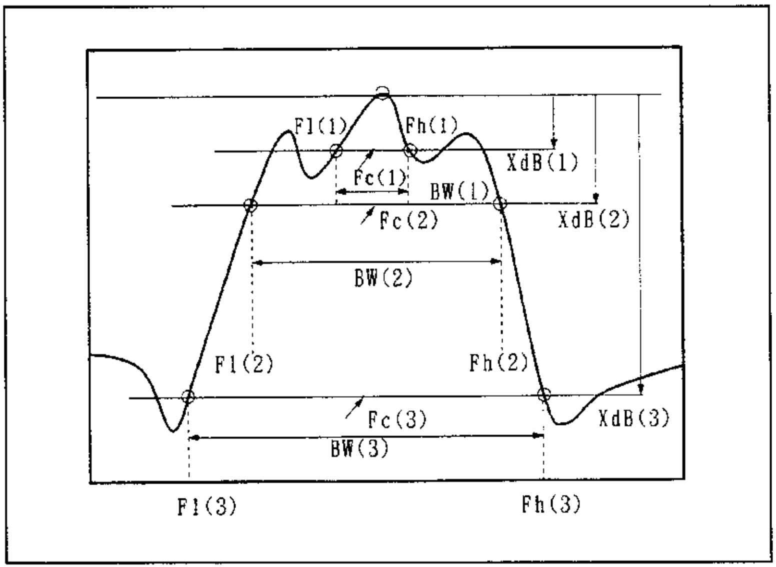


Figure 11-5 MBNDI

The example of program using MBNDI function is shown as follows.

```
100 DIM Levels(3)          ! Defines an array used for
                             attenuation level specification.
110 DIM DataBuffer(3,4)    ! Defines an array used for getting
                             analysis value.
120 Levels(1) = 1.0        ! Specifies the first attenuation
                             level to be analyzed to 1.0dB.
130 Levels(2) = 3.0        ! Specifies the second attenuation
                             level to be analyzed to 3.0dB.
```

```
140 Levels(3) = 10.0          ! Specifies the third attenuation
                                level to be analyzed to 10.0dB.
150 P=PMAX(0,1200,0)          ! Gets the measurement point of
                                maximum response.
160 N=MBNDI(0,1200,p,3,Levels(1),DataBuffer(1,1),0)
                              ! Analyzes multiple levels.
170 PRINT DataBuffer(1,1)     ! Low frequency of band width of Fl(1)
                                -attenuation level 1.0dB.
180 PRINT DataBuffer(1,2)     ! High frequency of band width of Fh(1)
                                -attenuation level 1.0dB.
190 PRINT DataBuffer(1,3)     ! Center frequency of band width of
                                Fc(1)-attenuation level 1.0dB.
200 PRINT DataBuffer(1,4)     ! Band width of BW(1)- attenuation
                                level 1.0dB.
210 PRINT DataBuffer(2,1)     ! Low frequency of band width of Fl(2)
                                -attenuation level 1.0dB.
220 PRINT DataBuffer(2,2)     ! High frequency of band width of Fh(2)
                                -attenuation level 1.0dB.
230 PRINT DataBuffer(2,3)     ! Center frequency of band width of
                                Fc(2)-attenuation level 1.0dB.
240 PRINT DataBurrer(2,4)     ! Band width of BW(2)-attenuation
                                level 1.0dB.
250 PRINT DataBuffer(3,1)     ! Low frequency of band width of Fl(3)
                                -attenuation level 1.0dB.
260 PRINT DataBuffer(3,2)     ! High frequency of band width of Fh(3)
                                -attenuation level 1.0dB.
270 PRINT DataBuffer(3,3)     ! Center frequency of band width of
                                Fc(3)-attenuation level 1.0dB.
280 PRINT DataBurrer(3,4)     ! Band width of BW(3)- attenuation
                                level 1.0dB.
```

When the analysis of attenuation level is performed inward from outside to the standard pointer, the MBNDO function is used.

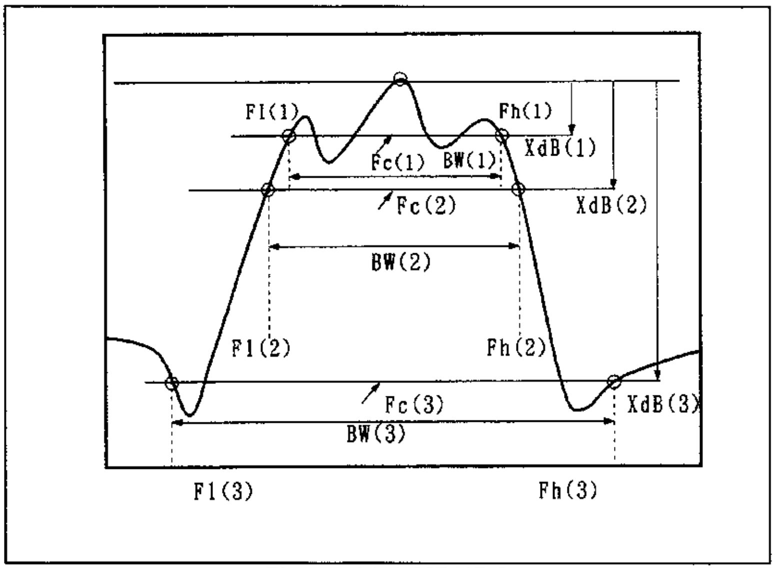


Figure 11-6 MBNDO

The example of program using MBNDO function is shown as follows.

```
100 DIM Levels(3)            ! Defines an array used for
                               attenuation level specification.

110 DIM DataBuffer(3,4)      ! Defines an array used for getting
                               analysis value.

120 Levels(1) = 1.0          ! Specifies the first attenuation
                               level to be analyzed to 1.0dB.

130 Levels(2) = 3.0          ! Specifies the second attenuation
                               level to be analyzed to 3.0dB.

140 Levels(3) = 10.0         ! Specifies the third attenuation
                               level to be analyzed to 10.0dB.

150 P = PMAX(0,1200,0)       ! Gets the measurement point of
                               maximum response.

160 N = MBNDO(0,1200,p,3,Levels(1),DataBuffer(1,1),0)
                             ! Analyzes multiple levels.

170 PRINT DataBuffer(1,1)    ! Low frequency of band width of Fl(1)
                               -attenuation level 1.0dB.

180 PRINT DataBuffer(1,2)    ! High frequency of band width of Fh(1)
                               -attenuation level 1.0dB.

190 PRINT DataBuffer(1,3)    ! Center frequency of band width of
                               Fc(1)-attenuation level 1.0dB.

200 PRINT DataBuffer(1,4)    ! Band width of BW(1)- attenuation
                               level 1.0dB.

210 PRINT DataBuffer(2,1)    ! Low frequency of band width of Fl(2)
                               -attenuation level 1.0dB.
```

```
220 PRINT DataBuffer(2,2)    ! High frequency of band width of Fh(2)
                               -attenuation level 1.0dB.
230 PRINT DataBuffer(2,3)    ! Center frequency of band width of
                               Fc(2)-attenuation level 1.0dB.
240 PRINT DataBurrer(2,4)    ! Band width of BW(2)-attenuation
                               level 1.0dB.
250 PRINT DataBuffer(3,1)    ! Low frequency of band width of Fl(3)
                               -attenuation level 1.0dB.
260 PRINT DataBuffer(3,2)    ! High frequency of band width of Fh(3)
                               -attenuation level 1.0dB.
270 PRINT DataBuffer(3,3)    ! Center frequency of band width of
                               Fc(3)-attenuation level 1.0dB.
280 PRINT DataBurrer(3,4)    ! Band width of BW(3)- attenuation
                               level 1.0dB.
```

11.6.4 Ripple Analysis Functions Using Example (1)

The following built-in functions are used to analyze the ripples and get the result.

RPL1	Gets the maximum value of difference between the highest value and lowest value.
RPL2	Gets the maximum value of difference between the neighboring highest value and lowest value.
RPL3	The total maximum value of the difference between the neighboring highest value and lowest value and the difference between the neighboring lowest value and highest value.
RPL4	Gets the maximum value of difference between the neighboring lowest value and highest value.
RPL5	Gets the maximum value of the highest value.
RPL6	Gets the minimum value of the highest value.
RPLF	Gets the frequency difference between the first highest point and lowest point.
RPLR	Gets the response difference between the first highest point and lowest point.
RPLH	Gets the response value of the first highest point.
FRPLH	Gets the frequency value of the first highest point.
PRPLH	Gets the measurement point of the first highest point.
RPLL	Gets the measurement point of the first lowest point.
FRPLL	Gets the frequency value of the first lowest point.
PRPLL	Gets the measurement point of the first lowest point.

As the searching target , ripple is specified with the coefficients of abscissa axis cant rate and ordinate axis cant rate. The cant rate coefficient of abscissa axis is specified with address point,while the cant rate coefficient of ordinate axis is specified with response value.
For instance , when the ripple occurs 0.5dB up and down per one point in RPL1 function, it is shown as follows.

```
100  MaxDiff = RPL1(0,1200,1,0.5.0)
```

(1) RPL1

RPL1 function is used to get the maximum value of difference between the highest value and the lowest value in the specified range.

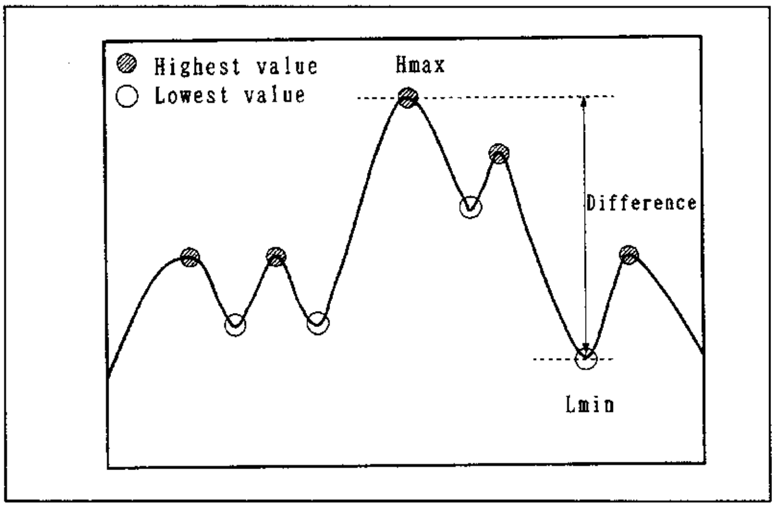


Figure 11-7 RPL1

The example of program showing the maximum value of difference between the highest and the lowest is shown as follows.

```
100 MaxDiff = MAX(0,1200,0) ! Gets the maximum value of differ-
110 PRINT MaxDiff             ence between the highest value and
                              the lowest value.
```

(2) RPL2 , RPL4

These functions are used to get the maximum value of difference between the neighboring highest value and lowest value.
However, RPL2 is used to detect the difference between the highest value and the lowest value to the right of the highest value , while RPL4 is used to detect the difference between the highest value and the lowest value to the left of the highest value.

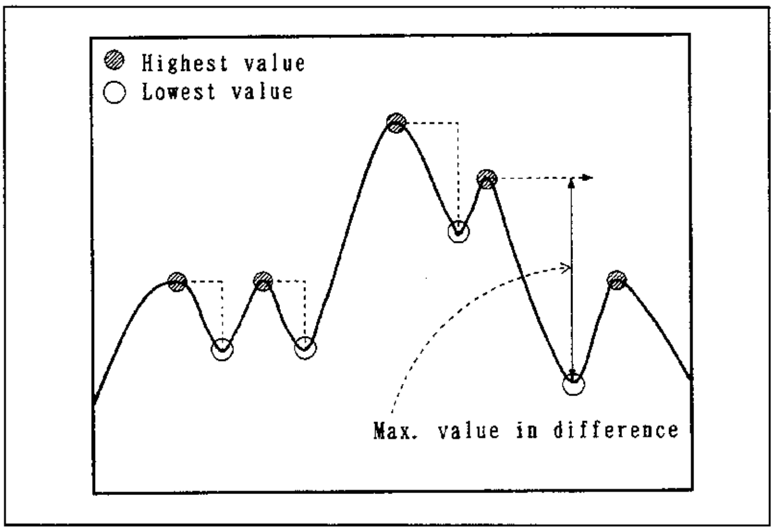


Figure 11-8 RPL2

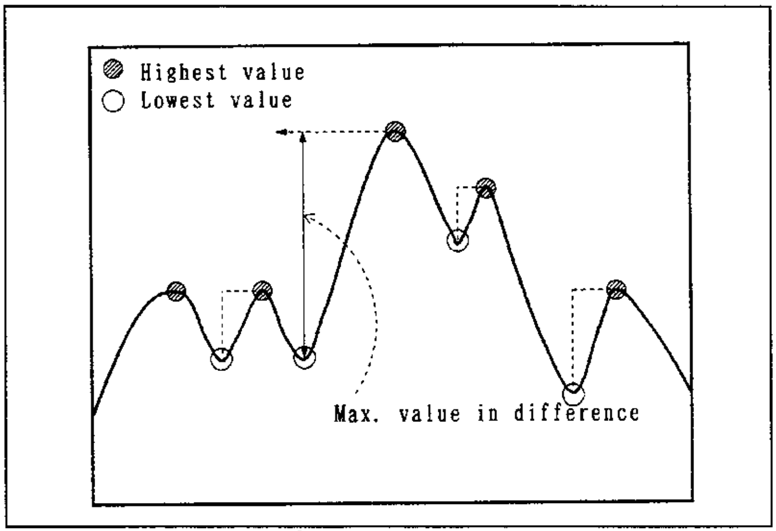


Figure 11-9 RPL4

The example of program is shown as follows.

```
100 P    = PMAX(0,1200,0)          ! Gets the measurement point.
110 RMax = RPL2(0,P,1,0.5,0)       ! Searches the right side.
120 LMax = RPL4(0,P,1,0.5,0)       ! Searches the left side.
```

(3) RPL3

RPL3 function is used to get the total maximum value that is obtained by adding the difference between the neighboring highest value and lowest and the difference between the lowest value and the highest value.

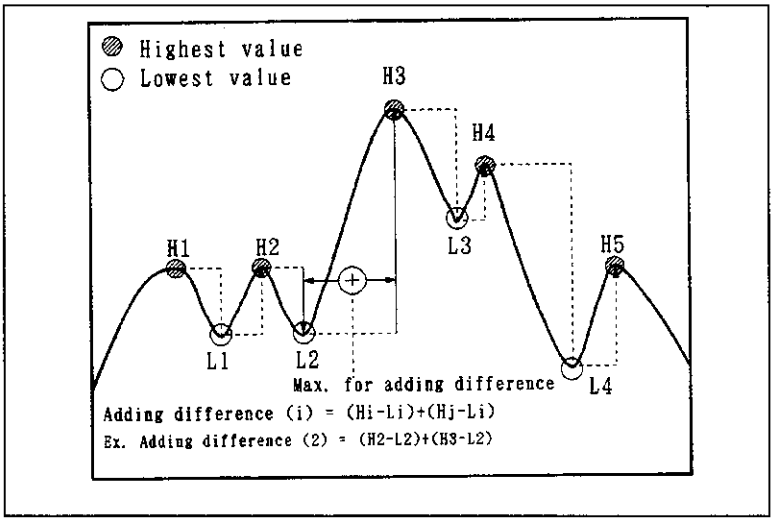

▨ Highest value
○ Lowest value
H3
H4
H1
H2
L3
H5
L1
L2
Max. for adding difference
L4
Adding difference (i) = (Hi-Li)+(Hj-Li)
Ex. Adding difference (2) = (H2-L2)+(H3-L2)

Figure 11-10 RPL3

The example of program used to get the maximum value of added differences is shown as follows.

```
100 MaxAdding = RPL3(0,1200,1,0.5,0)
```

(4) RPL5 , RPL6

These functions are used to get the maximum value and minimum value of the highest value. It is used when ripple spurious is analyzed.

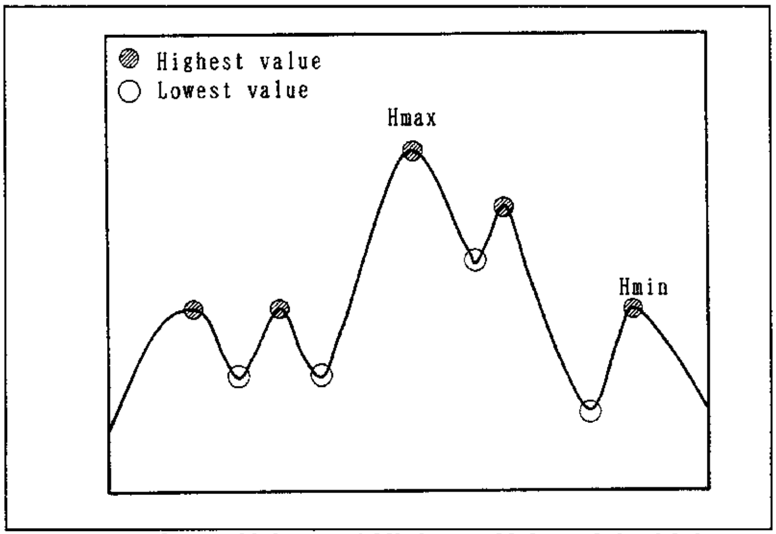


Figure 11-11 Maximum Value and Minimum Value of the highest value.

The example of program is shown as follows.

```
100 P0 = POINT1(10.0E6,0)        ! Start range of analysis is 10MHz.
110 P1 = POINT1(20,0E6,0)        ! End range of analysis is 20MHz.
120 Hmax = RPL5(P0,P1,1,0.5,0)   ! Gets the maximum value of the
                                   highest value.
130 Hmin = RPL6(P0,P1,1,0.5,0)   ! Gets the minimum value of the
                                   highest value.
```

(5) RPLF, RPLR, RPLH, RPLL, FRPLH, FRPLL, PRPLH, PRPLL

These functions are used to analyze the ripples of the highest point and the lowest point which is detected first. RPLF and RPLR are used to calculate the response difference (or frequency difference) between the highest point and the lowest point; RPLH and RPLL, get the response value of the highest point or the lowest point; FRPLH and FRPLL, get the frequency value of the highest point or the lowest point and PRPLH and PRPLL, get the measurement point of the highest point or the lowest point.

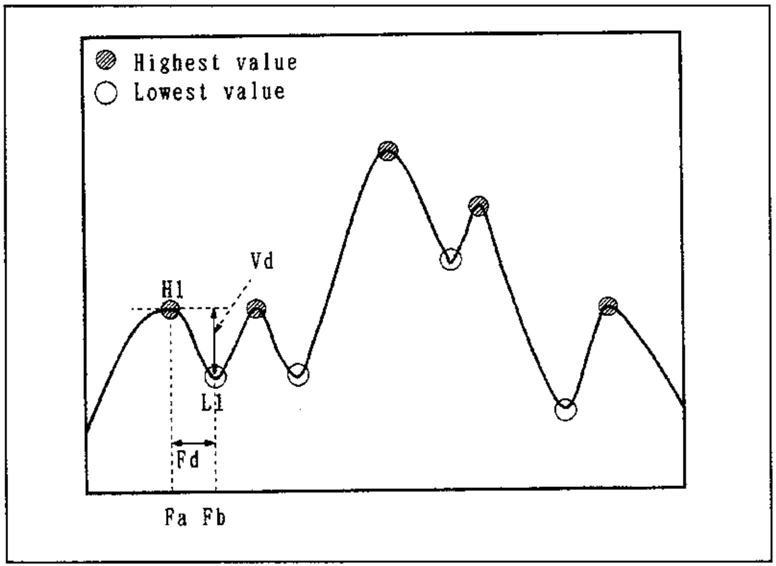


Figure 11-12 Response and Frequency of Ripple

The example of program is shown as follows.

```
100 Fd = RPLF(0,1200,1,0.5,0)   ! Gets the frequency difference
                                  between highest point and lowest
                                  point.
110 Vd = RPLR(0,1200,1,0.5,0)   ! Gets the response difference
                                  between highest point and lowest
                                  point.
120 H1 = RPLH(0,1200,1,0.5,0)   ! Gets the response of highest
                                  point.
130 L1 = RPLL(0,1200,1,0.5,0)   ! Gets the response of lowest
                                  point.
```

```
140 Fa = FRPLH(0,1200,1,0.5,0)  ! Gets the frequency of highest
                                  point.
150 Fb = FRPLL(0,1200,1.0.5,0)  ! Gets the frequency of lowest
                                  point.
160 Pa = PRPLH(0,1200,1,0.5,0)  ! Gets the measurement point of
                                  highest point.
170 Pb = PRPLL(0,1200,1,0.5,0)  ! Gets the measurement point of
                                  lowest point.
```

However, this program is not in practice, because every time you call up the built-in functions, the overwrite for the searching should be considered.
If you know the measurement points of the highest point and the lowest point, the frequency and response value can be calculated with FREQ and VALUE functions. The program is transferred in practice after it is changed to as follows.

```
100 Pa = RPLH(0,1200,1,0.5,0)   ! Gets the measurement point of
                                  highest point.
110 Pb = RPLL(0,1200,1,0.5,0)   ! Gets the measurement point of
                                  lowest point.
120 Fa = FREQ(Pa,0)             ! Calculates the frequency of
                                  highest point.
130 Fb = FREQ(Pb,0)             ! Calculates the frequency of
                                  lowest point.
140 H1 = VALUE(Pa,0)            ! Calculates the response of
                                  highest point.
150 L1 = VALUE(Pb,0)            ! Calculates the response of
                                  lowest point.
160 Fd = Fb - Fa                ! Calculates the frequency differ-
                                  ence between highest point and
                                  lowest point.
170 Vd = H1 - L1                ! Calculates the response differ-
                                  ence between highest point and
                                  lowest point.
```

11.6.5 Ripple Analysis Function Using Example (2)

The following built-in functions are used first to get all ripples to be analyzed , then analyze them after the ripple number is specified. When multiple ripples are analyzed, the analysis functions described below are used.

NRPLH Gets the count of highest points.

NRPLL Gets the count o lowest points.

PRPLHN Gets the measurement point of the n-th highest point.

PRPLLN Gets the measurement point of the n-th lowest point.

FRPLHN Gets the frequency value of the n-th highest point.

FRPLLN Gets the frequency value of the n-th lowest point.

VRPLHN	Gets the response value of the n-th highest point.
VRPLLN	Gets the response value of the n-th lowest point.
PRPLHM	Gets the measurement points of multiple highest point.
PRPLLM	Gets the measurement points of multiple lowest point.
FRPLHM	Gets the frequency value of multiple highest point.
FRPLLM	Gets the frequency value of multiple lowest point.
VRPLHM	Gets the response value of multiple highest point.
VRPLLM	Gets the response value of multiple lowest point.

(1) NRPLH , NRPLL

These two functions are used to analyze the count of highest point or lowest point. When the built-in functions are used to analyze the ripples after the ripple number is specified,the ripple count is obtained in advance according to NRPLH and NRPLL.

(2) PRPLHN, PRPLLN, FRPLHN, FRPLLN,VRPLHN, VRPLLN

These functions are used to perform the analysis for those ripples obtained with NRPLH and NRPLL,then specified by numbers.

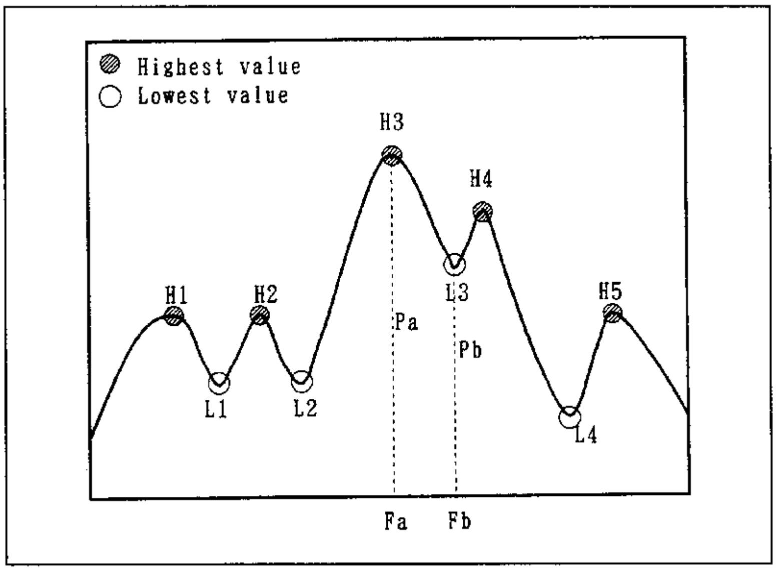


Figure 11-13 Analysis of Specified Ripple

The example of program is shown as follows.

```
100 Nh = NRPLH(0,1200,1,0,5,0)    ! Searches the highest point and
                                    enables to perform the number-
                                    specified analysis.

110 N1 = NRPLL(0,1200,1,0,5,0)    ! Searches the lowest point and
                                    enables to perform the number-
                                    specified analysis.

120 Pa = PRPLHN(3,0)              ! Gets the measurement point of
                                    the third highest point.
```

```
130 Fa = PRPLHN(3,0)            ! Gets the frequency point of
                                  the third highest point.
140 H3 = VRPLHN(3,0)            ! Gets the response value of
                                  the third highest point.
150 Pb = PRPLLN(3,0)            ! Gets the measurement point of
                                  the third lowest point.
160 Fb = FRPLLN(3,0)            ! Gets the frequency of the third
                                  lowest point.
170 L3 = VRPLLN(3.0)            ! Gets the response value of the
                                  third lowest point.
```

(3) PRPLHM, PRPLLM, FRPLHM, FRPLLM, VRPLHM, VRPLLM

These functions are used to get the analysis values of all ripples with NRPLH and NRPLL.

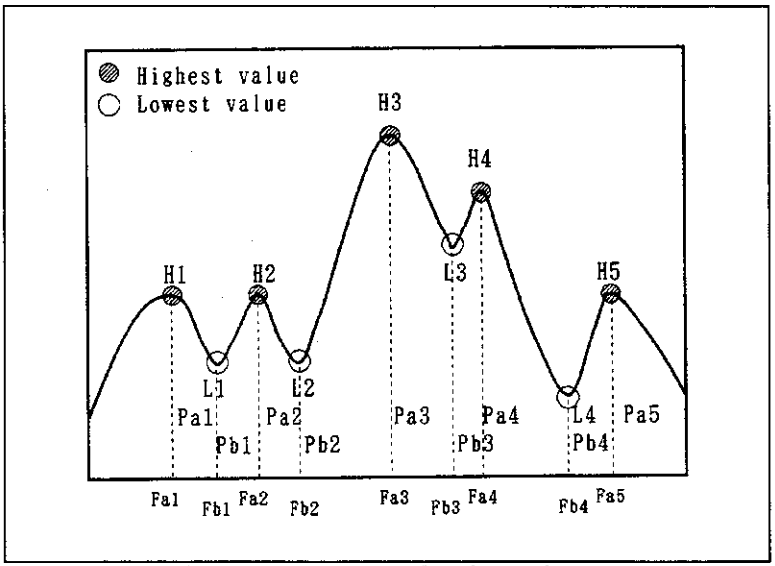


Figure 11-14 Analysis of All Ripple

The example of program is shown as follows.

```
100 INTEGER Pa(300),Pb(300)
110 DIM Fa(300),Fb(300)
120 DIM Va(300),Vb(300)
130 Nh = NRPLH(0,1200,1,0.5,0)   ! Searches highest point and ena-
                                   bles number specified analysis.
140 N1 = NRPLL(0,1200,1,0.5,0)   ! Searches lowest point and ena-
                                   bles number specified analysis.
150 Na = PRPLHM(Pa(1),0)         ! Gets measurement points of all
                                   the highest point.
160 Nb = PRPLLM(Pb(1),0)         ! Gets measurement points of all
                                   the lowest point.
```

```
170 Na = FRPLHM(Fa(1),0)          ! Gets frequencies of all the
                                    highest point.
180 Nb = FRPLLM(Fb(1),0)          ! Gets frequencies of all the
                                    lowest point.
190 Na = VRPLHM(Va(1),0)          ! Gets response values of all the
                                    highest point.
200 Nb = VRPLLM(Vb(1),0)          ! Gets response values of all the
                                    lowest point.
```

11.6.6 Using Example of Direct Search Functions

The following built-in functions are used to search the response value given in the specified range.

DIRECT	Gets the address point of specified response.
DIRECTL	Gets the left measurement point corresponding to the specified response.
DIRECTH	Gets the right measurement point corresponding to the specified response.
CDIRECT	Gets the frequency of specified response.
CDIRECTL	Gets the left real frequency corresponding to specified response.
CDIRECTH	Gets the right real frequency corresponding to specified response.
DDIRECT	Gets the address point width of specified response.
CDDIRECT	Gets the band width of specified response.
ZEROPHS	Searches the frequency of the first zero phase.

(1) DIRECT, DIRECTL, DIRECTH, CDIRECT, CDIRECTL, CDIRECTH

These functions are used to specifies response value and then search the place that is coincident with the response value.
For the DIRECT function whose name is leaded by C , the search range is specified with frequency , while for others, the search range is specified with address point .
The function whose name is ended by L searches from low frequency to high frequency, and that whose name is ended by H searches from high frequency to low frequency, so as to find the analysis value corresponding to the real measurement value. However, when no measurement point is coincident, then take the one that comes immediately after the specified response value.

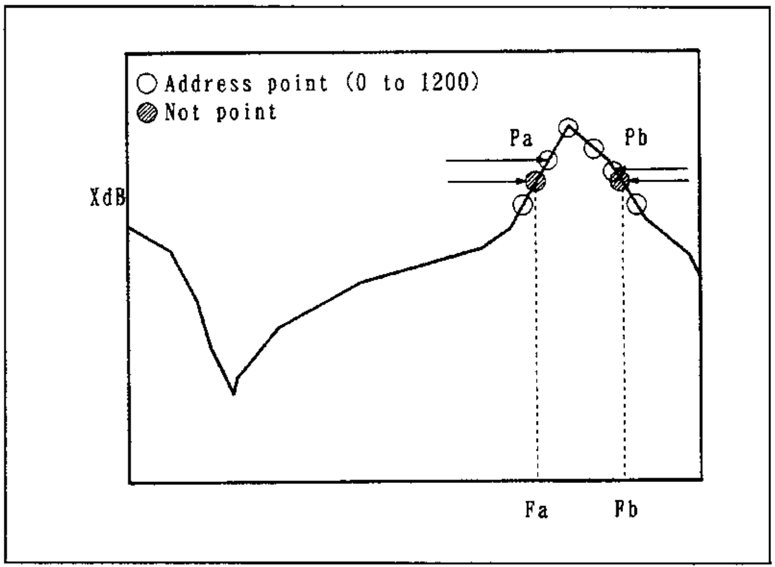


Figure 11-15 Direct Search

The example of program is shown as follows.

```
100 P = DIRECT(0,1200,-10,0)        ! Address point of response
                                      value -10dB.
110 Pa = DIRECTL(0,1200,-10,0)      !
120 Pb = DIRECTH(0,1200,-10,0)      !
130 F  = CDIRECT(5,500.0E6,-10,0)   ! Frequency of response value
                                      -10dB
140 Fa = CDIRECTL(5,500.0E6,-10,0)  !
150 Fb = CDIRECTH(5,500.0E6,-10,0)  !
```

(2) DDIRECT, CDDIRECT

These functions are used to search two measurement points corresponding to the specified response value and get the point width.

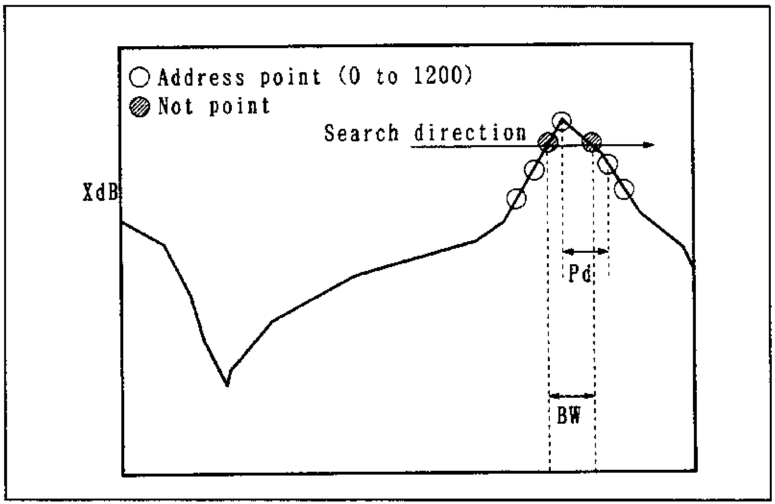


Figure 11-16 Band Width Corresponding to Response

The example of program is shown as follows.

```
100 Pd = DDIRECT(0,1200,-10,0)      ! Address point width.
110 BW = CDDIRECT(5,500,0E6,-10,0)! Band width(interpolating with
                                     frequency.)
```

(3) ZEROPHS

ZEROPHS function is used to search the frequency at which the phase value first becomes zero degree between the specified address points.

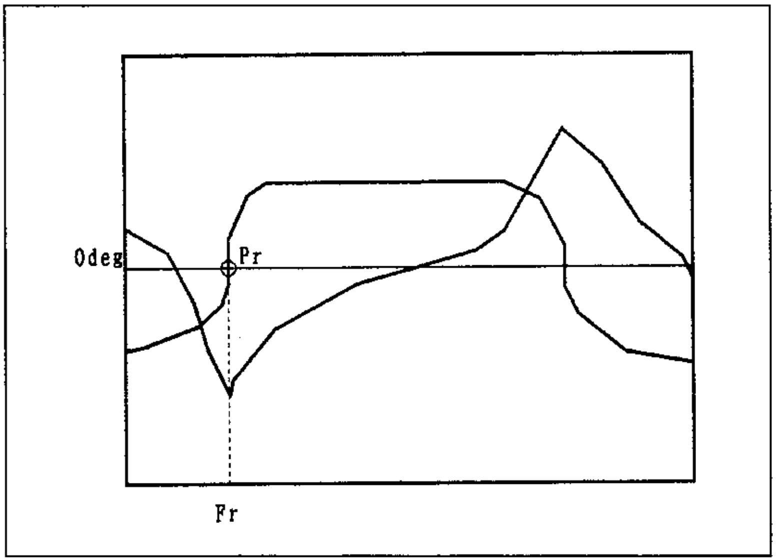


Figure 11-17 Search of Zero Phase

The example of program is shown as follows.

```
100 Pr = PMAX(0,1200,0)     ! Gets the measurement point of maximum
                              response.
110 Pa = PMIN(0,1200,0)     ! Gets the measurement point of minimum
                              response.
120 Fr = ZEROPHS(Pr,Pa,0) ! Gets the frequency of zero phase.
```

11.6.7 Data Transferring

The following built-in functions are used to transfer data between built-in and incorporated BASIC.

TRANSR	Loads data from the memory of analysis channel.
TRANSW	Writes data to the memory of analysis channel.

The example of program is shown as follows.

```
100 DIM buf(2,1200)                  ! Defines data array.
110 N = TRANSR(0,1200,Buf(1,1),0)    ! Reads in the first wave form
                                       data of CH1.
120 N = TRANSR(0,1200,Buf(2,1),8)    ! Reads in the second wave form
                                       data of CH2.
```

11.7 Setting Limit Line

In this section , a program example of setting the limit line is explained.

A band-pass filter of 880MHz is used for the test specimen (DUT). After setup, normalized, then the limit line is set as shown below.

Segment	0	1	2	3	4
frequency	780MHz	820MHz	866MHz	898MHz	960MHz
upper value	-40dB	-40dB	-10dB	-10dB	-40dB
lower value	-65dB	-65dB	-30dB	-30dB	-65dB

(Note) Can not be used by R3752/53.

The program example is shown in Example 11-6.

Example 11-6 Setting Limit Line

```
100  ! *************************************
110  !
120  !          SET  LIMIT  LINE  TABLE
130  !
140  !  *************************************
150  !
160  INTEGER  I
170  OUTPUT 31;"OLDC OFF"
180  OUTPUT 31;"SYST:PRES"
190  OUTPUT 31;"DISP:ACT 2"
200  OUTPUT 31;"FREQ:CENT 880MHZ;SPAN 200MHZ"
210  !
220  CLS:CURSOR 0,16
230  INPUT "Connect THRU, then press [X1].",D$
240  OUTPUT 31;"CORR:COLL NORM;*OPC?"
250  ENTER 31;DUMMY$
260  !
270  FOR I=0 TO 4
280  READ ST,UP,LO
290  OUTPUT 31;"DISP:LIM:SEGM",I,":STIM",ST,"MHZ;UPP",UP,";LOW",LO
300  OUTPUT 31;"DISP:LIM:SEGM",I,":COL 3;WCOL 6"
310  NEXT
320  OUTPUT 31;"DISP:LIM:STAT ON;LINE ON"
330  CLS
340  STOP
350  !
360  DATA 780,-40, -65
370  DATA 820,-40,-65
380  DATA 866,-10,-30
390  DATA 898,-10,-30
400  DATA 960,-40,-65
```

There are two ways to set the limit line. one is setting all the segments at a time by using DISPlay [: WINDow [<chno>]] : LIMit [<parano>] : DATA <block>. Another is setting each segment in integrated way by using DISPLay [: WINDow [<chno>]]:LIMit [<parano>] : SEGMent <n> <Lock>. The parameter of each cement is set here, respectively

The program of Example 11-6 is explained below.

Program Explanation of Example 11-6	
100 to 150	Comment lines.
160	Turns variable I to integer type. (Because the segment is specified with interger.)
170	Releases the R3762/63 conversion command mode.
180	Initializes the setting of network analyzer.
190	Activates channel 2.
200	Turns scan frequency to center 880MHz, and span to 100MHz.
210	Comment line.
220	Clears the characters on the screen and moves the cursor.
230	Displays the message and waits for input.
240	Gets normalize-data and requests stop notification.
250	Waiting till the getting is ended.
260	Comment line.
270	Changes the segment number I from 0 to 4 in sequence.
280	Reads in the data of frequency, upper limit value and lower limit value.
290	Sets frequency , upper limit value and lower limit value to segment I.
300	Sets the color of limit line and wave form.
310	Moves to the next segment.
320	Turns the limit test decision and limit line display ON.
330	Clears the screen.
340	Ends.
350	Comment line.
360 to 400	Frequency , upper limit value and lower limit value of each segment.

11.8 Four Screens Display of All S Parameters

This section describes an example of program that is used to perform four-screen display of all S parameters.
A band-pass filter of 880MHz is used as the device under test (DUT).
After setup, two-port full-calibration is performed and the following four screens are displayed.

[CH1] S11 Smith chart SMITH (R+jx)	[CH2] S12 Amplitude/phase LOG MAG & PHASE
[CH3] S22 Smith chart SMITH (R+jx)	[CH4] S21 Amplitude/Group delay time LOG MAG & DELAY

(Note) ***Can not be used by R3752/53.***
S parameter measurement in R3764/65/66/67 is enabled only when C type or A type +S parameter test set is used.

The program example is shown in Example 11-7.

Example 11-7 Four-Screen Display of all S Parameter

```
100  ! **************************************
110  !
120  !       2-PORT FULL CALIBRATION
130  !       AND 4 CHANNELS DISPLAY
140  !
150  !  **************************************
160  !
170  *MAIN
180    GOSUB *SETUP
190    GOSUB *CAL
200    GOSUB *DISP4CH
210    STOP
220    !
230   *SETUP
240     OUTPUT 31;“OLDC OFF”
250     OUTPUT 31;“SYST:PRES”
260     OUTPUT 31;“FREQ:CENT  880MHZ;SPAN  100MHZ”
270     OUTPUT 31;“BAND 100HZ”
280     OUTPUT 31;“DISP:FORM ULOW”
290     CLS:CURSOR 0,16
```

```
300     RETURN
310   !
320   *CAL
330     INPUT "Connect OPEN to port 1, then press [X1].",D$
340     OUTPUT 31;"CORR:COLL S11O"
350     GOSUB *SWPEND
360     INPUT "Connect SHORT to port 1, then press [X1].",D$
370     OUTPUT 31;"CORR:COLL S11S"
380     GOSUB *SWPEND
390     INPUT "Connect LOAD to port 1, then press [X1].",D$
400     OUTPUT 31; "CORR:COLL S11L"
410     GOSUB *SWPEND
420     INPUT "Connect OPEN to port 2, then press [X1].",D$
430     OUTPUT 31;"CORR:COLL S22O"
440     GOSUB *SWPEND
450     INPUT "Connect SHORT to port 2, then press [X1].",D$
460     OUTPUT 31;"CORR:COLL S22S"
470     GOSUB *SWPEND
480     INPUT "Connect LOAD to port 2, then press [X1].",D$
490     OUTPUT 31;"CORR:COLL S22L"
500     GOSUB *SWPEND
510     !
520      INPUT "Connect THRU between port 1 and 2, then press [X1].",D$
530     OUTPUT 31;"CORR:COLL GTHRU"
540     GOSU *SWPEND
550     !
560     OUTPUT 31;"CORR:COLL OIS"
570     GOSUB *SWPEND
580     !
590     OUTPUT 31;"CORR:COLL:SAVE"
600     OUTPUT 31;"BAND:AUTO ON"
610     CLS
620     RETURN
630     !
640   *DISP4CH
650     OUTPUT 31;"DISP:DUAL ON;FORM ULOW"
```

```
660     OUTPUT 31;"FUNC1:POW S11"
670     OUTPUT 31;"FUNC2:POW S12"
680     OUTPUT 31;"FUNC3:POW S22"
690     OUTPUT 31;"FUNC4:POW S21"
700     OUTPUT 31;"CALC1:FORM SCH"
710     OUTPUT 31;"CALC2:FORM MLOP"
720     OUTPUT 31;"CALC3:FORM SCH"
730     OUTPUT 31;"CALC4:FORM MLOD"
740     RETURN
750     !
760   *SWPEND
770     OUTPUT 31;"*OPC?"
780     ENTER 31;D$
790     RETURN
```

In order to display the submajor (channel 3,4), 3 or 4 must be specified in the channel specification <chno> of the measurement mode specification command [SENSe:] FUNCtion <chno> [: ON] " <input>" or [SENSe:] FUNCtion <chno>: POWER <input>. Specifying measurement format of channel 3 and 4 etc. is performed in advance with the measurement mode specification after the channel has been displayed.
The example of program is shown as follows.

Program Explanation of Example 11-7	
100 to 160	Comment lines.
170	Level MAIN of main routine.
180	Calls out initial setting routine SETUP.
190	Calls out correction routine CAL.
200	Calls out four screen display routine DISP4 CH.
210	Ends.
220	Comment line.
230	Level SETUP of initial setting routine.
240	Releases R3762/63 convention command mode.
250	Initializes the setting of network analyzer.
260	Turns scan frequency to center 880MHz and span to 100MHz.
270	Turns resolution band width to 100Hz.
280	Performs screen split display of upper and lower two parts.

290	Clears the characters on the screen and moves the cursor.
300	Gets rid of initial setting routine.
310	Comment line.
320	Level CAL of correction routine.
330	Displays message and waits for input. (be the same below.)
340	Gets correction data (S11 OPEN)
350	Waits for the end of getting (be the same below.)
360 to 380	Gets correction data (S11 SHORT).
390 to 410	Gets correction data (S11 LOAD) .
420 to 440	Gets correction data (S22 OPEN) .
450 to 470	Gets correction data (S22 SHORT) .
480 to 500	Gets correction data (S22 LOAD) .
510	Comment line.
520 to 540	Gets correction data (GROUP THRU).
550 to 560	Gets correction data (Omits ISOLATION correction).
580	Comment line.
590	Calculates error coefficient from correction data.
600	Enables the resolution band width to be set automatically.
610	Clears the characters on the screen.
620	Gets rid of correction data.
630	Comment line.
640	Level DISP4CH of four screen display routine.
650	Enables two channel display, upper and lower two parts split display.
660	Turns the measurement mode of channel 1 to S11.
670	Turns the measurement mode of channel 2 to S12.
680	Turns the measurement mode of channel 3 to S22.
690	Turns the measurement mode of channel 4 to S21.
700	Turns the measurement format of channel 1 to smith chart (R+jx).
710	Turns the measurement format of channel 2 to amplitude/phase.
720	Turns the measurement format of channel 3 to smith chart (R+jx).

730	Turns the measurement format of channel 4 to amplitude/phase.
740	Gets rid of four screen display routine.
750	Comment line.
760	Level SWPEND of wait sweep end routine.
770	Requests the operation end notification.
780	Gets the notification.
790	Gets rid of wait sweep end routine.

MEMO

12 Using Example of External Controller

To connect a computer with this instrument by using GPIB and exchange the data between them, it is necessary to know the computer language and program creation.
This chapter is explained with BASIC (N88-BASIC, QuickBasic, HP-BASIC) and C language. It is not required to know all of them.
However, it is difficult to use GPIB skilfully without knowing the program of BASIC. Therefore, the users are advised to learn something about the computer language before programming that is related to GPIB to be described from now, so as to use it in practice.

The following documents are recommended to be prepared for the programming.

- R3764/66, R3765/67 Operation Manual
- R3764/66, R3765/67 Programming Manual
- GPIB Address Allocation Table
- Manual of Personal Computer
- Manual of GPIB Interface Board

12.1 Before Programming

The GPIB is an interface that connects the network analyzer (this instrument) with other controller or peripheral apparatus with the cable for GPIB.
In this section, a program used to control this instrument with an external controller (Personal computer) that is connected to the instrument by GPIB cable is created.

There exist some computers that can use the GPIB command as soon as the power is switched on, however, it is necessary to load the dedicated programs from the floppy disk according to the controller. Besides, there are programs that can be used to specify the using area of memory before and after loading or output necessary commands to open the input / output port.
Set up the controller to be used after reading the manual attentively.

This chapter describes a method of program creation, centering around an example that takes PC-9801 computer in which the NEC pure GPIB interface board is prepared as an external controller. Employed language is N88 Japanese BASIC.

Set this instrument to the specified condition after the setup of external controller has done. To control this instrument from the external controller, the instrument should be set to GPIB mode and connected by GPIB cable, and the GPIB address of this instrument must be set. (Refer to 12.1.3.)

12.1.1 GPIB Mode

There are the following two types of GPIB mode in this instrument.

- SYSTEM CONTROLLER mode

Enables to measure the function and control the machines connected with this instrument according to the built-in BASIC programs.

- TAKER/ LISTENER mode

Enables to control this instrument with external controller.
Since the built-in BASIC interpreter is shared, the load of external controller can be reduced.

12.1.2 Connecting with This Instrument

The GPIB connector at the rear panel of this instrument is connected with the GPIB connector of external controller with GPIB cable that is optionally available.
When connecting, read the manuals about the interface board and computer to be used carefully, before connecting them.

(Note) The GPIB connector of external controller is on the GPIB interface board. Therefore, it is necessary to purchase the GPIB interface board when using NEC-9800 series and IBM - PC compatible computer.

The type name of GPIB cable varies with the length. The following table shows the name of type and the length.

Table 12-1 GPIB Cable (optionally available)

Type name	Length
408JE - 1P5	0.5m
408JE - 101	1m
408JE - 102	2m
408JE - 104	4m

12.1.3 Setting GPIB Address

When this instrument is controlled by external controller by using GPIB, it is necessary to set the GPIB address of this instrument.
When the address is set, it is stored in non-erasable memory of this instrument. Except for changing the address, it is not necessary to set it again.
The methods of setting GPIB address are different in R3764/66 and R3765/67, and described separately as follows.

(Note) The setting of GPIB address is performed with the front panel keys.

- Setting address in R3764/66 (Refer to " 4.1.5 " of R3764/66 operation manual.)

① Press [•] (CONFIG) in the program mode, and turn to CONFIG mode.
The list of system variables is displayed on the fluorescent display tube.

② Press [•] (CONFIG) and hold it down till the level of ADDRESS is displayed in highlight.

③ Input the address with numeric keys, then press [ENT].

④ To store the setting value to the non-erasable memory of this instrument, press [ENT] again. Since the message for ensuring is displayed on the fluorescence color display-tube, press [ENT] when you want to store it.

- Setting address in R3765/67 (Refer to " 7.10.2 " of R3765/67 operation manual.)

  ① Press [LCL] to get GPIB menu mode.

  ② Select {SET ADDRESSES} from the menu, and convert to SET ADDRESSES menu mode.

  ③ When {ADDRESS R3765} is selected from the next menu, the currently set address is displayed on the active area.

  ④ Here, input the address using numeric keys and press [X1].
  Then, the address of this instrument is set and stored in non-erasable memory.

(Note) ***When setting GPIB address, take care that the address allocated to external controller and the addresses of other connected machines can not be overlapped.***
The address specified here is that used when this instrument is controlled by using an external controller. When this instrument is controlled with built-in BASIC, the address is 31.

12.2 Writing Method of Program

Till now, the program examples using BASIC language of this instrument have been explained. The programs written with BASIC language of this instrument can be executed in all series of R3764/66 and R3765/67.
However, the programs performed on external controller are different according to the computer, operation condition of combined interface board and the language with which the program is written. That is, the writing method of program (Style) varies with the environment of employed controller.
In this chapter, the following are shown as the main external controllers.

Table 12-2 Specialty of Main External Controller

Using language	Computer	GPIB Board
N88-Japanese BASIC	PC-9801	Pure board
HP-BASIC	HP-9000	(Built- in)
QuickBASIC	PC / AT	NI - 488.2
MicrosoftC	PC / AT	NI - 488.2

In this section , a simple program writing method for the programs performed with above-mentioned external controllers is explained. The outlines of creating programs is presented below.

Outline of program:

(1) Initializes controller.

(2) Sets the measurement condition of this instrument.

(3) Searches measurement data with the built-in BASIC of this instrument (preparation).

(4) Loads measurement data from the built-in BASIC of this instrument.

(5) Displays the measurement data on the display of computer.

(6) Ends the program.

12.2.1 Writing Method of Program in N88-BASIC

Turn PC-9801 into BASIC mode and input the following program.

Example 12-1 GPIB Control Program (N88-BASIC) on PC-9801

```
1000  ' ****************************************
1010  ' *                                      *
1020  ' *          GPIB  CONTROL  PROGRAM      *
1030  ' *                                      *
1040  ' *  TARGET:   PC-9801(PURE)             *
1050  ' *  LANGUAGE: N88-BASIC                 *
1060  ' *  FILE:     N88STYLE.BAS              *
1070  ' ****************************************
1080  '
1090  ' (1) INITIALIZE
1100  '
1110  ISET  IFC
1120  ISET  REN
1130  NA = 11
1140  '
1150  ' (2) SETUP
1160  '
1170  PRINT @NA;"OLDC OFF "
1180  PRINT @NA;"FREQ:CENT 150MAHZ"
1190  PRINT @NA;"FREQ:SPAN 300MAHZ"
1200  '
1210  ' (3) SEARCH DATA BY BUILTIN
1220  '
1230  PRINT @NA;"@AP=POINT1(1.5e+8,0)"
1240  PRINT @NA;"@FR=FREQ(AP,0)"
1250  PRINT @NA;"@LV=VALUE(AP,0)"
1260  PRINT @NA;"@PH=VALUE(AP,8)"
1270  '
1280  ' (4) GETTING DATA
```

```
1290  '
1300  PRINT @NA;"@OUTPUT 11;FR"
1310  INPUT @NA;F
1320  PRINT @NA;"@OUTPUT 11;LV"
1330  INPUT @NA;L
1340  PRINT @NA;"@OUTPUT 11;PH"
1350  INPUT @NA;P
1360  '
1370  ' (5) DISPLAY DATA
1380  '
1390  FR=F/10^6
1400  PRINT USING "FREQ  = ####.### [MHz]";FR
1410  PRINT USING "LEVEL = ####.### [dB]";L
1420  PRINT USING "PHASE = ####.### [deg]";p
1430  '
1440  ' (7) ENDING
1450  '
1460  END
```

Then, input RUN, and press the return key to execute the program.

The result is as follows.

Execution result:

```
FREQ  = 150.000 [MHz]
LEVEL =  -3.855 [dB]
PHASE = 148.070 [deg]
```

When using GPIB, it is necessary to output GPIB interface-clear and remote-enable signal first. In N88-BASIC, ISET IFC command and ISET REN command are used.

- ISET IFC

  Send IFC (interface-clear). It is used to initialize GPIB interface.
  When this instrument is controlled from external controller with GPIB, it is must be specified.

- ISET REN

  After REN (remote-enable) is sent, turn this instrument into the remote state.
  When this instrument is in remote state, the [• REMOTE] LED on the front panel of the instrument is switched on.
  When this instrument is not in remote state (in local start), the execution is not performed even although the GPIB command is sent. Therefore, make sure to turn the instrument to the remote state.

When using N88-BASIC, the PRINT @ command is used to send GPIB command to this instrument and the INPUT @ command is used to receive. These commands are corresponding to the OUTPUT and INPUT commands of built-in BASIC in this instrument. The GPIB address

of the instrument is specified after @ (at - mark).
In example 12 - 1" , the address is 11.

When command is sent to the built-in BASIC, @ is added at the beginning of BASIC command. The command proceeded by @ is processed with GPIB command which is used to set measurement condition and with other paths. In Example 12 - 1, the measurement data is obtained by using the built-in BASIC. (For detail, refer to " 12.2.3 Transferring Built-in BASIC Command added '@'.")

12.2.2 Writing Method of Program in HP-BASIC

When this instrument is controlled with HP-9000, the program becomes as follows.

Example 12-2 GPIB Control Program (HP-BASIC) on HP-9000

```
1000  !  ************************************************
1010  !  *                                              *
1020  !  *          GPIB  CONTROL  PROGRAM              *
1030  !  *                                              *
1040  !  * TARGET:   HP-9000(PURE)                      *
1050  !  * LANGUAGE: HP-BASIC                           *
1060  !  * FILE:     HPSTYLE. BAS                       *
1070  !  ************************************************
1080  !
1090  !  (1) INITIALIZE
1100  !
1110  ASSIGN @Na TO 711
1120  !
1130  !  (2) SETUP
1140  !
1150  OUTPUT @Na;"OLDC OFF"
1160  OUTPUT @Na;"FREQ:CENT 150MAHZ"
1170  OUTPUT @Na;"FREQ:SPAN  300MAHZ"
1180  !
1190  !  (3) SEARCH DATA BY BUILTIN
1200  !
1210  OUTPUT @Na;"@AP=POINT1(1.5e+8,0)"
1220  OUTPUT @Na;"@FR=FREQ(AP,0)"
1230  OUTPUT @Na;"@LV=VALUE(AP,0)"
1240  OUTPUT @Na;"@PH=VALUE(AP,8)"
```

```
1250  !
1260  !  (4) GETTING DATA
1270  !
1280  OUTPUT @Na;"@OUTPUT 11;FR"
1290  ENTER  @Na;F
1300  OUTPUT @Na;"@OUTPUT 11;LV"
1310  ENTER  @Na;L
1320  OUTPUT @Na;"@OUTPUT 11;PH"
1330  ENTER  @Na;P
1340  !
1350  !  (5) DISPLAY DATA
1360  !
1370  Fr=F/10^6
1380  PRINT "FREQ  [MHz] = ";
1390  PRINT USING "DDDD.DDD";Fr
1400  PRINT "LEVEL  [dB] = ";
1410  PRINT USING "DDDD.DDD";L
1420  PRINT "PHASE [deg] = ";
1430  PRINT USING "DDDD.DDD";P
1440  !
1450  !  (7) ENDING
1460  !
1470  END
```

To specify the address with HP-BASIC, it is necessary to define I/O path with ASSING command at the start. In this program, @Na of I/O path name is created with line 1110 ASSING @Na TO 711 and it is allocated to this instrument of address11.

The created I/O path name is used when the command is sent to this instrument or when data is received. The command sending is performed with OUTPUT command, while the data receiving is performed with ENTER command. However, the I/O path name must be written after the command.

12.2.3 Writing Method of Program in QuickBASIC

The example of program that used QuickBASIC is shown as follows.

(Note) In this program, NI - 488.2 for PC/AT is used as GPIB interface board.

* NI - 488.2 : Register mark of National Instrument

Example 12-3 GPIB control Program (QuickBASIC) on PC/AT

```
'  ********************************************
'  *                                          *
'  *           GPIB  CONTROL  PROGRAM         *
'  *                                          *
'  *  TARGET:    PC/AT(NI-488.2)              *
'  *  LANGUAGE: QuickBASIC                    *
'  *  FILE:     QBSTYLE.BAS                   *
'  ********************************************

REM  $INCLUDE: 'qbdecl.bas'

DECLARE SUB naout (na%, msg$)
DECLARE SUB nainp (na%, dat$)
DECLARE SUB naerr (msg$)
DECLARE SUB gpiberr (msg$)

'  (1) INITIALIZE
'
BDNAME$ = "GPIBO"
dvname$ = "DEV11"

CALL IBFIND(BDNAME$, brd0%)
IF (brd0% < 0) THEN CALL gpiberr("ibfind1 error")

CALL IBSIC(brd0%)
IF (IBSTA% AND EERR) THEN CALL gpiberr("ibsic error")
CALL IBSRE(brd0%, 1)
IF (IBSTA% AND EERR) THEN CALL gpiberr("ibsre error")

CALL IBFIND(dvname$, na%)
```

```
IF (na% < 0) THEN CALL gpiberr("ibfind2 error")

'  (2) SETUP
'
CALL naout(na%, "OLDC OFF")



CALL naout(na%, "FREQ:CENT 150MAHZ")
CALL naout(na%, "FREQ:SPAN 300MAHZ")

'  (3) SEARCH DATA BY BUILTIN
'
CALL naout(na%, "@AP=POINT1(1,5e+8,0)")
CALL naout(na%, "@LV=VALUE(AP,0)")
CALL naout(na%, "@FR=FREQ(AP,0)")
CALL naout(na%, "@LV=VALUE(AP,0)")
CALL naout(na%, "@PH=VALUE(AP,8)")

'  (4) GETTING  DATA
'
CALL naout(na%, "@OUTPUT 11;FR")
CALL nainp(na%, fdat$)
CALL naout(na%, "@OUTPUT 11;LV")
CALL nainp(na%, ldat$)
CALL naout(na%, "@OUTPUT 11;PH")
CALL nainp(na%, pdat$)

'  (5) DISPLAY DATA
'
F =  VAL(fdat$)
1 =  VAL(ldat$)
p =  VAL(pdat$)
fr =  F / 10^6
PRINT USING "FREQ  = ####.### [MHz]"; fr
PRINT USING "LEVEL = ####.###  [dB]"; l
PRINT USING "PHASE = ####.### [deg]"; p
```

```
'  (7) ENDING
'
CALL IBONL(na%, 0)
CALL IBONL(brd0%, 0)
END

' This routine prints the result of status variables.
'
SUB gpiberr (msg$) STATIC
        PRINT msg$
        PRINT “ibsta=&H″; HEX$(IBSTA%);“ <″;
        IF IBSTA% AND EERR THEN  PRINT “ ERR″;
        IF IBSTA% AND TIMO THEN  PRINT “ TIMO″;
        IF IBSTA% AND EEND THEN  PRINT “ EEND″;
        IF IBSTA% AND SRQI THEN  PRINT “ SRQI″;
        IF IBSTA% AND RQS  THEN  PRINT “ RQS″;
        IF IBSTA% AND CMPL THEN  PRINT “ CMPL″;
        IF IBSTA% AND LOK  THEN  PRINT “ LOK″;
        IF IBSTA% AND RREM THEN  PRINT “ RREM″;
        IF IBSTA% AND CIC  THEN  PRINT “ CIC″;
        IF IBSTA% AND AATN THEN  PRINT “ AATN″;
        IF IBSTA% AND TACS THEN  PRINT “ TACS″;
        IF IBSTA% AND LACS THEN  PRINT “ LACS″;
        IF IBSTA% AND DTAS THEN  PRINT “ DTAS″;
        IF IBSTA% AND DCAS THEN  PRINT “ DCAS″;
        PRINT “>″
        PRINT “iberr=″;IBERR%;
        IF IBERR% = EDVR THEN PRINT “ EDVR <DOS Error>″
        IF IBERR% = ECIC THEN PRINT “ ECIC <Not CIC>″
        IF IBERR% = ENOL THEN PRINT “ ENOL <No listner>″
        IF IBERR% = EADR THEN PRINT “ EADR <Address error>″
        IF IBERR% = EARG THEN PRINT “ EARG <Invalid argment>″
        IF IBERR% = ESAC THEN PRINT “ ESAC <Not Sys Ctrlr>″
        IF IBERR% = EABO THEN PRINT “ EABO <Op. aborted>″
        IF IBERR% = ENEB THEN PRINT “ ENEB <No GPIB board>″
        IF IBERR% = EOIP THEN PRINT " EOIP <Async I/O in prg>"
        IF IBERR% = ECAP THEN RINTT " ECAP <No capability>"
        IF IBERR% = EFSO THEN PRINT " EFSO <Fils sys. error>"
        IF IBERR% = EBUS THEN PRINT " EBUS <Command error>"
        IF IBERR% = ESTB THEN PRINT " ESTB <Status byte lost>"
        IF IBERR% = ESRQ THEN PRINT " ESRQ <SRQ stuck on>"
        IF IBERR% = ETAB THEN PRINT " ETAB <Table Overflow>"
        PRINT "ibcnt="; IBCNT%
        CALL IBONL(na%, 0)
        CALL IBONL(brd0%, 0)
        STOP
END SUB

'  This routine would notify you that the na returned an invalid
   serial  poll

'  response byte.
'
SUB naerr (msg$) STATIC
        PRINT msg$
        PRINT "Status Byte  = "; SPR%
        CALL IBONL(na%, 0)
        CALL IBONL(brd0%, 0)
        STOP
END  SUB

'  This routine nainps the data from device
'
SUB nainp (na%, dat$) STATIC
        RD$ = SPACE$(27)

        CALL  IBRD(na%,  RD$)
        IF (IBSTA% AND EERR) THEN CALL gpiberr("ibrd error")
        dat$ = LEFT$(RD$, IBCNT%)
END  SUB
```

```
'   This routine naouts the command to device
'
SUB naout (dsc%, msg$) STATIC
CALL ibwrt(dsc%, msg$)
emsg$ = "ibwrt error:" + msg$
IF (IBSTA% AND EERR) THEN CALL gpiberr(emsg4)
END  SUB
```

In N-488.2 , various tools necessary for program development except for the interface board are included. When NI-488.2 system is purchased, install it to the computer after reading the manual carefully.
The functions and variables leaded by character ib are the library functions and variables provided by NI-488.2 system. In this program, library functions of IEEE488.1 level are used.

The NI-488.2 library and files used here are described below.

library and files of NI-488.2:

- qbdecl.bas

  A function declaration file of QUICKBASIC program.
  When NI-488.2 is installed,QBASIC directory is created, and then copied with the sample program that is following the directory together.
  Copy this file to one's own operation disk.

- ibfind

  Search the GPIB interface board and GPIB device that is connected with the board, and assign the intrinsic values to them. The value obtained with ibfind is used as argument in NI-488.2 library.
  In Example 12-3 , assigns to board (o) brdo% and this instrument to na% (address 11) .
  There are two systems of library functions: functions operated with board level and functions operated with device specification.

- ibsic

  Show IFC (Interface-clear) message. This operation is performed corresponding to GPIB board.
  When ibsic is executed, performs the initial setting for GPIB interface board is performed, and the board is turned to the CIC (controller-in-charge) state.

- ibsre

  Control REN (remote - enable) signal. This operation is performed corresponding to GPIB board.
  When ibsre (brd$0, 1) is executed, this instrument is specified to listen address, and turned to the remote state at the same time.

- ibwrt

  Send GPIB message to the specified device.
  In Example 12 - 3 , it is called out to send the GPIB command to this instrument in output sub -procedure.

- ibrd

  Receive GPIB message from specified device.
  In Example 12 - 3 , it is called out to receive the data from this instrument in input sub - procedure.

- ibonl

  Release assigned device with ibfind.
  Be sure to call it when the program is ended.

- ibsta%, iberr%, ibcnt%

  These are status variables set with library functions.
  These variables and the value set to the variables are defined with qbdecl. bas file.
  Immediately after the library function is called, the status must be called out to check whether the operation is normal.

Since the GPIB operation command is incorporated into the BASIC language in N88-BASIC and HP-BASIC, the GPIB command is described as the same as usual statement.
Besides, when an error occurs, it can be caught with the BASIC language system, Therefore, it is not necessary to program a special error processing.

However, it is a different case in QuickBASIC. Since the GPIB command is not incorporated into the language, the library functions except for the linked are called up by the use of CALL statement.

Therefore, the occurred error can not be caught with GPIB operation in QuicdBASIC.
That is, whenever the library function is called, the status variable must be checked. If this checking is performed soon after all the functions have been called, the program line will become longer, so that it can not be read easily. There is little problem for those library functions that are not called frequently, but it is not desirable for those called frequently.

In Example 12 - 3 , the dedicated sub-procedures used for calling ibewrt and ibrd are described. They are known as output and input. When the command is sent to this instrument, output is called, and when data is received, input is called only. These sub-procedures are used to call out the library, and then perform the necessary error processing in practice.
However, the error processing is performed practically with sub-procedures gpiberr and baerr. When the error processing is described with sub-procedures, the total program becomes easy to read.

To execute the program of Example 12 - 3 on MS-DOS, the execution file is created in the following sequence.

Sequence of creating execution file:

① Inputs the following command.

```
LIB QBIB.LIB + QBIB.OBJ;
```

Create library QBIB. LIB from object file QBIB. OBJ used for QuickBASIC provided in NI-488. 2 system.
LIB command of MS-DOS is used for creating library.

② Input the following command.

```
BC qbstyle.bas;
```

Compile program file qbstyle. bas with BC command of QuickBASIC. The result can be an object file QBSTYLE. OBJ.

③ Input the following command.

```
LINK QBSTYLE.OBJ,,,QBIB.LIB;
```

Link QBSTYLE. OBJ and QBIB. LIB with LINK command of MS-DOS. The execution file QBSTYLE. EXE is created with the above operation.

④ Input QBSTYLE and press ***Enter***, then it can be executed.

12.2.4 Writing Method of Program in C

The example of program using ANSI-C is shown as follows.

(Note) In this program, NI - 488. 2 for PC/AT is used as GPIB interface board.

Example 12-4 GPIB control Program (ANSI-C) on PC/AT

```
/*
 *          GPIB  CONTROL  PROGRAM
 *
 *  TARGET:    PC/AT(NI-488.2)
 *  LANGUAGE: C (ANSI-C STYLE)
 *  FILE:      CSTYLE. C
 */

#include <stdio.h>
#include <stdlib.h>
#include <setjmp.h>
#include <errno.h>
#include "decl.h"

#define DATSIZ 27

jmp_buf jmpbuf;


void gpiberr(char *msg)
 {
```

```
printf("%s\n", msg);
printf("ibsta=&H%x < ", ibsta );
if (ibsta & ERR)  printf("ERR");
if (ibsta & TIMO) printf("TIMO");
if (ibsta & SRQI) printf("SRQI");
if (ibsta & RQS)  printf("RQS");
if (ibsta & CMPL) printf("CMPL");
if (ibsta & LOK)  printf("LOK");
if (ibsta & CIC)  printf("CIC");
if (ibsta & TACS) printf("TACS");
if (ibsta & LACS) printf("LACS");
if (ibsta & DTAS) printf("DTAS");
if (ibsta & DCAS) printf("DCAS");
printf(" >\n");

printf("iberr= %d ", iberr);
switch (iberr)
   {
  case EDVR: printf("EDVR <DOS Error>"); break;
  case ECIC: printf("ECIC <Not CIC>"); break;
  case ENOL: printf("ENOL <No listner>"); break;
  case EADR: printf("EADR <Address error>"); break;
  case EARG: printf("EARG <Invalid argment>"); break;
  case ESAC: printf("ESAC <Not Sys Ctrlr>"); break;
  case EABO: printf("EABO <Op. aborted>"); break;
  case ENEB: printf("ENEB <No GPIB board>"); break;
  case EOIP: printf("EOIP <Async I/O in prg>"); break;
  case ECAP: printf("ECAP <No capability>"); break;
  case EFSO: printf("EFSO <Fils sys. error>"); break;
  case EBUS: printf("EBUS <Command error>"); break;
  case ESTB: printf("ESTB <Status byte lost>"); break;
  case ESRQ: printf("ESRQ <SRQ stuck on>"); break;
  case ETAB: printf("ETAB <Table Overflow>"); break;
  }
printf("ibcntl= %d\n\n", ibcntl);
longjmp(jmpbuf, EIO);
```

```
  }


void outstr(int dsc, char *cmd)
 {
  ibwrt(dsc, cmd, strlen(cmd));
  if (ibsta & ERR) gpiberr(“ibwrt error”);
}

void inpstr(int dsc, char *cmd, char *buf, unsigned bufsiz)
 {
  if (cmd && *cmd)
     {
      ibwrt(dsc, cmd, strlen(cmd));
      if (ibsta & ERR) gpiberr(“ibwrt error”);
   }
 ibrd(dsc, buf, bufsiz);
 if ( ibsta & ERR) gpiberr(“ibrd error”);
 buf[ibcnt] = ‘\0’;
}


main(int argc, char **argv)
 {
  char   sf[DATSIZ+1];
  char   sl[DATSIZ+1];
  char   sp[DATSIZ+1];
  int    bd = - 1;
  int    na = - 1;
  int    err;

  if ( err = setjmp(jmpbuf))
     {
       if (na >= 0) ibonl(na,0);
       if (bd >= 0) ibonl(bd,0);
```

```
    exit (1);
 }

/*      (1) INITIALIZE
 */
if ((bd = ibfind(“GPIB0”)) < 0) bpiberr(“ibfind error”);
if ((na = ibfind(“DEV11”)) < 0) gpiberr(“ibfind error”);
if (ibsic(bd)    & ERR) gpiberr(“ibsic error”);
if (ibsre(bd, 1) & ERR) gpiberr(“ibsre error”);

/*      (2) SETUP
 */
outstr(na, “OLDC OFF”);
outstr(na, “FREQ:CENT 150MAHZ”);
outstr(na, “FREQ:SPAN 300MAHZ”);

/*      (3) SEARCH DATA BY BUILTIN
 */
outstr(na, “@AP=POINT1(1.5e+8,0)”);
outstr(na, “@LV=VALUE(AP, 0)”);
outstr(na, “@FREQ(AP,)”);
outstr(na, “@LV=VALUE(AP,0)”);
outstr(na, “@PH=VALUE(AP,8)”);

/*      (4) GETTING DATA
 */
inpstr(na, “@OUTPUT 11;FR”, sf, DATSIZ);
inpstr(na, “@OUTPUT 11;LV”, sl, DATSIZ);
inpstr(na, “@OUTPUT 11;PH”, sp, DATSIZ);

/*       (5) DISPLAY DATA
 */
printf(“ FREQ  = %4.4f MHz\n”, atof(sf)/1.0e6);
printf(“ LEVEL = %4.4f  Db\n”, atof(sl));
printf(“ PHASE = %4.4f deg\n”, atof(sp));
```

```
    /*          (6) ENDING
     */
    ibonl(na, 0);
    ibonl(bd, 0);
}
```

A library used for C programming is prepared in NI-488. 2 package and used in this program. The library functions in this program is the same as that of Example 12 - 3 written above.

When this program is compiled with Microsoft, it is performed in the following sequence.
Compile sequence:

① Copy DECL. H and MCIB. OBJ from C package of NI-488. 2 to one's own operation disk.

② Compile the program of Example 12 - 4 .
Input the following command.

```
CL MCSTYLE.C MCIB.OBJ
```

The execution file MCSTYLE. EXE is created with the above operations.

③ Input MCSTYLE and press ***Enter***, then the execution can be performed.

12.3 Remote Control with External Controller

In order to control this instrument remotely from external controller, GPIB commands used to exchange the functions to commands are employed.
The built-in functions performing wave form data analysis, etc. can not be executed by usual GPIB commands. In this case, it is carried out through sending the commands that is obtained by adding the at - mark (@) to the built-in BASIC. When @ is added and sent at the start of transferring command, it is sent to the built-in BASIC and processed.

12.3.1 Transferring Usual GPIB Command

The following programs turn the measurement format of this instrument to the LOGMAG.

Example 12-5 Transferring GPIB Command in N88-BASIC

```
1000 ISET IFC
1010 ISET REN
1020 NA=11
1030 PRINT @NA;"OLDC OFF"
1040 PRINT @NA;"CALC:FORM MLOG"
1050 END
```

Example 12-6 Transferring GPIB Command in HP-BASIC

```
1000 ASSIGN @Na TO 711
1010 OUTPUT @Na;"OLDC OFF"
1020 OUTPUT @Na;"CALC:FORM MLOG"
1030 END
```

Example 12-7 Transferring GPIB Command in QuickBASIC

```
REM $INCLUDE:'qbdecl.bas'

CALL ibfind("GPIB0",bd%)
CALL ibfind("DEV11",na%)
CALL ibsic(bd%)
CALL ibsre(bd%, 1)
CALL ibwrt(na%, "OLDC OFF")
CALL ibwrt(na%, "CALC:FORM MLOG")
CALL ibonl(na%, 0)
CALL ibonl(bd%, 0)
END
```

Example 12-8 Transferring GPIB Command in C

```
#include <stdio.h>
#include <stdlib.h>
#include "decl.h"

main(int argc, char **argv)
 {
  int   bd, na;
  bd = ibfind("GPIB0");
  na = ibfind("DEV11");
  ibsic(bd)
  ibsre(bd, 1);
  ibwrt(na, "OLDC OFF", 8);
  ibwrt(na, "CALC:FORM MLOG", 13);
  ibonl(na, 0);
  ibonl(bd, 0);
}
```

12.3.2 Transferring Built-in BASIC Commands Added by "@"

The commands to which the at mark (@) is added are used when the wave form analysis function, etc. are performed. A part of processing is executed with built-in BASIC, so that the load of external controller can be reduced.

The programs shown below are used to evaluate the maximum value of measurement data using the commands added by @, then receive the analysis data.

Example 12-9 Transferring BASIC Command in N88-BASIC

```
1000 ISET IFC
1010 ISET REN
1020 NA=11
1030 PRINT @NA;"OLDC OFF"
1040 PRINT @NA;"@V=MAX(0,1200,0)"
1050 PRINT @NA;"@OUTPUT 11;V"
1060 INPUT @NA;V
1070 PRINT V
1080 END
```

Example 12-10 Transferring BASIC Command in HP-BASIC

```
1000 ASSIGN @Na TO  711
1010 OUTPUT @Na;"OLDC OFF"
1020 OUTPUT @Na;"@V=MAX(0,1200,0)"
1030 OUTPUT @Na;"@OUTPUT 11;V"
1040 ENTER  @Na;V
1050 PRINT V
1060 END
```

Example 12-11 Transferring BASIC Command in QuickBASIC

```
REM $INCLUDE: 'qbdecl.bas'

CALL ibfind("GPIBO",bd%)
CALL ibfind("DEV11",na%)
CALL ibsic(bd%)
CALL ibsre(bd%,1)
CALL ibwrt(na%,"OLDC OFF")
CALL ibwrt(na%,"@V=MAX(0,1200,0)")
CALL ibwrt(na%,"@OUTPUT 11;V")
dat$=SPACE$(27)
```

```
CALL ibrd(na%,dat$)
dat$=LEFT$(dat$,ibcnt%)
PRINT dat$
CALL ibonl(na%,0)
CALL ibonl(bd%,0)
END
```

Example 12-12 Transferring BASIC Command in C

```
#include <stdo.h>
#include <stdlib.h>
#include "decl.h"

main(int argc, char ** argv)
 {
  char  dat[28];
  int   bd, na ;

  bd = ibfind("GPIBO");
  na = ibfind("DEV11");
  ibsic(bd);
  ibsre(bd, 1);
  ibwrt(na,"OLDC OFF", 8);
  ibwrt(na,"@V=MAX(0,1200,0)", 16);
  ibwrt(na,"@OUTPUT 11;V",12);
  ibrd(na, dat, 27);
  dat[ibcnt] = '\0';
  printf(dat);
  ibonl(na, 0);
  ibonl(bd, 0);
}
```

12.4 Detecting Scan End

In this section, a method of detecting scan end with the external controller is explained.
First , the trigger mode of this instrument is switched to INIT : CONT OFF.
When INIT command is sent in this state, only one scan is executed.
Next, the scan end is detected from the bit state of status register (A register used to report the current state of the machine.)

When the scan is ended, one of the status register known as Sweeping of Standard Operation Event Status Register is set to 1.
Since the bit of Standard Operation Event Enable Register corresponding to this status bit is set to 1, a service request can be generated when the scan is ended. (Refer to "4 Status byte" of R3764/66, R3765/67 Programming Manual.)

12.4.1 Detecting Scan End with N88-BASIC

The following program is used to detect the scan end during the measurement by interrupting processing with the N88-BASIC of PC-9801.

When the bit fills up the status register, a service request can be generated. In the following program , since the Standard Operation Event Enable Register and SRE are enabled, SRQ is generated when the scan is ended.

If the interrupt processing is declared in the program, when SRQ occurs, the processing performed till that time will be interrupted, and perform the processing specified with interrupting . In this program, when SRQ occurs, the infinite loop interrupts and the performance jumps to level * MEAS.END.

Example 12-13 Detecting Scan End with N88-BASIC

```
1000 ISET IFC
1010 ISET REN
1020 NA=11
1030 POLL NA,P
1040 ON SRQ GOSUB *MEAS.END
1050  '
1060  *MEAS.SETUP
1070      PRINT @NA;"OLDC OFF"
1080      PRINT @NA;"INIT:CONT OFF"
1090      PRINT @NA;"*CLS;*SRE 128;:STAT:OPER:ENAB 8"
1100      PRINT @NA;"INIT"
1110      SRQ ON
1120  '
1130  *MEAS.WAIT
1140      GOTO  *MEAS.WAIT
1150  '
1160  *MEAS.END
```

```
1170        POLL NA,P
1180        P = P AND 128
1190        IF P<>128 THEN RETURN
1200        PRINT "SWEEP END"
1210        END
```

12.4.2 Detecting Scan End with HP-BASIC

The following program is used to detect the scan end during the measurement by interrupting the processing with the HP-BASIC of HP-9000 series.
It defines the branch destination (Measand) of interrupting with ON INTR command, and enables the interrupting to be performed with ENABLE INTR.

Example 12-14 Detecting Scan End with HP-BASIC

```
1000 ASSIGN @Na TO 711
1010 !
1020 OUTPUT @Na;"OLDC OFF"
1030 OUTPUT @Na;":INIT:CONT OFF"
1040 Stat=SPOLL(@Na)
1050 OUTPUT @Na;"*CLS;*SRE 128;:STAT:OPER:ENAB 8"
1060 OUTPUT @Na;"INIT"
1070 ON INTR 7 GOTO Measend
1080 ENABLE INTR 7;255
1090 Measwait:!
1100     GOTO  Measwait
1110 !
1120 Measend:!
1130     PRINT "SWEEP END"
1140 END
```

12.4.3 Detecting Scan End with QuickBASIC

The following program is used to detect the scan end during measurement using NI-488.2 library functions with QuickBASIC on PC/AT.
The spoll is a sub-procedure used to detect service request.
That waits until the series requests are generated with the library function ibwait of NI-488.2 and reads in the status byte with the library function ibrsp of NI-488.2.
When error occurred some where, sub-procedure gpiberr is called out.
The gpiberr is called when error occurred to execute STOP command after displaying the error message.

For the using method of library functions, refer to manual of NI-488.2.

Example 12-15 Detecting Scan End with QuickBASIC

```
REM $INCLUDE: ‘qbdecl.bas’

DECLARE SUB spoll (dsc%, spr%)
DECLARE SUB gpiberr (msg$)

CALL ibfind(“GPIBO”, bd%)
CALL ibfrind(“DEV11”, na%)
CALL ibsic(bd%)
CALL ibsre(bd%, 1)
CALL ibwrt(na%, “OLDC OFF”)
CALL ibwrt(na%, “:INIT:CONT OFF”)
CALL ibrsp(na%, spr%)
CALL ibwrt(na%, “*CLS;*SRE 128;:STAT:OPER:ENAB 8”)
CALL ibwrt(na%, “SWE:TIME 0”)
CALL ibwrt(na%, “INIT”)
CALL spoll(na%, spr%)
PRINT “SWEEP END”
CALL ibonl(na%, 0)
CALL ibonl(bd%, 0)
END

’ Error Handler
’
SUB gpiberr (msg$) STATIC
        PRINT msg$
        PRINT “ibsta=&H”; HEX$(ibsta%)
        CALL ibonl(na%, 0)
        CALL ibonl(bd%, 0)
        STOP
END SUB

’ SRQ Handler
’
SUB spoll (na%, spr%) STATIC
        spr% = 0
        mask% = &H800
        CALL ibwait(na%, mask%)
        IF (ibsta% AND EERR) THEN CALL gpiberr(“ibwait error”)
        CALL ibrsp(na%, spr%)
        IF (ibsta% AND EERR) THEN CALL gpiberr (“ibsta error”)
        IF (spr%<>&HCO) THEN CALL gpiberr(“R3764/66,R3765/67 SRQ
                                                              error”)
END SUB
```

12.4.4 Detecting Scan End with C

The following program is used to detect the scan end during measurement using NI-488.2 library functions with Microsoft C on PC/AT.
The same as described in previous section, it detects the service request with spoll and returns 0 when succeeded.

For the using method of library functions, refer to manual of NI-488.2.

Example 12-16 Detecting Scan End with Microsoft C

```
#include <stdio.h>
#include <stdlib.h>
#include “decl.h”

static int spoll(int dsc, char *spr)
 {
  if (ibwait(dsc, TIMO|RQS) & (ERR|TIMO))
     {
      fprintf(stderr, “ibwait error: &H%x\n”, ibsta);
      return -1;
   }
  if (ibrsp(dsc, spr) & ERR)
     {
      fprintf(stderr, “ibrps error: &H%x\n”, ibsta);
      return -1;
   }
  if (*spr & 0xff) != 0x0C0)
     {
      fprintf(stderr, “R3764/66, R3765/67 error: &H%x\n”, *spr);
      return -1;
   }
 return 0;
}

main(int argc, char **argv)
 {
  int   bd, na;
  int   err;
  char  spr;

  bd = ibfind(“GPIB0”);
  na = ibfind(“DEV11”);
  ibsic(bd);
  ibsre(bd, 1) ;
  ibwrt(na, “OLDC OFF”, 8);
  ibwrt(na, “:INIT:CONT OFF”, 14);
  ibrsp(na, &spr);
  ibwrt(na, “*CLS;*SRE 128;:STAT:OPER:ENAB 8”, 31);
  ibwrt(na, “SWE:TIME 0”, 10);
  ibwrt(na, “INIT”, 4);
  if ((err = spoll(na, &spr)) == -1)
     {
      ibonl(na, 0);
      ibonl(bd, 0);
      exit(1);
   }
  printf(“SWEEP END\n”);
  ibonl(na, 0);
  ibonl(bd, 0);
}
```

12.5 Transferring Trace Data

The data transferring between an external controller and built-in BASIC can be performed in either way, with ASCII format or binary format.
In binary format, a format corresponding to the external controller can be selected from six types of format. (Refer to "7.7 Format Subsystem" of R3764/66, R3765/67 Programming Manual.)
In ASCII format, the data transferring can be performed with a simple operation.
The speed of transferring data in binary format is higher than in ASCII format. When the data transferring speed is not required specially, the ASCII format is applicable for its simple program. However, when a high speed of data transferring is required, the binary format is used.

This section describes the programs transferring data in two different format: ASCII and binary.

12.5.1 Transferring Trace Data from This Instrument to PC-9801

The following programs are used to assign the trace data of this instrument to the array TR of N88-BASIC.
The data transferring with ASCII format is shown in Example 12 - 17 and the that with binary format is shown in Example 12-18.
In N88-BASIC, there is no GPIB command used for transferring block data. Therefore, in this program, the data receiving part is described with machine language, and the BIOS routine is called up directly.
In addition, Microsoft single-precision floating binary is used in this library format.

Example 12-17 Data Input of PC-9801 (ASCII format)

```
1000 '  ********************************************
1010 '  *                                          *
1020 '  *           INPUT TRACE DATA FROM NA       *
1030 '  *               (ACSII FORMAT)             *
1040 '  *                                          *
1050 '  *  TARGET:    PC-9801(PURE)                *
1060 '  *  LANGUAGE: N88-BASIC                     *
1070 '  *  FILE:      N88TINPA.BAS                 *
1080 '  ********************************************
1090 '
1100 DIM TR$ ( 1201, 2 )
1110 '
1120  ISET IFC
1130  ISET REN
1140 'NA=11
1150 '
1160  *TINP.EXEC
1170      PC98=IEEE(1) AND &H1F
1180      CMD DELIM=3
```

```
1190        PRINT @NA;“OLDC OFF” @
1200        PRINT @NA;“FORM:DATA ASC” @
1210        FOR N=1 TO 2
1220            PRINT @NA;“TRAC? FDAT”+CHR$(48+N) @
1230            WBYTE &H3F,&H5F,&H40+NA,&H20+PC98;
1240            GOSUB *TINP.RECEIVE
1250            WBYTE &H3F,&H5F,&H40+PC98,&H20,NA;
1260        NEXT
1270        GOSUB *TINP.PRINT
1280        WBYTE &H3F,&H5F;
1290        END
1300    '
1310    *TINP.PRINT
1320        PRINT @NA;“SWE:POIN?”  @
1330        INPUT @NA;PTS
1340        FOR I=0 TO PTS-1
1350            PRINT 1,TR$(I,1),TR$(I,2)
1360        NEXT
1370        RETURN
1380    '
1390    *TINP.RECEIVE
1400        I%=0
1410        A$=” ”
1420        *RECEIVE.NEXT
1430            RBYTE  ; D%
1440            S%=IEEE(2) AND &H8
1450            IF D%=44 THEN *RECEIVE.SEPARATE
1460            A$=A$+CHR$(D%)
1470            IF S%<>0 THEN *RECEIVE.SEPARATE
1480            GOTO *RECEIVE.NEXT
1490        *RECEIVE.SEPARTE
1500            TR$(I%,N)=LEFT$(A$,22)
1510            I%=I%+1
1520            A$=“ ”
1530            LOCATE 0,24:PRINT I%;
1540            IF S%=0 THEN *RECEIVE.NEXT
```

```
1550         PRINT
1560     RETURN
```

Example 12-18 Data Input of PC - 9802 (Binary format)

```
1000 ' ***********************************************
1010 ' *                                             *
1020 ' *         INPUT TRACE DATA FROM NA            *
1030 ' *             (BINARY FORMAT)                 *
1040 ' *                                             *
1050 ' *  TARGET:   PC-9801(PURE)                    *
1060 ' *  LANGUAGE: N88-BASIC                        *
1070 ' *  FILE:     N88TINPB.BAS                     *
1080 ' ***********************************************
1090 '
1100 CLEAR &H100:DEF SEG=SEGPTR(2)
1110 DIM TR1!(1202),TR2!(1202)
1120 GOSUB *SETGPIB.RECEIVE
1130 '
1140 ISET IFC
1150 ISET REN
1160 NA=11
1170 '
1180 *TINP.EXEC
1190     PC98=IEEE 1) AND &H1F
1200     CMD DELIM=3
1210     PRINT @NA;"OLDC OFF" @
1220     PRINT @NA;"FORM:DATA MBIN,32" @
1230     FOR N=1 TO 2
1240         PRINT @NA;"TRAC:DATA? FDAT"+CHR$(48+N) @
1250         WBYTE &H3F,&H5F,&H40+NA,&H20+PC98;
1260         NUM%=4816
1270         IF N=1 THEN CALL RECEIVE.DATA(TR1!(0),NUM%)
1280         IF N=2 THEN CALL RECEIVE.DATA(TR2!(0),NUM%)
1290         WBYTE &H3F,&H5F,&H40+PC98,&H20+NA;
1300      NEXT
1310      GOSUB *TINP.PRINT
```

```
1320       WBYTE *H3F,&H5F;
1330       END
1340 '
1350 *TINP.PRINT
1360      PRINT @NA;“SWE:POIN?” @
1370      INPUT @NA;PTS
1380      FOR I=1 TO PTS
1390          PRINT I,TR1!(I+1),TR2!(I+1)
1400      NEXT
1410      RETURN
1420 '
1430 '  Call GPIB BIOS of RECEIVE DATA
1440 '  SYNTAX: CALL RECEIVE.DATA(VAR,SIZE%)
1450 '
1460 *SETGPIB. RECEIVE
1470      RECEIVE.DATA = &HO
1480      RESTORE *GPIB.BIOS.RECEIVE
1490      FOR ADR = 0 TO &H38
1500          READ BYTE: POKE ADR,BYTE
1510      NEXT
1520      RETURN
1530 '
1540 *GPIB. BIOS. RECEIVE
1550      DATA &H50                :'PUSH  AX
1560      DATA &H51                :'PUSH  CX
1570      DATA &H52                :'PUSH  DX
1580      DATA &H06                :'PUSH  ES
1590      DATA &H56                :'PUSH  SI
1600      DATA &H57                :'PUSH  DI
1610      DATA &H55                :'PUSH  BP
1620      DATA &H53                :'PUSH  BX
1630      DATA &H8B,&H4F,&H02      :'MOV   CX,2[BX]
1640      DATA &H8E,&HC1           :'MOV   ES,CX
1650      DATA &H8B,&H37           :'MOV   SI,[BX]
1660      DATA &H26,&H8B,&H0C      :'MOV   CX,ES:[SI]   ; DATA LENGTH
1670      DATA &H8B,&H7F,&H04      :'MOV   DI,4[BX];    ; DATA OFFSET
```

```
1680    DATA &H8E,&H47,&H06      :'MOV   ES,6[BX]      ; SEGMENT BASE
1690    DATA &HBB,&H00,&H00      :'MOV   BX,00H        ; COMMAND LENGTH
1700    DATA &HBE,&H00,&H00      :'MOV   SI,00H        ; COMMAND OFFSET
1710    DATA &HB0,&H80           :'MOV   AL,80H        ; EOI  ONLY
1720    DATA &HB4,&H05           :'MOV   AH,05H        ; RECEIVE DATA
1730    DATA &HCD,&HD1           :'INT   0DIH          ; CALL GPIB BIOS
1740    DATA &H5B                :'POP   BX
1750    DATA &H53                :'PUSH  BX
1760    DATA &H8B,&H4F,&H02      :'MOV   CX,2[BX]
1770    DATA &H8E,&HC1           :'MOV   ES,CX
1780    DATA &H8B,&H37           :'MOV   SI,[BX]
1790    DATA &H26,&H89,&H14      :'MOV   ES:[SI],DX
1800    DATA &H5B                :'POP   BX
1810    DATA &H5D                :'POP   BP
1820    DATA &H5F                :'POP   DI
1830    DATA &H5E                :'POP   SI
1840    DATA &H07                :'POP   ES
1850    DATA &H5A                :'POP   DX
1860    DATA &H59                :'POP   CX
1870    DATA &H58                :'POP   AX
1880    DATA &HCF                :'IRET
1890    ' TOTAL 39H byte
```

When binary format is used, the first 8 bytes becomes the header. Here, the data including header are loaded in array TR.
TR is a single-precision floating array, the data count per point is two, and the byte count per point is 8 bytes. Thus, in this program, the original trace data is stored from the place where the subscript of array TR is 2.

12.5.2 Transferring trace Data from PC - 9801 to This Instrument

The following programs are used to transfer the array data from the program of N88 - BASIC program to this instrument.

(Caution) ***This program can not be used originally.***
When using the program in practice, first get the data with the trace data input program presented above and store it in a file, etc. then run the program from reading the data into the array TR.

Example 12-19 Data Output of PC-9801 (ASCII format)

```
1000  '   *********************************************
1010  '   *                                           *
1020  '   *          OUTPUT  TRACE  DATA  TO  NA      *
1030  '   *              (ASCII FORMAT)               *
1040  '   *                                           *
1050  '   *  TARGET:   PC-9801(PURE)                  *
1060  '   *  LANGUAGE: N88-BASIC                      *
1070  '   *  FILE:     N88TOUTA.BAS                  *
1080  '   *********************************************
1090  '
1100  DIM TR$(1201,2)

1120  '
5000  '
5010  ISET IFC
5020  ISET  REN
5030  NA=11
5040  '
5050  *TOUT.EXEC
5060      PC98=IEEE(1) AND &H1F
5070      CMD DELIM=3
5080      PRINT @NA;"OLDC OFF" @
5090      PRINT @NA;"FORM:DATA ASC" @
5100      FOR N=1 TO 2
5110          PRINT @NA;"TRAC:DATA FDAT"+CHR$(48)+"," @
5120          WBYTE &H3F,&H5F,&H40+PC98,&H20+NA;
5130          GOSUB *TOUT.SEND
5140      NEXT
5150      WBYTE &H3F,&H5F;
5160      END
5170 '
5180 *TOUT.SEND
5190     I%=0
5200     *SEND. NEXT
5210         IF TR$(I%+1,N)=" " GOTO *SEND.LAST
```

```
5220         PRINT @;TR$(I%,N)+CHR$(44)
5230         LOCATE 0,23:PRINT I%,TR$(I%,N);
5240         I%=I%+1
5250         GOTO *SEND.NEXT
5260     *SEND.LAST
5270         PRINT @;TR$(I%,N) @
5280         LOCATE 0,23:PRINT I%,TR$(I%,N)
5290     RETURN
```

Example 12-20 Data Output of PC-9801 (Binary format)

```
1010 '  ********************************************
1020 '  *          OUTPUT TRACE DATA TO NA          *
1030 '  *              (BINARY FORMAT)              *
1040 '  *                                           *
1050 '  *  TARGET:   PC-9801(PURE)                  *
1060 '  *  LANGUAGE: N88-BASIC                      *
1070 '  *  FILE:     N88TOUTB. BAS                  *
1080 '  ********************************************
1090 '
1100 CLEAR &H100:DEF SEG=SEGPTR(2)
1110 DIM TR1!(1202),TR2!(1202)
1120 GOSUB *SETGPIB.SEND
1130 '
1140 ISET IFC
1150 ISET REN
1160 NA=11
1170 '
1180 *TOUT.EXEC
1190     PC98=IEEE(1) AND &H1F
1200     CMD DELIM=3
1210     PRINT @NA;"OLDC OFF" @
1220     PRINT @NA;"FORM:DATA MBIN,32" @
1230     NUM%=4816
1240     FOR N=1 TO 2
1250         PRINT @NA;"TRAC:DATA FDAT"+CHR$(48)+","
1260         WBYTE &H3F,&H5F,&H40+PC98,&H20+NA;
```

```
1270            IF N=1 THEN CALL SEND.DATA(TR1(0),NUM%)
1280            IF N=2 THEN CALL SEND.DATA(TR2(0),NUM%)
1290        NEXT
1300        WBYTE &H3F,&H5F;
1310        END
1320 '
1330 ' Call GPIB BIOS of SEND DATA
1340 ' SYNTAX: CALL SEND.DATA(VAR.SIZE%)
1350 '
1360 *SETGPIB.SEND
1370        SEND.DATA = &H39
1380        RESTORE *GPIB.BIOS.SEND
1390        FOR ADR = &H39 TO &H65
1400            READ BYTE: POKE ADR,BYTE
1410        NEXT
1420        RETURN
1430 '
1440 *GPIB.BIOS.SEND
1450        DATA &H50               :'PUSH    AX
1460        DATA &H51               :'PUSH    CX
1470        DATA &H52               :'PUSH    DX
1480        DATA &H06               :'PUSH    ES
1490        DATA &H56               :'PUSH    SI
1500        DATA &H57               :'PUSH    DI
1510        DATA &H55               :'PUSH    BP
1520        DATA &H53               :'PUSH    BX
1530        DATA &H8B,&H4F,&H02     :'MOV     CX,2[BX]
1540        DATA &H8E,&HC1          :'MOV     ES, CX
1550        DATA &H8B,&H37          :'OV      SI,[BX]
1560        DATA &H26,&H8B,&H0C     :'MOV     CX,ES:[SI]    ; DATA LENGTH
1570        DATA &H8B,&H7F,&H04     :'MOV     DI,4[BX]      ; DATA OFFSET
1580        DATA &H8E,&H47,&H06     :'MOV     ES,6[BX]      ; SEGMENT BASE
1590        DATA &HBB,&H00,&H00     :'MOV     BX,00H        ; COMMAND LENGTH
1600        DATA &HBE,&H00,&H00     :'MOV     SI,00H        ; COMMAND OFFSET
1610        DATA &HB0,&H80          :'MOV     AL,80H        ; EOI ONLY
1620        DATA &HB4,&H04          :'MOV     AH,04H        ; RECEIVE DATA
```

```
1630    DATA &HCD,&HD1          :'INT   0D1H       ; CALL GPIB BIOS
1640    DATA &H5B               :'POP   BX
1650    DATA &H5D               :'POP   BP
1660    DATA &H5F               :'POP   DI
1670    DATA &H5E               :'POP   SI
1680    DATA &H07               :'POP   ES
1690    DATA &H5A               :'POP   DX
1700    DATA &H59               :'POP   CX
1710    DATA &H58               :'POP   AX
1720    DATA &HCF               :'IRET
1730    ' TOTAL 2DH byte
```

12.5.3 Transferring Trace Data from This Instrument to HP - BASIC

The following programs are used to assign trace data of this instrument to array Tr of HP-BASIC. In binary format, IEEE Double-precision floating is selected.

Example 12-21 Data Input of HP-BASIC (ASCII format)

```
1000  !  ***********************************************
1010  !  *                                             *
1020  !  *           INPUT TRACE DATA FROM NA          *
1030  !  *               (ASCII  FORMAT)               *
1040  !  *                                             *
1050  !  *   TARGET:    HP-9000(PURE)                  *
1060  !  *   LANGUAGE:  HP-BASIC                       *
1070  !  *   FILE:      HPTINPA.BAS                    *
1080  !  ***********************************************
1090  !
1100  ASSIGN @Na TO 711
1110  DIM Tr1(1:201),Tr2(1:201)
1120  !
1130  OUTPUT @Na;"OLDC OFF"
1140  OUTPUT @Na;"SWE:POIN 201"
1150  OUTPUT @Na;"FORM:DATA ASC"
1160  !
1170  OUTPUT @Na;"TRAC? FDAT1"
1180  ENTER @Na;Tr1(*)
1190  !
```

```
1200  OUTPUT @Na;"TRAC? FDAT2"
1210  ENTER @Na;Tr2(*)
1220  !
1230  FOR I=1 TO 201
1240    PRINT "No.";I;":";Tr1(I),Tr2(I)
1250  NEXT I
1260  !
1270  END
```

Example 12-22 Data Input of HP-BASIC (BINARY format)

```
1000  !  *************************************************
1010  !  *                                               *
1020  !  *           INPUT TRACE DATA FROM NA            *
1030  !  *               (BINARY FORMAT)                 *
1040  !  *                                               *
1050  !  *   TARGET:    HP-9000(PURE)                    *
1060  !  *   LANGUAGE:  HP-BASIC                         *
1070  !  *   FILE:      HPTINPB.BAS                      *
1080  !  *************************************************
1090  !
1100  ASSIGN @Na TO 711
1110  ASSIGN @Dt TO 711;FORMAT OFF             ! BINARY DATA PASS
1120  DIM Tr(1:201,1:2)
1130  !
1140  OUTPUT @Na;"OLDC OFF"
1150  OUTPUT @Na;"SWE:POIN 201"
1160  OUTPUT @Na;"FORM:BORD NORM"
1170  OUTPUT @Na;"FORM:DATA REAL,64"
1180  !
1190  OUTPUT @Na;"TRAC? FDAT1"
1200  ENTER  @Na USING "%,8A";Header$          ! READ HEADER STRING
1210  ENTER  @Dt;Tr1(*)                        ! READ ALL TRACE DATA
1220  ENTER  @Na USING "%,1A";Terminate$       ! READ TERMINATOR
1230  !
1240  OUTPUT @Na;" TRAC? FDAT2"
1250  ENTER @Na USING "%,8A";Header$           ! READ HEADER STRING
```

```
1260  ENTER @Dt;Tr2(*)                      ! READ ALL TRACE DATA
1270  ENTER @Na USING "%,1A";Terminate$     ! READ TERMINATOR
1280  !
1290  I=1
1300  WHILE I <202
1310    PRINT "No.";I;":";Tr1(I),Tr2(I)
1320    I=I+1
1330  END WHILE
1340  !
1350  END
```

12.5.4 Transferring Trace Data from This Instrument to QuickBASIC

The following programs are used to assign the trace data of this instrument into array tr of QuickBASIC being executed with PC/AT.

(Note) NI - 488.2 interface board and library functions are used.

Example 12-23 Data Input of QuickBASIC (ASCII format)

```
'   ************************************************
'   *                                              *
'   *            INPUT TRACE DATA FROM NA          *
'   *                (ASCII  FORMAT)               *
'   *                                              *
'   *  TARGET:   PC/AT(NI-488.2                    *
'   *  LANGUAGE: QuickBASIC                        *
'   *  FILE:     QBTINPA.BAS                       *
'   ************************************************

REM $INCLUDE: 'qbdecl.bas'

DECLARE SUB  gpinit (bdname$, bd%)
DECLARE SUB  nainit (bd%, naname$, dv%)
DECLARE SUB  tisetup (dv%, name$)
DECLARE SUB  tireceive (bd%, dat!())
DECLARE SUB  tiprint (dv%, dat1!(), dat2!())
DECLARE SUB  prterr (msg$)

DIM  tr1!(1 TO 201), tr2!(1 TO 201)
CALL gpinit("GPIB0", bd%)
CALL nainit(bd%, "DEV11", na%)
CALL tisetup(na%,  FDAT1")       ' trace 1
CALL tireceive(bd%, tr1!())
CALL tisetup(na%, "FDAT2")       ' trace 2
CALL tireceive bd%, tr2!())
CALL tiprint(na%, tr1!(), tr2!())
CALL ibonl(na%, 0)
CALL ibonl(dv%, 0)
END

'    This routine open the gpib board and initialize
'
SUB gpinit  (bdname$, bd%) STATIC

        CALL ibfind(bdname$, bd%)      ' OPEN BOARD
        IF (bd% < 0) THEN
                CALL prterr("ibfind error")
                STOP
        END IF
        CALL ibsic(bd%)                 ' INTERFACE CLEAR
        IF (ibsta% AND EERR) THEN
                CALL prterr("ibsic error")
                CALL ibonl(bd%, 0)
                STOP
        END IF
        CALL ibsre(bd%, 1)               ' REMOTE ENABLE
        IF (ibsta% AND EERR) THEN
                CALL prterr("ibsre error")
                CALL ibonl(bd%, 0)
                STOP
        END IF
END SUB

'    This routine open N.A and initialize
'
```

```
SUB nainit (bd%, dvname$, dv%) STATIC

        CALL ibfind(dvname$, dv%)
        IF (dv%< 0) THEN
               CALL prterr("ibfind error")
               CALL ibonl(bd%, 0)
               STOP
        END  IF

        cmb$ = "OLDC OFF"
        CALL ibwrt(dv%, cmd$)
        IF (ibsta% AND EERR) THEN
                CALL prterr("ibwrt error")
                CALL ibonl(dv%, 0)
                CALL ibonl(bd%, 0)
                STOP
        END  IF

END  SUB

'  This  routine  prints  the  result  of  status  variables.
'
SUB prterr (msg$) STATIC

        PRINT msg$
        PRINT "ibsta=&H" ; HEX$(ibsta%);" <";
        IF ibsta% AND EERR THEN PRINT " ERR";
        IF ibsta% AND TIMO THEN PRINT " TIMO";
        IF ibsta% AND EEND THEN PRINT " EEND";
        IF ibsta% AND SRQI THEN PRINT " SRQI";
        IF ibsta% AND RQS  THEN PRINT " RQS";
        IF ibsta% AND CMPL THEN PRINT " CMPL";
        IF ibsta% AND LOK  THEN PRINT " LOK";
        IF ibsta% AND RREM THEN PRINT " RREM";
        IF ibsta% AND CIC  THEN PRINT " CIC";
        IF ibsta% AND AATN THEN PRINT " AATN";
        IF ibsta% AND TACS THEN PRINT “ TACS ”;
        IF ibsta% AND LACS THEN PRINT “ LACS ”;
        IF ibsta% AND DTAS THEN PRINT “ DTAS ”;
        IF ibsta% AND DCAS THEN PRINT “ DCAS ”;
        PRINT  “ > ”

        PRINT “iberr=”; iberr%;
        IF iberr% = EDVR THEN PRINT “ EDVR <DOS Error>”
        IF iberr% = ECIC THEN PRINT “ ECIC <NOT CIC>”
        IF iberr% = ENOL THEN PRINT “ ENOL <NO listner>
        IF iberr% = EADR THEN PRINT “ EADR <Address error>”
        IF iberr% = EARG THEN PRINT “ EARG <Invalid argment>”
        IF iberr% = ESAC THEN PRINT “ ESAC <Not Sys Ctrlr>”
        IF iberr% = EABO THEN PRINT “ EABO <Op. aborted>”
        IF iberr% = ENEB THEN PRINT “ ENEB <No GPIB board>
        IF iberr% = EOIP THEN PRINT “ EOIP <Async I/O in prg>”
        IF iberr% = ECAP THEN PRINT “ ECAP <No capability>”
        IF iberr% = EFSO THEN PRINT “ EFSO <Fils sys. error>”
        IF iberr% = EBUS THEN PRINT “ EBUS <Command error>”
        IF iberr% = ESTB THEN PRINT “ ESTB <Status byte lost>”
        IF iberr% = ESRQ THEN PRINT “ ESRQ <SRQ stuck on>”
        IF iberr% = ETAB THEN PRINT “ ETAB <Table Overflow>”
        PRINT “ibcnt=”; ibcnt%

END SUB
 '   This routine print received data
'
SUB tiprint (dv%, dat1!(), dat2!()) STATIC

        cmd$ = “SWE:POIN?”
        CALL ibwrt(dv%, cmd$)
        cmd$ = SPACE$(23)
        CALL ibrd(dv%, cmd$)
        pts% = VAL(cmd$)
        FOR num% = 1 TO pts%
                PRINT num%, dat1!(num%), dat2!(num%)
        NEXT

END  SUB

'   This routine receives trace data
'
SUB tireceive (bd%, buf!()) STATIC

        v% = &H427
        CALL ibeos(bd%, v%)

        cmd$ = "?_K"           ' UNL UNT MLA 0 TAD 11
        CALL ibcmd(bd%, cmd$)
        IF (ibsta AND EERR) THEN
          CALL prterr("ibcmd error")
          STOP
        END IF
        eoi% = 0
        num% = 1
        WHILE eoi% = 0
                cmd$ = SPACE$(23)     ' [S#.###################,]
                CALL ibrd(bd%, cmd$)      ' READ ONE DATA
                dat$ = LEFT$(cmd$, 22)
                buf!(num%) = VAL(dat$)
                eoi% = ibsta% AND EEND
                num% = num% + 1
         WEND

         v% = &H40A
         CALL ibeos(bd%, v%)

END SUB
'   This  routine  setups
'
SUB tisetup (dv%, name$) STATIC
```

```
            cmd$ = "SWE:POIN 201"                 '  BASE  COMMAND
            CALL ibwrt(dv%, cmd$)
            cmd$ = "FORM:DATA ASC;:TRAC?" + name$' TRACE INPUT COMMAND
            CALL ibwrt(dv%, cmd$)

    END  SUB
```

12.5.5 Transferring Trace Data from This Instrument to C

The following programs is used to assign the trace data of this instrument into array tr of C being executed with PC/AT.

(Note) NI-488.2 interface board and library functions are used.

Example 12-24 Data Input of C (ASCII format)

```
/*
 *       INPUT TRACE DATA FROM NA
 *          (ASCII FORMAT)
 *
 *  TARGET:   PC/AT(NI-488.2)
 *  LANGUAGE: C (ANSI-C STYLE)
 *  FILE:     MCTINPA.C
 */

#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include "decl.h"

double  databuf1[210], databuf2[201];             /* data buffer */

/* prterr - print gpib error message and status code
 */
static void prterr(char *msg)
 {
  printf("%s\n", msg);
  printf("ibsta=&H%x < ", ibsta);
  if (ibsta & ERR)  printf("ERR");
  if (ibsta & TIMO) printf("TIMO");
  if (ibsta & SRQI) printf("SRQI");
  if (ibsta & RQS)  printf("RQS");
  if (ibsta & CMPL) printf("CMPL");
  if (ibsta & LOK)  printf("LOK");
  if (ibsta & CIC)  printf("CIC");
  if (ibsta & TACS) printf("TACS");
  if (ibsta & LACS) printf("LACS");
  if (ibsta & DTAS) printf( DTAS");
  if (ibsta & DCAS) printf ( " DCAS " );
  printf(' >\n);

  printf("iberr= %d", iberr);
  switch(ibberr)
     {
    case EDVR: printf("EDVR <DOS Error>"); break;
    case ECIC: printf("ECIC <Not CIC>"); break;
    case ENOL: printf("ENOL <No listner>"); break;
    case EADR: printf("EADR <Address error>"); break;
    case EARG: printf("EARG <Invalid argment>"); break;
    case ESAC: printf("ESAC <Not Sys Ctrlr>"); break;
    case EABO: printf("EABO <Op. aborted>"); break;
    case ENEB: printf("ENEB <No GPIB board>"); break;
    case EOIP: printf("EOIP <Async I/O in prg>"); break;
    case ECAP: printf("ECAP <No capability>"); break;
    case EFSO: printf("EFSO <Fils sys. error>"); break;
    case EBUS: printf("EBUS <Command error>"); break;
    case ESTB: printf("ESTB <Status byte lost>"); break;
    case ESRQ: printf("ESRQ <SRQ stuck on>"); break;
    case ETAB: printf("ETAB <Table Overflow>"); break;
    }
  printf("ibcntl= %d\n\n", ibcntl);
}

/* gpinit - open gpib board and initialize
 */
static int gpinit(char *bdname)
 {
  int   bd;

  if ((bd = ibfind(bdname)) < 0)          /* open board */
     {
      prterr (“ibfind error”):
      return -1;
   }

  if (ibsic(bd) & ERR)                    /* interface clear */
     {
     prterr(“ibsic error”);
     ibonl(bd, 0);
     return -1;
   }
  if (ibsre(bd, 1) & ERR)                  /* remote enable */
     {
     prterr(“ibsre error”);
     ibonl(bd, 0);
     return -1;
   }
 return bd ;                               /* return descriptor */
}

/* nainit - open N.A port and initialize
 */
static int nainit(char *dvname)
 {
  int   dv;

  if ((dv = ibfind(“DEVII”)) < 0)          /* open N.A */
     {
     prterr(“ibfind error”);
     return -1;
   }

  ibwrt (dv,“OLDC OFF”,9);              /* default connand */
  if (ibsta & ERR)
     {
      prterr("ibwrt error");
      ibonl(dv, 0);
      return -1;
   }
  return dv;
}                                       /* return descriptor */

/* tisetup - setups
 */
static int tisetup (int dv)
 {
  ibwrt(dv, "SWE:POIN 201", 13);
  ibwrt(dv, "FORM:DATA ASC", 14);
  return 0;
}

/* tireceive - receives trace data
 */
static int tireceive(int bd, double *buf, unsigned bufsiz, char *name)
 {
  unsigned int len, cnt = 0;
  char  s[32], n[20];

  *n = 0;
  strcat(strcpy(n, "TRAC?"), name);
  ibwrt(bd, n, strlen(n));              /* query */
  ibeos(bd, 0x427);
  ibcmd(bd, "?_ K", 4);                 /* UNL UNT MLA 0 TAD 11 */

  while (cnt < bufsiz)
     {
      ibrd(bd, s, 23);                /* [S#.###################,] */
      s[ibcntl] = 0;
      *buf++ = atof(s);
```

```
        cnt ++ ;
         if (ibsta & END) break ;        /* with EOI */
      }

    ibeos(bd, 0x40A);
    ibcmd(bd, “?_+@”, 4);                /* UNL UNT MLA 11 TAD 0 */
    return cnt;
  }

  /* tiprint - print trace data
   */
  static int tiprint(double *data1, double *data2, unsigned num)
   {
    unsigned i;

    for (i = 0; i < num; ++i)
       {
        printf(“ %4d: %1.7e\t%1.7e\n”, i, *data1, *data2);
       data1 ++ ;
       data2 ++ ;
      }
  }

  / * main entry
   */
  main(int argc, char **argv)
   {
    int   bd, na;
    int   num;

    if ((bd = gpinit(“GPIB0”)) == -1)
      exit(1);
    if ((na = nainit(“DEV11”)) == -1)
       {
        ibonl(bd, 0);
        exit(1);
    }

  tisetup(na);
  num = tireceive(bd, databuf1, 201, "FDAT1");
  num = tireceive(bd, databuf2, 201, "FDAT2");
  tiprint(&databuf1[0], &databuf2[0], num);

  ibonl(na, 0);
  ibonl(bd, 0);
}
```

Example 12-25 Data Input of C (Binary format)

```
/*
 *        INPUT TRACE DATA FROM NA
 *            (BINARY FORMAT)
 *
 * TARGET:    PC/AT(NI-488.2)
 * -LANGUAGE: C (ANSI-C STYLE)
 * FILE:      MCTINPB.C
 */
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include "decl.h"

static int gpinit(char *bdname);
static int nainit(char *dvname);
static int tisetup(int dv);
static int tireceive(int bd, double *buf, unsigned bufsiz);
static int tiprint(double * data, unsigned points);
static void prterr(char *msg);

double databuf[402];                      /* data buffer */

/* main entry
```

```
 */
main(int argc, char **argv)
 {
  int   bd, na;
  int   num;

  if ((bd = gpinit(“GPIBO”)) == -1)
    exit(1);
  if ((na = nainit(“DEV11”)) == -1)
     {
      ibonl(bd, 0);
      exit(1);
   }

  tisetup(na);
  num = tireceive(bd, &databuf[0], 402);
  tiprint(&databuf[0], num);

  ibonl(na, 0);
  ibonl(bd, 0);
}

/* gpinit - open gpib board and initialize
 */
static int gpinit(char *bdname
{
int   bd;

if ((bd = ibfind(bdname)) < 0)                /* open board */
   {
   prterr(“ibfind error”);
   return -1;
}

  if (ibsic(bd) & ERR)                      /* interface clear */
     {
      prterr("ibsic error");
      ibonl(bd, 0);
      return -1;
   }
  if (ibsre(bd, 1) & ERR)                   /* remote enable */
     {
      prterr("ibsre error");
      ibonl(bd, 0);
      return -1;
   }
  return bd;                                /* return descriptor */
}

/* nainit - open N.A port and initialize
 */
static int nainit(char *dvname)
 {
  in  dv;

  if ((dv = ibfind("DEV11")) < 0)           /* open N.A */
     {
      prterr("ibfind error");
      return -1;
   }
  ibwrt(dv, "OLDC OFF", 9);                 /* default command */
  if (ibsta & ERR)
     {
      prterr("ibwrt error");
      ibonl(dv, 0);
      return -1;
   }
  return  dv ;
}                                           /* return descriptor */

/* tisetup - setups
 */
```

```
static int tisetup(int dv)
 {
  ibwrt(dv, "SWE:POIN 201", 13);
  ibwrt(dv, "FORM:BORD SWAP;DATA REAL,64", 28);
  return 0;
}

/* tireceive - receives trace data
 */
static int tireceive(int bd, double *buf, unsigned bufsiz)
 {
  unsigned cnt;
  char  s[32];

  ibwrt(bd, "TRAC:DATA? DATA", 16);        /* query */

  ibconfig(bd, 12, 0);                     /* disable EOS detection */
  ibwrt(bd, "?_ K", 5);                    /* UNL UNT MLA 0 TAD 11 */

  ibrd(bd, s, 8);                          /* read header */
  ibrd(bd, (char *)buf, sizeof(double)*bufsiz);
  cnt = ibcnt;
  if (!(ibsta&END)) ibrd(bd, s, 1);        /* read terminator */

  ibconfig(bd, 12, '\n');                  /* enable EOS detection */
  return cnt/sizeof(double);
}

/* tiprint - print trace data
 */
static int tiprint(double *data, unsigned num)
 {
  unsigned points = num>>1;
  unsigned i;

  for (i=0; i<points; ++i)
```

```
     {
      printf ( %4d: %1.7e\t%1.7e\n", i, *data, *(data+1));
      data += 2;
    }
}

/* prterr - print gpib error message and status code
 */
static void prterr(char *msg)
 {
  printf("%s\n", msg);
  printf("ibsta=&H%x < ", ibsta);

  if (ibsta & ERR)     printf("ERR");
  if (ibsta & TIMO)    printf("TIMO");
  /*if (ibsta & EEND) printf("EEND");*/
  if (ibsta & SRQI)    printf("SRQI");
  if (ibsta & RQS)     printf("RQS");
  if (ibsta & CMPL)    printf("CMPL");
  if (ibsta & LOK)     printf("LOK");
  /*if (ibsta & RREM) printf("RREM");*/
  if (ibsta & CIC)     printf("CIC");
  /*if (ibsta & AATN) printf("AATN");*/
  if (ibsta & TACS)    printf("TACS");
  if (ibsta & LACS)    printf("LACS");
  if (ibsta & DTAS)    printf("DTAS");
  if (ibsta & DCAS)    printf("DCAS");
  printf(" >\n");

  printf("iberr= %d", iberr);
  switch (iberr)
     {
    case EDVR: printf("EDVR <DOS Error>"); break;
    case ECIC: printf("ECIC <Not CIC>"); break ;
    case ENOL:  printf("ENOL <No listner>"; break;
    case EADR:  printf("EADR <Address error>"); break;
```

```
        case EARG:  printf("EARG <Invalid argment>"); break;
        case ESAC:  printf("ESAC <Not Sys Ctrlr>"); bread;
        case EABO:  printf("EABO <Op. aborted>"); break;
        case ENEB:  printf("ENEB <No GPIB board>"); break;
        case EOIP:  printf("EOIP <Async I/O in prg>"); break;
        case ECAP:  printf("ECAP <No capability>"); break;
        case EFSO:  printf("EFSO <Fils sys. error>"); break;
        case EBUS:  printf("EBUS <Command error>"); break;
        case ESTB:  printf("ESTB <Status byte lost>"); break;
        case ESRQ:  printf("ESRQ <SRQ stuck on>"); break;
        case ETAB:  printf("ETAB <Table Overflow>"); break;
        }
      printf("ibcnt=  %d\n\n", ibcntl);
    }
```

12.6 Using Built-in BASIC and External Controller Simultaneously

This section describes a method in which the measurement is performed by executing the BASIC program of this instrument from an external controller, then the measurement data is received and displayed with the external controller.
Besides , a measurement program is created with built-in BASIC of this instrument side and a program example which performs a filter analysis with the built-in functions is presented.
Then, a program which receives the measurement data by the use of SRQ (service request) is created with the program of external controller side. The program of external controller is different according to the employed computer and language. In the first place, N88-BASIC programs of NEC PC-9801 is described.

The following shows the outline of programs described in this section.

External controller side		Built-in BASIC side
(1) Initialize interface.		
(2) Execute the program of this instrument after loading.	→	(1) Set the measurement condition of this instrument. (2) Pausing
(3) CONT the program of this instrument.	→	(3) Start the scan and waits till the scan is ended with WAIT EVENT.
(4) Loop waiting till the SRQ comes.		(4) Analyze the measurement data.
(5) Perform serial poll when SRQ comes.	←	(5) Output SRQ with REQUEST command.
(6) Receive data and displays it on the screen.	←	(6) Send data.
(7) Repeat from (3).		(7) Repeat from (2).

When the data sending-receiving is performed between external controller and built-in BASIC of this instrument, the exchanging written above becomes necessary.

12.6.1 Sending Program of Built-in BASIC

A program that performs the filter analysis and sends the analysis data using the built-in BASIC of this instrument is shown as follows.

Example 12-26 Sending Program of Built-in BASIC

```
1000 !*********************************************
1010 !*                                           *
1020 !*           DATA TRANSFER PROGRAM           *
1030 !*                                           *
1040 !* TARGET: NETWORK ANALYZER (to PC-9801)     *
1050 !* FILE:   NSEND.BAS                         *
1060 !*********************************************
1070 INTEGER EV
1080 DIM L(2),F(2,4)
1090 !
1100 *MAIN
1110     GOSUB *SETUP
1120     CLS
1130     *MEAS_LOOP
1140         CURSOR 0,0
1150         PAUSE
1160         GOSUB *MEAS
1170         GOSUB *SEND
1180         GOTO *MEAS_LOOP
1190 !
1200 *SETUP
1210     NA=31 :PC=11 :EV=1 :L(1)=3.0 :L(2)=60.0
1220     OUTPUT NA;"OLDC OFF"
1230     OUTPUT NA;"SYST:PRES;:INIT:CONT OFF;:STAT:OPER:ENAB 8;*SRE
                                                          128;*OPC?"
1240     ENTER  NA;A
1250     OUTPUT NA;"FREQ:SPAN 20MAHZ;CENT 12MAHZ"
1260     RETURN
1270 !
1280 *MEAS
1290     SPOLL(NA)
```

```
1300      OUTPUT NA;"INIT":WAIT EVENT EV
1310      AP=PMAX(0,1200,0)
1320      NP=MBNDI(0,1200,AP,2,L(1),F(1,1),0)
1330      QF=F(1,3)/F(1,4)                  ! QF = CF(3dB)     / BW(3dB)
1340      SF=F(2,4)/F(1,4)                  ! SF = BW'(60dB)   / BW(3dB)
1350      RETURN
1360 !
1370 *SEND
1380      REQUEST 65
1390      OUTPUT PC;F(1,1) :OUTPUT PC;F(1,2) :OUTPUT PC;F(1,3) :
                                                   OUTPUT PC;F(1,4)
1400      OUTPUT PC;QF     :OUTPUT PC;SF
1410      RETURN
```

Input this program and store it in a floppy disk.
The file is named as " NSEND.PGM " when it is stored.
The file name is referred to when it is loaded from external controller.

In this program, the pause is performed automatically after the initialization and the measurement condition of this instrument is set.
After @ CONT is sent from the external controller, the measurement is performed , then the measurement data is analyzed and sent to the external controller.
After this series of processing is performed, the pause is performed again.

12.6.2 Receiving Program of N88-BASIC

The receiving program of PC-9801 is shown as follows.

Example 12-27 Receiving Program of N88-BASIC

```
1000  ' **********************************************
1010  ' *                                            *
1020  ' *          CONTROL AND RECEIVE DATA          *
1030  ' *                                            *
1040  ' *  TARGET: PC-9801                           *
1050  ' *  FILE:   NRECEIVE.BAS                      *
1060  ' **********************************************
1070 ISET IFC
1080 ISET REN
1090 NA=11
1100 POLL NA,P
1110 ON SRQ GOSUB *SRINT
1120
1130 A$=“A:/NSEND.BAS”
1140 M$=CHR$(34)+A$+CHR$(34)
1150 L$=“@LOAD”+M$
1160 PRINT @NA;“@SCRATCH”
1170 PRINT @NA;L$
1180 PRINT @NA;“@RUN”
1190 '
1200 CLS
1210 *MEAS.LOOP
1220     GOSUB *MEAS.CONT
1230     GOSUB *RECEIVE
1240     GOTO *MEAS.LOOP
1250 '
1260 *MEAS.CONT
1270     LOCATE 6,9  :PRINT “CONNECT DUT”
1280     LOCATE 6,10 :INPUT “IF OK THEN PRESS ANY KEY”,D$
1290     PRINT @NA ;“@CONT”
1300     URQ=0
```

```
1310      SRQ ON
1320      RETURN
1330 '
1340 *RECEIVE
1350      IF URQ=0 THEN GOTO *RECEIVE
1360      INPUT @NA;LF:INPUT @NA;RF:INPUT @NA;CF:INPUT @NA;BW
1370      INPUT @NA;QF:INPUT @NA;SF
1380      LOCATE 5,1:PRINT USING "C.F = ####.###### [MHz]";CF/10^6
1390      LOCATE 5,2:PRINT USING "L.F = ####.###### [MHz]";LF/10^6
1400      LOCATE 5,3:PRINT USING "R.F = ####.###### [MHz]";RF/10^6
1410      LOCATE 5,4:PRINT USING "BW  = ####.###### [MHz]";BW/10^6
1420      LOCATE 5,5:PRINT USING "QF  = ####.###### ";QF
1430      LOCATE 5,6:PRINT USING "SF  = ####.###### ";SF
1440      RETURN
1450 '
1460 *SRINT
1470      POLL NA,P
1480      P = P AND 1
1490      IF P<>0 THEN URQ=1
1500      RETURN
```

This program is input with BASIC mode of PC-9801.
The program is performed in the following sequence.

Execution sequence:

① Insert the floppy disk in which the " Example 12-26 " is stored into the drive.

② Execute the program of PC-9801 side.
Input RUN and press Return key.

When the program is executed on PC-9801 side, the program of this instrument side is loaded from the floppy disk and executed automatically.
When it is executed, according to the program , this instrument is set and the measurement is started.
When the measurement is ended and the result is produced, it is sent to PC-9801 side.

The result is as follows.

Execution result:

```
C.F =  146.716000 [MHz]
L.F =  137.156000 [MHz]
R.F =  157.699000 [MHz]
BW  =   21.964200 [MHz]
```

```
QF  =    6.679790
SF  =    0.000000
```

When PC-9801 matches the execution result as shown here, it is recognized that the data is sent and received certainly.
In R3752 series, the mark function can be substituted by creating programs like this.

12.6.3 Receiving Program of HP-BASIC

The example of receiving program that used HP-BASIC is shown as follows.

Example 12-28 Receiving Program of HP-BASIC

```
1000 !  *****************************************
1010 !  *                                       *
1020 !  *        CONTROL AND RECEIVE DATA       *
1030 !  *                                       *
1040 !  *  TARGET: HP-BASIC                     *
1050 !  *  FILE:   HPREC.BAS                    *
1060 !  *****************************************
1070 !
1080 DIM L$[24]
1090 !
1100 ASSIGN @Na TO 711
1110 ON INTR 7 GOSUB Srint
1120 !
1130 A$="A:/NSEND.BAS"
1140 M$=CHR$(34)+A$+CHR$(34)
1150 L$="@LOAD"+M$
1160 !
1170 OUTPUT @Na;"@SCRATCH"
1180 OUTPUT @Na;L$
1190 OUTPUT @Na;"@RUN"
1200 !
1210 Meas_loop:!
1220     GOSUB Meas_cont
1230     GOSUB  Receive
1240     GOTO Meas_loop
1250 !
1260 Meas_cont:!
```

```
1270      PRINT "CONNECT DUT"
1280      INPUT "IF OK THEN PRESS ANY KEY",D$
1290      OUTPUT @Na;"@CONT"
1300      Urq=0
1310      ENABLE INTR 7;255
1320      RETURN
1330 !
1340 Receive:!
1350      IF Urq=0 THEN GOTO *Receive
1360      DISABLE INTR 7
1370      ENTER @Na;Lf
1380      ENTER @Na;Rf
1390      ENTER @Na;Cf
1400      ENTER @Na;Bw
1410      ENTER @Na;Qf
1420      ENTER @Na;Sf
1430      PRINT "C.F [MHz] = ";
1440      PRINT USING "DDDD.DDDDDD";Cf/10^6
1450      PRINT "L.F [MHz] = ";
1460      PRINT USING "DDDD.DDDDDD";Lf/10^6
1470      PRINT "R.F [MHz] = ";
1480      PRINT USING "DDDD.DDDDDD";Rf/10^6
1490      PRINT "BW  [MHz] = ";
1500      PRINT USING "DDDD.DDDDDD";Bw/10^6
1510      PRINT "Q         = ";
1520      PRINT USING "DDDD.DDDDDD";Qf
1530      PRINT "SF        = ";
1540      PRINT USING "DDDD.DDDDDD";Sf
1550      RETURN
1560 !
1570 Srint:!
1580      Stat = SPOLL(@Na) AND 1
1590      IF Stat<>0 THEN Urq = 1
1600      RETURN
1610 END
```

12.6.4 Receiving Program of QuickBASIC

The example of receiving program that used QuickBASIC is shown as follows.

Example 12-29 Receiving Program of QuickBASIC

```
'  ****************************************
'  *                                      *
'  *        CONTROL AND RECEIVE DATA      *
'  *                                      *
'  *  TARGET:   PC/AT(NI-488.2)           *
'  *  LANGUAGE: QuickBASIC                *
'  *  FILE:     QBREC.BAS                 *
'  ****************************************

REM $INCLUDE:  'qbdecl.bas'

DECLARE SUB gpinit (bdname$, bd%)
DECLARE SUB nainit (bd%, naname$, dv%)
DECLARE SUB nasetup (dv%)
DECLARE SUB nacont (dv%)
DECLARE SUB nareceive (bd%)
DECLARE SUB prterr (msg $)

CALL gpinit("GPIB0", bd%)
CALL nainit(bd%, "DEV11", na%)
CALL nasetup(na%)

Measloop:
        CALL nacont(na%)
        CALL nareceive(na%)
        GOTO Measloop

CALL ibonl(na%, 0)
CALL ibonl(dv%, 0)
END
```

```
'    This routine open the gpib board and initialize
'
SUB gpinit (bdname$, bd%) STATIC

        CALL ibfind(bdname$, bd%)                  ' OPEN BOARD
        IF (bd% < 0) THEN
                CALL prterr("ibfind error")
                STOP
        END IF

        CALL ibsic(bd%)              ' INTERFACE CLEAR
        IF (ibsta% AND EERR) THEN
                CALL prterr ("ibsic error")
                CALL ibonl (bd%, 0)
                STOP
        END IF

        CALL ibsre(bd%, 1)                ' REMOTE ENABLE
        IF (ibsta% AND EERR) THEN
                CALL prterr ("ibsic error")
                CALL ibonl (bd%, 0)
                STOP
        END  IF

END SUB

'    This routine continue the N.A  BASIC  PROGRAM
'
SUB nacont (dv%) STATIC

         PRINT "CONNECT DUT"
         INPUT "IF OK THEN PRESS ANY KEY", key$
         cmd$ = "@CONT"
         CALL ibwrt(dv%, cmd $)

END SUB
```

```
'     This routine open N.A and initialize
'
SUB nainit (bd%, dvname$, dv%) STATIC

        CALL ibfind(dvname$, dv%)
        IF (dV% < 0) THEN
                CALL prterr("ibfind error")
                CALL ibonl(bd%, 0)
                STOP
        END IF

        cmd$ = "OLDC OFF"
        CALL ibwrt(dv%, cmd$)
        IF (ibsta% AND EERR) THEN
                CALL prterr ("ibwrt error")
                CALL ibonl(dv%, 0)
                CALL ibonl(bd%, 0)
                STOP
        END IF

END SUB

'     This routine receives data and print
'
SUB nareceive (dv%) STATIC

        mask% = &H4800
Nawait:
               CALL ibwait(dv%, mask%)
               CALL ibrsp(dv%, spr%)
               spr% = spr% AND 1
               IF (spr% = 0) GOTO Nawait

       str1$ = SPACE$(23)
       str2$ = SPACE$(23)
       str3$ = SPACE$(23)
       str4$ = SPACE$(23)
       str5$ = SPACE$(23)
       str6$ = SPACE$(23)

       CALL ibrd(dv%, str1$)
       CALL ibrd(dv%, str2$)
       CALL ibrd(dv%, str3$)
       CALL ibrd(dv%, str4$)
       CALL ibrd(dv%, str5$)
       CALL ibrd(dv%, str6$)

       PRINT USING "C.F = ####.###### [MHz]"; VAL(str3$) / 10^6
       PRINT USING "L.F = ####.###### [MHz]"; VAL(str1$) / 10^6
       PRINT USING "R.F = ####.###### [MHz]"; VAL(str2$) / 10^6
       PRINT USING "BW  = ####.###### [MHz]"; VAL(str4$) / 10^6
       PRINT USING "QF  = ####.######"; VAL(str5$)
       PRINT USING "SF  = ####.######"; VAL(str6$)

END SUB

'    This routine setups
'
SUB nasetup (dv%) STATIC

        cmd% ="@STOP"
        CALL ibwrt(dv%, cmd$)
        CALL ibwrt(dv%, cmd$)
        cmd$ = "@SCRATCH"
        CALL ibwrt(dv%, cmd$)
        cmd$ = "@LOAD" + CHR$(34) + "A:/NSEND.BAS" + CHR$(34)
        CALL ibwrt(dv%, cmd$)
        cmd% = "@RUN"
        CALL ibwrt(dv%, cmd$)

END SUB
```

```
'   This routine prints the result of status variables.
'
SUB prterr (msg$) STATIC

        PRINT msg$
        PRINT "ibsta=&H"; HEX$(ibsta%); " <";
        IF ibsta% AND EERR THEN PRINT " ERR";
        IF ibsta% AND TIMO THEN PRINT " TIMO";
        IF ibsta% AND EEND THEN PRINT " EEND";
        IF ibsta% AND SRQI THEN PRINT " SRQI";
        IF ibsta% AND RQS  THEN PRINT " RQS";
        IF ibsta% AND CMPL THEN PRINT " CMPL";
        IF ibsta% AND LOK  THEN PRINT " LOK";
        IF ibsta% AND RREM THEN PRINT " RREM";
        IF ibsta% AND CIC  THEN PRINT " CIC";
        IF ibsta% AND AATN THEN PRINT " AATN";
        IF ibsta% AND TACS THEN PRINT " TACS";
        IF ibsta% AND LACS THEN PRINT " LACS";
        IF ibsta% AND DTAS THEN PRINT " DTAS";
        IF ibsta% AND DCAS THEN PRINT " DCAS";
        PRINT " >"

        PRINT "iberr="; iberr%;
        IF iberr% = EDVR THEN PRINT " EDVR <DOS Error>"
        IF iberr% = ECIC THEN PRINT " ECIC <Not CIC>"
        IF iberr% = ENOL THEN PRINT " ENOL <No listner>"
        IF iberr% = EADR THEN PRINT " EADR <Address error>"
        IF iberr% = EARG THEN PRINT " EARG <Invalid argument>"
        IF iberr% = ESAC THEN PRINT " ESAC <Not Sys Ctrlr>"
        IF iberr% = EABO THEN PRINT " EABO <Op. aborted>"
        IF iberr% = ENEB THEN PRINT " ENEB <No GPIB board>"
        IF iberr% = EOIP THEN PRINT " EOIP <Async I/O in prg>"
        IF iberr% = ECAP THEN PRINT " ECAP <No capability>"
        IF iberr% = EFSO THEN PRINT " EFSO <File sys. error>"
        IF iberr% = EBUS THEN PRINT " EBUS <Command error>"
        IF iberr% = ESTB THEN PRINT " ESTB <Status byte lost>"
        IF iberr% = ESRQ THEN PRINT " ESRQ <SRQ stuck on>"
        IF iberr% = ETAB THEN PRINT " ETAB <Table Overflow>"
        PRINT " ibcnt = "  ;   ibcnt%

END SUB
```

12.6.5 Receiving Program of ANSI-C

The example of receiving program that used C is shown as follows.

Example 12-30 Receiving Program of C

```
/*
 *      CONTROL AND RECEIVE DATA
 *
 * TARGET:    PC/AT(NI-488.2)
 * LANGUAGE: C (ANSI-C STYLE)
 * FILE:     MCREC.C
 */
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include "decl.h"

/* prterr - print gpib error message and status code
 */
static void prterr(char *msg)
 {
  printf("%s\n", msg);
  printf("ibsta=&H%x <", ibsta);
  if (ibsta & ERR)  printf("ERR");
  if (ibsta & TIMO) printf("TIMO");
  if (ibsta & SRQI) printf("SRQI");
  if (ibsta & RQS)  printf("RQS");
  if (ibsta & CMPL) printf("CMPL");
  if (ibsta & LOK)  printf("LOK");
  if (ibsta & CIC)  printf("CIC");
  if (ibsta & TACS) printf("TACS");
```

```
      if (ibsta & LACS) printf(“LACS”);
      if (ibsta & DTAS) printf(“DTAS”);
      if (ibsta & DCAS) printf(“DCAS”);
      printf(“ >\n”);

      printf(“iberr= %d”, iberr);
      switch(iberr)
         {
        case EDVR: printf(“EDVR <DOS Error>”); break;
        case ECIC: printf(“ECIC <Not CIC>”); break;
        case ENOL: printf(“ENOL <No listner>”); break;
        case EADR: printf(“EADR <Address error>”); break;
        case EARG: printf(“EARG <Invalid argument>”); break;
        case ESAC: printf(“ESAC <Not Sys Ctrlr>”); break;
        case EABO: printf(“EABO <Op.aborted>”); break;
        case ENEB: printf(“ENEB <No GPIB board>”); break;
        case EOIP: printf(“EOIP <Async I/O in prg>”); break;
        case ECAP: printf(“ECAP <No capability>”); break;
        case EFSO: printf(“EFSO <Fils sys. error>”); break;
        case EBUS: printf(“EBUS <Command error>”); break;
        case ESTB: printf(“ESTB <Status byte lost>”); break;
        case ESRQ: printf(“ESRQ <SRQ stuck on>”); break;
        case ETAB: printf(“ETAB <Table Overflow>”); break;
        }
      printf ( “ibcntl= %d\n\n”, ibcntl );
    }

    /* gpinit - open gpib board and initialize
     */
    static int gpinit(char *bdname)
     {
      int   bd;

      if ((bd = ibfind(bdname)) < 0)                  /* open board */
         {
         prterr(“ibfind error”);
      return -1;
    }

  if (ibsic(bd) & ERR)                          /* interface clear */
      {
      prterr("ibsic error");
      ibonl(bd, 0);
      return -1;
    }

  if (ibsre(bd, 1) & ERR)                       /* remote enable */
      {
      prterr("ibsre error");
      ibonl(bd, 0);
      return -1;
    }
  return bd;                                    /* return descriptor */
}

/* nainit - open N.A Port and initialize */
 */
static int nainit(char *dvname)
 {
  int   dv;
  if ((dv = ibfind("DEV11")) < 0)               /* open R3752/R3753 */
      {
      prterr("ibfind error");
      return -1;
    }

  ibwrt (dv, "OLDC OFF", 8);                    /* default command */
  if (ibsta & ERR)
      {
      prterr("ibwrt error");
      ibonl(dv, 0);
      return -1;
      }
    return dv;
  }                                                   /* return descriptor */

  /* nasetup - setups
   */
  static int nasetup(int dv)
   {
    ibwrt(dv, "@STOP", 5);
    ibwrt(dv, "@STOP", 5);
    ibwrt(dv, "@SCRATCH", 8);
    ibwrt(dv, "@LOAD \"A:/NSEND.BAS\" ", 20);
    ibwrt(dv, "@RUN", 4);
    return 0 ;
  }

  /* nacont - continue INTERNAL BASIC
   */
  static int nacont(int dv)
   {
    int    f;
    char   c;

    printf("CONNECT DUT\n")
    printf("IF OK THEN PRESS KEY [y/n]");
    fflush(stdout);

    while (1)
       {
        c = getchar();
        if (c == EOF)
           {
            f = -1;
            break;
          }
       if (c == 'y' || c == 'Y')
```

```
        {
         f = 0;
         break;
       }
     if (c == ‘n’ || c == ‘N’)
        {
         f = -1;
         break;
      }
}
  fflush(stdin);
  ibwrt(dv, “@CONT”, 5);
  return f;
}

/* nareceive - receives trace data
 */
static int nareceive(int na, int bd)
 {
  char  buf[6][24];
  int   i;
  char  s;

  while (1)
     {
      ibwait(na, (int)0x4800);
      ibrsp(na, &s);
      if (s & 1) break;                 /* check user request bit */
    }

  ibcmd(bd, “?_K”, 4);                  /* UNL UNT MLA 0 TAD 11 */
  for (i = 0; i < 6; i++)
     {
      ibrd(bd, &buf[i][0], 23);
      buf[i] [ibcntl] = ‘\0’;
   }
  ibcmd(bd, “?_+@”, 4);                   /* UNL UNT MLA 11 TAD 0 */

  printf(“C.F = %4.6f [MHz]\n”, atof(&buf[2][0])/1.0e6);
  printf(“L.F = %4.6f [MHz]\n”, atof(&buf[0][0])/1.0e6);
  printf(“R.F = %4.6f [MHz]\n”, atof(&buf[1][0])/1.0e6);
  printf(“BW  = %4.6f [MHz]\n”, atof(&buf[3][0])/1.0e6);
  printf(“QF  = %4.6f\n”, atof(&buf[4][0]));
  printf(“SF  = %4.6f\n”, atof(&buf[5][0]));

  return 6;
}

/* main entry
 */
main(int argc, char **argv)
 {
  int     bd, na;
  int     num;

  if ((bd = gpinit(“GPIB0”)) == -1)
    exit(1);
  if ((na = nainit(“DEV11”)) == -1)
     {
      ibonl(bd, 0);
      exit(1);
   }

  nasetup(na);
  while (1)
     {
      if (nacont(na) == -1) break;
      nareceive (na, bd);
   }
  ibonl(na, 0);
  ibonl(bd, 0);
}
```

12.7 Download of BASIC Program

This section describes a method with which the program used for this instrument is down loaded from external controller to this instrument side and then executed.
On the contrary, next section " 12.8 " will describe a method with which the programs is up loaded from this instrument.
First of all, the down-load and up-load are simply described.

- Download

  An external controller is used to create the program file and save it to the floppy disk in advance. The download is that transferring the program to the memory of this instrument side via GPIB.

- Upload

  Contrary to the download, the upload is that transferring the program existing in the memory of this instrument side to the memory of external controller via GPIB.

In the case of download, when the program file used for built-in BASIC is managed with an external controller, the program can be loaded without using floppy disk.
That is, the built-in BASIC can be controlled as the same as other GPIB commands.

The program creates down loads the program to be executed with built-in BASIC, then executes it after exchanging it with the program to be executed on the external controller.
The two programs in Example 12-26 and Example 12-27 can be used.

Outline of program:

(1) Initialize external controller.

(2) Open the program file to be down-loaded and transfer the content to this instrument.

(3) After download, load the program to be executed with the external controller and execute it.

12.7.1 Download Program of N88-BASIC

The download program to be executed with PC-9801 is shown as follows.

Example 12-31 Download Program of N88-BASIC

```
1000 ' ****************************************
1010 ' *                                      *
1020 ' *           DOWN LOAD PROGRAM          *
1030 ' *                                      *
1040 ' * TARGET: PC-9801                      *
1050 ' * FILE:   NDOWNLD.BAS                  *
1060 ' ****************************************
1070 NA=11
1080 ISET IFC
1090 ISET REN
1100 CMD DELIM=2
1110 CMD TIMEOUT=3
1120 '
1130 ON ERROR GOTO *ERRORMES
1140 RINT "PROGRAM TRANSFER (PC to NA)"
1150 '
1160 *DNLD.ENTER
1170     FLAG=0
1180     INPUT "ENTER DOWNLOAD PROGRAM";F$
1190     OPEN F$ FOR INPUT AS #1
1200     IF FLAG=1 THEN *DNLD.ENTER
1210     PRINT @NA;"@SCRATCH" @
1220 '
1230 *DNLD.LOOP
1240     LINE INPUT #1,DB$
1250     DB$="@"+DB$
1260     PRINT DB$
1270     PRINT @NA;DB$ @
1280     IF EOF(1) THEN *DNLD.EXIT ELSE *DNLD.LOOP
1290 '
1300 *DNLD.EXIT
```

```
1310      CLOSE
1320      PRINT "COMPLETE DOWNLOAD"
1330 '
1340 *EXEC.ENTER
1350      FLAG=0
1360      INPUT "ENTER STARTING PROGRAM";F$
1370      IF F$=" " THEN END
1380      OPEN F$ FOR INPUT AS #1
1390      IF FLAG=1 THEN *EXEC.ENTER
1400      CLOSE
1410      ON ERROR GOTO 0
1420      RUN F$
1430      END
1440 '
1450 *ERRORMES
1460      FLAG=1
1470      PRINT "ERROR: LINE=";ERL;" NO.=";ERR
1480      INPUT "RETRY? (Y/N)";A$
1490      IF A$="Y" OR A$="y" THEN RESUME NEXT
1500      ON ERROR GOTO 0
1510      END
```

Input this program under the BASIC mode of PC-9801.
Save it to a floppy disk after input.

The program execution is performed in the following sequence.

Execution sequence:

① Create a download program , then save it to a floppy disk.
Here, Example 12-26 is used.

② Create a program to be controlled with external controller, then save it to a floppy disk.
Here, a part of Example 12-27 is used after correcting it .
Turn line 1190 to 1190 ! PRINT NA ; L $ command line.

③ Execute download program.
After ENTER DOWNLOAD PROGRAM is displayed. it turns to waiting input state.

④ Here, Input the file name of program to be down-loaded. (For instance, NSEND. BAS etc.)
After input the file name, press Return key, then the file is loaded and the download starts.
The download is performed by each line of the program , and the line being transferred is displayed on the screen so as to be checked.
When the download is ended, COMPLETE DOWNLOAD is displayed on the screen.

⑤ Next, as ENTER STARTING PROGRAM is displayed , input the file name of control program to be executed with external controller.
The entered program is exchanged with the present download program and executed.
When only download is performed, press Return key only .
The execution result is the same as the result of Example 12-27.

12.7.2 Download Program of HP-BASIC

The download program to be executed with HP-BASIC is shown as follows.

Example 12-32 Download Program of HP-BASIC

```
1000 ! ****************************************
1010 ! *                                      *
1020 ! *          DOWN  LOAD  PROGRAM         *
1030 ! *                                      *
1040 ! * TARGET: HP-BASIC                     *
1050 ! * FILE:   HPDNLD.BAS                   *
1060 ! ****************************************
1070 !
1080 ASSIGN @Na TO 711
1090 DIM Line$[512]
1100 !
1110 PRINT "PROGRAM TRANSFER (HP-9000 TO NA)"
1120 INPUT "ENTER DOWNLOAD PROGRAM";Name$
1130 OUTPUT @Na;"*RST"
1140 OUTPUT @Na;"@SCRATCH"
1150 !
1160 ON ERROR GOTO Done
1170 ASSIGN @File TO Name$
1180 !
1190 LOOP
1200     Line$=" "
1210     ENTER @File;Line$            ! READ ONE LINE
```

```
1220      OUTPUT @Na;"@"+Line$          ! TRANSFER ONE LINE
1230 END LOOP
1240 !
1250 Done:!                             ! END OF FILE
1260      OFF ERROR
1270      ASSIGN @File TO *             ! CLOSE FILE
1280 END
1290
1300
```

12.7.3 Download Program in QuickBASIC

The download program to be executed with QuickBASIC is shown as follows.

Example 12-33 Download Program Used for QuickBASIC

```
' ******************************************
' *                                        *
' *            DOWN LOAD PROGRAM           *
' *                                        *
' *. TARGET:   PC/AT(NI-488.2)             *
' * LANGUAGE: QuickBASIC                   *
' * FILE:     QBDNLD. BAS                  *
' ******************************************

REM $INCLUDE: 'qbdecl.bas'

DECLARE SUB gpinit (bdname$, bd%)
DECLARE SUB nainit (bd%, naname$, dv%)
DECLARE SUB prterr (msg$)

DIM LineBuffer$(512)
DIM cmd$(512)

PRINT "PROGRAM TRANSFER (PC/AT TO NA)"
INPUT "ENTER DOWNLOAD PROGRAM"; Name$
CALL  gpinit("GPIB0", bd%)
CALL  nainit(bd%, "DEV11", na%)
```

```
OPEN  Name$ FOR INPUT AS #1

DO UNTIL EOF(1)
        LINE INPUT #1, LineBuffer$        ' READ ONE LINE
        PRINT LineBuffer$                 ' PRINT TO DISPLAY
        cmd$ = "@" + LineBuffer$
        CALL ibwrt(na%, cmd$)             ' TRANFER PROGRAM TO NA
LOOP

CLOSE #1
CALL ibonl(na%, 0)
CALL ibonl(dv%, 0)
END

'    This routine open the gpib board and initialize
'
SUB gpinit (bdname$, bd%) STATIC

        CALL ibfind(bdname$, bd%)        ' OPEN BOARD
        IF (bd% < 0) THEN
                CALL prterr("ibfind error")
                STOP
        END IF

        CALL ibsic(bd%)                   ' INTERFACE CLEAR
        IF (ibsta% AND EERR) THEN
                CALL prterr("ibsic error")
                CALL ibonl(bd%, 0)
                STOP
        END IF

        CALL ibsre(bd%, 1)             ' REMOTE ENABLE
        IF (ibsta% AND EERR) THEN
                 CALL prterr("ibsre error")
                 CALL ibonl(bd%, 0)
                 STOP
        END IF

END SUB

'    This routine open N.A and initialize
'
SUB nainit (bd%, dvname$, dv%) STATIC

        CALL ibfind(dvname$, dv%)
        IF (dv% < 0) THEN
                CALL prterr(“ibfind error”)
                CALL ibonl(bd%, 0)
                STOP
        END IF

        cmd$ = “OLDC OFF;*RST”
        CALL ibwrt(dv%, cmd$)
        IF (ibsta% AND EERR) THEN
                 CALL prterr(“ibwrt error”)
                 CALL ibonl(dv%, 0)
                 CALL ibonl(bd%, 0)
                 STOP
        END IF

        cmd$ = “@SCRATCH”
        CALL ibwrt(na%, cmd$)

END SUB

'    This routine prints the result of status variables.
'
SUB prterr (msg$) STATIC

        PRINT msg$
        PRINT “ibsta=&H”; HEX$(ibsta%); “ <”;
        IF ibsta% AND EERR THEN PRINT “ ERR”;
        IF ibsta% AND TIMO THEN PRINT “ TIMO”;
        IF ibsta% AND EEND THEN PRINT “ EEND”;
        IF ibsta% AND SRQI THEN PRINT “ SRQI”;
        IF ibsta% AND RQS  THEN PRINT “ RQS”;
        IF ibsta% AND CMPL THEN PRINT “ CMPL”;
        IF ibsta% AND LOK  THEN PRINT “ LOK”;
        IF ibsta% AND RREM THEN PRINT “ RREM”;
        IF ibsta% AND CIC  THEN PRINT “ CIC”;
        IF ibsta% AND AATN THEN PRINT “ AATN”;
        IF ibsta% AND TACS THEN PRINT “ TACS”;
        IF ibsta% AND LACS THEN PRINT “ LACS”;
        IF ibsta% AND DTAS THEN PRINT “ DTAS”;
        IF ibsta% AND DCAS THEN PRINT “ DCAS”;
        PRINT “>”

        PRINT  “ iberr = ” ;  iberr% ;
        IF iberr% = EDVR THEN PRINT “ EDVR <DOS Error>”
        IF iberr% = ECIC THEN PRINT “ ECIC <Not CIC>”
        IF iberr% = ENOL THEN PRINT “ ENOL <NO listener>”
        IF iberr% = EADR THEN PRINT “ EADR <Address error>”
        IF iberr% = EARG THEN PRINT “ EARG <Invalid argument>”
        IF iberr% = ESAC THEN PRINT “ ESAC <Not Sys Ctrlr>”
        IF iberr% = EABO THEN PRINT “ EABO <Op. aborted>”
        IF iberr% = ENEB THEN PRINT “ ENEB <No GPIB board>”
        IF iberr% = EOIP THEN PRINT “ EOIP <Async I/O in prg>”
        IF iberr% = ECAP THEN PRINT “ ECAP <No capability>”
        IF iberr% = EFSO THEN PRINT “ EFSO <Files sys. error>”
        IF iberr% = EBUS THEN PRINT “ EBUS <Command error>”
        IF iberr% = ESTB THEN PRINT “ ESTB <Status byte lost>”
        IF iberr% = ESRQ THEN PRINT “ ESRQ <SRQ stuck on>”
        IF iberr% = ETAB THEN PRINT “ ETAB <Table Overflow>”
        PRINT “ibcnt”; ibcnt%

END SUB
```

12.7.4 Download Program in C

The download program to be executed with C is shown as follows.

Example 12-34 Download Program used for C

```
/*
 *     DOWN LOAD PROGRAM
 *
 * TARGET:    PC/AT(NI-488.2)
 * LANGUAGE: C (ANSI-C STYLE)
 * FILE:      MCDNLD.C
 */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include "decl.h"

static char line[512];

/* prterr - print gpib error message and status code
 */
static void prterr(char *msg)
 {
  printf("%s\n", msg);
  printf("ibsta=&H%x < ", ibsta);
  if (ibsta & ERR)  printf("ERR");
  if (ibsta & TIMO) printf("TIMO");
  if (ibsta & SRQI) printf("SRQI");
  if (ibsta & RQS)  printf("RQS");
  if (ibsta & CMPL) printf("CMPL");
  if (ibsta & LOK)  printf("LOK");
  if (ibsta & CIC)  printf("CIC");
  if (ibsta & TACS) printf("TACS");
  if (ibsta & LACS) printf("LACS");
  if (ibsta & DTAS) printf("DTAS");
```

```
    if (ibsta & DCAS) printf("DCAS");
    printf(" >\n");

    printf("iberr= %d", iberr);
    switch(iberr)
       {
      case EDVR: printf("EDVR <DOS Error>"); break;
      case ECIC: printf("ECIC <Not CIC>"); break;
      case ENOL: printf("ENOL <No listner>"); break;
      case EADR: printf("EADR <Address error>"); break;
      case EARG: printf("EARG <Invalid argument>"); break;
      case ESAC: printf("ESAC <Not Sys Ctrlr>"); break;
      case EABO: printf("EABO <Op.aborted>"); break;
      case ENEB: printf("ENEB <No GPIB board>"); break;
      case EOIP: printf("EOIP <Async I/O in prg>"); break;
      case ECAP: printf("ECAP <No capability>"); break;
      case EFSO: printf("EFSO <Fils sys. error>"); break;
      case EBUS: printf("EBUS <Command error>"); break;
      case ESTB: printf("ESTB <Status byte lost>"); break;
      case ESRQ: printf("ESRQ <SRQ stuck on>"); break;
      case ETAB: printf("ETAB <Table Overflow>"); break;
      }
    printf("ibcntl= %d\n\n", ibcntl);
  }


  /* gpinit - open gpib board and initialize
   */
  static int gpinit(char *bdname)
   {
    int     bd;

    if ((bd = ibfind(bdname)) < 0)          /* open board */
       {
       prterr ("ibfind error");
       return -1;
   }

  if (ibsic(bd) & ERR)                  /* interface clear */
     {
     prterr(“ibsic error”);
     ibonl(bd, 0);
     return -1;
   }

  if (ibsre(bd, 1) & ERR)               /* remote enable */
     {
     prterr(“ibsre error”);
     ibonl(bd, 0);
     return -1;
   }
  return bd;                            /* return descriptor */
}

/* nainit - open N.A Port and initialize
 */
static int nainit(char *dvname)
 {
  int   na;
  if ((na = ibfind(dvname)) < 0         /* open R3752/R3753 */
     {
      prterr(“ibfind error”);
      return -1;
    }

  ibwrt(na, “OLDC OFF;*RST”, 14);       /* default command */
  if (ibsta & ERR)
     {
     prterr(“ibwrt error”);
     ibonl(na, 0);
     return -1;
   }
```

```
    ibwrt(na, "@SCRATCH", 9);
    return na;
  }                                              /* return descriptor */

  /* main entry
   */
  main(int argc, char **argv)
   {
    char  name[64], *s;
    FILE  *fp;
    int   bd, na;
    int   n;

    printf("PROGRAM TRANSFER (PC/AT to NA)\n");
    printf("Enter DOWNLOAD PROGRAM ?");
    fflush(stdout);
    fflush(stdin);
    if (scanf("%s", name) <=0)
       {
        fprintf(stderr, "File name error\n");
        exit(1);
     }

    if ((bd = gpinit("GPIB0")) == -1)
      exit(1) ;
    if ((na = nainit("DEV11")) == -1)
       {
        ibonl(bd, 0);
        exit(1);
     }

    if ((fp = fopen(name, "r")) == NULL)
       {
        fprintf(stderr, "%s: not found\n", name);
        ibonl(na, 0);
        ibonl(bd, 0);
        exit ( 1) ;
      }
    line[0] = '@';
    while ((s = fgets(&line[1], 510, fp)) && *s ! = NULL)
       {
        printf("%s", line):
        ibwrt(na, line, strlen(line));
     }

    fclose(fp);
    ibonl(na, 0);
    ibonl(bd, 0);
  }
```

12.8 Upload of BASIC Program

This section describes a method with which the program existing in the memory of this instrument side is uploaded to the external controller.
Since the up loaded program is controlled by external controller side, the using method is almost the same as that of the download program, pay attention to the following note.

(Note) When the program of this instrument is uploaded with the upload program described in this example, let the last line of the program of this instrument be 65535 END. This is used to decide the lost line of the upload program.

The example of upload program to be executed with N88-BASIC is shown as follows.

Example 12-35 Upload Program Used for N88-BASIC

```
1000 ' ***************************************
1010 ' *                                     *
1020 ' *              UPLOAD PROGRAM         *
1030 ' *                                     *
1040 ' * TARGET: PC-9801                     *
1050 ' * FILE :  NDOWNLD.N88                 *
1060 ' ***************************************
1070 NA=11
1080 ISET IFC
1090 ISET REN
1100 CMD DELIM=0
1110 CMD TIMEOUT=3
1120 '
```

```
1130 ON ERROR GOTO *ERRORMES
1140 PRINT “PROGRAM UPLOAD (NA to PC)”
1150 '
1160 *UPLD.ENTER
1170     FLAG=0
1180     INPUT “ENTER NEW FILE NAME”;F$
1190     OPEN F$ FOR OUTPUT AS #1
1200     IF FLAG=1 THEN *UPLD.ENTER
1210 '
1220     PRINT “UpLoading... (Saving“;F$;”)”
1230     PRINT @NA;“@GLIST”
1240 '
1250 *UPLD. LOOP
1260     LINE INPUT @NA;DA$
1270     PRINT DA$
1280     PRINT #1,DA$
1290     IF DA$ <>“65535 END” THEN *UPLD.LOOP
1300     CLOSE
1310     PRINT “COMPLETE UPLOAD”
1320     ON ERROR GOTO 0
1330     END
1340 '
1350 *ERRORMES
1360     FLAG=1
1370     PRINT “ERROR: LINE=”;ERL;“NO.=”;ERR
1380     INPUT “RETRY? (Y/N)”;A$
1390     IF A$=”Y” OR A$=“y” THEN RESUME NEXT
1400     ON ERROR GOTO 0
1410     END
```

12.9 Transferring Correction Data

This section presents an example of program that performs input and output all correction data of two port calibrations between external controllers.
Transferring correction data is carried out with the same method as used for tracing (wave form) data.
There are two formats of transferring data : ASCII and binary, just as for the tracing data, the transferring can be performed in higher speed with binary format. Here, a transferring program in binary format is taken as an example and presented with N88-BASIC and C.

12.9.1 Transferring Correction Data Between This Instrument and PC-9801 (N88-BASIC)

This program is used to transfer the correction data between this instrument and PC-9801.

(Note) NEC pure GPIB interface board is used.

In the case of data receiving, the correction data from this instrument is transferred to PC-9801 and stored in the disk drive on PC-9801. Perform the two port calibrations in advance.
On the other hand, in the case of data sending , the data is transferred to this instrument after the data in the disk drive is loaded. It is necessary to keep the setting conditions such as point count, etc. as they are stored.

When this program is executed, 1 : Receive (SAVE) , 2 : Send (LOAD) ? is displayed. Input 1 to receive (save) the data , and input 2 to send (regenerate) the data. Since File name = ? is displayed next , input the file name.

Example 12-36 Transferring Correction Data Between This Instrument and PC-9801 (Binary format)

```
1000 ' *************************************
1010 '
1020 '       TRANSFER 2PORT CAL. DATA
1030 '
1040 ' *************************************
1050 '
1060 CLEAR &H100
1070 DEF SEG=SEGPTR(2)
1080 DIM TR1!(1201*2+4)
1090 '
1100 GOSUB *SETUP.GPIBCALL
1110 ISET IFC: ISET REN
1120 CMD DELIM=3
1130 PC98=IEEE(1) AND &H1F               ' my GPIB address
1140 NA=11                               ' target GPIB address
1150 BITLEN%=32                          ' bit length (32 or 64)
```

```
1160 PRINT @NA;"OLDC OFF" @
1170 PRINT @NA;"SWE:POIN?" @
1180 INPUT @NA;POINTS%
1190 PRINT @NA;"FORM MBIN,"+STR$(BITLEN%) @
1200 DT=12                                    ' number of traces
1210 NUMSET%=2*POINTS%*(BITLEN%/8)+9
1220 '
1230 *FORM.DATA
1240 DATA EDF,ESF,ERF,ELF,ETF,EXF
1250 DATA EDR,ESR,ERR,ELR,ETR,EXR
1260 '
1270 CLS
1280 INPUT "1:Receive(SAVE), 2:Send(LOAD)";MODE
1290 IF MODE<1 OR 2<MODE THEN END
1300 INPUT "File name = ";FILENAME$
1310 ON MODE GOSUB *SAVE.CALDATA,*LOAD.CALDATA
1320 END
1330 '
1340 ' save full calibration data
1350 *SAVE.CALDATA
1360 RESTORE *FORM.DATA
1370 OPEN FILENAME$ FOR OUTPUT AS #1
1380 FOR J=1 TO DT
1390 READ FORM$
1400 GOSUB *RECEIVE.TRACE
1410 GOSUB *WRITE.TRACE
1420 NEXT
1430 CLOSE #1
1440 RETURN
1450
1460 ' load full calibration data
1470 *LOAD.CALDATA
1480 RESTORE *FORM.DATA
1490 OPEN FILENAME$ FOR INPUT AS #1
1500 FORM$="DATA":GOSUB *RECEIVE.TRACE
1510 FOR J=1 TO DT
```

```
1520 READ FORM$
1530 GOSUB *READ.TRACE
1540 GOSUB *SEND.TRACE
1550 NEXT
1560 CLOSE #1
1570 PRINT @NA;"CORR:CSET:STAT ON" @
1580 RETURN
1590 '
1600 ' receive data on one trace
1610 *RECEIVE.TRACE
1620 PRINT @NA;"TRAC:DATA?" +FORM$ @          ' trace data read
1630 WBYTE &H3F,&H5F,&H40+NA,&H20+PC98;        ' set TALKER/LISTENER
1640 NUM%=NUMSET%                              ' read buffer size
1650 CALL RECEIVE.DATA(TR1!(0),NUM%)           ' read data
1660 RETURN
1670 '
1680 ' send data on one trace
1690 *SEND.TRACE
1700 NUM%=NUMSET%
1710 PRINT @NA;"TRAC:DATA"+FORM$+","
1720 CALL SEND.DATA(TR1!(0),NUM%)
1730 RETURN
1740 '
1750 ' write trace data into the file
1760 *WRITE.TRACE
1770 FOR I=0 TO 2*POINTS%-1
1780 PRINT #1,TR1!(1+2)
1790 NEXT 1
1800 RETURN
1810 '
1820 ' read trace data from the file
1830 *READ.TRACE
1840 FOR I=0 TO 2*POINTS%-1
1850 INPUT #1,TR1!(I+2)
1860 NEXT I
1870 RETURN
```

```
1880 '
1890 ' setup system calls
1900 *SETUP.GPIBCALL
1910 RECEIVE.DATA=&H0: SEND.DATA=&H39
1920 RESTORE *GPIB.BIOS
1930 FOR ADR=0 TO &H65
1940 READ BYTE$: POKE ADR,VAL("&H"+BYTE$)
1950 NEXT
1960 RETURN
1970 '
1980 *GPIB.BIOS
1990 DATA 50,51,52,06,56,57,55,53, 8B,4F,02,8E,C1,8B,37,26
2000 DATA 8B,0C,8B,7F,04,8E,47,06, BB,00,00,BE,00,00,B0,80
2010 DATA B4,05,CD,D1,5B,53,8B,4F, 02,8E,C1,8B,37,26,89,14
2020 DATA 5B,5D,5F,5E,07,5A,59,58, CF,50,51,52,06,56,57,55
2030 DATA 53,8B,4F,02,8E,C1,8B,37, 26,8B,0C,8B,7F,04,8E,47
2040 DATA 06,BB,00,00,BE,00,00,B0, 80,B4,04,CD,D1,5B,5D,5F
2050 DATA 5E,07,5A,59,58,CF
```

12.9.2 Transferring Correction Data Between This Instrument and PC/AT (C language)

This program is used to transfer the correction data between this instrument and PC/AT.
In the case of data receiving , the correction data from this instrument is transferred to PC/AT, and stored in the disk drive on PC/AT. Perform the two port calibrations in advance.
On the other hand, in the case of data sending, the data is transferred to this instrument after the data in the disk drive is loaded. It is necessary to keep the setting conditions such as point count, etc. as they are stopped.

When this program is executed, 1 : Receive (SAVE) , 2: Send (LOAD) ? is displayed.
Input 1 to receive (save) the data and input 2 to send (regenerate) the data. Since File name = ? is displayed next , input the file name.

(Note) NI-488.2 interface board and library functions are used.

Example 12-37 Transferring Correction Data Between This Instrument and PC/AT(Binary format)

```
/* Transfer two port full calibration data via GPIB
 * between R376X and DOS/V PC with NI-488 GPIB board
 * FORMat [:DATA] REAL,32
 * How to compile: bcc trans.c mcib.lib
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "decl.h"               /* NI-488.2 headers <DOS> */

#define ADDRESS 11              /* target GPIB address */
#define KIND    12              /* kinds of cal. data */
#define COMP    2               /* 1(FDATn) or 2(others) */
#define BYTE    4               /* 4(REAL,32) or 8(REAL,64) */
#define HEAD    8               /* block header length */
#define FOOT    1               /* block footer(LF) length */
#define TRAC    9               /* command " RAC E**," length */
#define BUFLEN (HEAD+1201 * BYTE * COMP+FOOT)
#define EOT__CONFIG 1           /* ibeot configuration value */
#define EOS__CONFIG 0           /* ibeos configuration value */

char *form[] =  {
  "EDF","ESF","ERF","ELF","ETF","EXF",
  "EDR","ESR","ERR","ELR","ETR","EXR"
  } ;

void gpib__err(int id, char *msg)
 {
  if (id == -1)
    fprintf(stderr,"%s¥n", msg);
  else
     {
      fprintf(stderr, "%s: ibsta=0x%x, iberr=%d, ibcnt=%d¥n",
              msg, ibsta, iberr, ibcnt);
```

```
        ibonl(id,0);
      }
    exit(-1);
  }



int gpib__init(int address)
 {
   int id;

  if ((id = ibdev(0, address, 0, T1s, EOT__CONFIG, EOS__CONFIG)) <0)
     gpib__err(id, "ibdev error");

   if (ibdma(0, 1) & ERR)            /* enable DMA transfer */
     gpib__err(id, "ibdma error");

   return id;                        /* return device identifier */
 }

void gpib__end(int id)
 {
   if (ibonl(id, 0) & ERR)           /* interface offline */
     gpib__err(id,"ibonl error");
 }

int send__buf(int id, char *buf)
 {
   int len;

   len = strlen(buf);
   if (ibwrt(id, buf, (long)len) & ERR)       /* IBWRT */
     gpib__err(id, "ibwrt error");
   return ibcntl;                    /* return actual sent bytes */
 }

int receive__buf(int id, char *buf)
```

```
 {
  int eval, count = 0;

  while (1)
     {
      if ((eval = ibrd(id, buf, (long)BUFLEN)) & ERR)   /* IBRD */
        gpib__err(id, “ibrd error”);
      count += ibcntl;                          /* sum total length */
      if (eval & END)                           /* END or EOS detected */
        break;
    }
  return count;                  /* return actual received bytes */
}

float btof(char *buf)            /* 32bit raw binary to float */
 {
  char tmp[4];

  tmp[3] = buf[0];
  tmp[2] = buf[1];
  tmp[1] = buf[2];
  tmp[0] = buf[3];
  return *((float *)tmp)
}

void ftob(float *f, char *buf)    /* float to 32bit raw binary */
 {
  buf[3] = *((char *)f + 0);
  buf[2] = *((char *)f + 1);
  buf[1] = *((char *)f + 2);
  buf[0] = *((char *)f + 3);
}

void save__caldata(int id, char *buf, char *filename)
 {
  FILE *fp;
  int i, j, len;

  if ((fp = fopen(filename, "w")  == NULL)
    gpib__err(-1, "File open error");
  for (j = 0; j < KIND; j++
     {
     sprintf(buf, "TRAC? %3s", form[j]);
     send__buf(id, buf);
     len = receive__buf(id, buf);

     for (i = 0; i < (len - HEAD - FOOT) / BYTE; i++)
       fprintf(fp, "%f¥n", btof (buf + HEAD + i * BYTE));
    }
  fclose(fp);
}

void load__caldata(int id, char *buf, char *filename)
 {
  FILE *fp;
  float f;
  int i, j, pts;

  send__buf(id, "SWE:POIN?");
  receive__buf(id, buf);
  sscanf(buf, "%d", &pts);

  if ((fp = fopen(filename, "r")) == NULL)
    gpib__err(-1, "File open error");

  for (j = 0; j < KIND; J++)
     {
      sprintf(buf, "TRAC %3s,#6%06d", form[j], pts * BYTE * COMP);

      for (i = 0; i < pts * COMP; i++)
         {
         fscanf(fp, "%f", &f);
         ftob(&f, buf + TRAC + HEAD + i * BYTE);
      }
     *(buf + TRAC + HEAD + pts * BYTE * COMP)  = ‘¥n’;

     if (ibwrt(id, buf, (long)(TRAC + HEAD + pts * COMP * BYTE +
                                                       FOOT))& ERR)
       gpib__err(id, “ibwrt error”)
     }
  send__buf (id,  “ CORR:CSET:STAT  ON”);
}

void main(void)
 {
  int id;
  char *buf, filename[20];
  if ((buf = malloc(BUFLEN)) == NULL)
    gpib__err(-1, “Memory allocation error”);

  id = gpib__init (ADDRESS);
  send__buf(id, “OLDC OFF”);
  send__buf(id, “FORM REAL,32”);

  while (1)
     {
     printf(“1:Receive(SAVE), 2:Send(LOAD) ?”);
     if (strchr (“12”, *gets(buf)) != NULL)
       break;
     }

  printf(“File name =?”);
  gets (filename);

  switch(* buf)
     {
    case ‘1’:
      save__caldata(id, buf, filename);
      break;
    case `2':
      load__caldata(id, buf, filename);
      break;
    }

  gpib__end(id);
}
```

ALPHABETICAL INDEX

IMPORTANT INFORMATION FOR ADVANTEST SOFTWARE

PLEASE READ CAREFULLY: This is an important notice for the software defined herein. Computer programs including any additions, modifications and updates thereof, operation manuals, and related materials provided by Advantest (hereafter referred to as "SOFTWARE"), included in or used with hardware produced by Advantest (hereafter referred to as "PRODUCTS").

SOFTWARE License

All rights in and to the SOFTWARE (including, but not limited to, copyright) shall be and remain vested in Advantest. Advantest hereby grants you a license to use the SOFTWARE only on or with Advantest PRODUCTS.

Restrictions

(1) You may not use the SOFTWARE for any purpose other than for the use of the PRODUCTS.
(2) You may not copy, modify, or change, all or any part of, the SOFTWARE without permission from Advantest.
(3) You may not reverse engineer, de-compile, or disassemble, all or any part of, the SOFTWARE.

Liability

Advantest shall have no liability (1) for any PRODUCT failures, which may arise out of any misuse (misuse is deemed to be use of the SOFTWARE for purposes other than it's intended use) of the SOFTWARE. (2) For any dispute between you and any third party for any reason whatsoever including, but not limited to, infringement of intellectual property rights.

LIMITED WARRANTY

1. Unless otherwise specifically agreed by Seller and Purchaser in writing, Advantest will warrant to the Purchaser that during the Warranty Period this Product (other than consumables included in the Product) will be free from defects in material and workmanship and shall conform to the specifications set forth in this Operation Manual.

2. The warranty period for the Product (the "Warranty Period") will be a period of one year commencing on the delivery date of the Product.

3. If the Product is found to be defective during the Warranty Period, Advantest will, at its option and in its sole and absolute discretion, either (a) repair the defective Product or part or component thereof or (b) replace the defective Product or part or component thereof, in either case at Advantest's sole cost and expense.

4. This limited warranty will not apply to defects or damage to the Product or any part or component thereof resulting from any of the following:

   (a) any modifications, maintenance or repairs other than modifications, maintenance or repairs (i) performed by Advantest or (ii) specifically recommended or authorized by Advantest and performed in accordance with Advantest 's instructions;

   (b) any improper or inadequate handling, carriage or storage of the Product by the Purchaser or any third party (other than Advantest or its agents);

   (c) use of the Product under operating conditions or environments different than those specified in the Operation Manual or recommended by Advantest, including, without limitation, (i) instances where the Product has been subjected to physical stress or electrical voltage exceeding the permissible range and (ii) instances where the corrosion of electrical circuits or other deterioration was accelerated by exposure to corrosive gases or dusty environments;

   (d) use of the Product in connection with software, interfaces, products or parts other than software, interfaces, products or parts supplied or recommended by Advantest;

   (e) incorporation in the Product of any parts or components (i) provided by Purchaser or (ii) provided by a third party at the request or direction of Purchaser or due to specifications or designs supplied by Purchaser (including, without limitation, any degradation in performance of such parts or components);

   (f) Advantest's incorporation or use of any specifications or designs supplied by Purchaser;

   (g) the occurrence of an event of force majeure, including, without limitation, fire, explosion, geological change, storm, flood, earthquake, tidal wave, lightning or act of war; or

   (h) any negligent act or omission of the Purchaser or any third party other than Advantest.

5. **EXCEPT TO THE EXTENT EXPRESSLY PROVIDED HEREIN, ADVANTEST HEREBY EXPRESSLY DISCLAIMS, AND THE PURCHASER HEREBY WAIVES, ALL WARRANTIES, WHETHER EXPRESS OR IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, (A) ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE AND (B) ANY WARRANTY OR REPRESENTATION AS TO THE VALIDITY, SCOPE, EFFECTIVENESS OR USEFULNESS OF ANY TECHNOLOGY OR ANY INVENTION.**

6. **THE REMEDY SET FORTH HEREIN SHALL BE THE SOLE AND EXCLUSIVE REMEDY OF THE PURCHASER FOR BREACH OF WARRANTY WITH RESPECT TO THE PRODUCT.**

7. **ADVANTEST WILL NOT HAVE ANY LIABILITY TO THE PURCHASER FOR ANY INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL OR PUNITIVE DAMAGES, INCLUDING, WITHOUT LIMITATION, LOSS OF ANTICIPATED PROFITS OR REVENUES, IN ANY AND ALL CIRCUMSTANCES, EVEN IF ADVANTEST HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES AND WHETHER ARISING OUT OF BREACH OF CONTRACT, WARRANTY, TORT (INCLUDING, WITHOUT LIMITATION, NEGLIGENCE), STRICT LIABILITY, INDEMNITY, CONTRIBUTION OR OTHERWISE. TORT (INCLUDING, WITHOUT LIMITATION, NEGLIGENCE), STRICT LIABILITY, INDEMNITY, CONTRIBUTION OR OTHERWISE.**

8. **OTHER THAN THE REMEDY FOR THE BREACH OF WARRANTY SET FORTH HEREIN, ADVANTEST SHALL NOT BE LIABLE FOR, AND HEREBY DISCLAIMS TO THE FULLEST EXTENT PERMITTED BY LAW ANY LIABILITY FOR, DAMAGES FOR PRODUCT FAILURE OR DEFECT, WHETHER ARISING OUT OF BREACH OF CONTRACT, TORT (INCLUDING, WITHOUT LIMITATION, NEGLEGENCE), STRICT LIABILITY, INDEMNITY, CONTRIBUTION OR OTHERWISE.**

CUSTOMER SERVICE DESCRIPTION

In order to maintain safe and trouble-free operation of the Product and to prevent the incurrence of unnecessary costs and expenses, Advantest recommends a regular preventive maintenance program under its maintenance agreement.

Advantest's maintenance agreement provides the Purchaser on-site and off-site maintenance, parts, maintenance machinery, regular inspections, and telephone support and will last a maximum of ten years from the date the delivery of the Product. For specific details of the services provided under the maintenance agreement, please contact the nearest Advantest office listed at the end of this Operation Manual or Advantest 's sales representatives.

Some of the components and parts of this Product have a limited operating life (such as, electrical and mechanical parts, fan motors, unit power supply, etc.). Accordingly, these components and parts will have to be replaced on a periodic basis. If the operating life of a component or part has expired and such component or part has not been replaced, there is a possibility that the Product will not perform properly. Additionally, if the operating life of a component or part has expired and continued use of such component or part damages the Product, the Product may not be repairable. Please contact the nearest Advantest office listed at the end of this Operation Manual or Advantest's sales representatives to determine the operating life of a specific component or part, as the operating life may vary depending on various factors such as operating condition and usage environment.

SALES & SUPPORT OFFICES

Advantest Korea Co., Ltd.
22BF, Kyobo KangNam Tower,
1303-22, Seocho-Dong, Seocho-Ku, Seoul #137-070, Korea
Phone: +82-2-532-7071
Fax: +82-2-532-7132

Advantest (Suzhou) Co., Ltd.
Shanghai Branch Office:
Bldg. 6D, NO.1188 Gumei Road, Shanghai, China 201102 P.R.C.
Phone: +86-21-6485-2725
Fax: +86-21-6485-2726

Shanghai Branch Office:
406/F, Ying Building, Quantum Plaza, No. 23 Zhi Chun Road,
Hai Dian District, Beijing,
China 100083
Phone: +86-10-8235-3377
Fax: +86-10-8235-6717

Advantest (Singapore) Pte. Ltd.
438A Alexandra Road, #08-03/06
Alexandra Technopark Singapore 119967
Phone: +65-6274-3100
Fax: +65-6274-4055

Advantest America, Inc.
3201 Scott Boulevard, Suite, Santa Clara, CA 95054, U.S.A
Phone: +1-408-988-7700
Fax: +1-408-987-0691

ROHDE & SCHWARZ Europe GmbH
Mühldorfstraße 15 D-81671 München, Germany
(P.O.B. 80 14 60 D-81614 München, Germany)
Phone: +49-89-4129-13711
Fax: +49-89-4129-13723

ADVANTEST®

http://www.advantest.co.jp

ADVANTEST CORPORATION
Shin-Marunouchi Center Building, 1-6-2 Marunouchi, Chiyoda-ku, Tokyo 100-0005, Japan
Phone: +81-3-3214-7500