

**Manual Supplement  
Using HP Instrument BASIC  
with the HP 8711**



HP Part No. 08711-90113  
Printed in USA July 1993

Microwave Instruments Division

# Contents

---

<b>1. Introduction</b>	
Overview of HP Instrument BASIC . . . . .	1-2
Using HP Instrument BASIC . . . . .	1-2
Typographical Conventions . . . . .	1-3
<b>2. Recording Programs</b>	
Keystroke Recording . . . . .	2-1
What is Keystroke Recording? . . . . .	2-1
IBASIC Programs and the HP-IB Buffer . . . . .	2-2
What's in a Recorded Program . . . . .	2-2
The OUTPUT Statement . . . . .	2-3
The ASSIGN Statement . . . . .	2-3
SCPI Mnemonics . . . . .	2-3
How Recording Works . . . . .	2-4
Operations That Do Not Record . . . . .	2-5
Front Panel Operations Without Mnemonics . . . . .	2-5
HP Instrument BASIC Operations . . . . .	2-6
Operations Requiring Additional Programming . . . . .	2-6
Synchronization . . . . .	2-6
Active Control of the HP-IB Interface . . . . .	2-6
Mnemonics With No Corresponding Front Panel Operation . . . . .	2-7
Avoiding Recording Errors . . . . .	2-7
Use Instrument Preset . . . . .	2-7
Specifically Select Parameters . . . . .	2-8
Use HP-IB Echo . . . . .	2-8
Using the Built-in Editor to Write Programs . . . . .	2-8
<b>3. Running, Pausing and Stopping Programs</b>	
AUTOSTART Programs . . . . .	3-1
Running and Continuing a Program . . . . .	3-2
Pausing a Program . . . . .	3-3
Stopping a Program . . . . .	3-3
<b>4. Saving and Recalling Programs</b>	
Selecting a Disk . . . . .	4-2
Saving a Program . . . . .	4-2
AUTOSTART Programs . . . . .	4-3
Recalling a Program . . . . .	4-3

<b>5. Developing Programs</b>	
External Editors . . . . .	5-2
HP IBASIC for Windows . . . . .	5-2
HP BASIC . . . . .	5-2
ASCII Word Processors . . . . .	5-3
Using <b>Edit</b> . . . . .	5-3
The IBASIC Editor . . . . .	5-4
Recording into an Existing Program . . . . .	5-4
Editing with an External Keyboard . . . . .	5-5
Inserting Lines . . . . .	5-6
Editing Lines . . . . .	5-6
Entering Program Lines . . . . .	5-7
Editing from the Front Panel . . . . .	5-7
Character Entry . . . . .	5-7
The Label Window . . . . .	5-8
Inserting Lines . . . . .	5-8
Removing Program Text . . . . .	5-8
Deleting Characters . . . . .	5-8
Deleting Lines . . . . .	5-9
Recalling a Deleted Line . . . . .	5-9
Renumbering, Copying, Moving, and Indenting Lines . . . . .	5-9
Using <b>IBASIC Display</b> . . . . .	5-10
Using <b>Utilities</b> . . . . .	5-12
<b>6. Debugging Programs</b>	
Setting Breakpoints . . . . .	6-2
Examining Variables . . . . .	6-2
Examining Strings . . . . .	6-3
Examining Arrays . . . . .	6-3
Displaying the Last Error Encountered . . . . .	6-3
<b>7. Graphics and Display Techniques</b>	
Using the Partitions . . . . .	7-1
Allocating Partitions . . . . .	7-2
De-Allocating Partitions . . . . .	7-3
Operation with No Partition . . . . .	7-3
Displaying Text . . . . .	7-4
Graphics Initialization and Scaling . . . . .	7-5
Using Graphics . . . . .	7-6
Drawing Figures . . . . .	7-6
<b>GRAPHICS EXCEPTIONS</b> . . . . .	7-8
GRID, RECTANGLE, POLYGON, and POLYLINE scaling differences . . . . .	7-8
Labeling with Different Partitions . . . . .	7-9
HP 8711 User Graphics Commands . . . . .	7-9

<b>8. Interfacing with External Devices</b>	
Communication with Devices . . . . .	8-1
HP-IB Device Selectors . . . . .	8-1
Moving Data Through the HP-IB . . . . .	8-2
General Structure of the HP-IB . . . . .	8-3
The System Controller . . . . .	8-3
Using the Serial and Parallel Ports . . . . .	8-3
General Bus Management . . . . .	8-5
REMOTE . . . . .	8-5
Host Instrument . . . . .	8-6
LOCAL LOCKOUT . . . . .	8-6
Host Instrument . . . . .	8-6
LOCAL . . . . .	8-6
Host Instrument . . . . .	8-6
TRIGGER . . . . .	8-7
Host Instrument . . . . .	8-7
CLEAR . . . . .	8-7
Host Instrument . . . . .	8-7
ABORT . . . . .	8-7
Aborting the Internal Bus . . . . .	8-7
HP-IB Service Requests . . . . .	8-8
Setting Up and Enabling SRQ Interrupts . . . . .	8-8
Servicing SRQ Interrupts . . . . .	8-9
Example . . . . .	8-9
Conducting a Serial Poll . . . . .	8-9
Passing and Regaining Control . . . . .	8-10
The IBASIC HP-IB Model . . . . .	8-10
External and Internal Busses . . . . .	8-10
Service Request Indicators . . . . .	8-10
IBASIC as the Active Controller . . . . .	8-11
Passing Active Control to the Instrument . . . . .	8-11
IBASIC as a Non-Active Controller . . . . .	8-12
Interfacing with an External Controller . . . . .	8-12
Transferring Data Between Programs . . . . .	8-13
Using OUTPUT and ENTER statements . . . . .	8-13
Data Transfer Example — External Program . . . . .	8-14
Data Transfer Example — Internal Program . . . . .	8-16
Setting and Querying Variables . . . . .	8-17
Downloading and Uploading Programs . . . . .	8-18
Downloading . . . . .	8-18
Uploading . . . . .	8-20

<b>9. Using Subprograms</b>	
User-Created Subprograms . . . . .	9-1
Built-In High-Speed Subprograms . . . . .	9-2
Example Programs . . . . .	9-4
Avoiding Multiple Loads of Subprograms . . . . .	9-4
<b>10. HP 8711 IBASIC Keyword Summary</b>	
<b>11. Example Programs</b>	
Program Summaries . . . . .	11-1
DATA_EXT — Data transfer between internal and external programs . . . . .	11-1
DATA_INT — Data transfer between internal and external programs . . . . .	11-1
DOWNLOAD — Download program to HP 8711 . . . . .	11-1
DRAW8711 — Drawing setup diagrams . . . . .	11-2
DUALCTRL — Two controller operation . . . . .	11-2
REPORT — Using the parallel port . . . . .	11-2
UPLOAD — Upload program from HP 8711 . . . . .	11-2
USER_BIT — Using the USER bit . . . . .	11-2
USERKEYS — Customized softkeys . . . . .	11-2
BAR, AVG, LOG — Barcode Programs . . . . .	11-2
DATA_EXT — Data transfer between internal and external programs . . . . .	11-3
DATA_INT — Data transfer between internal and external programs . . . . .	11-6
DOWNLOAD — Download program to HP 8711 . . . . .	11-8
DRAW8711 — Drawing setup diagrams . . . . .	11-10
DUALCTRL — Two controller operation . . . . .	11-13
REPORT — Using the parallel port . . . . .	11-16
UPLOAD — Upload program from HP 8711 . . . . .	11-20
USER_BIT — Using the USER bit . . . . .	11-22
USERKEYS — Customized softkeys . . . . .	11-24
BAR — Using Barcode Reader . . . . .	11-27
AVG — Using Barcode Reader . . . . .	11-32
LOG — Using Barcode Reader . . . . .	11-36

## Figures

---

4-1. The <b>SAVE RECALL</b> Screen . . . . .	4-2
5-1. The HP IBASIC Program Editor . . . . .	5-3
5-2. The PC Keyboard . . . . .	5-6
5-3. The IBASIC display partitions . . . . .	5-12
7-1. Display partitions on the HP 8711 . . . . .	7-1
7-2. Using INPUT with no display partition . . . . .	7-3
7-3. Printing to a display partition . . . . .	7-4
7-4. Pixel Dimensions with Available Display Partitions . . . . .	7-5
7-5. "HELP" program output . . . . .	7-8
11-1. Sample Barcodes . . . . .	11-40

## Tables

---

5-1. IBASIC Display Partitions . . . . .	5-11
7-1. IBASIC Display Partitions . . . . .	7-2
7-2. SCPI graphics commands . . . . .	7-10

## Introduction

---

A built-in HP Instrument BASIC (IBASIC) controller (option 1C2) can be ordered with the HP 8711 RF Network Analyzer. An upgrade (HP model no. 86224A) is also available to add the controller to an HP 8711 that was not purchased with this option.

This manual describes creating and using IBASIC software on the HP 8711 RF Network Analyzer. It demonstrates how to use IBASIC's programming, editing and debugging features. It also describes how to save and recall programs and how certain instrument-specific IBASIC features are implemented in the HP 8711.

The reader should become familiar with the operation of the HP 8711 network analyzer before programming it. This manual introduces the IBASIC operating and programming environment and provides examples of intermediate and advanced IBASIC programs. It assumes familiarity with the HP 8711 and HP Instrument BASIC for Windows or HP Series 200/300 BASIC.

Related information can be found in the following references:

- Information on the IBASIC language, including keyword descriptions, error messages, interface specifics and programming techniques is available in the *HP Instrument BASIC Users Handbook*:
- Information on operating the HP 8711 is available in the *HP 8711 Operating Manual* and the *HP 8711 User's Guide*.
- Information on programming the HP 8711, including example programs, is available in the analyzer's *HP-IB Programming Guide*.
- Information on the HP 8711's HP-IB command mnemonics is available in the analyzer's *HP-IB Command Reference*. This document includes a menu map of the HP 8711 that shows the corresponding command for each key in addition to a complete alphabetical listing of HP-IB commands.
- Information concerning HP Instrument BASIC for Windows (HP model no. E2200A) is available in *Installing and Using HP Instrument BASIC for Windows*.
- Information on the SCPI (Standard Commands for Programmable Instruments) programming language is available in *A Beginners Guide to SCPI*.
- Information on using the HP-IB is available in the *Tutorial Description of the Hewlett-Packard Interface Bus*.

---

## Overview of HP Instrument BASIC

When installed in your HP 8711, HP Instrument BASIC (IBASIC) can be used for a wide range of applications, from simple recording and playback of measurement sequences to remote control of other instruments.

IBASIC is a complete system controller residing inside your analyzer. It communicates with the analyzer via HP-IB commands over an internal interface bus (select code 8). It can also communicate with other instruments, computers, and peripherals using the external HP-IB interface (select code 7) or the serial (select code 9) or parallel (select code 15) I/O ports.

---

### Note



The HP 8711 can also be controlled by an external controller. It has a factory default external HP-IB address of 16. When using IBASIC to control other instruments, no other device should use the same address.

The external HP-IB address can be changed using either the front panel keys **SYSTEM OPTIONS** **HP-IB** **HP8711 Address** or the SCPI mnemonic "SYST:COMM:GPIB:ADDR".

---

---

## Using HP Instrument BASIC

You need not be proficient in a programming language to successfully use HP Instrument BASIC (IBASIC). In keystroke recording mode, IBASIC automatically builds an executable program by capturing measurement sequences as they are performed. With little or no editing of these program lines, you can immediately put your program to work controlling and automating your HP 8711 network analyzer.

IBASIC's programming interface includes an editor. Softkeys are available to allow you to run or continue a program or configure the display.

The IBASIC command set is a subset of the command set of HP Series 200/300 BASIC. In fact, IBASIC programs can be run on any HP BASIC workstation with very few changes. When an external PC keyboard (with a DIN connector) is connected to the HP 8711, the IBASIC user interface emulates the user interface of the HP Series 200/300 BASIC. The PC keyboard can be used for command entry, editing and program inputs.

IBASIC programs can be developed on external computers as long as correct IBASIC syntax is used. The recommended environment for external program development is HP Instrument BASIC for Windows (HP model no. E2200A) running on a PC compatible. Refer to chapter 8, "Interfacing with the HP-IB," for information on interfacing IBASIC and other computing environments. Porting information can be found in the "HP Instrument BASIC Language Reference" section of the *HP Instrument BASIC Users Handbook*.

---

## Typographical Conventions

The following conventions are used in this manual when referring to various parts of the HP Instrument BASIC and HP 8711 operation environments:

<b>HARDKEY</b>	The name of a hardkey on the front panel of the HP 8711. This notation is also used to represent keys on an external keyboard connected to the HP 8711's DIN interface.
<b>Softkey</b>	The label of a softkey.
<b>Softkey ON off</b>	Upper case selection in a softkey indicates the state AFTER the softkey is pressed.
<b>HARDKEY</b> <b>Softkey 1</b> <b>Softkey 2</b>	A series of hardkeys and softkeys represents the path to a given softkey or menu.
<b>Bold</b>	Bold typeface is used to emphasize a particular word or phrase.
<i>Italic</i>	Italic typeface is used when referring to the name of a different manual.
< element >	Angle brackets are used to signify a syntax element in a statement.

## Recording Programs

---

IBASIC programs for the HP 8711 can be created from the instrument's front panel using an external PC keyboard (option 1CL), on a PC compatible running IBASIC for Windows, on an HP Series 200/300 controller running HP BASIC, or on a workstation or PC using a text editor.

Keystroke recording, described in this chapter, is ideal for creating simple programs or measurement sequences for instrument control. If a program requires data processing, decision making, prompts for an operator or graphical setup diagrams these must be entered using another technique. Alternative methods of program development can be used to supplement keystroke recording and create more sophisticated programs. These methods are covered in chapter 5, "Developing Programs".

---

### Keystroke Recording

Of all the available methods of creating IBASIC programs, keystroke recording is by far the easiest. It requires only a couple of steps to set up and run, and can be accomplished with very little knowledge of programming.

#### What is Keystroke Recording?

Keystroke recording is a way to automatically create IBASIC measurement sequence programs. To enable recording, simply press **SYSTEM OPTIONS** **IBASIC** **Key Record ON off**. Then press the normal key sequences of a measurement on the analyzer. Press **SYSTEM OPTIONS** **IBASIC** **Key Record on OFF** to terminate the recording. The resulting program can then be run by pressing **Run** in the **SYSTEM OPTIONS** **IBASIC** menu.

IBASIC programs communicate with the HP 8711 over an internal bus. They use the same set of commands used by external controllers for remote operation of the instrument. Keystroke recording works by finding the bus command, called a SCPI mnemonic, that fits each operation performed from the front panel and then building a program line to duplicate that operation when executed.

All program lines built by keystroke recording are entered into the analyzer's program buffer. If the buffer contains no existing lines, a complete executable program will be created. If there is a program in the buffer when recording is turned on, the recorded statements are simply inserted into the existing program. Refer to chapter 5, "Developing Programs," for a description of how to record into existing programs.

## IBASIC Programs and the HP-IB Buffer

Recorded programs work by sending HP-IB commands to the instrument. These commands are queued into an input buffer by the instrument. An IBASIC program generally outputs the commands much faster than the instrument can execute them. This often causes the program to complete while the instrument is still executing commands in the input buffer. The instrument continues processing these commands until the buffer is empty.

This may have some side-effects if you are not aware of this interaction. For example, it may not be immediately obvious that the program has actually finished, since the instrument is still functioning "remotely." This could cause confusion if you try to pause and continue a program that has actually completed.

You can clear the buffer from within your program by inserting the statement `CLEAR 8` at the beginning of your program (see chapter 5 for information on editing programs).

Another side-effect of the speed with which the analyzer processes commands is that it is possible for a command to execute before a previous command has completed execution. The most common example of this is a data query that executes before a measurement sweep is complete. This interaction can lead to erroneous data being collected. For more information on synchronizing the execution of commands, refer to the "Synchronization" section of the *HP-IB Programming Guide*.

---

## What's in a Recorded Program

If you look at any program created using keystroke recording you will find that it is composed of three fundamental IBASIC statements: `ASSIGN`, `OUTPUT` and `END`. The following simple program demonstrates these statements:

```
1  ASSIGN @Hp8711 TO 800
2  OUTPUT @Hp8711;"SOUR1:POW -10 dBm"
10 END
```

The `ASSIGN` and `END` statements are automatically created when keystroke recording is used to create a new program (as opposed to modifying an existing one).

There will only be one `ASSIGN` statement at the beginning of a program and one `END` statement at the end, but in a typical program there will be many `OUTPUT` statements. Since the `OUTPUT` statement does the actual work of controlling the HP 8711, let's take a closer look at how it is used.

## The OUTPUT Statement

The IBASIC statement

```
OUTPUT < destination >; < data >
```

tells the internal computer to send some information (data) to a device at a specific address (destination). The destination can be a device selector (a number), or a name representing a number, called a path name. The data can take several forms but in recorded IBASIC programs it is a string containing commands for the instrument (a mnemonic).

Although the OUTPUT command is very flexible it is used only one way when generated by a recording. The following represents a typical OUTPUT command from a recording session:

```
OUTPUT @Hp8711;"SOUR1:POW -10 dBm"
```

Notice that the OUTPUT command is followed by a name representing a device selector (@Hp8711), followed by a semicolon and the data ("SOUR1:POW -10 dBm").

## The ASSIGN Statement

The destination in an OUTPUT statement specifies the address of the device. In recorded programs this address is represented by the I/O path name @Hp8711. The following line appears in all recorded programs before any OUTPUT statements:

```
ASSIGN @Hp8711 to 800
```

The ASSIGN statement allows you to substitute an I/O path name (a variable preceded by the @ symbol) for a device selector number. Therefore, after the above ASSIGN statement, the program line

```
OUTPUT @Hp8711;"SOUR1:POW -10 dBm"
```

is equivalent to

```
OUTPUT 800;"SOUR1:POW -10 dBm"
```

The device selector 800 specifies the host instrument as the destination of any data sent by the OUTPUT command. The program communicates with the HP 8711 via select code 8, the internal HP-IB interface, which is only used for communication between IBASIC programs and the analyzer. The analyzer will respond to any address on the internal interface from 800 to 899 (800 is typically used).

## SCPI Mnemonics

The data sent to the HP 8711 by the OUTPUT command is called a SCPI (Standard Commands for Programmable Instruments) mnemonic and is found in quotes following the device selector path name and semicolon:

```
OUTPUT @Hp8711;"SOUR1:POW -10 dBm"
```

SCPI is a standard instrument control programming language providing commands that are common from one product to another, reducing the number of "device specific" commands. It uses easy to learn, self explanatory syntax that provides flexibility for both novice and expert programmers.

The SCPI mnemonic codes used by IBASIC are the same ones used to control the instrument remotely via an external computer. External computers communicate with the HP 8711 over the external HP-IB bus while IBASIC programs communicate with it over the internal bus. In our example, the mnemonic "SOUR1:POW -10 dBm" tells the instrument to set the source power to -10 dBm.

For more information on HP-IB interfacing using IBASIC refer to chapter 8, "Interfacing with the HP-IB." The SCPI mnemonics for the HP 8711 are documented in the *HP-IB Command Reference*.

---

## How Recording Works

To fully understand IBASIC recording, it is important to understand the relationship between front panel instrument operation and the program that is generated to emulate that operation.

---

### Note



SCPI mnemonics entered in a program during a recording session do not have a one-to-one correlation with the actual keys that are pressed during that session.

---

The fact that the generated SCPI mnemonics do not exactly correspond to the keys actually pressed is important to remember. As you press a sequence of keys to perform an operation, the corresponding SCPI mnemonic for that **operation** is generated. The operation may take one keystroke or several, but the mnemonic is not generated until after a valid sequence of keystrokes is completed.

In other words, it is the functional operation of the instrument that is recorded as a mnemonic, not the keystrokes that it takes to perform that operation.

For example, recording the simple key sequence: **POWER** **Level** **-** **1** **0** **Enter** requires six keystrokes and produces only one mnemonic, "SOUR1:POW -10 dBm", which is generated **after** the sequence is completed. This is then automatically formed into the command:

```
OUTPUT @Hp8711;"SOUR1:POW -10 dBm"
```

and inserted into the program.

This means that if you accidentally press the wrong key in a sequence, it may not show up in the recorded program. Additionally, you cannot exactly mimic keystrokes to leave the instrument in a specific front panel state, unless it is a state that appears as a natural consequence of a completed operation.

As shown in the above example, pressing the **POWER** hardkey in a recording session has the effect of bringing up the **POWER** menu, but does not, by itself, generate a program line. You could not therefore leave the instrument with the **POWER** menu displayed.

---

## Operations That Do Not Record

Although keystroke recording works automatically in most situations, there are some operations that cannot be captured or can only be partially captured using this method. These generally fall into one of the following areas:

- Front panel operations with no corresponding SCPI mnemonic (such as transitional key sequences).
- IBASIC front panel operations (such as some of the softkey operations found under the **SYSTEM OPTIONS** **IBASIC** menu).
- Operations requiring additional programming steps (such as passing control of the HP-IB to the instrument for hardcopy output).
- HP-IB operations with no front panel equivalent (such as HP-IB query commands or data transfer).

---

**Note** Do not recall programs in keystroke record mode; doing so may overwrite previously recorded program steps.



---

## Front Panel Operations Without Mnemonics

There are some areas of front panel operation which have no corresponding SCPI mnemonics.

- Most operations on the front panel that require numeric entry allow you to use the knob to increment or decrement the current value. This will not record as a program line. You must always use the numeric keypad or step keys to enter any value if you want the operation to be recorded.
- During a measurement sequence it may take several key presses to cause an operation that will generate a mnemonic. The transitional sequences between actual instrument events are not recordable.
- Any default states you setup prior to recording or encounter while recording (and consequently do not select) are not recorded.

---

**Note** Instrument states that are not specifically selected or changed are not recorded.



Since these default states are not recorded, you must either actively select them to generate a program statement or make sure the instrument is in the same exact state when the program is run as when it was recorded. This is discussed further in the "Avoiding Recording Errors" section of this chapter.

## HP Instrument BASIC Operations

Some softkeys under the **SYSTEM OPTIONS** **IBASIC** menu cannot be recorded. Operations on programs, such as **Run**, **Continue**, **Edit** and **SAVE RECALL Programs**, do not record. You can, however, record display partitions and all other save and recall operations not having to do with IBASIC programs.

Although IBASIC operations cannot be recorded, many do have corresponding SCPI mnemonics that allow an external controller to control and communicate with internal IBASIC programs. For more information refer to chapter 8, "Interfacing with the HP-IB."

## Operations Requiring Additional Programming

Some operations that work well when performed from the front panel have circumstances that require special attention when used in a program. This is due to two kinds of problems, synchronization and active control.

### Synchronization

Timing and synchronization must always be anticipated where one event must complete before another can occur. One example of this is when you need to detect a state in the instrument before issuing the next command. For example, suppose you want your program to perform a limit test on data, but only after a sweep has been completed. You can record the command to perform the limit test by pressing key sequences. However, to detect when the instrument has completed a sweep, you must edit the program and include a routine that waits for a status register to indicate the end of the sweep.

---

### Note



Synchronization is only a problem with **overlapped** commands (such as the command to trigger a sweep), that is commands that don't hold off the processing of subsequent commands. The HP 8711 adds an extra command \*WAI when an overlapped command is created using keystroke recording. \*WAI prevents the HP 8711 from executing any further commands until the overlapped command has finished. For more information on synchronization see the "Synchronization" section of the *HP-IB Programming Guide*.

---

### Active Control of the HP-IB Interface

Some operations require the analyzer to be the active controller on the external HP-IB bus. This generally means that the analyzer must be the System Controller (or active control must be passed to it from an external controller, if one is connected). When an IBASIC program begins running, however, the instrument's active control of the external interface is automatically passed to the program, so active control must be passed back to the analyzer before these operations can be performed.

These operations include all of the following actions when they are directed to HP-IB devices. Note that active control of the HP-IB interface is only a problem if that bus is being used. Hardcopy output to devices on the serial or parallel ports do not require control of the HP-IB.

HARD COPY	Start
HARD COPY	Abort
SAVE RECALL	Select Disk
SAVE RECALL	Save State or Re-Save State
SAVE RECALL	Recall State
SAVE RECALL	Save ASCII Save Chan 1 or Save Chan 2

You can keystroke record any of these operations but you will not be able to successfully run the program that is generated. You will need to enter the program lines necessary to first pass control to the analyzer and then wait for control to be passed back to the program.

See the "Passing and Regaining Control" section of chapter 8 for an example of passing control to the analyzer.

### Mnemonics With No Corresponding Front Panel Operation

Several of the HP 8711 SCPI mnemonics for the instrument perform operations that are not available from the front panel and which, therefore, cannot be recorded. These include operations such as querying instrument status, transferring data over HP-IB, setting and clearing status registers and general HP-IB housekeeping.

These operations are useful for the more advanced HP-IB programmer using IBASIC. Because they fall outside the direct operating realm of the HP 8711, they cannot be recorded. They can be added to a recorded program using the built-in editor or another editing environment. See the *HP-IB Command Reference* for a complete description of the analyzer's HP-IB command set.

---

## Avoiding Recording Errors

### Use Instrument Preset

In most cases, you should perform a preset before recording a measurement sequence and again before running the recorded program. This sets the instrument to its default state and avoids the risk of creating a program that depends on instrument settings that were present at the time of the keystroke recording but may be different when the program is run.

You can include the command to perform a preset in your program by pressing **PRESET** immediately after turning recording on. This inserts the following line prior to all other OUTPUT statements in your program:

```
OUTPUT @Hp8711;"SYST:PRES;*WAI"
```

### Specifically Select Parameters

If you do not want the instrument preset before a recorded program is run (for example, you may be recording a section of a larger measurement sequence), be sure to specifically activate every instrument setting that you will need in your automated sequence. For example, if you want the data format to be Log Mag, press **FORMAT** and then **Log Mag**, even though Log Mag is already the default setting. This will generate a program line to specifically set the data format to Log Mag.

In some cases you may have to select another setting first and then re-select the original setting in order to generate the correct program line. For example, if you want to generate a program line to set the sweep trigger to Continuous, and you discover that it is already set to Continuous when you start recording, press **MENU** **Trigger** **Hold** first — then press **Continuous**. You can easily remove unwanted program lines generated by this procedure in the editor.

### Use HP-IB Echo

HP-IB Echo is a useful HP 8711 feature that allows you to view the SCPI mnemonic or mnemonics corresponding to any operation executed from the front panel. To turn on HP-IB Echo, press **SYSTEM OPTIONS** **HP-IB** and **HP-IB Echo ON off**. After doing this you will see a mnemonic appear in a dialogue box on the screen as you complete any key sequence that has a matching SCPI mnemonic. This is the exact mnemonic that is generated in your recorded program during a recording session.

Using HP-IB Echo you can preview the SCPI mnemonic commands that will be stored in your program before you actually record them. While this is not essential, it can be very useful when you are in doubt as to what a particular key sequence will record, or precisely when a key sequence corresponding to a mnemonic is completed.

---

## Using the Built-in Editor to Write Programs

The HP 8711 has a built-in editor which can be used to write or modify IBASIC programs. Using the HP 8711's built-in editor allows you to write more powerful programs than you can create with simple keystroke recording.

For more information about using the editor to write programs see chapter 5, "Developing Programs."

## Running, Pausing and Stopping Programs

---

Program control — starting, pausing and stopping an IBASIC program — can be managed from the HP 8711 front panel using various hardkeys and softkeys. These actions and their corresponding keys are described in this chapter.

A special case is an autostart program which executes automatically on power-up if it exists on the HP 8711's built-in floppy disk drive.

IBASIC programs may also be remotely controlled via SCPI commands over the HP-IB. For information on running, pausing and stopping programs from an external controller see chapter 8, "Interfacing with the HP-IB."

---

### AUTOSTART Programs

When the HP 8711 is powered up, it automatically searches the built-in floppy disk drive for a program named `AUTOST` or `AUTOST.BAS`. When an `AUTOST` program is found, it is automatically loaded and executed.

The `AUTOST` program can be used for anything from configuring the HP 8711 for specific measurements, much like an internal instrument state Save/Recall register, to diagramming measurement setups using graphics commands, as in a guided measurement sequence.

Refer to chapter 4, "Saving and Recalling Programs," for information on using the HP 8711 to name programs before they are saved.

---

## Running and Continuing a Program

To run an IBASIC program that is already in the HP 8711 program buffer, press the **Run** softkey in the **SYSTEM OPTIONS** **IBASIC** menu. The RUN command can also be executed from an external keyboard in either of two ways.

- Press the function key **F1** that corresponds to the **Run** softkey (see note below).
- Type RUN on a command line and press **Enter**. A command line is always available when an IBASIC display is partitioned. (See chapter 5 “Developing Programs” for information about display partitions.) You can also activate a command line from an external keyboard with no IBASIC displays partitioned by pressing the **ESC** key on your external keyboard.

---

### Note



When an external keyboard is connected, its function keys **F1** through **F8** always represent the HP 8711's eight softkeys. The analyzer's hardkeys are each represented by a combination of **Shift** or **Ctrl** and one of the function keys. Refer to Appendix B at the end of Chapter 7 in the *HP 8711 Operating Manual* for more information on the external keyboard interface. The **SYSTEM OPTIONS** **IBASIC** menu can be accessed from an external keyboard using **Ctrl** + **F3** (for **SYSTEM OPTIONS**) and **F1** (for **IBASIC**).

---

The RUN command is executed in two phases: prerun initialization and program execution.

The prerun initialization phase consists of:

- Reserving memory space for variables specified in COM (both labeled and blank), DIM, REAL or INTEGER statements, or implied in the main program segment. Numeric variables are initialized to 0; string variables are initialized to the null string.
- Checking for syntax errors that require more than one program line to detect. Included in this are errors such as incorrect array references, and mismatched parameter or COM lines.

After prerun has been successfully completed, the program will begin the execution phase. Program lines will be executed until one of the following events occurs:

1. An END or STOP statement is encountered in the program.
2. The **PRESET** hardkey is pressed to reset the instrument.
3. The **Pause** softkey is pressed to pause the program.
4. A PAUSE statement is encountered in the program.

To continue a program from a paused state, press the **Continue** softkey (**F2** on an external keyboard) in the **SYSTEM OPTIONS** **IBASIC** menu. Continuing a paused program resumes program operation from where it was paused, retaining the current program context (variable values, etc).

---

## Pausing a Program

When an IBASIC program is running on the HP 8711 a softkey menu is always available. This “Program Running” menu has seven user-defined softkeys and a **Pause** softkey. Press the **Pause** softkey to suspend execution of a program. **Pause** is the eighth softkey and is represented by **F8** on an external keyboard.

The program can also be paused by inserting a **PAUSE** statement in the program. The instrument responds as if you had pressed the **Pause** softkey. Refer to chapter 5, “Developing Programs,” to learn how to insert statements in your recorded program. Note that **PAUSE** is one of the IBASIC keywords included in the editor’s label window (described in chapter 5).

To continue the program, press the **Continue** softkey (**F2** on an external keyboard) in the **SYSTEM OPTIONS** **IBASIC** menu. This menu automatically appears when a program is paused.

Pausing a program does not close any files that have been opened by the program. You will not be able to perform any of the following disk operations after pausing a program that has left a file open on that medium:

- **RENAME FILE**
- **DELETE FILE**
- **DELETE ALL FILES**
- **COPY FILES**
- **COPY DISK**
- **FORMAT DISK**

To close all open files, you must complete the execution of the program or perform an IBASIC **RESET**. This can be done by pressing the **PRESET** hardkey. The **PRESET** hardkey is represented by **Ctrl** + **F4** on an external keyboard. Keystroke recorded programs do not open files and therefore avoid this problem.

---

## Stopping a Program

To stop a program completely, press the **PRESET** hardkey at any time while the program is running. This causes an IBASIC **RESET**. Placing a **STOP** statement in your program will also terminate the program, but does not perform a BASIC **RESET** operation. The **END** statement can also be used to stop program execution, but it must be the last line in the main program segment.

The program remains in the program buffer after execution stops until it is cleared. To clear the program buffer, press **SYSTEM OPTIONS** **IBASIC** **Utilities** **Clear Program** or turn off the instrument.

For more information on the **PAUSE** and **STOP** statements see the “HP Instrument BASIC Language Reference” section of the *HP Instrument BASIC Users Handbook*.

## Saving and Recalling Programs

---

IBASIC programs can reside in memory, on disk, or in an external computer.

To transfer a program between the instrument's buffer and a mass storage device, use the **SAVE RECALL Programs** menu. To access the **SAVE RECALL Programs** menu using an external keyboard, use **Ctrl** + **F1** (for **SAVE RECALL**) and **F5** (for **Programs**).

The GET, SAVE, LOAD, STORE, RE-STORE, and RE-SAVE commands can be used within a program or from an IBASIC command line to transfer program files to and from mass storage. An autoloading feature also exists to allow for a program (named AUTOST or AUTOST.BAS) to be automatically recalled from disk at power-up and run.

Another mode of program transfer is between the analyzer and an external controller, such as an HP Series 200/300 controller or a PC compatible running IBASIC for Windows. Using an external controller, you can combine the convenience of keystroke recording in IBASIC with the ease of program editing in a dedicated external workstation by recording the measurement sequence and then uploading the program to the external controller for further editing. Fully developed programs may be downloaded from an external controller as well. The methods of transferring programs between the HP 8711 and an external controller are described in detail in chapter 8, "Interfacing with the HP-IB."

This chapter describes all program transfer operations between the program buffer and the HP 8711 internal memory, internal floppy disk drive and external mass storage devices.

---

**Note**

The IBASIC file system can work with both LIF (Logical Interchange Format) and DOS (Disk Operating System) formatted disks. When it catalogs or loads files from a disk, the HP 8711 automatically recognizes the correct disk format.

---

---

## Selecting a Disk

When the **SAVE RECALL** **Programs** menu is selected the HP 8711 automatically catalogs the selected disk or memory. The selected disk is one of the following mass storage devices:

- Internal Memory
- Internal Floppy Disk Drive
- External Disk Drive

To select a mass storage device press the **Select Disk** softkey in the **SAVE RECALL** menu. Then press the key corresponding to your choice. The HP-IB address of the external disk drive is set under the **SAVE RECALL** **Select Disk** **Configure Ext. Disk** menu.

---

## Saving a Program

To save the current contents of the HP 8711 program buffer to a file, press **Save Program** in the **SAVE RECALL** **Programs** menu. If desired, specify the type of file, binary or ASCII, with the **File Type** softkey; default is ASCII. The program is saved to an ASCII file with a default name on the currently selected mass storage device or disk. Each time the **Save Program** key is used a new file is created. These files are named PROG0.BAS, PROG1.BAS, ... with the number being changed for each new file. For portability, save files in ASCII: binary files can not be read by an HP 8711 with firmware revision 1.90 or below.

The screenshot shows the 'SAVE RECALL' screen with a file directory and a menu of options. The file directory is as follows:

FILE NAME	TYPE	SIZE	LAST CHANGE
..<PARENT>	<DIR>		
PROG0	PRG	7673	21-OCT-91 10:37
STATE0	STA	13056	17-JAN-92 13:52
TRANS	STA	3328	19-JAN-92 08:14
REFLS	STA	3328	19-JAN-92 08:20

The menu on the right side of the screen includes the following options:

- Programs
- Save Program
- Re-Save Program
- File Type  
bin ASCII
- Recall Program
- Save AUTOST
- IBASIC
- Prior Menu

pd61c

Figure 4-1. The **SAVE RECALL** Screen

If you are re-saving a program — that is, saving a file to a disk that already contains the file name — press **SAVE RECALL** **Programs** and use the arrow keys to highlight the name of the file to be re-saved. Then press **Re-Save Program** and the file is saved. The disk is automatically catalogued when the **SAVE RECALL** menu is selected.

The **Re-Save Program** softkey can also be used to save a new program with a non-default file name. Highlight any file or directory on the disk and press **Re-Save Program**. Enter the new program's name using the external keyboard or the internal label maker. If no file with that name exists on the disk a new file is created.

---

## AUTOSTART Programs

IBASIC allows you to designate a program to be automatically loaded and run when the instrument is first powered up. To make an autoloading program save it with the file name AUTOST on the internal floppy disk drive. This can be done from the **SAVE RECALL Programs** menu by pressing **Save AUTOST** or by using the **Re-Save Program** softkey and entering the file name AUTOST.

At power-up, IBASIC searches the internal floppy disk drive for the file AUTOST. If found, it is loaded and executed immediately.

---

## Recalling a Program

To recall a program file from mass storage to the program buffer, use the **SAVE RECALL Programs** menu to catalog the disk. Select the desired mass storage device or disk, use the arrow keys to highlight the file and press **Recall Program**.

The recalled program file is entered into the program buffer one line at a time and checked for syntax errors. Lines with syntax errors are commented out and the IBASIC syntax error is displayed briefly in an error message and written to the CRT at the same time. To view error messages logged to the CRT, use the **SYSTEM OPTIONS IBASIC IBASIC Display** menu to allocate a screen partition for IBASIC.

---

### Note



Any program recalled to the program buffer using the **SAVE RECALL Programs** menu will overwrite the current contents of the program buffer.

---

## Developing Programs

---

For many applications, you can use keystroke recording to create and run programs without needing to alter the program code that is generated. However, with some knowledge of the IBASIC language and the program development capabilities of the HP 8711, you can significantly increase the power of your recorded programs or create your own programs from the ground up.

This chapter describes the operation of the following keys in the **SYSTEM OPTIONS** **IBASIC** menu, and any softkeys found in their underlying menus:

- **Edit**
- **IBASIC Display**
- **Utilities**

The ability to change and enhance your program and its operating environment is found primarily under the **SYSTEM OPTIONS** **IBASIC** **Edit** menu.

**Edit** places you in the editor where you can make changes to your program on a line-by-line basis.

**IBASIC Display** menu allows you to select what part, if any, of the CRT display is available for the use of IBASIC. An IBASIC display partition provides you with a command line you can use to execute IBASIC commands from an external keyboard. It also provides an area for viewing graphics and program output.

**Utilities** allows you to Clear Programs from the program buffer, allocate memory for program use, or secure program lines.

---

## External Editors

In addition to using the built-in IBASIC editor, programs can be developed in the following external environments.

- HP IBASIC for Windows editor
- HP BASIC editors
- ASCII word processors

The external editing environments provide many advantages, the most notable being speed and flexibility. Precautions must be taken because most of them do not provide the syntax checking available when using the internal editor.

After editing a program in an external environment, the best practice is to GET the program from an IBASIC command line using the following procedure (instead of using the **SAVE RECALL** keys described in chapter 4).

1. Partition an IBASIC display (as described later in this chapter).
2. Use an external keyboard to enter the command GET "PROG0:,4" (this command loads a program file PROG0 from the internal floppy disk drive).
3. Watch the IBASIC display as the program is loaded — syntax errors result in error messages displayed on the screen.
4. Edit the program to correct any errors found.

### HP IBASIC for Windows

IBASIC for Windows is the recommended environment for external program development. This is because the syntax checking done by its editor can be changed by modifying the contents of a compatibility file (IBASIC.INI). To make the IBASIC for Windows editor check the syntax of a program written to run on the HP 8711 (IBASIC instrument version 2.0) the compatibility file should contain the following line.

```
COMPATIBILITY_INSTR=2.0
```

For more information on IBASIC for Windows and the compatibility file refer to *Installing and Using IBASIC for Windows*.

### HP BASIC

The HP BASIC editor checks for the syntax of the version of HP BASIC being used. Because IBASIC is a subset of HP BASIC it may not find all of the errors — the most common error is the use of HP BASIC commands that are not supported by IBASIC. For a listing of the commands supported by IBASIC refer to the *HP Instrument BASIC User's Handbook*.

## ASCII Word Processors

When an ASCII word processor is used to edit a program no syntax checking occurs until the program is loaded by the instrument. Another complication with using a word processor is that program line numbers are not automatically renumbered when new lines are inserted.

It is recommended that you renumber the program (REN 10,10), as described later in this chapter, to reduce the possibility of errors. (Errors in numbering lines usually do not result in a syntax error, they write over other program lines.)

---

## Using Edit

The editor is used for creating and altering lines in an IBASIC program. Those familiar with the editor found in IBASIC for Windows or HP Series 200/300 BASIC will find it somewhat similar to the instrument's IBASIC editor; others should find it easy to learn and use. This section tells you how to edit and enter an IBASIC program.

To start the editor, press the **Edit** softkey ( **F4** on an external keyboard) in the **SYSTEM OPTIONS** **IBASIC** menu. You will see the program, if one is in the buffer, appear on the display with a cursor on the first line of the program, as shown in Figure 5-1. If the program buffer is empty, the first line number 10 appears with the cursor positioned to begin entering text.

CALL	PRINT	ABORT	SUB	SUBEND	DATA	LOCAL	DIM	BIT	ABCDEFGHIJKLM	Edit
------	-------	-------	-----	--------	------	-------	-----	-----	---------------	------

```
10  |=====|
20  ! This program measures the transmission and
30  ! reflection characteristics of a bandpass filter
40  |=====|
50  ASSIGN @Hp8711 TO 800
60  ON KEY 0 LABEL "TRAN" CALL Transmission
70  ON KEY 1 LABEL "REFL" CALL Reflection
80  ON KEY 3 LABEL "SETUP" CALL Setup_diag
90  ON KEY 5 LABEL "EXIT" GOTO End_prog
100 LOOP
110   DISP "WAITING FOR SELECTION"
120 END LOOP
130 End_prog:DISP
140 END
150 |=====|
160 SUB Transmission
170 Transmission:!
180 OUTPUT @Hp8711:"CONF 'FILT:TRAN'"
190 OUTPUT @Hp8711:"DISP:ANN:FREQ1:MODE CSPAN"
200 OUTPUT @Hp8711:"SENS1:FREQ:CENT 175 MHZ"
```

Prior Menu

Figure 5-1. The HP IBASIC Program Editor

The HP 8711 editor is accompanied by a "Label Window" at the top of the screen. This window is filled with characters and IBASIC keyword commands and has its own cursor.

The current program line (the line containing the cursor) always appears as two lines on the screen, allowing you to enter up to 108 characters if needed. All other lines have only their first 51 characters displayed (excluding line numbers).

Each line has a numeric field in the first 6 columns in which program line numbers are right justified. Although program lines are automatically numbered by the editor, you can edit the current line number to copy or move it to a different location in the program. The range of line numbers is from 1 to 32767. To end an editing session press the **Prior Menu** softkey ( **F8** ) on an external keyboard) in the edit menu. This will return you to the

**SYSTEM OPTIONS** **IBASIC** menu.

## The IBASIC Editor

The editor has two sets of softkey menus, the **Edit** keys and the **Character Entry** keys. The edit menu is activated when you press **SYSTEM OPTIONS** **IBASIC** **Edit** . The menu box above the softkeys shows the label **Edit**.

The edit menu provides the following softkeys:

- **Insert Line** ( **F1** )
- **Insert Char** ( **F2** )
- **Delete Line** ( **F3** )
- **Recall Line** ( **F4** )
- **Delete Char** ( **F5** )
- **Enter** ( **F6** )
- **Prior Menu** ( **F8** )

The character entry menu is described in the "Editing from the Front Panel" section of this chapter.

## Recording into an Existing Program

One way to enter lines into your program is to use the keystroke recording capabilities of IBASIC. To record measurement sequences or other front panel operations into your program follow the procedure described below.

1. Activate the editor by pressing **SYSTEM OPTIONS** **IBASIC** **Edit** .
2. Use the step keys on the analyzer or the cursor keypad on an external keyboard to position the cursor on the line above which you want the recorded statements inserted.
3. Press **Prior Menu** to exit the editor.
4. Press **Key Record ON off** to activate keystroke recording.

5. Record the measurement sequence or front panel operation.
6. Press **SYSTEM OPTIONS** **IBASIC** **Key Record on OFF** to conclude the recording session.

The inserted recording acts the same as if you had pressed **Insert Lines** in the editor, and generated OUTPUT statements in insert mode.

---

**Note**

The ASSIGN @Hp8711 to 800 statement is **NOT** generated when you are recording into an existing program and **MUST** be included in your program prior to any recorded OUTPUT commands. If you initially created the program using recording, this statement should already exist. If it does not exist, you will need to enter it.

---

### Editing with an External Keyboard

With an external keyboard connected to the HP 8711, it is easy to edit or create an IBASIC program using the internal editor. Note that the Front Panel Editor described in the next section is always available, even when an external keyboard is in use.

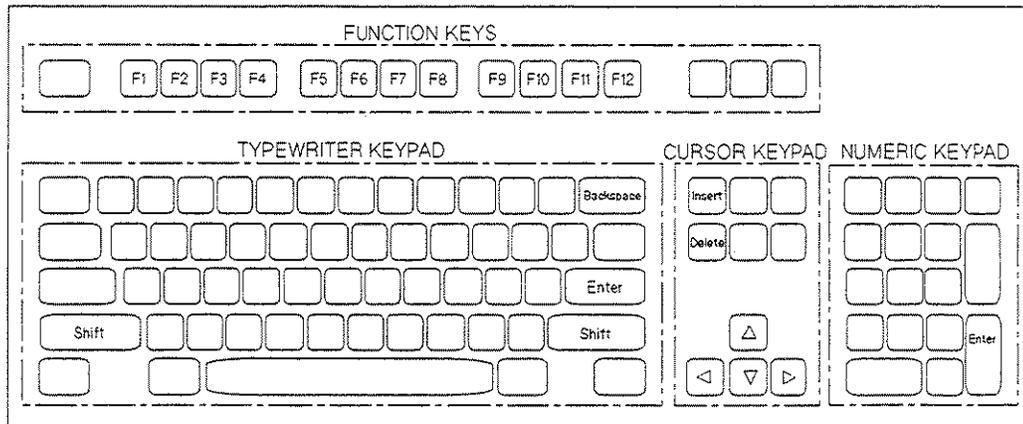
---

**Note**

The HP 8711 and the IBASIC editor work with PC-AT compatible keyboards (US only) that have a standard DIN interface. Foreign language keyboards will not cause an error, they simply will not be recognized as different from the US keyboard. A compatible keyboard can be purchased by ordering option 1CL with the HP 8711. The keyboard can be purchased alone by ordering the HP C1405 keyboard, option ABA (US English).

---

The PC-AT keyboard, Figure 5-2, has four major key areas: the typewriter keypad, the numeric keypad, the cursor keypad, and the function keys. Alphanumeric text can be entered using the typewriter and numeric keypads as needed. The cursor keypad can be used to move the cursor up/down a line or left/right to the next character positions. The function keys of the keyboard map to the softkeys on the HP 8711 front panel.



**Figure 5-2. The PC Keyboard**

Connect the keyboard to the rear panel DIN connector of the HP 8711 with the power off. Turn on the power, reset the instrument and load the IBASIC program to be edited. Select the **SYSTEM OPTIONS** **IBASIC** **Edit** menu and use the cursor keypad to position the cursor within the program for editing operations. The Page Up and Page Down keys on the keyboard scroll through the program quickly and easily.

### Inserting Lines

Insert one or more program lines above an existing line by placing the cursor on that line and pressing **Shift** + **Insert** on the keyboard (press both keys simultaneously). This key combination functions as a toggle to turn insert mode on and off.

As an example, assume you want to insert some lines between two adjacent program lines numbered 90 and 100. Place line 100 in the current line position and press **Shift** + **Insert**. The program display “opens” and a new line, number 91, appears between line 90 and line 100. Enter the inserted line and another inserted line, number 92, will appear. If, after continuing to enter lines in this manner, the inserted line number increments to 100, then the current line 100 will be renumbered one higher to accommodate the inserted line.

To stop inserting lines either press **Shift** + **Insert** again or use the cursor keys to move to another program line. Make sure you have entered any changes to your final inserted line before exiting the insert mode. Remember any changes you have made to the current line will be lost if you move the cursor to another line without pressing **Enter**.

### Editing Lines

Use the cursor keypad on the keyboard to move around the program for editing. The left and right arrow keys move within a program line while the up and down arrow keys move between lines. The alphanumeric keypad on the keyboard can be used for entering or editing text. Another key that is useful is the **Delete** key, which deletes the character highlighted by the cursor.

When you finish editing or changing a program line, store it into the program by pressing **Enter** on the keyboard. The computer checks the line for syntax errors and converts letter

case to the required form for names and keywords (IBASIC commands). If no errors are detected, it then stores the line in the program buffer.

## Entering Program Lines

When you finish entering or changing a program line, to store it into the program buffer you must ENTER it in one of four ways:

1. Use the **Enter** key on the front panel of the analyzer.
2. Use the **Enter** softkey on the instrument.
3. Use the **Enter** key on the external keyboard.
4. Use the function key on the keyboard ( **F6** ) that represents the **Enter** softkey.

The computer checks the line for syntax errors and converts letter case to the required form for names and keywords (IBASIC commands). If no errors are detected, it then stores the line.

---

### Note



If you edit or enter text on the current program line and then move off the line without pressing ENTER, all editing on the line will be lost.

---

## Editing from the Front Panel

Use the step keys to move the cursor up and down the lines in the program. When the cursor is located at the beginning of a line you want to change, use the knob to position the cursor within the line.

### Character Entry

The character entry menu and the associated label window are activated by pressing the **Insert Line** or **Insert Char** softkeys. The knob and step keys now move the cursor in the label window.

Use the knob or step keys to move the label window's cursor until it highlights the desired letter or keyword and press **Select Char/Word**. Continue editing until the line is correct. Press **Enter**. The computer checks the line for syntax and then stores it in the program if the syntax is correct. Press **Prior Menu** to return to the edit menu.

The character entry menu provides the following softkeys:

- **Select Char/Word** ( **F1** ) Inserts the character or word highlighted by the label window cursor at the position marked by the program cursor.
- **Space** ( **F2** ) Inserts a space at the position marked by the program cursor.
- **Delete Char** ( **F3** ) Deletes the character highlighted by the program cursor.
- **Backspace** ( **F4** ) Deletes the last character before the program cursor.
- **Enter** ( **F5** ) Enters the edited program line.
- **Prior Menu** ( **F8** ) Returns to the edit menu and de-activates the label window.

### The Label Window

The label window is a scrolling list of the most common characters, symbols and keywords used in IBASIC programming. It contains the uppercase alphabet, the numbers 0 to 9, symbols such as single and double quotation marks, parentheses, signs for mathematical and string operations as well as numerous other characters and symbols. It also contains the following IBASIC keywords:

ABORT	ENTER	NOT
ASSIGN	FOR	OUTPUT
BIT	GOTO	PAUSE
CALL	IF	PRINT
CLEAR	INPUT	SUB
DATA	INTEGER	SUBEND
DIM	LIST	THEN
DISP	LOCAL	TO
END	NEXT	WAIT

### Inserting Lines

To insert one or more program lines above any existing line, place the cursor on the existing line and press **Insert Line**. This causes the cursor to move to a new line that appears above the existing one. Enter and store the inserted line and another inserted line will appear. Remember, each line must be ENTERed or any changes will be lost when the cursor is moved to a different line.

### Removing Program Text

You can remove individual characters or entire lines from within the editor.

### Deleting Characters

The **Delete Char** softkey removes the character under the cursor and moves all characters to the left one place. Repeatedly pressing **Delete Char** will cause text to the right of the cursor to be removed one character at a time. The **Delete Char** softkey functions the same in both the line number and program statement fields. When used in the line number field, it deletes only line numbers to the right of the cursor (not program statement characters).

When using an external keyboard there are other keys that perform the same function as the **Delete Char** softkey. These are the **Delete** key in the cursor keypad and the function key that maps to the appropriate softkey, **F5** for the edit menu or **F3** for the character entry menu.

Another way to remove text on a line is by backspacing. Pressing the **- / ←** hardkey or the **Backspace** softkey on the front panel of the HP 8711 removes the letter to the left of the cursor and moves the cursor (and all characters to the right of the cursor) one space to the left. The **F4** function key or the **Backspace** key on the typewriter keypad of the external keyboard perform the same function. When the cursor is on a line number, using backspace simply moves the cursor back one position without deleting the number.

### Deleting Lines

The **Delete Line** softkey allows you to remove the current program line. When the current program line disappears, all subsequent lines in the display move up one line, but are not renumbered. The cursor maintains its column-relative position on the next highest numbered line.

If **Delete Line** is pressed when the cursor is on the last program line, the line text is removed but the line number remains with the cursor resting in the first column of line. This puts the editor in insert mode on the last line of the program (see "Inserting Lines"). (To get out of insert mode, simply move the cursor up one line.)

Pressing **Delete Line** will NOT remove a subprogram line with the SUB keyword in it unless all program lines belonging to that subprogram have already been deleted. A block of program lines can be deleted by executing the command `DELETE x,y` from an IBASIC command line (where x is the first line number in the block and y is the last line number).

When using an external keyboard there are other keys that perform the same function as the **Delete Line** softkey. These are **Shift** + **Delete** in the cursor keypad and the function key (**F3**) that maps to the **Delete Line** softkey in the edit menu.

### Recalling a Deleted Line

The last line that was deleted using **Delete Line** is buffered in the analyzer. To recall this line press the **Recall Line** softkey or **F4** on an external keyboard. Press **Enter** to restore the line to the program.

### Renumbering, Copying, Moving, and Indenting Lines

If you want to change the line number of an edited program line, simply move the cursor to the line number field and enter the line number you want. Changing the line number causes a copy operation, not a move. Therefore, if you only want to move the line, change the line number first, press **Enter** and then delete the original line. If you want to create an edited copy of the current line, edit the line and then change the line number and press **Enter**. The edits will only appear in the copied line.

If you are inserting a program line and you change the line number, the line will move to its new location when you ENTER it. The editor will remain in insert mode at the new location in the program.

You will notice that when the cursor is in the line number field, entries operate in an overtyping fashion rather than in the insert fashion as in the text portion of the program line. Also the  (backspace) key simply moves the cursor over line numbers without deleting the number.

---

**Note**

To renumber the entire program, IBASIC supports the RENumber command — BUT — you need an external keyboard to execute it. The command can be executed by following the steps listed below.

1. EXIT the edit mode by pressing **Prior Menu** until the **SYSTEM OPTIONS** **IBASIC** menu is active.
  2. Partition an IBASIC display as described next in this chapter.
  3. Enter the command **REN x,y** (where x is the new beginning line number and y is the increment) from the command line of the IBASIC display.
  4. Another way to “renumber” program lines with an external keyboard is to use the **COPYLINES** and **MOVELINES** commands. Use the **INDENT** command to make your code more readable.
- 

---

## Using **IBASIC Display**

Pressing the **SYSTEM OPTIONS** **IBASIC** **IBASIC Display** softkey ( **F7** on an external keyboard) allows you to allocate a partition of the analyzer’s display to be used by your program or, alternately, to return any allocated partition to the analyzer. The **IBASIC Display** menu contains the following softkeys:

- **None** ( **F1** )
- **Full** ( **F2** )
- **Upper** ( **F3** )
- **Lower** ( **F4** )
- **Prior Menu** ( **F8** )

The HP 8711 display is divided into two small partition areas (Upper and Lower) or one large area (Full), which encompasses both the Upper and Lower partition areas.

All screen output commands, such as **PRINT** and **DRAW**, require that you allocate a partition of the screen in order to view the results of the command. This can be performed in your program or interactively using the **IBASIC Display** softkey.

Allocating display partitions can be accomplished from within your program using the SCPI mnemonic "DISP:PROG" and specifying the parameter UPPER, LOWER or FULL. For example the statement

```
OUTPUT 800;"DISP:PROG FULL"
```

allocates the entire display, corresponding to selecting **Full** from the **IBASIC Display** menu.

An IBASIC display partition cannot occupy the same location as a measurement channel display. When an IBASIC display is partitioned it limits the amount of the CRT available to simultaneously show measurement data. Table 5-1 shows the **IBASIC Display** menu softkeys, their corresponding SCPI mnemonics, their functions and the measurement data that can be viewed when the display partition is allocated.

**Table 5-1. IBASIC Display Partitions**

SOFTKEY	SCPI MNEMONIC	ALLOCATES	VISIBLE DATA
<b>None</b>	DISPlay:PROG <sup>r</sup> am OFF	No Display	Channels 1 and 2
<b>Full</b>	DISPlay:PROG <sup>r</sup> am FULL	The Whole Display	None
<b>Upper</b>	DISPlay:PROG <sup>r</sup> am UPPer	Upper Channel Area	Channel 2 only
<b>Lower</b>	DISPlay:PROG <sup>r</sup> am LOWer	Lower Channel Area	Channel 1 only

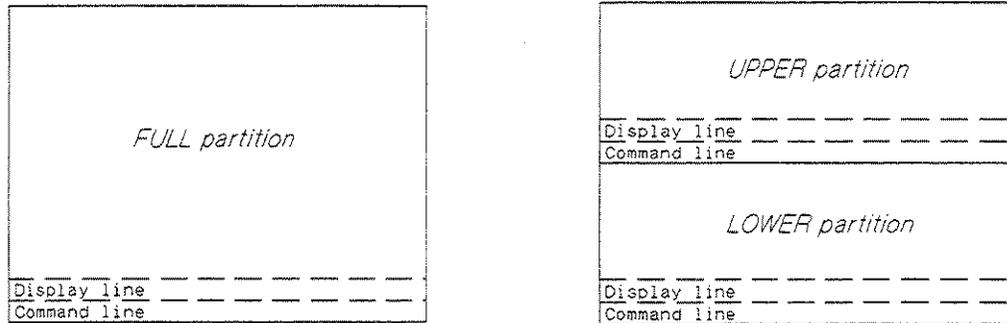
**Note**



When the UPPER or LOWER display partition is selected, the measurement display automatically selects the "split-screen" format. This format uses half of the CRT to display each channel's measurement data. Channel 1 data is always shown on the upper half of the screen, channel 2 data is shown on the lower half. The split-screen format allows measurement data to be viewed simultaneously with IBASIC program output. For more information about the split-screen format, or other parts of the measurement display, refer to the *HP 8711 Operating Manual*.

Most display allocation should be handled by your program via the SCPI mnemonics. These softkeys are best utilized during program development.

An IBASIC partition can be very useful during program development. It can be used to view program output, to query variables and to execute IBASIC commands (such as GET and REN) outside of your program. Figure 5-3 shows the relative size and location of the different IBASIC partitions and their command and display lines.



**Figure 5-3. The IBASIC display partitions**

More information about using display partitions within a program is available in chapter 7, “Graphics and Display Techniques.”

---

## Using Utilities

Pressing the **System Options** **IBASIC Utilities** softkey (F6 on an external keyboard) allows you to clear the program buffer, allocate memory for program use, or secure your program.

- Clear Program (F1)
- Memory Size (F2)
- Secure (F3)

Executing the **Clear Program** erases the current program buffer and frees all memory currently allocated. Memory size (see below) is reset to 8192 bytes. You will be prompted to ensure you do not accidentally erase the program.

**Memory Size** allows you to set stack memory to be used by your program. At power up it is set by default to 8192 bytes. However, when a program is (RUN), the 8711 will try to automatically set the Memory Size large enough to accommodate the program’s Stack and COM memory requirements.

For some programs the automatic memory sizing will be too small and you will get the message:

Error 2 in 100 Memory overflow

When this error occurs, you must manually set the **Memory Size** to the value in bytes required by your program, up to the available memory in your system.

**Secure** is used to secure lines of your program. Secured lines can not be listed, edited, or displayed. After you press this key you will see:

- Start Line # (softkey 1)
- End Line # (softkey 2)
- Perform Secure (softkey 4)

After you have set the start and stop line numbers, execute the **Perform Secure** operation.

---

**Caution**

Once you have secured your program lines, there is no way to remove the security. Do not secure the only copy of your program. Make a copy of your program, Secure the copy, and keep the original in a safe place. This prevents the unauthorized users from listing your program.

---

## Debugging Programs

---

The process of creating programs usually involves correcting errors. You can minimize these errors by using keystroke recording for measurements and other front panel sequences and by writing structured, well-designed programs.

Of course bugs can and do appear in even the best designed programs and IBASIC contains some features that can help you to track them down. Some IBASIC capabilities useful for program debugging are simple and, used properly, can be very helpful. Some of these capabilities are:

- RUN or CONTINUE your program
- STEP through your program, executing one line at a time
- Display the last error encountered in your program
- Examine program variables

By examining the values assigned to variables at various places in the program, you can get a much better idea of what is really happening in your program.

By inserting a PAUSE statement in your program you can pause the program at any line and then examine the values of variables at that point in the program. You can then press **Continue** in the **SYSTEM OPTIONS** **IBASIC** menu to resume operation to the next PAUSE statement (or the program end).

These capabilities can be used together to effectively examine the program's operation and solve your particular problems.

---

### Note



Most of the debugging techniques described in this chapter make use of an external keyboard. The HP 8711 and the IBASIC editor work with PC-AT compatible keyboards (US only) that have a standard DIN interface. Foreign language keyboards will not cause an error, they simply will not be recognized as different from the US keyboard. A keyboard can be ordered with the HP 8711 network analyzer by ordering option 1CL.

---

---

## Setting Breakpoints

A common method of debugging a program involves the use of breakpoints. A breakpoint causes the program to stop before executing a specified line so that you can examine the program state at that point. In IBASIC this can be accomplished by inserting PAUSE statements in the program code. Note that PAUSE is one of the IBASIC keywords included in the editor's label window (described in chapter 5, "Developing Programs"). When the program is then run, you can use the command line to check or change variable values.

Execution of the program can be resumed in one of two ways.

- Press **Step** ( **F3** on an external keyboard) to execute next program line.
- Press **Continue** ( **F2** on an external keyboard) to continue the program until the next PAUSE, STOP or END statement is encountered.

---

## Examining Variables

To examine a variable it is necessary to pause the program. Pausing the program can be accomplished by pressing the **Pause** softkey ( **F8** on an external keyboard) that is available when a program is running, or by inserting a PAUSE statement in your program.

A command line becomes active when an IBASIC program is paused or stopped and an IBASIC display partition is present. (For information on creating an IBASIC display partition, see "Using IBASIC Display" in chapter 5, "Developing Programs.") You may also activate the command line when no IBASIC window is partitioned by pressing the **ESC** key on the external keyboard. A cursor will appear in the lower left portion of the screen when the command line is active. Strike the **ESC** key again to de-activate. Once the command line is active, a variable can be examined in two ways. Both methods require the use of an external keyboard.

1. Enter the variable name (without a line number) on the command line. This results in the value assigned to that variable being shown in the display line of the IBASIC window.
2. Execute the command PRINT Value from the command line (where Value is the name of the variable being examined). This results in the value assigned to that variable being shown on the print screen of the IBASIC window.

To examine a variable without accessing a command line it is necessary to add the statement PRINT Value (or DISP Value) to the program before the PAUSE statement that temporarily stops the program. PAUSE, PRINT and DISP are all keywords that are included in the IBASIC editor's label window (see chapter 5, "Developing Programs" for a description of the label window).

---

**Note**

An IBASIC display partition must be active to view the results of a PRINT statement or to access a command line. The display line (accessed with the DISP command) is available even when no IBASIC display is present.

---

## Examining Strings

Enter string variables as you would any other variable. Any string variable entered without delimiters will display as much of the string as will fit on the display line of the screen (up to 58 characters).

To select only a section of a string, use the IBASIC substring syntax (see the “HP Instrument BASIC Programming Techniques” section of the *HP Instrument BASIC Users Handbook*). For example, to examine the 7 character substring starting at the second character of A\$ enter A\$[2;7] on the command line or execute the command PRINT A\$[2;7].

## Examining Arrays

To select an array to be examined you can either select individual elements or the entire array. For example the entry:

```
I_array(1),I_array(2),I_array(3)
```

selects the elements 1 through 3 of the array I\_array to be displayed.

You may select an entire array to be examined by entering the array variable name and specifying a wildcard (\*) for the element (such as I\_array(\*)). If I\_array(20) is an integer array, and the first and second elements are set to 100, entering I\_array(\*) would display:

```
100 100 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Individual array elements (e.g., I\_array(17)) can also be specified in the same way as any other single variable.

---

## Displaying the Last Error Encountered

It is sometimes useful to review the last error encountered by a program that is being run. This is done from the command line by examining the value assigned to the variable name ERRM\$. This value will include the error number and message of the last error encountered by the program.

An additional method of displaying the error message is to use an error trapping subroutine.

For example, insert the following line at the beginning of a program.

```
ON ERROR GOSUB Errormsg
```

The subroutine Errormsg should then be included at the end of the program (after execution is stopped but before the END command).

```
100 Errormsg: !  
110 DISP ERRM$  
120 PAUSE  
130 RETURN
```

The error message is automatically shown on the display line of the IBASIC window and program execution is paused when an error message is encountered.

## Graphics and Display Techniques

The HP 8711 has two measurement channels which can be displayed simultaneously. The instrument's screen can be split into two trace areas for this purpose (upper for channel 1 and lower for channel 2). Additionally, the two measurements can be overlaid onto one full size screen (the default setting).

IBASIC programs have the ability to allocate portions of the instrument's display for program output, including text and graphics. This section provides a description of the various programming techniques used to do both. Any of the three measurement display areas, called display partitions, can be used by an IBASIC program. These partitions are shown in Figure 7-1.

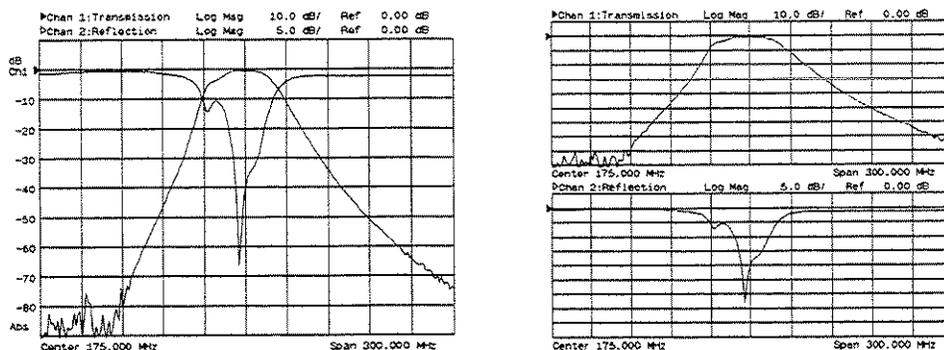


Figure 7-1. Display partitions on the HP 8711

In this manual the term **Measurement Window** is used to refer to a part of the instrument's display when it is being used to show measurement data. The part of the instrument's display allocated for use by IBASIC is referred to as an **IBASIC Window**.

### Using the Partitions

Many IBASIC commands (such as PRINT, DISP, CLEAR SCREEN, MOVE, DRAW and GCLEAR) require a CRT as an output device. These commands output data to the screen by writing to a screen buffer. Since IBASIC programs share all the hardware resources with the instrument, the display must be shared for instrument and program use.

In order to view this output buffer, a portion of the display must be released from the instrument. When no program is running, you can do this manually, using the **SYSTEM OPTIONS** **IBASIC** **IBASIC Display** softkey menu. To do this within a running program requires sending a command to the analyzer both to borrow a part of the display and again to return it for the instrument's use.

This process is called the allocation of display partitions. Manual allocation of display partitions is described in chapter 5, "Developing Programs." Table 7-1 below includes a summary of the available partitions, their locations and the SCPI mnemonic used to select each partition.

**Table 7-1. IBASIC Display Partitions**

SOFTKEY	SCPI MNEMONIC	ALLOCATES
None ( F1 )	DISPlay:PROGram OFF	No Display
Full ( F2 )	DISPlay:PROGram FULL	The Whole Display
Upper ( F3 )	DISPlay:PROGram UPPer	Upper Channel Area
Lower ( F4 )	DISPlay:PROGram LOWEr	Lower Channel Area

## Allocating Partitions

To request a display partition from the analyzer for use by an IBASIC program, send the instrument the corresponding SCPI mnemonic. "DISP:PROG UPPer" allocates the upper partition, "DISP:PROG LOWEr" allocates the lower partition, and "DISP:PROG FULL" allocates the full screen partition.

For example, to print a message to the upper partition area, you might use a program segment like this:

```

30 ASSIGN @Hp8711 TO 800
40 OUTPUT @Hp8711;"DISP:PROG UPPer"
50 CLEAR SCREEN
60 PRINT "This is the upper partition"

```

To be sure that you are not writing to a partition that has not yet been assigned, you should include a WAIT statement or, even better, add a SCPI query command followed by an ENTER statement to synchronize the program with the instrument. The previous example might then look like this:

```

30 ASSIGN @Hp8711 TO 800
40 OUTPUT @Hp8711;"DISP:PROG UPPer"
42 OUTPUT @Hp8711;"DISP:PROG?"
44 ENTER @Hp8711;Screen$
46 IF Screen$<>"UPP" THEN GOTO 42
50 CLEAR SCREEN
60 PRINT "This is the upper partition"

```

The mnemonic DISP:PROG? (line 42 above) requests the instrument to send the current partition status. The ENTER statement on the next line reads that status and then continues.

## De-Allocating Partitions

To return the display partition to the analyzer for use as a measurement screen, use the "DISP:PROG OFF" mnemonic. This should be done before the termination of any program that has allocated a display partition. It may also be required within the program to allow the user to view instrument measurement data. The following example demonstrates this command:

```
830 OUTPUT @Hp8711;"DISP:PROG OFF"
```

## Operation with No Partition

IBASIC programs can also access the analyzer's display when no partition has been allocated. This can be done through the use of certain areas of the screen. One of these areas is to the right of the measurement display. This area is reserved for softkey labels. It can be accessed using the ON KEY statement.

A second area is a display line (or command line) that appears when no part of the display is allocated for use by IBASIC. This display line, which is located at the lower left corner of the active channel graticule, appears when needed by the INPUT or DISP commands or when activated. To activate the command line, press **ESC** on an external keyboard. Figure 7-2 shows an example of the use of this display line. When the INPUT command is being used, the IBASIC editor's label window and character entry softkey menu appear. Refer to chapter 5, "Developing Programs," for a description of the IBASIC editor.

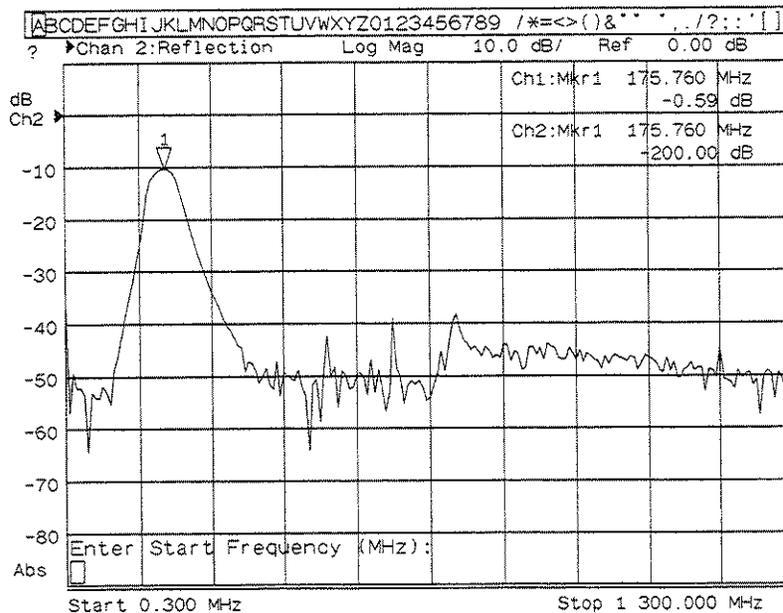


Figure 7-2. Using INPUT with no display partition

In addition to the commands described above, the HP 8711 has "User Graphics" commands that can write to any of the display partitions. These commands can be used to write to measurement windows as well as the IBASIC window. These commands are described in the "SCPI Graphics Commands" section of this chapter.

---

## Displaying Text

Most of IBASIC's text capabilities are covered in detail in the "HP Instrument BASIC Programming Techniques" section of the *HP Instrument BASIC Users Handbook*. The PRINT statement works the same way in every display partition. Information is printed starting at the top left corner of the current partition and continues until the display line of the partition is reached. The screen then scrolls up to allow additional lines to be printed. Figure 7-3 shows the different display partitions and the location of text printed to them. Note that causing the screen to scroll does not affect any graphics displayed on the screen, since text and graphics are written to different planes of the display.

All partitions have a width of 58 characters. The height varies according to partition. Both upper and lower partitions contain 10 lines, while the full partition contains 22 lines.

This information is useful if you are using the PRINT TABXY statement to position text. For example, the following program segment prints a message in the center of the full partition (assuming it has been allocated earlier in the program).

```
100 Maxlines=22
110 Tabx=(58-LEN("This is CENTERED text.))/2
120 PRINT TABXY(Tabx,Maxlines/2);"This is CENTERED text."
```

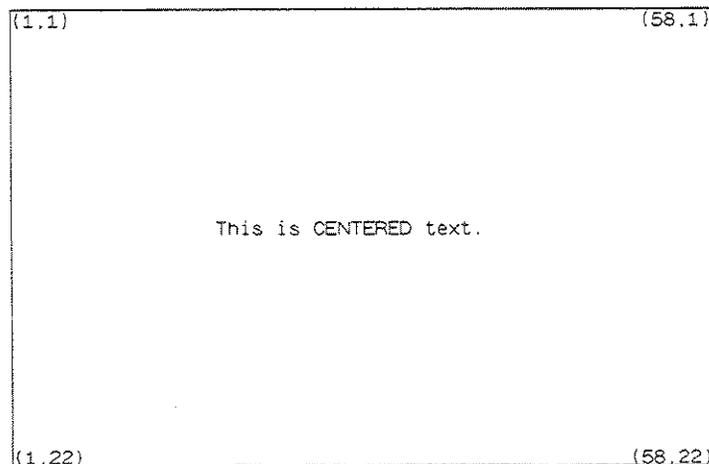


Figure 7-3. Printing to a display partition

A useful technique to get text onto the screen quickly is to write your display message to a long string using the OUTPUT statement, and then print the string to the screen. This speeds up screen display time considerably. The following program segment demonstrates this:

```

60 DIM Temp$(100),Big$(2000)
70 OUTPUT Temp$;"This is the first line of text"
80 Big$=Big$&Temp$
90 OUTPUT Temp$;"This is the second line of text"
100 Big$=Big$&Temp$
110 PRINTER IS CRT; WIDTH 2000
120 PRINT Big$

```

You can also print to the screen using the OUTPUT statement in conjunction with the display address (1). For example, line 150 below writes a string to the screen.

```

150 OUTPUT CRT;"OUTPUT 1 WORKS WELL TOO"

```

## Graphics Initialization and Scaling

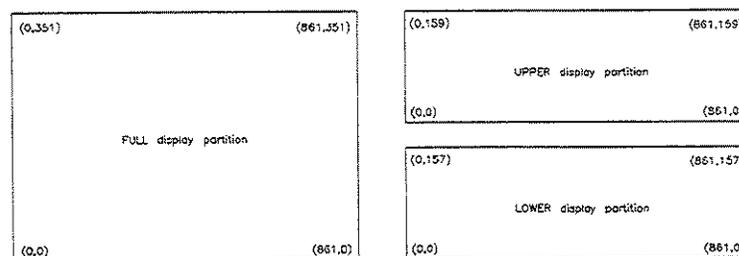
In all partitions, display coordinate 0,0 is at the bottom left corner and clipping occurs automatically if the X,Y coordinate exceeds the displayable range of the current partition. Figure 7-4 shows the different partitions and the pixel dimensions (GESCAPE values) for each.

After a GINIT command, the display is dimensioned as 100 GDU's (Graphical Display Units) high and 245 GDU's wide (assuming full partition). This gives a RATIO result of 2.45 and provides the same results as issuing a WINDOW 0,245,0,100 command. In order to prevent circles from appearing oval shape, this ratio should be maintained. You can also issue a WINDOW 0,861,0,351 command. This will maintain the same ratio but the display will now be dimensioned in actual pixel unit. This may be more useful than the default GINIT values since fractional display units are not needed, allowing integers only to be used; thus speeding execution. These are also the same values that are returned by utilizing the GESCAPE command (see BAR program example). The GESCAPE command will always return the current pixel dimension sizes. Because the results of this command can vary drastically with partition size, you must first partition the display BEFORE executing the GINIT and GESCAPE commands.

### Note



Upon power up, the default display coordinates are 0,861,0,351 and will remain that until a GINIT is performed. It is recommended that a GINIT command always be part of any graphics program and that it be executed only after the display partition is set.



pd63o

Figure 7-4. Pixel Dimensions with Available Display Partitions

---

## Using Graphics

IBASIC's graphics commands are easy to understand and use. You can use the `MOVE` statement to move the "pen" to a specific pixel location (without drawing) and then draw a line from the current pen location to another pixel coordinate using the `DRAW` statement. The `GCLEAR` statement removes all graphics.

The `PEN` command provides an easy method of erasing lines drawn by the `DRAW` command. When `PEN 1` is issued (the default state), all `DRAW` commands act normally, drawing a line with the full intensity. When `PEN 0` is issued, all `DRAW` commands erase any pixels their path encounters. Where there are no lines in the path, no change is visible. As an example of using the `MOVE` and `DRAW` commands, the following statement moves the logical pen to a point 100 units to the right of, and 150 units above, the lower left corner of the display:

```
100 MOVE 100,150
```

This statement then draws a line to coordinates (200,10):

```
110 DRAW 200,10
```

Finally, these two statements erase the previously drawn line:

```
120 PEN 0
130 DRAW 100,150
```

As with text output, the program has to be assigned a partition before graphics can be viewed. Text and graphics output to a partition appear on separate planes.

---

## Drawing Figures

Some IBASIC keywords listed below may be used to simplify drawings and setup diagrams. See also, the paragraph below titled "Graphics Exceptions".

- `POLYGON` - Draws all or part of a regular polygon
- `RECTANGLE` - Draws a rectangle
- `LABEL` - Produces alphanumeric labels
- `CSIZE` - Sets size and aspect ratio of labels
- `LDIR` - Defines the angle at which a label is to be drawn
- `LORG` - Defines the relative origin of a label

These keywords are used in the "BAR" program example listed in Chapter 9, "Example Programs", and on the IBASIC Example Programs Disk. The keywords appear in the subprograms 'Box', 'Circle', and 'Label' described below.

```
1620 ! Draw a box in the active IBASIC partition
1621 ! Xpos,Ypos specify the CENTER of the box
1622 ! Xsize,Ysize are width and height dimensions
1623 ! Sc is a scaling factor for the figure being drawn
1624 ! 1.79 is a correction factor used by the 8711 only
1630 Box:SUB Box(Xpos,Ypos,Xsize,Ysize)
```

```

1640  COM /Scale/ Sc,INTEGER X,Y
1650  MOVE X+(Xpos-Xsize/2)*Sc,Y+(Ypos-Ysize/2)*Sc
1660  RECTANGLE Xsize*Sc,Ysize*Sc*1.79 !8711 Pixel H:W Ratio
1670  SUBEND

1681  ! Draw a circle in the active IBASIC partition
1682  ! Xpos,Ypos specify the center of the circle
1683  ! Radius is the size of the circle
1684  ! Sc is a scaling factor for the figure being drawn
1690  Circle:SUB Circle(Xpos,Ypos,Radius)
1700  COM /Scale/ Sc,INTEGER X,Y
1710  MOVE X+Xpos*Sc,Y+Ypos*Sc
1720  POLYGON Radius*Sc,16,16
1730  SUBEND

1890  ! Creates a label in the active IBASIC partition
1891  ! Text$ is the alphanumeric label
1892  ! Xpos,Ypos is the position of the label
1893  ! Lorg references the label orientation to Xpos,Ypos
1894  ! Ldr is the angle in which the label will be drawn
1895  ! Pen is pen number (0 erases)
1896  ! Sc is a scaling factor for the figure being drawn
1900  Label:SUB Label(Text$,Xpos,Ypos,Size,Lorg,Ldr,Pen)
1910  COM /Scale/ Sc,INTEGER X,Y
1920  LORG Lorg
1930  LDIR Ldr
1940  CSIZE Size*Sc,1
1950  MOVE X+Xpos*Sc,Y+Ypos*Sc
1960  PEN Pen
1970  LABEL Text$
1980  PEN 1
1990  SUBEND
2000  !

```

The following program displays a "HELP" screen and demonstrates many of the techniques discussed so far. Running this program produces the screen display shown in Figure 7-5.

```

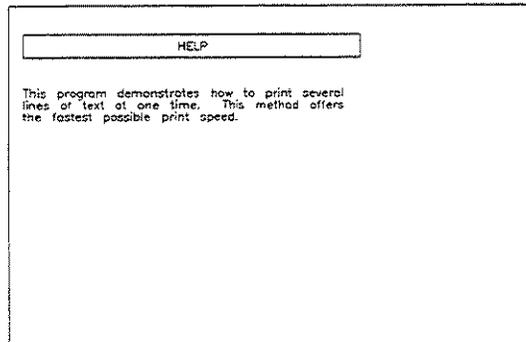
10  DIM A$[58],String$[1000]
20  ASSIGN @Hp8711 TO 800
30  OUTPUT @Hp8711;"DISP:PROG FULL;*WAI"
40  GINIT
50  GCLEAR
60  MOVE 0,89
70  RECTANGLE 200,14
80  PRINT TABXY(24,2);"HELP"
90  OUTPUT A$;"This program demonstrates how to print several"
100 String$=String$&A$
110 OUTPUT A$;"lines of text at one time. This method offers"
120 String$=String$&A$
130 OUTPUT A$;"the fastest possible print speed."
140 String$=String$&A$

```

```

150  PRINTER IS CRT;WIDTH 1000 ! Prevent auto cr/lf
160  PRINT TABXY(1,5);String$
170  END

```



pd62a

Figure 7-5. "HELP" program output

## GRAPHICS EXCEPTIONS

The following graphics commands do not conform to the keyword description found in the *HP Instrument BASIC Users Handbook*:

**VIEWPORT** - Does not create isotropic units that are physically square. Does not soft clip the display area.

**CLIP** - The HP 8711 does not support graphics clipping.

**SHOW** - Does not create isotropic units.

**POLYLINE, POLYGON, RECTANGLE, RPLOT** - The HP 8711 does not support the **FILL** or **EDGE** options. Also see next paragraph.

## GRID, RECTANGLE, POLYGON, and POLYLINE scaling differences

When the display is initialized using **GINIT**, the display will be scaled to a height of 100 GDU's and a width of 245 GDU's. The ratio is 2.453 and the pixel height-to-width ratio is fixed at 1.79 (non square pixels). This can cause scaling difficulties if not well understood, and will produce different results than is seen on HP BASIC computers or workstations. The following examples should help clarify some scaling issues.

After **GINIT**, performing a **GRID 10,10** command will produce a grid array 10 high and 24.5 wide. The individual grids will be rectangular (taller than wide). To produce square grids, perform a **GRID 10\*1.79,10** command. This will produce square grids; 10 high and just under 14 wide. If you move the starting point to approximate center (**MOVE 120,50**) and request a square 55 wide by 55 high (**RECTANGLE 55,55**), the HP 8711 will automatically scale this so as to appear square. The width will be 55 GDU's but the height will be 55/1.79 or 30.7 units high. This will appear square and is quite a different result than would be obtained by attempting to plot a "square" 55 units on each side; this would instead, produce a rectangle.

A similar scaling is done with the **POLYGON** command. If a **POLYGON 80** command is given, the HP 8711 will produce a circle with a horizontal radius of 80, but with a vertical radius of 44.7 GDU's; even so, it will appear circular.

The following is a simple rule to remember with GINIT values (or the equivalent WINDOW ratio) on the HP 8711: The HP 8711 will produce circles with the POLYGON/POLYLINE command and squares with the RECTANGLE command (assuming equal x,y) in all cases. However, the radius or width (in GDU's) will be accurate only in the horizontal axis and will be 1.79 times LESS in the vertical axis.

Try this simple program to demonstrate the above examples. Un-comment line 60 and comment out line 50 to show the difference in the two GRID statements. The rectangle may be hard to see since it will partially lie on a gridline; its lower left corner is at the centered dot.

```
10  ASSIGN @Hp8711 TO 800
20  OUTPUT @Hp8711;"DISP:PROG FULL"
30  GINIT
40  GCLEAR
50  GRID 10,10      ! makes rectangular grids
60  ! GRID 10*1.79,10 ! makes square grids
70  MOVE 120,50    ! move to center
80  POLYGON 1      ! make small dot
90  RECTANGLE 55,55 ! makes square
100 POLYGON 80
110  END
```

### Labeling with Different Partitions

The LABEL command may be used to label graphs, however, the following should be noted. Labels that may be of the correct size for a full screen partition will appear half as big if a GINIT is performed after the 8711 has been set to either the upper or lower half partition. This is because the CSIZE command scales according to display height, not width. Since the display height is one-half, the character size will also be one-half. Labels that are scaled properly for full screen displays will not be scaled properly for half screen displays and vice-versa.

---

## HP 8711 User Graphics Commands

In addition to the commands described earlier in this chapter, there are several SCPI mnemonics that can be used to create graphics and messages on the display of the HP 8711.

These commands are instrument specific mnemonics, not standard IBASIC commands. They are also different from the previously described IBASIC commands in that they do not require an IBASIC display partition. This means that they can be used to write or draw directly to a measurement window.

These commands, listed in Table 7-2 are SCPI mnemonics and are programmable from an external controller as well as from IBASIC. The commands are of the form

```
DISPlay:WINDow[1|2|10]:GRAPhics:<command>.
```

The number specified in the WINDow part of the command selects where the graphics are to be written.

WINDow1 draws the graphics to the channel 1 measurement window.  
 WINDow2 draws the graphics to the channel 2 measurement window.  
 WINDow10 draws the graphics to an IBASIC display partition.

**Note**



When SCPI graphics commands are used to write directly to a measurement window they write to the static graphics plane (the same plane where the graticule is drawn). There is no sweep-to-sweep speed penalty once the graphics have been drawn.

**Table 7-2. SCPI graphics commands**

SCPI Command	Function
DISP:WIND[1 2 10]:GRAP:CIRC <radius>	Draws a circle of the specified radius (in pixels) centered at the current pen location using the current pen color
DISP:WIND[1 2 10]:GRAP:CLEAR	Clears the user graphics
DISP:WIND[1 2 10]:GRAP:COLOR <num>	Sets the color of the user graphics pen — choose from 1 to draw, 0 to erase
DISP:WIND[1 2 10]:GRAP:DRAW <x>,<y>	Draws a line from the current pen location to the specified pen location (in pixels) using the current pen color
DISP:WIND[1 2 10]:GRAP:LABEL <text>	Prints a label with lower left corner at the current pen location using the current font type
DISP:WIND[1 2 10]:GRAP:LABEL:FONT <font>	Sets the text font —choose from BOLD, SLANT, SMALL, NORMAL, HBOLD, ... (use of H before a font type highlights the text)
DISP:WIND[1 2 10]:GRAP:MOVE <x>,<y>	Moves to the specified location (in pixels)
DISP:WIND[1 2 10]:GRAP:RECT <xsize>,<ysize>	Draws a rectangle of the specified size (in pixels) with lower left corner at the current pen location using the current pen color

For more information about the HP 8711's user graphics commands, refer to the *HP-IB Programming Guide* and the *HP-IB Command Reference*. To see graphics example programs, refer to chapter 11, "Example Programs".

## Interfacing with External Devices

---

This section describes the techniques necessary for programming the HP-IB interface. It describes how this interface works and how to use it to control or interface with systems containing various HP-IB devices. It also describes how to interface with external devices using the serial and parallel interfaces.

The HP-IB interface is Hewlett-Packard's implementation of the IEEE-488.1 Digital Interface for Programmable Instrumentation. The acronym HP-IB stands for "Hewlett-Packard Interface Bus," and is often referred to as the "bus." The interface is easy to use and allows great flexibility in communicating data and control information between an HP Instrument BASIC program and external devices.

IBASIC is essentially an HP-IB instrument controller residing inside an instrument. It uses the instrument's HP-IB interface for external communication and an internal HP-IB interface to communicate with the instrument. This unique arrangement presents a few differences between IBASIC's implementation of HP-IB control and the standard HP IBASIC for Windows or HP Series 200/300 BASIC controllers. A description of the interaction of IBASIC with the host instrument and the external HP-IB interface is given in the section entitled "The IBASIC HP-IB Model."

---

### Communication with Devices

#### HP-IB Device Selectors

Since the HP-IB allows several devices to be interconnected, each device must be uniquely identified. Specifying the select code of the HP-IB interface (such as 7 or 8) to which a device is connected is not enough to uniquely identify each specific device on the bus.

Each device on the bus has a primary address that identifies it. This address can be set by the user. It must be unique to allow individual access of each device. When a particular HP-IB device is to be accessed, it must be identified with both its interface select code and its bus address.

The interface select code is the first part of an HP-IB device selector. IBASIC programs run inside an instrument and communicate with it over an internal bus (interface select code 8). IBASIC programs can also communicate with external devices using the instrument's HP-IB interface (select code 7).

The second part of an HP-IB device selector is the device's primary address, an integer in the range of 0 through 30. For example, to specify the device on the interface at select code

7 with a primary address of 22, use device selector 722. Secondary HP-IB addressing is also supported for those devices requiring it. These devices will have at least 5-digit service selection such as 72201.

Since the HP 8711 is the only device on the internal interface, its primary address on that interface is arbitrary and the instrument will respond to any primary address with a select code equal to 8 (e.g., 800, 811, 822, etc.).

---

**Note**

Each device's address must be unique. The HP 8711 is shipped from the factory with a primary address of 16. No other device on the bus should use the same address.

The procedure for setting the address of an HP-IB device is given in the installation manual for each device. To set the address of the HP 8711, use the **HP8711 Address** softkey in the **SYSTEM OPTIONS** **HP-IB** menu, or the SCPI mnemonic **SYST:COMM:GPIB:ADDR**.

---

## Moving Data Through the HP-IB

Data is output and entered into the program through the HP-IB with the **OUTPUT** and **ENTER** statements, respectively. The only difference between the **OUTPUT** and **ENTER** statements for the HP-IB and those for other interfaces is the addressing information within HP-IB device selectors.

### Examples

```
100 Hpib=7
110 Device_addr=22
120 Device_selector=Hpib * 100 + Device_addr
130 !
140 OUTPUT Device_selector;"F1R7T2T3"
150 ENTER Device_selector;Reading

320 ASSIGN @Hpib_device TO 702
330 OUTPUT @Hpib_device;"Data message"
340 ENTER @Hpib_device;Number

440 OUTPUT 800;"SOUR1:POW -10 dBm"

480 ENTER 724;Readings(*)
```

## General Structure of the HP-IB

Communications through the HP-IB are made according to a precisely defined standard (the IEEE 488.1 standard). The rules set by IEEE 488.1 ensure that orderly communication takes place on the bus. For more information about the structure of the HP-IB and the IEEE 488.1 standard, refer to the *Tutorial Description of the Hewlett-Packard Interface Bus*.

Devices that communicate over the HP-IB perform one or more of the following three functions.

- Talk — send data over the bus
- Listen — receive data over the bus
- Control — control the exchange of data on the bus

### The System Controller

The controller is a device that has been designated to control the communication occurring on the bus. It specifies which device talks, which device listens and when the exchange of data takes place.

An HP-IB system can have more than one device with the ability to control the bus, but only one of these devices is allowed to control the exchange of data at any given time. The device that is currently controlling the exchange of data is called the **Active Controller**.

One device must be able to take control of the bus even if it is not the active controller. The device designated as the **System Controller** is the only device with this ability. To designate the HP 8711 as the system controller use the **System Controller** softkey in the **SYSTEM OPTIONS** **HP-IB** menu.

The system controller is generally designated before running a program and should not be changed under program control. An exception to this is when an IBASIC program is running on the HP 8711's internal controller. If the IBASIC program controls other HP-IB devices, the analyzer must be designated as the system controller.

A SCPI mnemonic `SYST:COMM:GPIB:CONT <ON|OFF>` can be used to make the HP 8711 the system controller. Program execution should be carefully synchronized, using `*OPC?` and waiting for a reply before any `OUTPUT 7xx` command is sent. (Refer to the "Synchronization" section of the *HP-IB Programming Guide* for more information on the `*OPC?` command.)

### Using the Serial and Parallel Ports

The HP 8711 has two additional ports that can be used to control peripherals, material handlers or other devices. Active control of the HP-IB interface is not needed when these ports are being used. These ports are a parallel port and a serial port for use with hardcopy output to non-HP-IB printers and plotters.

In addition to the serial and parallel ports, there are also two BNC connectors on the rear panel of the HP 8711. These connectors provide access (using TTL signal levels) to two programmable bits.

- Limit Test Result bit — indicates the results of a pass/fail limit test
- User bit — to be used as needed (for example to use with a foot pedal)

IBASIC programs can be used to directly control all of these ports and bits. This control can be accomplished using the READIO and WRITEIO commands as described below.

WRITEIO	9,0	Outputs a byte to the serial port. The byte is output serially according to the configuration of the serial port (baud rate and handshake settings).
WRITEIO	15,0	Outputs 8-bit data to the D0 through D7 lines of the parallel port. D0 is the least significant bit, D7 is the most significant bit. Checks the status lines for <ul style="list-style-type: none"><li>• Out of Paper</li><li>• Printer Not on Line</li><li>• BUSY</li><li>• ACKNOWLEDGE</li></ul>
WRITEIO	15,1	Sets/clears the "user" bit according to the least significant bit (LSB) of the data written. An LSB of 1 sets the bit high, while an LSB of 0 clears the bit.
WRITEIO	15,2	Sets/clears the "limit test result" bit according to the least significant bit (LSB) of the data written. An LSB of 1 sets the bit high, while an LSB of 0 clears the bit.
WRITEIO	15,3	Outputs 8-bit data to the D0 thru D7 lines of the parallel port. Do is the least significant bit, D7 is the most significant bit. Does not check the status lines.
READIO	9,0	Reads the serial port.
READIO	15,0	Reads the 8-bit data port, D0 thru D7.
READIO	15,1	Reads the 8-bit status port <ul style="list-style-type: none"><li>• D0 - Acknowledge</li><li>• D1 - Busy</li><li>• D2 - Out of Paper</li><li>• D3 - On Line</li><li>• D4 - Printer Error</li><li>• D5 - TTL Limit Test Result bit</li><li>• D6 - TTL User bit</li><li>• D7 - HP-IB System Controller</li></ul>

An example program, REPORT, demonstrating peripheral control over the parallel port is provided in chapter 9, "Example Programs."

---

## General Bus Management

The HP-IB standard provides several mechanisms that allow managing the bus and the devices on the bus. Here is a summary of the IBASIC statements that use these control mechanisms.

**ABORT** — abruptly terminates all bus activity and resets all devices to their power-on HP-IB states.

**CLEAR** — sets selected (or all) devices to a pre-defined, device-dependent HP-IB state.

**LOCAL** — returns selected (or all) devices to local (front panel) control.

**LOCAL LOCKOUT** — disables selected (or all) devices' front panel controls.

**REMOTE** — puts selected (or all) devices into their device-dependent, remote modes.

**SPOLL** — performs a serial poll of the specified device (which must be capable of responding).

**TRIGGER** — sends the trigger message to a device (or selected group of devices).

These statements (and functions) are described in the following discussion. However, the actions that a device takes upon receiving each of the above commands are, in general, different for each device. For external devices, refer to the particular device's manuals to determine how it will respond.

All of the bus management commands, with the exception of **ABORT**, require that the program be the active controller on the interface. A running IBASIC program is always active controller on the internal interface (select code 8). For the program to be active controller on the external interface (select code 7), the instrument must either be set as system controller or have control passed to it from an external controller. The program automatically assumes the controller status of the host instrument. For more information refer to "The IBASIC HP-IB Model" section later in this chapter.

---

### Note

In this section the term **Host Instrument** refers to the instrument where the IBASIC controller is located.



---

### REMOTE

Most HP-IB devices can be controlled either from the front panel or from the bus. If the device's front panel controls are currently functional, it is in the Local state. If it is being controlled through the HP-IB, it is in the Remote state. Unless operating in the Local Lockout mode, each HP-IB device has method (usually a key) to return itself to Local (front panel) control.

When HP 8711 is being controlled by a program running on an external controller, the **Return to Local** softkey is always available to return the analyzer to Local control.

The Remote message is automatically sent to all devices whenever the system controller is powered on, reset, or sends the Abort message. A device also enters the Remote state automatically whenever it is addressed. The **REMOTE** statement also outputs the Remote

message, which causes all (or specified) devices on the bus to change from local control to remote control. The host instrument must be designated as the system controller before an IBASIC program can execute the REMOTE statement on select code 7.

#### **Host Instrument**

The REMOTE statement has no effect on the host instrument since it is always in remote control whenever an IBASIC program is running. Specifying the internal interface in a REMOTE statement will not generate an error, but will have no effect.

## **LOCAL LOCKOUT**

The Local Lockout message effectively locks out the "local" switch present on most HP-IB device front panels. It maintains system integrity by preventing a user from interfering with system operations by pressing buttons. As long as Local Lockout is in effect, no bus device can be returned to local control from its front panel.

The Local Lockout message is sent by executing the LOCAL LOCKOUT statement. This message can be sent to all devices on the external interface by specifying the bus address (7). Specifying a single address on the bus (i.e. 722) sends the command to only the device at that address. The Local Lockout message is cleared when the Local message is sent by executing the LOCAL statement. However, executing the ABORT statement does not cancel the Local Lockout message.

#### **Host Instrument**

The Local Lockout message is not supported for the host instrument since front panel control is always necessary in order to pause or abort the program. Specifying the internal interface in a LOCAL LOCKOUT statement will not generate an error, but will have no effect.

## **LOCAL**

During system operation, it may be necessary for an operator to interact with one or more external devices. For instance, an operator might need to work from the front panel to make special tests or to troubleshoot. It is also good systems practice to return all devices to local control when remote-control operations are complete. Executing the LOCAL statement returns the specified devices to local (front panel) control.

If primary addressing is specified, the Go-to-Local message is sent only to the specified device(s). However, if only the interface select code is specified (LOCAL 7), the Local message is sent to all devices on the external interface and any previous Local Lockout message (which is still in effect) is automatically cleared.

#### **Host Instrument**

The LOCAL statement has no effect on the host instrument since it is always in remote control whenever an IBASIC program is running. Specifying the internal interface in a LOCAL statement will not generate an error.

## **TRIGGER**

The TRIGGER statement sends a Trigger message to a selected device or group of devices. The purpose of the Trigger message is to initiate some device-dependent action; for example, it can be used to trigger a digital voltmeter to perform its measurement cycle. Because the response of a device to a Trigger message is strictly device-dependent, neither the Trigger message nor the interface indicates what action is initiated by the device.

Specifying only the interface select code outputs a Trigger message to all devices currently addressed to listen on the bus. Including a device address in the statement triggers only the device addressed by the statement.

### **Host Instrument**

The TRIGGER statement is not supported by the HP 8711.

## **CLEAR**

The CLEAR statement provides a means of "initializing" a device to its predefined device-dependent state. When the CLEAR statement is executed, the Clear message is sent either to all devices or to the specified device, depending on the information contained within the device selector. If only the interface select code is specified, all devices on the specified HP-IB interface are cleared. If primary-address information is specified, the Clear message is sent only to the specified device. Only the active controller can send the Clear message.

### **Host Instrument**

The CLEAR statement is fully compatible on the internal interface.

## **ABORT**

This statement may be used to terminate all activity on the external bus and return the HP-IB interfaces of all devices to reset (or power-on) condition. Whether this affects other modes of the device depends on the device itself. The IBASIC program must be either the active or the system controller to perform this function. If it is the system controller and has passed active control to another device, executing this statement causes active control to be returned. Only the interface select code may be specified; primary-addressing information (such as 724) must not be included.

### **Aborting the Internal Bus**

ABORT is not supported for select code 8. Executing ABORT 8 will not generate an error.

## HP-IB Service Requests

Most HP-IB devices, such as voltmeters, frequency counters, and network analyzers, are capable of generating a “service request” when they require the active controller to take action. Service requests are generally made after the device has completed a task (such as making a measurement) or when an error condition exists (such as a printer being out of paper). The operating and/or programming manuals for each device describe the device’s capability to request service and conditions under which the device will request service. To request service, the device sends a Service Request message (SRQ) to the active controller. The mechanism by which the active controller detects these requests is the SRQ interrupt. Interrupts allow an efficient use of system resources, because the system may be executing a program until interrupted by an event’s occurrence. If enabled, the external event initiates a program branch to a routine which “services” the event (executes remedial action).

### Setting Up and Enabling SRQ Interrupts

In order for an HP-IB device to be able to initiate a service routine in the active controller, two prerequisites must be met: the SRQ interrupt event must have a service routine defined, and the SRQ interrupt must be enabled to initiate the branch to the service routine.

The following program segment shows an example of setting up and enabling an SRQ interrupt.

```
100 Hpib=7
110 ON INTR Hpib GOSUB Service_routine
120 !
130 Mask=2
140 ENABLE INTR Hpib;Mask
```

Since IBASIC recognizes only SRQ interrupts, the value assigned to the mask is meaningless. However, a mask value may be present as a placeholder for compatibility with HP Series 200/300 BASIC programs.

When an SRQ interrupt is generated by any device on the bus, the program branches to the service routine when the current line is exited (either when the line’s execution is finished or when the line is exited by a call to a user-defined function). The service routine, in general, must perform the following operations:

1. Determine which device(s) are requesting service
2. Determine what action is requested
3. Clear the SRQ line
4. Perform the requested action
5. Re-enable interrupts
6. Return to the former task (if applicable)

---

#### Note



The ON INTR statement must always precede the ENABLE INTR statement when the two are used in the same program.

---

## Servicing SRQ Interrupts

The SRQ is a level-sensitive interrupt; in other words, if an SRQ is present momentarily but does not remain long enough to be sensed by the controller, an interrupt will not be generated. The level-sensitive nature of the SRQ line also has further implications, which are described in the following paragraphs.

### Example

Assume that only one device is currently on the bus. The following service routine serially polls the device requesting service and clears the interrupt request. In this case, the controller does not have to determine which device was requesting service because only one device is present. Since only service request interrupts are enabled in IBASIC, the type of interrupt does not need to be determined either. The service is performed, and the SRQ event is re-enabled to generate subsequent interrupts.

```
500 Serv_rtn: Ser_poll=SPOLL(@Device)
510 ENTER @Device;Value
520 PRINT Value
530 ENABLE INTR 7 ! Use previous mask.
540 RETURN
```

The IEEE standard states that when an interrupting device is serially polled, it is to stop interrupting until a new condition occurs (or the same condition occurs again). To “clear” the SRQ line, a serial poll must be performed on the device. By performing this serial poll, the controller acknowledges to the device that it has seen the request for service and is responding. The device then removes its request for service (by releasing SRQ).

If the SRQ line had not been released, the controller would have branched to the service routine immediately upon re-enabling interrupts on this interface. This is due to the level-sensitive nature of the SRQ interrupt.

Also note that once an interrupt is sensed and logged, the interface cannot generate another interrupt until the first interrupt is serviced. The controller disables all subsequent interrupts from an interface until a pending interrupt is serviced.

### Conducting a Serial Poll

A sequential poll of individual devices on the bus is known as a Serial Poll. A byte of device-specific status is returned in response to a Serial Poll. This byte is called the “Status Byte” message and, depending on the device, may indicate an overload, a request for service, or a printer being out of paper. The particular response of each device depends on the device.

The SPOLL function performs a Serial Poll of the specified device; the program must currently be the active controller in order to execute this function.

#### Examples

```
ASSIGN @Device TO 700
Status_byte=SPOLL(@Device)

Spoll_724=SPOLL(724)
```

The Serial Poll is meaningless for an interface since it must poll individual devices on the interface. Therefore, primary addressing must be used with the SPOLL function.

## Passing and Regaining Control

Active control of the bus can be passed between controllers using the PASS CONTROL command. The following statements first define the HP-IB interface's select code and the new active controller's primary address and then pass control to that controller.

```
100 Hp_ib=7
110 New_ac_addr=20
120 PASS CONTROL 100*Hp_ib+New_ac_addr
```

Once the new active controller has accepted active control, the controller passing control assumes the role of a non-active controller on the specified HP-IB interface. The concept of using pass control with IBASIC is discussed in the next section, "The IBASIC HP-IB Model."

---

## The IBASIC HP-IB Model

The fact that IBASIC resides in, and coexists with an instrument poses a large set of possible interactions, both internal to the instrument and externally with other controllers and instruments. This section defines the principal players and rules of order when IBASIC is running within the host instrument.

### External and Internal Busses

There is physically only one HP-IB port and one HP-IB address for the HP 8711. IBASIC has access to two HP-IB ports: the "real" external port (select code 7) and a "virtual" internal port (select code 8), through which it communicates with the HP 8711.

The HP 8711 has only one output buffer, one input buffer and one set of status registers. Commands and data from both ports are placed in the same input buffer and data read out of both ports comes from the same output buffer. The instrument will not provide any kind of arbitration between an external controller and an IBASIC program.

The HP 8711 always behaves as if there is only one controller. If an IBASIC program is running, it is assumed to be the controller and therefore will receive all SRQ's from the host instrument (via the internal port). An IBASIC program cannot generate an SRQ on the external port while running.

### Service Request Indicators

An external controller may perform a serial poll (SPOLL) at any time without affecting a running IBASIC program. There are two Service Request Indicators (SRI) - one for the external port and one for the internal port. The internal SRI can only be cleared by an IBASIC program performing an SPOLL on device 800. The external SRI can only be cleared by an SPOLL from an external controller and can only be set when there is not an active IBASIC program.

The two SRI's will be set to their OR'd value when a program starts, and again when it finishes. This assures that any pending SRQ's can be serviced by the instrument's new controller.

The pausing or termination of a program will cause the Program Running bit in the Device Status register to go low. This can be used to generate an external SRQ. (For an example, see the DUALCTRLR example in chapter 9, "Example Programs.")

## **IBASIC as the Active Controller**

The IBASIC program is always the active controller on the internal interface (select code 8). When a program starts running, the HP-IB controller status of the instrument is automatically passed to the program. For example, if the instrument is set as System Controller, a program running in the instrument automatically becomes system controller and active controller on the external bus and the instrument relinquishes active control. When the program stops, the instrument regains active control.

Also, if an instrument set as Talker/Listener is passed control from an external controller, any program running in the instrument becomes active controller on the external interface.

Thus, there are two cases where a program running in an instrument can be active controller on the external interface:

- When the host instrument is set as System Controller and the program has not passed control
- When the host instrument is set as Talker/Listener and the instrument has been passed control from an external controller.

## **Passing Active Control to the Instrument**

The only way that the HP 8711 can gain active control of the external interface while a program is running is if the program is currently the active controller on select code 7 and passes control to the instrument. Normally, the active controller on the 7 bus can pass control to any device on the interface by using the statement

```
PASS CONTROL 7xx
```

where "xx" represents the address of the device on the bus. Because an IBASIC program does not interface with the host instrument over select code 7, a different method is used to pass control in this case. To pass active control of the external interface from an IBASIC program to the host instrument, use the statement

```
PASS CONTROL 8xx
```

where "xx" represents any two digit number from 00 to 99. This allows the instrument to control external plotters, printers and disk drives. When the instrument is finished with its HP-IB control activity, it automatically passes control back to the program.

---

**Note**

Control over the internal bus is used to govern access to the external bus. When the instrument is given control over the internal bus, it is actually given access to the external HP-IB hardware.

---

### **IBASIC as a Non-Active Controller**

IBASIC programs are always the active controller on the internal interface. There are two cases where an IBASIC program does not have control of the external HP-IB interface:

- When the host instrument is set as Talker/Listener and active control has **NOT** been passed from an external device
- When the host instrument is set as System Controller and the program has passed control to either the host instrument or another device on the external interface

In both of these cases, the program cannot perform activities of any kind on the external interface.

---

**Note**

An IBASIC program cannot act as a device on the external bus. To communicate with an external controller, the IBASIC program must be active controller and the external controller must act as the device (see the “Interfacing with an External Controller” section that follows).

---

---

## **Interfacing with an External Controller**

So far, we have discussed the ability to interface IBASIC programs with a network of external devices using the HP-IB. The idea of including an external controller in that network, and interfacing an IBASIC program with a program running in that computer presents some new possibilities.

External controller programs can interface with IBASIC programs (referred to as “internal programs”) over HP-IB in two basic ways:

First, the two programs can pass data back and forth using simple **OUTPUT** and **ENTER** statements. This requires coordination of both the internal and external programs and also requires that the internal program be the active controller during the interaction. To get an internal program and an external program to work together successfully, you should have a good understanding of the HP-IB model, presented earlier in this chapter.

Second, the external program can make use of the extensive set of HP 8711 HP-IB commands that interface with IBASIC programs. These mnemonics fall under the subsystem **PROGRAM** and allow the external controller to remotely perform many of the IBASIC front panel activities. This includes the ability to run, stop, pause, continue and delete an internal program. You can also remotely query or set the values of numeric and string variables.

Also included in the HP 8711 HP-IB command set are commands that allow you to transfer programs and program data to and from the instrument. Programs can be transferred (uploaded and downloaded) between an external controller and the program buffer in the instrument, and data can be transferred between an external program and a non-running internal program by setting and querying internal program variables. These SCPI mnemonics are described in the *HP-IB Command Reference*.

## Transferring Data Between Programs

### Using OUTPUT and ENTER statements

All data sent from an external controller to the instrument's external port is received by the instrument and not by any program running in it. Therefore, a non-active controller IBASIC program can never enter or output data via the external interface. This means that in order to pass data between an external controller and an internal program using OUTPUT and ENTER statements, the internal program must be given active control and the external controller must become the non-active controller. HP IBASIC for Windows and HP Series 200/300 BASIC controllers have the ability to enter and output data via HP-IB while acting as a non-active controller.

---

#### Note



Moving data through the HP-IB and running a measurement in the host instrument at the same time can slow both operations significantly. It is recommended that you do not perform these operations simultaneously.

---

One method of passing data between the two controllers is to set the instrument as Talker/Listener and run a program on the external controller that starts the IBASIC program and passes control to it. The IBASIC program can then output data to, and enter data from, the external controller. The following two programs, also included in chapter 9, "Example Programs," demonstrate how to transfer data between an internal program and an external controller program. The first program, DATA\_EXT, is run from an external controller. It assumes that a disk containing the corresponding IBASIC program DATA\_INT is in the disk drive of the HP 8711. It remotely loads the IBASIC program, starts it and then transfers active control to it. The IBASIC program DATA\_INT, with active control of the interface, queries the external program for name of the drive to catalog, and then outputs the catalogued string to the external program and passes active control back. After receiving the catalog data, the external program goes into a loop (line 1080) executing a command that continues to generate an error until the host computer again becomes active controller when control is passed back.

## Data Transfer Example — External Program.

```
10 !-----
20 !
30 ! BASIC program: DATA_EXT - Data transfer (external)
40 !
50 ! This program demonstrates how to transfer data from
60 ! an IBASIC program running on the HP 8711 to an
70 ! HP BASIC program (or an IBASIC program running
80 ! externally). This program was designed to run on a
90 ! computer or PC. It loads a program into the HP 8711,
100 ! runs it, and then gives it control of the bus.
110 ! This program then acts as a device on the bus;
120 ! sending and receiving data.
130 !
140 ! Before running this program, a disc with the program
150 ! 'DATA_INT' should be in the HP 8711's internal drive.
160 !
170 !-----
180 !
190 ! Initialize variables for the interface select code
200 ! and the HP-IB address of the HP 8711.
210 !
220 Scode=7
230 Address=16
240 Na=Scode*100+Address
250 !
260 ! Abort any bus traffic, clear the input/output queues
270 ! of the analyzer, clear the analyzer's status
280 ! registers and the display.
290 !
300 ABORT Scode
310 CLEAR Na
320 OUTPUT Na;"*CLS"
330 CLEAR SCREEN
340 !
350 ! Dimension an array to hold the catalog listing.
360 !
370 DIM Directory$(1:100)[85]
380 !
390 ! Prompt the operator to insert the disk in the
400 ! HP 8711, load the program and wait until done.
410 !
420 INPUT "Put disc with program 'DATA_INT' into the HP 8711. Press <ENTER> ",A$
430 DISP "Loading program on HP 8711 ... "
440 OUTPUT Na;"PROG:EXEC 'GET ""DATA_INT""'"
450 OUTPUT Na;"*OPC?"
460 ENTER Na;Opc
470 !
480 ! Read the analyzer's event status register and
490 ! check for any errors when loading file.
```

```

500 !
510 OUTPUT Na;"*ESR?"
520 ENTER Na;Esr
530 IF Esr>0 THEN
540     BEEP
550     DISP "Error occurred while loading 'DATA_INT' ... Program stopped."
560     STOP
570 END IF
580 !
590 ! Determine the HP-IB address of the controller
600 ! and set the pass control back address.
610 !
620 INTEGER Stat,Addr
630 STATUS 7,3;Stat
640 Addr=BINAND(Stat,31)
650 OUTPUT Na;"*PCB ";Addr
660 !
670 ! Send the command to run the DATA_INT program.
680 !
690 DISP "Running the program ... "
700 OUTPUT Na;"PROG:STAT RUN"
710 !
720 ! Monitor the program's status. When it has
730 ! paused, set the variable for the controller's
740 ! HP-IB address.
750 !
760 OUTPUT Na;"PROG:STAT?"
770 ENTER Na;Prog$
780 IF Prog$<>"PAUS" THEN GOTO 760
790 OUTPUT Na;"PROG:NUMB 'Host',";Addr
800 OUTPUT Na;"PROG:STAT CONT"
810 !
820 ! Pass control of the bus to the HP 8711.
830 !
840 PASS CONTROL Na
850 !
860 ! Wait until addressed to talk by the HP 8711,
870 ! then send the name of the disk to catalog.
880 !
890 OUTPUT Scode;":INTERNAL"
900 !
910 ! Wait until addressed to listen by the HP 8711,
920 ! then read the directory from the analyzer.
930 !
940 DISP "Reading data ... "
950 ENTER Scode;Directory$(*)
960 !
970 ! Print the catalog to the computer's display.
980 !
990 FOR I=1 TO 100

```

```

1000 IF LEN(Directory$(I))>0 THEN PRINT Directory$(I)
1010 NEXT I
1020 !
1030 ! Try to return the HP 8711 to LOCAL control.
1040 ! If the analyzer is still the active controller
1050 ! an error will be generated and the program
1060 ! will loop until control of the bus is received.
1070 !
1080 ON ERROR GOTO 1090
1090 LOCAL Na
1100 DISP ""
1110 END

```

#### Data Transfer Example — Internal Program.

```

10 !-----
20 !
30 ! IBASIC program: DATA_INT - Data transfer (internal)
40 !
50 ! This program demonstrates how to transfer data to
60 ! and from an external controller. In this example a
70 ! catalog listing is transferred from the HP 8711 to
80 ! the external controller. For more information look
90 ! at the program listing for 'DATA_INT'.
100 !
110 ! This IBASIC program is intended to run on the
120 ! HP 8711's internal controller.
130 !
140 !-----
150 !
160 ! Dimension an array to hold the catalog listing.
170 !
180 DIM Directory$(1:100)[85]
190 !
200 ! Pause the program and wait for the controller to
210 ! set the 'Host' variable with its' HP-IB address.
220 ! The controller continues this program after the
230 ! variable has been passed.
240 !
250 Host=0
260 PAUSE
270 !
280 ! Address the external controller to talk, read
290 ! the device to catalog. If the HP 8711 is not
300 ! active controller on the bus an error will occur
310 ! and the program will loop until control is
320 ! received.
330 !
340 ON ERROR GOTO 140
350 ENTER Host;Stor_dev$
360 OFF ERROR

```

```

370 !
380 ! Catalog the requested storage device into
390 ! the string array.
400 !
410 DISP "Reading catalog ... "
420 CAT Stor_dev$ TO Directory$(*)
430 !
440 ! Address the external controller to listen,
450 ! send the catalog array to the controller.
460 !
470 DISP "Transferring data ... "
480 OUTPUT Host;Directory$(*)
490 !
500 ! Pass control back to the external controller.
510 !
520 PASS CONTROL Host
530 DISP "DONE"
540 END

```

### Setting and Querying Variables

Another means of transferring data between an internal and an external program involves the ability to set and query internal program variables from an external program. The "PROG:[:SElected]:NUMBER" and "PROG:[:SElected]:STRing" mnemonics (and their query counterparts) are part of the HP 8711 HP-IB commands. The internal program must not be running when these commands are executed.

The command

```
PROG:NUMB < string >, < value >
```

sets the value of a numeric variable in the program. The command

```
PROG:STR < string >, < value >
```

sets the value of a string variable in the program. In both the PROG:NUMB and PROG:STR commands and queries, < string > is the variable name and must be string data (in quotes). In the PROG:STR command, < value > is also string data (in quotes).

Numeric and string parameters can also be queried. The query

```
PROG:NUMBER? < string >
```

returns the value of the specified numeric variable.

Arrays of REAL or INTEGER type may be sent or queried but arrays of strings are not allowed. Array elements are separated by commas.

Examples

```
OUTPUT 716;"PROG:NUMBER 'Test',99"
```

```
OUTPUT @Ibasic;"PROG:STRING 'A$','String Data'"
```

```
OUTPUT 716;"PROG:NUMB? 'Iarray(*)'"
```

The following program segment sends both numeric and string variable queries and enters the resulting data:

```
10  ASSIGN @Prog TO 716
20  OUTPUT @Prog;"FORM ASCII,3"
30  OUTPUT @Prog;"PROG:NUMB? 'Test'"
40  ENTER @Prog; Testval
50  PRINT "The value of the variable Test = ";Testval
60  OUTPUT @Prog;"PROG:STR? 'A$'"
70  ENTER @Prog; Str$
80  PRINT "A$ = ";Str$
90  END
```

## Downloading and Uploading Programs

Programs can be transferred between an external controller and program memory using the HP-IB download command "PROG:SELected:DEFine" and its upload query "PROG:SELected:DEFine?". Programs that use these mnemonics are run in the external controller.

### Downloading

Program data transferred (downloaded) from the external controller to the instrument is always transferred as an "arbitrary block." The arbitrary block may be a definite length or indefinite length block. The indefinite length block is by far the easiest and is simply a block of data that begins with the characters "#0" preceding the first line and ends with a line-feed character accompanied by an EOI signal on the HP-IB interface.

When using the mnemonic PROG:DEF to download program lines, the #0 must not be followed by a line-feed. Each program line must have a line number at its beginning and a line-feed at its end. To end the arbitrary block of program lines, a single line-feed must be output with the OUTPUT END parameter, which sends the EOI (End or Identify) signal on the HP-IB control lines.

The following example, DOWNLOAD, is also included in chapter 9, "Example Programs," and on the *IBASIC Example Programs Disk*.

```
10  !-----
20  !
30  ! BASIC program:  DOWNLOAD - Download program to HP 8711
40  !
50  ! This program demonstrates how to download an IBASIC
60  ! program to the HP8711. This program is designed to
70  ! run on an external controller.
80  !
90  !-----
100 !
110 ! Initialize variables for the interface select code
120 ! and the HP-IB address of the HP 8711.
130 !
```

```

140 Scode=7
150 Address=16
160 Na=Scode*100+Address
170 !
180 ! Initialize variables, abort any bus traffic and
190 ! clear the input/output queues of the analyzer.
200 !
210 DIM Line$[255]
220 ABORT Scode
230 CLEAR Na
240 !
250 ! Get the program's filename and open the file.
260 !
270 Get_filename: INPUT "Program to be transferred?",Filename$
280 ON ERROR GOTO No_file
290 DISP "Checking file ... "
300 ASSIGN @Basic_prog TO Filename$;FORMAT ON
310 OFF ERROR
320 !
330 ! Clear the contents of the analyzer's program buffer.
340 !
350 OUTPUT Na;"PROG:DEL:ALL"
360 !
370 ! Change the EOL (end of line) character to line feed
380 ! and initialize the line counter.
390 !
400 Transfer: ASSIGN @Prog TO Na;EOL CHR$(10)
410 Line_count=0
420 !
430 ! Initiate the program transfer (an indefinite length
440 ! block data transfer).
450 !
460 OUTPUT @Prog;"PROG:DEF #0";
470 !
480 ! Read each program line from the file and send it to
490 ! the HP 8711. Loop until the end of file is reached.
500 !
510 ON ERROR GOSUB End_file
520 LOOP
530 ENTER @Basic_prog;Line$
540 OUTPUT @Prog;Line$
550 Line_count=Line_count+1
560 DISP "Lines transferred: ";Line_count
570 END LOOP
580 !
590 ! End the data transfer (output a line feed with EOI)
600 ! and close the file. Return the analyzer to LOCAL
610 ! control and stop this program.
620 !
630 End_block: OUTPUT @Prog;CHR$(10) END

```

```

640 ASSIGN @Basic_prog TO *
650 DISP "Transfer complete"
660 LOCAL Na
670 STOP
680 !
690 ! This subroutine is the error handler for opening
700 ! the file - if the file won't open it returns to
710 ! get a new file name.
720 !
730 No_file: BEEP
740 DISP "CAN'T OPEN: "";Filename$;"" -- ";
750 GOTO Get_filename
760 RETURN
770 !
780 ! This subroutine is the error handler for the
790 ! data transfer. When the end of file is reached
800 ! it generates an error. Execution is resumed
810 ! outside of the transfer loop.
820 !
830 End_file: IF ERRN=59 THEN GOTO End_block
840 DISP ERRM$;" occurred during data transfer"
850 STOP
860 RETURN
870 END

```

Notice that the OUTPUT statement on line 460 is terminated with a semicolon. This suppresses the line-feed that would otherwise occur.

As each line of the program is downloaded it is checked for syntax. If an error is found, the error message is displayed on the CRT and the line is commented and checked for syntax again. If it still causes an error (for example the line may be too long) the line is discarded.

Any lines that currently exist in the memory buffer will remain unless they are overwritten by downloaded program lines. This makes it easy to edit lines in an external controller and then download only the edited lines into an existing program. If you want to completely overwrite the current program in memory, you must delete the program first. This can be done remotely using the extended command PROG:DEL:ALL (see line 350).

### Uploading

The mnemonic PROG:DEF? is used to upload a program from the program buffer. The entire program is then returned as a definite length arbitrary block. A definite length block starts with the "#" character followed by a single digit defining the number of following digits to read as the block length.

The following example, UPLoad, demonstrates an uploading routine run on an external controller. It is included in chapter 9, "Example Programs," and on the *IBASIC Example Programs Disk*.

```

10 !-----
20 !
30 ! BASIC program: UPLOAD - Upload program from HP 8711
40 !
50 ! This program uploads the current IBASIC program
60 ! in the HP 8711's program buffer to an ASCII file
70 ! on the controller's current mass storage device.
80 !
90 !-----
100 !
110 ! Assign an I/O path name to the HP 8711, initialize
120 ! the variables, and clear the analyzer's input/output
130 ! queues.
140 !
150 ASSIGN @Hp8711 TO 716
160 DIM Prog_line$[256]
170 CLEAR @Hp8711
180 !
190 ! Enter the name of the file to be created.
200 !
210 INPUT "ENTER NAME OF FILE TO UPLOAD PROGRAM TO ",Filename$
220 PRINT Filename$
230 !
240 ! Query the HP 8711 for the contents of its
250 ! program buffer.
260 !
270 OUTPUT @Hp8711;"PROG:DEF?"
280 !
290 ! Read the block header, the number of digits in
300 ! the file size, and the file size.
310 !
320 ENTER @Hp8711 USING "#,A,D";Prog_line$,Ndigits
330 ENTER @Hp8711 USING "#,%VAL$(Ndigits)&"D";Nbytes
340 !
350 ! Create the target ASCII file on the current mass
360 ! storage device and assign it an I/O path name.
370 !
380 Openfile(@File,Filename$,Nbytes)
390 ASSIGN @File TO Filename$;FORMAT ON
400 !
410 ! Read the program one line at a time, and write
420 ! it to the new file. Print each line on the
430 ! display as it is read.
440 !
450 LOOP
460   ENTER @Hp8711;Prog_line$
470   EXIT IF LEN(Prog_line$)=0
480   PRINT Prog_line$
490   OUTPUT @File;Prog_line$
500 END LOOP

```

```

510 !
520 ! Close the new file.
530 !
540 ASSIGN @File TO *
550 END
560 !-----
570 SUB Openfile(@File,Filename$,Fsize)
580 !-----
590 !
600 ! This subprogram creates an ASCII file with the
610 ! name 'Filename$' of the specified size 'Fsize'.
620 ! Error trapping is used to detect any errors in
630 ! opening the file. If the controller is HP IBASIC
640 ! for Windows a DOS file is created, otherwise the
650 ! LIF format is used.
660 !
670 !-----
680 ON ERROR GOTO Openerr
690 IF SYSTEM$("SYSTEM ID")="IBASIC/WINDOWS" THEN
700     CREATE Filename$,1
710 ELSE
720     IF Fsize MOD 256>0 THEN Fsize=Fsize+256
730     CREATE ASCII Filename$,Fsize DIV 256
740 END IF
750 !
760 Openerr: IF ERRN<>54 THEN PRINT ERRM$
770 SUBEND

```

The subroutine `Openfile` (lines 570 through 770) creates an ASCII file to save the uploaded program to. The number of 256 byte records declared in the `CREATE ASCII` statement (line 730) is simply the file size (declared in the definite block header) divided by 256. Line 720 accommodates any remainder in this calculation by increasing the file size number by one record if any remainder exists.

Although this simple method works for many uploaded programs, there may still be a problem with the file size caused by the `OUTPUT` statement in line 490. This is because every ASCII line in a LIF file contains a two byte length header and possibly one additional pad byte to make the length an even number of bytes. These extra bytes are not included in the definite length block header information. You can account for this extra overhead by allocating an extra 10 to 15 percent of space when you create the ASCII file. For example, the `Openfile` subroutine could be rewritten as:

```

570 SUB Openfile(@File,Filename$,Fsize)
680 ON ERROR GOTO Openerr
715 Fsize=Fsize+(Fsize*0.15)
720 IF Fsize MOD 256>0 THEN Fsize=Fsize+256
730 CREATE ASCII Filename$,Fsize DIV 256

```

## Using Subprograms

---

HP 8711 products shipped with the IBASIC option, revision 2, can run subprograms. The subprograms may be user-created or built-in.

### User-Created Subprograms

You can use the LOADSUB keyword with subprograms of your own creation. LOADSUB enables you to append subprograms to other programs and is supported as described in the RMB manual. When using LOADSUB, keep in mind the following:

- Subprograms must be stored to files using the STORE keyword when first created.
- Subprograms may be stored from the external keyboard or from the front panel if the [File Type] format is BIN.
- BIN type files are generally not transportable between the HP 8711 and other development systems (only ASCII files are compatible with other systems).

Typical examples of LOAD/STORE:

From an external keyboard:

```
LOAD "MYFILE"
```

```
STORE "MYFILE"
```

From the front panel:

```
Save/Recall Programs File Type BIN Save/Program
```

```
Save/Recall Programs File Type BIN Recall/Program
```

Typical examples of LOADSUB:

```
LOADSUB subprogram_name FROM "filename"
```

```
LOADSUB ALL FROM "filename"
```

User-created subprograms are appended to the end of the BASIC program currently stored in the EDIT buffer.

---

## Built-In High-Speed Subprograms

In addition, you can use LOADSUB to access pre-compiled routines stored as instrument firmware in internal memory. Any IBASIC program running on the HP 8711 can access these subprograms; programs running on external computers can not. The external program must use the equivalent code listed in the table below in place of a built-in subprogram.

IBASIC programs which use the built-in subprograms are simpler and run faster. For example, most data transfer operations run twice as fast when using the built-in subprograms; math operations run many times faster. Built-in subprograms are stored in memory designated as "MEM,0,0".

Independent files are stored in internal memory. Subprograms within these files may be loaded as a group or loaded individually into active programs.

- "XFER" file adds support to transfer trace data between the instrument and the IBASIC program.
- "MATH" file adds high speed support for complex array operations.

**Built-in Subprogram Description (Filenames found in :MEM,0,0)**

Filename	Subprogram Name (parameter list)	Description
XFER	Read_fdata(INTEGER Chan,REAL A(*))	Read real formatted data
	Read_fmem(INTEGER Chan,REAL A(*))	Read real formatted mem
	Read_cdata(INTEGER Chan,REAL A(*))	Read complex data
	Read_cmem(INTEGER Chan,REAL A(*))	Read complex memory
	Write_fdata(INTEGER Chan,REAL A(*))	Write real formatted data
	Write_fmem(INTEGER Chan,REAL A(*))	Write real formatted mem.
	Write_cdata(INTEGER Chan,REAL A(*))	Write complex data
	Write_cmem(INTEGER Chan,REAL A(*))	Write complex memory
	Read_rdata(INTEGER Chan,Input\$, REAL A(*))	Read raw complex data
	Write_rdata(INTEGER Chan,Input\$,REAL A(*))	Write raw complex data
	Read_corr(INTEGER Chan, N,REAL A(*))	Read complex error coef.
	Write_corr(INTEGER Chan, N,REAL A(*))	Write complex error coef.
MATH		
Define Complex Array Operations	Cmplx_mag(REAL Cdata(*),Mag(*),INTEGER Sz)	Mag of complex array
	Cmplx_arg(REAL Cdata(*),Arg(*),INTEGER Sz) !Arg of complex array	
	Cmplx_conjg(REAL A(*),B(*)) !Complex conj of array A to B	
Define Complex Number Operations	Cadd(REAL Op1(*),INTEGER Row1,REAL Op2(*),INTEGER Row2,REAL Ans(*),INTEGER Rowans)	Complex Ans=Op1+Op2
	Csub(REAL Op1(*),INTEGER Row1,REAL Op2(*),INTEGER Row2,REAL Ans(*),INTEGER Rowans)	Complex Ans=Op1-Op2
	Cmul(REAL Op1(*),INTEGER Row1,REAL Op2(*),INTEGER Row2,REAL Ans(*),INTEEGER Rowans)	Complex Ans=Op1*Op2
	Cdiv(REAL Op1(*),INTEGER Row1,REAL Op2(*),INTEGER Row2,REAL Ans(*),INTEEGER Rowans)	Complex Ans=Op1/Op2

## Example Programs

```
1
10 LOADSUB Read_fdata FROM "XFER:MEM,0,0"
20
30
40
50 REAL Trace_array(1:201)
60 Read_Data( 1, Trace_array(*))
70 LOADSUB ALL FROM "MATH:MEM,0,0"
80 END
90 SUB Read_fdata(INTEGER Chan,REAL A(*))
100 SUB Cmplx_mag(REAL Cdata(*),Mag(*),INTEGER Sz)
110 SUB Cmplx_arg(REAL Cdata(*),Arg(*),INTEGER Sz)
120 ...
```

*Example use of built in sub programs*

*Appends Read\_data sub program to end of this program. This sub program can now be called.*

*Reads Channel 1 data into Trace\_array(\*)*

*Appends all math sub programs defined in "MATH" to the end of this program. Read real formatted data.*

*Mag of complex array.*

*Arg of complex array.*

### Note



Built in sub programs can not be edited since they are compiled and built into the firmware. However, any sub program can be deleted by the DELSUB keyword support in revision 2 IBASIC.

### RUNTIME Built in subprogram Errors

Number	Description
8,9,16	Improper or inconsistent dimensions found which specify array size. Using the wrong number of subscripts when referencing an array element.
983	Wrong type or number of parameters. An improper parameter list for a machine resident function.

## Avoiding Multiple Loads of Subprograms

To avoid multiple LOADS of a subprogram which has already been loaded, the following example may be used.

```
10 ON ERROR GOTO 30
20 DELSUB Read_fdata
30 LOADSUB Read_fdata FROM "XFER:MEM,0,0"
40 OFF ERROR
```

## HP 8711 IBASIC Keyword Summary

This chapter summarizes the HP Instrument BASIC keyword implementation in the HP 8711A with firmware revision 2.0. Table 10-1 is alphabetical. It indicates the type of support for each entry and notes exceptions, if any. A bullet in the second column indicates that the keyword is NEW to IBASIC revision 2. Exceptions are major differences between the keywords descriptions in the "HP Instrument BASIC Language Reference" and their implementation in the HP 8711. When differences are too extensive to be summarized, see the "HP Instrument BASIC Language Reference."

Table 10-2 is categorical, but otherwise a repeat of the first.

**Table 10-1. Alphabetical List of IBASIC Revision 2.0 Keywords Supported by the HP 8711A with Firmware Revision 2.0**

HP IBASIC Keyword	New Item	Support FP=Front Panel EK=External Keyboard P=Programmable	Exceptions
&		FP,EK,P	
*		EK,P	
+		EK,P	
-		EK,P	
/		K,P	
<.<=,<>,=>,>=		P	
ABORT		EK,P	Select Code = 7,8,9,15
ABS		EK,P	
ACS		FP,EK,P	
ALLOCATE	•	EK,P	
AND		FP,EK,P	
ASN		FP,EK,P	
ASSIGN		EK,P	
ATN		FP,EK,P	
AXES	•	EK,P	
BEEP		EK,P	
BINAND		FP,EK,P	
BINCMP		FP,EK,P	
BINEOR		FP,EK,P	
BINIOR		FP,EK,P	
BIT		FP,EK,P	

**Table 10-1. Alphabetical List of IBASIC Revision 2.0 Keywords Supported by the HP 8711A with Firmware Revision 2.0 (continued)**

HP IBASIC Keyword	New Item	Support FP=Front Panel EK=External Keyboard P=Programmable	Exceptions
CALL		EK,P	
CASE		P	
CASE ELSE		P	
CAT		FP,EK,P	Supports 58 columns. See manual.
CHRS		FP,EK,P	
CLEAR		EK,P	Select Code = 7,8,9,15
CLEAR SCREEN		EK,P	
CLS		EK,P	
COM		P	
CONT		EK,FP	Line number support from EK only
COPY		FP,EK,P	
COPYLINES	•	EK	
COS		FP,EK,P	Abs vals less than 1.7083127722e+10
CREATE	•		
CREATE ASCII		EK,P	
CREATE BDAT		EK,P	
CREATE DIR	•		
CRT		EK,P	ENTER CRT(ENTER 1) not supported
CSIZE	•	EK,P	
DATA		P	
DATE	•	EK,P	
DATES	•	EK,P	
DEALLOCATE	•	EK,P	
DEF FN		P	
DEG		FP,EK,P	
DEL		FP,EK	Front Panel deletes only 1 line
DELSUB	•	EK,P	
DET	•	EK,P	
DIM		P	
DISABLE		P	
DISABLE INTR		P	Interface Select Code = 7 or 8
DISP		EK,P	
DIV		EK,P	
DOT	•	EK,P	
DRAW		EK,P	

**Table 10-1. Alphabetical List of IBASIC Revision 2.0 Keywords Supported by  
the HP 8711A with Firmware Revision 2.0 (continued)**

HP IBASIC Keyword	New Item	Support FP=Front Panel EK=External Keyboard P=Programmable	Exceptions
DROUND		EK,P	
DUMP ALPHA		none	Use HPIB command
DVAL		FP,EK,P	
DVAL\$		FP,EK,P	
EDIT		FP,EK	Front Panel EDITs default line #
ELSE		P	
ENABLE		P	
ENABLE INTR		P	Interface Select Code = 7 or 8
END		P	
END IF		P	
END LOOP		P	
END SELECT		P	
END WHILE		P	
ENTER		EK,P	
ERRL()		P	
ERRLN()		EK,P	
ERRM\$		EK,P	
ERRN		EK,P	
EXIT IF		P	
EXOR		FP,EK,P	
EXP		EK,P	
FN		P	
FNEND		P	
FOR NEXT		P	
FRACT		EK,P	
FRAME	•	EK,P	
GCLEAR		EK,P	
GET		FP,EK,P	
GINIT	•	EK,P	
GOSUB		P	
GOTO		P	
GRID	•	EK,P	
IDRAW	•	EK,P	
IF THEN		P	

**Table 10-1. Alphabetical List of IBASIC Revision 2.0 Keywords Supported by the HP 8711A with Firmware Revision 2.0 (continued)**

HP IBASIC Keyword	New Item	Support FP=Front Panel EK=External Keyboard P=Programmable	Exceptions
IMAGE		P	
IMOVE	•	EK,P	
INDENT	•	EK	
INITIALIZE		FP,EK,P	
INPUT		P	See Manual.
INT		EK,P	
INTEGER		P	
IPLOT	•	EK,P	
IVAL		FP,EK,P	
IVAL\$		FP,EK,P	
KBD		P	Returns select code =2.
LABEL	•	EK,P	
LDIR	•	EK,P	
LEN		FP,EK,P	
LET		EK,P	
LGT		EK,P	
LIST		EK,P	Valid Device Selectors
LOAD	•	FP,EK,P	
LOADSUB	•	FP,EK,P	
LOADSUB ALL FROM ...	•	FP,EK,P	
LOCAL		EK,P	Select Code 7 only.
LOCAL LOCKOUT		EK,P	Select Code 7 only.
LOG		EK,P	
LOOP		P	
LORG	•	EK,P	
LWCS		FP,EK,P	
MAT		EK,P	
MAT REORDER	•	EK,P	
MAT REORDER ... BY	•	EK,P	
MAT foo=CSUM(bar)	•	EK,P	
MAT foo=IDN	•	EK,P	
MAT foo=INV(bar)	•	EK,P	
MAT foo=RSUM(bar)	•	EK,P	

**Table 10-1. Alphabetical List of IBASIC Revision 2.0 Keywords Supported by the HP 8711A with Firmware Revision 2.0 (continued)**

HP IBASIC Keyword	New Item	Support FP=Front Panel EK=External Keyboard P=Programmable	Exceptions
MAX		EK,P	
MAXLEN		FP,EK,P	
MAXREAL		EK,P	
MIN		EK,P	
MINREAL		EK,P	
MOD		EK,P	
MODULO		EK,P	
MOVE		EK,P	
MOVELINES	•	EK	
MSI		FP,EK,P	MSI may be altered by the instr.
NOT		FP,EK,P	
NUM		FP,EK,P	
ON OFF CYCLE	•	P	
ON OFF ERROR		P	
ON OFF INTR		P	Interface Select Code = 7 or 8
ON OFF KEY		P	Key selectors 1 thru 7
ON OFF TIMEOUT		P	Interface Select Code = 7 or 8
OPTION BASE	•	P	
OR		FP,EK,P	
OUTPUT		EK,P	Select Code 1,7,8
PASS CONTROL		EK,P	Select Code 7 or 8
PAUSE		EK,FP,P	
PDIR	•	EK,P	
PEN		EK,P	0=erase 1=draw
PENUP	•	EK,P	
PI		EK,P	
PIVOT	•	EK,P	
PLOT	•	EK,P	
POLYGON	•	EK,P	FILL not supported. Scaling diffs.
POLYLINE	•	EK,P	
POS		FP,EK,P	
PRINT		EK,P	
PRINTER IS		EK,P	
PROUND		EK,P	

**Table 10-1. Alphabetical List of IBASIC Revision 2.0 Keywords Supported by the HP 8711A with Firmware Revision 2.0 (continued)**

HP IBASIC Keyword	New Item	Support FP=Front Panel EK=External Keyboard P=Programmable	Exceptions
PRT		EK,P	
PURGE		FP,EK,P	
RAD		FP,EK,P	
RANDOMIZE		EK,P	
RATIO	•	EK,P	
RE-SAVE		FP,EK,P	
RE-STORE		FP,EK,P	
READ		EK,P	
READIO		EK,P	Select Code 9 or 15. See manual.
REAL		P	
RECTANGLE	•	EK,P	FILL not supported. Scaling diffs.
REDIM	•	EK,P	
REM		P	
REMOTE		EK,P	Select Code 7
REN		EK	
RENAME		FP,EK,P	
REPEAT UNTIL		P	
RESTORE	•	P	
RETURN		P	
REVS		FP,EK,P	
RND		EK,P	
ROTATE		FP,EK,P	
RPLOT	•	EK,P	Fill not supported. Scaling diffs.
RPT\$		FP,EK,P	
RUN		EK,FP,P	
SAVE		FP,EK,P	
SCRATCH		FP,EK	Front Panel executes SCRATCH A.
SECURE		FP,EK	
SELECT		P	
SET TIME foo	•	FP,EK,P	
SET TIMEDATE foo	•	EK,P	
SGN		EK,P	
SHIFT		FP,EK,P	
SHOW	•	EK,P	

**Table 10-1. Alphabetical List of IBASIC Revision 2.0 Keywords Supported by the HP 8711A with Firmware Revision 2.0 (continued)**

HP IBASIC Keyword	New Item	Support FP=Front Panel EK=External Keyboard P=Programmable	Exceptions
SIN		FP,EK,P	
SPOLL		EK,P	Select Code 7
SQR		EK,P	
SQRT		EK,P	
STEP		FP,EK	
STOP		FP,P	
STORE	•	FP,EK,P	
SUB		P	
SUBEND		P	
SUBEXIT		P	
SUM	•	EK,P	
SYSTEM PRIORITY		P	
SYSTEM\$		EK,P	
TAB()		EK,P	
TABXY()		EK,P	
TAN		FP,EK,P	
TIME	•	EK,P	
TIMES	•	EK,P	
TRIGGER		EK,P	Select Code 7
TRIM\$		FP,EK,P	
TRN	•	EK,P	
UPC\$		FP,EK,P	
USING		EK,P	
VAL		FP,EK,P	
VAL\$		FP,EK,P	
VIEWPORT	•	EK,P	
WAIT		EK,P	
WHERE	•	EK,P	
WHILE		P	
WIDTH		EK,P	
WINDOW	•	EK,P	
WRITEIO		EK,P	Select Code 9 or 15. See manual.
^		EK,P	

**Table 10-2. Categorical List of IBASIC Revision 2.0 Keywords Supported by  
the HP 8711A with Firmware Revision 2.0**

HP Instrument BASIC Keyword	Support FP=Front Panel EK=External Keyboard P=Programmable	Exceptions
<b>Program Entry/Editing</b> COPYLINES DEL DELSUB EDIT INDENT LIST MOVELINES REM REN SECURE	EK FP,EK EK FP,EK EK EK,P EK P EK FP,EK	Front Panel deletes only 1 line. Front Panel EDITs default line #. See Manual. Valid Device Selectors #7xx, #7xxxx, #9, #15.
<b>Program Debugging</b> ERRL() ERRLN() ERRMS ERRN STEP	P EK,P EK,P EK,P FP,EK	
<b>Memory Allocation</b> ALLOCATE COM DEALLOCATE DELSUB DIM INTEGER LOADSUB OPTION BASE REAL SCRATCH	EK,P P EK,P EK,P P P EK,P P P FP,EK	Front Panel executes SCRATCH A.
<b>Relational Operators</b> <.<=,<>,=>,>=	P	
<b>General Math</b> * + - /	EK,P EK,P EK,P EK,P	

**Table 10-2. Categorical List of IBASIC Revision 2.0 Keywords Supported by the HP 8711A with Firmware Revision 2.0 (continued)**

HP Instrument BASIC Keyword	Support FP=Front Panel EK=External Keyboard P=Programmable	Exceptions
ABS	EK,P	
DIV	EK,P	
DROUND	EK,P	
EXP	EK,P	
FRACT	EK,P	
INT	EK,P	
LET	EK,P	
LGT	EK,P	
LOG	EK,P	
MAX	EK,P	
MAXREAL	EK,P	
MIN	EK,P	
MINREAL	EK,P	
MOD	EK,P	
MODULO	EK,P	
PI	EK,P	
PROUND	EK,P	
RANDOMIZE	EK,P	
RND	EK,P	
SGN	EK,P	
SQR	EK,P	
SQRT	EK,P	
^	EK,P	
<b>Binary Functions</b>		
BINAND	FP,EK,P	
BINCMP	FP,EK,P	
BINEOR	FP,EK,P	
BINIOR	FP,EK,P	
BIT	FP,EK,P	
ROTATE	FP,EK,P	
SHIFT	FP,EK,P	
<b>Trigonometric Operations</b>		
ACS	FP,EK,P	
ASN	FP,EK,P	
ATN	FP,EK,P	
COS	FP,EK,P	Abs vals less than 1.7083127722e+10
DEG	FP,EK,P	

**Table 10-2. Categorical List of IBASIC Revision 2.0 Keywords Supported by  
the HP 8711A with Firmware Revision 2.0 (continued)**

HP Instrument BASIC Keyword	Support FP=Front Panel EK=External Keyboard P=Programmable	Exceptions
RAD	FP,EK,P	
SIN	FP,EK,P	
TAN	FP,EK,P	
<b>String Operations</b>		
&	FP,EK,P	
CHRS	FP,EK,P	
DVAL\$	FP,EK,P	
DVAL	FP,EK,P	
IVAL\$	FP,EK,P	
IVAL	FP,EK,P	
LEN	FP,EK,P	
LWCS	FP,EK,P	
MAXLEN	FP,EK,P	
NUM	FP,EK,P	
POS	FP,EK,P	
REVS	FP,EK,P	
RPTS	FP,EK,P	
TRIMS	FP,EK,P	
UPCS	FP,EK,P	
VAL\$	FP,EK,P	
VAL	FP,EK,P	
<b>Logical Operations</b>		
AND	FP,EK,P	
EXOR	FP,EK,P	
NOT	FP,EK,P	
OR	FP,EK,P	
<b>Mass Storage</b>		
CAT	FP,EK,P	Supports 58 columns. See manual.
COPY	FP,EK,P	
CREATE		
CREATE ASCII	EK,P	
CREATE BDAT	EK,P	
CREATE DIR		
GET	FP,EK,P	

**Table 10-2. Categorical List of IBASIC Revision 2.0 Keywords Supported by  
the HP 8711A with Firmware Revision 2.0 (continued)**

HP Instrument BASIC Keyword	Support FP=Front Panel EK=External Keyboard P=Programmable	Exceptions
INITIALIZE LOAD LOADSUB LOADSUB ALL FROM ... MSI	FP,EK,P FP,EK,P FP,EK,P FP,EK,P FP,EK,P	MSI may be altered by the instr. When save/recalling programs to/from DOS subdirectories.
PURGE RE-SAVE RENAME RE-STORE SAVE STORE	FP,EK,P FP,EK,P FP,EK,P FP,EK,P FP,EK,P FP,EK,P	
<b>Program Control</b>		
CALL	EK,P	
CASE	P	
CASE ELSE	P	
CONT	EK,FP	Line number support from EK only
DEF FN	P	
ELSE	P	
END	P	
END IF	P	
END LOOP	P	
END SELECT	P	
END WHILE	P	
EXIT IF	P	
FN	P	
FNEND	P	
FOR NEXT	P	
GOSUB	P	
GOTO	P	
IF THEN	P	
LOOP	P	
PAUSE	EK,FP,P	
REPEAT UNTIL	P	

**Table 10-2. Categorical List of IBASIC Revision 2.0 Keywords Supported by  
the HP 8711A with Firmware Revision 2.0 (continued)**

HP Instrument BASIC Keyword	Support FP=Front Panel EK=External Keyboard P=Programmable	Exceptions
RETURN	P	
RUN	EK,FP,P	
SELECT	P	
STOP	FP,P	
SUB	P	
SUBEND	P	
SUBEXIT	P	
SYSTEMS	EK,P	
WAIT	EK,P	
WHILE	P	
<b>Event Initiated Branching</b>		
DISABLE	P	
DISABLE INTR	P	Interface Select Code = 7 or 8.
ENABLE	P	
ENABLE INTR	P	Interface Select Code = 7 or 8. Must not precede an ON INTR statement.
ON OFF CYCLE	P	
ON OFF ERROR	P	
ON OFF INTR	P	Interface Select Code = 7 or 8. Must precede ENABLE INTR statement.
ON OFF KEY	P	Key selectors 1 thru 7
ON OFF TIMEOUT	P	Interface Select Code = 7 or 8
SYSTEM PRIORITY	P	
<b>Graphics Control</b>		
GCLEAR	EK,P	
GINIT	EK,P	
RATIO	EK,P	
SHOW	EK,P	
VIEWPORT	EK,P	
WHERE	EK,P	
WINDOW	EK,P	
<b>Graphics Plotting</b>		
DRAW	EK,P	
IDRAW	EK,P	
IMOVE	EK,P	

**Table 10-2. Categorical List of IBASIC Revision 2.0 Keywords Supported by the HP 8711A with Firmware Revision 2.0 (continued)**

HP Instrument BASIC Keyword	Support FP=Front Panel EK=External Keyboard P=Programmable	Exceptions
IPLOT	EK,P	
MOVE	EK,P	
PDIR	EK,P	
PEN	EK,P	0=erase 1=draw
PENUP	EK,P	
PIVOT	EK,P	
PLOT	EK,P	
POLYGON	EK,P	FILL not supported. Scaling diffs.
POLYLINE	EK,P	
RECTANGLE	EK,P	FILL not supported. Scaling diffs.
RPLOT	EK,P	FILL not supported. Scaling diffs.
<b>Graphics Axis and Labeling</b>		
AXES	EK,P	
CSIZE	EK,P	
FRAME	EK,P	
GRID	EK,P	
LABEL	EK,P	
LDIR	EK,P	
LORG	EK,P	
<b>HP-IB Control</b>		
ABORT	EK,P	Select Code = 7,8,9,15
CLEAR	EK,P	Select Code = 7,8,9,15
LOCAL	EK,P	Select Code 7 only.
LOCAL LOCKOUT	EK,P	Select Code 7 only.
PASS CONTROL	EK,P	Select Code 7 or 8
REMOTE	EK,P	Select Code 7
SPOLL	EK,P	Select Code 7
TRIGGER	EK,P	Select Code 7
<b>Clock and Calendar</b>		
DATE	EK,P	
DATES	EK,P	
SET TIME foo	FP,EK,P	
SET TIMEDATE foo	EK,P	
TIME	EK,P	
TIMES	EK,P	

**Table 10-2. Categorical List of IBASIC Revision 2.0 Keywords Supported by the HP 8711A with Firmware Revision 2.0 (continued)**

HP Instrument BASIC Keyword	Support FP=Front Panel EK=External Keyboard P=Programmable	Exceptions
<b>General Device Input/Output</b>		
ASSIGN	EK,P	
BEEP	EK,P	
CRT	EK,P	ENTER CRT(ENTER 1) not supported
DATA	P	
DISP	EK,P	
ENTER	EK,P	
IMAGE	P	
INPUT	P	See Manual.
KBD	P	Returns select code =2.
OUTPUT	EK,P	Select Code 1,7,8
PRINT	EK,P	
PRINTER IS	EK,P	
PRT	EK,P	
READ	EK,P	
READIO	EK,P	Select Code 9 or 15. See manual.
RESTORE	P	
TAB()	EK,P	
TABXY()	EK,P	
USING	EK,P	
WIDTH	EK,P	
WRITEIO	EK,P	Select Code 9 or 15. See manual.
<b>Display and Keyboard Control</b>		
CLEAR SCREEN	EK,P	
CLS	EK,P	
<b>Array Operations</b>		
DET	EK,P	
DOT	EK,P	
MAT	EK,P	
MAT foo=IDN	EK,P	
MAT foo=INV(bar)	EK,P	
MAT foo=CSUM(bar)	EK,P	
MAT foo=RSUM(bar)	EK,P	
MAT REORDER	EK,P	
MAT REORDER ... BY	EK,P	
REDIM	EK,P	
SUM	EK,P	
TRN	EK,P	

## Example Programs

---

### Program Summaries

This chapter contains listings of the example programs referred to throughout this manual. These programs are all available on the *IBASIC Example Programs Disk* that accompanies this manual.

In addition to these example programs, there are two additional disks of examples for the HP 8711. These disks are the *HP 8711 Example Programs Disk — DOS Format* and the *HP 8711 Example Programs Disk — LIF Format*. These disks are included with the network analyzer when it is delivered. All the programs on these disks are designed to run on the analyzer's internal IBASIC controller.

The example programs listed in this chapter include the following:

#### **DATA\_EXT — Data transfer between internal and external programs**

This program is designed to run on an external controller — either HP IBASIC for Windows running on a PC or HP BASIC running on an HP workstation.

This program demonstrates how to transfer data from an IBASIC program running on the HP 8711 to an HP BASIC or IBASIC program running externally. It loads a program into the HP 8711, runs it, sets a variable and then gives it control of the bus. This program then acts as a device on the bus (sending and receiving data).

#### **DATA\_INT — Data transfer between internal and external programs**

This program is designed to run on the HP 8711's internal IBASIC controller.

This program demonstrates how to transfer data to and from an external controller. In this example a catalog listing is transferred from the HP 8711 to the external controller. A numeric variable value is also downloaded from the external controller to the HP 8711's program.

#### **DOWNLOAD — Download program to HP 8711**

This program demonstrates how to download an IBASIC program to the HP 8711. It is designed (in HP BASIC or HP IBASIC for Windows) to run on an external workstation or PC.

### **DRAW8711 — Drawing setup diagrams**

This program draws the HP 8711 network analyzer and a device under test to the full screen IBASIC display partition. The drawing can be scaled to fit the application.

### **DUALCTRL — Two controller operation**

This program demonstrates how the external controller and HP IBASIC can work together. It is designed to run on an external controller (in HP BASIC or HP IBASIC for Windows). The program downloads an IBASIC program to the HP 8711 and runs it twice. After each run, two program variables are read from the analyzer and displayed.

### **REPORT — Using the parallel port**

This program uses the HP 8711 to generate a report, making a hardcopy on a printer connected to the parallel port. It uses a subprogram to send the output to the parallel port one line at a time.

### **UPLOAD — Upload program from HP 8711**

This program uploads the IBASIC program in the HP 8711's program buffer to an ASCII file on the external controller's current mass storage device.

### **USER\_BIT — Using the USER bit**

This program demonstrates how to read and write to the USER bit. The USER bit is a TTL signal accessible by a BNC connector on the HP 8711's rear panel. IBASIC's graphics commands are used to draw the USER bit value to the display.

### **USERKEYS — Customized softkeys**

This program provides an example of how the HP 8711's softkeys can be customized. The example demonstrates how to set up six instrument states, store them to the analyzer's internal memory, and setup two interactive softkey menus to choose between them.

### **BAR, AVG, LOG — Barcode Programs**

You may use barcode readers to simplify your measurement setups. The HPCK-1210 KeyWand scanner or compatible barcode scanner will work with the HP 8711. Connect your barcode scanner to the DIN keyboard connector. You may connect a keyboard or other DIN key input device in parallel with the barcode scanner. The barcode scanner will work in place of, or in addition to, your keyboard.

The INPUT statement is used to read the barcode from the scanner. When the input statement is encountered, the program will wait until the user has completed an input. The input is completed whenever a carriage return is received from the keyboard or a barcode has been successfully scanned by the barcode scanner.

The following three programs, designed to run on the HP 8711's internal IBASIC controller, demonstrate the use of barcode scanner applications as well as other useful applications. While a barcode scanner is useful in demonstrating these programs, it is not required; one can simply press ENTER and the program will input default values. Sample barcodes are provided for experimentation. The three programs are as follows:

BAR - This program demonstrates basic barcode scanning to select one of three filter setups depending upon what is scanned. RF stimulus is set and response limits are read, set and

tested for each device. Depending upon result, the program prints "PASS" or "FAIL" on the CRT. Most useful in this program is a subprogram to draw an HP 8711 representation on the CRT. This code can be re-used in any user application that may require a guided setup.

The HP 8711 image (and DUT image) can be both scaled to any size, and offset in the X or Y axis as required. This is an excellent program to familiarize yourself with graphic routines using IBASIC graphics commands.

AVG - This program first reads a DUT barcode and sets the RF stimulus accordingly. It then displays a running average of all similar devices and constantly updates the display with both the current DUT and the current average of all devices tested so far. Also demonstrates the use of two of the built-in CSUB routines for reading and writing trace data from/to the HP 8711.

LOG - This program will very quickly store measured trace data for one of three filters to internal HP 8711 memory in a format that can be read by spreadsheet programs for further analysis. Because the data is stored to RAM, the time delay inherent with disks is not an issue; trace data can be stored in a fraction of a second. With 101 data points per trace selected, the internal memory will hold over 20 device test results. At this point, the program automatically transfers the data to disk. Of course, more data points will take longer to store and fill the memory sooner. The program will read the barcode and select the stimulus accordingly. It then measures the device and upon request, stores it under a unique name dependent upon model number and serial number. Once the internal memory is full, or at any user requested time, all trace data is transferred to disk.

## DATA\_EXT — Data transfer between internal and external programs

```
10  !-----  
20  !  
30  ! BASIC program: DATA_EXT - Data transfer (external)  
40  !  
50  ! This program demonstrates how to transfer data from  
60  ! an IBASIC program running on the HP 8711 to an  
70  ! HP BASIC program (or an IBASIC program running  
80  ! externally). This program was designed to run on a  
90  ! computer or PC. It loads a program into the HP 8711,  
100 ! runs it, and then gives it control of the bus.  
110 ! This program then acts as a device on the bus;  
120 ! sending and receiving data.  
130 !  
140 ! Before running this program, a disc with the program  
150 ! 'DATA_INT' should be in the HP 8711's internal drive.  
160 !  
170 !-----  
180 !  
190 ! Initialize variables for the interface select code  
200 ! and the HP-IB address of the HP 8711.  
210 !  
220 Scode=7  
230 Address=16  
240 Na=Scode*100+Address
```

```

250 !
260 ! Abort any bus traffic, clear the input/output queues
270 ! of the analyzer, clear the analyzer's status
280 ! registers and the display.
290 !
300 ABORT Scode
310 CLEAR Na
320 OUTPUT Na;"*CLS"
330 CLEAR SCREEN
340 !
350 ! Dimension an array to hold the catalog listing.
360 !
370 DIM Directory$(1:100)[85]
380 !
390 ! Prompt the operator to insert the disk in the
400 ! HP 8711, load the program and wait until done.
410 !
420 INPUT "Put disc with program 'DATA_INT' into the HP 8711. Press <ENTER>",&A$
430 DISP "Loading program on HP 8711 ... "
440 OUTPUT Na;"PROG:EXEC 'GET ""DATA_INT:INTERNAL""'"
450 OUTPUT Na;"*OPC?"
460 ENTER Na;Opc
470 !

480 ! Read the analyzer's event status register and
490 ! check for any errors when loading file.
500 !
510 OUTPUT Na;"*ESR?"
520 ENTER Na;Esr
530 IF Esr>0 THEN
540     BEEP
550     DISP "Error occurred while loading 'DATA_INT' ... Program stopped."
560     STOP
570 END IF
580 !
590 ! Determine the HP-IB address of the controller
600 ! and set the pass control back address.
610 !
620 INTEGER Stat,Addr
630 STATUS 7,3;Stat
640 Addr=BINAND(Stat,31)
650 OUTPUT Na;"*PCB ";Addr
660 !
670 ! Send the command to run the DATA_INT program.
680 !
690 DISP "Running the program ... "
700 OUTPUT Na;"PROG:STAT RUN"
710 !
720 ! Monitor the program's status. When it has
730 ! paused, set the variable for the controller's
740 ! HP-IB address.

```

```

750 !
760 OUTPUT Na;"PROG:STAT?"
770 ENTER Na;Prog$
780 IF Prog$<>"PAUS" THEN GOTO 760
790 OUTPUT Na;"PROG:NUMB 'Host',";Scode*100+Addr
800 OUTPUT Na;"PROG:STAT CONT"
810 !
820 ! Pass control of the bus to the HP 8711.
830 !
840 PASS CONTROL Na
850 !
860 ! Wait until addressed to talk by the HP 8711,
870 ! then send the name of the disk to catalog.
880 !
890 OUTPUT Scode;":INTERNAL"
900 !
910 ! Wait until addressed to listen by the HP 8711,
920 ! then read the directory from the analyzer.
930 !
940 DISP "Reading data ... "
950 ENTER Scode;Directory$(*)
960 !
970 ! Print the catalog to the computer's display.

980 !
990 FOR I=1 TO 100
1000 IF LEN(Directory$(I))>0 THEN PRINT Directory$(I)
1010 NEXT I
1020 !
1030 ! Try to return the HP 8711 to LOCAL control.
1040 ! If the analyzer is still the active controller
1050 ! an error will be generated and the program
1060 ! will loop until control of the bus is received.
1070 !
1080 ON ERROR GOTO 1090
1090 LOCAL Na
1100 DISP ""
1110 END

```

## DATA\_INT — Data transfer between internal and external programs

```
10 !-----
20 !
30 ! IBASIC program: DATA_INT - Data transfer (internal)
40 !
50 ! This program demonstrates how to transfer data to
60 ! and from an external controller. In this example a
70 ! catalog listing is transferred from the HP 8711 to
80 ! the external controller. For more information look
90 ! at the program listing for 'DATA_INT'.
100 !
110 ! This IBASIC program is intended to run on the
120 ! HP 8711's internal controller.
130 !
140 !-----
150 !
160 ! Dimension an array to hold the catalog listing.
170 !
180 DIM Directory$(1:100)[85]
190 !
200 ! Pause the program and wait for the controller to
210 ! set the 'Host' variable with its' HP-IB address.
220 ! The controller continues this program after the
230 ! variable has been passed.
240 !
250 Host=0
260 PAUSE
270 !
280 ! Address the external controller to talk, read
290 ! the device to catalog. If the HP 8711 is not
300 ! active controller on the bus an error will occur
310 ! and the program will loop until control is
320 ! received.
330 !
340 ON ERROR GOTO 340
350 ENTER Host;Stor_dev$
360 OFF ERROR
370 !
380 ! Catalog the requested storage device into
390 ! the string array.
400 !
410 DISP "Reading catalog ... "
420 CAT Stor_dev$ TO Directory$(*)
430 !
440 ! Address the external controller to listen,
450 ! send the catalog array to the controller.
460 !
470 DISP "Transferring data ... "
```

```
480 OUTPUT Host;Directory$(*)
490 !
500 ! Pass control back to the external controller.
510 !
520 PASS CONTROL Host
530 DISP "DONE"
540 END
```

## DOWNLOAD — Download program to HP 8711

```
10 !-----
20 !
30 ! BASIC program: DOWNLOAD - Download program to HP 8711
40 !
50 ! This program demonstrates how to download an IBASIC
60 ! program to the HP8711. This program is designed to
70 ! run on an external controller.
80 !
90 !-----
100 !
110 ! Initialize variables for the interface select code
120 ! and the HP-IB address of the HP 8711.
130 !
140 Scode=7
150 Address=16
160 Na=Scode*100+Address
170 !
180 ! Initialize variables, abort any bus traffic and
190 ! clear the input/output queues of the analyzer.
200 !
210 DIM Line$[255]
220 ABORT Scode
230 CLEAR Na
240 !
250 ! Get the program's filename and open the file.
260 !
270 Get_filename: INPUT "Program to be transferred?",Filename$
280 ON ERROR GOTO No_file
290 DISP "Checking file ... "
300 ASSIGN @Basic_prog TO Filename$;FORMAT ON
310 OFF ERROR
320 !
330 ! Clear the contents of the analyzer's program buffer.
340 !
350 OUTPUT Na;"PROG:DEL:ALL"
360 !
370 ! Change the EOL (end of line) character to line feed
380 ! and initialize the line counter.
390 !
400 Transfer: ASSIGN @Prog TO Na;EOL CHR$(10)
410 Line_count=0
420 !
430 ! Initiate the program transfer (an indefinite length
440 ! block data transfer).
450 !
460 OUTPUT @Prog;"PROG:DEF #0";
470 !
```

```

480 ! Read each program line from the file and send it to
490 ! the HP 8711. Loop until the end of file is reached.
500 !
510 ON ERROR GOSUB End_file
520 LOOP
530 ENTER @Basic_prog;Line$
540 OUTPUT @Prog;Line$
550 Line_count=Line_count+1
560 DISP "Lines transferred: ";Line_count
570 END LOOP
580 !
590 ! End the data transfer (output a line feed with EOI)
600 ! and close the file. Return the analyzer to LOCAL
610 ! control and stop this program.
620 !
630 End_block: OUTPUT @Prog;CHR$(10) END
640 ASSIGN @Basic_prog TO *
650 DISP "Transfer complete"
660 LOCAL Na
670 STOP
680 !
690 ! This subroutine is the error handler for opening
700 ! the file - if the file won't open it returns to
710 ! get a new file name.
720 !
730 No_file: BEEP
740 DISP "CAN'T OPEN: """;Filename$;"" -- ";
750 GOTO Get_filename
760 RETURN
770 !
780 ! This subroutine is the error handler for the
790 ! data transfer. When the end of file is reached
800 ! it generates an error. Execution is resumed
810 ! outside of the transfer loop.
820 !
830 End_file: IF ERRN=59 THEN GOTO End_block
840 DISP ERRM$;" occurred during data transfer"
850 STOP
860 RETURN
870 END

```

## DRAW8711 — Drawing setup diagrams

```
10  !-----
20  !
30  ! IBASIC program: DRAW8711 - Drawing setup diagrams
40  !
50  ! This program draws the HP 8711 network analyzer
60  ! and a device under test to the full screen IBASIC
70  ! display partition. The drawing can be scaled to
80  ! fit the application. Setting the scale factor to
90  ! 1.0 creates a drawing of about 400 pixels wide
100 ! (1/2 screen width) and 100 pixels high (1/3 screen
110 ! height).
120 !
130 !-----
140 !
150 ! Setup an I/O path name for the internal bus and
160 ! declare variables.
170 !
180 INTEGER XO,YO
190 REAL Scale
200 ASSIGN @Hp8711 TO 800
210 !
220 ! Preset the analyzer and wait until it is done.
230 !
240 OUTPUT @Hp8711;"SYST:PRES;*OPC?"
250 ENTER @Hp8711;Opc
260 !
270 ! Allocate the full screen as an IBASIC display
280 ! and clear the graphics buffer.
290 !
300 OUTPUT @Hp8711;"DISP:PROG FULL"
310 OUTPUT @Hp8711;"DISP:WIND10:GRAP:CLEAR"
320 !
330 ! Setup the origin and scale parameters for the
340 ! drawing. Draw the network analyzer and dut.
350 !
360 XO=100
370 YO=100
380 Scale=1.
390 CALL Draw_na(XO,YO,Scale)
400 CALL Draw_dut(XO,YO,Scale)
410 END
420 !-----
430 SUB Draw_na(INTEGER XO,INTEGER YO,REAL Sc)
440  !-----
450  !
460  ! This subroutine draws the HP 8711 at origin XO,YO
470  ! and scale Sc. The drawing is done to the IBASIC
```

```

480 ! display (window 10) using the HP 8711's user
490 ! graphics commands.
500 !
510 !-----
520 ASSIGN @Hp8711 TO 800
530 OUTPUT @Hp8711;"DISP:WIND10:GRAP:MOVE "&VAL$(XO)&","
&VAL$(YO)
540 OUTPUT @Hp8711;"DISP:WIND10:GRAP:RECT "&VAL$(INT(Sc*350))&","
&VAL$(INT(Sc*100))
550 OUTPUT @Hp8711;"DISP:WIND10:GRAP:MOVE "&VAL$(XO+INT(Sc*10))&","
&VAL$(YO+INT(Sc*10))
560 OUTPUT @Hp8711;"DISP:WIND10:GRAP:RECT "&VAL$(INT(Sc*180))&","
&VAL$(INT(Sc*80))
570 OUTPUT @Hp8711;"DISP:WIND10:GRAP:MOVE "&VAL$(XO+INT(Sc*200))&","
&VAL$(YO+INT(Sc*80))
580 OUTPUT @Hp8711;"DISP:WIND10:GRAP:RECT "&VAL$(INT(Sc*15))&","
&VAL$(INT(Sc*8))
590 OUTPUT @Hp8711;"DISP:WIND10:GRAP:MOVE "&VAL$(XO+INT(Sc*200))&","
&VAL$(YO+INT(Sc*70))
600 OUTPUT @Hp8711;"DISP:WIND10:GRAP:RECT "&VAL$(INT(Sc*15))&","
&VAL$(INT(Sc*8))
610 OUTPUT @Hp8711;"DISP:WIND10:GRAP:MOVE "&VAL$(XO+INT(Sc*200))&","
&VAL$(YO+INT(Sc*60))
620 OUTPUT @Hp8711;"DISP:WIND10:GRAP:RECT "&VAL$(INT(Sc*15))&","
&VAL$(INT(Sc*8))
630 OUTPUT @Hp8711;"DISP:WIND10:GRAP:MOVE "&VAL$(XO+INT(Sc*200))&","
&VAL$(YO+INT(Sc*50))
640 OUTPUT @Hp8711;"DISP:WIND10:GRAP:RECT "&VAL$(INT(Sc*15))&","
&VAL$(INT(Sc*8))
650 OUTPUT @Hp8711;"DISP:WIND10:GRAP:MOVE "&VAL$(XO+INT(Sc*200))&","
&VAL$(YO+INT(Sc*40))
660 OUTPUT @Hp8711;"DISP:WIND10:GRAP:RECT "&VAL$(INT(Sc*15))&","
&VAL$(INT(Sc*8))
670 OUTPUT @Hp8711;"DISP:WIND10:GRAP:MOVE "&VAL$(XO+INT(Sc*200))&","
&VAL$(YO+INT(Sc*30))
680 OUTPUT @Hp8711;"DISP:WIND10:GRAP:RECT "&VAL$(INT(Sc*15))&","
&VAL$(INT(Sc*8))
690 OUTPUT @Hp8711;"DISP:WIND10:GRAP:MOVE "&VAL$(XO+INT(Sc*200))&","
&VAL$(YO+INT(Sc*20))
700 OUTPUT @Hp8711;"DISP:WIND10:GRAP:RECT "&VAL$(INT(Sc*15))&","
&VAL$(INT(Sc*8))
710 OUTPUT @Hp8711;"DISP:WIND10:GRAP:MOVE "&VAL$(XO+INT(Sc*200))&","
&VAL$(YO+INT(Sc*10))
720 OUTPUT @Hp8711;"DISP:WIND10:GRAP:RECT "&VAL$(INT(Sc*15))&","
&VAL$(INT(Sc*8))
730 OUTPUT @Hp8711;"DISP:WIND10:GRAP:MOVE "&VAL$(XO+INT(Sc*265))&","
&VAL$(YO+INT(Sc*80))
740 OUTPUT @Hp8711;"DISP:WIND10:GRAP:RECT "&VAL$(INT(Sc*70))&","
&VAL$(INT(Sc*13))
750 OUTPUT @Hp8711;"DISP:WIND10:GRAP:MOVE "&VAL$(XO+INT(Sc*230))&","

```

```

&VAL$(YO+INT(Sc*81))
760  OUTPUT @Hp8711;"DISP:WIND10:GRAP:RECT "&VAL$(INT(Sc*20))&","
&VAL$(INT(Sc*10))
770  OUTPUT @Hp8711;"DISP:WIND10:GRAP:MOVE "&VAL$(XO+INT(Sc*275))&","
&VAL$(YO+INT(Sc*85))
780  OUTPUT @Hp8711;"DISP:WIND10:GRAP:RECT "&VAL$(INT(Sc*50))&","
&VAL$(INT(Sc*3))
790  OUTPUT @Hp8711;"DISP:WIND10:GRAP:MOVE "&VAL$(XO+INT(Sc*295))&","
&VAL$(YO+INT(Sc*50))
800  OUTPUT @Hp8711;"DISP:WIND10:GRAP:CIRC "&VAL$(INT(Sc*8))
810  OUTPUT @Hp8711;"DISP:WIND10:GRAP:MOVE "&VAL$(XO+INT(Sc*245))&","
&VAL$(YO+INT(Sc*15))
820  OUTPUT @Hp8711;"DISP:WIND10:GRAP:CIRC "&VAL$(INT(Sc*4))
830  OUTPUT @Hp8711;"DISP:WIND10:GRAP:MOVE "&VAL$(XO+INT(Sc*325))&","
&VAL$(YO+INT(Sc*15))
840  OUTPUT @Hp8711;"DISP:WIND10:GRAP:CIRC "&VAL$(INT(Sc*4))
850  SUBEND
860  !-----
870  SUB Draw_dut(INTEGER XO,INTEGER YO,REAL Sc)
880  !-----
890  !
900  ! This subprogram draws a device under test (dut)
910  ! and connects it to the HP 8711 that was drawn
920  ! with an origin at XO,YO and a scale of Sc.
930  !
940  !-----
950  ASSIGN @Hp8711 TO 800
960  OUTPUT @Hp8711;"DISP:WIND10:GRAP:MOVE "&VAL$(XO+INT(Sc*245))&","
&VAL$(YO+INT(Sc*15))
970  OUTPUT @Hp8711;"DISP:WIND10:GRAP:DRAW "&VAL$(XO+INT(Sc*245))&","
&VAL$(YO-INT(Sc*20))
980  OUTPUT @Hp8711;"DISP:WIND10:GRAP:DRAW "&VAL$(XO+INT(Sc*265))&","
&VAL$(YO-INT(Sc*20))
990  OUTPUT @Hp8711;"DISP:WIND10:GRAP:MOVE "&VAL$(XO+INT(Sc*265))&","
&VAL$(YO-INT(Sc*22))
1000  OUTPUT @Hp8711;"DISP:WIND10:GRAP:RECT "&VAL$(INT(Sc*40))&","
&VAL$(INT(Sc*4))
1010  OUTPUT @Hp8711;"DISP:WIND10:GRAP:MOVE "&VAL$(XO+INT(Sc*305))&","
&VAL$(YO-INT(Sc*20))
1020  OUTPUT @Hp8711;"DISP:WIND10:GRAP:DRAW "&VAL$(XO+INT(Sc*325))&","
&VAL$(YO-INT(Sc*20))
1030  OUTPUT @Hp8711;"DISP:WIND10:GRAP:DRAW "&VAL$(XO+INT(Sc*325))&","
&VAL$(YO+INT(Sc*15))
1040  SUBEND

```

## DUALCTRL — Two controller operation

```
10  !-----  
20  !  
30  ! BASIC program: DUALCTRL - Two controller operation  
40  !  
50  ! This program is designed to run on an external  
60  ! controller. It demonstrates how the external  
70  ! controller and HP IBASIC can work together. The  
80  ! program downloads an IBASIC program to the HP 8711  
90  ! and runs it twice. After each run, two program  
100 ! variables are read from the analyzer and displayed.  
110 !  
120 !-----  
130 !  
140 ! Initialize the variables for the interface select  
150 ! code and the HP-IB address of the HP 8711.  
160 !  
170 Scode=7  
180 Address=16  
190 Na=Scode*100+Address  
200 !  
210 ! Prepare the analyzer for remote operation, clear  
220 ! the analyzer's input/output queues, the display  
230 ! and scratch any program in the buffer.  
240 !  
250 CLEAR Na  
260 CLEAR SCREEN  
270 OUTPUT Na;"PROG:DEL:ALL"  
280 !  
290 ! Download the program as an indefinite block length  
300 ! data transfer, terminate the data transfer by  
310 ! sending a carriage return and EOI.  
320 !  
330 DISP "Downloading the program ... "  
340 ASSIGN @Prog TO Na  
350 OUTPUT @Prog;"PROG:DEF #0";  
360 OUTPUT @Prog;"10 COM INTEGER Times_run,Test$[10]"  
370 OUTPUT @Prog;"20 Times_run=Times_run+1"  
380 OUTPUT @Prog;"30 IF Times_run=1 THEN Test$=""PASS""  
390 OUTPUT @Prog;"40 IF Times_run=2 THEN Test$=""FAIL""  
400 OUTPUT @Prog;"50 FOR I= 1 TO 20"  
410 OUTPUT @Prog;"60 BEEP"  
420 OUTPUT @Prog;"70 NEXT I"  
430 OUTPUT @Prog;"80 END"  
440 OUTPUT @Prog;CHR$(10) END  
450 !  
460 ! Initialize interrupt registers - clear the status byte,  
470 ! the service request enable register, the standard event
```

```

480 ! enable register, and preset the other status registers.
490 !
500 OUTPUT Na;"*CLS"
510 OUTPUT Na;"*SRE 0"
520 OUTPUT Na;"*ESE 0"
530 OUTPUT Na;"STAT:PRES"
540 !
550 ! Set up the status registers to generate an interrupt
560 ! on negative transition of the Program Running bit
570 ! (bit 14 in the Operational Status register).
580 !
590 OUTPUT Na;"STAT:OPER:NTR #HFFFF"
600 OUTPUT Na;"STAT:OPER:ENAB 16384"
610 OUTPUT Na;"*CLS"
620 OUTPUT Na;"*SRE 128"
630 !
640 ! Run the program, read and display the variables.
650 !
660 DISP "Running the program ... "
670 OUTPUT Na;"PROG:EXEC 'RUN'"
680 Display_res(Na,Scode)
690 OUTPUT Na;"PROG:EXEC 'RUN'"
700 Display_res(Na,Scode)
710 !
720 ! Return the analyzer to front panel control, this
730 ! is the end of the program.
740 !
750 LOCAL Na
760 DISP "DONE !"
770 END
780 !
790 !-----
800 !
810 SUB Display_res(Na,Scode)
820 !-----
830 !
840 ! This subprogram waits for an SRQ interrupt to
850 ! signal that an IBASIC program running on the
860 ! analyzer has finished. It then reads and clears
870 ! the HP-IB status registers. The values of two
880 ! program variables are then read and displayed.
890 !
900 !-----
910 !
920 ! Setup branching to an interrupt handling routine,
930 ! enable the interrupts and wait until one occurs.
940 !
950 ON INTR Scode GOTO Read_results
960 ENABLE INTR Scode;2
970 Idle: GOTO Idle

```

```
980 Read_results: !
990  !
1000 ! The program has finished running - read and clear
1010 ! the operational status register and status byte.
1020 !
1030 A=SPOLL(Na)
1040 OUTPUT Na;"STAT:OPER:EVENT?"
1050 ENTER Na;Event
1060 OUTPUT Na;"*CLS"
1070 !
1080 ! Read a numeric variable (Times_run) and a string
1090 ! variable (Test$) and display the values.
1100 !
1110 OUTPUT Na;"PROG:NUMB? 'Times_run'"
1120 ENTER Na USING "X,K";Times_run
1130 OUTPUT Na;"PROG:STR? 'Test$'"
1140 ENTER Na USING "X,K";Test$
1150 DISP "Times_run: ";Times_run,"Test$: ";Test$
1160 PRINT "Times_run: ";Times_run,"Test$: ";Test$
1170 SUBEND
```

## REPORT — Using the parallel port

```
10  !-----
20  !
30  ! IBASIC program: REPORT - Using the parallel port
40  !
50  ! This program uses the 8711 to generate a report,
60  ! making a hardcopy on a printer connected to the
70  ! parallel port. It uses a subprogram to send the
80  ! output to the parallel port one line at a time.
90  !
100 ! This example uses five different font types that
110 ! may or may not be supported for your printer.
120 ! These character fonts are available for HP LaserJet
130 ! printers. Refer to your printer manual to modify
140 ! the example fonts for your printer.
150 !
160 !-----
170 !
180 ! Assign an I/O path name for the internal bus and
190 ! declare and initialize variables.
200 !
210 COM /Cset/ Block$[50],Title$[50],Slant$[50],Banner$[50],Medium$[50]
220 ASSIGN @Hp8711 TO 800
230 Esc$=CHR$(27)
240 !
250 ! Preset the analyzer, put it in Trigger HOLD mode,
260 ! allocate the full IBASIC display and clear the
270 ! screen.
280 !
290 OUTPUT @Hp8711;"SYST:PRES;*WAI"
300 OUTPUT @Hp8711;"ABOR;:INIT:CONT OFF;*WAI"
310 OUTPUT @Hp8711;"DISP:PROG FULL"
320 CLEAR SCREEN
330 !
340 ! Define the escape sequence for each font that is
350 ! used. Refer to your printer manual.
360 !
370 Block$=Esc$&"&100"&Esc$&"(8U"&Esc$&"(s1p10h12v0 s0b0T"
380 Title$=Esc$&"&100"&Esc$&"(8U"&Esc$&"(s1p8h12v0s0 b0T"
390 Slant$=Esc$&"&100"&Esc$&"(7J"&Esc$&"(s0p6h14v1s0 b0T"
400 Banner$=Esc$&"&100"&Esc$&"(7J"&Esc$&"(s0p4h24v0 s0b0T"
410 Medium$=Esc$&"&100"&Esc$&"(7J"&Esc$&"(s0p8h14v 0s0b0T"
420 !
430 ! Select the font to use writing the company name
440 ! and address, send the company name and address.
450 !
460 CALL Send_line(Title$,1)
470 CALL Send_line("COMPANY NAME",1)
```

```

480 CALL Send_line("CITY, STATE, COUNTRY",1)
490 CALL Send_line(" ",1)
500 !
510 ! Select the font to use writing the device name,
520 ! send the device name.
530 !
540 CALL Send_line(Banner$,1)
550 CALL Send_line(" ",0)
560 CALL Send_line(" ",1)
570 CALL Send_line(" ",1)
580 CALL Send_line(" ",1)
590 CALL Send_line(" BPF-175 Bandpass Filter",1)
600 CALL Send_line(" ",1)
610 CALL Send_line(" ",1)
620 CALL Send_line(" ",0)
630 CALL Send_line(" ",1)
640 CALL Send_line(" ",1)
650 !
660 ! Select the font to use writing the device
670 ! specifications, send the information.
680 !
690 CALL Send_line(Slant$,1)
700 CALL Send_line(" ",1)
710 CALL Send_line("PASS BAND (MHZ)      3 dB          60 +/- 5",1)
720 CALL Send_line(" ",1)
730 CALL Send_line("                20 dB          90 +/- 5",1)
740 CALL Send_line(" ",1)
750 CALL Send_line("                40 dB          120 +/- 5",1)
760 CALL Send_line(" ",1)
770 CALL Send_line("SWR PASSBAND (typical)  1.8:1",1)
780 CALL Send_line(" ",1)
790 CALL Send_line("SWR STOPBAND (typical)  1.8:1",1)
800 CALL Send_line(" ",1)
810 CALL Send_line("Cost per unit:          36.95",1)
820 !
830 ! Select the font to use for the performance data
840 ! title, send the title.
850 !
860 CALL Send_line(Block$,1)
870 CALL Send_line("                ",0)
880 CALL Send_line("                Transmission Characteristics",1)
890 !
900 ! Return the display to the analyzer.
910 !
920 OUTPUT @Hp8711;"DISP:PROG OFF"
930 !
940 ! Setup the device measurement. This example
950 ! measures the transmission response of a
960 ! bandpass filter at 175 MHz.
970 !

```

```

980 OUTPUT @Hp8711;"DISP:ANN:FREQ1:MODE SSTOP"
990 OUTPUT @Hp8711;"SENS1:FREQ:STAR 10 MHz;STOP 400 MHz;*WAI"
1000 OUTPUT @Hp8711;"DISP:WIND1:TRAC:Y:PDIV 20 dB;RLEV -50 dB;RPOS 5"
1010 OUTPUT @Hp8711;"DISP:ANN:TITL ON;TITL1:DATA 'HP 8711 RF NETWORK ANALYZER'"
1020 !
1030 ! Take a measurement sweep and wait for it to
1040 ! complete. Perform a -3 dB bandwidth search.
1050 !
1060 OUTPUT @Hp8711;"INIT1;*OPC?"
1070 ENTER @Hp8711;Opc
1080 OUTPUT @Hp8711;"CALC1:MARK1 ON;MARK:BWID -3"
1090 !
1100 ! Select the parallel port and the printer's
1110 ! control language as the hardcopy device.
1120 ! Set the printer resolution and margins -
1130 ! turn off automatic form feed.
1140 !
1150 OUTPUT @Hp8711;"HCOP:DEV:LANG PCL;PORT CENT"
1160 OUTPUT @Hp8711;"HCOP:DEV:RES 300"
1170 OUTPUT @Hp8711;"HCOP:PAGE:MARG:LEFT 40"
1180 OUTPUT @Hp8711;"HCOP:PAGE:WIDT 110"
1190 OUTPUT @Hp8711;"HCOP:ITEM1:FFE:STAT OFF"
1200 !
1210 ! Send the measurement data (graph and marker
1220 ! values) to the printer.
1230 !
1240 OUTPUT @Hp8711;"HCOP"
1250 !
1260 ! Select the fonts and send the "footer"
1270 ! information for the report.
1280 !
1290 CALL Send_line(Banner$,1)
1300 CALL Send_line(" ",1)
1310 CALL Send_line(" ",1)
1320 CALL Send_line("IN STOCK IMMEDIATE DELIVERY!",1)
1330 CALL Send_line(Medium$,1)
1340 CALL Send_line(" ",0)
1350 CALL Send_line("For more information: Call 1-800-Filter",1)
1360 !
1370 ! Send a form feed to the printer.
1380 !
1390 WRITEIO 15,0;12
1400 END
1410 !-----
1420 SUB Send_line(String$,INTEGER Crlf)
1430 !-----
1440 !
1450 ! The subprogram sends a string to the parallel port
1460 ! (I/O port 15). The Crlf flag determines whether
1470 ! a carriage return (ASCII 13) and line feed (ASCII

```

```
1480 ! 10) are needed at the end of the string.
1490 !
1500 !-----
1510 INTEGER Length
1520 Length=LEN(String$)
1530 FOR I=1 TO Length
1540   WRITEIO 15,0;NUM(String$[I;1])
1550 NEXT I
1560 IF Crlf=1 THEN
1570   WRITEIO 15,0;10
1580   WRITEIO 15,0;13
1590 END IF
1600 SUBEND
```

## UPLOAD — Upload program from HP 8711

```
10  !-----
20  !
30  ! BASIC program: UPLOAD - Upload program from HP 8711
40  !
50  ! This program uploads the current IBASIC program
60  ! in the HP 8711's program buffer to an ASCII file
70  ! on the controller's current mass storage device.
80  !
90  !-----
100 !
110 ! Assign an I/O path name to the HP 8711, initialize
120 ! the variables, and clear the analyzer's input/output
130 ! queues.
140 !
150 ASSIGN @Hp8711 TO 716
160 DIM Prog_line$[256]
170 CLEAR @Hp8711
180 !
190 ! Enter the name of the file to be created.
200 !
210 INPUT "ENTER NAME OF FILE TO UPLOAD PROGRAM TO ",Filename$
220 PRINT Filename$
230 !
240 ! Query the HP 8711 for the contents of its
250 ! program buffer.
260 !
270 OUTPUT @Hp8711;"PROG:DEF?"
280 !
290 ! Read the block header, the number of digits in
300 ! the file size, and the file size.
310 !
320 ENTER @Hp8711 USING "#,A,D";Prog_line$,Ndigits
330 ENTER @Hp8711 USING "#,"&VAL$(Ndigits)&"D";Nbytes
340 !
350 ! Create the target ASCII file on the current mass
360 ! storage device and assign it an I/O path name.
370 !
380 Openfile(@File,Filename$,Nbytes)
390 ASSIGN @File TO Filename$;FORMAT ON
400 !
410 ! Read the program one line at a time, and write
420 ! it to the new file. Print each line on the
430 ! display as it is read.
440 !
450 LOOP
460   ENTER @Hp8711;Prog_line$
470   EXIT IF LEN(Prog_line$)=0
```

```

480 PRINT Prog_line$
490 OUTPUT @File;Prog_line$
500 END LOOP
510 !
520 ! Close the new file.
530 !
540 ASSIGN @File TO *
550 END
560 !-----
570 SUB Openfile(@File,Filename$,Fsize)
580 !-----
590 !
600 ! This subprogram creates an ASCII file with the
610 ! name 'Filename$' of the specified size 'Fsize'.
620 ! Error trapping is used to detect any errors in
630 ! opening the file. If the controller is HP IBASIC
640 ! for Windows a DOS file is created, otherwise the
650 ! LIF format is used.
660 !
670 !-----
680 ON ERROR GOTO Openerr
690 IF SYSTEM$("SYSTEM ID")="IBASIC/WINDOWS" THEN
700 CREATE Filename$,1
710 ELSE
720 IF Fsize MOD 256>0 THEN Fsize=Fsize+256
730 CREATE ASCII Filename$,Fsize DIV 256
740 END IF
750 !
760 Openerr: IF ERRN<>54 THEN PRINT ERRM$
770 SUBEND

```

## USER\_BIT — Using the USER bit

```
10  !-----
20  !
30  ! IBASIC program: USER_BIT - Using the USER bit
40  !
50  ! This program reads and writes to the USER bit.
60  ! IBASIC's graphics commands are used to draw the
70  ! USER bit value to the display.
80  !
90  !-----
100 !
110 ! Assign an I/O path name to the internal bus and
120 ! initialize variables.
130 !
140 ASSIGN @Hp8711 TO 800
150 INTEGER Beeper,Count
160 Count=0
170 Beeper=0
180 !
190 ! Preset the analyzer, setup measurement and display
200 ! parameters for a measurement and put the analyzer
210 ! in Trigger HOLD mode.
220 !
230 OUTPUT @Hp8711;"SYST:PRES;*WAI"
240 OUTPUT @Hp8711;"DISP:ANN:FREQ1:MODE SSTOP"
250 OUTPUT @Hp8711;"SENS1:FREQ:STAR 100 MHz;STOP 400 MHz;*WAI"
260 OUTPUT @Hp8711;"DISP:WIND1:TRAC:Y:PDIV 20 dB;RLEV -60 dB;RPOS 5"
270 OUTPUT @Hp8711;"SENS1:SWE:POIN 101;TIME .1 s;*WAI"
280 OUTPUT @Hp8711;"ABOR;:INIT1:CONT OFF;*WAI"
290 !
300 ! Wait for all the setup operations to be complete
310 ! before continuing the program.
320 !
330 OUTPUT @Hp8711;"*OPC?"
340 ENTER @Hp8711;Opc
350 !
360 ! Allocate the lower display partition.
370 !
380 OUTPUT @Hp8711;"DISP:PROG LOW"
390 !
400 ! Setup a softkey menu to enable and disable the
410 ! beeper. Clear the analyzer's input/output queues.
420 !
430 ON KEY 1 LABEL "Beep   Enable" GOSUB Beep_on
440 ON KEY 2 LABEL "Beep   Disable" GOSUB Beep_off
450 CLEAR @Hp8711
460 !
470 ! Trigger 100 sweeps. Beep (if the beeper flag is set)
```

```

480 ! and toggle the USER bit after each sweep.
490 !
500 DISP "USER bit example program. End of sweep toggles USER bit."
510 PRINT "Draw the end of sweep USER bit value ... "
520 MOVE 0,50
530 FOR I=1 TO 100
540   OUTPUT @Hp8711;"INIT1;*OPC?"
550   ENTER @Hp8711;Opc
560   GOSUB Toggle
570 NEXT I
580 DISP "End program"
590 STOP
600 !
610 ! The Odd flag's value alternates between 1 and 0
620 ! depending on the number of sweeps that have been
630 ! taken. It is the value that is written to the
640 ! USER bit.
650 !
660 Toggle: !
670 IF Odd=0 THEN
680   WRITEIO 15,1;0
690   Odd=1
700 ELSE
710   WRITEIO 15,1;1
720   Odd=0
730 END IF
740 IF Beeper=1 THEN
750   BEEP
760 END IF
770 !
780 ! Read the value of the USER bit and draw it to the
790 ! IBASIC display.
800 !
810 Val=READIO(15,1)
820 Val=Val*30
830 DRAW 8*(I-1),Val+50
840 DRAW 8*I,Val+50
850 RETURN
860 !
870 ! These two subroutines set a flag that is used
880 ! to turn on or off the beeper.
890 !
900 Beep_on: Beeper=1
910 RETURN
920 !
930 Beep_off: Beeper=0
940 RETURN
950 END

```

## USERKEYS — Customized softkeys

```
10  !-----
20  !
30  ! IBASIC program: USERKEYS - Customized softkeys
40  !
50  ! This program provides an example template for use
60  ! in customizing the HP 8711's softkeys. The example
70  ! demonstrates how to set up six instrument states,
80  ! store them to the analyzer's internal memory, and
90  ! setup two interactive softkey menus to choose
100 ! between them.
110 !
120 !-----
130 !
140 ! Assign an I/O path name to the internal bus, preset
150 ! the analyzer, wait until the preset is complete,
160 ! turn on Trigger HOLD mode and set the display scale
170 ! and reference values.
180 !
190 ASSIGN @Hp8711 TO 800
200 OUTPUT @Hp8711;"SYST:PRES;*OPC?"
210 ENTER @Hp8711;Opc
220 OUTPUT @Hp8711;"ABOR;:INIT1:CONT OFF;*WAI"
230 OUTPUT @Hp8711;"DISP:WIND1:TRAC:Y:PDIV 20 dB;RLEV -60 dB;RPOS 5"
240 !
250 ! Setup six instrument states and store them to the
260 ! internal memory.
270 !
280 GOSUB Save_1
290 GOSUB Save_2
300 GOSUB Save_3
310 GOSUB Save_4
320 GOSUB Save_5
330 GOSUB Save_6
340 !
350 ! Setup the Main Menu keys.
360 !
370 GOSUB Menu_1
380 !
390 ! Wait until a softkey is pressed.
400 !
410 Suspend: !
420 WAIT 100000
430 GOTO Suspend
440 STOP
450 !
460 ! This subroutine sets up the softkey menus -
470 ! Menu1 sets up the main menu, Menu2 sets up
```

```

480 ! the second level menu.
490 !
500 Menu_1: BEEP
510 DISP "MAIN MENU"
520 ON KEY 1 LABEL "Setup #1" GOSUB Load_1
530 ON KEY 2 LABEL "Setup #2" GOSUB Load_2
540 ON KEY 3 LABEL "Setup #3" GOSUB Load_3
550 ON KEY 5 LABEL "Autoscale" GOSUB Autoscale
560 ON KEY 6 LABEL " Next Menu" GOSUB Menu_2
570 RETURN
580 !
590 Menu_2: BEEP
600 DISP "MORE MENU"
610 ON KEY 1 LABEL "Setup #4" GOSUB Load_4
620 ON KEY 2 LABEL "Setup #5" GOSUB Load_5
630 ON KEY 3 LABEL "Setup #6" GOSUB Load_6
640 ON KEY 5 LABEL "Autoscale" GOSUB Autoscale
650 ON KEY 6 LABEL "Prior Menu" GOSUB Menu_1
660 RETURN
670 !
680 ! This subroutine automatically sets the scale and
690 ! reference values of the display.
700 !
710 Autoscale: OUTPUT @Hp8711;"DISP:WIND1:TRAC:Y:AUTO ONCE"
720 OUTPUT @Hp8711;"DISP:WIND2:TRAC:Y:AUTO ONCE"
730 RETURN
740 !
750 ! These six subroutines each set up the analyzer to
760 ! make a different measurement and store that setup
770 ! to the instrument's internal memory.
780 !
790 Save_1: OUTPUT @Hp8711;"SENS1:STAT ON;*WAI"
800 OUTPUT @Hp8711;"DISP:ANN:FREQ1:MODE SSTOP"
810 OUTPUT @Hp8711;"SENS1:FREQ:STAR 100 MHz;STOP 400 MHz;*WAI"
820 OUTPUT @Hp8711;"INIT1;*WAI"
830 OUTPUT @Hp8711;"MMEM:STOR:STAT 1,'MEM:STATE1.STA'"
840 RETURN
850 !
860 Save_2: OUTPUT @Hp8711;"SENS2:STAT ON;*WAI"
870 OUTPUT @Hp8711;"SENS2:FUNC 'XFR:POW:RAT 1,0';DET NBAN;*WAI"
880 OUTPUT @Hp8711;"DISP:ANN:FREQ2:MODE CSPAN"
890 OUTPUT @Hp8711;"SENS2:FREQ:CENT 200 MHz;SPAN 300 MHz;*WAI"
900 OUTPUT @Hp8711;"INIT2;*WAI"
910 OUTPUT @Hp8711;"MMEM:STOR:STAT 1,'MEM:STATE2.STA'"
920 RETURN
930 !
940 Save_3: OUTPUT @Hp8711;"CALC2:FORM SWR"
950 OUTPUT @Hp8711;"INIT2;*WAI"
960 OUTPUT @Hp8711;"MMEM:STOR:STAT 1,'MEM:STATE3.STA'"
970 OUTPUT @Hp8711;"CALC2:FORM MLOG"

```

```

980 RETURN
990 !
1000 Save_4:OUTPUT @Hp8711;"SENS2:STAT OFF"
1010 OUTPUT @Hp8711;"SENS1:SWE:POIN 1601;*WAI"
1020 OUTPUT @Hp8711;"INIT1;*WAI"
1030 OUTPUT @Hp8711;"MMEM:STOR:STAT 1,'MEM:STATE4.STA'"
1040 RETURN
1050 !
1060 Save_5:OUTPUT @Hp8711;"CALC1:MARK:BWID -3;FUNC:TRAC ON"
1070 OUTPUT @Hp8711;"INIT1;*WAI"
1080 OUTPUT @Hp8711;"MMEM:STOR:STAT 1,'MEM:STATE5.STA'"
1090 RETURN
1100 !
1110 Save_6:OUTPUT @Hp8711;"SENS1:BWID 250 Hz;*WAI"
1120 OUTPUT @Hp8711;"SENS1:SWE:POIN 101;*WAI"
1130 OUTPUT @Hp8711;"INIT1;*WAI"
1140 OUTPUT @Hp8711;"MMEM:STOR:STAT 1,'MEM:STATE6.STA'"
1150 RETURN
1160 !
1170 ! These six subroutines each recall one of the
1180 ! measurement setups that were stored earlier.
1190 !
1200 Load_1:DISP "Setup 1"
1210 OUTPUT @Hp8711;"MMEM:LOAD:STAT 1,'MEM:STATE1.STA';*WAI"
1220 OUTPUT @Hp8711;"INIT1;*WAI"
1230 RETURN
1240 !
1250 Load_2:DISP "Setup 2"
1260 OUTPUT @Hp8711;"MMEM:LOAD:STAT 1,'MEM:STATE2.STA';*WAI"
1270 OUTPUT @Hp8711;"INIT2;*WAI"
1280 RETURN
1290 !
1300 Load_3:DISP "Setup 3"
1310 OUTPUT @Hp8711;"MMEM:LOAD:STAT 1,'MEM:STATE3.STA';*WAI"
1320 OUTPUT @Hp8711;"INIT2;*WAI"
1330 RETURN
1340 !
1350 Load_4:DISP "Setup 4"
1360 OUTPUT @Hp8711;"MMEM:LOAD:STAT 1,'MEM:STATE4.STA';*WAI"
1370 OUTPUT @Hp8711;"INIT1;*WAI"
1380 RETURN
1390 !
1400 Load_5:DISP "Setup 5"
1410 OUTPUT @Hp8711;"MMEM:LOAD:STAT 1,'MEM:STATE5.STA';*WAI"
1420 OUTPUT @Hp8711;"INIT1;*WAI"
1430 RETURN
1440 !
1450 Load_6:DISP "Setup 6"
1460 OUTPUT @Hp8711;"MMEM:LOAD:STAT 1,'MEM:STATE6.STA';*WAI"
1470 OUTPUT @Hp8711;"INIT1;*WAI"

```

```
1480 RETURN
1490 END
```

## BAR — Using Barcode Reader

```
10 !-----
20 !
30 ! IBASIC program: BAR - Using barcode reader
40 !
50 ! This HP 8711 IBASIC program was written for a barcode
60 ! reader, but it is not required. Sets the 8711's
70 ! state depending on model # of DUT being measured.
80 ! Expects to see BARCODE with the following format:
90 ! Model Number (6 char), space, Serial Number (5 char)
100 ! Valid Models: BPF175, BPF200, SAW134
110 ! REV A.02.00 930615.JVV
120 !
130 !-----
140 !
150 COM /Hplib/ @Hp8711
160 COM /Scale/ Sc,INTEGER X,Y
170 DIM Name$[50],Stat$[50],Scan$[90],Lim$(1:3,1:5)[30],Test$(0:1)[4]
180 INTEGER Tab,Fail_flg,G(1:4)
190 !
200 Init: !
210 Test$(0)="PASS"
220 Test$(1)="FAIL"
230 ASSIGN @Hp8711 TO 800
240 Sc=1 ! Scales the 8711 drawing and DUT
250 X=5 ! Starting X posn of 8711 plot
260 Y=35 ! " Y "
270 Tab=38 ! Tab position for text
280 OUTPUT @Hp8711;"SYST:PRES;*OPC?"
290 ENTER @Hp8711;Opc
300 OUTPUT @Hp8711;"DISP:PROG UPP"
310 GINIT
320 GCLEAR
330 GESCAPE 1,3;G(*)
340 WINDOW G(1),G(3),G(2),G(4)
350 OUTPUT @Hp8711;"SENS1:STAT OFF;:SENS2:STAT ON"
360 OUTPUT @Hp8711;"DISP:WIND2:TRAC:Y:RPOS 9"
370 OUTPUT @Hp8711;"ABOR;:INIT:CONT OFF"
380 Setup: !
390 BEEP 500,.1
400 INPUT "Enter Operator's Name:",Name$
410 BEEP 3000,.03
420 INPUT "Enter Station Number:",Stat$
430 BEEP 3000,.03
440 OUTPUT @Hp8711;"SYST:DATE?"
```

```

450 ENTER @Hp8711;Year,Month,Day
460 CALL Draw_na      ! Draw HP8711.
470 Box(670,35,340,130)  ! Draw text box

480 PRINT TABXY(Tab,3);"Oper: ";Name$[1,15]
490 PRINT TABXY(Tab,4);"Station: ";Stat$[1,11]
500 PRINT TABXY(Tab,5);"Date: ";Year;Month;Day
510 Meas_dev: !
520 LOOP
530   CALL Draw_dut(1)
540   CALL Scan_dut(Scan$,Cent$,Span$,Loss$,Lim$(*))
550   PRINT TABXY(Tab,7);"Model: "&Scan$[1,6]
560   PRINT TABXY(Tab,8);"Serial: "&Scan$[8,12]
570   GOSUB Set_stim
580   DISP "MEASURING THE DEVICE"
590   OUTPUT @Hp8711;"ABOR;:INIT2:CONT OFF;:INIT2;*WAI"
600   OUTPUT @Hp8711;"CALC2:MARK1 ON;MARK:FUNC MAX"
610   OUTPUT @Hp8711;"CALC2:MARK1:Y?"
620   ENTER @Hp8711;Loss
630   PRINT TABXY(Tab,9);"Loss (dB): ";Loss
640 Disp_result: !
650   OUTPUT @Hp8711;"STAT:QUES:LIM:COND?"
660   ENTER @Hp8711;Fail_flg
670   Fail_flg=BIT(Fail_flg,1) ! Bit 1 is for ch2
680   IF Fail_flg THEN BEEP 2100,.5
690   Label(Test$(Fail_flg),125,50,24,5,0,1)
700 Continue: !
710   CALL Draw_dut(0)
720   BEEP 300,.05
730   INPUT "Disconnect DUT. Measure another? (Y/n)",Ans$
740   EXIT IF UPC$(Ans$[1,1])="N"
750   Label(Test$(Fail_flg),125,50,24,5,0,0)
760 END LOOP
770 OUTPUT @Hp8711;"ABOR;:INIT:CONT ON"
780 STOP
790 !
800 Set_stim: ! Set Freqs and Limit lines
810 OUTPUT @Hp8711;"DISP:ANN:FREQ:MODE CSPAN"
820 OUTPUT @Hp8711;"SENS:FREQ:CENT "&Cent$&" MHZ;SPAN "&Span$&" M HZ"
830 OUTPUT @Hp8711;"DISP:WIND2:TRAC:Y:RLEV";-PROUND(VAL(Loss$),1);"DB"
840 FOR I=1 TO 3 ! SET LIMIT LINES
850   OUTPUT @Hp8711;"CALC2:LIM:SEGM"&VAL$(I)&":TYPE "&Lim$(I,1)&";STAT ON"
860   OUTPUT @Hp8711;"CALC2:LIM:SEGM"&VAL$(I)&":FREQ:STAR "&Lim$(I,2)&
MHZ;STOP "&Lim$(I,3)&" MHZ"
870   OUTPUT @Hp8711;"CALC2:LIM:SEGM"&VAL$(I)&":AMPL:STAR "&Lim$(I,4)&" ;STOP
"&Lim$(I,5)
880 NEXT I
890 OUTPUT @Hp8711;"CALC2:LIM:DISP ON;STAT ON"
900 RETURN
910 END
920 ! ##### SUBPROGRAMS #####

```

```

930 !
940 Draw_na:SUB Draw_na
950 ! This draws HP 8711 at origin X,Y
960 Box(231,50,460,100) ! Frame
970 Box(231,50,462,102)
980 Box(125,50,180,72) ! CRT
990 Box(125,50,182,75)
1000 FOR I=19 TO 82 STEP 9 ! Keys
1010 Box(235,I,15,5)
1020 NEXT I
1030 Box(285,88,15,7) ! BEGIN
1040 Box(375,88,105,12) ! Drive
1050 Box(375,88,75,4)
1060 Circle(365,60,15) ! Knob
1070 Circle(300,15,10) ! Out
1080 Circle(410,15,10) ! in
1090 Box(15,20,7,10) ! Switch
1100 Circle(15,33,4)
1110 Label("RF OUT",300,28,8,5,0,1)
1120 Label("RF IN",410,28,8,5,0,1)
1130 SUBEND
1140 !
1150 Draw_dut:SUB Draw_dut(INTEGER Pen)
1160 ! This connects DUT to HP 8711
1170 PEN Pen
1180 Connect(300,15,320,-20,0)
1190 Box(355,-20,70,15)
1200 Connect(410,15,390,-20,0)
1210 PEN 1
1220 SUBEND
1230 !
1240 Scan_dut:SUB Scan_dut(Scan$,Cent$,Span$,Loss$,Lim$(*))
1250 LOOP
1260 Invalid=0
1270 Scan$="BPF175 12345" ! Default model/serial
1280 BEEP 500,.05
1290 INPUT "Connect and scan the Device.",Scan$ ! SCAN BARCODE HERE
1300 IF LEN(Scan$)<12 THEN !Valid device needs 12 char.
1310 Invalid=1
1320 ELSE
1330 Model$=Scan$[1,6]
1340 SELECT UPC$(TRIM$(Model$))
1350 CASE "BPF175","BPF177"
1360 RESTORE F1
1370 CASE "BPF200"
1380 RESTORE F2
1390 CASE "SAW134"
1400 RESTORE F3
1410 CASE ELSE

```

```

1420     Invalid=1
1430     END SELECT
1440     END IF
1450     EXIT IF NOT Invalid
1460     DISP Scan$;" <<--is INVALID! Try again."
1470     BEEP 1500,.2
1480     WAIT 1
1490     END LOOP
1500     BEEP 3000,.03
1510     READ Cent$,Span$,Loss$,Lim$(*)
1520     ! Limit lines format: Center, Span, Loss, (LIM TYPE, STRT, STP, STRTdB,
STPdB)
1530 F1:DATA 175,250,2 ! 175 MHz BPF
1540 DATA "LMIN", 160,190,-5,-5
1550 DATA "LMAX", 100,140,-50,-9
1560 DATA "LMAX", 210,240,-7,-30
1570 F2:DATA 200,100,1 ! 200 MHz BPF
1580 DATA "LMIN", 196,204,-3,-3
1590 DATA "LMAX", 180,190,-40,-10
1600 DATA "LMAX", 210,220,-10,-40
1610 F3:DATA 134,40,22 ! 134 MHz SAW BPF
1620 DATA "LMIN", 128,140,-27,-27
1630 DATA "LMAX", 123,125,-65,-30
1640 DATA "LMAX", 143,145,-30,-65
1650 SUBEND
1660 !
1670 Box:SUB Box(Xpos,Ypos,Xsize,Ysize)
1680 COM /Scale/ Sc,INTEGER X,Y
1690 MOVE X+(Xpos-Xsize/2)*Sc,Y+(Ypos-Ysize/2)*Sc
1700 RECTANGLE Xsize*Sc,Ysize*Sc*1.79 ! 1.79 = 8711 Pixel H:W Ratio
1710 SUBEND
1720 !
1730 Circle:SUB Circle(Xpos,Ypos,Radius)
1740 COM /Scale/ Sc,INTEGER X,Y
1750 MOVE X+Xpos*Sc,Y+Ypos*Sc
1760 POLYGON Radius*Sc,16,16
1770 SUBEND
1780 !
1790 Connect:SUB Connect(X1,Y1,X2,Y2,How)
1800 COM /Scale/ Sc,INTEGER X,Y
1810 MOVE X+X1*Sc,Y+Y1*Sc
1820 SELECT How
1830 CASE 1 ! ... diagonal
1840 DRAW X+X2*Sc,Y+Y2*Sc
1850 CASE 0
1860 DRAW X+X1*Sc,Y+Y2*Sc
1870 DRAW X+X2*Sc,Y+Y2*Sc
1880 CASE -1
1890 DRAW X+X2*Sc,Y+Y1*Sc
1900 DRAW X+X2*Sc,Y+Y2*Sc

```

```
1910 END SELECT
1920 SUBEND

1930 !
1940 Label:SUB Label(Text$,Xpos,Ypos,Size,Lorg,Ldr,Pen)
1950 COM /Scale/ Sc,INTEGER X,Y
1960 LORG Lorg
1970 LDIR Ldr
1980 CSIZE Size*Sc,1
1990 MOVE X+Xpos*Sc,Y+Ypos*Sc
2000 PEN Pen
2010 LABEL Text$
2020 PEN 1
2030 SUBEND
2040 !
2050 Amp:SUB Amp(Xpos,Ypos,Size) ! Draws > Triangle
2060 COM /Scale/ Sc,INTEGER X,Y
2070 MOVE X+(Xpos+Size/2)*Sc,Y+Ypos*Sc
2080 POLYGON Size*Sc,3,3
2090 SUBEND
```

## AVG — Using Barcode Reader

```
10  ! -----
20  !
30  ! IBASIC program: AVG - Using barcode reader
40  !
50  ! This HP 8711 IBASIC program uses a barcode reader.
60  ! Displays running average of selected BPF passbands.
70  ! Finds linear avg of log data (ie Avg of 1dB & 5dB =3)
80  ! Expects to see BARCODE with the following format:
90  ! Model Number (6 char), space, Serial Number (5 char)
100 ! Valid Models: BPF175, BPF200, SAW134
110 ! REV A.01.00  930615.JVV
120 !
130 ! -----
140 !
150 Init: !
160 COM /HpiB/ @Hp8711
170 COM Csub_loaded
180 DIM A(1:1601),M(1:1601)
190 INTEGER Points,N,I,Chan
200 Points=201 ! # of trace points
210 Chan=2
220 ASSIGN @Hp8711 TO 800
230 IF NOT Csub_loaded THEN
240   LOADSUB Read_fdata FROM "XFER:MEM 0,0"
250   LOADSUB Write_fmем FROM "XFER:MEM 0,0"
260   Csub_loaded=1
270 END IF
280 OUTPUT @Hp8711;"SYST:PRES;*OPC?"
290 ENTER @Hp8711;0pc
300 OUTPUT @Hp8711;"DISP:PROG UPP"
310 GINIT
320 GCLEAR
330 OUTPUT @Hp8711;"DISP:ANN:MESS:STAT 0"
340 OUTPUT @Hp8711;"SENS1:STAT OFF;:SENS2:STAT ON"
350 OUTPUT @Hp8711;"SENS2:SWE:POIN ";Points ! points
360 OUTPUT @Hp8711;"DISP:WIND2:TRAC:Y:RPOS 9;PDIV 1 DB;*OPC?"
370 ENTER @Hp8711;0pc
380 N=0
390 Setup: !
400 LOOP
410   GOSUB Scan_next
420   ON KEY 1 LABEL " AVER THIS DATA" GOSUB Avg_this
430   ON KEY 3 LABEL "SCAN   ANOTHER" GOSUB Scan_next
440   ON KEY 5 LABEL "DONE" GOSUB Exit
450   LOOP
460   DISP "SELECT A SOFTKEY."
470   WAIT 1
```

```

480     DISP
490     WAIT .3
500     END LOOP
510 END LOOP
520 !
530 Exit: !
540 CLEAR SCREEN
550 DISP "PROGRAM PAUSED!"
560 LOCAL @Hp8711
570 PAUSE
580 RETURN
590 !
600 Scan_next: !
610 LOOP
620     Scan_dut(Model$,Serial$,Cent$,Span$,Loss$)
630     IF Model$="ABORT" THEN GOTO Exit
640     IF NOT N THEN Curr_model$=Model$
650     EXIT IF Model$=Curr_model$
660     DISP "Inconsistent Model #, Try again!"
670     BEEP 2100,.1
680     WAIT 1
690 END LOOP
700 CLEAR SCREEN
710 PRINT TABXY(1,4);"Device currently under test:"
720 PRINT "Model # ";Model$;" Serial # ";Serial$
730 PRINT TABXY(1,6);"# Avg'd:";N
740 PRINT TABXY(1,7);"Status of Serial # "&Serial$&": MEASURING  "
750 GOSUB Set_stim
760 RETURN
770 !
780 Avg_this: !
790 PRINT TABXY(1,7);"Status of Serial # "&Serial$&": READING DATA"
800 Read_fdata(Chan,A(*))
810 N=N+1
820 PRINT TABXY(1,7);"Status of Serial # "&Serial$&": AVERAGING  "
830 IF N=1 THEN
840     MAT M= A
850     OUTPUT @Hp8711;"TRAC CH2SMEM,CH2SDATA"
860     OUTPUT @Hp8711;"CALC2:MATH (IMPL);:DISP:WIND2:TRAC1 ON;TRAC2 *ON;*WAI"
870     OUTPUT @Hp8711;"ABOR;:INIT2:CONT ON;*WAI"
880 ELSE
890     FOR I=1 TO Points
900         M(I)=(N-1)/N*M(I)+A(I)/N
910     NEXT I
920 END IF

930 PRINT TABXY(1,6);"# Averaged:";N
940 PRINT TABXY(1,7);"Status of Serial # "&Serial$&": WRITING DATA"
950 Write_fmем(Chan,M(*))
960 PRINT TABXY(1,7);"Status of Serial # "&Serial$&": AVG COMPLETE"
970 GOSUB Scan_next

```

```

980 RETURN
990 !
1000 Set_stim:! Set Freqs
1010 OUTPUT @Hp8711;"DISP:ANN:FREQ:MODE CSPAN"
1020 OUTPUT @Hp8711;"SENS:FREQ:CENT "&Cent$&" MHZ;SPAN "&Span$&" MHZ"
1030 OUTPUT @Hp8711;"DISP:WIND2:TRAC:Y:RLEV -"&Loss$&" DB;*OPC?"
1040 ENTER @Hp8711;Opc
1050 RETURN
1060 !
1070 END
1080 !
1090 ! ##### SUBPROGRAMS #####
1100 !
1110 Scan_dut:SUB Scan_dut(Model$,Serial$,Cent$,Span$,Loss$)
1120   ALLOCATE Scan$[80]
1130   LOOP
1140     Invalid=0
1150     Scan$="ABORT"
1160     Scan$="BPF175 12345" !##### These 3 lines for demo only
1170     S$=VAL$(RND*1.E+9) !##### Generates random S/N
1180     Scan$[8,12]=S$[3,7] !##### Delete all to enable abort.
1190     BEEP 500,.05
1200     INPUT "Connect & scan DUT or leave blank to exit.",Scan$ !SCAN *BARCODE
1210     IF LEN(Scan$)<12 THEN ! Valid device needs 12 char.
1220       Invalid=1
1230     ELSE
1240       Model$=Scan$[1,6]
1250       SELECT UPC$(TRIM$(Model$))
1260         CASE "BPF175","BPF177"
1270           RESTORE F1
1280         CASE "BPF200"
1290           RESTORE F2
1300         CASE "SAW134"
1310           RESTORE F3
1320         CASE ELSE
1330           Invalid=1
1340         END SELECT
1350       END IF
1360     EXIT IF NOT Invalid
1370     IF POS(UPC$(Scan$),"ABORT") THEN
1380       Model$="ABORT"
1390       SUBEXIT
1400     END IF
1410     DISP Scan$;" <<--is INVALID! Try again."
1420     BEEP 1500,.2
1430     WAIT 1
1440   END LOOP
1450   BEEP 3000,.03
1460   Serial$=Scan$[8,12]
1470   READ Cent$,Span$,Loss$

```

1480 ! Data format: Center, Span, Loss  
1490 F1: DATA 175,50,2 ! 175 MHz BPF  
1500 F2: DATA 200,12,1 ! 200 MHz BPF  
1510 F3: DATA 134,15,22 ! 134 MHz SAW BPF  
1520 SUBEND

## LOG — Using Barcode Reader

```
10  ! -----
20  !
30  ! IBASIC program: LOG - Using barcode reader
40  !
50  ! This HP 8711 IBASIC program uses a barcode reader.
60  ! Stores ASCII trace data in internal memory until full.
70  ! Then copies stored files to disc.
80  ! Expects to see BARCODE with the following format:
90  ! Model Number (6 char), space, Serial Number (5 char)
100 ! Valid Models: BPF175, BPF200, SAW134
110 ! REV A.01.00  930615.JVV
120 !
130 ! -----
140 !
150 Init: !
160 COM /Hpib/ @Hp8711
170 IF SYSTEM$("SYSTEM ID")="HP 8711" THEN
180   ASSIGN @Hp8711 TO 800
190 ELSE
200   CLEAR 716
210   ASSIGN @Hp8711 TO 716
220 END IF
230 OUTPUT @Hp8711;"SYST:PRES;*OPC?"
240 ENTER @Hp8711;0pc
250 OUTPUT @Hp8711;"DISP:PROG UPP"
260 GINIT
270 GCLEAR
280 GOSUB Warning ! May be deleted
290 OUTPUT @Hp8711;"DISP:ANN:MESS:STAT 0"
300 OUTPUT @Hp8711;"SENS1:STAT OFF;:SENS2:STAT ON"
310 OUTPUT @Hp8711;"SENS2:SWE:POIN 201" ! 201 points
320 OUTPUT @Hp8711;"DISP:WIND2:TRAC:Y:RPOS 9"
330 OUTPUT @Hp8711;"MMEM:MSIS 'MEM:'"
340 OUTPUT @Hp8711;"MMEM:INIT 'MEM:',DOS"
350 OUTPUT @Hp8711;"MMEM:STOR:STAT:IST OFF;CORR OFF;TRAC OFF;*OPC?"
360 ENTER @Hp8711;0pc
370 Setup: !
380 LOOP
390   GOSUB Scan_next
400   ON KEY 1 LABEL "STORE THIS DATA" GOSUB Stor_mem
410   ON KEY 2 LABEL "STORE MEM TO DISK" CALL Store_disk
420   ON KEY 3 LABEL "SCAN  ANOTHER" GOSUB Scan_next
430   ON KEY 5 LABEL "DONE" GOSUB Exit
440   LOOP
450   DISP "SELECT A SOFTKEY"
460   WAIT 1
470   DISP
```

```

480     WAIT .3
490     END LOOP
500     END LOOP
510     !
520 Exit: !
530     Store_disk
540     CLEAR SCREEN
550     DISP "PROGRAM PAUSED!"
560     LOCAL @Hp8711
570     PAUSE
580     RETURN
590     !
600 Scan_next: !
610     Scan_dut(Model$,Serial$,Cent$,Span$,Loss$)
620     IF Model$="ABORT" THEN GOTO Exit
630     CLEAR SCREEN
640     PRINT TABXY(1,3);"Device currently under test:"
650     PRINT
660     PRINT "Model # ";Model$;" Serial # ";Serial$
670     PRINT TABXY(1,7);"Status of Serial # "&Serial$&": MEASURING      "
680     GOSUB Set_stim
690     RETURN
700     !
710 Stor_mem: !
720     PRINT TABXY(1,7);"Status of Serial # "&Serial$&": STORING TO RAM"
730     Store_ram(Model$,Serial$)
740     PRINT TABXY(1,7);"Status of Serial # "&Serial$&": STORING DONE "
750     GOSUB Scan_next
760     RETURN
770     !
780 Set_stim: ! Set Freqs
790     OUTPUT @Hp8711;"DISP:ANN:FREQ:MODE CSPAN"
800     OUTPUT @Hp8711;"SENS:FREQ:CENT "&Cent$&" MHZ;SPAN "&Span$&" MHZ"
810     OUTPUT @Hp8711;"DISP:WIND2:TRAC:Y:RLEV -"&Loss$&" DB;*OPC?"
820     ENTER @Hp8711;Opc
830     RETURN
840     !
850 Warning: !
860     BEEP 3000,.3
870     PRINT TABXY(15,4);"WARNING!"
880     PRINT "This program will initialize the INTERNAL memory."
890     PRINT "All internally saved files will be lost!"
900     PRINT
910     PRINT "Do you wish to continue? (y/N)"
920     INPUT "Continue?",Ans$
930     CLEAR SCREEN
940     IF UPC$(Ans$[1,1])="Y" THEN RETURN
950     END

960     !
970     ! ##### SUBPROGRAMS #####

```

```

980 !
990 Scan_dut:SUB Scan_dut(Model$,Serial$,Cent$,Span$,Loss$)
1000  ALLOCATE Scan$[80]
1010  LOOP
1020      Invalid=0
1030      Scan$="ABORT"
1040      Scan$="BPF175 12345" !##### These 3 lines for demo only
1050      S$=VAL$(RND*1.E+9) !##### Generates random S/N
1060      Scan$[8,12]=S$[3,7] !##### Delete all to enable abort.
1070      BEEP 500,.05
1080      INPUT "Connect & scan DUT or leave blank to exit.",Scan$ !SCAN *BARCODE
1090      IF LEN(Scan$)<12 THEN ! Valid device needs 12 char.
1100          Invalid=1
1110      ELSE
1120          Model$=Scan$[1,6]
1130          SELECT UPC$(TRIM$(Model$))
1140          CASE "BPF175","BPF177"
1150              RESTORE F1
1160          CASE "BPF200"
1170              RESTORE F2
1180          CASE "SAW134"
1190              RESTORE F3
1200          CASE ELSE
1210              Invalid=1
1220          END SELECT
1230      END IF
1240      EXIT IF NOT Invalid
1250      IF POS(UPC$(Scan$),"ABORT") THEN
1260          Model$="ABORT"
1270          SUBEXIT
1280      END IF
1290      DISP Scan$;" <<--is INVALID! Try again."
1300      BEEP 1500,.2
1310      WAIT 1
1320      END LOOP
1330      BEEP 3000,.03
1340      Serial$=Scan$[8,12]
1350      READ Cent$,Span$,Loss$
1360      ! Data format: Center, Span, Loss
1370 F1: DATA 175,300,2 ! 175 MHz BPF
1380 F2: DATA 200,100,1 ! 200 MHz BPF
1390 F3: DATA 134,30,22 ! 134 MHz SAW BPF
1400 SUBEND

1410 !
1420 SUB Store_ram(Model$,Serial$)
1430  COM /Hplib/ @Hp8711
1440  Id$=Model$[3,4]&"_"&Serial$ ! 2 unique chars + Ser
1450  ALLOCATE Err$[80]
1460  DISABLE
1470  REPEAT

```

```

1480     OUTPUT @Hp8711;"*CLS"
1490     OUTPUT @Hp8711;"MMEM:MSIS 'MEM:'"
1500     OUTPUT @Hp8711;"MMEM:STOR:TRAC CH2FDATA, '&Id$&";*WAI"
1510     OUTPUT @Hp8711;"SYST:ERR?"
1520     ENTER @Hp8711;Err$
1530     SELECT VAL(Err$)
1540     CASE 0    ! No Problem
1550     CASE -254 ! Internal Mem full
1560         CALL Store_disk
1570     CASE -257 ! dupl file name
1580         OUTPUT @Hp8711;"MMEM:DEL '&Id$&";*WAI" !ERASE OLD
1590     CASE ELSE
1600         BEEP 2000,.5
1610         DISP Err$;
1620         INPUT " Fix, Press ENTER",Ans$
1630     END SELECT
1640     UNTIL VAL(Err$)=0
1650     ENABLE
1660 SUBEND
1670 !
1680 SUB Store_disk
1690     COM /Hplib/ @Hp8711
1700     ALLOCATE Err$[80]
1710     BEEP 700,.1
1720     DISP "Standby: Transferring internal files to disk."
1730     LOOP
1740         OUTPUT @Hp8711;"*CLS"
1750         OUTPUT @Hp8711;"MMEM:COPY '*.*' , 'INT:';*WAI"
1760         OUTPUT @Hp8711;"SYST:ERR?"
1770         ENTER @Hp8711;Err$
1780         EXIT IF NOT VAL(Err$)
1790         GOSUB Trap_err
1800     END LOOP
1810     OUTPUT @Hp8711;"MMEM:MSIS 'MEM:';DEL '*.*'"
1820     SUBEXIT
1830     !
1840 Trap_err: !
1850     IF VAL(Err$)=-250 THEN SUBEXIT! no file to xfer
1860     BEEP 2000,.5
1870     CLEAR SCREEN
1880     PRINT TABXY(1,4);"DISK ERROR DETECTED"
1890     PRINT "*** "&Err$&" ***"
1900     INPUT "Fix above problem, then press ENTER",Ans$
1910     CLEAR SCREEN
1920 SUBEND

```



YES



BPF175 00123



NO



BPF175 00156



RUN



BPF175 00560



PAUSE



BPF175 00758



ABORT



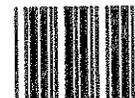
BPF175 01023



CONT



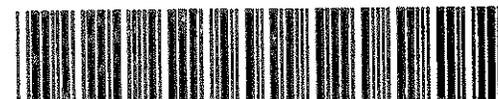
BPF200 00045



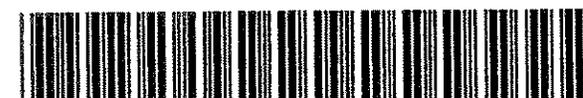
ENTER (*Blank*)



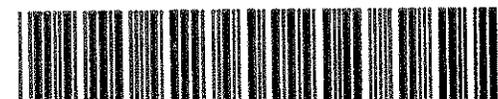
BPF200 00059



BENCH 4500 (*Station*)



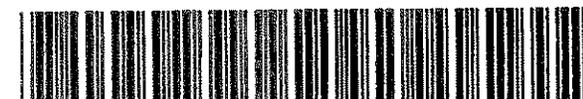
BPF200 00077



BENCH 5300 (*Station*)



SAW134 00707



J VALLELUNGA



SAW134 00721



LENE CORRIGAN



SAW134 00776

Figure 11-1. Sample Barcodes