ADVANTEST®
ADVANTEST CORPORATION

R3752/53/64/65/66/67H Series
R3765/67G Series
R3754 Series
Network Analyzer
Programming Guide

MANUAL NUMBER FEE-8324174C02

Applicable models
R3752AH/BH/EH
R3753AH/BH/EH
R3764AH/BH/CH
R3765AH/BH/CH
R3766AH/BH/CH
R3767AH/BH/CH
R3765AG/BG/CG
R3767AG/BG/CG
R3754A/B/C

First printing January 20, 1997
Printed in Japan

Safety Summary

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

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

• Warning Labels

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

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

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

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

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

• Basic Precautions

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

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

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

- **Caution Symbols Used Within this Manual**

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

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

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

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

- **Safety Marks on the Product**

The following safety marks can be found on Advantest products.

⚠ : ATTENTION - Refer to manual.

⏚ : Protective ground (earth) terminal.

ϟ : DANGER - High voltage.

⚠ : CAUTION - Risk of electric shock.

- **Replacing Parts with Limited Life**

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

There is a possibility that each product uses different parts with limited life. For more inform tion, refer to Chapter 1.

Main Parts with Limited Life

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

• Hard Disk Mounted Products

The operational warnings are listed below.

- Do not move, shock and vibrate the product while the power is turned on.
  Reading or writing data in the hard disk unit is performed with the memory disk turning at a high speed. It is a very delicate process.

- Store and operate the products under the following environmental conditions.
  An area with no sudden temperature changes.
  An area away from shock or vibrations.
  An area free from moisture, dirt, or dust.
  An area away from magnets or an instrument which generates a magnetic field.

- Make back-ups of important data.
  The data stored in the disk may become damaged if the product is mishandled. The hard disc has a limited life span which depends on the operational conditions. Note that there is no guarantee for any loss of data.

• Precautions when Disposing of this Instrument

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

Harmful substances: (1) PCB (polycarbon biphenyl)
(2) Mercury
(3) Ni-Cd (nickel cadmium)
(4) Other
Items possessing cyan, organic phosphorous and hexadic chromium and items which may leak cadmium or arsenic (excluding lead in sol der).

Example: fluorescent tubes, batteries

Environmental Conditions

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

- An area free from corrosive gas
- An area away from direct sunlight
- A dust-free area
- An area free from vibrations

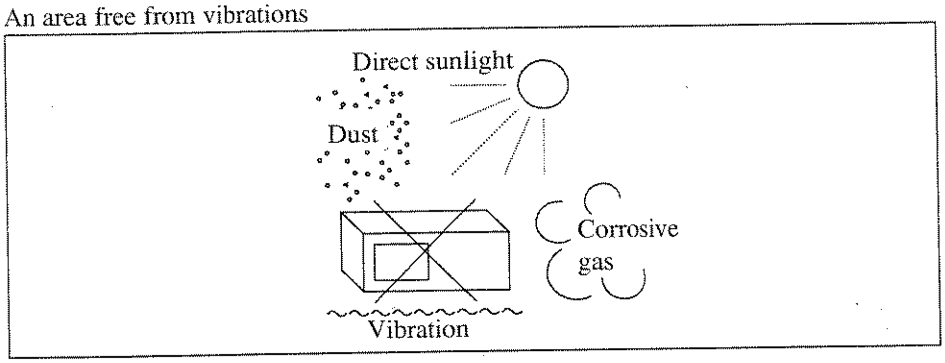
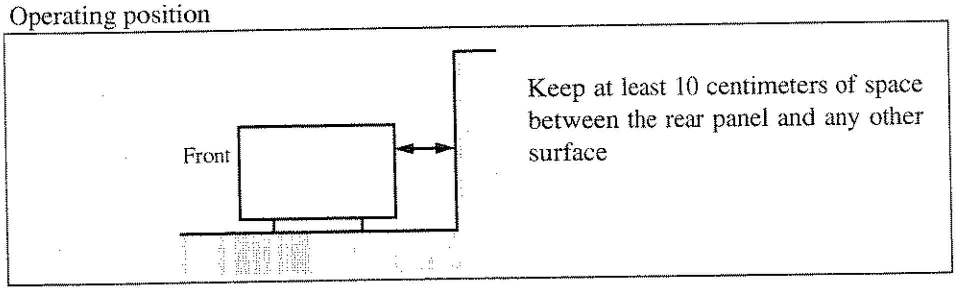


Figure-1 Environmental Conditions

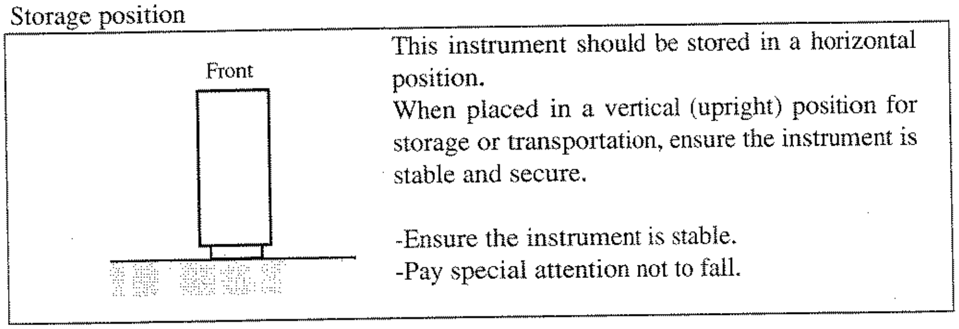
- Operating position



Keep at least 10 centimeters of space between the rear panel and any other surface

Figure-2 Operating Position

- Storage position



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

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

Figure-3 Storage Position

This instrument can be used safely under the following conditions:

- Altitude of up to 2000 m
- Installation Categories II
- Pollution Degree 2

PREFACE

This manual describes how to create and execute BASIC programs using the built-in editor in a main unit of the R3752/53/64/65/66/67H Series, R3765/67G Series or R3754 Series.

1. Relevant manuals

- Manuals that describe the names, functions and key operations of the network analyzer.

  R3752H series Operation Manual
  R3753H series Operation Manual
  R3764/66H series Operation Manual
  R3765/67H series Operation Manual
  R3765/67G series Operation Manual
  R3754 Series Network Analyzer User Manual (Functional Descriptions)

- Manuals that describe the built-in BASIC and GPIB

  R3752/53H, R3754 series Programming Manual
  R3764/65/66/67H, R3765/67G series Programming manual

2. Distinction of panel key and soft key in this manual.

Panel keys :	(Example)	**[CH 1] , [5]**
Soft keys :	(Example)	*{POWER}, {LOG MAG}*
IBM PC keyboards :	(Example)	***ENTER*, *Backspace***

3. Distinction between the commands used in this manual

Commands entered from the keyboard :	(Example)	EDIT command, Print
Commands selected from editor menus:	(Example)	**EDIT** command, **Print**

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

1. Introduction

The R3752/53/64/65/66/67H Series, R3765/67G Series and R3754 Series (analyzers) supports the BASIC programming environment, so BASIC programs created on external personal computers etc. can be executed. In addition, the editor described in this manual can be used to develop programs.

1.1 Features of the BASIC Language Provided

The BASIC language can load and execute BASIC programs without line numbers. When a BASIC program without line numbers is loaded, it is numbered automatically from the first line. Since line numbers are not necessary, program editing can be performed easily.

Program files can be loaded and executed using the editor. When the programs are executed with this editor, any errors occurring during the execution (runtime error) are traced.

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

1.2 Equipment Hardware Environment

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

- Network analyzer R3764/65/66/67H Series, R3765/67G Series or R3754 Series
  Or network analyzers with system software version B00 or higher, R3752/53H Series.
- IBM-PC compatible keyboard 101 or 106
- External CRT display (for R3752/64/66H Series)

2. Fundamentals of Programming

2.1 Program Mode

2.1.1 Starting the Program Mode

1. Starting the program mode

   To create a BASIC program using the key board connected to the network analyzer, the system must be switched to the program mode from the measurement mode.

- R3752/64/66H series

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

- R3753/65/67H Series, R3765/67G Series and R3754 Series

  When the power is switched on, the measurement mode is started automatically with these series. To switch to the program mode, proceed as follows:

  1. To enter the program mode, press **[RUN]** on the front panel.

     The program menu is displayed and the cursor appears on the screen. (Refer to Figure 2-1.)

  2. To exit the program mode, press any key on the front panel except **[CH1]** or **[CH2]**.

     When the program mode has been exited, the cursor disappears from the screen.

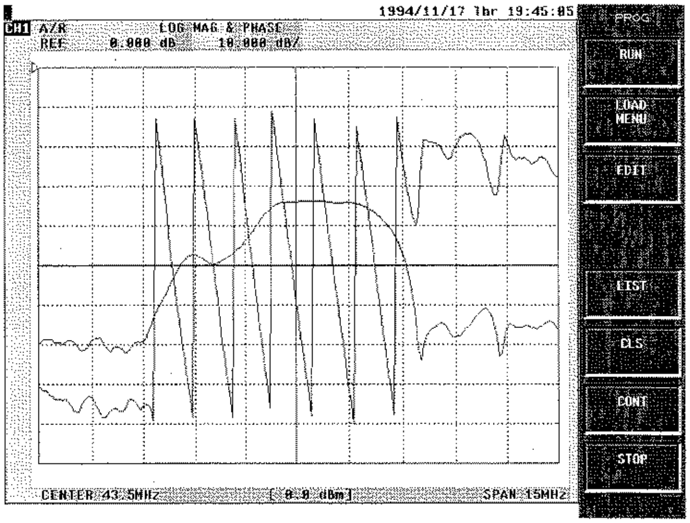


Figure 2-1 Program (Direct) Mode Screen

2. Program mode state

   BASIC commands and programs can be entered from the external keyboard when in the pr mode.

   The program mode consists of the "direct mode" and the "edit mode". When first entered, tl gram mode is set to the direct mode.

   In the direct mode, entered character strings are executed immediately after they are sent to I directly. (Refer to 2.1.2.)

   In the edit mode, entered character strings are not executed immediately. (Refer to 2.6.)

   The measurement can also be performed when in the program mode. In the direct mode, th sured waveform is left as the waveform displayed. Character strings entered and the PRINT of the BASIC are shown overlapped with the measurement waveform.

   The part in which the character string is displayed is called the "text screen", and the part in the measurement waveform is displayed is called the "measurement screen". These two scre displayed synthetically.

   As a result, the characters on the screen may not be clear, depending on their position in rel the measurement waveform. If this occurs, use one of the following methods. (Refer to (3).)

3. Methods used for displaying text clearly.

   There are two methods as follows:

- Disable the measurement screen using the DISP statement to display the text screen only (th form data cannot be seen .).
- Divide the full screen into two parts: the upper and the lower (the upper is the measurement 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 CONSOL ment in the program mode.

NOTE: ***Only the active channel is available in this method. This method is not available when dual ch ON (DUAL CH ON).***

2.1.2 Direct Mode

In the direct mode, the cursor move key cannot move 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.

To remove a character, press the ***Backspace*** key.

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

There are two ways to input commands from the keyboard.

- Inputting a command which is executed immediately (direct format).
- Creating a processing procedure using multi-commands (program format).

When the direct mode is used, each statement of BASIC is executed directly and its' operation can be ensured. If the statement is executed in direct mode, the operation of the statement can be ensured even if the program is not executed as a whole,

- Executing commands in direct mode

1. Enter CLS and press ***Enter***, so that the full text screen can be deleted.
2. Input the following statement.

```
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 immed is used only for calculations or commands which do not need to be stored, such as a list of programs been input so far.

2.2.1 PRINT Command

The PRINT command is used to evaluate equations entered after PRINT and it displays the resul screen. It can also be used to display character strings instead of equations on the screen.

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

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

  Input the following statements while in direct mode, then press ***Enter*** and check the results c eration.

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 strings on the screen.

  A character string is a sequence of characters enclosed in double quotation marks " ".

1. Input the following statement.

```
PRINT  "Hello, world"
```

2. Press ***Enter***.

   Execution result: Hello, world

Calculation results can be assigned to a variable and then output by using that variable.

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.

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

2. Press ***Enter***.

   Enter the following to dipslay the value of A.

```
PRINT A;
```

   Execution result: 64.0

3. Enter the following to dipslay the value of B.

```
PRINT B
```

   Execution result: 13.0

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

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.

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

2. Press ***Enter***.

   Enter the following to dipslay the value of A.

```
PRINT A;
```

   Execution result: 64.0

3. Enter the following to dipslay the value of B.

```
PRINT B
```

   Execution result: 13.0

In these example, the equations $(5+2)^2$ and $2\times5+6\div2$ are calculated and the answers are assigned to A and B, respectively. The contents of A and B are then displayed using the PRINT command.

2.3 Creating and Executing Programs

In the program mode, programs are stored in the memory, then a command is executed. As a result, program has been executed it still remains in memory, even when the command, having been execu appears. These programs can be executed several times.

2.3.1 Program Input and Execution (LIST, RUN)

Programs can be input by adding a line number to the front of command input statements describe

A line of this type is called "program line" while a line without a specified line number is called "co line".

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

Each program line of BASIC is preceded by a line number.The line number indicates the sequen which the program lines are stored into the memory, and the program execution follows this orde

- Program example

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

1. Displaying the program lines (LIST).

   The LIST command is used to display a list of the program lines in the scroll area of the scre

   1. Input the following command.

   ```
   LIST
   ```

   2. Press ***Enter***.

2. Displaying part of the program (LIST start line, end line).

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

   NOTE: ***If the starting line is omitted, the program is displayed from the first line to the end line If the end line is omitted, a list from the start line to the last line of the program is displ both of the two lines are omitted, the entire program is displayed.***

   1. Input the following statement.

   ```
   LIST 10,40
   ```

   2. Press ***Enter***.

   Execution result: displays the program from line 10 to line 40.

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

   The RUN command is used to execute the program entered.

   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)

The SCRATCH command is used to erase the programs and variables that have been input. When entering a new program, be sure to erase the programs entered previously.

- 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 calculation data has been integrated into the programs in advance. The following is a program which is designed to allow the user to enter the data, and then print out the result.

- Program example for calculating the area of a triangle

  1. Input CLS command to clear the screen.

  2. Input the following program.

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

  3. Input the Run command to execute the program.

  4. You can now enter values for "TEIHEN?=" and "TAKASA?=" using the keyboard.

     The area of triangle is calculated and displayed as "KOTAE".

2.4 Saving Programs (INITIALIZE, SAVE)

Programs input with line numbers will be stored in memory, however they will be lost when the p turned off. To save programs to floppy disks, use the following procedure.

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

NOTE: ***2DD floppy disks cannot be initialized with a 2HD format.***
Conversely, 2HD disk cannot be initialized with the 2DD format.

The software structure of a floppy disk in the network analyzer is the same as that of MS-DOS. Th the floppy disks initialized with the MS-DOS format can be used just as they are.

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

- Initializing floppy disks (INITIALIZE)

  For the 2DD floppy disk:

  1. Close all the opened files with 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 ume names can be up to 12 characters long, and can be chosen freely. The size of the floppy disk after initialization is 720K bytes.

- For the 2HD floppy disk

  Specifications are 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 to 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 up to 8 characters and have an extension of 3 characters at 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 a 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 a personal computer, the program can be saved in this sub-directory.
When a program is saved 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 available which can be used to create a sub-directory in 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:

1. ***Input the file name under which the program was saved.***
2. ***When the new program is loaded, the program entered previously is removed. (The LOAD command is executed with the SCRATCH command.)***

2.5.1 Searching File Names (CAT)

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

1. Insert the floppy disk into the drive.
2. Input the CAT command.

```
CAT
```

Execution result: The following is displayed in the scroll area of the te

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

The information shown on the screen from left to right represents tl number, file name, character counts of file and file attribute.

2.5.2 Outputting to Printer (GLIST, LLIST)

A copy of the program can also be printed out on the printer using a printer device.

Printer output can be performed in the following two ways.

- Using the GLIST command, output to a printer via GPIB.
- Using the 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, th GPIB should be set to SYSTEM CONTROLLER in advance and the GPIB address of printer m

- 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: Prints out the lines from 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 cannot move freely. Therefore, to a program line that has been input previously, it is necessary to redo the whole line. This is very incon

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

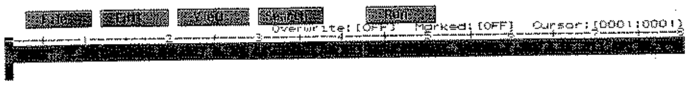
Here, you can edit simple programs with this editor. For details about the editor, refer to " 3. Func BASIC Editor".

2.6.1 Starting the Editor

- Turning to edit mode

  Press ***F12*** on the external keyboard to enter the edit mode. In the edit mode, the screen ap 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 available in the editor.

Program editing, file management, printing, and other functions 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 can be found here. By using these commands, creating and revising programs can be carried out in the view window.

2.6.3 Opening Menu

Almost every command of the editor can be 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 ...) on the keyboard.

To open a menu, press the corresponding function key.

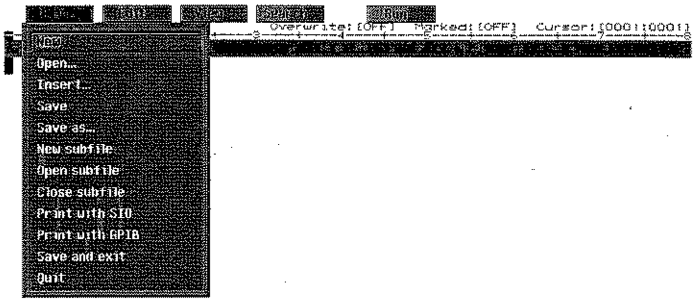
If the menu has been opened, selecting commands is done using the direction keys (↑, ↓, ←, →)

- 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 a command, press the ***Esc*** key.



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

Figure 2-3 F1: File Menu

2.6.4 Choosing the Edit Command

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

If a command name is followed by ellipsis dots (...), after selecting the command, a dialog box shown in Figure 2-4. If a command name is not followed by ellipsis, the command is execut ately.

2.6.5 Using a Dialog Box

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

1. Components of a dialog box

   Dialog boxes are used to provide editor information.

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

2. Operation of a dialog box

   If you open a dialog box for the first time, the default setting is displayed.

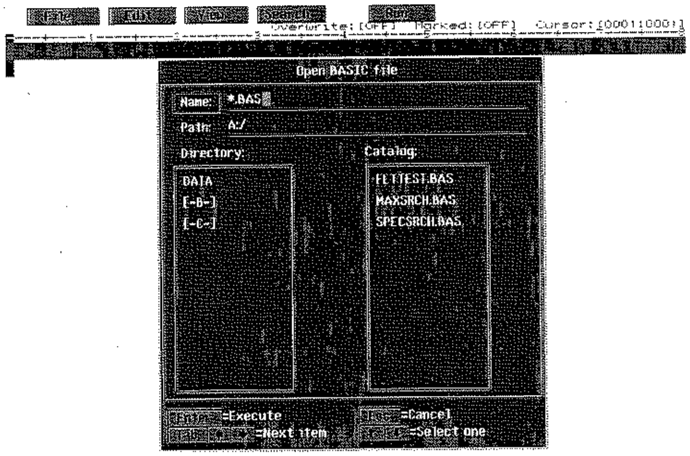
   After that, if you open the dialog box, the previously selected setting is displayed.

   To choose any items in a dialog box, use ***Tab***, ← or →.

- Opening the 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 highlighted.
  3. Press ***Enter*** to open the dialog box.
  4. Press ***Tab*** or → continually to move the cursor to the "Catalog :" list

     When there are a lot of items in a list box, changing the selection in t be carried out by pressing ↓.
  5. Press ***Esc*** to close the dialog box, without reading the file.



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

Name: *.BAS

Path: A:/

Directory:

DATA
[-B-]
[-C-]

Catalog:

FLTTEST.BAS
MAXSRCH.BAS
SPECSRCH.BAS

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

2.6.6 Executing the Edit Command Directly

The edit command can be executed directly when it is 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 commands

   The shortcut keys used for a particular command are indicated to the right of the command name.

   (Example) When cutting and pasting

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

2. Other keys used in combination

   Almost every key is used in combination with ***Shift***, ***Alt***, ***Ctrl*** and the 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 the Editor

- Closing editor

1. Execute the **Quit** command from the F1: File menu.
2. When a program has not been saved, a message appears asking whethe to quit without saving the program.

   Pressing ***Y*** changes the environment to direct mode without saving the Pressing ***N*** returns the screen to the original edit and executes the pro again after saving the file.

2.7 Editing Programs

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

Editing a program refers to operations which can revise,add to or correct a written program. When a program line is input in the direct mode, a line number must be included. However, line numbers are not necessary in the edit mode.

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

CAUTION: ***The program examples described later are shown without line numbers.***

2.7.1 Inserting Characters

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

After doing this, 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.

```
PRNT "NUMBER"
```

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

```
PRINT "NUMBER"
```

Pressing ***Insert*** adds a space at the cursor location. 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 can be entered on each line.
However, the screen can only display up to 80 characters on one line.
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 in a program, press ***Insert*** while holding

- 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 "(this program as follows.

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

```
INPUT A
C=A*B
```

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

```
INPUT  A
▨
C=A*B
```

A blank line also can be inserted by pressing ***Enter***. In this case, tl moved to the head of the next line.

To split a line into two parts, press the ***Insert*** key while holding ***Ctrl*** dc cute ***Enter*** at the middle of the line. This causes the characters from position to the end of the line to 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 with ***Backspace***

    1. Input the following line.

```
PRIINT  "A"□
```

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

```
PRIINT  "A"
```

    3. Press ***Backspace***.

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

```
PRINT  "A"
```

    If ***Backspace*** is used 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 using the cursor move keys.

```
PRIINT    "A"
```

    3. Press Delete.

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

```
PRINT    "A"
```

    If ***Delete*** is used at the end of statement, the lines can be connected.
    The next line will move to the cursor position completely.

2.7.4 Block Editing

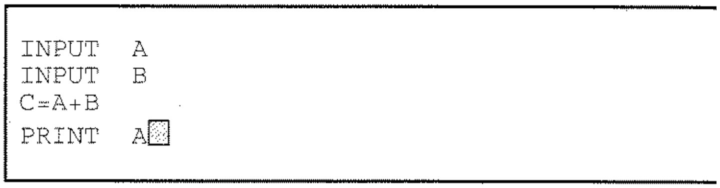
Block editing is an operation used to specify the edit lines in an integrated way.

Using the editor, editing can be performed on both the characters and lines. When the lines in an integrated way, deleting and moving can be performed in a wide range in integrated wa grated lines are called "block". To select a block, move the cursor to the head of the target the mark, then execute the edit command.

1. Setting mark

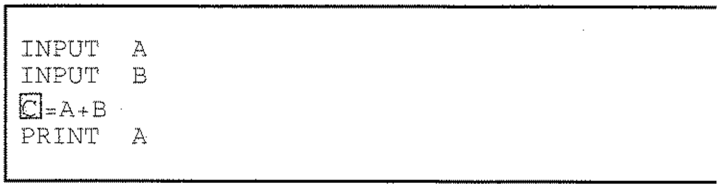
   The mark is set at the head of block editing line.

   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 m



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

   3. Press ***Space*** while 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 t ing. If you press ***Space*** while holding ***Alt*** down, then the ma exchange. The cursor moves to the mark position, and the mark is rent cursor position.

2. Cut and copy

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

NOTE: ***When the mark is not set, the cutting and copying cannot 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 dele grated way.

To copy a block means that the copying is performed for the selected block.

The cut block and copied block are stored in an area of memory called the "clip board"

The contents of the clip board cannot be viewed directly. They are displayed when p formed. Anytime cutting, copying or line deletes are executed, the lines or text are sto board. The clip board will contain the most recent lines or text.

The content of clip board can be inserted at the cursor position by using the pasting scribed later.

- Executing the **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*** while holding ***Shift*** down, the command can be executed with one key operation.
If you press ***Backspace*** only, merely one character to the left of the cursor position is deleted.

3. Pasting a text block

The text in the clip board remains until new text is sent to the clip board or the editor is closed.

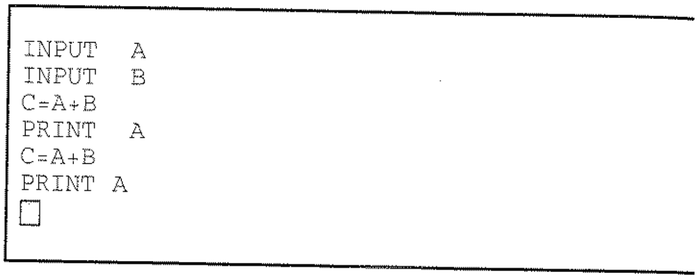
- Pasting program line

  1. Input the following program line

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

  2. Press ***Space*** while holding ***Ctrl*** down to set the mark at the cursor position.
  3. Use the arrow key to move the cursor to the head of C=A+B.
  4. Press ***Space*** while holding ***Alt*** down, then the mark and cursor exchange.
  5. Press ***F2*** to open the F2: Edit menu.
  6. Execute the **Copy** command.
  7. Press ***F2*** to open the F2: Edit menu.

8. Execute the **Paste** command.



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

If you press ***Insert*** while holding ***Shift*** down, this command can be one key operation.
If you Press ***Insert*** only, one space is inserted into the cursor pos

3. Functions of BASIC Editor

In this chapter, various BASIC editor functions described in " 2.6 Editing Mode" are explained in more detail. As mentioned above, all the operations necessary for programming such as editing and debugging can be performed using the pull-down menu holded by editor.

The following items are described.

- starting the BASIC Editor
- executing menu commands
- selecting dialog box options
- scrolling list box
- selecting dialog box and text in windows
- changing the window size
- using the direct screen
- using the watching window

3.1 Starting the BASIC Editor

The editor can be started from the external keyboard. (Refer to 3.1.1.)

3.1.1 Sarting the Editor

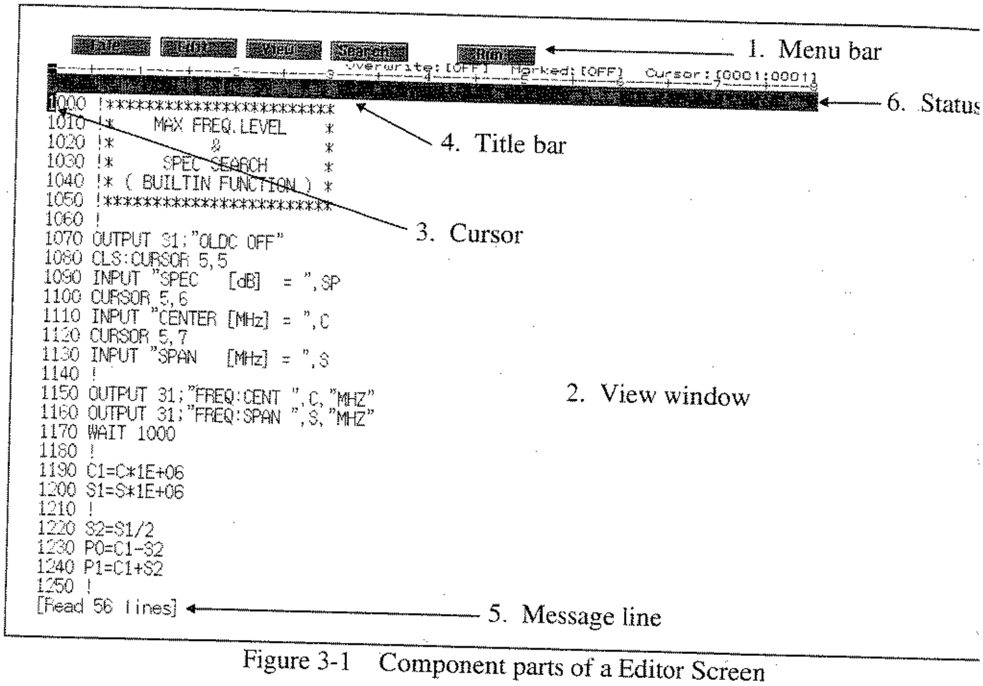
- Changing to the edit mode
  Press *F12* of the external keyboard.

3.1.2 BASIC Editor Screen

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

In this screen, the measurement trace and BASIC text screen are not displayed.

The component parts of an 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

1.	Menu bar	Shows the name of each menu.
2.	View window	Shows the text of a program.
3.	Cursor	Indicates the position where the text is to be input an The cursor appears in the active window.
4.	Title bar	Shows the buffer name·file name of program and sub For a BASIC program, the buffer name is represente For other subfiles except represented as the same as f For files added by editing, an asterisk (*) is displayec of title bar.
5.	Message line	Shows any errors occuring in the course of editing an information.
6.	Status	Shows the status (Overwrite/Marked/Cursor) of the e

Overwrite: shows Insert/Overwrite mode.

Marker: shows the status of the text block. (Specifies text range)

Cursor: shows the position of the cursor on the scr y.

3.2 Opening Menu and Executing Command

Each command of the editor can be found in the pull-down menus of the menu bar. Figure 3-2 shows one of the pull-down menus. F1 : File menu.

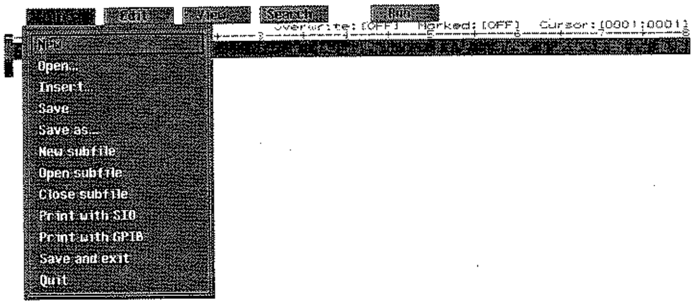


Figure 3-2 F1 : File Menu

The programming environment is designed to make operation as simple as possible.

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

In addition, some commands can be executed directly using their associated shortcut keys without opening the menu. (Refer to 3.2.2.)

For commands followed by an ellipsis (...) there are options associated with the command which need to be specified before it can be executed.

When commands of this kind are executed, a dialog box appears in the view window so that the particular options for that command can be specified.

3.2.1 Executing Commands through key operation

This section describes the method used to open menus using the external keyboard and exec commands.

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

If you press ← or → after a menu has been opened, then the menu to the left or right of th opened.

When a menu is open, its' commands are displayed, and the current selection is indicated i

Press ↑ and ↓, to move the selection up and down.

Press ***Enter***, to execute the selected command.

To cancel the command, press ***Esc*** and the current menu is closed.

When a command followed by an ellipsis (...) is executed, a dialog box appears on the sc items necessary for command execution.

To cancel the command and close the dialog box, press ***Esc***.

3.2.2 Executing Commands using Shortcut keys

This section describes command execution using the shortcut keys.

Using the shortcut keys, the menu commands can be executed using one key stroke.

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

Table 3-1 Shortcut Key Operation

Key operation	Explanation
Shift + Backspace	Cuts the selected area.
Shift + Insert	Pastes the contents of the clip board.
Ctrl + F2	Cuts from the cursor position to the end of line.
Alt + F2	Pastes the contents of the clip board.
Ctrl + F3	Loads the next buffer.
Alt + F3	Shows the **Buffer list ...** dialog box.
Ctrl + F4	Searches for the next target.
Alt + F4	Searches for the next 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 the editing screen and output screen.
Shift + F12	Converts the split window alternately.

NOTE: ***The key operation Shift + Backspace means press the Backspace key while holding the shift key down. (The following are same.)***

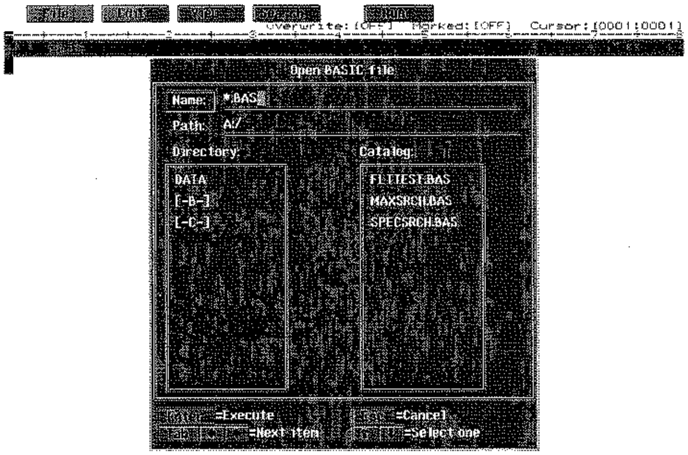
3.3 Using a Dialog Box

1. Function of a dialog box

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

   The dialog box performs the following functions.

   - allowing the user to input file names
   - allowing the user to set options



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

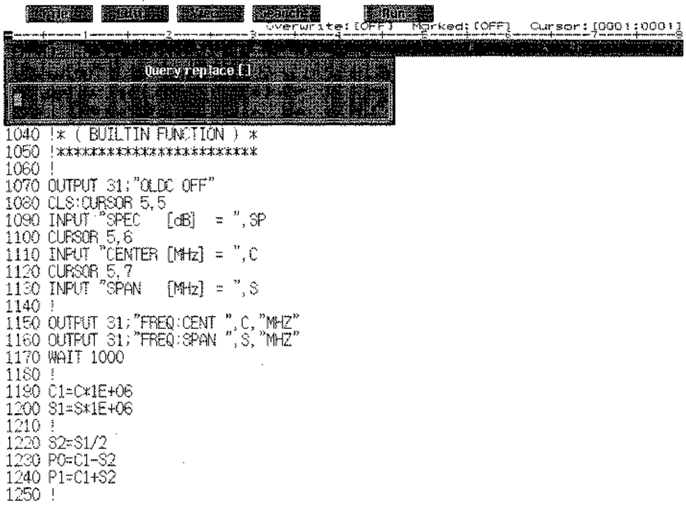
Name: *.BAS

Path: A:/

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

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

Figure 3-3 Dialog Box for the **Open ...** Command



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

```
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 for the **Replace ...** Command

2. Moving the cursor

   To move the cursor to an item in a dialog box:

   - Press ***Tab*** to move the cursor to the next item.
   - Press ***Tab*** while 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 the current directory.

     Inputs a new path name with the Name field or selects a suitable directory with the Directory box or changes the path display.

   - Text input field:

     Shows the entered text.

   - List box:

     Shows a catalog of directory and files etc.

   - Command button :

     Shows the catalog of keys necessary for command execution.

     Presses the corresponding keys.

     To execute the command, press ***Enter***.

3.4 Using the Message Line

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

It is used to confirm command execution or cancellation, or to notify the user when errors c

The message line is erased when the next operation is performed.

3.5 Using the Window

The part showing the loaded program is called " View window ". The view window can perf editing.

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

3.5.1 Function of Each Window

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

To split the view window, execute **Split window** command using F3 : View menu.

To return the screen to normal, execute the command again.

3.5.2 Changing the Active Window

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

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

3.5.3 Changing the Window Size

The window size can be enlarged or reduced according to the number of lines displayed.
Also, a window can be displayed on the full screen.

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

Key operation	Explanation
Ctrl + PageUp	Enlarges the active window by one line.
Ctrl + PageDown	Reduces the active window by one line.
Shift + F12	Causes the active window to be full screen.

3.5.4 Scrolling the Active Window

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

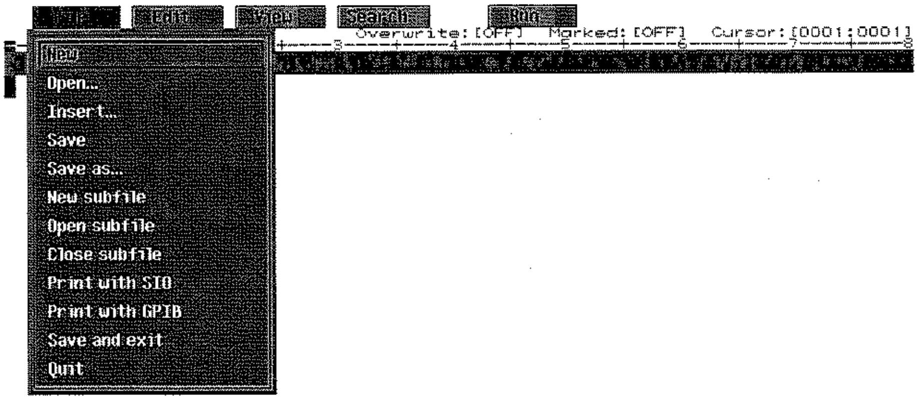
Press the direction keys to move the cursor to the end of the screen, to start scrolling.
However, scrolling to the left or right can be performed only on the current cursor line.

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 one 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.

4. F1: File Menu

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

Executing the commands of F1: File menu, allows you to 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.



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

- **New** command

  Removes the BASIC program which was previously loaded.

  Use it when you start to write a new program.
- **Open ...** command

  Loads a program which has been saved to a floppy disk.

  Select the file from the files listed in the dialog box or the catalog of directory.
- **Insert ...** command

  Merges the contents of two files into one file.
- **Save** command

  Writes the contents of a file displayed in an active window onto 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 normal text file. This 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 memory.
- **Print with SIO** command

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

- **Print with GPIB** command

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

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

  Closes the editor.

4.1 New Command

The **New** command of the F1: File menu deletes all of the BASIC program that has been l so that a entirely new program can be entered.
When a program in memory is not saved, the following message is displayed in the messa

Discard changes [y / n] ?

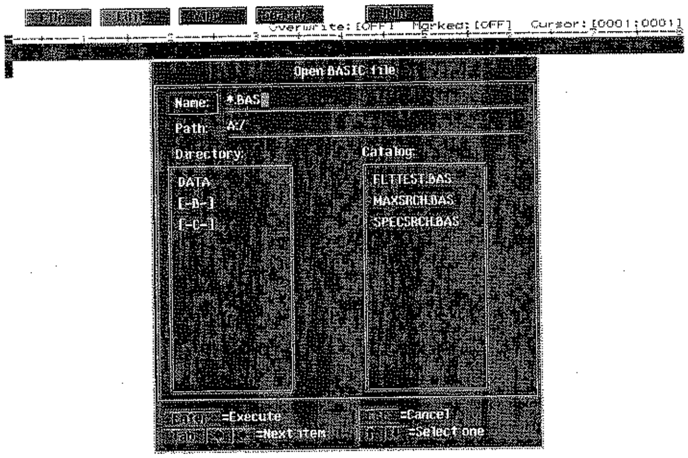
Pressing ***Y*** releases the program from memory without saving it to a floppy disk, and any edited are displayed in the active window.

Pressing ***N*** starts to edit another buffer.

Execute the **New** command after saving it to a floppy disk with the **Save** command or the mand.

4.2 Open ... Command

The **Open ...** command is used to load a program which has been saved to a floppy disk.
When this command is executed, a dialog box appears and displays the catalog of files with the extension ".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.



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

4.2.1 Specifying Files

The catalogs of directories and the accessible drives are displayed in the Directory box.
The file catalog is displayed in the Catalog box of this dialog box.
When it can not be displayed on one screen, scroll up and down with ↑ and ↓, separately.
There are the 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 name displayed in the Name field, press ***Backspace*** to delete characters one at a time.

- Selecting from the Catalog box

  Keep pressing ***Tab*** to move the cursor to the Catalog box.

  Use ↑ and ↓, to move the highlight indication over to the file to be loaded, then press ***Enter***.

4.2.2 Catalog display of Directory Content

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

All subdirectories on the selected directory and the catalog of files with the extension ".BAS" tory are displayed.

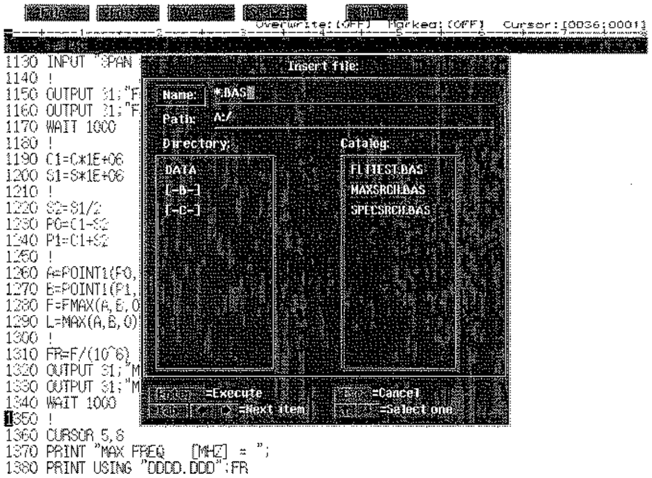
When ***Enter*** is pressed, the file is loaded in the memory.

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

- Display all the files in current directory.
  Input * in the Name field and then press ***Enter***.
- Display the root directory file of floppy disk.
  Input A:/ in the Path field and then press ***Enter***.
- Display all the files in subdirectory called SUB.
  In the Directory box, display the name SUB in highlight and press ***Enter***.
  Then, input * in the Name field and press ***Enter***.
- Display files from the previous directory.
  In Directory box, display in highlight and then press ***Enter***.
- Display files whose name is composed of six characters.
  Input ?????? in the Name field, then press ***Enter***.
- Display files whose name begins with B.
  Input B* in the Name field, then press ***Enter***.
- Display files with the extension DAT.
  Input *.DAT in the Name field, then press ***Enter***.
- Display files with an extension composed of two characters.
  Input *.?? in the Name field, then press ***Enter***.
- Display files with the last character of its' extension between B and FEEEE.
  Input *.*[B-F] in the Name field, then press ***Enter***.

4.3 Insert ... Command

This command is used to insert the contents of other files into a working file at the position of the cursor. When the **Insert ...** command is executed, a dialog box as shown in Figure 4-2 appears.
The method of using this dialog box is exactly the same as that of 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 F0=C1-S2
1240 F1=C1+S2
1250 !
1260 A=POINT1(F0,
1270 B=POINT1(F1,
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, SPECSRCH.BAS

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

4.4 Save Command

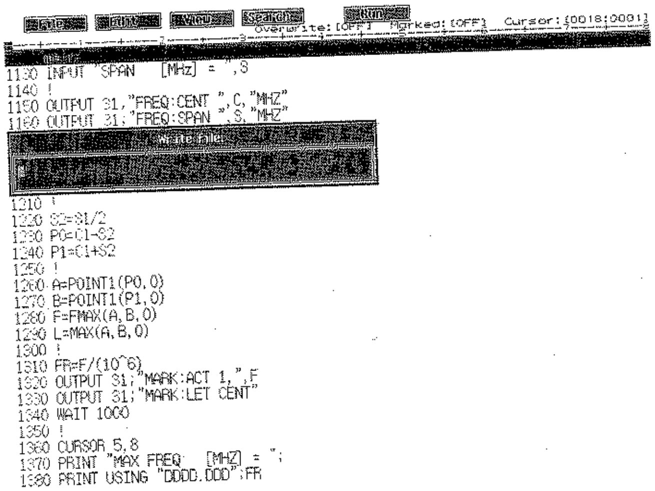
The **Save** command is used to save the contents of the operating file to a floppy disk file.

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 currently 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 be When a new name is specified to a file, the old file remains in the floppy disk with the original n

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.



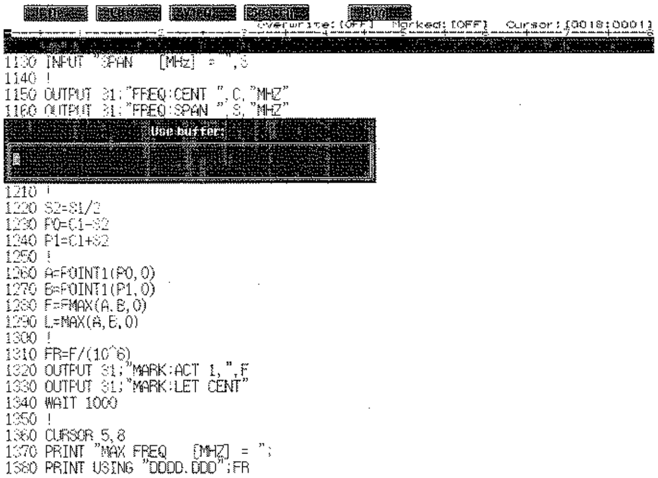
```
1130 INPUT "SPAN   [MHz] = ",S
1140 !
1150 OUTPUT 31,"FREQ:CENT ",C,"MHZ"
1160 OUTPUT 31;"FREQ:SPAN ",S,"MHZ"
1210 !
1220 S2=S1/2
1230 P0=C1-S2
1240 P1=C1+S2
1250 !
1260 A=POINT1(P0,0)
1270 B=POINT1(P1,0)
1280 F=FMAX(A,B,0)
1290 L=MAX(A,B,0)
1300 !
1310 FR=F/(10^6)
1320 OUTPUT 31;"MARK:ACT 1,",F
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 ordinary text files. 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.



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

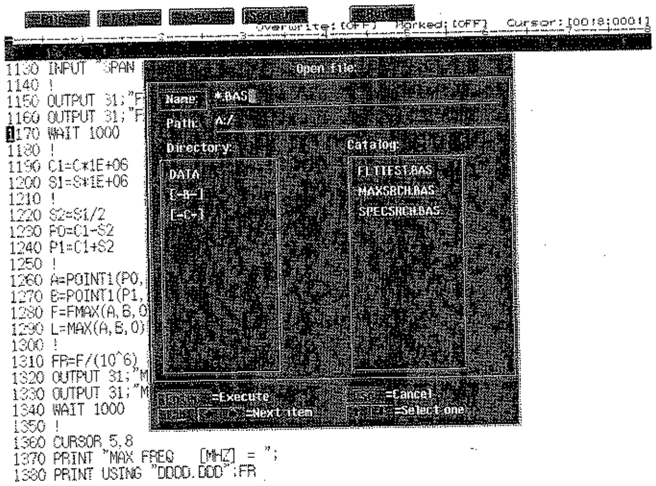
For program files, only one file can be loaded. But for subfiles, more than one can be loaded at the same time. The subfile can display the file in the course of editing through the buffer name using **Buffer list ...** of F3 :View window, or by displaying the file information.

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 text files from a floppy disk. A file loaded with this 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.



Name: *.BAS

Path: A:/

DATA

[-B-]

[-C-]

FLTTEST.BAS

MAXSRCH.BAS

SPECSRCH.BAS

```
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
```

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 program files, only one file can be loaded. But for subfiles, more than one can be loaded. can display the file in the course of editing by using the buffer name with the **Buffer list ...** of F dow, or by displaying the file information.

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

4.8 Close subfile Command

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

Discard changes [y / n] ?

Pressing ***Y*** erases the current program from memory without saving it to a floppy disk and dis file which is being edited in the active window.

Pressing ***N***, moves it an editing buffer.

Execute the **Close subfile** command after saving the file to a floppy disk with **Save** command command.

4.9 Print with SIO Command

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

When using the **Print with SIO** command, connect the printer with the RS-232 port of the network 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 port.

When using the **Print with GPIB** command, connect the printer with the GPIB port of the network analyzer.

4.11 Save and exit Command

The **Save and exit** command is used to save all those programs 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 file whose name is not specified exists, " No file name " is displayed and the screen returns to the original. Specify a name for the file with **Save as ...** command.

4.12 Quit Command

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

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 the files have been saved, execute the **Save and exit** command.

5. Basic Operation of Editor

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

The following items are explained.

- Entering text and moving the cursor
- Deleting and inserting text
- Moving and copying blocks of text
- Searching and replacing characters, words and statements
- A 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 the cursor.
In the overwrite mode, the character at the cursor position is replaced by the entered character.

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

The current mode can be checked at the status line.
The following message should appear:

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

5.2 Selecting Text

When you operate a text block using 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 ***Ctrl*** and hold it down, then press the ***Space*** key to set the mark.

   When the mark is set, the Marked : [ ] in the status line is shown as ON.

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

Once the Mark is set, the texts from where the mark is set to the text where the cursor is currently is selected.

5.3 Indenting Text

You can indent text to make it easier to read.
To indent a text, enter a ***Space*** or ***Tab*** at the beginning of a line. The indent is set at the column where the space is added. 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 in an active window can be performed by the simple combined operations of following is a catalog of edit command.

Table 5-1 Catalog of Edit Command (1 of 2)

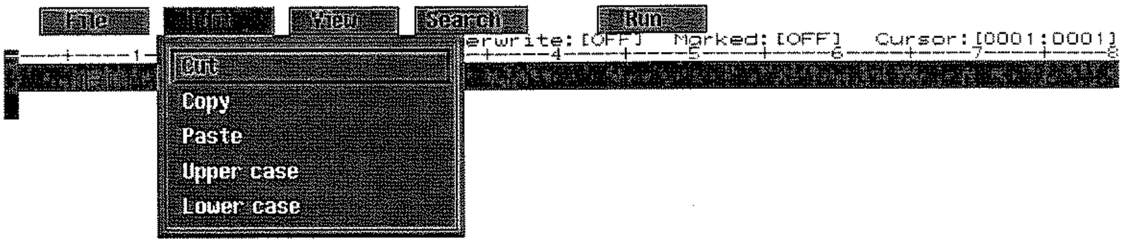
	Key operation	Explanation
Moving Cursor	←	Moves one character left.
	→	Moves one character right.
	Ctrl + ←	Moves one word left.
	Ctrl + →	Moves one word right.
	↑	Moves one line up.
	↓	Moves one line down.
	Home	Moves to the beginning of the file.
	End	Moves to the end of the file.
	Ctrl + ***Home***	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 a mark at the cursor position.
	Alt + ***Space***	Switches between the positions of mark and cursor.
Insert	***Enter***	Inserts a change line.
	Insert	Inserts one space.
	Ctrl + ***Insert***	Inserts a blank line.
	Shift + ***Insert***	Inserts the contents of clip board.
	Alt + ***Insert***	Switches between Insert/Overwrite modes.
Delete	***Backspace***	Deletes the character to the left of the cursor.
	Delete	Deletes the character at the cursor position.
	Ctrl + ***Backspace***	Deletes the word to the left of the cursor position.
	Ctrl + ***Delete***	Deletes the word to the right of the cursor position.
	Shift + ***Backspace***	Deletes the selected text.
	Shift + ***Delete***	Deletes from the cursor to the end of the line.

Table 5-1 Catalog of Edit Command (2 of 2)

	Key operation	Explanation
Scroll	***Ctrl + ↑***	Scrolls one line upward.
	Ctrl + ↓	Scrolls one line downward.
	PageUp	Scrolls one page upward.
	PageDown	Scrolls one page downward.
	Ctrl + Pageup	Enlarges a window by one line.
	Ctrl + Pagedown	Reduces a window by one line.

6. F2: Edit menu

Commands used to write or change between programs and texts are found in F2 : Edit menu.
Using these commands, you can cut, copy and paste text.
It is necessary to select the text to be edited in advance. For instructions on how to do this, refer to " 5.2 Selecting Text ".



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

- **Cut** command

  Used to remove the selected text and place it in the clip board.
- **Copy** command

  Used to place a copy of the selected text to the clip board.
- **Paste** command

  Used to insert the contents of the clip board in the text at the cursor position.
- **Upper case** command

  Used to change the selected text to upper case characters.
- **Lower case** command

  Used to change the selected text to lower case characters.

6.1 Clip Board

The clip board is used to store text which has been cut or copied temporarily,
Text which has been cut or copied with the corresponding commands from the Edit menu is stored in the clip board and remains there until a new selection of text is cut or copied. Only one block of text at a time can be stored in the clip board. The stored text can be inserted into the text at the location of the cursor using the **Paste** command. There is no limit on the number of times the stored text can be pasted.

6.2 Cut Command

This command is used to cut selected text from the screen and place it in the clip board. To execute the **Cut** command, first select the text to be cut.
When **Cut** and **Paste** commands are used, blocks of text can be moved easily.

- Moving lines and blocks of text

    1. Select the text to be moved. (Refer to 5.2.)
    2. Execute the **Cut** command from the F2 : Edit menu.
    3. Move the cursor to the position where the cut text is to be inserted.
    4. Execute the **Paste** command from the F2 : Edit menu.

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

6.3 Copy Command

This command is used to copy text just as it is, and place it in the clip board. To execute the **Copy** command, first select the text to be copied.

When **Copy** and **Paste** commands are used, part of the program or the whole program can be copied.

- Copying part of the program or the whole program

    1. Select the text to be copied. (Refer to 5.2.)
    2. Execute the **Copy** command from the F2 : Edit menu.
    3. Move the cursor to the position where the copied text is to be inserted.
    4. Execute the **Paste** command from the F2 : Edit menu.
    5. Correct the copied text as required.

6.4 Paste Command

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

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

6.5 Upper case Command

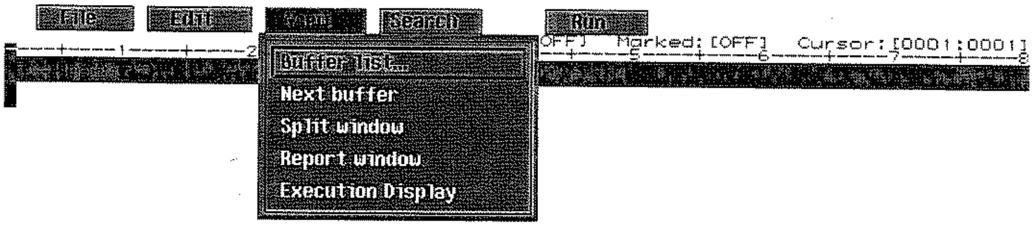
This command is used to change the alphabetic characters of the selected text to upper case characters.
To execute the **Upper case** command, first select the text to be changed.

6.6 Lower case Command

- This command is used to change the alphabetic characters of the selected text to the lower case characters.
  To execute the **Lower case** command, first select the text to be changed.

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.



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

- **Buffer list ...** command

  Used to look at programs and the 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 in the buffer in an active window.

- **Split window** command

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

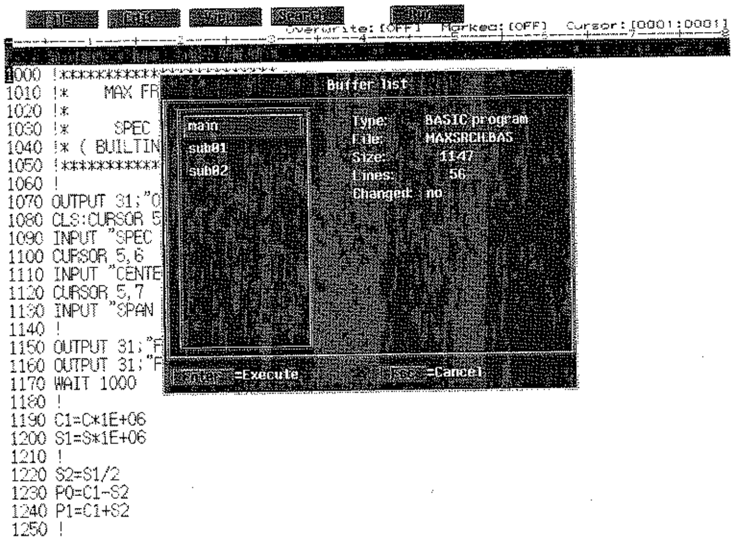
- **Execution display** command

  Used to convert the editor screen and measurement screen.

7.1 Buffer list ... Command

When using the **Buffer list ...** command, you can see a catalog of the files that have been loaded and select a 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 !
```

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 files

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

2. Execute one of the following operations after displaying the target buffer name in highlight with the ←, →, ↑ or ↓ keys.

3. To display the item highlighted in the active window, press the ***Enter*** key.

4. To cancel this operation, press the ***Esc*** key.

- Short cut key : ***Alt + F3***

7.2 Next buffer Command

When more than one file has been loaded into the edit memory, executing the **Next buffer** command displays the next buffer in the active window in an alphabetical sequence. When there are no files 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 split, you can operate with two parts of a file shown at the same time.

- Operating with more than one file shown at the same time

  1. Split the view window using the **Split window** command.
  2. A file is displayed in the 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 is resident.)

- Operating with the view window split by the **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 from the F3 : View menu again, the active window will be enlarged to fill the entire view window and the other window will close.

- Changing the window size

  To change the window size, make the window active and then do the following.

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

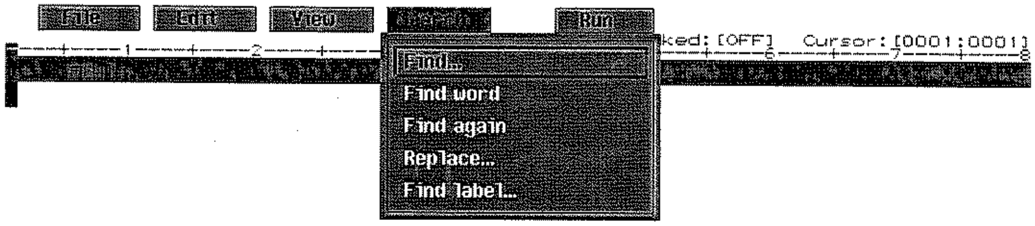
7.4 Execution display Command

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 swtched.

- Shortcut key : ***F12***

8. F4: Search Menu

Searching for a certain character string and replacing it with something else can be done by using the commands in the F4 : Search menu. You can use the search function to quickly move to the text you are searching for. In addition, when you want to change a variable name, you can do this easily using the replace function.



- **Find ...** command

  Used to search for a specified character string and move the cursor to the position directly after that string.

- **Find word** command

  Used to search for a character string that matches the word at the cursor position, and move the cursor to the position directly after that string.

- **Find again** command

  Used to search for the same character string again.

- **Replace ...** command

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

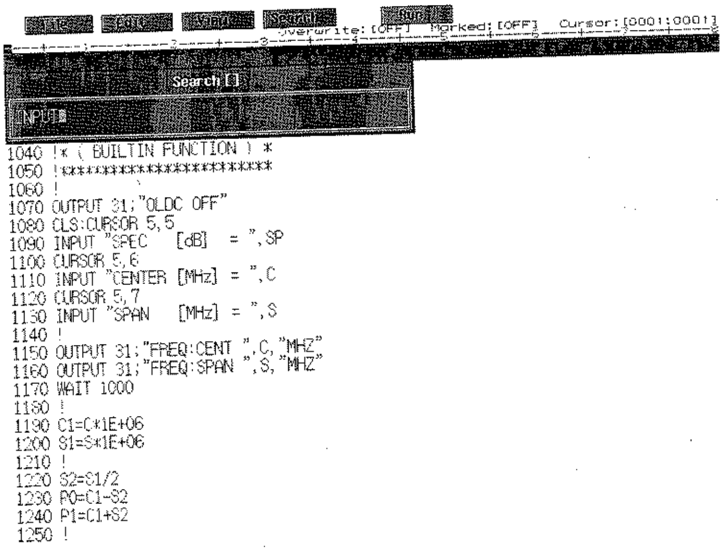
- **Find label ...** command

  Used to search for the specified line label.

8.1 Find ... Command

This command is used to search for the specified character string from the current cursor position. When the target character string is found, the cursor is moved to a position after it.
Character strings may be composed of any character (including spaces etc.).

When the **Find ...** command is executed, a dialog box as shown in Figure 8-1 appears. Enter the text you want to search for in the dialog 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 search function searches the entire file from the position of the cursor.

- Searching text

  1. Execute the **Find ...** command from the F4: Search menu.
  2. Enter the text you want to seach for in the dialog box.

When the specified character string is found, the cursor is moved to a position directly after the text.

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 is removed when the next operation is executed.

8.2 Find word Command

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

- Using the **Find word** command

1. Move the cursor to the word you want to select. The length of the selected word should be less than one line.
2. Execute the **Find word** command from the F4: Search menu.

8.3 Find again Command

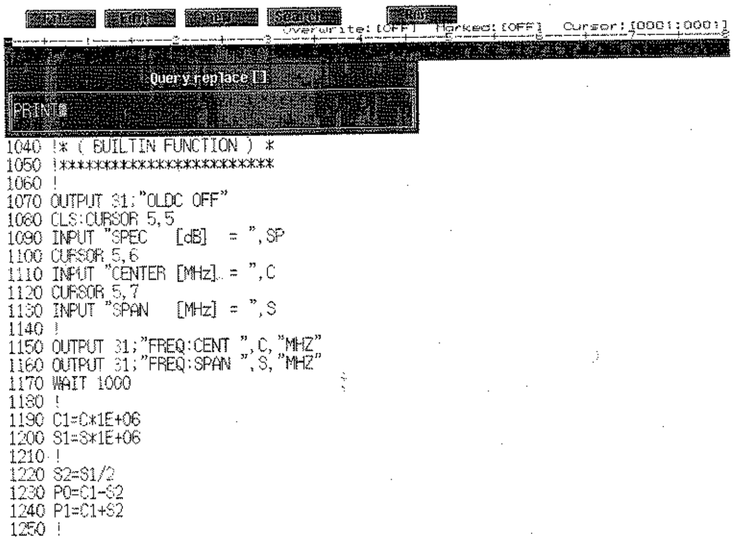
This command is used to search for the same text again. Text specified by the **Find ...** , Find word and **Replace ...** commands can be searched for repeatedly using this command.
When a character string is not yet been entered, the **Find again** command can not be used.

• Shortcut keys:	***Ctrl + F4***	(Backwards from the cursor position.)
	Alt + F4	(Forwards from the cursor position.)

8.4 Replace ... Command

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

When the **Replace...** command is executed, a dialog box appears as shown in Figure 8-2.
Enter the text you want to replace in this dialog box.



```
PRINT
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 from the F4: Search menu.
2. Enter the text you want to replaced.
3. Enter the text you want to replace the entered text with.
4. Press ***Enter*** to replace the text.

5. Once the character string you want replaced is found, the following message is displayed in the message line.

   Replace ' find-text' with ' replace-text'?

   Press ***Y*** to replace the text.

   Press ***N*** to cancel the operation for this particular text string and move to the next text string.

   The chart below lists a number of keys which affect the operation of this command. Pushing any of the keys listed causes the corresponding effect.

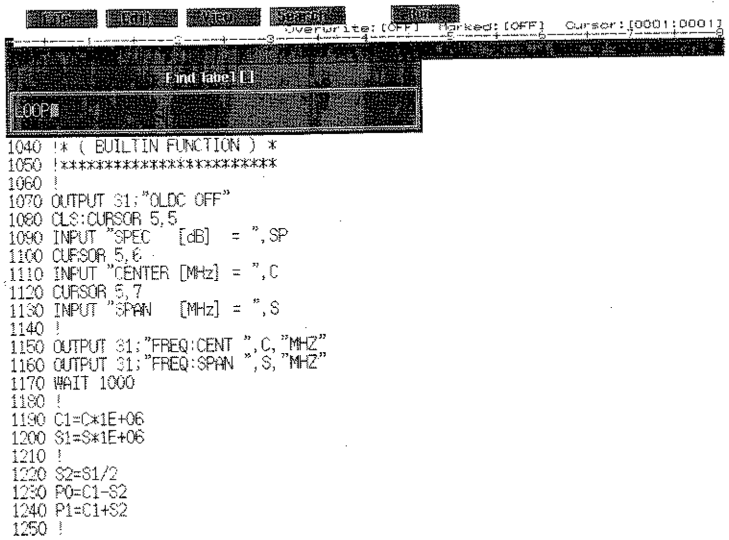
Key operation	Explanation
Esc	Stops searching with moved to the searched character string.
!	Replaces the remainder in a batch.
?	Displays a list of the key operations.
.	Stops searching and returns the cursor to the position from where the search started.
Y	Replaces the text and searches for the next example of the same text.
N	Searchs for the next example without replacing the text string.

When the character string being searched for is not found, the message "Not found" is displayed in the message line. This message is removed when the next operation is executed.

8.5 Find label ... Command

This command is used to search for a line label in a BASIC program.
The line label are preceded by an asterisk (*) so to search for label, add an asterisk to the text you input in the Find text input field.

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



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

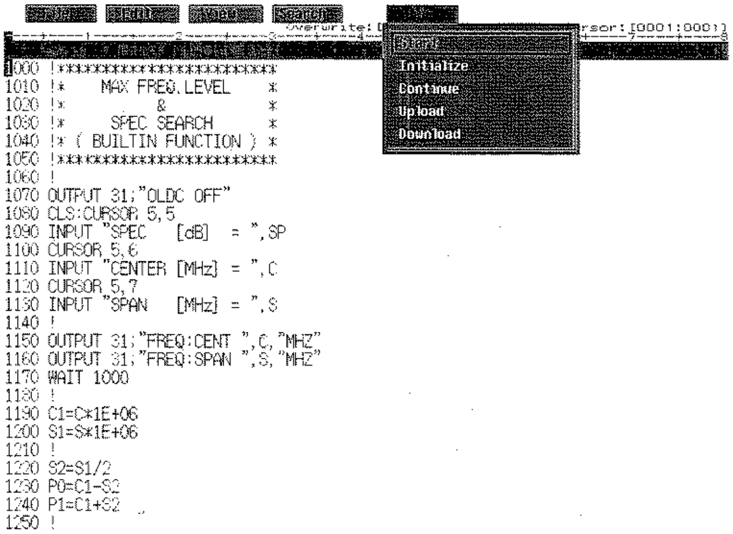
```
LOOP
```

```
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 commands related to the BASIC programming environment are listed in F5: Run menu. These commands allow you to upload or download programs from 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 the program.

- **Initialize** command

  Resets all numeric variables to 0.

- **Continue** command

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

- **Upload** command

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

- **Download** command

  Loads a program being edited to basic memory.

9.1 Start Command

The **Start** command is used to run programs.
It loads the program into basic memory first when a program which is being editied has not been loaded yet or when no changes have been made.
When there is no line number at the beginning line of a program, the line number is added automatically.

To pause a program while it is running, press ***Pause***.

To restart the program, select the **Continue** command from the F5: Run menu.

To run the program from the beginning, use the **Start** command.

The screen changes to the measurement screen when the program starts, 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 has been debugged.
This command resets all the values of variables in a program to 0.
This command is used to prepare for 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.
If a program has been paused, using this command continues the program from the statement where it was interupted. If the program has not yet been interupted, it is 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 into basic memory without executing it.
The programs being edited which have no line number are loaded into basic memory with line numbers automatically (Line numbers can not be directly added to the program by editing.)

10. Automatic Measurement on the Network Analyzer

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

NOTE: ***The program presented in this chapter is an example for use with the R3752/53H Series. When it is used for the R3764/65/66/67H Series, R3765/67G Series or R3754 Series, this program needs to be changed to match the initial setting, frequency range, etc. for the particular model as necessary.***

10.1 Program with OUTPUT and ENTER Commands

10.1.1 Executing the 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 the R3752H Series.)

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 in Example 10-1 is executed using the RUN (BASIC command), a marker is shown. The frequency and level at that position are displayed.

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

This program shows an example using ceramic bandpass filter as a DUT (Device Under Test), with a center frequency of 380MHz.

In R3752H Series, there is no GPIB command to deal with the marker. Therefore, a built-in function is used when trace analysis is performed with this Series. For details on the built-in functions, refer to 10.2. The program is as follows (This program also can be used for R3752H Series.).

Example 10-2 Program showing the OUTPUT and ENTER Commands (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
```

Program Explanation :

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

Explanation of Example 10-1	
100 to 130	Comment line.
140	Sets the GPIB command mode for the network analyzer.
150	Sets the first marker at 380MHz.
160	The first marker 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 are necessary here, it assigns the frequency to F, the level to L and the phase to P.)
180	Displays the variable F on the screen using the PRINT statement.
190	Displays the variable L on the screen using the PRINT statement.
200	Displays the variable P on the screen using the PRINT statement.
210	Ends the program.

This program is carried out under the settings as they are after the power is switched on.

Explanation of program command:

The OUTPUT, ENTER and RUM (or !) commands used for the program 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 strings to the device specified by the device address number.

   The "31" of OUTPUT 31 written in line 140 of this program is the address number which means that the contents is sent to the measuring section of the network analyzer. OLDC OFF (Setting of IEEE488.2-1987 command mode) is executed in the network analyzer. The control of the network analyzer can be performed by using the OUTPUT command and GPIB command.

   Furthermore, other external instruments can also 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 data through the GPIB from the device specified by the device address and then assign that data to the BASIC variables (which may be either numeric or character strings).

In Example 10-1, ENTER is used as a response. Here, it is used in combination with OUTPUT.

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

- Using the OUTPUT command in line 150, this program sets the first marker at 380 MHz for the device address 31 (which represents the network analyzer) in order to display it.
- Using the OUTPUT command in line 160, this program specifies " FETC ?" to the network analyzer in the same way. The question mark after the GPIB command is used when you want to know the setting and measurement values (query of data). In this case, marker data (frequency and level) are prompted.
- Using the ENTER command in line 170, this program receives the marker data and assigns it to the variables F (for frequency), L (for level) and P (for phase).

  The data sent differs according to the GPIB command used. For details, refer to the manual " Programming Manual ".

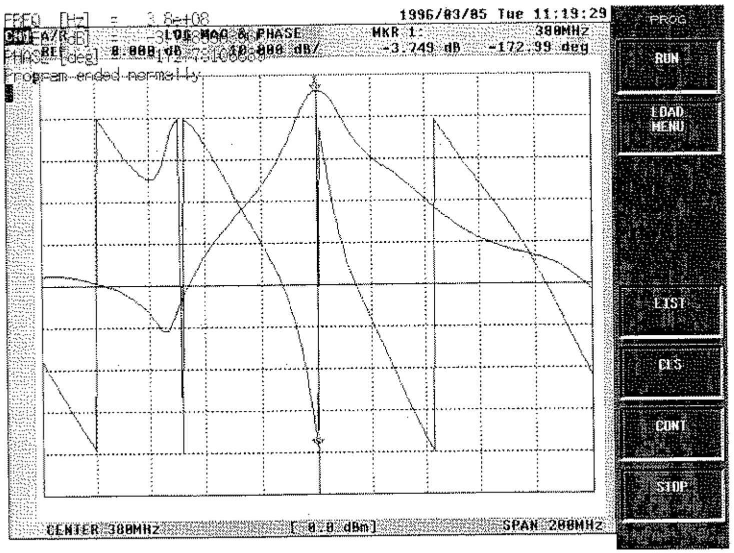
3. REM command

   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 can be substituted with ! (exclamation mark).

$$\left.\begin{array}{l}\text{10 REM PROGRAM1}\\ \text{10 ! PROGRAM1}\end{array}\right\}\text{Same meaning}$$

When this program is used with R3753H Series, it is displayed as shown in Figure 10-1.

The results of this program should look as follows:



```
FREQ  [Hz]  =  3.8e+08
A/R  [dB]  =  -3[illegible]
PHASE [deg]  =  -172.7310668
Program ended normally.
```

Figure 10-1 Program Execution Screen Display

Two traces are displayed.
The R3753H Series measurement FORMAT uses LOG MAG & PHASE traces as the default setting.
The ▼ marker is seen at a slightly higher position from the centre, showing the frequency of 380MHz at this position.
The uppermost line of graph is used as a reference line.
REF 20.000dB is written in the upper left part of the screen, meaning that the reference value at present is 20dB. However, the interval between each division in the 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 for 380MHz shown by the marker is 0.781dB.
Since the level value changes for each scan, this vaule is not fixed. This program's execution result 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 shown in Example 10-1, FREQ is shown in MHz (megahertz) and the values of LEVEL and PHASE are shown with a number that is rounded to three decimal places so that they can be easily understood.

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 is output to the GPIB printer as shown below:

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	Prepares the network analyzer to accept a new command.
160	Sets the first marker of the network analyzer at 380 MHz
170	Prompts to send the first marker data of the network analyzer.
180	Assigns the data sent by line 170 to the variables F, L, P.
190	Command line.
200	In order to show 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. (Does 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 to the third decimal place. (Changes the line.)
240	Outputs LOGMAG [dB] = to the printer. (Does 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 to the third decimal pace. (Changes the line.)
260	Outputs PHASE [deg]= to the printer. (Does 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 to the third decimal place. (Changes the line.)
280	Ends the program.

A new command is used in the program shown in Example 10-3 that is not used in the program in Example 10-1.This is the GPRINT USING command shown in lines 220 to 270.
This command can also be used as GPRINT only.
GPRINT is almost the same as the PRINT command, however the data (variables and character strings enclosed by double quotations) is output to the GPIB board without being displayed on the screen.
When a printer is connected to the GPIB board, the data also can be printed out.
A semicolon added to the end of a line means that the line does not change. The next output follows the last output without changing the line.

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

The PRINT USING command outputs characters and values according to the image specification determined by the print format setting. See line 230 of the program in Example 10-3.
"DDDD.DDD"; FR means that the value assigned to FR will be printed to the third decimal place and if the integer part is only three digits, the remaining position should be blank spaces.
"MDDD.DDD" in line 250 and line 270 has the same meaning. When the value assigned to the variable L or P is negative, a minus sign is added to the front of the value, and when the value is positive, a blank space is printed instead.
In this example, the execution result is output to the 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 allow you to compute and analyze the captured measurement data at high speeds. For these functions, it is not necessary to use the commands OUTPUT and ENTER to transfer the data as it is done previously. Since this operation can be performed directly at a high speed using the built-in CPU, the processing time is greatly reduced.

The marker analysis function is not available for the R3752H Series. To analyze trace data with this series, you need to create a program that can the built-in functions.

10.2.1 Using Built-in Functions

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

For instance, the format of the built-in function CVALUE (used to specify a frequency value and then evaluate the measurement response value (level) of that frequency.) is as follows.

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

The following program, which evaluates the level at the frequency for the DUT (device under test) connected to CH1, is given as an example of this function.

Example 10-4 Program using the 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 in line 110 in which specified frequency is entered as A, and CH is set to 0 because the DUT is connected to CH1.

The resulting level value "L" is then displayed on the screen by the PRINT command in line 120. This

demonstrates how a built-in function can be incorporated into an expression so it can be used for normal variable computation.
For the detail on the built-in functions, refer to “4.4 Built-in functions” in the programming manual.

10.2.2 Program with Built-in Functions

Here, a program is created using more than one built-in function.
When the program in Example 10-3 shown below has been rewritten to include built-in functions. The changes occur in lines 150 thru 190, however the program executes in the same way as the original.

Example 10-5 Program using 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 a ceramic BPF of 380 MHz as the DUT, just as the program in Example 10-3 did. The POINT1 function is used in line 160. This function specifies the frequency, and calculates the address point of the measurement frequency which is nearest to the specified frequency. (Address points are used to specify an analysis range for the measurement data and the position where the data is being measured. The value range is 0 to 1200.) Here, the frequency is changed to address point using the POINT1 function at the beginning.
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 to CVALUE and POINT1.
In line 170 the FREQ function is used to determine the frequency using the obtained address and this value is assigned as F. Since the frequency is 380MHz, it should not have been used. However, it is used here in order to describe how to obtain the frequency value from the address point. The writing format used for the FREQ function is as follows.

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

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

VALUE (Address point, Analysis channel)

In line 190, the phase is determined as P.

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

With a short program such as this, there is no discernable difference in the execution speed, however a long and complicated program using built-in functions will be operate at a higher speed.
For details on the built-in functions and the analysis channel, refer to "4.4 Built-in Functions" in the programming manual.

10.2.3 Program to Judge Measurement Value

The programs described up to this point are used to evaluate amplitude and phase by assigning a specified frequency value into the programs directly.
In the next program, the center frequency (CENTER) and span value (SPAN) are entered using the INPUT command, and then the frequency and amplitude for the maximum amplitude point of the center frequency range are calculated Finally, whether the amplitude value at the maximum amplitude point has reached the standard value or not is determined.

Explanation of decision program:

The following standard values are initially assigned to the variables using the INPUT command. The standard values are used to judge which levels are necessary for CENTER, SPAN and amplitude values.

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

When entering these values, input the SPEC value just as it is. Since the unit for CENTER and SPAN is MHz, you don't need to type MHz. For example, input 150 when the value is 150MHz.
After the necessary values have been 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 a new GPIB command mode. The GPIB program code FREQ: CENT and FREQ: SPAN are used for the frequency setting of CENTER and SPAN.

When OLDC is turned ON, the command names used in the older R3751 and R3762 Series can be used however the new command mode enables the program to be read with ease.

In the following, the values for CENTER and SPAN (represented by C and S) are changed from MHz to Hz.

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

The CENTER value (in Hz) is assigned to C1 and the SPAN value, to 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 using 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 points that can be used with the 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 determined 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 frequency of the maximum amplitude point is assigned to variable F and the amplitude value, to L . These values are then displayed on the screen using the PRINT command.

```
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.
The unit used for F (maximum frequency) is converted from Hz to MHz and then the values are displayed in the following sequence; maximum frequency, level and SPEC (standard value). These values are displayed up to the third decimal place, and the PRINT USING command is used to to align the digits.
Finally, the level at maximum frequency is compared with the input standard value (SPEC). When the maximum frequency is acceptable, "SPEC OK !!" is displayed by the PRINT command, and when it is unacceptable, "SPEC NG !!" is displayed.

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

The full program is shown as follows.

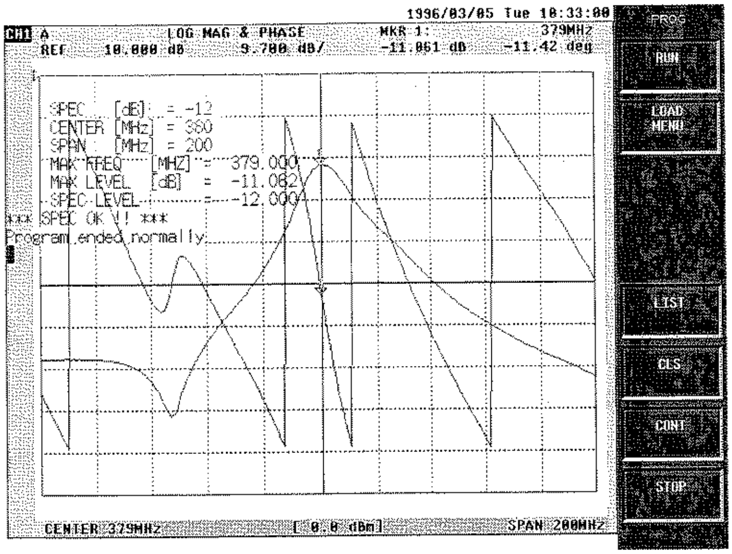
Example 10-6 Program for Deciding the Measurement Value

```
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 executing the program in Example 10-6, the SPEC value is prompted first.
Since a ceramic filter of 380MHz is used as a DUT, the measurement is performed with the following settings; SPEC level (SPEC) -10dB, CENTER 380MHz and SPAN 200MHz.

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

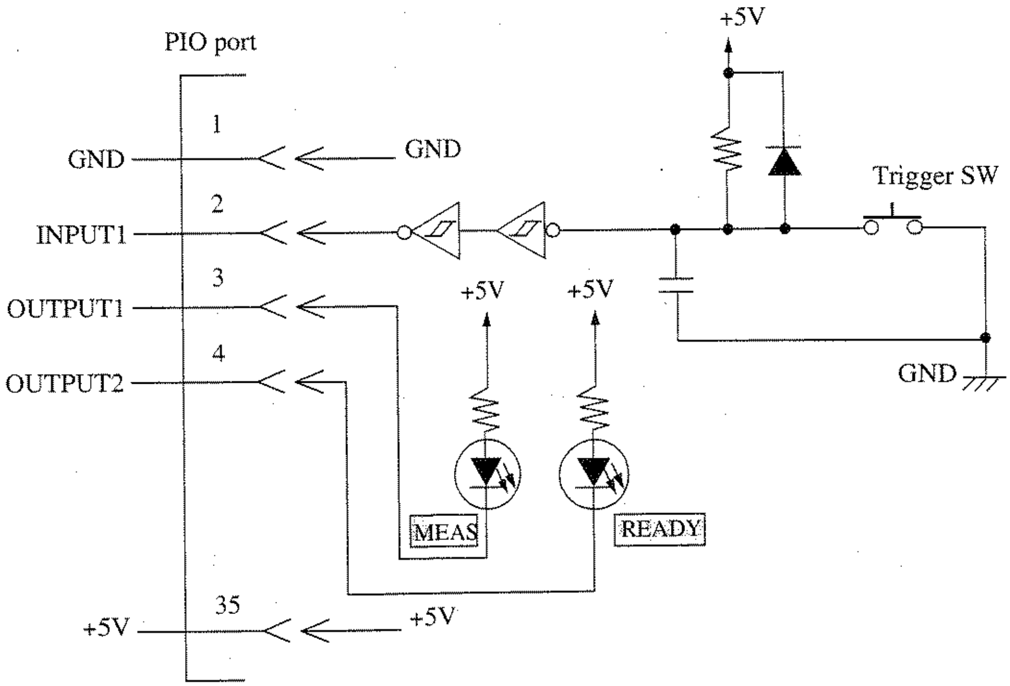
Execution result:



```
SPEC    [dB]  = -12
CENTER [MHz]  = 380
SPAN   [MHz]  = 200
MAX FREQ    [MHZ]  =  379.000
MAX LEVEL   [dB]   =  -11.082
SPEC LEVEL         =  -12.000
*** SPEC OK !! ***
Program ended normally
```

Figure 10-2   Execution Result of Program Output to Parallel I/O Port

In the program in Example 10-6, the result is displayed as "OK !!" or "NG !!" using the PRINT command, but here the result is output through the parallel I/O port. Using INPUT 1 (External input) from the parallel I/O port, the program is created to start the measurement with the trigger switch.
The circuit diagram shown below is an example of this kind.



Circuit example: When the program is executed by trigger switch

Creating the Program:

1. The port mode setting 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 the port mode is set, since the parallel I/O port is used.
  Here, all of A, B, C and D ports are set to output ports.

- Lines 172 and 173 reset OUTPUT1 and OUTPUT2. That is, they turn off the LED (OUTPUT1) being measured and LED (OUTPUT2) to be measured.

2. A program is added between lines 260 and 270 to pause the measurement until the appropriate 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 set to turn on the LED. This indicates that the analyser is READY to start measurement, as indicated by the LED.

- The WAIT in line 263 is used to provide a short interval for turning on the switch (WAIT time : msec ; 0 to 65535)

- In line 264, A = 0 until the switch is pressed. Line 264 will continually loop to line 262 until A = 1.
  If the switch is pressed, 1 is assigned to A according to the specification ENTER 34; A (External input) in line 262, and the operation proceeds to line 265.

- In line 265, the OUTPUT2 LED is turned off. This means the switch input has been received and the system is no longer in the waiting state.
  If the switch is pressed, the LEDs of OUTPUT1 and OUTPUT2 are lit.
  The LED of OUTPUT2 is turned off by line 265, while OUTPUT1 is constantly lit as it has not been otherwise specified. This indicates the measurement is in progress.

3. The code used to turned off the LED when the measurement stops, has been added between lines 430 and 440.

```
431 OUTPUT  35;80
```

- In line 431, the OUTPUT1 LED is reset and the LED is turned off

4. The I/O port that is used depends on the SPEC value that results (either OK or NG). OK means that the output is sent to port A and NG means that the output is sent to port B. The line between 530 and 540 contains the line used for a result of OK and the line between 590 and 600 contains the line used for a result of NG.

```
531 OUTPUT 33;1
591 OUTPUT 34;1
```

- When the judgement is OK, 1 is output to port A. When it is NG, 1 is output to port B.
  Installing a LED, to No.5 (1 of port A) and another to No.13 (1 of port B) of the parallel I/O connector. The result (either OK or NG) can be determined from which LED lights up.

The program is shown in Example 10-7.

Example 10-7 Program for Deciding the 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
173 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 !
```

(Cont'd)

```
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 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 !! ***"
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
```

11. Examples of waveform Analysis Programs

This chapter gives instructions on how to use the built-in functions and shows examples of programs used for waveform analysis.

NOTE: ***The programs presented in this chapter are examples used for the R3752/53H Series. To use them with the R3764/65/66/67H Series, R3765/67G Series or R3754 Series, the initial settings and frequency range need to be changed accordingly.***

11.1 MAX and MIN Level automatic Measurement Program

An example of a program used to measure the maximum and minimum level values automatically using the built-in functions MAX and MIN is shown below.

Example 11-1 Automatic Measurement Program using MAX and MIN level (1 of 2)

```
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;"OLDC 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 0DBM"
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
```

11.1 MAX and MIN Level automatic Measurement Program

(2 of 2)

```
1410      PRINT "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 for the maximum response value, while the MIN function searches for the minimum response value.
These functions are also used to evaluate the response value of resonance points and anti-resonance points.
Example 11-1 is explained below.

Program Explanation of Example 11-1 (1 of 2)

1000 to 1040	Comment lines
1050	Main routine (MAIN) label
1060	Calls out the initial setting routine SETUP.
1070	Calls out the calibration routine CAL.
·1080	Clears the screen.
1090	Measurement loop routine (MEAS_LOOP) label
1100	Calls out the measurement routine MEAS.
1110	Calls out the display routine RESULT.
1120	Measurement loop.
1130 to 1140	Comment lines.
1150	Initial setting routine (SETUP) label
1160	Releases IEEE 488.1-1987 command mode.
1170	Sets the active channel to CH1, input port to A/R and the 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.

Program Explanation of Example 11-1 (2 of 2)

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	Exits the initial setup routine.
1270 to 1280	Comment lines.
1290	Calibration routine (CAL) lable
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 for input.
1340	Executes calibration.
1350	Waits untill the calibration has completed.
1360	Exits from the calibration routine.
1370 to 1380	Comment lines.
1390	Measurement routine (MEAS) lable
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 for input.
1440	Comment line.
1450	Gets the maximum value.
1460	Gets the minimum value.
1470	Exits from the measurement routine.
1480 to 1490	Command lines.
1500	Display routine (RESULTS) lable
1510	Clears the screen.
1520 to 1540	Comment line.
1550 to 1570	Moves the cursor and displays the minimum value.
1580	Exits from the display routine.

11.2 Ceramic Filter Automatic Measurement Program

An example of a program used to evaluate the insertion loss and the frequency of 3dB bandwidths is shown in Example 11-2.

Example 11-2 Ceramic Filter Automatic Measurement Program (1 of 2)

```
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)
```

(2 of 2)

```
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 Example 11-2 is used to evaluate the bandwidth. It searches those points that attenuated only for the specified attenuation level at the specified frequency, and then evaluates the bandwidth .

Searching starts from the specified address point and moves outside.

In this program, the maximum response frequency is evaluated using the FMAX function, and a bandwidth 3dB down is evaluated from this frequency using the CBND function.

Program Explanation of Example 11-2 (1 of 2)	
1000 to 1040	Comment lines
1050	Main routine (MAIN) lable
1060	Calls out the initial setting routine SETUP.
1070	Calls out calibration routine CAL.
1080	Clears the screen.
1090	Measurement repetition loop (MEAS_LOOP) lable
1100	Calls out the measurement routine MEAS.
1110	Calls out the display routine RESULT.
1120	Measurement loop.
1130 to 1140	Comment lines.
1150	Initial setting routine (SETUP) lable
1160	Releases the IEEE 488.1-1987 command mode.
1170	After presetting the network analyzer, this line switches it to the single scan mode, and sets it to produce SRQ requirement when the scan is ended.
1180	Waits until the the setup has completed.
1190	Sets the frequency scan center to 10KHz.
1200	Sets the frequency scan to 10.7MHz.
1210	Performs a serial poll and drops the RSV bit.
1220	Exits from the initial setting routine.
1230	Comment line.
1240	Calibration routine (CAL) lable
1250	Moves the cursor

Program Explanation of Example 11-2 (2 of 2)	
1260	Displays the message “ CONNECT [THROUGH] ” .
1270	Moves the cursor.
1280	Displays the message “ IF OK THEN PRESS ‘ ENT ’ or ‘X1’ ” and waits for the input.
1290	Executes the calibration.
1300	Waits until the calibration has completed.
1310	Exits from the calibration routine.
1320	Comment line.
1330	Measuring routine (MEAS) lable
1340	Moves the cursor.
1350	Displays the message “ CONNECT DUT” .
1360	Moves the cursor.
1370	Displays the message “ IF OK THEN PRESS ‘ENT’ or ‘X1’ ” and waits for input.
1380	Comment line.
1390	Specifies the branch destination for the service request interruption.
1400	Enables the interruption.
1410	Executes the scan one time.
1420 to 1430	Repeats the interruption waiting loop.
1440	Comment lines.
1450	Branch destination level LPOUT of service request interrupt.
1460	Performs a serial poll and drops the RSV bit.
1470	Disables the interruption.
1480	Calculates the maximum value of level with MAX function and assigns it to variable I_LOSS.
1490	Calculates the measurement frequency of the maximum level with FMAX function and assigns it to variable MAX_F.
1500	Calculates the bandwidth of 3dB with CBAND function and assigns it to variable BW3DB.
1510	Exits from the interrupt processing routine.
1520	Comment line.
1530	Display routine RESULS lable
1540 to 1560	Moves the cursor and displays the insertion loss value by moving.
1570 to 1590	Moves the cursor and displays the 3dB bandwidth frequency.
1600	Exits from the display routine.

11.3 Ripple Analysis Program

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

Example 11-3 Ripple Analysis Program (1 of 2)

```
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
```

(2 of 2)

```
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 analyazed using PMAX, BNDL and BNDH. The PMAX function calculates the measurement point of the maximum response and calculates the bandwidth from that measurement point. The BNDL function calculates the frequency for the low frequency band and the BNDH function calculates the frequency for the high frequency band. Then, this range is specified for the ripple analysis.

The ripple analysis is performed after the frequency has been converted into an address point by the POINT2 function. Various ripple analysis functions such as RPL2 and RPL4 are used in this program. Both of these functions are used to calculate the maximum values of the neighboring highest value and lowest value, but the ways in which they do this are quite 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.

Example 11-3 is explained below.

Program Explanation of Example 11-3 (1 of 2)	
1000 to 1050	Comment lines
1060	Defines the character string array.
1070	Comment line.
1080	Display routine (MAIN) lable
1090	Calls out the initial setting routine SETUP.
1100	Clears the screen.
1110	Measurement repetition loop (MEAS_LOOP) lable
1120	Calls out the measurement routine MEAS.
1130	Calls out the display routine RESULT.
1140	Measurement loop.
1150	Comment line.
1160	Initial setting routine (SETUP) lable
1170	Assigns address 31 to the variable NA .
1180	Releases IEEE488.1-1987 command mode.
1190	Presets the network analyzer and switches it to single scan mode.
1200	Sets the split screen mode.
1210	Sets the calculation format to LOGMAG & DELAY
1220	Sets the scan centre frequency to 17.9MHz, and SPAN to 30kHz.
1230	Sets the resolution bandwidth to 1kHz.

Program Explanation of Example 11-3 (2 of 2)	
1240	Sets the scan time to one second.
1250	Setup routine.
1260	Comment line.
1270	Measurement routine (MEAS) lable
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 for input.
1320	Comment line.
1330	Executes the scan one time and sends OPC quarry.
1340	Waits until the end of the scan.
1350	Automatically sets the Y axis.
1360	Comment line.
1370	Calculates the measurement point of maximum level and assigns it to variable A1.
1380	Calculates the frequency on the low frequency side of the 3dB bandwidth, and assigns it to variable A2.
1390	Calculates the frequency on the high frequency side of the 3dB bandwidth, and assigns it to variable A3.
1400	Converts frequency A2 to an address point, and assigns it to A4.
1410	Converts frequency A3 to an address point, and assigns it to A5.
1420	Calculates the maximum value of the neighboring highest value and lowest value from the amplitude data with RPL2 function.
1430	Calculates the maximum values of the neighboring highest value and lowest value from the amplitude data with RPL4 function.
1440 to 1480	Assigns the largest value to variable B3
1490	Calculates the maximum values of the neighboring highest value and lowest value from the delay data with RPL2 function.
1500	Calculates the maximum values of the neighboring highest value and lowest value from the delay data with RPL4 function.
1510 to 1550	Assigns the largest value to variable C3.
1560	Exits from the measurement routine.
1570	Comment line.
1580	Display routine (RESULTS) lable
1590 to 1600	Displays the ripple analysis value of the amplitude data.
1610 to 1620	Displays the ripple analysis value of the delay data.
1630	Exits from the display routine.

11.4 Example of Band-pass Filter Measurement

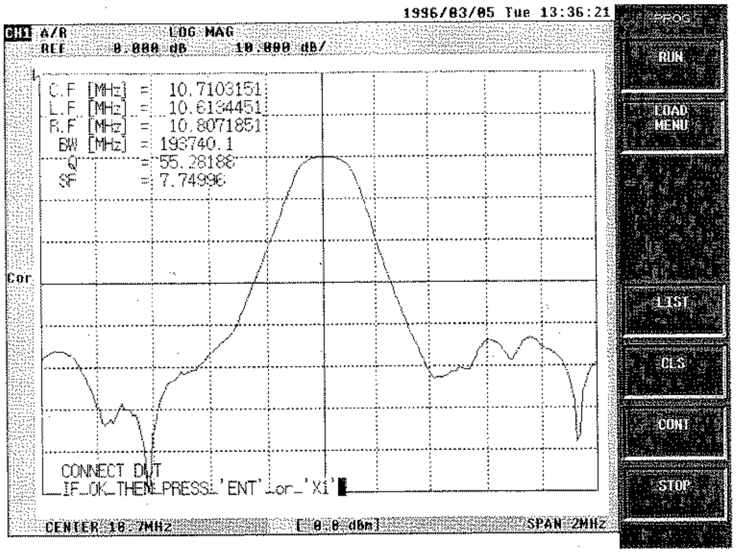
In this example, the band-pass filter measurement of the center frequency 10.7MHz is used as an example to explain the filter analysis program.

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 executed, the screen display changes 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 the Execution Result (Measurement of Band-pass Filter)

After the measurement point of the maximum response value has been found with the PMAX function, calculate the bandwidth and frequency of the two measurement points using the MBND function whose attenuations are 3dB and 60dB, respectively.
Using the MBND function, the analysis of more than one attenuation level at a time can be performed. The low band frequency, high band frequency, the center frequency and the bandwidth can each be obtained.

11.5 Example of Crystal Resonant Point Measurement

In this section, a program seeking from the measurement of transferring resonant point and anti-resonant point of ceramic oscillator (f = 45.1 MHz) is explained .
An example of a program which does this is shown in Example 11-5 .

Example 11-5 Measurement of Crystal Resonant Point (1 of 2)

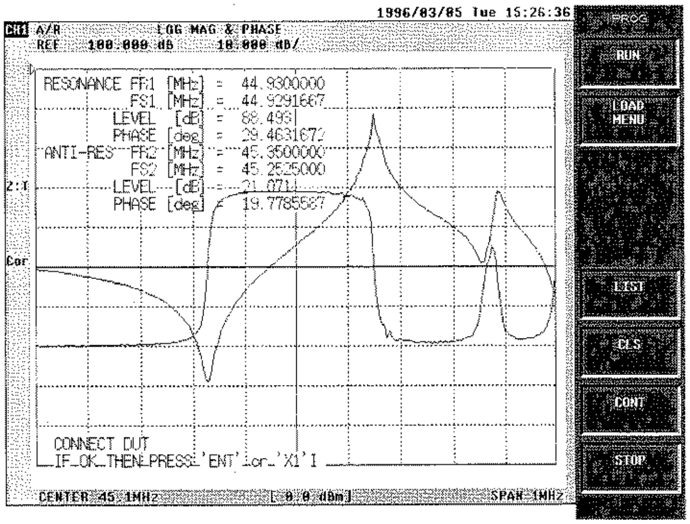
```
1000  !*******************************************
1010  !
1020  !           SEARCH  RESONANCE  POINT
1030  !           f=45.1MHz
1040  !
1050  ! FILE:RESONA.BAS
1060  !*******************************************
1070  *MAIN
1080      GOSUB *SETUP
1090      GOSUB *CAL
1100      CLS
1110      *MEAS_LOOP
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
```

(2 of 2)

```
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 display appears as shown in Figure 11-2. Here, πCircuit jig (PIC-001) is used for the setup.

Execution result:



```
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'
```

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

The resonant point and anti-resonant point can be determined through the following.

- Search using the maximum and minimum level values.
- Search using the zero phase.

Here, the measurement points of maximum and minimum level value are sought for first using the FMAX and FMIN functions.
Then, this program searches for the zero phase point near the measurement point using the ZEROPHS function.

11.6 Using Built-in Functions

The network analyzer is equipped with built-in functions related to waveform analysis, such as bandwidth analysis and ripple analysis.
In the network analyzer (except for the R3752H Series), the mark function is used to perform waveform analysis. However, with the built-in functions, you can perform all the operations by calling up one function.

For an explanation of the built-in functions, refer to "4.4 Built-in Functions" in the Program Manual.
In this section, an example of how to use the built-in functions is explained.

11.6.1 Basic Function

The built-in functions listed below are used to perform basic conversions such as calculating the 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 bandwidth corresponding to the specified frequency bandwidth.
SWPOINT	Gets the last measurement point.
FREQ	Gets the frequency corresponding to the specified address point.
DFREQ	Gets the frequency bandwidth 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 an argument. To use other built-in functions, convert the frequency to a measurement point using these functions. The absolute range of the address point is 0 to 1200. The measurement point is value within this range. The measurement point varies with the measurement point count set by the measurement condition.
The address point data except for measurement point uses the value interpolated from the measurement point.

The following program is an example of this.

```
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 below 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 for both the maximum and minimum response between the address points of the specified channel. Then, the analysis value of the measurement point is transferred as a function value.

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

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

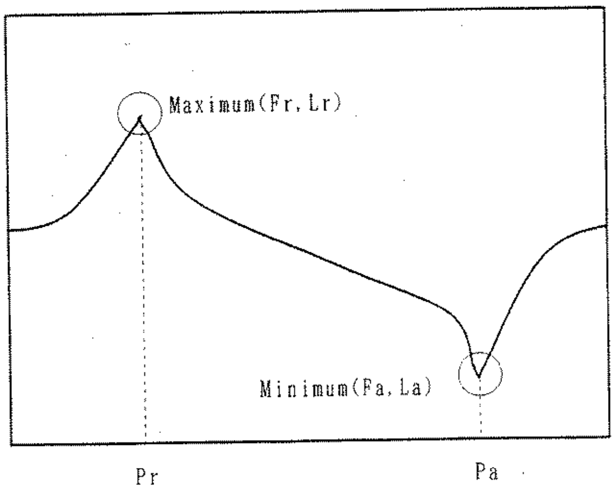


Figure 11-3 Maximum and Minimum Value Analysis Function

The following shows an example of a program used 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 the maximum or minimum response, it is not necessary to call out all these functions.

First, the measurement point is determined using PMAX or PMIN, then it is taken as the parameter used to call out the FREQ and VALUE functions.

In this way, the analysis value can be determined at a speed higher than that achieved 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 bandwidth from the specified address point.
BNDL	Gets the low frequency of bandwidth from the specified address point.
BNDH	Gets the high frequency of bandwidth from the specified address point.
CBND	Gets the bandwidth from the specified frequency.
CBNDL	Gets the low frequency of bandwidth from the specified frequency.
CBNDH	Gets the high frequency of bandwidth from the specified frequency.
MBNDI	Performs the multiple bandwidth analysis outwards.
MBNDO	Performs the multiple bandwidth analysis inwards.

1. BND, BNDL, BNDH, CBND, CBNDL, CBNDH

   These functions are used to analyze the attenuation point and bandwidth from the specified attenuation level. For those functions whose name start with C, the standard point of the function is specified with address pointer, and for those whose names do not start with C, they are specified with a frequency.

   Calculating the special filter parameter can be acheived by using a combination of these functions.

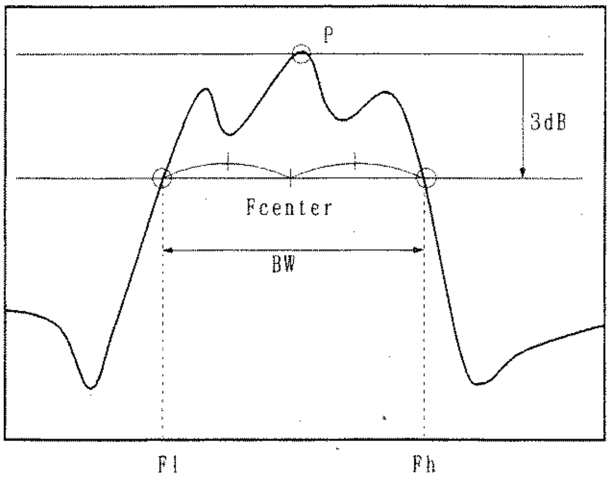


Figure 11-4 Analysis of Attenuation Level

The following is an example of a program using the attenuation level analysis

```
100 P  = PMAX(0,1200,0) ! Gets the measurement point of maximum
                          response.
110 BW = BND(P,3,0)     ! Gets the bandwidth of attenuation
                          level 3dB.
120 F1 = BNDL(P,3,0)    ! Gets the low frequency of attenuation
                          level 3dB of bandwidth.
130 Fh = BNDH(P,3,0)    ! Gets the high frequency of attenuation
                          level 3dB of bandwidth.
140 FC = (F1+Fh)*0.5    ! Calculates the center frequency.
150 Q  = SQR(F1*Fh)/ BW ! Calculates Q.
```

2. MBNDI, MBNDO

The MBNDI or MBNDO function is used when the doing an analysis of the multiple attenuation level.

These functions enable multiple attenuation points to be analyzed at the same time and the low frequency, high frequency, center frequency and bandwidth can each be obtained for one attenuation level.

When the attenuation level analysis is performed outward from the reference pointer, the MBNDI function is used.

11.6.3 Using Example of Attenuation Level Analysis Functions

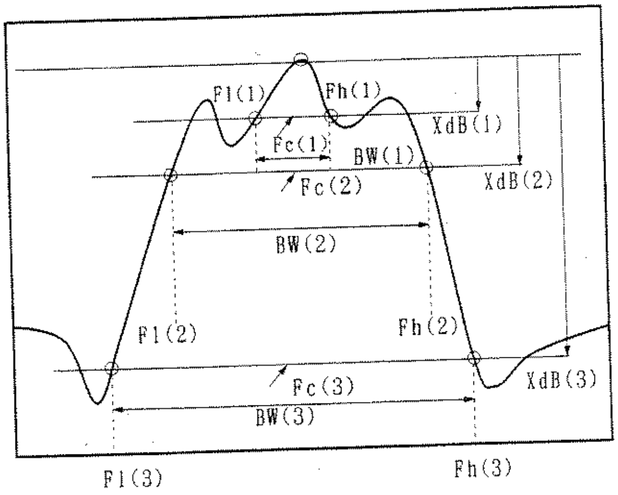


Figure 11-5 MBNDI

An example of a program using the MBNDI function is shown below.

```
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 bandwidth of Fl(1)
                                 -attenuation level 1.0dB.
180 PRINT DataBuffer(1,2)      ! High frequency of bandwidth of Fh(1)
                                 -attenuation level 1.0dB.
190 PRINT DataBuffer(1,3)      ! Center frequency of bandwidth of
                                 Fc(1)-attenuation level 1.0dB.
200 PRINT DataBuffer(1,4)      ! bandwidth of BW(1)- attenuation
                                 level 1.0dB.
210 PRINT DataBuffer(2,1)      ! Low frequency of bandwidth of Fl(2)
                                 -attenuation level 3.0dB.
220 PRINT DataBuffer(2,2)      ! High frequency of bandwidth of Fh(2)
                                 -attenuation level 3.0dB.
230 PRINT DataBuffer(2,3)      ! Center frequency of bandwidth of
                                 Fc(2)-attenuation level 3.0dB.
240 PRINT DataBurrer(2,4)      ! bandwidth of BW(2)-attenuation
                                 level 3.0dB.
250 PRINT DataBuffer(3,1)      ! Low frequency of bandwidth of Fl(3)
                                 -attenuation level 10.0dB.
260 PRINT DataBuffer(3,2)      ! High frequency of bandwidth of Fh(3)
                                 -attenuation level 10.0dB.
```

(Cont'd)

```
270 PRINT DataBuffer(3,3)    ! Center frequency of bandwidth of
                               Fc(3)-attenuation level 10.0dB.
280 PRINT DataBurrer(3,4)    ! bandwidth of BW(3)- attenuation
                               level 10.0dB.
```

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

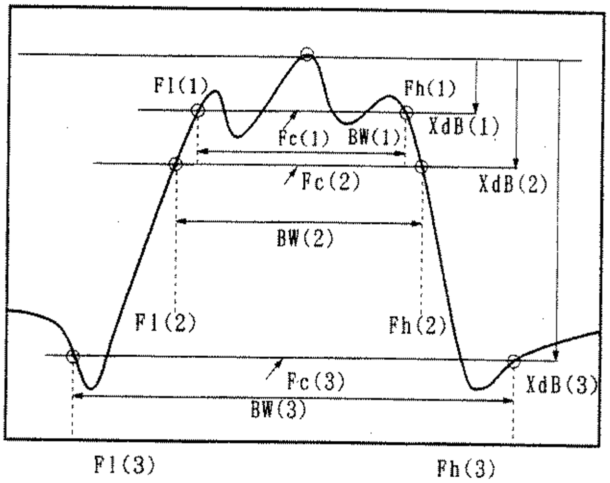


Figure 11-6 MBNDO

An example of a program using the MBNDO function is shown below.

```
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 bandwidth of Fl(1)
                               -attenuation level 1.0dB.
180 PRINT DataBuffer(1,2)    ! High frequency of bandwidth of Fh(1)
                               -attenuation level 1.0dB.
190 PRINT DataBuffer(1,3)    ! Center frequency of bandwidth of
                               Fc(1)-attenuation level 1.0dB.
200 PRINT DataBuffer(1,4)    ! bandwidth of BW(1)- attenuation
                               level 1.0dB.
```

(Cont'd)

```
210 PRINT DataBuffer(2,1)    ! Low frequency of bandwidth of Fl(2)
                               -attenuation level 3.0dB.
220 PRINT DataBuffer(2,2)    ! High frequency of bandwidth of Fh(2)
                               -attenuation level 3.0dB.
230 PRINT DataBuffer(2,3)    ! Center frequency of bandwidth of
                               Fc(2)-attenuation level 3.0dB.
240 PRINT DataBurrer(2,4)    ! bandwidth of BW(2)-attenuation
                               level 3.0dB.
250 PRINT DataBuffer(3,1)    ! Low frequency of bandwidth of Fl(3)
                               -attenuation level 10.0dB.
260 PRINT DataBuffer(3,2)    ! High frequency of bandwidth of Fh(3)
                               -attenuation level 10.0dB.
270 PRINT DataBuffer(3,3)    ! Center frequency of bandwidth of
                               Fc(3)-attenuation level 10.0dB.
280 PRINT DataBurrer(3,4)    ! bandwidth of BW(3)- attenuation
                               level 10.0dB.
```

11.6.4 Ripple Analysis Functions Using Example (1)

The following built-in functions are used to analyze ripples and produce a result.

RPL1	Gets the maximum value of the difference between the highest value and lowest value.
RPL2	Gets the maximum value of the difference between the neighboring highest value and lowest value.
RPL3	The total maximum value obtained by adding 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 the 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 an abscissa axis cant rate and ordinate axis cant rate. The cant rate coefficient of abscissa axis is specified with an address point, while the cant rate coefficient of ordinate axis is specified with a response value.
For example, 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

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

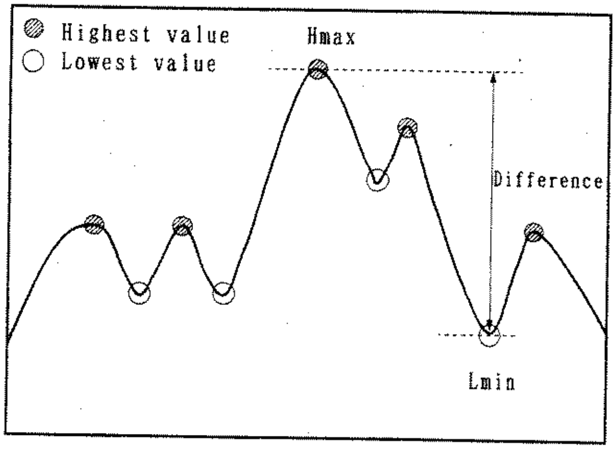


Figure 11-7 RPL1

An example of a program showing the maximum value of the difference between the highest and the lowest is shown below.

```
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 determine the maximum value of the 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.

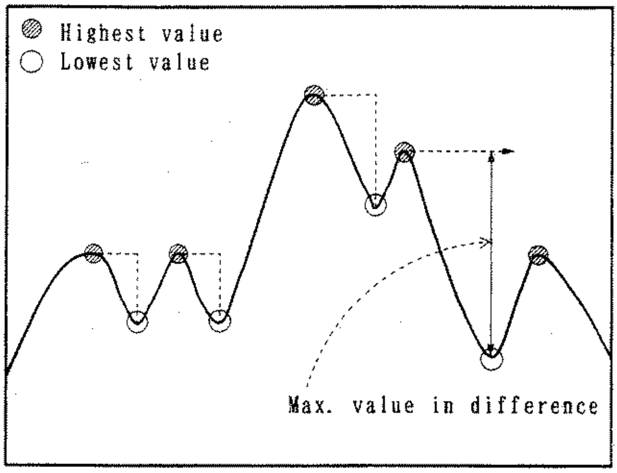


Figure 11-8 RPL2

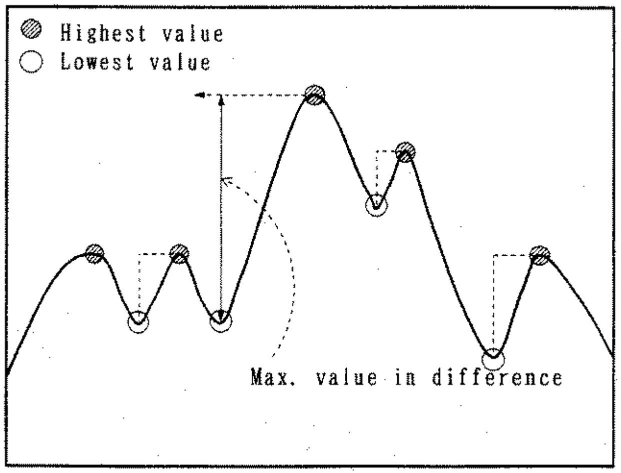


Figure 11-9 RPL4

An example is shown below.

```
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

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

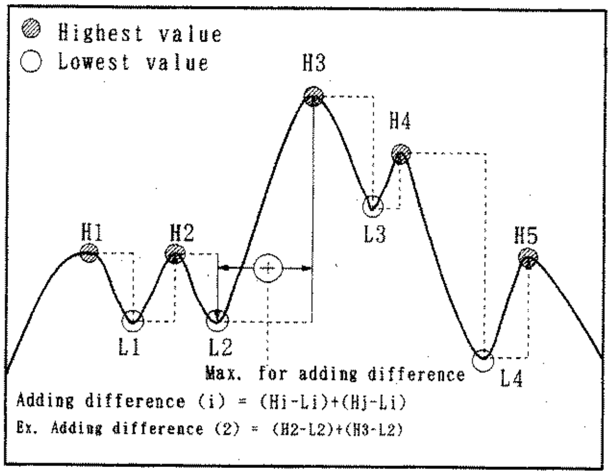

▨ Highest value
○ Lowest value
H3
H4
H1
H2
L3
H5
L1
L2
Max. for adding difference
L4
Adding difference (i) = (Hi-Li)+(Hj-Li)
Ex. Adding difference (2) = (H2-L2)+(H3-L2)

Figure 11-10 RPL3

An example is shown below.

```
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.

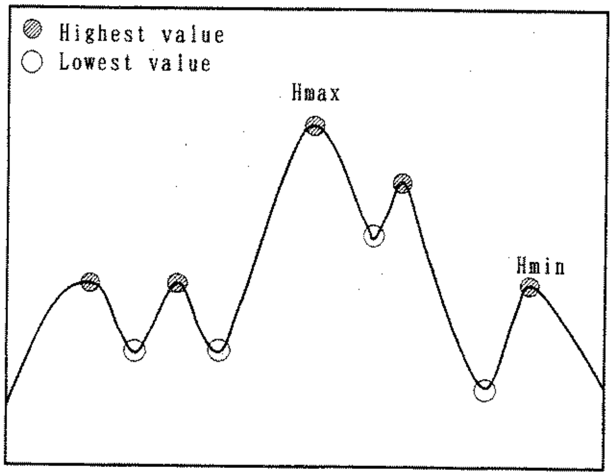


Figure 11-11 Maximum Value and Minimum Value of the highest value.

An example is shown below.

```
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 and FRPLH and FRPLL get the frequency value of the highest point or the lowest point and finally PRPLH and PRPLL get the measurement point of the highest point or the lowest point.

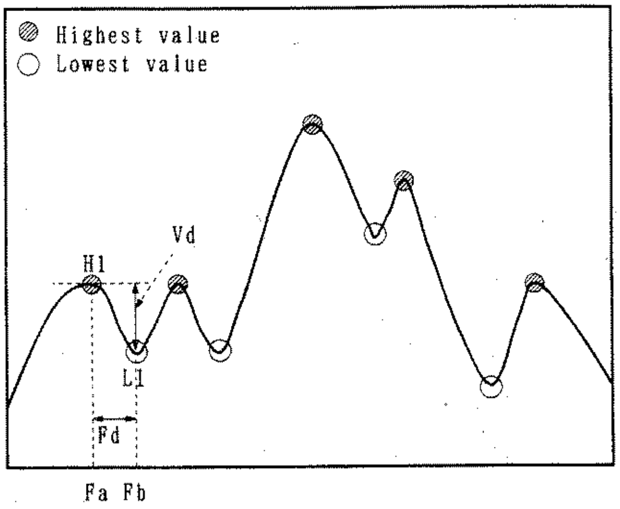


Figure 11-12 Response and Frequency of Ripple

An example is shown below.

```
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.
```

(Cont'd)

```
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 practical, because every time you call up the built-in functions, the overhead 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
                                   difference 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 Functions Using Example (2)

The following built-in functions are used to get and analyse ripples. When multiple ripples are analyzed, the analysis functions described below are used.

NRPLH	Gets the count of highest points.
NRPLL	Gets the count of lowest points.
PRPLN	Gets the measurement point of the n-th highest point.
PRPLN	Gets the measurement point of the n-th lowest point.
FRPLN	Gets the frequency value of the n-th highest point.
FRPLN	Gets the frequency value of the n-th lowest point.
VRPLN	Gets the response value of the n-th highest point.
VRPLN	Gets the response value of the n-th lowest point.
PRPLM	Gets measurement points for multiple highest points.
PRPLM	Gets measurement points for multiple lowest points.
FRPLM	Gets frequency values for multiple highest points.
FRPLM	Gets frequency values for multiple lowest points.
VRPLM	Gets response values for multiple highest points.
VRPLM	Gets response values for multiple lowest points.

1. NRPLH, NRPLL

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

2. PRPLHN, PRPLLN, FRPLHN, FRPLLN,VRPLHN, VRPLLN

   These functions are used to analyse selected ripples from those obtained by the functions NRPLH and NRPLL.

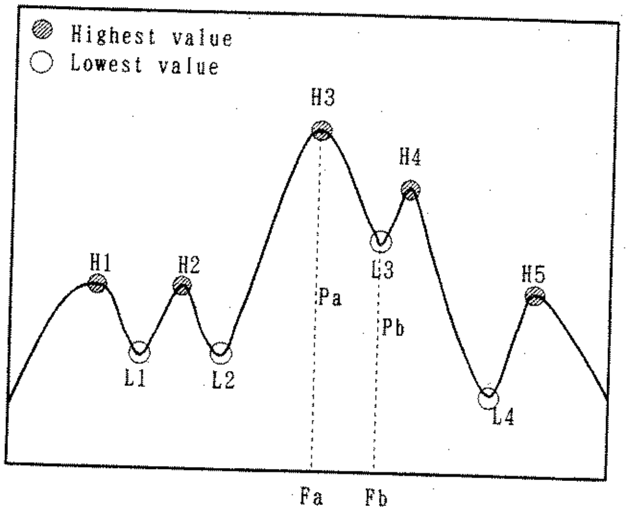


Figure 11-13 A Ripple Analysis

An example of this is shown below.

```
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.

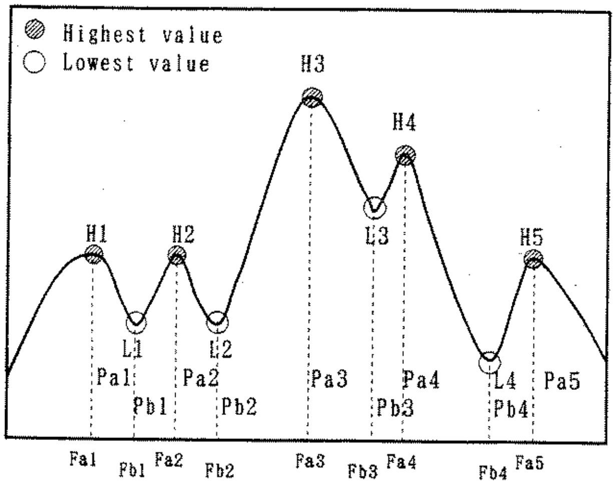


Figure 11-14 Analysis of All Ripples

An example of this is shown below.

```
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 for the response value given in the specified range.

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

1. DIRECT, DIRECTL, DIRECTH, CDIRECT, CDIRECTL, CDIRECTH

   These functions are used to specifie response values and then search the place that is coincident with the response value.
   For DIRECT functions whose name starts with C, the search range is specified with a frequency, while for others, the search range is specified with an address point .
   Functions whose name ends with L search from low frequency to high frequency, and those whose name ends with H search from high frequency to low frequency in order to find the analysis value corresponding to the real measurement value. However, when no measurement point is coincident, then use the one that comes immediately after the specified response value.

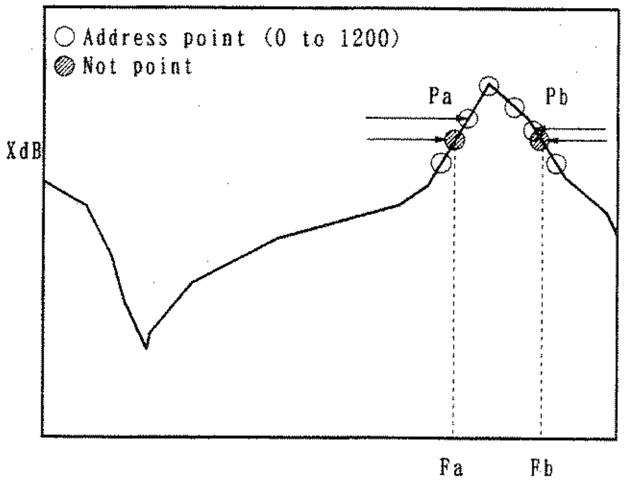


Figure 11-15 Direct Search

An example of this is shown below.

```
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 for two measurement points corresponding to the specified response value and get the point width between them.

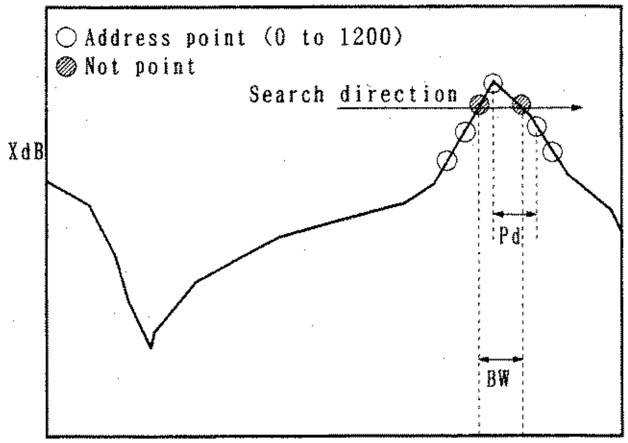


Figure 11-16 bandwidth Corresponding to Response

An example of this is shown below.

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

3. ZEROPHS

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

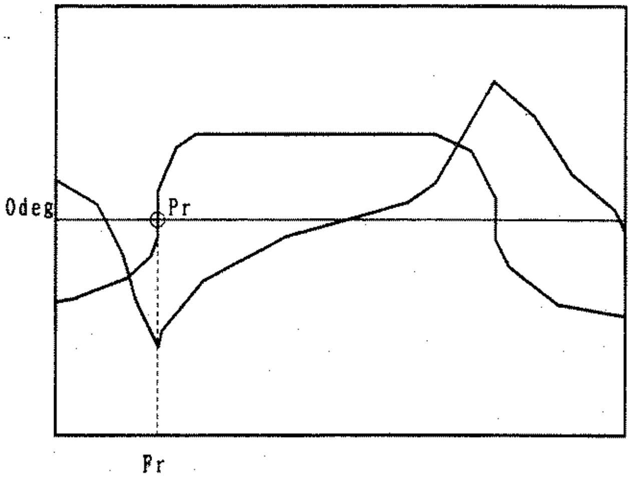


Figure 11-17 Search of Zero Phase

An example of this is shown below.

```
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 channel data between built-in and incorporated BASIC.

TRANSR Loads data from the analysis channel memory.

TRANSW Writes data to the analysis channel memory.

An example of this is shown below.

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

11.7 Setting Limit Line

In this section, a program example which is used to set 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: This cannot be used with the R3752H Series.

This is shown in Example 11-6.

Example 11-6 Setting the 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: the first is to set all the segments one at a time using DISPlay [: WINDow [<chno>]] : LIMit [<parano>] : DATA <block>. And the other method is to set each segment in an integrated way by using DISPLay [: WINDow [<chno>]]:LIMit [<parano>] : SEGMent <n> <Lock>. The parameter of each segment is set here, respectively

An example of this is explained below.

Program Explanation of Example 11-6	
100 to 150	Comment lines.
160	Sets variable I to integer type. (Because the segment is specified as an integer.)
170	Releases the R3762/63 conversion command mode.
180	Initializes the network analyzer.
190	Activates channel 2.
200	Sets the sweep frequency center to 880MHz, and the 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 the normalize-data and requests stop notification.
250	Waits until the normalize data has been received.
260	Comment line.
270	Changes the segment number I from 0 to 4 in sequence.
280	Reads the frequency data, upper limit value and lower limit value.
290	Sets the frequency, upper limit value and lower limit value to segment I.
300	Sets the limit line color and waveform.
310	Moves to the next segment.
320	Sets 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-Screen Display of All S Parameters

This section describes a 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: ***This cannot be used by the R3752/53H Series or R3754 Series.***
S parameter measurement in R3764/65/66/67H Series is enabled only when C type or A type +S parameter test sets is are used.

An example of this is shown below.

Example 11-7 Four-Screen Display of All S Parameter (1 of 2)

```
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"
```

(2 of 2)

```
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 (channels 3 or 4), 3 or 4 must be specified in <chno> of the measurement mode specification command [SENSe:] FUNCtion <chno> [: ON] “ <input>” or [SENSe:] FUNCtion <chno>: POWER <input>. Specifying the measurement format of channel 3 and 4 etc. is performed in advance with the measurement mode specification after the channel has been displayed.

An example of this is explained below.

Program Explanation of Example 11-7 (1 of 2)	
100 to 160	Comment lines.
170	Main routine (MAIN) lable
180	Calls out the initial setup routine SETUP.
190	Calls out the correction routine CAL.
200	Calls out the four screen display routine DISP4 CH.
210	Ends.
220	Comment line.
230	Initial setting routine (SETUP) lable
240	Releases the R3762/63 from the convention command mode.
250	Initializes the network analyzer.
260	Sets the scan frequency to center 880MHz and span to 100MHz.
270	Sets the resolution bandwidth 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	Exits the initial setup routine.
310	Comment line.
320	Correction routine (CAL) lable
330	Displays the message and waits for input. (applies below as well)
340	Gets the correction data (S11 OPEN)
350	Waits until the correction data has been collected (applies to the lines below)
360 to 380	Gets the correction data (S11 SHORT).
390 to 410	Gets the correction data (S11 LOAD) .
420 to 440	Gets the correction data (S22 OPEN) .
450 to 470	Gets the correction data (S22 SHORT) .
480 to 500	Gets the correction data (S22 LOAD) .
510	Comment line.
520 to 540	Gets the correction data (GROUP THRU).
550 to 560	Gets the correction data (Omits ISOLATION correction).
580	Comment line.
590	Calculates error coefficient from the correction data.
600	Enables the resolution bandwidth to be set automatically.
610	Clears the characters on the screen.
620	Exits from the correction data routine.
630	Comment line.

Program Explanation of Example 11-7 (2 of 2)	
640	Four screen display routine (DISP4CH) lable
650	Enables two channel display, upper and lower two parts split display.
660	Sets the measurement mode of channel 1 to S11.
670	Sets the measurement mode of channel 2 to S12.
680	Sets the measurement mode of channel 3 to S22.
690	Sets the measurement mode of channel 4 to S21.
700	Sets the measurement format of channel 1 to smith chart (R+jx).
710	Sets the measurement format of channel 2 to amplitude/phase.
720	Sets the measurement format of channel 3 to smith chart (R+jx).
730	Sets the measurement format of channel 4 to amplitude/phase.
740	Exits from the four screen display routine.
750	Comment line.
760	Wait sweep end routine (SWPEND) lable
770	Requests the operation end notification.
780	Gets the notification.
790	Exits from the wait sweep end routine.

12. Using Example of External Controller

To connect a computer to the network analyzer using GPIB, and exchange data between them, it is necessary to know computer language, and how to create a program. This chapter offers examples in BASIC (N88-BASIC, QuickBasic, HP-BASIC) and C language. While it is not necessary to know all of them it is recommended that you familarize yourself with at least BASIC prior to attempting any programming work.

Refer to the following documents for information on programming .

- Operation Manual or User Manual (Functional Descriptions)
- 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 to another controller or peripheral apparatus with a GPIB cable.
In this section, a program used to control the network analyzer with an external controller (Personal computer) that is connected by GPIB cable is shown.

Some computers can use GPIB commands as soon as their power is switched on however it is still 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 to output necessary commands to open the input/output port.
Set up the controller to be used after throughly reading the manual.

This chapter describes a method centered on an example that takes a PC-9801 computer which uses the NEC pure GPIB interface board as an external controller.
The programming language used is N88 Japanese BASIC.

Set the network analyzer as specified after the external controller has been setup. To control the network analyzer from the external controller, the network analyzer should be set to GPIB mode and connected by GPIB cable, and the GPIB address of the network analyzer must be set. (Refer to 12.1.3.)

12.1.1 GPIB Mode

The following two GPIB modes are used by the network analyzer.

- SYSTEM CONTROLLER mode

  Allows you to measure the function and control the machines connected to the network analyzer using the built-in BASIC programs.

- TAKER/ LISTENER mode

  Allows you to control the network analyzer with an external controller.
  Since the built-in BASIC interpreter is shared, the load of the external controller can be reduced.

12.1.2 Connecting the Network Analyzer

Connect the GPIB cable (available optionally) to the connector located on the back of the network analyzer, and then connect the cable with the connector on the external controller.

Make sure to read the interface board and computer manuals carefully before connecting them.

NOTE: ***It is necessary to purchase the GPIB interface board when using NEC-9800 series and IBM-PC compatible computer.***

GPIB cables are named according to their length. The following table lists the cable names and lengths.

Table 12-1 GPIB Cable (optionally available)

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

12.1.3 Setting GPIB Address

To control the network analyzer externally using GPIB, it is necessary to set a GPIB address for it. When the address is set, it is stored in the non-erasable memory of the network analyzer. Once this has been done, it does not have to done again unless you need to change the address.
The methods used to set the GPIB address depend upon the model used; methods for the R3752/64/66H Series, R3753H Series/R3754 Series and R3765/67H Series/R3765/67G Series are each described below.

NOTE: ***Use the front panel keys to set GPIB addresses.***

- For the R3752/64/66H Series

1. Press **[•](CONFIG)** from the program mode to change to CONFIG mode. The list of system variables is displayed.
2. Press **[•](CONFIG)** and hold it down untill the ADDRESS label is highlighted in the display.
3. Input the address with numeric keys, then press **[ENT]**.
4. To store the setting in the non-erasable memory of the network analyzer, press **[ENT]** again. Since the message for your confirmation is displayed on the display, press **[ENT]** when you wish to store it.

- For the R3753/65/67H Series, R3765/67G Series and R3754 Series

1. Press **[LCL]** to open GPIB menu.
2. Select ***{SET ADDRESSES}*** from the menu, and swich to the SET ADDRESSES menu.
3. When ***{ADDRESS R37XX}*** (R37XX : using model) is selected from the next menu, the currently set address is displayed in the active area.
4. Here, input the address using the numeric keys and press **[X1]**.
   Then, the address of the network analyzer is set and stored in non-erasable memory.

NOTE: ***When setting the GPIB address, make sure that the address allocated to the external controller and the addresses of other connected machines do not overlap.***
The address specified here is that used when the network analyzer is controlled by an external controller. When the network analyzer is controlled with built-in BASIC, use the address is 31.

12.2 Writing Method of Program

So far, examples of programs using BASIC have been explained. However, the programs performed on an external controller are different and depend on the type of computer being used, the operating condition of the interface board used and the language in which the program is written.

The main external controllers used in this chapter are listed in the chart below.

Table 12-2 Main Features of 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 method for writing programs using the above-mentioned external controllers is explained.

Program outline:

1. Initializes the controller.
2. Sets the measurement condition of the network analyzer.
3. Searches measurement data using the built-in BASIC of the network analyzer (preparation).
4. Loads measurement data from the built-in BASIC of the network analyzer.
5. Displays the measurement data on the computer.
6. Ends the program.

12.2.1 N88-BASIC Writing Method

Change the PC-9801 to 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, enter 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 signals first.
In N88-BASIC, the ISET IFC and ISET REN commands are used.

- ISET IFC

  Send IFC (interface-clear) to initialize the GPIB interface.
  When the network analyzer is controlled from an external controller with GPIB, it must be specified.

- ISET REN

  After REN (remote-enable) is sent, switch the network analyzer to the remote state.
  When the network analyzer is in the remote state, the [• REMOTE] LED on the front panel of the network analyzer is switched on. Commands sent will not be executed unless the network analyzer is set to REMOTE.

When using N88-BASIC, the PRINT @ command is used to send GPIB commands to the network analyzer and the INPUT @ command is used to receive commands. These commands correspond to the OUTPUT and INPUT commands of built-in BASIC in the network analyzer. The GPIB address of the network analyzer is specified after @.
In Example 12-1" , the address is 11.

When the command is sent to the built-in BASIC, @ is added at the beginning of each command. Commands preceded by @ are processed with different paths from GPIB control which is used to set measurement conditions. In Example 12-1, the measurement data is obtained by using the built-in BASIC. (For details, refer to "12.3.2 Transferring Built-in BASIC Commands by Adding "@".")

12.2.2 HP-BASIC Writing Method

When the network analyzer is controlled using a HP-9000, the program should be 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 using HP-BASIC, it is necessary to define the I/O path with the ASSING command at the start. In this program, the @Na of I/O path name is created by line 1110. ASSING @Na TO 711 and address 11 is allocated to the network analyzer.

The created I/O path name is used when the command is sent to the network analyzer or when data is received. Commands are sent by the OUTPUT command, while data is received using the ENTER command. However, the I/O path name must be included after the command.

12.2.3 Writing Method of Program in QuickBASIC

This example is written in QuickBASIC.

NOTE: In this program, NI-488.2 for PC/AT is used as the GPIB interface board.

* NI-488.2 : Register mark of National Instrument

Example 12-3 GPIB Control Program (QuickBASIC) on PC/AT (1 of 3)

```
' ****************************************
' *                                      *
' *         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$ = "GPIB0"
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$)
```

(2 of 3)

```
' (5) DISPLAY DATA
'
F = VAL(fdat$)
l = VAL(ldat$)
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 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%
        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.
'
```

(3 of 3)

```
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(emsg$)
END SUB
```

In N-488.2, various tools necessary for program development are included, however the interface board is not included. When the NI-488.2 system is purchased, install it in the computer only after reading the manual carefully.
The functions and variables starting with the characters ib are the library functions and variables used with the NI-488.2 system. In this program, IEEE488.1 library functions are used.

The NI-488.2 library and files used here are described below.

NI-488.2 library and files.

- qbdecl.bas

  A function declaration file of QUICKBASIC.
  When NI-488.2 is installed, a QBASIC directory is created, and then this is copied together with the sample program under the QBASIC directory.
  Copy this file to one's own operation disk.

- ibfind

  Searchs for the GPIB interface board and GPIB device that is connected to the board, and assigns an intrinsic value to them. The value obtained with ibfind is used as argument in NI-488.2 library.
  In Example 12-3 , it assigns brdo% to board (o) and address 11 to the network analyzer to na%.
  There are two types of library functions: functions operated with board level and functions operated with device assignment.

- ibsic

  Shows the IFC (Interface-clear) message. This operation is performed in communication with GPIB board.
  When ibsic is executed, it performs the initial setup for the GPIB interface board and the board is switched to the CIC (controller-in-charge) state.

- ibsre

  Control REN (remote-enable) signal. This operation is performed in communication with GPIB board.

When ibsre (brd$0, 1) is executed, the network analyzer is specified as the listen address, and set to REMOTE at the same time.

- ibwrt

  Sends a GPIB message to the specified device.

  In Example 12-3, ibwrt is used to send the GPIB command to the network analyzer as a sub -procedure.

- ibrd

  Receives a GPIB message from the specified device.
  In Example 12-3, ibrd is used to receive data from the network analyzer as a input sub-procedure.

- ibonl

  Releases the assigned device with ibfind.
  Be sure to call it when the program has ended.

- ibsta%, iberr%, ibcnt%

  These are status variables set by library functions.
  These variables are defined with qbdecl.bas file.
  Immediately after the library function is called, the status must be called out to check whether operation is normal.

Since GPIB operation commands are incorporated into N88-BASIC and HP-BASIC, GPIB commands are described as as general BASIC statements.
When an error occurs, it can be caught with the BASIC language system, so it is not necessary to program any special error processing.

However, this is not true for QuickBASIC. Since GPIB commands are not incorporated into the language, library functions (except for those linked) are accessed by CALL statements.

As a result, any errors that occur can not be caught when using GPIB operation from QuickBASIC so whenever the library function is called, the status variable must also be checked. If this is done soon after all the functions have been called, program lines will become longer, so that they 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, dedicated sub-procedures used for calling ibewrt and ibrd are described. They are referred to as output and input. When commands are sent to the network analyzer, 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.
However, error processing also takes place when sub-procedures gpiberr and baerr are used. When the error processing is described with sub-procedures, the program becomes easier to read.

To execute the program in Example 12-3 on MS-DOS, the execution file is created in the following sequence.

Execution file creation sequence:

1. Inputs the following command.

```
LIB QBIB.LIB + QBIB.OBJ;
```

   Create the library QBIB.LIB from the object file QBIB.OBJ is used for QuickBASIC provided with the NI-488.2 system.
   The MS-DOS LIB command is used to create the library.

2. Input the following command.

   ```
   BC qbstyle.bas;
   ```

   Compiles the program file qbstyle.bas using the BC command of QuickBASIC. The result can be an object file QBSTYLE.OBJ.

3. Input the following command.

   ```
   LINK QBSTYLE.OBJ,,,QBIB.LIB;
   ```

   Links QBSTYLE.OBJ and QBIB.LIB using the LINK MS-DOS command The execution file QBSTYLE.EXE is created with the above operation.

4. Input QBSTYLE and press ***Enter*** to execute the program.

12.2.4 Writing Method of Program in C

An example using ANSI-C is shown below.

NOTE: ***In this program, NI-488.2 for PC/AT is used as the GPIB interface board.***

Example 12-4 GPIB Control Program (ANSI-C) on PC/AT (1 of 3)

```
/*
 *      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;
```

(2 of 3)

```
     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) gpiberr("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 of 3)

```
 /*    (3) SEARCH DATA BY BUILTIN
  */
 outstr(na, "@AP=POINT1(1.5e+8,0)");
 outstr(na, "@LV=VALUE(AP,0)");
 outstr(na, "@FR=FREQ(AP,0)");
 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 the NI-488.2 package and used in this program. The library functions in this program are the same as those in Example 12-3 shown 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 shown in Example 12-4 .

   Input the following command.

   ```
   CL MCSTYLE.C MCIB.OBJ
   ```

   The execution file MCSTYLE.EXE is created by the above operation.
3. Input MCSTYLE and press ***Enter*** to execute the program.

12.3 Remote Control using the External Controller

In order to control the network analyzer remotely from an external controller, GPIB commands are employed. The built-in functions performing waveform data analysis, etc. cannot be executed using normal GPIB commands. Instead they are carried out by sending the commands with an at mark (@) added to the built-in BASIC. When @ is added and the command is sent to the built-in BASIC, it is processed.

12.3.1 Transferring Normal GPIB Command

The following programs change the measurement format of the network analyzer 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”, 14);
  ibonl(na, 0);
  ibonl(bd, 0);
}
```

12.3.2 Transferring Built-in BASIC Commands by Adding “@”

The commands with an at mark (@) added are used when waveform analysis function, etc. are performed. Part of the processing is executed using built-in BASIC to reduce the load the external controller has to handle.

The programs shown below are used to caluculate the maximum value of measurement data using these commands 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 a Scan End

In this section, a method for detecting scan ends with the external controller is explained.
First, the trigger mode of the network analyzer is switched to INIT : CONT OFF.
When the INIT command is sent, 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 has finished, one of the status registers 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 has ended. (Refer to "4 Status byte" of the Programming Manual).

12.4.1 Detecting a Scan End with N88-BASIC

The following program is used to detect the scan end during measurement by interrupting processing using N88-BASIC in the 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 up to that time will be paused, and the operation specified by the interrupt will be executed. In this program, when SRQ occurs, the infinite loop is interrupted and the program jumps to * MEAS.END.

Example 12-13 Detecting a 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 a Scan End with HP-BASIC

The following program is used to detect the scan end during measurement by interrupting processing, using HP-BASIC of the HP-9000 series.
It defines the branch destination (Measand) of the interruption using the ON INTR command, and enables the interruption using ENABLE INTR.

Example 12-14 Detecting a 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 a Scan End using QuickBASIC

The following program is used to detect the scan end during measurement using the NI-488.2 library functions with QuickBASIC on a PC/AT.
The spoll is a sub-procedure used to detect the service request which waits until the series requests are generated by the library function ibwait of NI-488.2 and reads in the status byte with the library function ibrsp of NI-488.2.
If an error occurs somewhere, the sub-procedure gpiberr is called out.
This sub-procedure executes a STOP command after displaying the error message.

For details on using the library functions, refer to the NI-488.2 manual.

Example 12-15 Detecting a 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 a Scan End using Microsoft C

The following program is used to detect the scan end during measurement using the NI-488.2 library functions with Microsoft C on a PC/AT.
As described in previous section, it detects the service request with a spoll and returns 0 when it has succeeded.

For details on how to use the library functions, refer to the NI-488.2 manual.

Example 12-16 Detecting a 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

Transferring data between an external controller and the built-in BASIC can be performed in either ASCII format or binary format.
In binary format, a format which corresponds to the external controller being used can be selected from the six available formats. (Refer to "7.7 Format Subsystem" of the Programming Manual.)
In ASCII format, the data can be transferred with a simple operation.
The speed of transferring data in binary format is higher than in ASCII format. When a high transfer speed is not required, the ASCII format is acceptable. However, when high speed is required, the binary format is recommended.

This section describes programs which show both methods of transferring data: ASCII and binary.

12.5.1 Transferring Trace Data from the Network Analyzer to PC-9801

The following programs are used to assign the trace data of the network analyzer to the array TR of N88-BASIC.
Transferring data in ASCII is shown in Example 12-17 and binary is shown in Example 12-18.
In N88-BASIC, there is no GPIB command used for transferring block data. Therefore, in this program, data reception 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) (1 of 3)

```
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
```

12.5.1 Transferring Trace Data from the Network Analyzer to PC-9801

(2 of 3)

```
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

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%)
```

(3 of 3)

```
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 is used, the first 8 bytes becomes the header. Here, the data, including the headers, 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 the Network Analyzer

The following programs are used to transfer the array data from N88-BASIC to the network analyzer.

CAUTION: ***This program can not be used before first getting trace data to use with it.***
Use the trace data input program presented above to do this and store the data in a file, and then run the program after reading the data into the array TR.

Example 12-18 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
```

12.5.2 Transferring trace Data from PC-9801 to the Network Analyzer

Example 12-19 Data Output of PC-9801 (Binary format) (1 of 2)

```
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
```

(2 of 2)

```
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 the Network Analyzer to HP-BASIC

The following programs are used to assign trace data to array Tr of HP-BASIC.
In binary format, IEEE Double-precision floating is selected.

Example 12-20 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 12-21 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;Trl(*)                        ! 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;":";Trl(I),Tr2(I)
1320    I=I+1
1330  END WHILE
1340  !
1350  END
```

12.5.4 Transferring Trace Data from the Network Analyzer to QuickBASIC

The following programs are used to assign the trace data of the network analyzer into array tr of QuickBASIC being executed with PC/AT.

NOTE: NI-488.2 interface board and library functions are used.

Example 12-22 Data Input of QuickBASIC (ASCII format) (1 of 3)

```
'    ************************************************
'    *                                              *
'    *            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("GPIBO", bd%)
CALL nainit(bd%, "DEV11", na%)
CALL tisetup(na%,  FDAT1")        ' trace 1
CALL tireceive(bd%, trl!())
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
```

(2 of 3)

```
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
```

(3 of 3)

```
 '   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 the Network Analyzer to C

The following programs is used to assign the trace data of the network analyzer into array tr of C being executed with PC/AT.

NOTE: ***NI-488.2 interface board and library functions are used.***

Example 12-23 Data Input of C (ASCII format) (1 of 3)

```
/*
 *       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;
```

(2 of 3)

```
     }
  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
 */
```

(3 of 3)

```
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("GPIBO")) == -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 12-24 Data Input of C (Binary format) (1 of 3)

```
/*
 *        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;
}
```

12.5.5 Transferring Trace Data from the Network Analyzer to C

(2 of 3)

```
  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 */
  ibcmd(bd, "?_ K", 4);                       /* 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);
}
```

(3 of 3)

```
/* 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 <Fil's 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", ibcnt1);
}
```

12.6 Using Built-in BASIC and External Controller Simultaneously

This section describes a method in which the network analyzer is used to perform the measurement by executing a BASIC program from an external controller, then the measurement data is received and displayed with the external controller.
In addition, a measurement program is created with built-in BASIC and a program example is given which shows how to perform 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 on the external controller side. The external controller program is different according to the computer and language used. In the first example, N88-BASIC programs used with the NEC PC-9801 are described.

The chart below shows an outline of the programs in the example.

External controller side		Built-in BASIC side
(1) Initialize the interface.		
(2) Executes the program after loading.	→	(1) Set the measurement condition of the network analyzer. (2) Pausing
(3) CONT the program of this instrument.	→	(3) Starts the scan and waits untill the scan has ended with WAIT EVENT.
(4) Loop waiting till the SRQ comes.		(4) Analyzes the measurement data.
(5) Performs a serial poll when SRQ comes.	←	(5) Outputs an SRQ with a REQUEST command.
(6) Receive data and displays it on the screen.	←	(6) Send data.
(7) Repeat from (3).		(7) Repeat from (2).

When data sending-receiving is performed between the external controller and the network analyzer built-in BASIC, 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 the network analyzer is shown below .

Example 12-25 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 the network analyzer 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 12-26 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 while in the 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 the Return key.

When the program is executed on the PC-9801 side, the program of the network analyzer side is loaded from the floppy disk and executed automatically.
When it is executed, according to the program, the network analyzer 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 shown here, it is recognized that the data is sent and received certainly.
In the 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 below.

Example 12-27 Receiving Program of HP-BASIC (1 of 2)

```
1000 !  ******************************************
1010 !  *                                        *
1020 !  *       CONTROL AND RECEIVE DATA         *
1030 !  *                                        *
1040 !  *  TARGET: HP-BASIC                      *
1050 !  *  FILE:   HPREC.BAS                     *
1060 !  ******************************************
1070 !
1080 DIM A$[64],M$[64],L$[64]
1090 !
1100 ASSIGN @Na TO 711
1110 ON INTR 7 GOSUB Srint
1120 !
1130 A$="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 !
```

12.6.3 Receiving Program of HP-BASIC

(2 of 2)

```
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 12-28 Receiving Program of QuickBASIC (1 of 3)

```
'  ***************************************** 
'  *                                       *
'  *        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
```

(2 of 3)

```
'    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$)
       PRINT USING "SF  = ####.######"; VAL(str6$)
```

(3 of 3)

```
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 below.

Example 12-29 Receiving Program of C (1 of 4)

```
/*
 *      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("ACS");
  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;
```

(2 of 4)

```
  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;
```

(3 of 4)

```
  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;
 }
```

(4 of 4)

```
/* 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 Downloading the BASIC Program

This section describes a method with which the program used for the network analyzer is downloaded from the external controller to the network analyzer and then executed.
On the contrary, the next section “12.8” will describe a method with which the programs can be uploaded from the network analyzer.

- 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 the network analyzer side via GPIB.

- Upload

  Upload is transferring the program existing in the memory of the network analyzer side to the memory of the external controller via GPIB.

When the program file used for built-in BASIC is managed with an external controller is downloaded, the program can be loaded without using the floppy disk and the built-in BASIC can be controlled as the same way 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 12-30 and Example 12-31 can be used for this.

Program outline:

1. Initialize the external controller.
2. Open the program file to be down-loaded and transfer the content to the network analyzer.
3. After downloading, 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 below.

Example 12-30 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 while in the BASIC mode of PC-9801.
Save it to a floppy disk after you have input it.

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 12-30 is used.

2. Create a program to be controlled with the external controller, then save it to a floppy disk.
   Here, a part of Example 12-31 is used after correcting it .
   Change line 1190 to 1190 ! PRINT NA ; L $ command line.

3. Execute the download program.
   After ENTER DOWNLOAD PROGRAM is displayed. It switches to waiting input state.

4. Here, enter the file name of the program to be downloaded. (For instance, NSEND.BAS etc.)
   After inputing the file name, press the Return key and then the file is loaded and the download starts.
   The download is performed line by line, and the lines are displayed as they are being transferred so they can be checked.
   When the download has ended, COMPLETE DOWNLOAD is displayed on the screen.

5. Next, as ENTER STARTING PROGRAM is displayed , input the file name of the control program to be executed with the external controller.
   The entered program is exchanged with the present downloaded program and executed.
   When only download is to be performed, press Return key.
   The execution result is the same as the result of Example 12-31.

12.7.2 Download Program of HP-BASIC

The download program to be executed with HP-BASIC is shown below.

Example 12-31 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 12-32 Download Program Used for QuickBASIC (1 of 3)

```
' ***************************************
' *                                     *
' *           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")
```

(2 of 3)

```
                  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>"
```

(3 of 3)

```
        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 below.

Example 12-33 Download Program used for C (1 of 3)

```
/*
 *      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;
```

(2 of 3)

```
    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("ibcnt1= %d\n\n", ibcnt1);
}

/* 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);
```

(3 of 3)

```
  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 a BASIC Program

This section describes a method with which a program existing in the memory of the network analyzer side can be uploaded to the external controller.
Since the uploaded program is controlled by the external controller side, the method used is almost the same as that for the download program.

NOTE: ***When the program of the network analyzer is uploaded with the upload program described in this example, make sure the last line of the program of the network analyzer is 65535 END. This is used to find the last line of the upload program.***

An example of an upload program to be used with N88-BASIC is shown below.

Example 12-34 Upload Program Used for N88-BASIC

```
1000 ' ******************************************
1010 ' *                                        *
1020 ' *              UPLOAD PROGRAM            *
1030 ' *                                        *
1040 ' * TARGET: PC-9801                        *
1050 ' * FILE :  NUPLD.BAS                      *
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 a program that can input and output all the correction data of two-port calibrations between external controllers.
Transferring correction data is carried out using the same method as that used for tracing (waveform) data. There are two formats used for transferring data; ASCII and binary. Like the tracing data, the transferring can be performed at a higher speed with binary format. Here, a transferring program in binary format is used as an example and presented with N88-BASIC and C.

12.9.1 Transferring Correction Data Between the Network Analyzer and a PC-9801 (N88-BASIC)

This program is used to transfer the correction data between the network analyzer and a PC-9801.

NOTE: An NEC pure GPIB interface board is used.

When receiving data, the correction data from the network analyzer is transferred to the PC-9801 and stored in its' disk drive. Perform the two-port calibrations in advance.
When sending data, the data is transferred to the network analyzer after the data in the disk drive is loaded. It is necessary to keep the 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 12-35 Transferring Correction Data Between the network analyzer and a PC-9801 (Binary format) (1 of 3)

```
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
```

(2 of 3)

```
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
```

(3 of 3)

```
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 5E,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 the Network Analyzer and a PC/AT (C language)

This program is used to transfer the correction data between the network analyzer and a PC/AT.
When receiving data, the correction data from the network analyzer is transferred to the PC/AT, and stored in the disk drive on the PC/AT. Perform the two-port calibrations in advance.
When sending data, the data is transferred to the network analyzer after the data in the disk drive has been loaded. It is necessary to keep the conditions such as point count, etc. as they were before.

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 12-36 Transferring Correction Data Between the Network Analyzer and a PC/AT(Binary format) (1 of 4)

```
/* 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 */
```

12.9.2 Transferring Correction Data Between the Network Analyzer and a PC/AT (C language)

(2 of 4)

```
#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");

  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 */
```

(3 of 4)

```
}

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';
```

(4 of 4)

```
     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