Agilent 35670A Dynamic Signal Analyzer

Using Instrument Basic with the Agilent 35670A

For Instruments with Firmware Revision
A.00.00

Agilent Technologies

Agilent Part Number 35670-90049
Printed in U.S.A.

Print Date: November 2000

8600 Soper Hill Road Everett, Washington 98205-1209 U.S.A.

NOTICE

The information contained in this document is subject to change without notice.

Agilent Technologies MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MANUAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Agilent Technologies shall not be liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

WARRANTY

A copy of the specific warranty terms applicable to your Agilent Technologies product and replacement parts can be obtained from your local Sales and Service Office.

Use of this manual and flexible disc(s) or tape cartridge(s) supplied for this pack is restricted

to this product only. Additional copies of the programs can be made for security and back-up purposes only.

RESTRICTED RIGHTS LEGEND

Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause in DFARS 252.227-7013.

Agilent Technologies , Inc.

395 Page Mill Road

Palo Alto, CA

94303-0870, USA

Rights for non-DOD U.S. Governament Departments and Agencies are as set forth in FAR 52.227-19(c)(1,2)

Saftey Summary

The following general safety precautions must be observed during all phases of operation of this instrument. Failure to comply with these precautions or with specific warnings elsewhere in this manual violates safety standards of design, manufacture, and intended use of the instrument. Agilent Technologies, Inc. assumes no liability for the customer's failure to comply with these requirements.

GENERAL

This product is a Safety Class 1 instrument (provided with a protective earth terminal). The protective features of this product may be impaired if it is used in a manner not specified in the operation instructions.

All Light Emitting Diodes (LEDs) used in this product are Class 1 LEDs as per IEC 60825-1.

ENVIRONMENTAL CONDITIONS

This instrument is intended for indoor use in an installation category II, pollution degree 2 environment. It is designed to operate at a maximum relative humidity of 95% and at altitudes of up to 2000 meters. Refer to the specifications tables for the ac mains voltage requirements and ambient operating temperature range.

BEFORE APPLYING POWER

Verify that the product is set to match the available line voltage, the correct fuse is installed, and all safety precautions are taken. Note the instrument's external markings described under Safety Symbols.

GROUND THE INSTRUMENT

To minimize shock hazard, the instrument chassis and cover must be connected to an electrical protective earth ground. The instrument must be connected to the ac power mains through a grounded power cable, with the ground wire firmly connected to an electrical ground (safety ground) at the power outlet. Any interruption of the protective (grounding) conductor or disconnection of the protective earth terminal will cause a potential shock hazard that could result in personal injury.

FUSES

Only fuses with the required rated current, voltage, and specified type (normal blow, time delay, etc.) should be used. Do not use repaired fuses or short-circuited fuse holders. To do so could cause a shock or fire hazard.

DO NOT OPERATE IN AN EXPLOSIVE ATMOSPHERE

Do not operate the instrument in the presence of flammable gases or fumes.

DO NOT REMOVE THE INSTRUMENT COVER

Operating personnel must not remove instrument covers. Component replacement and internal adjustments must be made only by qualified service personnel.

Instruments that appear damaged or defective should be made inoperative and secured against unintended operation until they can be repaired by qualified service personnel.

WARNING

The WARNING sign denotes a hazard. It calls attention to a procedure, practice, or the like, which, if not correctly performed or adhered to, could result in personal injury. Do not proceed beyond a WARNING sign until the indicated conditions are fully understood and met.

Caution

The CAUTION sign denotes a hazard. It calls attention to an operating procedure, or the like, which, if not correctly performed or adhered to, could result in damage to or destruction of part or all of the product. Do not proceed beyond a CAUTION sign until the indicated conditions are fully understood and met.

Safety Symbols

 Warning, risk of electric shock

 Caution, refer to accompanying documents

 Alternating current

 Both direct and alternating current

 Earth (ground) terminal

 Protective earth (ground) terminal

 Frame or chassis terminal

 Terminal is at earth potential.

 Standby (supply). Units with this symbol are not completely disconnected from ac mains when this switch is off

Table of Contents

Chapter 4: Saving and Recalling Programs

Chapter 5: Developing Programs

Chapter 6: Debugging Programs

Chapter 7: Graphics and Display Techniques

Chapter 8: Interfacing with the GPIB

Chapter 9: Interfacing with the RS-232-C Serial Port

Chapter 10: Interfacing with the Parallel Port

Chapter 11: Example Programs

Chapter 12: Instrument-Specific Instrument Basic Features

1

Introduction

1

Introduction

Welcome to Instrument Basic

This manual will help you learn about using your Instrument BASIC software on the Agilent 35670A. It shows you how to use the programming, editing and debugging features of Instrument BASIC. It also describes how to save and recall programs and how the Agilent 35670A implements Instrument BASIC features.

An additional aid is online help, which provides key-specific information on Instrument Basic features. This help is accessed in the same manner as it is for other features of the Agilent 35670A. Press the [**Help**] hardkey followed by the desired hardkey or softkey—or use the index.

Instrument BASIC Applications

Instrument BASIC can be used for a wide range of applications, from simple recording and playback of measurement sequences, to remote control of other instruments.

Instrument BASIC is a complete system controller residing inside your analyzer. It communicates with your analyzer via GPIB commands and can also communicate with other instruments, computers and peripherals over the GPIB interface.

Using Instrument BASIC

You need not be proficient in a programming language to successfully use Instrument BASIC. With keystroke recording, Instrument BASIC automatically builds an executable program by capturing measurement sequences as they are performed. With little or no editing of this generated code, you can put your program to work immediately controlling and automating your Agilent 35670A.

Instrument BASIC's programming interface includes an editor, a debugging program, and a set of programming utilities. The utilities allow you to set memory size as well as renumber, secure, indent, or delete your program. The remaining softkeys allow you to run or continue a program, print a listing or configure the display.

You can have up to five programs in memory at one time. Each has its own softkey that runs the program and appears in the [**BASIC**] menu. You can customize the program softkey label. You can obtain a program listing by pressing a softkey in one of the Instrument Basic menus.

The Instrument BASIC command set is similar to the command set of Agilent 9000 Series 300 BASIC. Instrument BASIC programs can run on any BASIC workstation with few, if any, changes. Refer to chapter 8, "Interfacing with the GPIB," for information on interfacing the Agilent 9000 Series 300 BASIC and Instrument BASIC environments. Porting information is located in the " Instrument BASIC Language Reference" section of the *Instrument Basic Users Handbook*.

How to Use This Manual

Read chapters 1 through 4 to learn how to record, run, save and recall programs with a minimum of editing and programming. This information is generally adequate for those who only need Instrument BASIC to record their measurement tasks.

Read chapter 5, “Developing Programs,” and chapter 6, “Debugging Programs” to learn how to edit programs with the front panel or with a keyboard.

Read chapter 7, “Display and Graphics Techniques,” to understand how Instrument BASIC’s graphics features apply to the Agilent 35670A.

Read chapter 8, “Interfacing with the GPIB,” to understand how the Instrument BASIC controller interfaces with external devices (such as plotters) and external controllers (such as Agilent 9000 Series 300 controllers).

Read chapters 9 and 10, to understand how the Instrument Basic controller interfaces with devices connected to the analyzer’s serial and parallel ports.

Refer to chapter 11 for example programs written in Instrument BASIC to run on the Agilent 35670A.

Chapter 12 couples this manual with the Instrument BASIC Users Handbook. The handbook, which serves users of Instrument BASIC on all instrument platforms, contains three sections:

- “Instrument Basic Programming Techniques”
- “ Instrument BASIC Interfacing Techniques”
- “ Instrument BASIC Language Reference”

Chapter 12 clarifies which parts of the handbook do not apply to the Agilent 35670A.

Typographical Conventions

The following conventions are used in this manual when referring to various parts of the Instrument BASIC and Agilent 35670A operating environments:

[**Hardkey**]	Brackets [ ] surrounding a bold-faced name indicate the name of a hardkey on the front panel of the Agilent 35670A.
[SOFTKEY]	Brackets [ ] surrounding a name indicate the name of a softkey.
[SOFTKEY **ON** OFF]	Bolded selection in a softkey indicates the state after `the softkey is pressed.`
[**Hardkey**] [SOFTKEY] [SOFTKEY]	A series of hardkeys and softkeys represents the path to a given softkey or menu.
[*Key*]	Brackets [ ] surrounding an italic typeface indicate the name of a key on the keyboard which can be used to edit Instrument Basic programs.
Italic	Italic typeface is used when referring to the name of a different manual. It is also used to emphasize a particular word or phrase.
< element >	Angle brackets are used to signify a syntax element in a statement.

Other Sources of Information

- *Agilent 35670A Operator's Guide*
- *Agilent 35670A Online Help*
- *Instrument Basic Users Handbook*:
  - Instrument Basic Programming Techniques
  - Instrument Basic Interfacing Techniques
  - Instrument Basic Language Reference

Need Assistance?

If you need assistance, contact your nearest Agilent Technologies Sales and Service Office listed in the Agilent Catalog, or contact your nearest regional office listed at the back of this guide. If you are contacting Agilent Technologies about a problem with your Agilent 35670A Dynamic Signal Analyzer, please provide the following information:

- Model number: Agilent 35670A
- Serial number and firmware version:

(To locate the analyzer's serial number and firmware version, press [**System Utility**] [S/N VERSION].)

- Options:
- Date the problem was first encountered:
- Circumstances in which the problem was encountered:
- Can you reproduce the problem?
- What effect does this problem have on you?

2

Recording Programs

2

Recording Programs

Keystroke Recording

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

You can record your program into any one of the five available locations in memory. Only one of these memory locations is active at any one time. You can select any one of the five memory locations as the currently active program.

What is Keystroke Recording?

Keystroke recording is a way to automatically create Instrument BASIC measurement sequence programs.

To enabling recording, press:

[**BASIC**]
[INSTRUMNT BASIC]
[ENABLE RECORDING]

Press the normal key sequences of a measurement on the analyzer. To stop recording, press the [**BASIC**] hardkey. To run the program, press the appropriate [RUN PROGRAM] softkey in the [**BASIC**] menu.

Instrument BASIC programs communicate with the analyzer over an internal bus. Instrument Basic uses the same set of commands that external controllers use for remote operation of the instrument. Keystroke recording works by finding the bus command, called an GPIB command, that fits each operation you perform from the front panel. It builds a program line that duplicates that operation when executed.

All program lines built by keystroke recording are entered into the analyzer's program memory. If the memory location does not contain any code, a complete executable program is inserted. If program statements exist in the memory location when recording is turned on, the recorded statements are inserted into the existing code. Chapter 5, "Developing Programs," describes how to record into existing programs.

Instrument BASIC Programs and the GPIB Buffer

Recorded programs work by sending GPIB commands to the analyzer. The analyzer queues the GPIB commands into its input buffer. An Instrument BASIC program generally outputs the commands much faster than the analyzer can execute them. The program often completes before the analyzer finishes executing the commands in the input buffer. The analyzer continues to process these commands until the buffer is empty.

This can be a problem if you are not aware of the possible delay. For example, it may not be obvious that the program has completed, since the analyzer is still functioning. This could cause confusion if you try to pause and continue a program that has actually finished.

You can clear the analyzer's input buffer by inserting the statement "CLEAR 8" at the beginning of your program. Refer to chapter 5 for more information on developing and editing programs.

What's in a Recorded Program

Any program created with keystroke recording is composed of three fundamental Instrument BASIC statements:

- ASSIGN
- OUTPUT
- END

The following simple program demonstrates these statements:

```
1 ASSIGN @Agilent35670a TO 800
2 OUTPUT @Agilent35670a;"FREQ:SPAN:FULL"
10 END
```

There is only one ASSIGN statement at the beginning of a program and only one END statement at the end, but in a typical program there are many OUTPUT statements. The OUTPUT statement does the actual work of controlling the Agilent 35670A.

The OUTPUT Statement

The Instrument BASIC statement

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

essentially 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 Instrument BASIC programs it is a string containing instructions to the analyzer.

The following command represents a typical OUTPUT statement generated from a recording session:

```
OUTPUT @Agilent35670a;"FREQ:SPAN:FULL"
```

The OUTPUT command is followed by a name representing the device selector (@Agilent35670a), followed by a semicolon, followed by the data. The data is in quotes ("FREQ:SPAN:FULL") and contains an instruction to the analyzer.

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 "@Agilent35670a." The following line appears in all recorded programs before any OUTPUT statements:

```
ASSIGN @Agilent35670a TO 800
```

The ASSIGN statement substitutes an I/O path name (a variable name preceded by the @ symbol) for a device selector number. After the above ASSIGN statement, the program line:

```
OUTPUT @Agilent35670a;"FREQ:SPAN:FULL"
```

is equivalent to:

```
OUTPUT 800;"FREQ:SPAN:FULL"
```

The device selector 800 specifies the host instrument as the destination of any data sent by the OUTPUT command. The program communicates with the analyzer via select code 8, the internal GPIB interface. This select code is used solely for communication between Instrument BASIC programs and the analyzer. The analyzer responds to any address on the internal interface from 800 to 899. (800 is typically used.)

GPIB Commands

The data sent to the analyzer by the OUTPUT command is called an GPIB command. Many of

the Agilent 35670 GPIB commands conform to SCPI—the Standard Commands for Programmable Instruments The GPIB command is found in quotes following the device selector path name and semicolon:

```
2 OUTPUT @Agilent35670a;"FREQ:SPAN:FULL"
```

The GPIB commands used in Instrument BASIC are the same ones used to remotely control the analyzer from an external computer. External computers communicate with the analyzer over the external bus while Instrument BASIC programs communicate with it over the internal bus. In our example, "FREQ:SPAN:FULL" tells the analyzer to set the start frequency to its minimum value and the stop frequency to its maximum value.

Note Many, but not all of the Agilent 35670A's GPIB commands conform to SCPI. Refer to GPIB Programming with the Agilent 35670A for a complete description of GPIB commands, including compliance to SCPI.

For more information on interfacing Instrument BASIC with a bus, see chapter 8, "Interfacing with the GPIB."

How Recording Works

To fully understand Instrument BASIC recording, it is important to understand the relationship between the analyzer's front panel operation and the program that is generated to emulate that operation.

Note GPIB commands entered in a program during a recording session do not necessarily have a one-to-one correlation with the actual keys that are pressed during that session.

It is important to know that GPIB commands correspond to an operation—not to the front panel's hardkeys and softkeys. It may take several keystrokes to perform an operation. Keystroke recording generates the appropriate GPIB command after you have pressed a valid sequence of keys.

In other words, the functional operation of the analyzer is recorded, not the exact series of keystrokes.

For example, recording the key sequence:

[**Freq**]
[FULL SPAN]

requires two keystrokes but produces only one command. The command, "FREQ:SPAN:FULL," is generated after the sequence is completed. Keystroke recording automatically formats this operation into the statement:

```
OUTPUT @Agilent35670a;"FREQ:SPAN:FULL"
```

and inserts it into the program.

If you accidentally press the wrong key in a sequence, it may not appear in the recorded program. It also means that you cannot exactly mimic keystrokes to leave the analyzer in a specific front-panel state. The analyzer's state appears only as a natural consequence of a completed operation.

For example, in the above example, pressing [**Freq**] in a recording session has the effect of bringing up the [**Freq**] menu. However, it does not, by itself, generate a line of code. You could not, therefore, set the analyzer to display the [**Freq**] menu.

Operations That Are Not Recorded

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

- Front panel operations with no corresponding GPIB command such as help text operations, GPIB controller status, RPG (knob) operations and transitional key sequences.
- Operations requiring additional programming steps, such as passing control to the analyzer for plotting or special handling of measurement operations which arm a trigger.
- GPIB operations with no equivalent front panel operations such as GPIB query commands.

Front Panel Operations Without GPIB Commands

There are some front panel operations which have no corresponding GPIB commands.

The help text available through the [**Help**] hardkey has no corresponding GPIB command. Help cannot be accessed from the GPIB. Therefore, the keystroke is not recorded.

You cannot remotely change the analyzer's controller status. This has two significant consequences:

- You cannot remotely change the state of the GPIB interface. For example, you cannot change the analyzer from Addressable Only to the System Controller.
- You cannot remotely abort an I/O operation when the analyzer has active control of the GPIB interface. I/O operations are for printing, plotting or using an external disk drive.

Any front-panel key sequences that perform these operations do not generate an GPIB command. They are not keystroke recorded.

You must use the numeric keys to enter a numeric value. Even if a front panel operation allows you to increment or to decrement a value by turning the knob, the entry is not recorded.

During a measurement sequence it may take several key presses to reach an operation that generates a command. The transitional sequences between actual instrument events are not recorded.

Any default settings you do not select while recording are not recorded.

Note It is important to remember instrument settings not specifically selected or changed *are not recorded*.

Since default states are not recorded, you must actively select them to generate a program statement. An alternate method is to make sure the analyzer is in the *same exact state* when the program runs as it was when the program was recorded. This is discussed later in this chapter in "Avoiding Recording Errors."

Instrument BASIC Operations

Softkeys under the [**BASIC**] key cannot be recorded because pressing this key turns off keystroke recording. In addition, [**Save/Recall**] operations that refer to an external disk drive or to another Instrument Basic program are not recorded. You can, however, record all the other save and recall operations which do not refer to Instrument BASIC programs or to an external disk drive.

Although operations in the [**BASIC**] menus cannot be recorded, many do have corresponding GPIB commands that allow an external controller to control and communicate with internal Instrument BASIC programs. See the example program TWO_CTRL in chapter 11 for more information.

Operations Requiring Additional Programming

Some operations that work well when performed from the front panel have special circumstances that need additional attention when used in an Instrument Basic program. These operations are synchronization and active control.

Synchronization

You must always anticipate timing and synchronization when 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, you may want your program to manually arm the trigger for several measurements, but only after each measurement has successfully completed. You can record the command to set the analyzer to manual arm mode, and the command to manually arm the trigger, by pressing key sequences. However, to detect when the analyzer has completed a measurement, you must edit the program and include a routine that waits for a status register to indicate the event has occurred. (For an example of this kind of program, see the MANARM program in chapter 11.)

Active Control

The [START PLOT/PRNT] operation, as well as any external disk drive operation, requires the analyzer to be the active controller on the external bus. This means that the analyzer must be set as the System Controller before the program runs; or, an external controller must pass active control to the analyzer. The instrument's active control of the external interface is automatically passed to the Instrument BASIC program when it begins running. Active control must be passed back to the analyzer before it can execute the print/plot or external disk operations.

Although you can keystroke record operations involving an external disk drive or [START PLOT/PRNT], you cannot successfully run the generated program. You need to add program lines to first pass active control to the analyzer and then wait for the active control of the bus to be passed back to the Instrument Basic program. See "Passing and Regaining Control" in chapter 8 for an example of passing control to the analyzer.

Operations Not Available From The Front Panel

Operations such as querying the analyzer's status, setting and clearing status registers, transferring data via the external bus, and transferring data via the serial or parallel ports are not available from the analyzer's front panel. These operations cannot be keystroke recorded, but they are useful for GPIB programming using Instrument BASIC. Refer to *GPIB Programming with the Agilent 35670A*, and to chapters 8, 9, and 10 in this manual for information about these types of operations.

Avoiding Recording Errors

This section describes ways to minimize the mistakes you can make when using keystroke recording.

Use Preset

You should preset the analyzer before recording a measurement sequence and again before running the recorded program. This sets the instrument to its default state. It 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 runs.

To include the command that presets the analyzer, press [**Preset**] [Do Preset] immediately after enabling keystroke recording. This inserts the following line before all the other OUTPUT statements in your program:

```
OUTPUT @Agilent35670a;"SYST:PRES"
```

This sets the analyzer to its default state.

Selecting Specific Parameters

You may not want to preset the analyzer before a recorded program runs because you are recording a section of a larger measurement sequence. In this case, be sure to activate every instrument setting you need in your automated sequence. For example, if you want the format to be SINGLE, press [**Disp Format**] and then [SINGLE], even though SINGLE is already the default setting. This generates a program line which specifically sets the format to SINGLE.

Use GPIB Echo

You can review the GPIB commands 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 an GPIB command is completed.

GPIB Echo is a facility that allows you to view GPIB commands corresponding to any operation executed from the front panel. A command appears in the upper left corner of the display (the third line) as you complete any key sequence that has a matching GPIB command. See figure 2-1. This command is the same as those generated in your recorded program during a recording session.

At power-up, the default status of GPIB Echo is off. To turn on GPIB Echo, press:

[**Local/GPIB**]

[GPIB ECHO **ON** OFF]

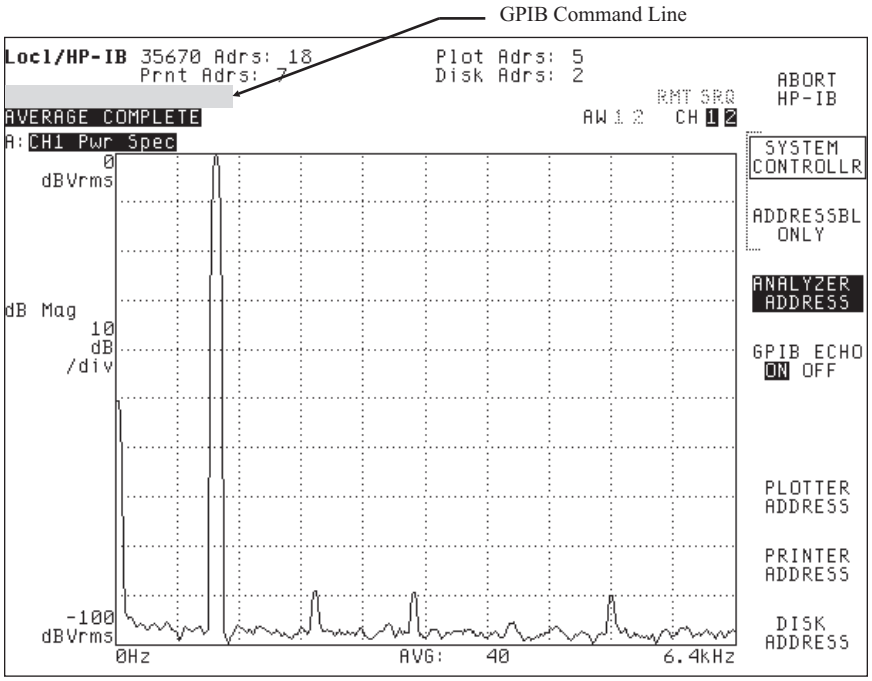

Figure 2-1. GPIB Echo

Program Buffers and the Active Program

You can record, load, or save your programs into any one of five available locations in memory called "program buffers." Only one of these buffers is active at any one time. This is called the active program. The active program defaults to Program 1 at power-up.

You can run any one of the five possible programs by pressing the softkey corresponding to that program in the [**BASIC**] menu. It becomes the active program. The name of the active program appears in the status line at the top of the screen. See figure 2-2.

You can also run the *currently active program* by pressing the [RUN PROGRAM] softkey in the [INSTRUMNT BASIC] menu.

Selecting the Active Program

Selecting any one of five possible programs to be the active program is simple. From the [**BASIC**] menu, press [INSTRUMNT BASIC] and then press [SELECT PROGRAM]. The five program softkeys are presented. Press the softkey of the program you want to be the active program. See figure 2-2.

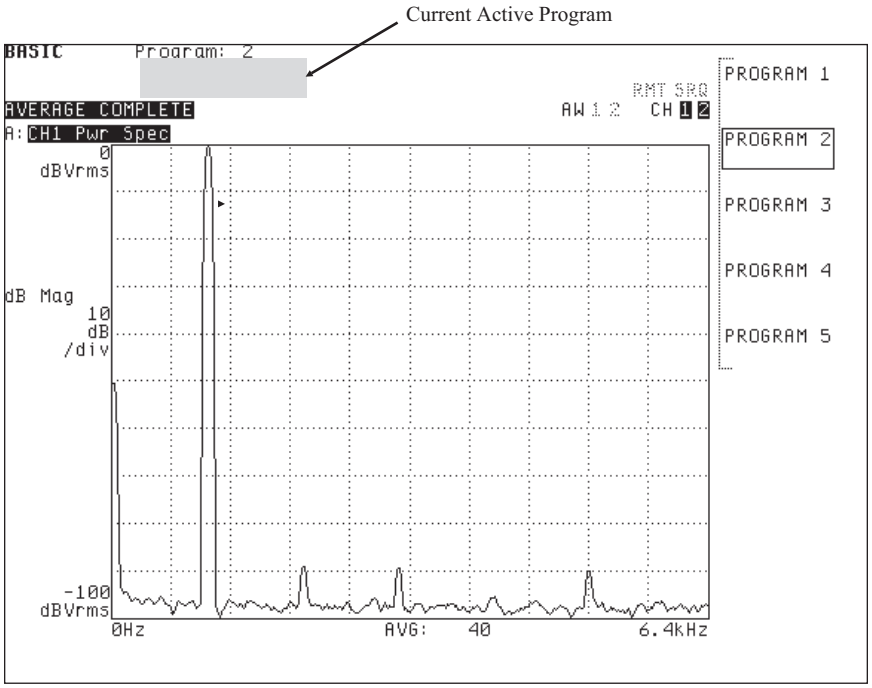

Figure 2-2. The [[Select Program] Menu

To record a front-panel operation into a non-active program, select the program with the [SELECT PROGRAM] menu, then press the [ENABLE RECORDING] softkey.

For example, the following keystrokes, select Program 4 as the active program and record a single front-panel operation.

[**BASIC**]
[INSTRUMENT BASIC]
[SELECT PROGRAM]
[PROGRAM 4]

[**Rtn**]
[ENABLE RECORDING]

[**Disp Format**]
[UPPER/LOWER]

[**BASIC**]

Program 4 is the active program and contains a three line program. Although it is very short, it is a representative recorded program.

Changing a Program Label

You can change the softkey label for the active program. Press the [LABEL PROGRAM] softkey in the [INSTRUMNT BASIC] menu. The alpha entry menu appears on the display. The current label appears in an entry window at the top of the screen. Use the menu and the alpha keys (or your keyboard) to change the name. When you finish editing the name, press [ENTER].

You can change the softkey label for each of the five program buffers by first selecting the appropriate program as the active program.

3

Controlling Programs

3

Controlling Programs

You can start, pause and stop an Instrument BASIC program from the Agilent 35670A front panel using various hardkeys and softkeys. This chapter describes how to control an Instrument Basic program.

GPIB commands can control Instrument Basic programs over the external bus. You can use an external controller to run Instrument Basic programs. For information on running, pausing and stopping programs from an external controller see chapter 8, “Interfacing with the GPIB.”

Running and Continuing a Program

The [**BASIC**] menu displays five program softkeys, corresponding to the five program buffers. See figure 3-1. The status line at the top of the screen indicates which program is currently the active program. These softkeys are initially labeled [RUN PROGRAM 1] through [RUN PROGRAM 5], but can be changed to display your own labels (see "Changing a Program Label" in chapter 2). Pressing one of these softkeys selects that program as the currently active program and runs the stored program. This menu gives you immediate access to running any one of five Instrument Basic programs.

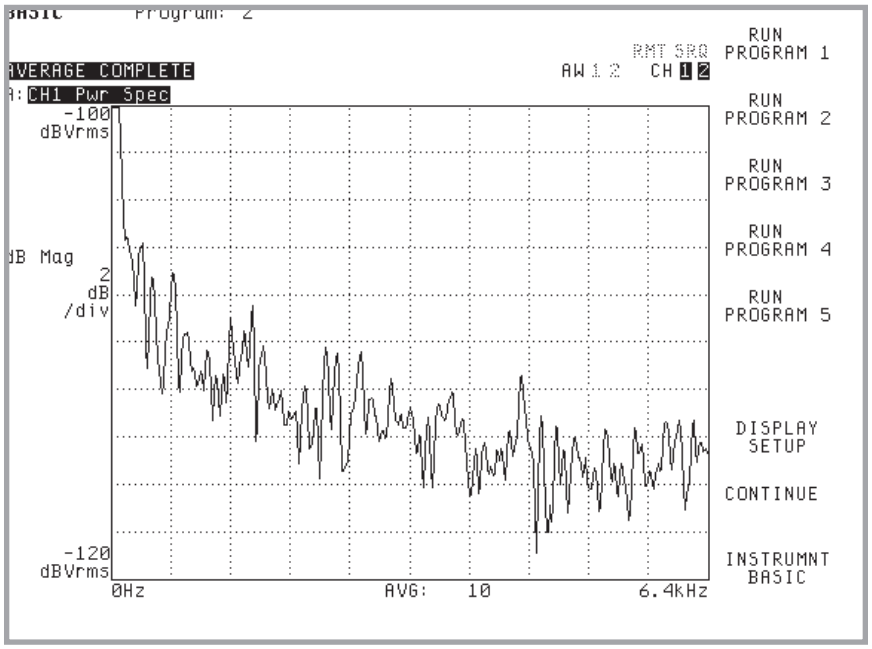

Figure 3-1. The [[BASIC] Menu

In the [**BASIC**] menu you will also find the [INSTRUMNT BASIC] softkey. Pressing this key brings you into the Instrument Basic operating environment. The menu that's displayed applies only to the currently active program.

To run the active program, press:

[**BASIC**]

[INSTRUMNT BASIC]

[RUN PROGRAM]

There is also a [RUN PROGRAM] key in the [**BASIC**] [INSTRUMNT BASIC] [DEBUG] menu. This key allows you to run the currently active program during program debugging (see chapter 6, "Debugging Programs"). Both of these softkeys perform the same Instrument BASIC RUN command.

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

The prerun phase consists of:

- Reserving memory space for variables specified in COM statements (both labeled and blank).
- Reserving memory space for variables specified by DIM, REAL, INTEGER, 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 result from multiple program statements. Incorrect array references and mismatched parameters or COM lists are examples of these types of syntax errors.

After the prerun phase successfully completes, the program continues executing until one of the following events occurs:

- The program encounters an END or STOP statement.
- The program encounters a PAUSE statement.
- You press the [**Local/GPIB**] hardkey to stop the program.
- You press the [**BASIC**] hardkey to pause the program.
- You press the [**Preset**] hardkey to stop the program.

Pausing a Program

You can pause a program by pressing the [**BASIC**] hardkey. Another way to pause a program is to insert a PAUSE statement into your program. (Refer to chapter 5, "Developing Programs," to learn how to insert statements into your recorded program.) In either case, the analyzer temporarily stops executing the program.

To continue a paused program, press the [CONTINUE] softkey in the [**BASIC**] menu or in the [INSTRUMNT BASIC] [DEBUG] menu. Continuing a paused program resumes the operation from where it was paused in the program. The program retains the values for any variables.

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 from the front panel after pausing a program that has left a file open on that medium:

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

An Instrument BASIC "RESET" closes all open files. Press the [**Local/GPIB**] hardkey while the program is running or press the [RESET] softkey in the [**BASIC**] [INSTRUMNT BASIC] [DEBUG] menu when the program is paused. There is one exception. An Instrument Basic "RESET" does not close a file if it is the device for the PRINTER IS statement.

Keystroke recorded programs do not open files and therefore avoid this problem.

Stopping a Program

To completely stop a program, press the [**Local/GPIB**] hardkey at any time while the program is running. This causes an Instrument BASIC "RESET." If the analyzer is under remote control, pressing the [**Local/GPIB**] hardkey twice also resets an Instrument Basic program. (The first press brings the instrument back to local and the second press resets the program.) A STOP statement in your program terminates the program but does not perform the reset operation.

Note While the program is executing an INPUT statement, pressing the [**Local/GPIB**] hardkey brings the program under local (front panel) control. This enables the front panel's alpha keys. Pressing [**Local/GPIB**] again, enters an "X" on the input line. In this case, press the [**Preset**] hardkey to abort the program.

Variables retain their value after an Instrument Basic RESET. Press [**BASIC**] [INSTRUMNT BASIC] [DEBUG] [EXAMINE VARIABLE] to examine variable values.

Pressing [**Preset**] stops a running program. It will also set the PRINTER IS device to the display (CRT).

For more information on the PAUSE and STOP statements see the " Instrument BASIC Language Reference" section of the Instrument Basic Users Handbook.

4

Saving and Recalling Programs

4

Saving and Recalling Programs

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

Transferring Programs

From the front panel you can transfer a program between memory and disk with the [**Save/Recall**] menus. Within a program, you can use the GET, SAVE, RE-SAVE, LOAD, STORE, and RE-STORE statements to transfer program files to and from disk. The Agilent 35670A has an autoload feature which automatically recalls and runs a program from disk at power-up.

You can transfer a program file between the analyzer and an external controller. You can keystroke record a measurement sequence and then upload the program to the external controller for further editing. Programs developed on an external controller can be downloaded as well. Chapter 8, "Interfacing with the GPIB," describes methods of transferring programs between the Agilent 35670A and an external controller.

This chapter describes transferring Instrument Basic programs between program memory and the Agilent 35670A's volatile, non-volatile, internal, and external disk drives. The autoload feature is described at the end of this chapter.

Disk Formats and File Systems

To successfully transfer an Instrument Basic program file, you must first understand the disk formats and file systems recognized by the Agilent 35670A.

Instrument Basic in the Agilent 35670A recognizes two disk formats: LIF (Logical Interchange Format), and DOS (Disk Operating System). Formatting or initializing a disk determines the format of a disk or file system.

A LIF disk contains only one directory. This format should be used to exchange programs and data with other BASIC computers.

A DOS disk has a hierarchical structure of directories. The DOS format should be used to exchange data with DOS computers.

The Instrument Basic Users Handbook refers to a third format, HFS (Hierarchical File System). The Agilent 35670A does not support HFS.

File Types

The Agilent 35670A supports three file types:

- ASCII
- BDAT
- untyped (referred to as DOS or HP-UX files)

"File type" is independent of disk format. ASCII, BDAT and untyped files exist on either LIF or DOS disks. Untyped files appear as HP-UX in the catalog of a LIF disk, or as DOS in the catalog of a DOS disk. To view the catalog of the analyzer's default disk, press:

[**Disk Utility**]
[CATALOG **ON** OFF]

In Instrument Basic, the "CREATE ASCII" command creates an ASCII file, the "CREATE BDAT" command creates a BDAT file, and the "CREATE" command creates an untyped file.

For more information, refer to "Disk I/O" in chapter 12 of this manual and the "Data Storage and Retrieval" chapter in the " Instrument BASIC Programming Techniques" section of the Instrument BASIC Users Handbook.

Note The [COPY FILE] operation in the [Disk Utility] menu does not translate file types when you copy files across different file systems (DOS/LIF). Verify you are using the appropriate file type before copying a file.

DOS Conventions

On DOS disks, file names must conform to DOS conventions. File names are limited to 8 characters followed by a period and a three character extension. The period and extension are not required. File names are not case sensitive. For example, the following file names are equivalent:

```
PROG.ASC = Prog.ASC
```

The Agilent 35670A does not allow "wild card" characters in file names. You can use a wild card in disk operations.

The Agilent 35670A recognizes a directory path. For example:

```
ASSIGN @File to "\DATA\TEST1\BEFORE"
```

opens the file named "BEFORE" in the sub-directory "TEST1" under the directory "DATA." Use a "\" or a "/" to separate directory and file names. The file specifier can include a directory path. You can create a DOS directory from the front panel with the [Disk Utility] [DEFAULT DISK] [CREATE DIRECTORY] key. You can create a DOS directory from a program with the CREATE DIR statement.

Using a DOS Disk to Transfer Data With a PC

You can transfer data from the Agilent 35670A to an IBM-compatible personal computer by writing an Instrument Basic program that outputs the data to a DOS disk.

To ensure a successful transfer, remember:

- Specify the correct disk format. Either format the disk on the PC, or use the Agilent 35670A and the correct format option with the [FORMAT DISK] operation in the [**Disk Utility**] menu. You can determine the format of a disk by looking at its catalog. Press [**Disk Utility**] [CATALOG **ON** OFF].

- Create untyped DOS files with the CREATE command. Untyped files on DOS disks are extensible. They "grow" to the size needed. ASCII and BDAT files are not extensible. They usually cannot be read by other DOS applications.

- Open files with the FORMAT ON option of the ASSIGN command. FORMAT ON directs Instrument Basic to store the data as ASCII characters.

You can also transfer Instrument Basic programs to a personal computer using a DOS disk. The GET statement recalls Instrument Basic programs from a DOS file into the analyzer's memory. The [RE-SAVE PROGRAM] softkey and the SAVE statement create untyped DOS files on a DOS disk.

For additional information about input and output operations, refer to "Front Panel Operations versus Keyword Statements" later in this chapter and "Disk I/O" in chapter 12, "Instrument-Specific Instrument Basic Features."

LIF Conventions

On LIF disks, file names must conform to LIF conventions. File names are limited to 10 characters which include all characters except ":", "<", and "|". Some LIF implementations do not allow lowercase letters.

LIF does not allow directories but you can label the disk with a volume name. The volume name is assigned at initialization.

Using a LIF Disk to Transfer Data with an BASIC computer

You can transfer data from the Agilent 35670A to an BASIC computer by writing an Instrument Basic program that outputs the data to a LIF disk.

To ensure a successful transfer, remember:

- Specify the correct disk format. Use the [FORMAT DISK] operation in the [**Disk Utility**] menu. You can determine the format of a disk by looking at its catalog. Press [**Disk Utility**] [CATALOG **ON** OFF].
- Use the CREATE ASCII command to create an ASCII file. A LIF protect code is not allowed on an ASCII file.
- Use the CREATE BDAT command to create a BDAT file. Instrument Basic allows and supports a LIF protect code on a BDAT file.
- Open files with the FORMAT option of the ASSIGN command. FORMAT ON directs Instrument Basic to store the data as ASCII characters. FORMAT OFF, which is faster and takes less space, defaults to BDAT representation.

You can also transfer Instrument Basic programs to an BASIC computer using a LIF disk. The GET statement recalls Instrument Basic programs from a LIF file into the analyzer's memory. The [RE-SAVE PROGRAM] softkey and the SAVE statement create ASCII files on a LIF disk.

For additional information about input and output operations, refer to "Front Panel Operations versus Keyword Statements" later in this chapter and "Disk I/O" in chapter 12, "Instrument-Specific Instrument Basic Features."

Program Buffers

The Agilent 35670A has five program buffers in memory. The BASIC menu accesses each program buffer with a set of softkeys initially labeled [PROGRAM 1] through [PROGRAM 5]. Each buffer can hold a separate program.

You can transfer programs between any disk and any program buffer but you cannot directly transfer programs between buffers. A program that is recalled into a buffer overwrites the previous contents of that buffer.

You can, however, append program files to build one functional program. See "Appending Program Files From Disk," later in this chapter.

Memory

Instrument Basic in the Agilent 35670A supports four mass storage devices:

- A volatile RAM disk (:MEMORY,0,0).
- A non-volatile RAM disk (:MEMORY,0,1).
- An internal disk drive (:INTERNAL).
- An external disk drive (Agilent Technologies Subset/80) (:EXTERNAL,7xx, uu).

To specify the default storage device, use the MASS STORAGE IS command.

Front Panel Operation versus Keyword Statements

There are two ways to transfer program contents between disk and memory. You can transfer programs by either:

- Using the [**Save / Recall**] key in the SYSTEM group on the front panel

or

- Using the keyword statements SAVE, RE-SAVE and GET

The choice of which to use requires some knowledge of the advantages of each as well as your own particular requirements. Both methods are discussed in the following section.

The [Save / Recall] Menu

With the [**Save/Recall**] menu you can perform a variety of disk and file operations, as well as transfer complete programs between any of the five program buffers and any disk file. These menus have the following advantage:

- You can transfer programs directly between any file on disk and any of the five program buffers.
- The analyzer allocates memory automatically when you recall a program.
- The utilities are similar to the other save and recall operations in the analyzer.
- You can select a file in the catalog without typing in the name.

The Keyword Statements (SAVE, RE-SAVE and GET)

In an Instrument Basic program, the keyword statements SAVE, RE-SAVE, and GET, save all or part of that program to disk. They also merge a program with a program from disk.

The SAVE, RESAVE, and GET keyword statements have the following advantages over the [**Save/Recall**] menu:

- You can store parts of a program to disk.
- You can recall programs and append them at any line in the currently active program.
- They are familiar to BASIC programmers.

Saving a Program to Disk

To save the current contents of the program buffer to a disk file, use the [**Save / Recall**] menus. This is the same system used for all disk access in the Agilent 35670A.

If you are saving a program to a new file name, press:

[**Save / Recall**]
[SAVE MORE]
[SAVE PROGRAM]

Type the name of the disk file in the entry window. You can use a keyboard or the front panel alpha keys. Instrument BASIC programs are stored as ASCII files on LIF disks and as untyped files on DOS disks.

Re-saving a program is similar to saving a file to a disk. In this case however, the disk already contains an existing file with the same name. The analyzer requires you to press an additional softkey, [OVERWRITE FILE], to confirm that you want to overwrite the existing file.

To make the re-save process easier, use the disk catalog to select a file name. The catalog describes the contents of the default disk. To use the catalog, press the Agilent 35670A keys as follows:

1. [**Save / Recall**]
2. [CATALOG **ON** OFF]
3. Use the knob to highlight the desired file name. The name appears in the entry window.
4. [SAVE MORE]
5. [SAVE PROGRAM]
6. [ENTER]
7. [OVERWRITE FILE]

The analyzer automatically re-saves the file with the file name you selected.

Recalling a Program from Disk

When you recall a program file from the disk, it is loaded into the active program buffer. Any program recalled to the program buffer using the [Save / Recall] menus overwrites the current contents of the active program buffer.

To recall a program file from the disk to the active program buffer, press the Agilent 35670A keys as follows:

1. [Save / Recall]
2. [RECALL MORE]
3. [RECALL PROGRAM]
4. Enter the file name in the entry window.
5. [ENTER]

As with any recall operation, you can use the catalog. Press the Agilent 35670A keys as follows:

1. [Save / Recall]
2. [CATALOG ON OFF]
3. Move the knob to select the file name. The name appears in the entry window.
4. [RECALL MORE]
5. [RECALL PROGRAM]
6. [ENTER]

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. The Instrument BASIC syntax error is displayed briefly in a pop-up message window. The error message is also written to the CRT. See chapter 5, Developing Programs, for information on allocating display partitions to view error messages.

Memory is allocated for the program's variables and working space (called the stack). When you use the [Save / Recall] menus to recall a program, memory is allocated automatically. For certain kinds of programs, the memory size may need to be increased.

See chapter 5 for more information on memory size.

Appending Program Files from Disk

To append program files from disk to the current program in memory, use the GET statement within a program. The GET statement recalls a specified file from the disk and appends it at a specified line in the current program (or at the beginning of the program if a line is not specified).

The following example program appends three program files to itself to build one functional program. It demonstrates how to merge files. It also provides a set of error-handing routines for your recorded programs.

The example program builds a shell composed of:

- an initialization program section.
- a typical keystroke recorded section.
- a cleanup section that contains error-traps and timeout-traps.

The core five line program (program lines 50 - 90) chains the other three programs segments to itself. These five program statements must be deleted or commented out before you run the program.

All of these files are on the Agilent 35670A Example Programs disk:

- SHELLBEG provides the setup and initialization.
- SHELLDEM is a typical keystroke recorded program.
- SHELLEND provides error-handling routines and cleanup.
- SHELLCHA pulls all files together using GET statements.

The file SHELLCHA contains the following program:

```
10      ! SHELLCHA: Program demonstrates chaining program segments
20      !———————————————————————————-
30      ! NOTE: Delete Lines 30 thru 90 immediately after
40      !       running SHELLCHAIN program
50      GET "SHELLBEG:,4",X,60
60      GET "SHELLDEM:,4",X,70
70      GET "SHELLEND:,4",X,80
80      DISP "Delete lines 30 - 90 before running program"
90      GOTO Endlabel
100     X:END
```

Line 50 performs a GET of the file “SHELLBEG” from the internal disk drive. It appends the file at line 100 (labeled “X:”) and overwrites that line. It then instructs the program to continue at line 60.

The “SHELLBEG” file has a label “X:” as its last line. The program in memory also has that same label as its last line. Line 60 performs a GET of the file “SHELLDEM,” appends it at the current label “X:” and then continues the program at line 70.

The SHELLDEM file also contains a label “X:” as its last line. Line 70 of the SHELLCHA program performs a GET, which appends the SHELLEND program file to the end of the program in memory.

Finally, line 90 of the SHELLCHA program skips to the label "Endlabel." This label, which was at the end of the SHELLEND file, is now at the end of the program in memory. The program would go on to execute the SHELLBEG program if the Endlabel line had been omitted. That is, without the Endlabel line, the program would run itself immediately after appending the three program files.

Note Remember to comment out or delete program lines 30-90 before running the combined program.

To use the SHELLCHA program with your own recorded program do the following:

1. Insert the label "X:" in the line containing the END statement of your recorded program. Make sure you have not used the label "X:" elsewhere in your program.
2. Recall the SHELLCHA program and change the file name in line 60 from "SHELLDEM" to the name of your program file.

Autoloading a Program

Instrument BASIC allows you to automatically load one or more programs and run a designated program when you turn on the analyzer. To make an autoloading program, save it to the nonvolatile RAM disk or to a floppy disk in the internal drive with one of the following names:

- AUTO_BAS
- AUTO_BA1
- AUTO_BA2
- AUTO_BA3
- AUTO_BA4
- AUTO_BA5

At power-up, Instrument BASIC searches the internal disk drive and then the nonvolatile RAM disk for files with these special names. It searches for files in the order listed above, but it does not search for AUTO_BA1 if AUTO_BAS is found. If AUTO_BAS is found, it is loaded into the first program buffer and executed after all other programs have been loaded. If AUTO_BA1 through AUTO_BA5 are found, they are loaded into the first through fifth program buffers—but they are not executed.

Since the AUTO_BAS program is run after all programs are loaded, you may find useful to have this program re-label the softkeys that run the loaded programs. The GPIB command "PROG:EXPL:LAB" is used to re-label these softkeys. For example, the following statement changes the first program buffer's softkey label to "FINAL TEST":

```
OUTPUT 800; "PROG:EXPL:LAB PROG1,'  FINAL  TEST'"
```

To disable automatic loading of the AUTO_BA* files, press the [**Preset**] hardkey while you turn on the analyzer.

5

Developing Programs

5

Developing Programs

Overview

For many applications, you can easily record and run programs without altering the program code that is generated with keystroke recording. However, with some knowledge of the Instrument BASIC language and the program development capabilities in the Agilent 35670A, you can add immeasurable power to your recorded programs. You can also create programs without using the keystroke recording feature.

This chapter describes the operation of the keys under the [INSTRUMNT BASIC] menu. See figure 5-1. At the end of the chapter the [DISPLAY SETUP] softkey (in the [BASIC] menu) is discussed. This softkey presents a menu that lets you manage a part of the screen display for output from the program.

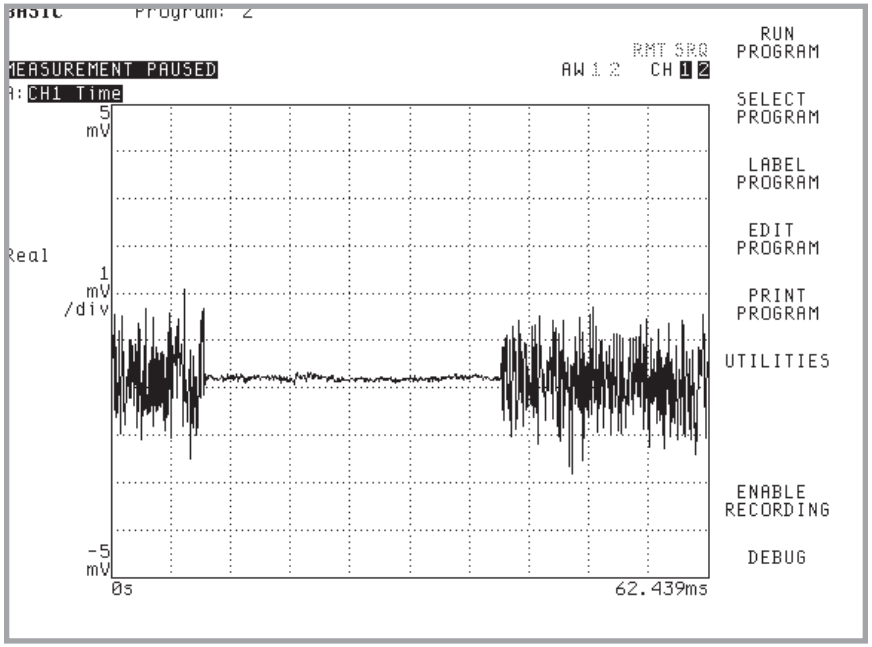

Figure 5-1. The [INSTRUMNT BASIC] Menu

The ability to change and enhance your program and its operating environment is found primarily under the [EDIT] and [UTILITIES] menus.

- Pressing [EDIT] places you in the Instrument Basic editor. You can make changes to your program on a line-by-line basis using a keyboard or the front panel alpha-numeric keys.

- Pressing [UTILITIES] presents a menu of helpful utilities. You can make global changes to the program and its environment. You can renumber lines, allocate memory size, and remove the program.

- Pressing the [PRINT PROGRAM] softkey prints a hard copy program listing to an attached printer.

Using the Instrument Basic editor

The Instrument BASIC editor allows you to create and alter program text. If you are familiar with the 9000 Series 300 BASIC editor, you will find it similar. If not, you should find the Instrument Basic editor easy to learn and to use. This section tells you how to enter and edit an Instrument BASIC program.

To start the editor, press the [EDIT] softkey in the [INSTRUMNT BASIC] menu. The program, if one exists, usually appears on the display with the cursor on the first line of the program. If the program buffer is empty, the first line number (10) appears with the cursor positioned to enter text.

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. The other lines display the first 51 characters (excluding line numbers).

The first 6 columns of each line are a numeric field specifying the program line number. Line numbers are right justified. Program lines are automatically numbered by the editor. You can manually edit the current line number to copy or move it a to different location in the program. Line numbers can also be renumbered in blocks with the [UTILITIES] [RENUMBER] softkey menu. Line number range from 1 to 32766.

Once in the [EDIT] menu you can use your keyboard or the front panel alpha-numeric keys.

Using the Instrument Basic Editor With a Keyboard

Using a keyboard makes developing Instrument Basic programs easy.

All of the “typewriter” keys are enabled. Letters can be entered in lower or upper case. All punctuation marks and special characters can be entered using the Agilent approved PC keyboard. See figure 5-2.

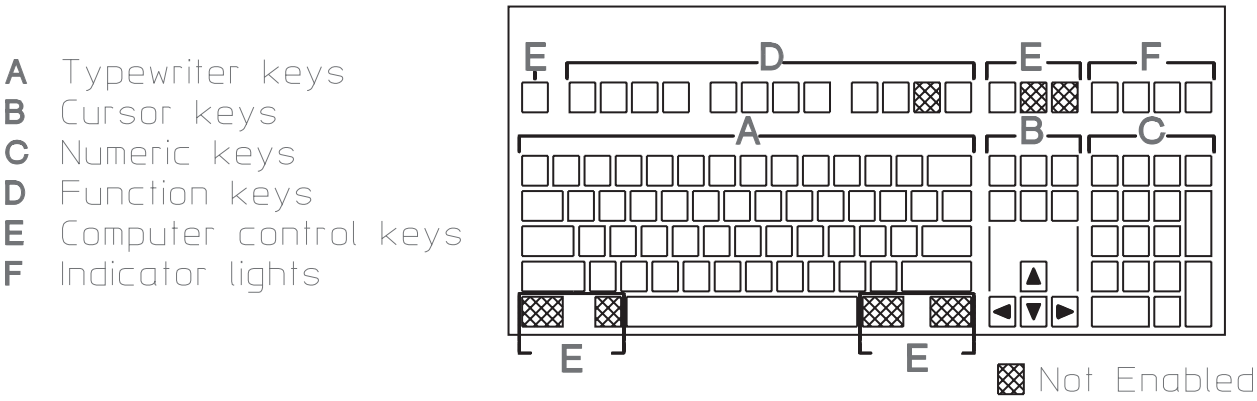

Figure 5-2. Using a keyboard with the Agilent 35670A.

The [*Enter*] key is used to store each line of program code and completes each alpha-numeric entry. The analyzer checks the line for syntax errors. If it detects an error, a pop-up message window displays the syntax error. If the analyzer does not detect an error, it stores the line.

Note If you edit or enter text on the current program line and then move off the line without pressing the [Enter] key, all editing on the line is lost.

The [*Tab*] inserts two spaces. Pressing [*Shift*] [*Tab*] moves the cursor backwards two spaces.

The Agilent 35670A softkey menus load into the keyboard function keys, [*F1*] through [*F9*]. The [***Rtn***] hardkey loads into [*F10*]. The [**Help**] hardkey loads into [*F12*]. [*F11*] is not enabled. See figure 5-2.

The “cursor” keys are enabled. The arrow keys indicate the direction in which they move the cursor.

The [*Home*] key moves the cursor to the beginning of the current line. The [*End*] key moves the cursor to the end of the current line.

The [*Page Up*] key moves the cursor a maximum of 15 lines upward. The [*Page Down*] key moves the cursor a maximum of 15 lines downward.

The [*Insert*] key inserts a new line of text. To get out of the insert mode, press the [*Insert*] key again or move the cursor off the current line. Remember, to save an edit you must press [*Enter*] while the cursor is on the current line.

The [*Delete*] key erases the character where the cursor is positioned. In addition, all characters to the right of the deleted character move one character to the left.

Pressing the [*Shift*] / [*Delete*] keys deletes the current program line.

Pressing the [*Alt*] / [*Delete*] keys ([*Alt Gr*] / [*Delete*] keys on a non-U.S. English keyboard) deletes all characters from the current cursor position to the end of the line.

The [*Print Screen*] key is enabled. You can print the entire screen (excluding the softkey menu text) to an attached printer.

The [*Alt*] key is not enabled except when used to preset the analyzer.

Key presses made with the keyboard that have no meaning in a given operating context are ignored, just as they are when pressed from the front panel.

Caution Pressing the [*Del*] key with the [*Alt*] key and the [*Ctrl*] key, presets the analyzer. (Just like a soft reboot in an IBM-compatible PC!)

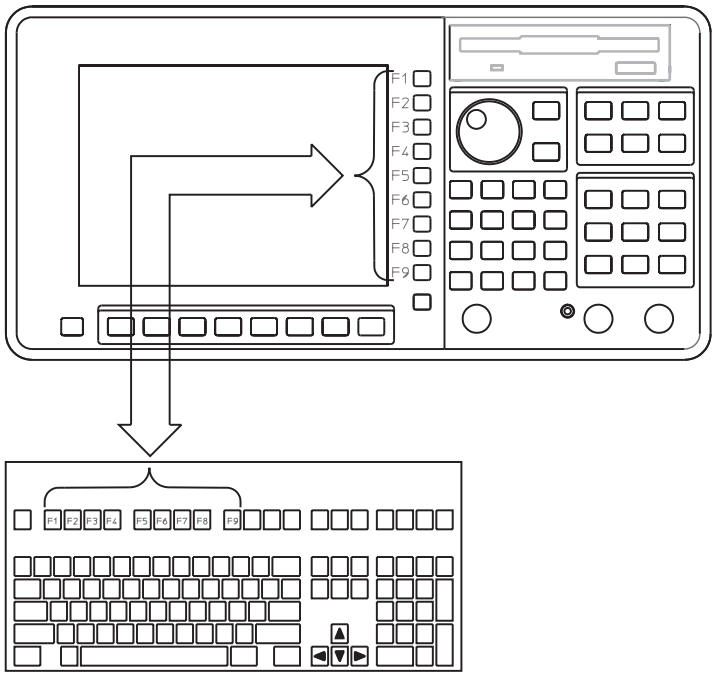

Figure 5-3. Mapping of Agilent 35670A Softkeys

To end an editing session, press the [*F9*] key, which corresponds to the [END EDIT] softkey. This returns you to the [INSTRUMNT BASIC] menu.

Connecting your keyboard

To connect the keyboard to the Agilent 35670A, plug the round connector into the analyzer's rear panel. See figure 5-4.

Caution Use only the Agilent approved keyboard on this product. Agilent does not warrant damage or performance loss caused by a non-Agilent approved keyboard.

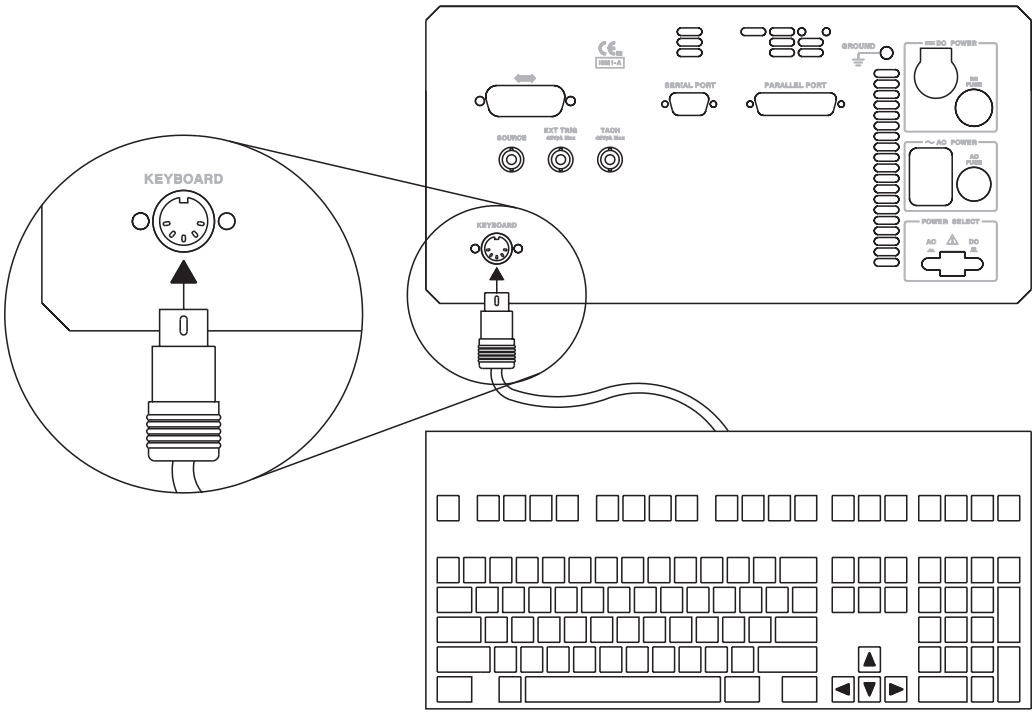

Figure 5-4. Connecting the Keyboard

If you are using an international keyboard, specify the type of keyboard with the [KEYBOARD SETUP] softkey in the [**System Utility**] menu.

Using the [EDIT] Softkeys

The [EDIT] menu contains the softkeys shown in figure 5-5.

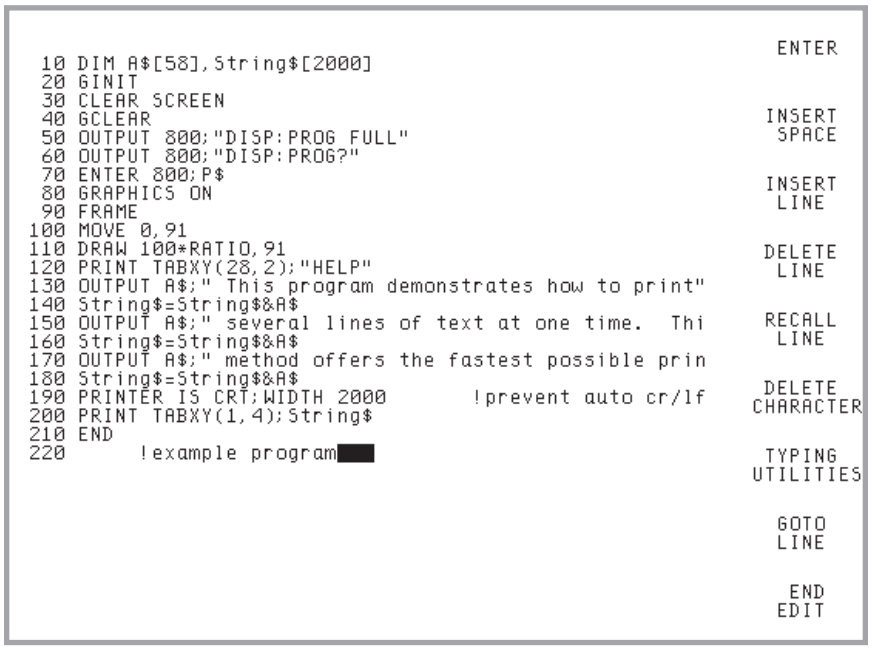

Figure 5-5. The [EDIT] Menu

You can move the cursor around in the program using the knob on the front panel or the cursor keys on the keyboard. When you get to a line you want to change, make the change and press the [ENTER] softkey or the [*Enter*] key on the keyboard. The analyzer checks the line for syntax and then stores it if the syntax is correct.

Getting Around in the Program

You can move the cursor from line to line within an existing program by:

- Using the knob
- Using the [GOTO LINE] softkey to jump directly to a specific line number or label
- Using the [ENTER] softkey (when not in insert mode) to step one line at a time
- Using the [*Enter*] key on the keyboard
- Using the cursor keys on the keyboard.

Using the Knob

You can move the cursor in the EDIT mode with the knob on the front panel.

The line that the cursor is on is always the edited line. Rotating the knob clockwise on the currently edited line moves the cursor to the right. Rotating the knob counterclockwise moves the cursor to the left. When the cursor is at the end (far right) of the edited line, turning the knob clockwise moves the cursor down to the end of the next line. Conversely, when the cursor is at the beginning (far left) of the edited line, turning the knob counterclockwise moves the cursor to the beginning of the preceding line.

Using GOTO LINE

To jump immediately to any line or label in the program press the [GOTO LINE] softkey ([F8] on the keyboard). Enter the line number or the label of the line into the entry window and press [ENTER]. To specify a label, use the keyboard or use the front-panel keys and the optional [(_)] underscore softkey. You can enter the label in capital letters and it automatically converts to the proper case.

If the specified line exists, it appears in the middle of the display as the current program line. If you have specified a line number that doesn't exist, the cursor is placed on the line number closest to it. Specifying a non-existent line label generates an error message, "Line not found in this context."

A quick way to go to the last line of the program is to enter a number much larger than the largest possible program line number such as 99999 (or any number greater than 32766 or the last line number of your program).

Using the [ENTER] Softkey

You can use the [ENTER] softkey to move the cursor down one line at a time. All other softkeys that move the cursor alter program text.

Using the Keyboard

See the previous section, Using the Instrument BASIC Editor With a Keyboard, for a description of getting around the program using the keyboard.

Entering Program Lines

When you finish entering or changing a program line, store it by pressing [ENTER]. The analyzer checks the line for syntax errors and converts letter case to the required form for names and keywords (Instrument BASIC commands). If it detects an error, a pop-up message window displays the syntax error. If no errors are detected, it 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 is lost.

Renumbering, Copying and Moving Lines

If you want to change the line number of an edited program line, move the cursor to the line number field and enter a new line number. Changing the line number copies the line. The line does not move. To move the line, change the line number, press [ENTER] and then delete the original line.

If you want to revise and move the current line, edit the line, change the line number and then press [ENTER]. The revision only appears in the copied line.

If you change the line number and you are in insert mode, you remain in insert mode at the new line number.

When the cursor is in the line number field, entries operate in an overtype mode rather than in the insert mode as in the text portion of the program line. The [Back Space] hardkey in the numeric keypad moves the cursor over line numbers without deleting the number.

Inserting Spaces

Use the [INSERT SPACE] softkey to place a space at the position of the cursor. The text to the right of the cursor moves one place to the right. This softkey is located in more than one menu.

To insert a space with your keyboard, press the space bar.

Inserting Lines

You can easily insert one or more program lines above any existing line by placing the cursor on the existing line and pressing [INSERT LINE]. The [INSERT LINE] softkey toggles the insert mode on or off.

If you are using a keyboard, press the [*Insert*] key to insert a line. Pressing the function key [*F3*], also turns the insert mode on or off.

In the following example we use the [INSERT LINE] softkey to insert lines between two adjacent programs lines numbered 90 and 100.

Move the cursor to line 100 and press [INSERT LINE]. A new line, numbered 91, appears between line 90 and line 100. After you type something on the inserted line, press [Enter] to store it and another line, numbered 92, appears. If you continue to insert new lines and the inserted line number increments to 100, the current line 100 is renumbered to 101 to accommodate the inserted line.

To get out of insert mode, press [INSERT LINE] again or use the knob to move off of the current line. (Remember, any edits you make to the currently inserted line are lost if you leave insert mode without pressing [ENTER].) Make sure you have entered any changes to your final inserted line before exiting the insert mode.

Recalling Deleted Lines

If you used the [DELETE LINE] softkey or the [*Shift*] [*Delete*] keys to remove a line, the [RECALL LINE] softkey automatically recalls that line. This is useful for recovering lines deletedby mistake.

It is also useful for moving a line. Delete the line, move to the desired area of the program and press [INSERT LINE] ([*F3*] on the keyboard). Press [RECALL LINE] ([*F5*] on the keyboard) and then edit the recalled line to the current line number.

Note Pressing [RECALL LINE] automatically aborts any changes made to the currently edited line.

Using the Front-Panel Alpha Keys

If you do not have a keyboard, you can use the alpha keys on the front panel of the Agilent 35670A. Nearly every hardkey is labeled with a corresponding letter of the alphabet. These may be familiar to you if you have performed any editing function on the Agilent 35670A, such as specifying a unique filename in a [**Save/Recall**] operation.

The alpha keys are arranged in alphabetical order from left to right, descending the front-panel hardkeys. See figure 5-6.

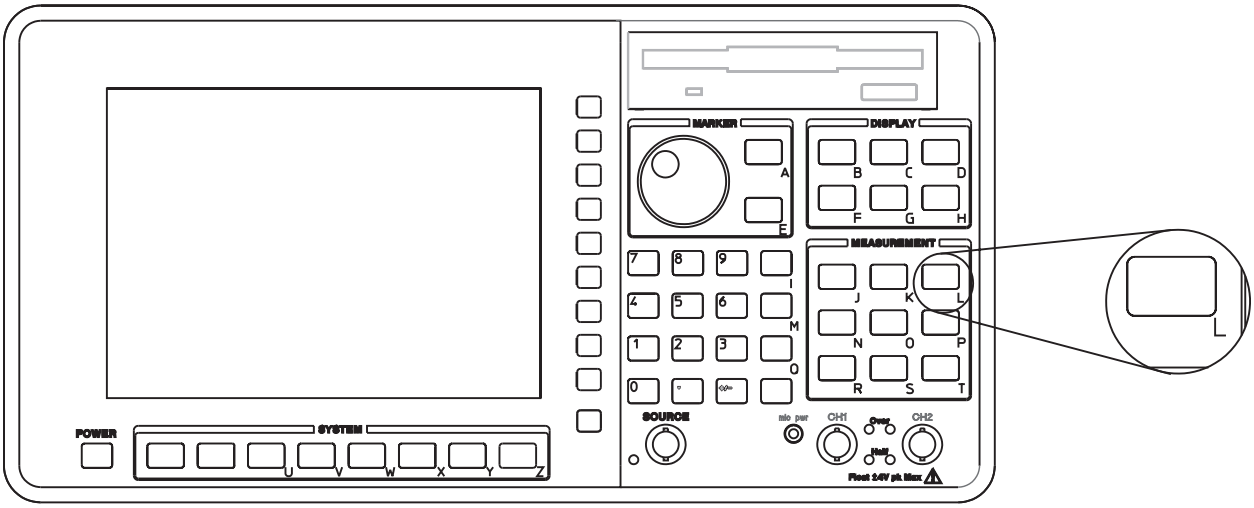

Figure 5-6. Agilent 35670A front panel alpha-numeric keys

You do not have to use the alpha keys to enter Instrument BASIC keywords. They can be entered via the [TYPING UTILITIES] menu. The front panel alpha keys are necessary to enter variable names, constants, labels and strings if you are not using a keyboard.

When in the Instrument Basic editor, the front-panel keys are automatically in alpha mode. Pressing an alpha key enters the character at the cursor position in the current program line. When you exit the editor ([END EDIT]) the front-panel keys return to their labeled hardkey function.

Changing Case

The case of an alpha key is determined by the state of the [TYPING UTILITIES] [**UPPERCASE** lowercase] key. The default is uppercase. To enter lowercase letters, press the [UPPERCASE **lowercase**] key.

If you are using the keyboard, the case is determined by the [Caps Lock] key.

Using [TYPING UTILITIES]

The [TYPING UTILITIES] allows you to enter non-alphabetic symbols and insert Instrument BASIC keywords without a keyboard. See figure 5-7.

Figure 5-7. The [TYPING UTILITIES] Menu

The [ENTER], [INSERT SPACE], and [DELETE CHARACTER] keys are carried over from the [EDIT] menu. In addition, this menu contains the [**UPPERCASE** lowercase] key and the [INSERT KEYWORD] key.

Entering Symbols

Symbols are available in four menus. Each menu is labeled "INSERT" followed by a list of the available symbols in that menu.

For example, to enter an equal symbol (=), press [TYPING UTILITIES], then press [INSERT +-*^/=( )]. This brings up a softkey menu with each of the symbols listed in the label as a separate softkey. Press the softkey labeled [=].

Entering Keywords

You do not have to type an entire Instrument Basic keyword if you use the [INSERT KEYWORD] softkey in the [TYPING UTILITIES] menu. When you are in the [INSERT KEYWORD] menu, pressing any alpha key presents a menu of keywords beginning with that letter.

For example, pressing [INSERT KEYWORD] and then the alpha key "F" (the [**Active Trace**] hardkey) presents a menu with the following softkeys:

- [FN]
- [FNEND]
- [FOR]
- [FORMAT]
- [FRACT( ]
- [CANCEL]

Pressing any one of these softkeys, other than [CANCEL] enters the corresponding text into the current program line. Keywords are always inserted in uppercase regardless of the current setting of the [**UPPERCASE** lowercase] softkey.

In cases where there are more than eight keywords starting with a particular letter, a softkey labeled [MORE] appears which allows you to access the rest of the keywords of that letter. When the last set of keywords is displayed, press [MORE] to get back to the first set. This allows you to cycle through all the keywords starting with a specific letter.

After pressing [INSERT KEYWORD] you can skip from one keyword menu to another simply by pressing another front-panel alpha entry key.

Notice that all keywords that require an argument are provided with the beginning parenthesis; for example, [FRACT( ]. The parenthesis indicates the keyword as requiring an argument. When this keyword is selected, the keyword and both parentheses are inserted in the program line with the cursor placed automatically between the two. All keywords that require an argument are inserted this way.

To return to the previous menu without selecting a keyword, press [CANCEL].

Recording into an Existing Program

Another way to enter lines into your program is to use the keystroke recording capabilities of Instrument BASIC. To record measurement sequence operations into your program, move the cursor to the line where you want the recorded statements inserted. Then press [END EDIT], press [ENABLE RECORDING] and proceed with your recording as you normally would. Press [**BASIC**] to conclude the recording session as usual.

The inserted recording acts the same as if you had pressed [INSERT LINE] in the editor, and generates OUTPUT statements in insert mode.

The "ASSIGN @Agilent35670a to 800" statement is not generated when you are recording into an existing program. The "ASSIGN @Agilent35670a to 800" statement must be included in your program prior to any recorded OUTPUT commands. If you initially created the program using keystroke recording, this statement should already exist. If it does not exist, you will need to enter it.

Removing Program Text

The Instrument Basic editor allows you to remove individual characters or entire lines. To learn how to remove the entire program see the description of the [UTILITY] [SCRATCH] softkey later in this chapter.

Deleting Characters using a keyboard

The [*Delete*] key on the keyboard erases the character where the cursor is positioned. In addition, all characters to the right of the deleted character move one character to the left.

The [*Backspace*] key also removes text. The cursor moves one space to the left and usually erases any characters in the cursor's path. It does not erase characters in the program line number field.

Pressing the [*Alt*] / [*Delete*] keys ([*Alt Gr*] / [*Delete*] keys on a non-U.S. English keyboard) deletes all characters from the current cursor position to the end of the line.

Deleting Characters using the [DELETE CHARACTER] softkey

The [DELETE CHARACTER] softkey, [F6] on the keyboard, removes the character under the cursor and moves all characters to the left one place. Repeatedly pressing [DELETE CHARACTER] causes text to the right of the cursor to be pulled in and deleted. The [DELETE CHARACTER] softkey functions the same in both the line number and program statement fields. However, in the line number field, only line numbers to the right of the cursor are pulled in and deleted. Program statement characters are not deleted when the cursor is in the line number field.

Another way to remove text on a line is with the [**Back Space**] key in the front panel's numeric key pad. Pressing [**Back Space**] 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. When the cursor is on a line number, pressing the [**Back Space**] key simply moves the cursor back one position without deleting the number.

Deleting Lines using a keyboard

Pressing the [*Shift*] [*Delete*] keys removes the current program line and places it in a buffer. 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 the [*Shift*] [*Delete*] keys are 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. This puts the editor in insert mode on the last line of the program (see "Inserting Lines"). To get out of insert mode, use the knob and move the cursor up one line.

Pressing the [*Shift*] [*Delete*] keys will not remove a subprogram line with a SUB keyword in it unless all program lines belonging to that subprogram are deleted first.

Deleting Lines using the [DELETE LINE] softkey

The [DELETE LINE] softkey, [*F4*] on the keyboard, removes the current program line in the same manner as pressing the [*Shift*] [*Delete*] keys on the keyboard.

To recall the last deleted line, press the [RECALL LINE] softkey, [*F5*] on the keyboard.

Using [UTILITIES]

There are some activities generally associated with editing that are located outside the [EDIT] menu, under the [INSTRUMNT BASIC] [UTILITIES] softkey. These editing utilities are more global in nature, rather than pertaining to single characters, words and lines as the editor does. See figure 5-8.

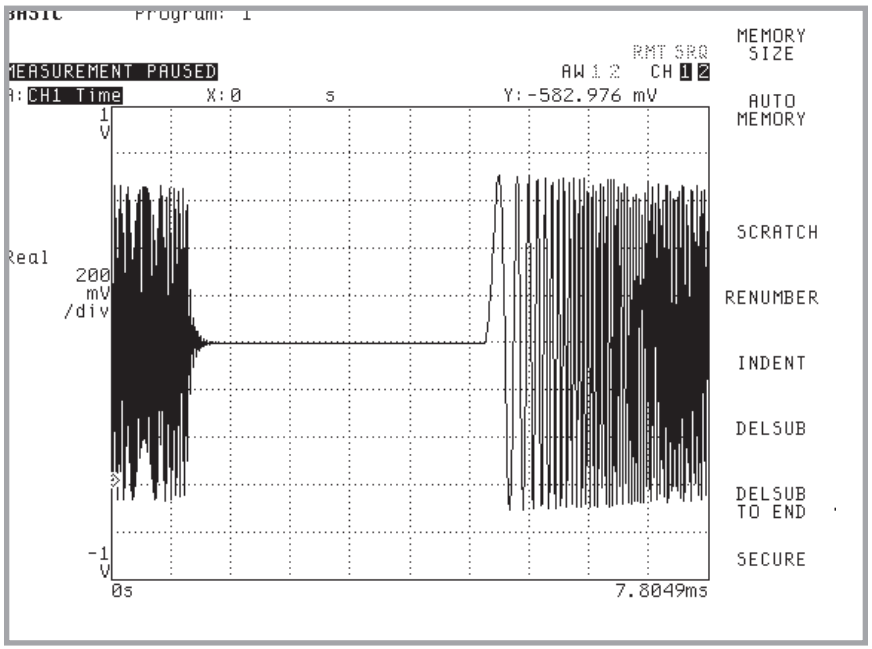

Figure 5-8. The [UTILITIES] Menu

If using a keyboard, the [UTILITIES] menu loads into the following function keys:

[MEMORY SIZE]	[F1]
[AUTO MEMORY]	[F2]
[SCRATCH]	[F4]
[RENUMBER]	[F5]
[SECURE]	[F6]
[INDENT]	[F7]
[DELSUB]	[F8]
[DELSUB TO END]	[F9]

The [UTILITIES] menu is mostly composed of Instrument BASIC keywords that can be executed interactively. All but [MEMORY SIZE] and [AUTO MEMORY] directly affect the contents of the program.Two other utility softkeys are not keywords: [MEMORY SIZE] and [AUTO MEMORY]. These utilities allow you to directly change the program's operating space.

MEMORY SIZE

Press [MEMORY SIZE] to display the amount of working space (commonly called the stack) currently allocated for the active program. The stack contains all variables not in COM as well as context information for functions and subprogram calls. The stack does not contain program code.

Instrument Basic allocates the size of the stack for the most efficient use of memory resources. If you use recursive subprograms, Instrument Basic may not allocate enough memory. If the analyzer runs out of stack space while the program is running it displays an error message, "Out of Memory" in a pop-up message window.

To increase the amount of memory allocated for the stack, press [MEMORY SIZE]. Enter the new amount using the numeric keys on the front panel or on the keyboard. The entry window is displayed when you press the first numeric key. Use the [EXP] softkey to enter the size using engineering notation. After entering the new memory size, press [ENTER]. Instrument Basic may adjust your entry to the closest available increment of memory.

If you enter a number which exceeds the available memory, the memory size will be set to the largest available stack size. The minimum amount allocated by Instrument Basic for the stack is 1122 bytes.

Memory available for Instrument Basic programs is dependent upon the amount of memory space allocated for other uses. To display the usage of all of the analyzer's memory, press the [**System Utility**] hardkey then press [MEMORY USAGE]. A table displays the amount of memory allocated for all Instrument Basic programs (code and program stacks) as well as memory allocated for other functions of the analyzer.

AUTOMEMORY

The [AUTO MEMORY] softkey resizes stack space automatically to fit the current active program. This is similar to the operation that occurs when a program is loaded with the [**Save / Recall**] menus. This is faster than using the [MEMORY SIZE] key and works well for most programs.

In some cases, [AUTO MEMORY] may allocate more memory than the Instrument Basic program needs. Use the [MEMORY SIZE] softkey to reduce the amount of memory allocated for your program. If you receive an "Out of Memory" error when you try to run the program you can use the [MEMORY SIZE] softkey to increase the memory size. Programs that use recursive functions or subprograms may need to have memory increased manually with the [MEMORY SIZE] softkey.

If you want [AUTO MEMORY] to allocate more memory for a particular program, you can append the following "dummy" subprogram:

```
.
.
100 SUB More_memory
110  DIM A(1000)
120  SUBEND
```

Increasing the size of array A causes more memory to be allocated.

SCRATCH

Pressing the [SCRATCH] softkey brings up a menu that allows you to clear the current program and/or variables. The softkeys load into the keyboard function keys as follows:

[SCRATCH]	[F1]
[SCRATCH C]	[F2]
[SCRATCH A]	[F3]
[PERFORM SCRATCH]	[F5]

You must first select a combination of program and/or variables to clear by pressing [SCRATCH], [SCRATCH C], or [SCRATCH A]. The scratch operation is not executed, however, until you press the [PERFORM SCRATCH] softkey.

SCRATCH

This key selects the current active Instrument BASIC program and all variables not in COM.

SCRATCH C

This key selects all variables including those in COM, but does not clear the program.

SCRATCH A

This softkey selects the current active Instrument BASIC program and all variables, including those in COM.

The analyzer does not clear the memory until you press [PERFORM SCRATCH]. To cancel a SCRATCH operation, press [**Rtn**] at any time prior to pressing [PERFORM SCRATCH].

RENUMBER

Pressing [RENUMBER] displays a menu that allows you to change the line numbering for the entire active program. The [RENUMBER] menu loads into the keyboard function keys as follows:

[START LINE #]	[F1]
[INCREMENT]	[F2]
[PERFORM RENUMBER]	[F4]

To select the number that is assigned to the first line in the program when renumbering lines in a program, press [START LINE #]. If you do not define the starting line number, the first line number defaults to 10.

Press [INCREMENT] to specify the increment between the renumbered line numbers. The default is 10. For example, if [START LINE #] is 10 and [INCREMENT] is 5, the line numbers will be 10 . . . 15 . . . 20 . . . 25 . . . and so on.

Once these parameters are defined, press [PERFORM RENUMBER] to execute the command. To cancel the renumbering operation, press [**Rtn**] at any time prior to pressing [PERFORM RENUMBER].

SECURE

The [SECURE] menu allows you to "protect" program lines. "Protected" program lines cannot be listed to a printer or viewed in EDIT mode. The [SECURE] softkeys load into the keyboard function keys as follows:

[START LINE #]	[F1]
[END LINE #]	[F2]
[PERFORM SECURE]	[F4]

To secure a block of the active program:

1. Press [START LINE #].
2. Enter the beginning line number of the program block. (The value defaults to 1.)
3. Press [END LINE #].
4. Enter the ending line number of the program block. (The value defaults to 32766.)
5. Press [PERFORM SECURE].

Since [START LINE #] value defaults to 1 and the [END LINE #] value defaults to 32766, you can secure the entire program by pressing [PERFORM SECURE] without altering the start line and end line values.

Secured lines cannot be printed or viewed in the editor. They appear only as an asterisk following the line number (*). Secured lines can, however, be deleted from the program using the editor. You may leave this menu at any time by pressing [**Rtn**].

Caution Secured program lines cannot be unsecured. Be sure to keep an unsecured version of the program for your own records.

INDENT

Pressing [INDENT] displays a menu that allows you to change the indentation for the entire active program. The [INDENT] menu loads into the keyboard function keys as follows:

[START COLUMN #]	[F1]
[INCREMENT]	[F2]
[PERFORM INDENT]	[F4]

Press [START COLUMN #] to specify the column in which the first character of the first statement should appear. Press [INCREMENT] to specify the number of spaces each line should move to the right or left when the nesting level of the program changes.

Once these parameters are defined, press [PERFORM INDENT] to execute the command. To cancel the renumbering operation, press [**Rtn**] at any time prior to pressing [PERFORM INDENT].

DELSUB and DELSUB TO END

[DELSUB] and [DELSUB TO END] allow you to delete subprograms and functions from your program. When you press [DELSUB], you are prompted to enter the name of the single subprogram or function you want to delete. Once you have typed the name in the prompt, press [ENTER] to delete the subprogram or function, or press [**Rtn**] to cancel the operation.

When you press [DELSUB TO END], you are prompted to enter the name of the first subprogram or function you want to delete. Once you have typed the name in the prompt, press [ENTER] to delete the subprogram or function—and all subprograms or functions that follow it—or press [**Rtn**] to cancel the operation.

Using [PRINT PROGRAM]

The [PRINT PROGRAM] softkey allows you to print the current contents of the active program buffer to an attached printer. However, you must first configure your printer under the [**Plot/Print**] hardkey. See online help for more information about configuring your printer.

Note If you press [PRINT PROGRAM] and do not have a printer connected or properly configured, Instrument BASIC continues attempting to print until you press either [**Local/GPIB**] or [**Preset**].

Once the printing operation is completed, the PRINTER IS device is set to the display (CRT).

Using [DISPLAY SETUP]

Pressing the [**BASIC**] [DISPLAY SETUP] key allows you to allocate a partition of the analyzer's display to be used by your program. Alternately, Instrument Basic can return any allocated partition of the display to the analyzer.

The Agilent 35670A display is divided into two small partition areas (UPPER, and LOWER) and one large area (FULL), which encompasses both the UPPER and LOWER partition areas.

See figure 5-9.

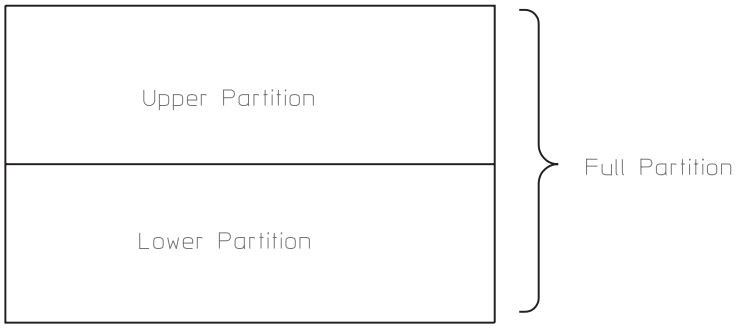

Figure 5-9. The Display Partitions

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 [DISPLAY SETUP] softkey.

The [DISPLAY SETUP] menu softkeys load into the following keyboard function keys:

[OFF]	[F1]
[FULL]	[F2]
[UPPER]	[F3]
[LOWER]	[F4]
[CLEAR SCREEN]	[F6]
[ALPHA ON OFF]	[F7]
[GRAPHICS ON OFF]	[F8]

You can allocate display partitions from within your program using the GPIB command “DISP:PROG” and specifying the parameter UPPer, LOWer or FULL. For example the statement

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

allocates the single trace box of the display. This command corresponds to selecting [FULL] from the [DISPLAY SETUP] menu. Table 5-1 shows the relationship between the [DISPLAY SETUP] softkeys and the corresponding GPIB commands required to program the same functions.

Table 5-1. The Display Partitions

MENU	ALLOCATES	GPIB Command
OFF	NO DISPLAY	DISP:PROG OFF
FULL	SINGLE TRACE AREA	DISP:PROG FULL
UPPER	UPPER TRACE AREA	DISP:PROG UPP
LOWER	LOWER TRACE AREA	DISP:PROG LOW

Most display allocation should be handled by your program with the GPIB commands. It is best to use these softkeys during program development.

[CLEAR SCREEN] clears all text and graphics from the active partition. [ALPHA ON OFF] enables and disables the display of alpha output in the active partition. [GRAPHICS ON OFF] enables and disables the display of graphics output in the active partition.

For more information about controlling the display, refer to chapter 7, “Graphics and Display Techniques.”

6

Debugging Programs

6

Debugging Programs

The process of creating programs usually involves correcting errors. You can minimize these errors by using keystroke recording for your measurement sequence program segments and by writing structured, well-designed programs.

Of course bugs can and do appear in even the best designed programs. Instrument BASIC contains some useful features to help you track them down.

Overview

The Instrument BASIC tools provided for program debugging are simple and, if used properly, can be very helpful. The [INSTRUMNT BASIC] menu contains the [DEBUG] softkey. See figure 6-1.

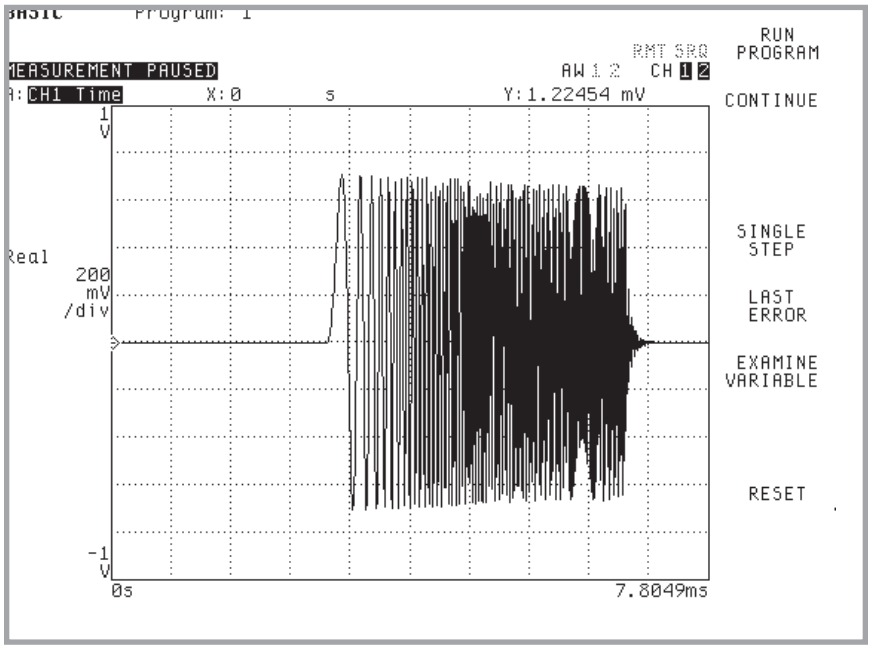

Figure 6-1. The December 7, 342 Menu

If using a keyboard, the [DEBUG] menu loads into the function keys as follows:

[RUN]	[F1]
[CONTINUE]	[F2]
[SINGLE STEP]	[F4]
[LAST ERROR]	[F5]
[EXAMINE VARIABLE]	[F6]
[RESET]	[F8]

The [DEBUG] menu provides several debugging facilities. For example, using the [DEBUG] menu you can:

- RUN or CONTINUE your program normally
- SINGLE STEP through your program 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.

Use the [SINGLE STEP] softkey to execute the program one line at a time. You can study the program's operation and examine variable values.

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. Press [CONTINUE] to resume operation to the next PAUSE statement or to the end of the program. Press [SINGLE STEP] to walk through program lines following the PAUSE statement.

By combining these different features you can examine the program's operation and solve your particular problem.

Using [EXAMINE VARIABLE]

Pressing [EXAMINE VARIABLE] displays an entry window that allows you to enter the name of the variable you want to examine. The default is the name of the last examined variable. It also brings up the editor (the alpha entry menu), so you can enter the variable name.

You must first perform a prerun operation to examine the value assigned to any variable in your program. A prerun is executed when you press either [RUN] or [SINGLE STEP]. After the prerun, press the [EXAMINE VARIABLE] key and enter the name of an existing variable in your program.

You can enter the variable as all uppercase letters. When you are finished entering a variable name, press [ENTER].

If you use [SINGLE STEP] and the program has not executed the line assigning that variable, the variable returns a value of zero.

Examining Strings

Enter string variables as you would any other variable. The entry window wraps to display a maximum of 10 lines of 42 characters each.

To select only a section of a string, use the Instrument BASIC substring syntax (see the " Instrument BASIC Programming Techniques" section in the Instrument Basic Users Handbook). For example, to examine the 7 character substring starting at the second character of A$ enter:

```
A$[2;7]
```

Examining Arrays

You can examine an entire array or individual elements of the array. For example the entry:

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

displays the elements 1 through 3 of the array I_array.

To select an entire array for examination enter the array variable name followed by an asterisk, (*); for example, I_array(*).

Example

I_array(20) is an integer array. The first and second elements are set to 100. After pressing [EXAMINE VARIABLES], enter "I_array(*)." The following is displayed:

```
I_array(*)  =  100  100  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0
```

An individual array element (for example, I_array(17)), is specified the same as any single variable.

Setting Breakpoints

A common method of debugging a program is the use of breakpoints. A breakpoint causes the program to stop at a defined point so that you can examine the program state at that point. In Instrument BASIC this is accomplished by inserting PAUSE statements in the program code. When the program runs, you can use [EXAMINE VARIABLE] to check or change variable values. Press [CONTINUE] to continue the program until the next PAUSE, STOP or END statement is encountered.

You can enter PAUSE statements and otherwise alter the contents of the active program by using the [**BASIC**] [EDIT] softkey. See chapter 5, “Developing Programs,” for a description of the Instrument BASIC editing capabilities in the Agilent 35670A.

Using [SINGLE STEP]

The [SINGLE STEP] softkey allows you to execute your program one line at a time. The line to be executed appears in the first line of the display. See figure 6-2.

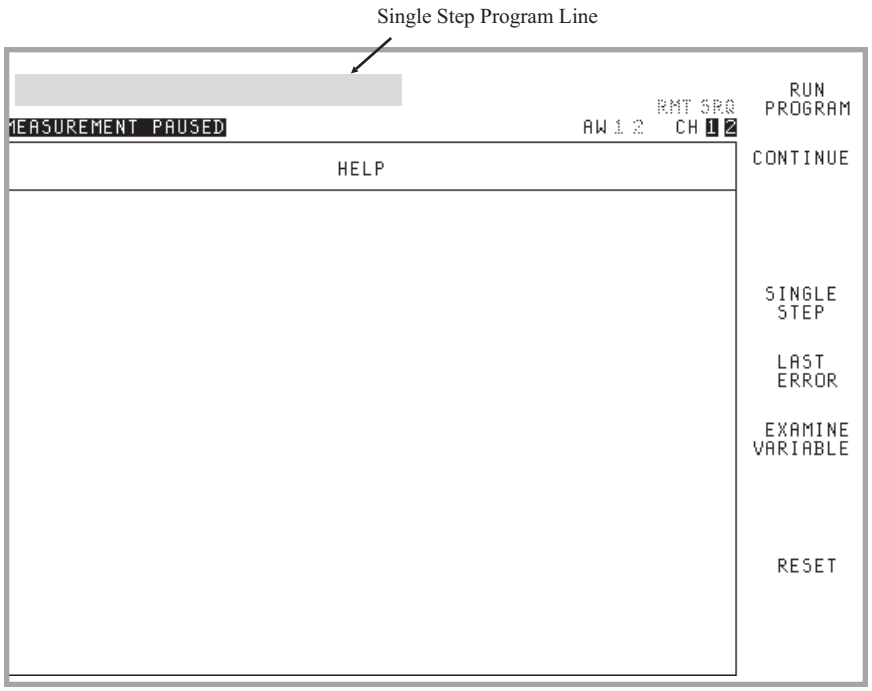

Figure 6-2. Single Step

You can use [SINGLE STEP] from the beginning of the program or from any point where it has been paused. To resume regular execution of a program after using [SINGLE STEP], press [CONTINUE].

[SINGLE STEP] can be very helpful when used in conjunction with the [EXAMINE VARIABLES] key and the PAUSE statement. By placing a PAUSE statement at a point of interest in your program, you can run the program until it pauses, then single step through the critical program lines, checking variables values or program operation. To resume program execution at any time, press [CONTINUE].

Using [RUN PROGRAM], [CONTINUE], and [LAST ERROR]

The [RUN PROGRAM] softkey executes a prerun sequence and then begins executing the program at the first program line. Execution continues until the analyzer reaches a PAUSE, STOP or END statement, or until the program is paused or stopped from the front pinanel.

The [CONTINUE] softkey allows you to resume regular program operation from a paused program or from a program in single step mode. This is identical to the [**BASIC**] [CONTINUE] softkey operation.

The [LAST ERROR] softkey displays the error number and message of the last error encountered by the program. This is the front panel equivalent to the BASIC command, ERRM$.

Using [RESET]

The [RESET] softkey allows you to bring the program environment back to its default state. This is especially useful when you are using single step mode and you want to restart the program. Pressing [RESET] sends an abort message to the GPIB interface, resets the program counter to the first program line and closes all open files with one exception. Pressing [RESET] does *not* close a file if it is the device for a PRINTER IS statement.

7

Graphics and Display Techniques

7

Graphics and Display Techniques

Instrument BASIC programs have the ability to allocate portions of the instrument's display for text and graphics. This section provides a description of the various programming techniques used to do both.

Using the Partitions

There are several Instrument BASIC commands that require a display as an output device. These include commands such as PRINT, CLEAR SCREEN, MOVE, DRAW and GCLEAR. Since Instrument BASIC programs share all hardware resources with the instrument, the display must be shared for instrument and program use. All commands that output data to the screen write to a screen buffer and in order to view this output buffer, a portion of the display must be released from the instrument. You can do this manually when the program is not running by using the [**BASIC**] [DISPLAY SETUP] softkey menu. Performing equivalent actions from within a program that is running, requires sending an GPIB message to the instrument; both to borrow a screen partition and again to give it back.

Allocating Partitions

The instrument's screen can be divided into two trace boxes (upper and lower). The upper and lower trace boxes can be combined into one large trace box for single trace displays. Any of these three trace boxes, called display partitions, can be used by an Instrument BASIC program.

There are two other non-partition areas of the screen that can be accessed by Instrument BASIC programs. The area on the right of the screen is reserved for softkey labels and can be accessed using the ON KEY statement. Also, a line at the top of the screen can be accessed via the DISP and INPUT statements.

To request one of the partitions from the analyzer, send the instrument the corresponding GPIB command. “DISP:PROG UPP” allocates the upper partition, “DISP:PROG LOW” allocates the lower partition, and “DISP:PROG FULL” allocates the full screen partition. See figure 7-1.

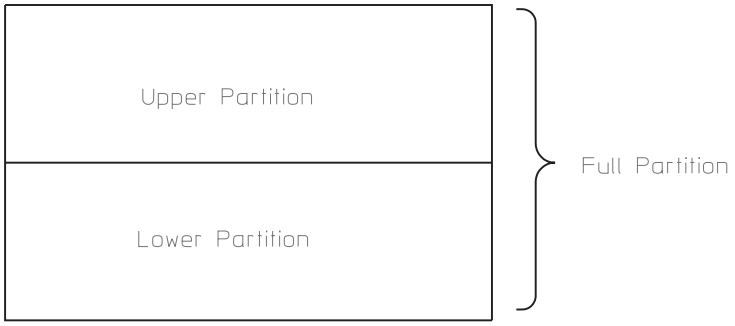

Figure 7-1. The Display Partitions

The following is an example of a program segment that prints a message to the upper trace box:

```
30  ASSIGN @Agilent35670a TO 800
40  OUTPUT @Agilent35670a;"DISP:PROG UPPER"
50  CLEAR SCREEN
60  PRINT "This is the upper partition"
```

To be sure that you are writing to a partition that has not been assigned, include a WAIT statement. Or, add an GPIB query command followed by an ENTER statement to synchronize the program with the instrument. The previous example would look like this:

```
30  ASSIGN @Agilent35670a TO 800
40  OUTPUT @Agilent35670a;"DISP:PROG UPPER"
50  OUTPUT @Agilent35670a;"DISP:PROG?"
60  ENTER  @Agilent35670a;Part$
70  CLEAR SCREEN
80  PRINT "This is the upper partition"
```

The command DISP:PROG? (line 50 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, use the “DISP:PROG OFF” command. 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 someone to view instrument trace data. The following example demonstrates this command:

```
830  OUTPUT @Agilent35670a;"DISP:PROG OFF"
```

Using Text

You can enable the display of text information by pressing [**BASIC**] [DISPLAY SETUP] [ALPHA **ON** OFF]. This information is generated primarily by the Instrument BASIC PRINT statement.

The PRINT statement works the same in every partition. Information is printed starting at the top of the current partition and continues until the bottom of the partition is reached where the screen then scrolls up to allow additional lines to be printed. Causing the screen to scroll does not effect any graphics displayed on the screen, because 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 each contain 14 lines. The full partition contains 29 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=29
110  PRINT TABXY(25,Maxlines/2);"CENTER"
.
.
```

The following program segment demonstrates a technique to get text onto the screen quickly. 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.

```
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, the statement

```
OUTPUT CRT;" OUTPUT 1 WORKS WELL TOO"
```

writes the quoted text to the screen.

The display responds to several of the standard ASCII “control codes.” These characters can be sent to the CRT by printing or sending the CHR$ function of the ASCII number. For example, CHR$(7) is the control code for the “bell” (CTRL-G) and has the effect of sounding the beeper. For more information on control codes recognized by the CRT, see the Instrument Basic Users Handbook.

Note It is sometimes a practice to embed these control codes in your PRINT statements when using external computers to develop programs. For example, the 9836 Series 200 Workstation allows you to enter control characters directly into the program using the “ANY CHAR” key. If you do this, do not attempt to use the Instrument BASIC editor on the program. This editor does not recognize embedded control codes and its actions may be unpredictable.

Using Graphics

You can enable the display of graphics information by pressing [**BASIC**] [DISPLAY SETUP] [GRAPHICS **ON** OFF]. This information is generated primarily by the Instrument BASIC graphics statements.

Graphics and Display Partitions

You can position a graphics display area anywhere within the FULL display partition using the VIEWPORT statement. However, when you select the UPPER or LOWER partition, the analyzer shows you only those graphics that would be displayed in the lower half of the FULL partition. You must define a display area that falls within this lower half if you want to ensure that all graphics output is displayed within an UPPER or LOWER partition. For example, the following program line defines a display area that completely fills the lower half of the FULL partition:

```
VIEWPORT 0,RATIO*100,0,49
```

Note If you are converting an Instrument BASIC program written for the Agilent 35665A to one that works for the Agilent 35670A, you can insert the following lines to make graphics statements work properly:

```
GRAPHICS ON
WINDOW 0,474,0,345
```

Graphics Line Buffering

When lines are drawn by a graphics statement, the endpoint coordinates and pen information for each line is normally saved in a graphics buffer in the analyzer's memory. This allows the lines to be redrawn automatically whenever you change the display partition. However, as the complexity of a graphic increases, the amount of memory required for the buffer also increases. You can prevent lines from being saved to the buffer, conserving the memory they would require, by sending the following GPIB command: "DISP:PROG:VECT:BUFF OFF". When you want lines to be saved again, send "DISP:PROG:VECT:BUFF ON".

Graphics Pens

The PEN statement determines whether other graphics statements will draw or erase lines. When you use a nonzero pen number, graphics statements draw lines. When you use the pen number "0," graphics statements erase lines—or more exactly, they erase those portions of any graphic elements that lie along the drawing path. The pen numbers used to draw a graphic on the analyzer's screen are also used to plot the same graphic on an external plotter. So if you use multiple pen numbers, the graphic will be drawn properly on the monochrome screen and plotted properly on a color plotter.

Example Program

The following program demonstrates many of the techniques discussed so far. Running this program produces the "HELP" screen displayed in figure 7-2.

```
10   DIM A$[58],String$[2000]
20   GINIT
30   CLEAR SCREEN
40   GCLEAR
50   OUTPUT 800;"DISP:PROG FULL"
60   OUTPUT 800;"DISP:PROG?"
70   ENTER 800;P$
80   GRAPHICS ON
90   FRAME
100  MOVE 0,91
110  DRAW 100*RATIO,91
120  PRINT TABXY(28,2);"HELP"
130  OUTPUT A$;" This program demonstrates how to print"
140  String$=String$&A$
150  OUTPUT A$;" several lines of text at one time.  This"
160  String$=String$&A$
170  OUTPUT A$;" method offers the fastest possible print speed."
180  String$=String$&A$
190  PRINTER IS CRT;WIDTH 2000     !prevent auto cr/lf
200  PRINT TABXY(1,4);String$
210  END
```

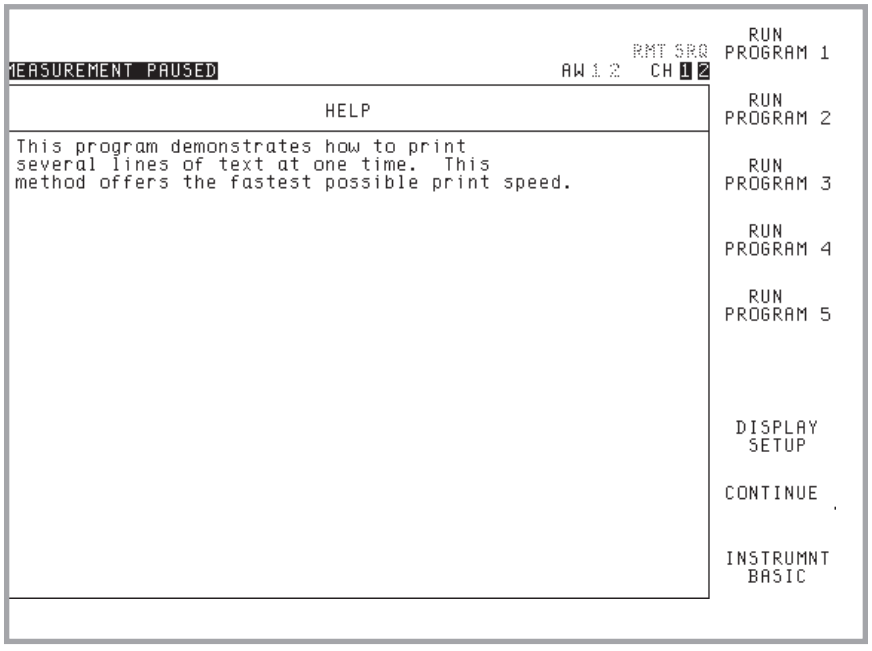

Figure 7-2. HELP Screen Output

8

Interfacing with the GPIB

8

Interfacing with the GPIB

Introduction

This section describes the techniques necessary for programming the GPIB interface. It also describes specific details of how this interface works and how to use it to control or interface with systems containing various GPIB devices.

The GPIB interface is Agilent Technologies's implementation of the IEEE-488.1 Digital Interface for Programmable Instrumentation. The acronym GPIB stands for "Agilent Technologies Interface Bus," and is often referred to as the "bus."

The GPIB Interface is both easy to use and allows great flexibility in communicating data and control information between the Instrument BASIC program and external devices.

Instrument BASIC is essentially an GPIB instrument controller residing inside an instrument. It uses the host instrument's GPIB interface for external communication and an internal GPIB interface to communicate with the host instrument. This unique arrangement presents a few differences between Instrument BASIC's implementation of GPIB control and the standard

Agilent 9000 Series 300 BASIC Controller. A description of the interaction of Instrument BASIC with the host instrument and the external GPIB interface is given in the section entitled "The Instrument Basic GPIB Model."

Communicating with GPIB Devices

This section describes programming techniques used to transfer data to and from GPIB devices. General bus operation is described in a later section.

GPIB Device Selectors

Since the GPIB allows the interconnection of several devices, each device must have a means of being uniquely accessed. A device selector consists of two parts: the interface select code and the device's primary address. When a particular GPIB 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 GPIB device selector. Instrument BASIC programs run inside a host instrument and communicate with it over the internal bus, which is addressed with select code 8. Instrument BASIC programs can also communicate with external devices via the host instrument's GPIB interface. The external bus select code is 7.

The second part of an GPIB device selector is the device's primary address. Each GPIB device has a primary address which can be configured. The address can range from 0 to 30. For example, to specify the device on the interface at select code 7 (external bus) with a primary address of 22, use device selector = 722.

Each device's address must be unique. The procedure for setting the address of an GPIB device is given in the installation manual for each device. Since the host instrument 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 8xx (e.g., 800, 811, 822, etc.).

Secondary Addressing

Many devices have operating modes which are accessed through the extended addressing capabilities defined in the bus standard. Extended addressing provides for a second address parameter in addition to the primary address. Examples of statements that use extended addressing are as follows:

```
100  ASSIGN @Device TO 72205  !22=primary, 05=secondary
110  OUTPUT @Device;Message$

200  OUTPUT 72205;Message$

150  ASSIGN @Device TO 7220529      !Additional secondary
160                                 !address of 29
170  OUTPUT @Device;Message$

120  OUTPUT 7220529;Message$
```

The range of secondary addresses is 00-31. Up to six secondary addresses may be specified—a total of 15 digits including interface select code and primary address. Refer to the device's operating manual for programming information associated with the extended addressing capability.

Moving Data Through the GPIB

Data is sent from the program through the GPIB with the OUTPUT statement. Data is entered into the program with the ENTER statement.

The following examples illustrate the use of GPIB device selectors with OUTPUT and ENTER statements.

Examples

```
100   GPIB=7
110   Device_addr=22
120   Device_selector=GPIB * 100 + Device_addr
130   !
140   OUTPUT Device_selector;"SYST:ERR?"
150   ENTER Device_selector;Reading

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

440   OUTPUT 800;"SOUR:FREQ 1 KHZ"

380   ENTER 724;Readings(*)
```

General Structure of the GPIB

Communications through the GPIB are made according to a precisely defined set of rules. These rules ensure that only orderly communication takes place on the bus.

For conceptual purposes, the organization of the GPIB can be compared to that of a committee. A committee uses “rules of order” to govern the manner in which they conduct their business. For example, a committee may conduct their meetings using “Robert’s Rules of Order.” For the GPIB, the rules of order are the IEEE 488.1 standard.

The GPIB System Controller is analogous to the chairman of a committee. Only one device can be designated System Controller and it is designated before running a program. The System Controller cannot be changed while under the control of a Instrument Basic program. However, as it is possible for a chairman to designate an “acting chairman” for the committee, so can control be passed to another device on the GPIB. This device is called the Active Controller. It can be any device capable of directing GPIB activities, such as an instrument (using printing and plotting functions) or a desktop computer.

When the System Controller is first turned on or reset, it assumes the role of Active Controller. These responsibilities may be subsequently passed to another device while the System Controller tends to other business. This ability to pass control allows more than one computer to be connected to the GPIB at the same time.

In an effective committee, only one person may speak at a time. It is the responsibility of the chairman to "recognize" which member is to speak. Usually, all committee members present are expected to listen at all times; however, this is not always the case on the GPIB. One of the most powerful features of the bus is the ability to selectively send data to individual (or groups of) devices. This allows fast talkers to communicate with fast listeners without having to wait for slower listeners on the bus.

During a committee meeting, the current chairman is responsible for telling the committee which member is to be the "talker" and which members are to be the "listeners." Before these assignments are given, she gets the attention of the members. The talker and listeners are designated and then the talker presents the data. The designation process may be repeated after the talker has completed his message.

On the GPIB, the Active Controller takes similar action when a talker and the listener(s) are designated. The attention signal line (ATN) is asserted while the talker and listener(s) are being addressed. ATN is then cleared, signaling that those devices not addressed to listen may ignore all subsequent data messages. Thus, the ATN line separates data from commands. Commands are accompanied with the ATN line being true, while data messages are sent with the ATN line being false.

On the GPIB, devices are addressed to talk and addressed to listen in an orderly manner. The Active Controller first sends a single command that causes all devices to stop listening. The talker's address is then sent, followed by the address(es) of the listener(s). After all listeners have been addressed, the data is sent from the talker to the listener(s). Only device(s) addressed to listen accept any data that is sent through the bus (until the bus is reconfigured by subsequent addressing commands).

The transfer of data, called a data message, exchanges information between devices on the GPIB. A committee conducts business by exchanging ideas and information between the speaker and those listening to his presentation. On the GPIB, data is transferred from the active talker to the active listener(s) at a rate determined by the slowest active listener on the bus. This restriction on the transfer rate is necessary to ensure that no data is lost by any device addressed to listen. The handshake used to transfer each data byte ensures that all data output by the talker is received by all active listeners.

Examples of Bus Sequences

With Instrument BASIC, all data transfers through the GPIB involve a talker and onlyone listener.

The following example illustrates the sequence of commands which are generated by the Active Controller to send data to an GPIB device through the bus with a simple OUTPUT statement.

```
OUTPUT 701;"DATA"
```

1. The unlisten command is sent.
2. The talker's address, which is also a command, is sent. In this case, the address of the active controller.
3. The listener's address (01), which is also a command, is sent.
4. The data bytes "D", "A", "T", "A", CR, and LF are sent; all bytes are sent using the GPIB's interlocking handshake to ensure that the listener has received each byte.

Similarly, all ENTER statements involve transferring data from a talker to only one listener. For instance, the following ENTER statement invokes the following sequence of commands and data-transfer operations.

```
ENTER 722;Voltage
```

1. The unlisten command is sent.
2. The talker's address (22), which is a command, is sent.
3. The listener's address, also a command, is sent. In this case, the listener's address is the active controller's address.
4. The data is sent by device 22 to the controller using the GPIB handshake.

General Bus Management

The GPIB standard provides several mechanisms that allow managing the bus and the devices on the bus. The following is a summary of the statements that invoke these control mechanisms.

ABORT is used to abruptly terminate all bus activity and reset all devices to their power-on states.

CLEAR is used to set all (or only selected) devices to a pre-defined, device-dependent state.

LOCAL is used to return all (or selected) devices to local (front panel) control.

LOCAL LOCKOUT is used to disable all devices' front panel controls.

REMOTE is used to put all (or selected) devices into their device-dependent, remote modes.

SPOLL is used to perform a serial poll of the specified device which must be capable of responding.

TRIGGER is used to send the trigger message to a device (or selected group of devices).

These statements (and functions) are described in the following sections. However, the actions that a device takes upon receiving each of the above statements are generally different for each device. For external devices, refer to the particular device's documentation to determine how it responds.

All of the bus management statements, with the exception of ABORT, require that the Instrument BASIC program be the Active Controller on the interface. A running program is always the Active Controller on the internal interface (select code 8). For the program to be the active controller on the external interface (select code 7), the host instrument must either be set as the System Controller or have control passed to it from the external controller. The program automatically assumes the controller status of the host instrument. For additional information refer to "The Instrument Basic GPIB Model" section later in this chapter.

REMOTE

External Devices

Most GPIB devices can be controlled either from the front panel or from the bus. The device is in the "Local" state if the front panel controls are currently functional. If the device is controlled through the GPIB, it is in the Remote state. Pressing the [**Local/GPIB**] key returns the device to Local (front panel) control; unless the device is in the "Local Lockout" state, or the device is the host instrument.

The Remote message is automatically sent to all devices whenever the System Controller is powered on, is reset, or sends the Abort message. A device enters the Remote state automatically whenever it is addressed. The REMOTE statement also sends the Remote message. This causes all (or specified) devices on the bus to change from local control to remote control. The host instrument must be set to System Controller before an Instrument BASIC program can execute the REMOTE statement on select code 7 (the external bus).

Examples

```
REMOTE 7


ASSIGN @Device TO 700
REMOTE @Device


REMOTE 700
```

Host Instrument

The REMOTE statement has no effect on the host instrument because it is always in remote control whenever an Instrument BASIC program is running. Specifying the internal interface in a REMOTE statement has no effect and does not generate an error.

LOCAL LOCKOUT

External Devices

The Local Lockout message effectively locks out the “local” switch present on most GPIB device front panels. This prevents anyone from interfering with the device’s system operations by pressing buttons. Local lockout maintains system integrity. 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 is sent to all devices on the external interface.

Examples

```
ASSIGN @GPIB TO 7
LOCAL LOCKOUT @GPIB


LOCAL LOCKOUT 7
```

The Local Lockout message is cleared when the Local message is sent by executing the LOCAL statement. Executing the ABORT statement does not cancel the Local Lockout message.

Host Instrument

The Local Lockout message is not supported for the host instrument because some front panel functionality is always necessary in order to pause or to abort the program. Specifying the internal interface in a LOCAL LOCKOUT statement does not generate an error and has no effect.

LOCAL

External Devices

It is good systems practice to return all devices to local control upon conclusion of remote-control operations. For example, an operator might need to troubleshoot or to work from the front panel to make special tests. 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. However, if the interface select code alone is specified (LOCAL 7), the Local message is sent to all devices on the external interface. Any previous Local Lockout message which is still in effect is automatically cleared.

Examples

```
ASSIGN @GPIB TO 7
LOCAL @GPIB


ASSIGN @Device TO 700
LOCAL @Device
```

Host Instrument

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

TRIGGER

External GPIB Devices

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

Examples

```
ASSIGN @GPIB TO 7
TRIGGER @GPIB

ASSIGN @Device TO 707
TRIGGER @Device
```

Specifying only the interface select code sends a GET message to all devices currently addressed to listen on the bus. Specifying a device's primary address in the statement triggers only the device addressed by the statement.

Host Instrument

The TRIGGER statement is fully compatible on the internal GPIB interface. The Agilent 35670A must be set to trigger on the GPIB for this statement to be effective.

```
OUTPUT @Agilent35670a;"TRIG:SOUR BUS"
TRIGGER @Agilent35670a
```

CLEAR

External GPIB Devices

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 GPIB 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.

Examples

```
ASSIGN @GPIB TO 7
CLEAR @GPIB

ASSIGN @Device TO 700
CLEAR @Device
```

Host Instrument

The CLEAR statement is fully compatible on the internal interface.

ABORT

External Devices

This statement terminates all activity on the external bus and returns all of the devices on the GPIB to a reset (or power-on) condition. Whether this affects other modes of the device depends on the device itself. The Instrument BASIC program must be the Active Controller 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 returns active control to the program. Only the interface select code is specified; primary-addressing information (such as 724) is not included.

Examples

```
ASSIGN @GPIB TO 7
ABORT @GPIB

ABORT 7
```

Aborting the Internal Bus

ABORT is not supported for the internal bus, select code 8. Executing ABORT 8 does not generate an error.

GPIB Service Requests

Most GPIB devices, such as voltmeters, frequency counters, and spectrum 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 documentation, operating or programming manuals, for each device describes the device’s capability to request service and the conditions in which the device requests 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 executes 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 and executes remedial action.

Setting Up and Enabling SRQ Interrupts

In order for an GPIB device to initiate a service routine in the Active Controller, two prerequisites must be met:

1. The SRQ interrupt event must have a defined service routine.
2. 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  GPIB=7
110  ON INTR GPIB GOSUB Service_routine
120  !
130  Mask=2
140  ENABLE INTR GPIB;Mask
```

Since Instrument BASIC 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 Agilent 9000 Series 300 BASIC programs.

When an SRQ interrupt is generated by any device on the bus, the program branches to the service routine when it exits the current line — either when the execution of the line is completed or when the line calls a user-defined function. The service routine must perform the following operations:

1. Determine which device is requesting service (parallel poll).
2. Determine what action is requested (serial poll).
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, the interrupt may not be immediately detected when the SRQ line goes low. This implies that an interrupt may not be generated if the SRQ is present momentarily but does not remain long enough to be sensed by the controller. The level-sensitive nature of the SRQ line also has implications, which are described in the following example.

Example of a SRQ Interrupt

Assume only one device is currently on the bus. The following service routine first serially polls the device requesting service, thereby clearing the interrupt request. In this case, the controller did not have to determine which device was requesting service because only one device is on the bus. Also, the type of interrupt is not determined because only service request interrupts are enabled in Instrument BASIC. The service is then 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
540  RETURN
```

The IEEE standard specifies that when an interrupting device is serially polled, it is to stop interrupting until a new condition arises (or the same condition arises again). In order to “clear” the SRQ line, it is necessary to perform a serial poll on the device. This poll is an acknowledgement from the controller to the device that it has seen the request for service and is responding. The device then removes its request for service by releasing the SRQ line. When the SRQ line is released, the line goes high.

If the SRQ line had not been released, the controller would have immediately branched to the service routine after enabling interrupts on the external interface (line 530). This is another implication of the level-sensitive nature of the SRQ interrupt.

Once an interrupt is sensed and logged, the interface cannot generate another interrupt until after the initial interrupt is serviced. The controller disables all subsequent interrupts from an interface until a pending interrupt is serviced. For this reason, it is necessary to allow for subsequent branching.

Conducting a Serial Poll

A sequential poll of individual devices on the bus in known as a Serial Poll. The status of a specific device is returned in response to a Serial Poll. One entire byte is used. This is called the "Status Byte" message. Depending on the device, the Status Byte may indicate an overload condition, a request for service, or a printer which is 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 Instrument Basic program must be the Active Controller in order to execute it.

Examples

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

Spoll_724=SPOLL(724)
```

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

Passing and Regaining Control

Passing control can be accomplished in one of two ways: it can be handled by the system, or it can be handled by the program. To handle it programmatically, use the PASS CONTROL statement. Instrument Basic or the analyzer can control the external bus (select code 7). The following statements first define the GPIB's select code, specify the new Active Controller's primary address and then pass control to that controller.

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

Once the new Active Controller has accepted active control, the controller which passed control assumes the role of a non-Active Controller on the GPIB.

Note An Instrument BASIC program cannot act as a device when in the role of non-Active controller.

Active control of the internal GPIB bus (select code 8) cannot be passed. The statement "PASS CONTROL 800" passes control of the external bus to the instrument. This is required whenever the analyzer performs a plot operation to a peripheral on the bus or the analyzer accesses an external disk drive. These concepts are discussed next in "The Instrument Basic GPIB Model."

The Instrument BASIC GPIB Model

The fact that Instrument BASIC resides in, and co-exists with an instrument creates a large set of possible interactions, both internally within the instrument as well as externally with other controllers and instruments. This section defines the principal players and rules of order when Instrument BASIC executes within the host instrument.

External and Internal Busses

There is physically only one GPIB port and one GPIB address for the Agilent 35670A. Instrument BASIC has access to two GPIB ports: the "real" external port (select code 7) and a "virtual" internal port (select code 8), through which it communicates with the Agilent 35670A. See figure 8-1.

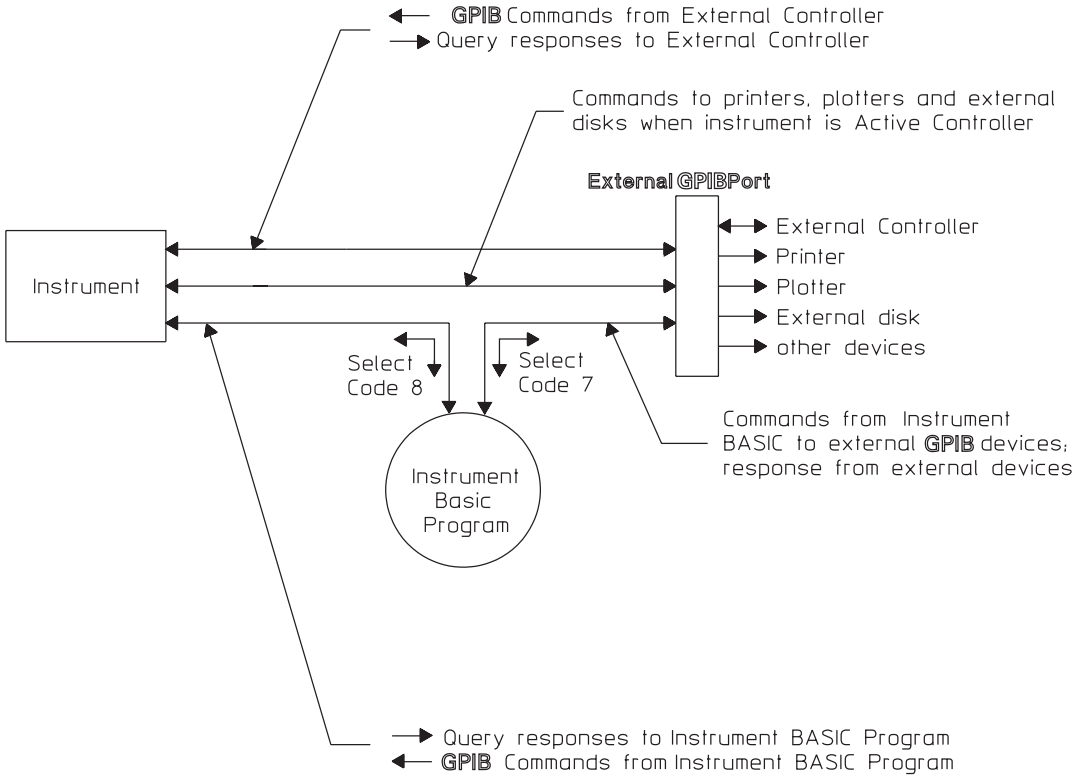

Figure 8-1. Agilent 35670A External and Internal Port

Service Request Indicators

An external controller may perform a serial poll (SPOLL) at any time without affecting a running Instrument BASIC 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 Instrument BASIC 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 no active Instrument BASIC program.

The two SRI's are 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 causes the PROGRAM_RUNNING bit in the Operation Status register to go low. This can be used to generate an external SRQ. (For an example, see the example program, TWO_CTLR, in chapter 11.)

Status Registers

The Agilent 35670A's status registers contain information about various analyzer conditions. There are eight register sets. Their reporting structure is summarized in figure 8-2.

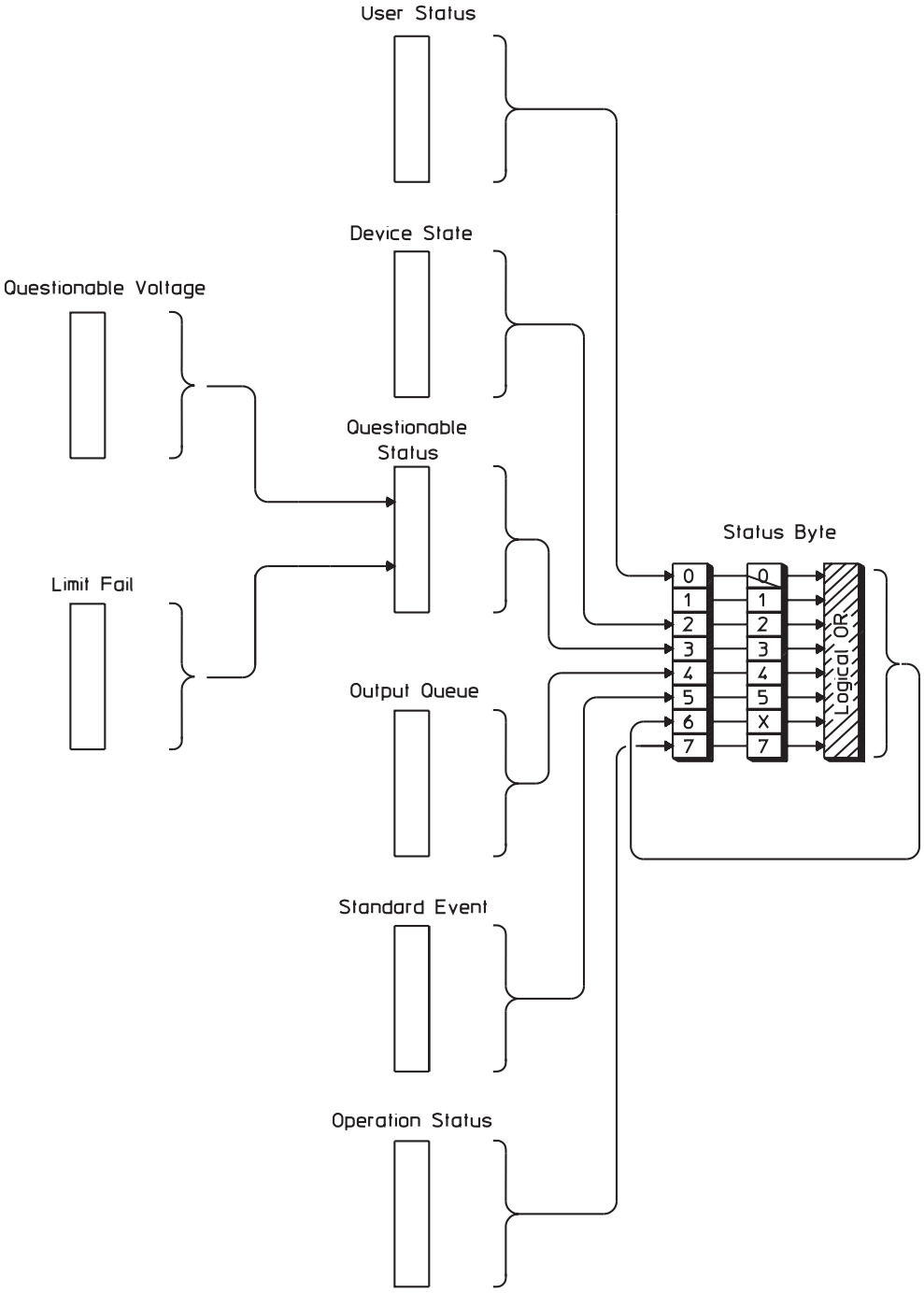

Figure 8-2. Agilent 35670A Status Registers

For more detailed information about the analyzer's register sets, refer to GPIB Programming with the Agilent 35670A.

Instrument BASIC as the Active Controller

The Instrument BASIC program is always the Active Controller on the internal bus (select code 8). When a program starts running, the GPIB controller status of the instrument is automatically passed to the program. See figure 8-3. For example, if the instrument is set as System Controller, a program running in the instrument automatically becomes the System Controller and the Active Controller on the external bus and the instrument relinquishes active control. When the program stops, the instrument regains active control.

Similarly, if an instrument set as Addressable Only is passed control from an external controller, any Instrument Basic program running in the instrument becomes active controller on the external interface.

There are two cases when a program running in an instrument can become the 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 Addressable Only and the instrument has been passed control from an external controller.

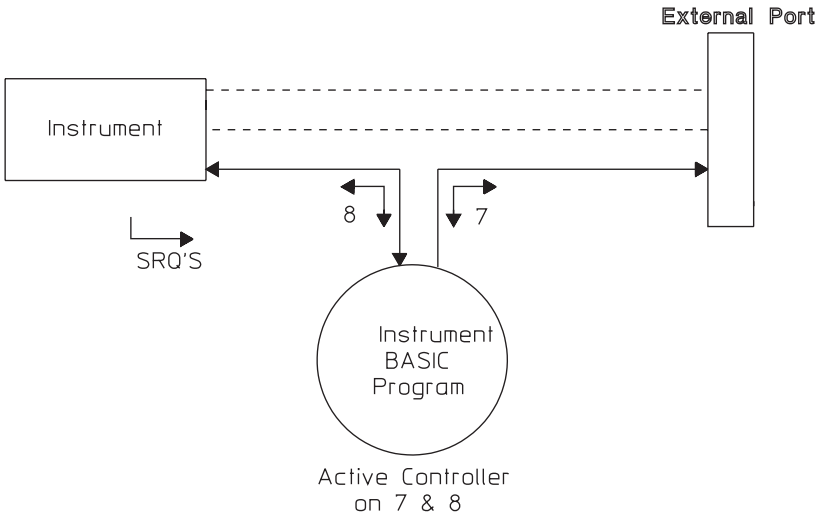

Figure 8-3. The Program as Active Controller on the External Interface

Passing Active Control to the Instrument

The only way that the Agilent 35670A can gain active control of the external interface while an Instrument Basic 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 external bus can pass control to any device on the interface by using the statement

```
PASS CONTROL 7xx
```

where "xx" represents the primary address of the device on the bus. However, since an Instrument BASIC program does not interface with the host instrument via select code 7, a different method must be used to pass control. To pass active control of the external interface from an Instrument BASIC 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. See figure 8-4. When the instrument is finished with its GPIB control activity, it automatically passes control back to the program. If the instrument is waiting for control and the Instrument Basic program terminates, control is implicitly passed back to the instrument. See figure 8-5.

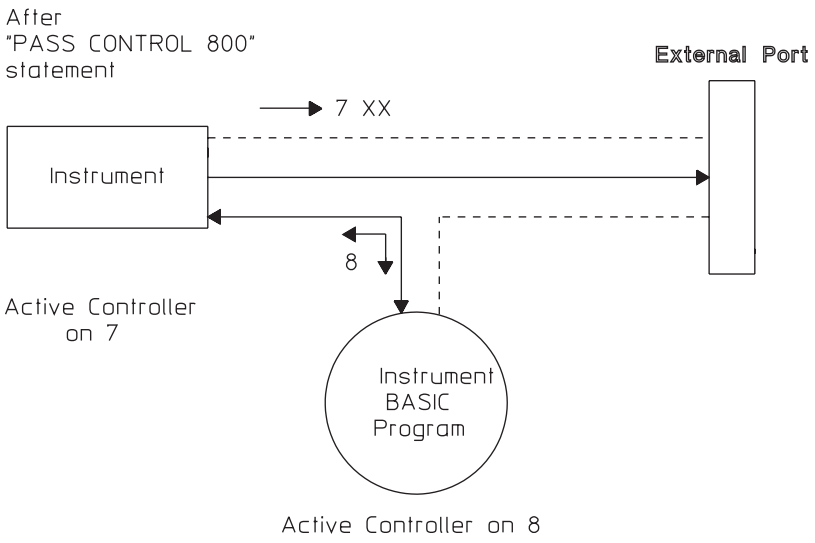

Figure 8-4. Passing Control of the External Interface to the Agilent 35670A

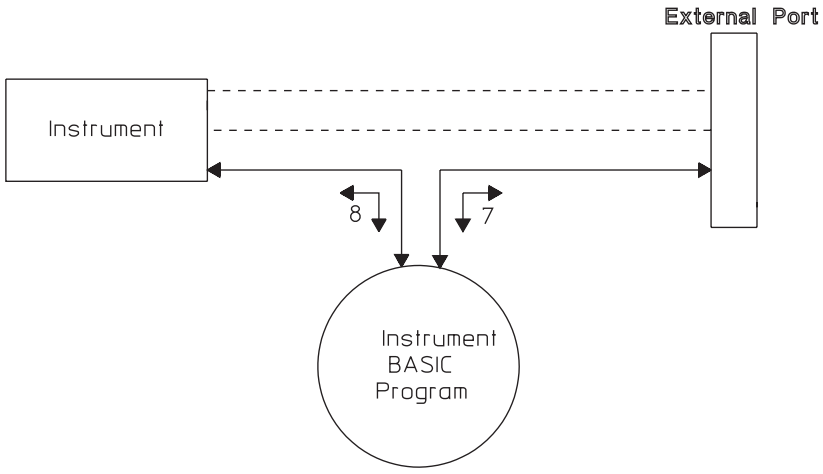

Figure 8-5. Control Passed Back to Program When Instrument Is Done

---

Note Control of the internal bus is used to govern access to the external bus. When the instrument is given control of the internal bus, it actually gains access to the external GPIB hardware.

Instrument BASIC as a Non-Active Controller

Instrument BASIC programs are always the Active Controller on the internal interface. There are two cases when an Instrument BASIC program does not have control of the external GPIB interface:

- When the host instrument is set as Addressable Only 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 to another device on the external interface.

In both of these cases, the Instrument Basic program cannot perform activities of any kind on the external bus. See figure 8-6.

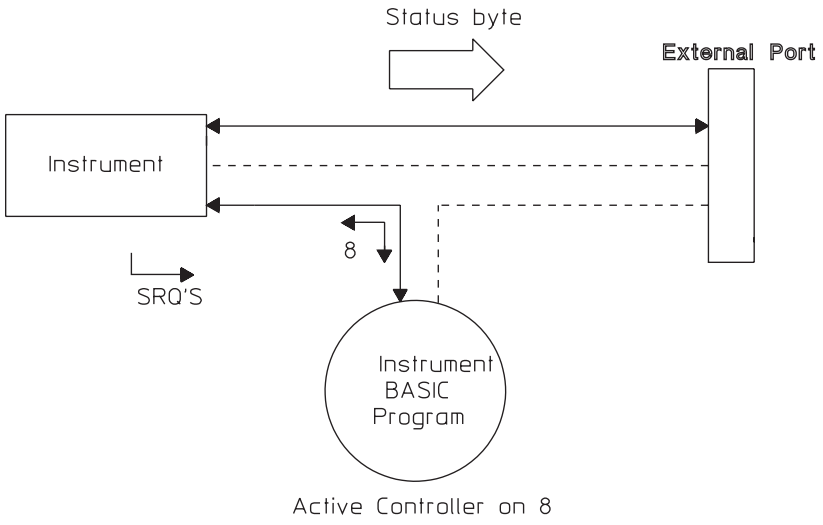

Figure 8-6. The Program as Non-Active Controller

Note An Instrument BASIC program cannot act as a device on the external bus. To communicate with an external controller, the Instrument BASIC program must be Active Controller and the external controller must act as the device (see "Interfacing with an External Controller").

Interfacing with an External Controller

So far, we have limited our discussion to the ability to interface Instrument BASIC programs via GPIB with a network of external devices. It is possible to include a computer in the network, and to interface an Instrument BASIC program with another program running in that computer.

External controller programs can interface with Instrument BASIC programs (hereafter referred to as "internal programs") over GPIB in two ways:

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 GPIB model, as presented earlier in this chapter.

The external program can make use of the extensive set of Agilent 35670A GPIB commands that interface with Instrument BASIC programs. These commands fall under the subsystems PROGram and MMEMory, and allow the external controller to remotely perform many of the Instrument BASIC front panel activities. This includes the ability to run, stop, pause, continue, get, save or delete an internal program. Commands in the SYSTem:COMMunicate:SERial subsystem configure the RS-232-C port. You can also remotely set a program's memory size and query or set the values of numeric and string variables.

Commands that allow you to transfer programs and program data to and from the instrument are included in the Agilent 35670A GPIB command set. Programs can be transferred (uploaded and downloaded) between an external controller and the program buffer in the instrument. Data can be transferred between an external program and a non-running internal program by setting and querying internal program variables. These commands are described in detail in GPIB Programming with the Agilent 35670A.

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 — not by any program running in it. Therefore, an Instrument BASIC program that is not the Active Controller cannot enter or output data via the external interface bus. 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. All Agilent 9000 Series 300 BASIC controllers have the ability to enter and output data via GPIB while acting as a non-Active Controller.

Note Moving data through the GPIB 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 concurrently.

One method of passing data between the two controllers is to first set the instrument as Addressable Only. Next, run an Agilent 9000 Series 300 BASIC program that starts the Instrument BASIC program and then passes control to it. Thereafter, the Instrument BASIC program can output data to, and enter data from, the external controller. The following two programs, found on the Agilent 35670A Example Programs disk, demonstrate how to transfer data between an internal program and an external controller program.

The first program, DTXFRB, runs on an Agilent 9000 Series 300 workstation. It assumes that a disk containing the corresponding Instrument BASIC program DTXFRA is in the Agilent 35670A disk drive. It remotely loads the Instrument BASIC program, starts it and then transfers active control to it. The Instrument BASIC program DTXFRA, with active control of the interface, queries the external program for the name of the drive to catalog, and then sends the cataloged string to the external program and passes back active control. After receiving the catalog data, the external program goes into a loop (line 460). This command continues to generate an error until control is passed back to the host computer, which again becomes the active controller.

```
10     !BASIC program:  DTXFRB — Data transfer BASIC to BASIC
20     !————————————————————————————————-
30     ! This program demonstrates how to transfer data from an  Instrument
40     ! BASIC program.  This program, which runs on the computer, loads a
50     ! a program into the Agilent35670A, runs it, and then gives it control of
60     ! the bus. This program then acts as a device on the bus; sending and
70     ! receiving data. Before running this program, a disc with the program
80     ! 'DTXFRA' should be in the Agilent35670A's internal drive.
90     ! The Agilent35670A should be at GPIB address 11 and the controller should
100    ! be at address 21.
110    !————————————————————————————————-
120    Scode=7                 !Select code for interface
130    Address=11              !Address for Agilent35670A
140    Agilent35670a=Scode*100+Address
150    CLEAR Agilent35670a
160    OUTPUT Agilent35670a;"*CLS"  !Clear the EVENT registers
170    CLEAR SCREEN            !Clear the display
180    !
190    DIM Directory$(1:100)[85] !Array to hold catalog listing
200    !
210    INPUT "Insert 'DTXFRA' disk into the Agilent35670A. Press <ENTER>",A$
220    DISP "Loading program on Agilent35670A..."
230    OUTPUT Agilent35670a;"MMEM:LOAD:PROG 'INT:DTXFRA'" !Load program from disk
240    OUTPUT Agilent35670a;"*OPC?"
250    ENTER Agilent35670a;Opc      !Wait here until program loaded
260    OUTPUT Agilent35670a;"*ESR?" !Read the EVENT STATUS reg
270    ENTER Agilent35670a;Esr
280    IF Esr>0 THEN           !Have any errors occurred
290     BEEP
300     DISP "Error while loading 'DTXFRA'...Cannot continue program."
310     STOP
320    END IF
330    !
340    OUTPUT Agilent35670a;"*PCB 21" !Set pass control back address
350    !                              to GPIB address for controller
360    DISP "Running the program..."
370    OUTPUT Agilent35670a;"PROG:STAT RUN"!Start the program
380    PASS CONTROL Agilent35670a          !Give program control of bus
390    !
400    OUTPUT Scode;":INTERNAL"       !Wait until addressed to talk
410    DISP "Reading data..."
420    ENTER Scode;Directory$(*)      !Wait until addressed to listen
430    !
440    FOR I=1 TO 100                 !Print the catalog
450     IF LEN(Directory$(I))>0 THEN PRINT Directory$(I)
460     NEXT I
470     !
480     Here: ON ERROR GOTO Here!Loop until control passed back
490     LOCAL Agilent35670a
500     DISP ""
510    END
```

```
10    ! BASIC program:  DTXFRA — Data transfer BASIC to BASIC
20    !————————————————————————————————-
30    ! This program demonstrates how to transfer data to and from an
40    ! external controller.  In this example a catalog listing is transferred
50    ! from the Agilent35670A to the external controller.  For more information
60    ! look at the program listing for 'DTXFRB'
70    !
80    ! This program is intended to be executed with Instrument Basic.
90    !————————————————————————————————-
100   DIM Directory$(1:100)[85]  !Create string array for catalog
110   !
120   Host=721                   !Address for external controller
130   !
140   ON ERROR GOTO 150          !Loop until control is passed to the Agilent35670A
150   ENTER Host;Stor_dev$       !Address Host to talk, read device to catalog
160   OFF ERROR
170   !
180   DISP "Reading catalog..."
190   CAT Stor_dev$ TO Directory$(*)!Catalog into the string array
200   !
210   DISP "Transferring data..."
220   OUTPUT Host;Directory$(*)      !Address Host to listen, write array
230   !
240   PASS CONTROL Host              !Pass control back to host
250   DISP "DONE"
260   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 "PROGram:NUMBer" and "PROGram:STRing" statements (and their query counterparts) are part of the Agilent 35670A GPIB commands. The internal program must not be running when these commands are executed.

The command

```
PROG:NUMBer <"label">,<numeric value>
```

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

```
PROG:STRing <"label$">,<"string value">
```

sets the value of a string variable in the program. In both the PROG:NUMB and PROG:STR commands and queries, the label must be a string in quotes. In the PROG:STRing command, the string variable data must also be in quotes.

Numeric and string parameters can also be queried. The query

```
PROG:NUMBer? <"label">
```

returns the value of the specified INTEGER or REAL variable. If you precede this GPIB command with the FORMat ASCII command (for example, OUTPUT 719;"FORM ASCii,5") the number returns as a readable ASCII number.

The query

```
PROG:STRing? <"label$">
```

returns the value of the specified string variable.

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

Examples

```
OUTPUT 711;"PROG:NUMB 'Test',99"

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

OUTPUT 711;"PROG:NUMB? 'Iarray(*)'"
```

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

```
10  ASSIGN @Prog TO 711
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 GPIB download command "PROG:DEFine" and its converse upload query "PROG:DEFine?." Programs that use these commands are executed 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 block or an indefinite length block. The indefinite length block is by far the easiest to transfer. It 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 GPIB interface.

When using the GPIB command "PROG:DEF" to download program lines, the "#0" should not be followed by a line-feed. Each program line then requires 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 GPIB control lines.

The following program runs on an external Agilent 9000 Series 300 workstation. It demonstrates downloading a short program into the program buffer of the instrument. It is included in the Agilent 35670A Example Programs disk. The file must be an ASCII file.

```
10      ! ————————————————————————————-
20      !  BASIC Program: DOWNLOAD670
30      ! This program downloads a file into an Agilent 35670A Instrument
40      ! BASIC program from an external controller.
50      ! The downloaded program must be an Agilent BASIC ASCII type file.
60      ! It will NOT work with DOS or HP-UX (untyped) files.
70      ! ————————————————————————————-
80      !
90      DIM Load_file$[20],Prog_line$[256],Command$[80],Name$[10]
100     DIM Diskname$[20],Answer$[2]
110     ASSIGN @Agilent35670a TO 711
120     !
130     ! The file of program to download must be an ASCII file.
140     !
150     INPUT "ENTER NAME OF FILE TO DOWNLOAD: ",Load_file$
160     ASSIGN @File TO Load_file$
170     INPUT "WHAT PROGRAM [1..5] DO YOU WANT TO DOWNLOAD TO?",Prognumber
180     OUTPUT @Agilent35670a;"PROG:NAME PROG"&VAL$(Prognumber)
190     OUTPUT @Agilent35670a;"PROG:DEL"
200     ON ERROR GOTO End_load
210     OUTPUT @Agilent35670a;"PROG:DEF #0";
220     LOOP
230      ENTER @File;Prog_line$
240      PRINT Prog_line$
250      OUTPUT @Agilent35670a;Prog_line$
260     END LOOP
270     !
280     End_load: !
290     OUTPUT @Agilent35670a;CHR$(10) END
300     INPUT "SAVE PROGRAM TO INSTRUMENT'S DEFAULT DRIVE? [Y/N]",Answer$
310     IF UPC$(Answer$)="Y" THEN
320      INPUT "ENTER NAME FOR DISK FILE: ",Diskname$
330      OUTPUT @Agilent35670a;"MMEM:STORE:PROGRAM '"&Diskname$&"'"
340     END IF
350     END
```

The OUTPUT statement on line 210 is terminated with a semicolon to suppress 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 in a pop-up message window 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 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 "PROG:DEL" command.

Uploading

The command “PROG:DEF?” is used to upload a program from the program buffer. The entire program is then returned as an 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 program demonstrates an uploading routine executed on an external controller. It is included in the Agilent 35670A Example Programs disk.

```
10     ! Agilent BASIC example program : UPLOAD670
20     ! ————————————————————————-
30     ! This program runs on an  BASIC workstation connected to
40     ! the Agilent 35670A with Instrument Basic installed.  The 35670A
50     ! must have its address set to 711 and must be set up as
60     ! ADDRESSABLE ONLY on the GPIB. This program uploads the
70     ! current program in the Agilent 35670A’s memory to an ASCII file
80     ! on the workstation’s current MSI disk.
90     ! ————————————————————————-
100    ASSIGN @Agilent35670a TO 711
110    DIM Prog_line$[256]
120    INPUT “ENTER FILE NAME TO UPLOAD PROGRAM INTO: ”,Filename$
130    PRINT Filename$
140    CLEAR @Agilent35670a
150    OUTPUT @Agilent35670a;"PROG:DEF?"
160    ENTER @Agilent35670a USING “#,A,D”;Prog_line$,Ndigits
170    ENTER @Agilent35670a USING “#,”&VAL$(Ndigits)&"D";Nbytes
180    PRINT Nbytes
190    Openfile(@File,Filename$,Nbytes)
200    ASSIGN @File TO Filename$
210    LOOP
220     ENTER @Agilent35670a;Prog_line$
230     EXIT IF LEN(Prog_line$)=0
240     PRINT Prog_line$
250     OUTPUT @File;Prog_line$
260    END LOOP
270    ASSIGN @File TO *
280    END
290    !
300    SUB Openfile(@File,Filename$,Fisize)
310     ON ERROR GOTO Openerr
320     IF Fisize MOD 256>0 THEN Fisize=Fisize+256
330     CREATE ASCII Filename$,Fisize DIV 256
340     Openerr:  !
350     IF ERRN<>54 THEN
360      PRINT ERRM$
370     END IF
380    SUBEND
```

The subroutine, Openfile, (lines 300 through 330) creates a LIF file in which to save the uploaded program. The number of 256 byte records declared in the CREATE ASCII statement (line 330) is simply the file size (declared in the definite block header) divided by 256. Line 320 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 250. 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:

```
300  SUB Openfile(@File,Filename$,Fisize)
310      ON ERROR GOTO Openerr
315      Fisize = Fisize + (Fisize * .15)
320      IF Fisize MOD 256>0 THEN Fisize=Fisize+256
330      CREATE ASCII Filename$,Fisize DIV 256
```

9

Interfacing with the RS-232-C Serial Port

9

Interfacing with the RS-232-C Serial Port

Introduction

This chapter describes the RS-232-C serial port, which is located on the analyzer's rear panel. It explains how you can configure the port and use it to communicate with external devices. There are examples throughout the chapter illustrating the use of this interface.

RS-232-C Serial Interface

The RS-232-C interface is used for simple asynchronous I/O applications such as driving printers, plotters, terminals and other peripherals or computers. It converts 8-bit parallel data into bit-serial data when it is transmitting. It converts bit-serial data back into parallel data when it is receiving.

You should first determine that the peripheral device and the analyzer are compatible. You can then configure the analyzer's serial interface—via front-panel softkeys or GPIB commands. The softkeys are located in the [**Plot/Print**] [MORE SETUP] [SERIAL SETUP] menu. The GPIB commands are located in the SYSTem:COMMunicate:SERial subsystem. You can specify speed, handshaking (flow control), parity, character length and the number of stop bits.

As shown it figure 9-1, both the analyzer and the Instrument BASIC program can use the RS-232-C serial port. The analyzer gains access to the port when you press [**Plot/Print**] [PLOT/PRNT DESTINATN] [OUTPUT TO SERIAL] or when you send the HCOP:DEST SER command via GPIB. The program gains access to the port with select code 9. You cannot change the port's select code.

Caution — If a real-time measurement is running, the RS-232-C port may lose incoming data. It is recommended that you pause a measurement while the port is receiving data.

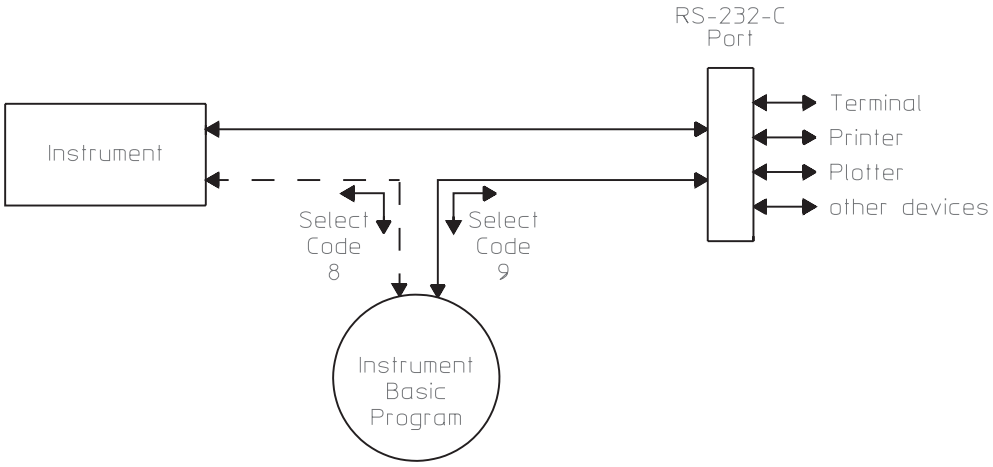

Figure 9-1. Agilent 35670A External RS-232-C Port

Asynchronous Data Communication

The terms Asynchronous Data Communication and Serial I/O refer to a technique of transferring information between two communicating devices by means of bit-serial data transmission. The data is sent, one bit at a time, and the characters are not synchronized with preceding or subsequent data characters. Each character is sent as a complete entity. Characters may be sent in close succession, or they may be sent sporadically as data becomes available. Start and stop bits are used to identify the beginning and end of each character, with the character data placed between them.

Asynchronous Transmission

The Agilent 35670A supports the asynchronous protocol. Asynchronous data communication is used in applications where high data integrity is not mandatory. Data is transmitted one character at a time. A start bit and one or more stop bits enclose the data character. Each character is individually synchronized or timed. The start and stop bits provide the timing.

Asynchronous Character Format

Each character consists of a start bit, 5 to 8 data bits, an optional parity bit and 1 or 2 stop bits. The Agilent 35670A does not support a 1.5 stop bit. The total time from the beginning of one start bit to the end of the last stop bit is called a character frame.

Figure 9-2 shows a structure of an asynchronous character and its relationship to previous and succeeding characters.

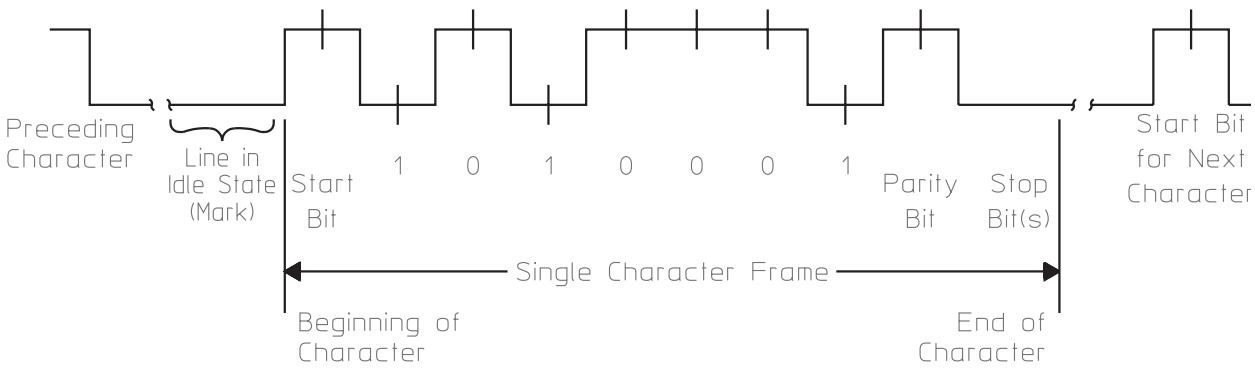

Figure 9-2. Asynchronous Format of a Single Character

Hardware Requirements

The analyzer's RS-232-C connector is located on the rear panel. It is a 9-pin, male, D-series connector that is set up for DTE (Data Terminal Equipment) applications. Figure 9-3 shows the pin locations on the connector.

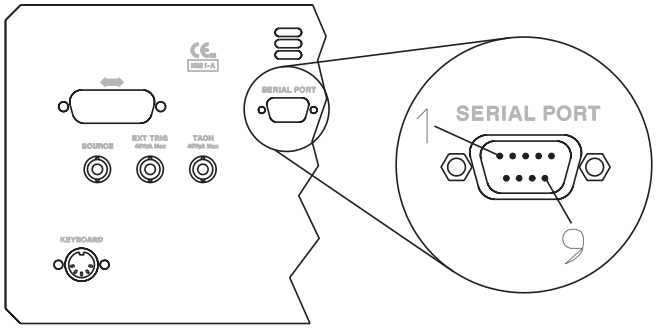

Figure 9-3. SERIAL PORT Pin Locations

Table 9-1 describes the signals on each pin of the analyzer's RS-232-C connector. You can use the Agilent 24542G cable to properly connect these signals to most Hewlett-Packard printers and plotters. This cable and many others are available through your local Agilent Technologies Sales and Service Office. The cable you order should not be more than 50 feet long.

Table 9-1. SERIAL PORT Signal Descriptions

Pin	Signal Description	I/O
1	not used	–
2	Received Data	Input
3	Transmitted Data	Output
4	Data Terminal Ready (fixed high)	Output
5	Signal Ground	–
6	Data Set Ready	Input
7	Request to Send (fixed high)	Output
8	not used	–
9	not used	–

Configuring the RS-232-C Port

Before information can be successfully transferred between two RS-232-C devices, you must configure their serial ports. To determine how their ports should be configured, answer the following questions about the transfer protocol:

- What line speed (baud rate) is being used?
- How many bits (excluding start, stop and parity bits) are included in each character?
- How many stop bits are required on each character you transmit?
- What parity is being used: none, odd, or even?
- What form of handshaking (pacing, flow control) is used?

This section of the chapter tells you how to control these protocol parameters on the Agilent 35670A—using either front-panel softkeys or GPIB commands. (To learn how to control them on your RS-232-C device, refer to its documentation.) The values you specify for these parameters are stored in non-volatile RAM, so they are not lost when you turn the analyzer off.

Speed (Baud Rate)

You can specify the rate at which data bits are transferred between the analyzer and the external device with the [**Plot/Print**] [MORE SETUP] [SERIAL SETUP] [BAUD RATE] softkey or with the SYSTem:COMMunicate:SERial[:RECeive]:BAUD command. Valid values are 300, 1200, 2400, 4800, and 9600 baud.

Character Length

You can specify character length with the [**Plot/Print**] [MORE SETUP] [SERIAL SETUP] [BITS/CHAR] softkey or with the SYSTem:COMMunicate:SERial[:RECeive]:BITS command. Valid values are 5, 6, 7, and 8 bits.

Number of Stop Bits

You can specify the number of stop bits with the [**Plot/Print**] [MORE SETUP] [SERIAL SETUP] [STOP BITS ONE TWO] softkey or with the SYSTem:COMMunicate:SERial[:RECeive]:SBITS command. Valid values are 1 and 2 bits.

Parity

Parity is a technique used to detect transmission error by counting the number of bits in a character. Parity options include:

- NONE - Parity bit is not included.
- ODD - Parity is odd; there is an even number of "1"s in character bits.
- EVEN Parity is even; there is an odd number of "1"s in character bits.

You can control two separate parity parameters: parity type and parity verification.

Parity Type

You can specify the parity type with the [**Plot/Print**] [MORE SETUP] [SERIAL SETUP] [PARITY] softkey or with the SYSTem:COMMunicate:SERial[:RECeive]:PARity[:TYPE] command. Valid values are NONE, ODD, and EVEN.

Parity Verification

You can enable and disable parity verification with the [**Plot/Print**] [MORE SETUP] [SERIAL SETUP] [PRTY CHK ON OFF] softkey or with the SYSTem:COMMunicate:SERial[:RECeive]:PARity:CHECk command. Valid values are ON and OFF.

Handshaking

Controlling the flow of data is important during the data transfer operation to avoid losing data. The input buffer of a printer may become full as the speed of character transmission exceeds the printer's ability to print. Data might be lost if the printer runs out of paper during the transmission. To ensure that the transmitter does not send characters faster than the receiver can process them, a pacing mechanism is used to control the flow of data. This pacing mechanism is called a "handshake." Handshaking is performed automatically as part of the OUTPUT or ENTER operation.

The Agilent 35670A supports one of handshaking protocol for receiving data: XON/XOFF. It supports two protocols for transmitting data: XON/XOFF and DSR/DTR. Check your external device's documentation to verify that it supports the protocol you want to use.

Receiver Handshaking

You can specify the handshaking method for received data with the [**Plot/Print**] [MORE SETUP] [SERIAL SETUP] [RCVR PACE] softkey or the SYSTem:COMMunicate:SERIAL[:RECeive]:PACE command. Valid values for the softkey are XON/XOFF and NONE. Corresponding values for the GPIB command are XON and NONE.

Transmitter Handshaking

You can specify the handshaking method for transmitted data with the [**Plot/Print**] [MORE SETUP] [SERIAL SETUP] [XMIT PACE] softkey or the SYSTem:COMMunicate:SERIAL:TRANsmit:PACE command. Valid values for the softkey are XON/XOFF, DSR/DTR, and NONE. Corresponding values for the GPIB command are XON, DSR, and NONE.

Transferring Data

The serial RS-232-C interface is designed for relatively simple serial I/O operations.

Entering and Outputting Data

When the RS-232-C interface is properly configured, you are ready to begin data transfers. Outbound data messages are created by OUTPUT statements. Inbound data messages are created by the interface as messages are received from the peripheral device. They are transferred to Instrument Basic by ENTER statements.

Any valid OUTPUT or ENTER statement and variable(s) list may be used, but you must be sure that the data format is compatible with the peripheral device. For example, non-ASCII data sent to an ASCII line printer may result in unexpected behavior.

Outbound Data Messages

Outbound data messages are created when an OUTPUT statement is executed. Data is transmitted directly from the outbound buffer.

The output operation completes after the last byte of the character has been sent. The Instrument Basic controller waits until the last byte in the statement variable list is transmitted by the interface.

An end-of-line (EOL) sequence is automatically sent at the end of data unless a semicolon or END appears at the end of a OUTPUT statement. The semicolon delimiter overrides EOL sequence output.

The Output Statement

Data items are transferred one byte at a time, beginning with the left-most item in the source list and continuing until all of the source items have been sent. Items in the list must be separated by either a comma or a semicolon. Depending on the use of item separators in the source lists, data items in the output may or may not be separated by item terminators. The end-of-line (EOL) sequence is described above.

Example Program Statements

```
OUTPUT 9;"Hello World"

PRINTER IS 9
PRINT "Hello World"
```

Inbound Data Messages

Inbound data messages are created by the RS-232-C interface as information is received from the peripheral device. ENTER statements are terminated when a new-line character (ASCII LF, decimal value 10) is encountered. A carriage-return (ASCII CR, decimal value 13) does not terminate a data string.

Caution — Running a real-time measurement while the analyzer is receiving data at a high baud rate, may result in an overrun condition.

The Enter Statement

Items in ENTER statements can be separated by either a comma or a semicolon. Trailing punctuation is not allowed. A data item is terminated with a new-line character (ASCII 10).

Example Program Statement

```
ENTER 9;A$
```

Clearing the Input Buffer

The input buffer holds 256 characters. Sometimes the input buffer contains characters from previous transmissions. The statement, CLEAR 9, clears the input buffer.

Error Detection

Four types of incoming data errors can be detected by the Agilent 35670A.

- Parity Errors are signaled when the parity bit does not match–even or odd–the number of "ones" (including the parity bit) as defined by the interface configuration. When parity is disabled, a parity check is not made.

- Framing errors are signaled when start and stop bits are not properly received during the expected time frame. They can be caused by a missing start bit, noise errors near the end of the character, or by improperly specifying character length at the Agilent 35670A or the peripheral device.

- Overrun errors result when the Agilent 35670A does not consume characters as fast as they arrive.

Overrun conditions can occur during real-time measurements. If a real-time measurement is running at the same time the RS-232-C port is transferring data at a high baud rate, you may encounter an overrun error. This happens as a result of sending a second character to the analyzer before it can read the first character.

Another type of overrun occurs when characters are sent to the analyzer but an Instrument Basic program is not reading the characters. There is a 256 byte input buffer which is filled as characters are received. If this input buffer is filled, an overrun error is generated.

- Received BREAKs are detected as a special type of framing error. They generate the same type of Instrument Basic error as framing errors.

The Device State Register

The Agilent 35670A's status registers monitor various analyzer conditions. As shown in figure 9-4, the Device State register set contains several bits that monitor the condition of RS-232-C data transfers:

- Bit 6 is set to 1 when a character is in the input buffer.
- Bit 7 is set to 1 when input is held off due to handshake protocol conditions.
- Bit 8 is set to 1 when output is held off due to handshake protocol conditions.
- Bit 9 is set to 1 when a framing error, overrun error, parity error, or break is detected.

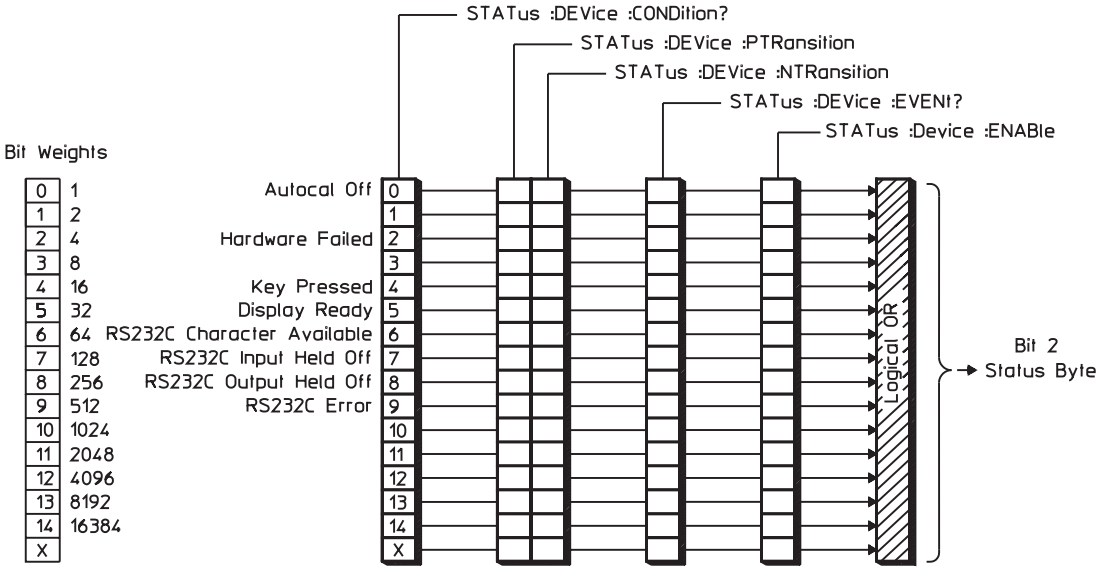

Figure 9-4. The Device State Register Set

Event-Initiated Branching

Instrument Basic allows event-initiated branching, which uses interrupts to redirect program flow. Each time the program finishes a line, the analyzer executes an "event-checking" routine. This "event-checking" routine causes the program to branch to a specified statement if an enabled event has occurred.

The Agilent 35670A supports one type of event-initiated branching. ON TIMEOUT generates an interrupt when an interface or device takes longer than a specified time to respond to a data-transfer handshake.

All "ON-event" statements have a corresponding "OFF-event" statement.

It is possible to temporarily disable an event-initiated branch. A special section of code can be "protected," that is, not be interrupted, with a DISABLE statement.

See "Program Structure and Flow" in "Instrument Basic Programming Techniques" (Instrument Basic Users Handbook) for additional information about event-initiated branching.

Timeouts

ON TIMEOUT defines and enables an event-initiated branch to be taken when an I/O timeout occurs on the specified interface. The timeout is specified in seconds. For the RS-232-C interface the maximum timeout is 25.5 seconds.

Timeouts apply to ENTER and OUTPUT statements, and operations involving the PRINTER IS, PRINTALL IS and PLOTTER IS external devices.

OFF TIMEOUT deactivates ON TIMEOUT. DISABLE does not affect ON TIMEOUT.

Examples

```
ON TIMEOUT 9, 1.5 GOTO 1200

ON TIMEOUT 9, .5 GOSUB Service_routine
```

10

Interfacing with the Parallel Port

10

Interfacing with the Parallel Port

Introduction

This chapter describes the parallel port, which is located on the analyzer's rear panel. It explains how you can use the interface to send data to an external plotter or printer.

The Parallel Interface

The parallel interface is used exclusively for output to plotters and printers. It is usually faster than the serial interface because it sends data one character (eight bits) at a time—on eight parallel data lines. (The serial port sends data one bit at a time—on a single data line.)

As shown it figure 10-1, both the analyzer and the Instrument BASIC program can use the parallel port. The analyzer gains access to the port when you press [**Plot/Print**] [PLOT/PRNT DESTINATN] [OUTPUT TO PARALLEL] or when you send the HCOP:DEST CENT command via GPIB. The program gains access to the port with select code 26. You cannot change the port's select code.

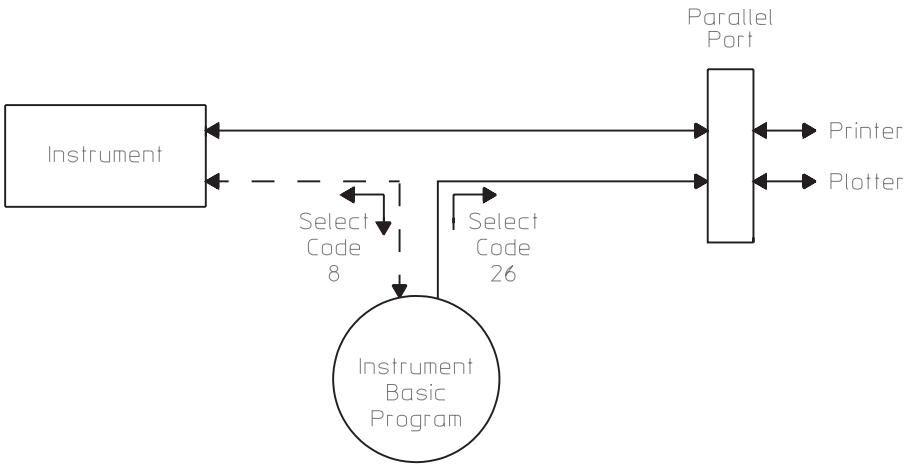

Figure 10-1. Agilent 35670A Parallel Port

Hardware Requirements

The analyzer's parallel connector is located on the rear panel. It is a 25-pin, female, D-series connector. Figure 10-2 shows the pin locations on the connector.

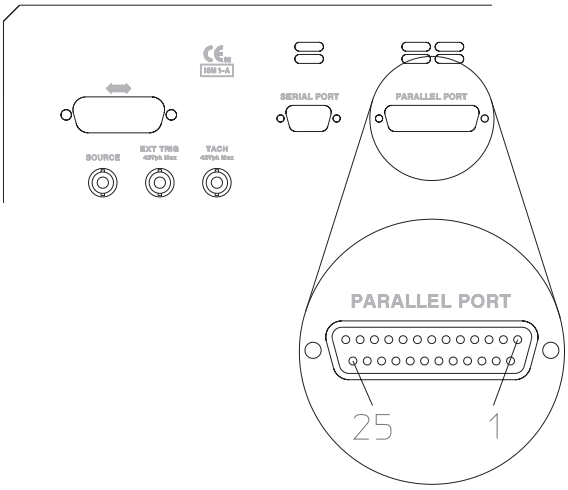

Figure 10-2. Parallel Port Pin Locations

Table 10-1 describes the signals on each pin of the analyzer's parallel connector. You can use the Agilent 92284A cable to properly connect these signals to most Hewlett-Packard printers and plotters. This cable and many others are available through your local Agilent Technologies Sales and Service Office. The cable you order should not be more than 10 feet long.

Table 10-1. PARALLEL PORT Signal Descriptions

Pin	Signal Description	I/O
1	Strobe	Output
2-9	Data 0-7	Outputs
10	Ack	Input
11	Busy	Input
12	Paper Empty	Input
13	Select	Input
14	not used	—
15	Error	Input
16	Init	Output
17	not used	—
18	Ground	—
19-25	not used	—

Transferring Data

Instrument BASIC programs use the OUTPUT statement to send data to plotters and printers. When an OUTPUT statement is executed, the program creates an outbound data message and sends it to the specified port. When the last character of the data message has been sent, the program sends an end-of-line (EOL) sequence to terminate output.

Note Be sure the data format of any item included in your OUTPUT statement is compatible with your plotter or printer. Sending incompatible data can result in unexpected behavior. (For example, do not send non-ASCII data to an ASCII line printer.)

The following examples both send data to a printer that is connected to the parallel port (select code 26):

```
OUTPUT 26;"Hello World"

PRINTER IS 26
PRINT "Hello World"
```

11

Example Programs

11

Example Programs

This chapter contains listings of some of the programs on the Agilent 35670A Example Programs disk. The programs included in this chapter are described below. You can load and run the program "READ_ME" for descriptions of other programs on the disk.

ARBSOURC Demonstrates programming the arbitrary source and transferring trace data to the data registers. The arbitrary source uses one of three waveforms, which have been downloaded to the data register. An existing trace is copied to the data register, uploaded, shifted left and right and then reloaded into the arbitrary source data register.

MANARM Demonstrates using the Standard Event Register to detect a WAITING_FOR_ARM event and generates an SRQ. Program handles SRQ interrupt and allows you to arm

the measurement.

OPC_SYNC Demonstrates synchronizing the program and the analyzer using the *OPC statement to set the OPERATION_COMPLETE bit in the Standard Event Status Register. The Status Register is masked to generate an SRQ when all pending operations have completed. The program handles the service request interrupt.

OPCQSYNC Demonstrates synchronizing the program and the analyzer using the *OPC? query. The programs pauses on an ENTER statement while it waits for the pending GPIB operations to complete and for a "1" to be returned in response to the *OPC? query.

RPNCALC Demonstrates waveform math capabilities of the analyzer by creating a immediate mode waveform calculator. Uses data registers to implement Reverse Polish Notation calculations. Allows most waveform math functions to be performed using trace data, register data, and constants.

TWO_CTLR Demonstrates using an external controller to download an Instrument Basic program, run it, and query variables.

WAI_SYNC Demonstrates synchronizing the program and the analyzer using the *WAI statement.

ARBSOURC

```
10     !————————————————————————
20     !  BASIC Program: ARBSOURCE
30     ! This program is used with the Agilent 35670a
40     ! Dynamic Signal Analyzer.  It allows a user
50     ! to create ramp, triangle, and square waves
60     ! and download them to an analyzer data register
70     ! to be used by the arbitrary source.
80     ! It also allows captured traces to be uploaded
90     ! to the program, shifted left or right, and then
100    ! downloaded into a data register.
110    !————————————————————————-
120    COM /Traces/ Trace_in(1:1024),Trace_out(1:1024)
130    COM /Assigns/ @Format_off,@Format_on,@Agilent35670a
140    ASSIGN @Format_on TO 800;FORMAT ON
150    ASSIGN @Format_off TO 800;FORMAT OFF
160    ASSIGN @Agilent35670a TO 800
170    INTEGER Byte_count,Block_count
180    ! Set data xfer to binary
190    OUTPUT @Format_on;"FORM:DATA REAL, 64"
200    Frequency=1024
210    High=2.5          ! high limit for arbitrary source
220    Low=-2.5          ! low limit for arbitrary source
230    !
240    !————————————————————————-
250    !  Setup display, and arbitrary source
260    !————————————————————————-
270    OUTPUT @Agilent35670a;"SYST:PRES"
280    OUTPUT @Agilent35670a;"DISP:FORM ULOW"
290    OUTPUT @Agilent35670a;"CALC1:FEED 'XTIM:VOLT 1'; *WAI"
300    OUTPUT @Agilent35670a;"CALC2:FEED 'D1'; *WAI"
310    OUTPUT @Agilent35670a;"ABOR;:INIT; *WAI"
320    OUTPUT @Agilent35670a;"INIT:CONT OFF"
330    Running=False
340    OUTPUT @Agilent35670a;"TRAC D1,TRAC1; *WAI"
350    OUTPUT @Agilent35670a;"SOUR:USER D1"
360    OUTPUT @Agilent35670a;"SOUR:FUNC USER"
370    OUTPUT @Agilent35670a;"SOUR:VOLT 0.99975 Vpk"
380    OUTPUT @Agilent35670a;"OUTP ON"
390    OUTPUT @Agilent35670a;"DISP:WIND1:TRAC:Y:AUTO ON"
400    OUTPUT @Agilent35670a;"CALC:ACT B"
410    OUTPUT @Agilent35670a;"DISP:WIND2:TRAC:Y:AUTO ON"
420    OUTPUT @Agilent35670a;"CAL:AUTO OFF"
430    !
440    !————————————————————————
450    !  Setup menu
460    !————————————————————————
470    ON KEY 0 LABEL "RAMP" GOSUB Ramp
480    ON KEY 1 LABEL "TRIANGLE" GOSUB Triangle
490    ON KEY 2 LABEL "SQUARE" GOSUB Square
500    ON KEY 3 LABEL "COPY     A ->D1" GOSUB Storetrace
```

```
510    ON KEY 4 LABEL "LOAD     D1 " GOSUB Data_in
520    ON KEY 5 LABEL "SHIFT    TRACE" CALL Shift_trace
530    ON KEY 6 LABEL "" GOTO Waiting
540    ON KEY 7 LABEL "PAUSE/   CONTINUE" GOSUB Pause_cont
550    ON KEY 8 LABEL "EXIT" GOTO Endit
560    !
570    Waiting:!GOTO Waiting
580    GOTO Waiting
590    !
600    !-----------------------------
610    Ramp:! create ramp wave in D1
620     GOSUB Enterfreq
630     DISP "CALCULATING RAMP WAVE"
640     N=1
650     FOR I=1 TO 1024/Period
660     Current=Low
670      FOR J=1 TO Period
680       Trace_out(N)=Current
690       N=N+1
700       Current=Current+(High-Low)/Period
710      NEXT J
720     NEXT I
730     !
740     CALL Data_out ! send Trace_out to D1
750     DISP ""
760     RETURN
770    !
780    !-----------------------------
790    Triangle:! create triangle wave in D1
800     GOSUB Enterfreq
810     DISP "CALCULATING TRIANGLE WAVE"
820     N=1
830     FOR I=1 TO 1024/Period
840      Current=Low
850      FOR J=1 TO Period/2
860       Trace_out(N)=Current
870       N=N+1
880       Current=Current+(High-Low)/Period
890      NEXT J
900      Current=Current-(High-Low)/Period
910      FOR J=1 TO Period/2
920       Current=Current-(High-Low)/Period
930       Trace_out(N)=Current
940       N=N+1
950      NEXT J
960     NEXT I
970     CALL Data_out ! send Trace_out to D1
980     DISP ""
990     RETURN
1000   !
1010   !-----------------------------
1020   Square:! create square wave in Trace_out array
1030    GOSUB Enterfreq
```

```
1040    DISP "CALCULATING SQUARE WAVE"
1050    Toggle=1
1060    N=1
1070    Current=Low
1080    FOR I=1 TO 1024/Period
1090     FOR J=1 TO Period
1100      Trace_out(N)=Current
1110      N=N+1
1120     NEXT J
1130     IF Toggle THEN Current=High
1140     IF NOT Toggle THEN Current=Low
1150     Toggle=NOT Toggle
1160    NEXT I
1170    CALL Data_out ! send Trace_out to D1
1180    DISP ""
1190   RETURN
1200   !
1210   !-----------------------------
1220   Storetrace:! Copy Trace 1 to D1
1230    OUTPUT @Agilent35670a;"TRAC D1,TRAC1"
1240   RETURN
1250   !
1260   ! Enter frequency, which must be power of two so
1270   ! that waveform is symmetrical in 1024 point block
1280   !-----------------------------
1290   Enterfreq:!
1300    INPUT "Enter frequency (as power of 2 = 128): ",Frequency
1310    IF Frequency  65536 THEN Frequency = 65536
1320    Period=128/Frequency*1024
1330   RETURN
1340   !
1350   !-----------------------------
1360   Data_in:! Read D1 trace data into Trace_in array
1370    OUTPUT @Format_on;"TRAC:DATA? D1"
1380    ENTER @Format_on USING "%,A,D";A$,Byte_count
1390    ENTER @Format_on USING "%,4D";Block_count
1400    ENTER @Format_off;Trace_in(*)
1410    ENTER @Format_on;A$
1420    OUTPUT @Agilent35670a;"ABOR;:INIT; *WAI"
1430    OUTPUT @Agilent35670a;"INIT:CONT OFF; *WAI"
1440   RETURN
1450   !
1460   !-----------------------------
1470   Pause_cont:! Pause or continues measurement
1480    IF Running THEN
1490     OUTPUT @Agilent35670a;"INIT:CONT OFF"
1500    ELSE
1510     OUTPUT @Agilent35670a;"INIT:CONT ON"
1520    END IF
1530    Running=NOT Running
1540   RETURN
1550   !
1560   Endit:END
```

```
1570   !
1580   SUB Data_out
1590   !————————————————————
1600   ! Sends trace data from Trace_out array to D1
1610   ! data register in Agilent 35670a and sets source
1620   ! output as arbitrary source
1630   !————————————————————
1640    COM /Traces/ Trace_in(1:1024),Trace_out(1:1024)
1650    COM /Assigns/ @Format_off,@Format_on,@Agilent35670a
1660    OUTPUT @Format_on;"TRAC:DATA D1,";
1670    OUTPUT @Format_on;"#48192";
1680    OUTPUT @Format_off;Trace_out(*);
1690    OUTPUT @Format_on;CHR$(10)
1700    FOR I=1 TO 1024
1710     Trace_in(I)=Trace_out(I)
1720    NEXT I
1730   Running=True
1740   SUBEND
1750   !
1760   SUB Shift_trace
1770   !————————————————————
1780   ! Shifts trace left or right by defined stepsize
1790   ! and loads shifted trace into D1
1800   !————————————————————
1810   COM /Traces/ Trace_in(1:1024),Trace_out(1:1024)
1820    Stepsize=10
1830    ON KEY 0 LABEL "",2 GOTO Waiting
1840    ON KEY 1 LABEL "SHIFT    LEFT",2 GOSUB Shift_left
1850    ON KEY 2 LABEL "SHIFT    RIGHT",2 GOSUB Shift_right
1860    ON KEY 3 LABEL "",2 GOTO Waiting
1870    ON KEY 4 LABEL "",2 GOTO Waiting
1880    ON KEY 5 LABEL "DEFINE   STEPSIZE",2 GOSUB Step_size
1890    ON KEY 6 LABEL "",2 GOTO Waiting
1900    ON KEY 7 LABEL "",2 GOTO Waiting
1910    ON KEY 8 LABEL "RETURN",2 GOTO Sub_end
1920    !
1930    Waiting:GOTO Waiting
1940    !
1950    Shift_left:! shift the trace left "Stepsize" bins
1960    DISP "SHIFTING D1 DATA"
1970    I=1
1980    FOR J=Stepsize TO 1024
1990     Trace_out(I)=Trace_in(J)
2000     I=I+1
2010    NEXT J
2020    FOR J=1 TO Stepsize-1
2030     Trace_out(I)=Trace_in(J)
2040     I=I+1
2050    NEXT J
2060    CALL Data_out
2070    DISP ""
2080   RETURN
2090   !
```

```
2100   !-----------------------------
2110   Shift_right:! shift the trace right "Stepsize" bins
2120    DISP "SHIFTING D1 DATA"
2130    I=1
2140    FOR J=Stepsize TO 1024
2150     Trace_out(J)=Trace_in(I)
2160     I=I+1
2170    NEXT J
2180    FOR J=1 TO Stepsize-1
2190     Trace_out(J)=Trace_in(I)
2200     I=I+1
2210    NEXT J
2220    CALL Data_out ! send trace to D1
2230    DISP ""
2240   RETURN
2250   !-----------------------------
2260   Step_size:!
2270    INPUT "Enter shift stepsize (1..1024)",Stepsize
2280   RETURN
2290   !-----------------------------
2300   Sub_end:SUBEND
```

MANARM

```
10     ! Instrument Basic example program: MANARM
20     ! ————————————————————————————
30     !
40     ! This program demonstrates using the instrument's
50     ! status registers to enable SRQs for event-
60     ! initiated program interrupts. In this case the
70     ! waiting_for_arm bit is detected.
80     !
90     ! ————————————————————————————
100    Sc=8
110    Addr=0
120    Device=(Sc*100)+Addr
130    ASSIGN @Agilent35670a TO Device
140    CLEAR SCREEN
150    OUTPUT @Agilent35670a;"SYST:PRES"
160    IF Sc=8 THEN
170     OUTPUT @Agilent35670a;"DISP:FORM ULOW"
180     OUTPUT @Agilent35670a;"DISP:PROG LOW"
190     CLEAR SCREEN
200    END IF
210    !
220    ! Setup registers to detect WAITING_FOR_ARM
230    !
240    ! clear any pending events
250    OUTPUT @Agilent35670a;"*CLS"
260    ! allow SRQ from operation register
270    OUTPUT @Agilent35670a;"*SRE 128"
280    ! allow SRQ from waiting_for_arm bit
290    OUTPUT @Agilent35670a;"STAT:OPER:ENAB 64"
300    ! latch waiting_for_arm TRUE
310    OUTPUT @Agilent35670a;"STAT:OPER:PTR 64"
320    ! do not latch waiting_for_arm FALSE
330    OUTPUT @Agilent35670a;"STAT:OPER:NTR 0"
340    !
350    ! set up interrupts
360    ON INTR Sc GOSUB Check_srq
370    ENABLE INTR Sc;2
380    !
390    OUTPUT @Agilent35670a;"FREQ:SPAN 100 Hz"
400    OUTPUT @Agilent35670a;"ARM:SOUR MAN"
410    OUTPUT @Agilent35670a;"ABOR;:INIT"
420    !
430    ! Wait for SRQ
440    !
450    Hang_out:GOTO Hang_out
460    !
470    Check_srq:   !
480    !
490     PRINT "SRQ Received"
500     Sb=SPOLL(Device)
```

```
510     PRINT "SPOLL(";Device;") = ";Sb
520     Queryarm(@Agilent35670a)
530     ENABLE INTR Sc
540    RETURN
550    !
560    END
570    !************************************************
580    ! Query Standard Event Status Register and arm
590    ! if waiting_for_arm event detected
600    !************************************************
610    SUB Queryarm(@Device)
620     OUTPUT @Device;"STAT:OPER:EVEN?"
630     ENTER @Device;Resp
640     PRINT "STAT:OPER:EVEN?: ";Resp
650     IF Resp=64 THEN
660      INPUT "PRESS ENTER TO ARM (ENTER 'Q' TO QUIT)",A$
670      IF UPC$(A$)="Q" THEN STOP
680      OUTPUT @Device;"ARM"
690      PRINT "ARMED!"
700      PRINT
710     END IF
720     SUBEND
```

OPC_SYNC

```
10    ! Instrument Basic program: OPCSYNC - Measurement synchronization
20    ! ————————————————————————————-
30    ! This program demonstrates how to use the *OPC command to
40    ! allow an SRQ to interrupt program execution.  *OPC will set
50    ! the OPERATION_COMPLETE bit in the EVENT STATUS register
60    ! when all pending GPIB commands have finished.  With the proper
70    ! register masks, this will generate a service request.
80    ! ————————————————————————————-
90    !
100   Scode=8                          ! Interface select code
110   Address=0
120   Agilent35670a=Scode*100+Address
130   !
140   OUTPUT Agilent35670a;"FREQ:SPAN 50 HZ" !Measurement will take 8 seconds
150   OUTPUT Agilent35670a;"*CLS"            !Clear the STATUS BYTE register
160   OUTPUT Agilent35670a;"*ESE 1"          !Program the EVENT STATUS ENABLE reg.
170   OUTPUT Agilent35670a;"*SRE 32"         !Program the STATUS BYTE ENABLE reg.
180   !
190   ON INTR Scode,2 GOTO Srq_handler  !Set up interrupt branching
200   ENABLE INTR Scode;2               !Allow SRQ to generate an interrupt
210   !
220   OUTPUT Agilent35670a;"ABORT; INIT"     !Start the measurement
230   OUTPUT Agilent35670a;"*OPC"            !Generate SRQ when all commands have
240                                    !finished.
250   Start_time=TIMEDATE
260   LOOP                             !Do something useful while waiting
270    DISP USING "14A, 2D.D";"Elapsed time :",TIMEDATE-Start_time
280    WAIT .1
290   END LOOP
300   !
310   Srq_handler:                      !Got an SRQ
320    Stb=SPOLL(Agilent35670a)              !Read STATUS BYTE and clear SRQ
330    BEEP
340    OUTPUT Agilent35670a;"*ESR?"          !Read and clear EVENT STATUS reg.
350    ENTER Agilent35670a;Esr
360    DISP "Got the SRQ! SPOLL returns:";Stb;"   ESR returns:";Esr
370   END
```

OPCQSYNC

```
10     !
20     ! Instrument Basic program: OPCQSYNC -  Measurement synchronization
30     ! ————————————————————————————————-
40     ! This program demonstrates how to use the *OPC? GPIB command
50     ! to hang the bus on a query before continuing on with the
60     ! program.  After all pending GPIB commands have finished,
70     ! the Agilent 35670a will return a ‘1’ in response to *OPC?.
80     ! ————————————————————————————————-
90     !
100    Scode=8
110    Agilent35670a=Scode*100
120    !
130    OUTPUT Agilent35670a;"SYST:PRES"        !Preset the Agilent35670a
140    OUTPUT Agilent35670a;"*OPC?"            !Pause on ENTER statement until
150    ENTER Agilent35670a;Opc                 !’*RST’ command has finished
160    !
170    OUTPUT Agilent35670a;"FREQ:SPAN 50 Hz"  !Measurement will take 8 seconds
180    DISP “Measurement started ...”
190    OUTPUT Agilent35670a;"ABOR; INIT"       !Start the measurement
200    OUTPUT Agilent35670a;"*OPC?"            !Pause until all pending GPIB
210    ENTER Agilent35670a;Opc                 !commands have finished.
220    BEEP
230    DISP “Measurement done”
240    OUTPUT Agilent35670a;"DISP:FORM ULOW"
250    OUTPUT Agilent35670a;"INIT:CONT OFF"
260    END
```

RPNCALC

```
10     !=============================================
20     !
30     ! RPNCALC
40     ! Reverse Polish Notation Waveform Calculator
50     !
60     ! This Instrument Basic program runs on the
70     ! Agilent 35670a Dynamic Signal Analyzer.
80     !
90     ! It uses data registers D1 through D4 to emulate
100    ! the stack common to most Agilent RPN calculators.
110    ! Traces from either the upper or lower trace
120    ! displays may be loaded into the X register (D1).
130    !
140    ! Unary operations (e.g., FFT and CONJ) operate
150    ! on the X register.  Binary operations (+-/*)
160    ! operate on the X and Y registers (D1 and D2).
170    ! All math operations place results in D1.
180    !
190    ! Rotate and Enter functions are available as well
200    ! as the ability to display various trace types
210    ! and to adjust trace coordinates and autoscale.
220    !
230    !=================================================
240    COM @Agilent35670a
250    DIM Trca$[20],Trcb$[20],Trc$[20]
260    ASSIGN @Agilent35670a TO 800
270    OUTPUT @Agilent35670a;"DISP:FORM ULOW"
280    OUTPUT @Agilent35670a;"INP2 ON"
290    Activetrc=1
300    OUTPUT @Agilent35670a;"CALC:ACT A"
310    Keys:!
320     ON KEY 0 LABEL "LOAD X " CALL Loadx
330     ON KEY 1 LABEL "ENTER" GOSUB Ent
340     ON KEY 2 LABEL "ROTATE     UP" GOSUB Rot_up
350     ON KEY 3 LABEL "ROTATE    DOWN" GOSUB Rot_down
360     ON KEY 4 LABEL "FUNCTIONS" CALL Functions
370     ON KEY 5 LABEL "+ - / * " CALL Ops
380     ON KEY 6 LABEL "DISPLAY  TRACE" CALL Select_trc
390     ON KEY 7 LABEL "EXIT" GOTO Exit1
400     ON KEY 8 LABEL "" GOTO Waiting
410     !
420     Waiting:GOTO Waiting !
430     !
440     Ent:      ! shifts up without replacing D1
450     Shift(1)
460    RETURN
470    !
480    !_____________________
490    Rot_up:   !shifts D1-D4 up and copies D5 to D1
500     Shift(1)
```

```
510     OUTPUT @Agilent35670a;"TRAC D1, D5"
520    RETURN
530    !————————————————————
540    Rot_down: !copies D1 to D5 and shifts D5-D2 down
550     OUTPUT @Agilent35670a;"TRAC D5, D1"
560     Shift(0)
570    RETURN
580    !
590    Exit1:!
600    END
610    !————————————————————
620    SUB Loadx
630    !————————————————————
640    ! Loads D1 register with the contents of
650    ! Trace A, Trace B, or a Constant
660    ! and shifts D2-D4 up
670    !————————————————————
680    COM @Agilent35670a
690    !
700     ON KEY 0 LABEL "TRACE A",2 GOSUB Channel1
710     ON KEY 1 LABEL "TRACE B",2 GOSUB Channel2
720     ON KEY 2 LABEL "",2 GOTO Twiddle
730     ON KEY 3 LABEL "CONSTANT",2 GOTO Constant
740     ON KEY 4 LABEL "",2 GOTO Twiddle
750     ON KEY 5 LABEL "",2 GOTO Twiddle
760     ON KEY 6 LABEL "",2 GOTO Twiddle
770     ON KEY 7 LABEL "",2 GOTO Twiddle
780     ON KEY 8 LABEL "RETURN",2 GOTO Sub_exit
790     !
800     Twiddle:GOTO Twiddle
810     !
820     Channel1:!
830      CALL Shift(1)
840      OUTPUT @Agilent35670a;"TRAC D1,TRAC1"
850      SUBEXIT
860     !
870     Channel2:!
880      CALL Shift(1)
890      OUTPUT @Agilent35670a;"TRAC D1,TRAC2"
900      SUBEXIT
910     !
920     Constant:!
930      CALL Shift(1)
940      CALL Sel_const
950     Sub_exit:!
960    SUBEND
970    !
980    SUB Shift(Direction)
990    !————————————————————
1000   ! Shifts stack (D1-D4) up or down by one
1010   ! using D5 as a temporary buffer
1020   !————————————————————
1030    COM @Agilent35670a
```

```
1040    IF Direction=1 THEN    ! shift up
1050     OUTPUT @Agilent35670a;"TRAC D5, D4"
1060     OUTPUT @Agilent35670a;"TRAC D4, D3"
1070     OUTPUT @Agilent35670a;"TRAC D3, D2"
1080     OUTPUT @Agilent35670a;"TRAC D2, D1"
1090    ELSE                   ! shift down
1100     OUTPUT @Agilent35670a;"TRAC D1, D2"
1110     OUTPUT @Agilent35670a;"TRAC D2, D3"
1120     OUTPUT @Agilent35670a;"TRAC D3, D4"
1130     OUTPUT @Agilent35670a;"TRAC D4, D5"
1140    END IF
1150   SUBEND
1160   !
1170   SUB Functions
1180   !————————————————————
1190   ! Performs immediate unary math on D1 trace
1200   !————————————————————
1210    COM @Agilent35670a
1220    ON KEY 0 LABEL "CONJ",2 GOSUB Conj
1230    ON KEY 1 LABEL "MAG",2 GOSUB Mag
1240    ON KEY 2 LABEL "REAL",2 GOSUB Realpart
1250    ON KEY 3 LABEL "IMAG",2 GOSUB Imagpart
1260    ON KEY 4 LABEL "SQRT",2 GOSUB Sqroot
1270    ON KEY 5 LABEL "FFT",2 GOSUB Fft
1280    ON KEY 6 LABEL "IFFT",2 GOSUB Ifft
1290    ON KEY 7 LABEL "PSD",2 GOSUB Psd
1300    ON KEY 8 LABEL "LN",2 GOSUB Ln
1310    ON KEY 9 LABEL "EX",2 GOSUB Expn
1320    DISP "Hit RTN for EX function"
1330    Waiting: GOTO Waiting
1340    !————————————————————
1350    Conj:!
1360     OUTPUT @Agilent35670a;"CALC:MATH:EXPR1 (CONJ(D1))"
1370     DISP ""
1380     GOTO Calc
1390    !————————————————————
1400    Mag:!
1410     OUTPUT @Agilent35670a;"CALC:MATH:EXPR1 (MAG(D1))"
1420     DISP ""
1430     GOTO Calc
1440    !————————————————————
1450    Realpart:!
1460     OUTPUT @Agilent35670a;"CALC:MATH:EXPR1 (REAL(D1))"
1470     DISP ""
1480     GOTO Calc
1490    !————————————————————
1500    Imagpart:!
1510     OUTPUT @Agilent35670a;"CALC:MATH:EXPR1 (IMAG(D1))"
1520     DISP ""
1530     GOTO Calc
1540    !————————————————————
1550    Sqroot:!
1560     OUTPUT @Agilent35670a;"CALC:MATH:EXPR1 (SQRT(D1))"
```

```
1570     DISP “”
1580     GOTO Calc
1590    !————————————————————
1600    Fft:!
1610     OUTPUT @Agilent35670a;"CALC:MATH:EXPR1 (FFT(D1))"
1620     DISP “”
1630     GOTO Calc
1640    !————————————————————
1650    Ifft:!
1660     OUTPUT @Agilent35670a;"CALC:MATH:EXPR1 (IFFT(D1))"
1670     DISP “”
1680     GOTO Calc
1690    !————————————————————
1700    Psd:!
1710     OUTPUT @Agilent35670a;"CALC:MATH:EXPR1 (PSD(D1))"
1720     DISP “”
1730     GOTO Calc
1740    !————————————————————
1750    Ln:!
1760     OUTPUT @Agilent35670a;"CALC:MATH:EXPR1 (LN(D1))"
1770     DISP “”
1780     GOTO Calc
1790    !————————————————————
1800    Expn:!
1810     OUTPUT @Agilent35670a;"CALC:MATH:EXPR1 (EXP(D1))"
1820     Calc:!  Performs calculation and stores it in D1
1830     OUTPUT @Agilent35670a;"CALC:MATH:STATE ON"
1840     OUTPUT @Agilent35670a;"TRAC D1, TRAC1"
1850     OUTPUT @Agilent35670a;"CALC:MATH:STATE OFF"
1860     DISP “”
1870   SUBEND
1880   !
1890   SUB Tracedisplay(Trace)
1900   !————————————————————
1910   ! Allows selected trace (A or B)
1920   ! to be stored to D6-D8, replaced by D1-D8,
1930   ! autoscaled, or have its trace coordinates
1940   ! Changed.
1950   ! Called by: Select_trc
1960   !————————————————————
1970   COM @Agilent35670a
1980    ON KEY 0 LABEL “”,3 GOTO Waiting
1990    ON KEY 1 LABEL “STORE IN D6",3 GOSUB D6
2000    ON KEY 2 LABEL “STORE IN D7",3 GOSUB D7
2010    ON KEY 3 LABEL “STORE IN D8",3 GOSUB D8
2020    ON KEY 4 LABEL “”,3 GOTO Waiting
2030    ON KEY 5 LABEL “RECALL   D1 - D8",3 GOSUB Disp
2040    ON KEY 6 LABEL “TRACE    COORD”,3 GOSUB Coord
2050    ON KEY 7 LABEL “AUTOSCALE”,3 GOSUB Auto
2060    ON KEY 8 LABEL “RETURN”,3 GOTO Sub_exit
2070   !
2080   Waiting:GOTO Waiting
2090   !
```

```
2100   D6:!  copy trace to D6
2110    OUTPUT @Agilent35670a;"TRAC D6, TRAC"&VAL$(Trace)
2120    RETURN
2130   D7:!  copy trace to D7
2140    OUTPUT @Agilent35670a;"TRAC D7, TRAC"&VAL$(Trace)
2150    RETURN
2160   D8:!  copy trace to D8
2170    OUTPUT @Agilent35670a;"TRAC D8, TRAC"&VAL$(Trace)
2180    RETURN
2190   Disp:! call Disp_regs
2200    Disp_regs(Trace)
2210    RETURN
2220   Coord:! call Trace_coord
2230    Trace_coord(Trace)
2240    RETURN
2250   Auto:! autoscale trace
2260    OUTPUT @Agilent35670a;"DISP:WIND"&VAL$(Trace)&":TRAC:Y:AUTO ON"
2270    RETURN
2280   Sub_exit:!
2290   SUBEND
2300   !
2310   SUB Select_trc
2320   !————————————————————
2330   ! Selects either Trace A or Trace B as active
2340   ! trace for Tracedisplay routines
2350   !————————————————————
2360    ON KEY 0 LABEL "",2 GOTO Waiting
2370    ON KEY 1 LABEL "UPPER",2 GOTO Sel_trc1
2380    ON KEY 2 LABEL "LOWER",2 GOTO Sel_trc2
2390    ON KEY 3 LABEL "",2 GOTO Waiting
2400    ON KEY 4 LABEL "",2 GOTO Waiting
2410    ON KEY 5 LABEL "",2 GOTO Waiting
2420    ON KEY 6 LABEL "",2 GOTO Waiting
2430    ON KEY 7 LABEL "",2 GOTO Waiting
2440    ON KEY 8 LABEL "RETURN",2 GOTO Sub_exit
2450    Waiting:GOTO Waiting
2460    !————————————————————
2470    Sel_trc1:!
2480     Tracedisplay(1)
2490     SUBEXIT
2500    !————————————————————
2510    Sel_trc2:!
2520     Tracedisplay(2)
2530     SUBEXIT
2540    !————————————————————
2550    Sub_exit:!
2560   SUBEND
2570   !
2580   SUB Disp_regs(Trc)
2590   !————————————————————
2600   ! Displays selected register in trace A or B
2610   ! Called by: Tracedisplay
2620   !————————————————————
```

```
2630    COM @Agilent35670a
2640    ON KEY 0 LABEL "D1",4 GOSUB D1
2650    ON KEY 1 LABEL "D2",4 GOSUB D2
2660    ON KEY 2 LABEL "D3",4 GOSUB D3
2670    ON KEY 3 LABEL "D4",4 GOSUB D4
2680    ON KEY 4 LABEL "D5",4 GOSUB D5
2690    ON KEY 5 LABEL "D6",4 GOSUB D6
2700    ON KEY 6 LABEL "D7",4 GOSUB D7
2710    ON KEY 7 LABEL "D8",4 GOSUB D8
2720    ON KEY 8 LABEL "RETURN",4 GOTO Sub_exit
2730    !
2740    Waiting:GOTO Waiting
2750    !
2760    D1:!
2770     OUTPUT @Agilent35670a;"CALC"&VAL$(Trc)&":FEED 'D1'; *WAI"
2780     RETURN
2790    !
2800    D2:!
2810     OUTPUT @Agilent35670a;"CALC"&VAL$(Trc)&":FEED 'D2'; *WAI"
2820     RETURN
2830    !
2840    D3:!
2850     OUTPUT @Agilent35670a;"CALC"&VAL$(Trc)&":FEED 'D3'; *WAI"
2860     RETURN
2870    !
2880    D4:!
2890     OUTPUT @Agilent35670a;"CALC"&VAL$(Trc)&":FEED 'D4'; *WAI"
2900     RETURN
2910    !
2920    D5:!
2930     OUTPUT @Agilent35670a;"CALC"&VAL$(Trc)&":FEED 'D5'; *WAI"
2940     RETURN
2950    !
2960    D6:!
2970     OUTPUT @Agilent35670a;"CALC"&VAL$(Trc)&":FEED 'D6'; *WAI"
2980     RETURN
2990    !
3000    D7:!
3010     OUTPUT @Agilent35670a;"CALC"&VAL$(Trc)&":FEED 'D7'; *WAI"
3020     RETURN
3030    !
3040    D8:!
3050     OUTPUT @Agilent35670a;"CALC"&VAL$(Trc)&":FEED 'D8'; *WAI"
3060     RETURN
3070    !
3080    Sub_exit:!
3090   SUBEND
3100   !
3110   SUB Trace_coord(Trace)
3120   !————————————————————
3130   ! Selects trace coordinates for trace
3140   ! Called by: Tracedisplay
3150   !————————————————————
```

```
3160    COM @Agilent35670a
3170    T$=VAL$(Trace)
3180    !
3190    ON KEY 0 LABEL "LINEAR   MAGNITUDE",4 GOSUB Lin
3200    ON KEY 1 LABEL "LOG      MAGNITUDE",4 GOSUB Log
3210    ON KEY 2 LABEL "DB       MAGNITUDE",4 GOSUB Db
3220    ON KEY 3 LABEL "PHASE",4 GOSUB Phase
3230    ON KEY 4 LABEL "UNWRAPPEDPHASE",4 GOSUB Unwrapped
3240    ON KEY 5 LABEL "REAL     PART",4 GOSUB Realpart
3250    ON KEY 6 LABEL "IMAGINARYPART",4 GOSUB Imagpart
3260    ON KEY 7 LABEL "NYQUIST  DIAGRAM",4 GOSUB Nyquist
3270    ON KEY 8 LABEL "RETURN",4 GOTO Sub_exit
3280    !
3290    Lin:!
3300     OUTPUT @Agilent35670a;"CALC"&T$&":FORM MLIN; *WAI"
3310     OUTPUT @Agilent35670a;"DISP:WIND"&T$&":TRAC:Y:SPAC LIN; *WAI"
3320     RETURN
3330    !
3340    Log:!
3350     OUTPUT @Agilent35670a;"CALC"&T$&":FORM MLIN"
3360     OUTPUT @Agilent35670a;"DISP:WIND"&T$&":TRAC:Y:SPAC LOG; *WAI"
3370     RETURN
3380    !
3390    Db:!
3400     OUTPUT @Agilent35670a;"CALC"&T$&":FORM MLOG; *WAI"
3410     RETURN
3420    !
3430    Phase:!
3440     OUTPUT @Agilent35670a;"CALC"&T$&":FORM PHAS; *WAI"
3450     RETURN
3460    !
3470    Unwrapped:!
3480     OUTPUT @Agilent35670a;"CALC"&T$&":FORM UPH; *WAI"
3490     RETURN
3500    !
3510    Realpart:!
3520     OUTPUT @Agilent35670a;"CALC"&T$&":FORM REAL; *WAI"
3530     RETURN
3540    !
3550    Imagpart:!
3560     OUTPUT @Agilent35670a;"CALC"&T$&":FORM IMAG; *WAI"
3570     RETURN
3580    !
3590    Nyquist:!
3600     OUTPUT @Agilent35670a;"CALC"&T$&":FORM NYQ; *WAI"
3610     RETURN
3620    !
3630    Sub_exit:!
3640   SUBEND
3650   !


3660   SUB Ops
3670   !————————————————————
```

```
3680   ! Performs binary operations on D1 and D2
3690   !————————————————————
3700    COM @Agilent35670a
3710    ON KEY 0 LABEL "   +",4 GOSUB Plus
3720    ON KEY 1 LABEL "   -",4 GOSUB Minus
3730    ON KEY 2 LABEL "   /",4 GOSUB Divide
3740    ON KEY 3 LABEL "   *",4 GOSUB Mult
3750    FOR I=4 TO 8
3760     ON KEY I LABEL "",4 GOTO Waiting
3770    NEXT I
3780    !
3790    Waiting:GOTO Waiting
3800    !
3810    Plus:!
3820     OUTPUT @Agilent35670a;"CALC:MATH:EXPR1 (D1 + D2)"
3830     GOTO Calc
3840    !
3850    Minus:!
3860     OUTPUT @Agilent35670a;"CALC:MATH:EXPR1 (D2 - D1)"
3870     GOTO Calc
3880    !
3890    Divide:!
3900     OUTPUT @Agilent35670a;"CALC:MATH:EXPR1 (D2 / D1)"
3910     GOTO Calc
3920    !
3930    Mult:!
3940     OUTPUT @Agilent35670a;"CALC:MATH:EXPR1 (D1 * D2)"
3950     GOTO Calc
3960    !
3970    Calc:!  perform calculation and place in D1
3980     OUTPUT @Agilent35670a;"CALC:MATH:STATE ON"
3990     OUTPUT @Agilent35670a;"TRAC D1, TRAC1"
4000     OUTPUT @Agilent35670a;"CALC:MATH:STATE OFF"
4010   SUBEND
4020   !
4030   SUB Sel_const
4040   !————————————————————
4050   ! Allows a constant to be loaded into D1
4060   ! Called by: Loadx
4070   !————————————————————
4080   COM @Agilent35670a
4090    ON KEY 0 LABEL "K1",4 GOTO K1
4100    ON KEY 1 LABEL "K2",4 GOTO K2
4110    ON KEY 2 LABEL "K3",4 GOTO K3
4120    ON KEY 3 LABEL "K4",4 GOTO K4
4130    ON KEY 4 LABEL "K5",4 GOTO K5
4140    ON KEY 5 LABEL "",4 GOTO Waiting
4150    ON KEY 6 LABEL "ENTER    CONSTANT",4 GOTO Enter_const
4160    ON KEY 7 LABEL "",4 GOTO Waiting
4170    ON KEY 8 LABEL "",4 GOTO Waiting
4180    Waiting:GOTO Waiting
```

```
4190    !
4200    K1:!
4210     OUTPUT @Agilent35670a;"CALC:MATH:EXPR1 (K1)"
4220     GOTO Calc
4230    !
4240    K2:!
4250     OUTPUT @Agilent35670a;"CALC:MATH:EXPR1 (K2)"
4260     GOTO Calc
4270    !
4280    K3:!
4290     OUTPUT @Agilent35670a;"CALC:MATH:EXPR1 (K3)"
4300     GOTO Calc
4310    !
4320    K4:!
4330     OUTPUT @Agilent35670a;"CALC:MATH:EXPR1 (K4)"
4340     GOTO Calc
4350    !
4360    K5:!
4370     OUTPUT @Agilent35670a;"CALC:MATH:EXPR1 (K5)"
4380     GOTO Calc
4390    !
4400    Enter_const:! enter constant value into K1
4410     INPUT "Enter constant value",C$
4420     OUTPUT @Agilent35670a;"CALC:MATH:CONS1 "&C$
4430     OUTPUT @Agilent35670a;"CALC:MATH:EXPR1 (K1)"
4440    !
4450    Calc:!
4460     OUTPUT @Agilent35670a;"CALC:MATH:STATE ON"
4470     ! constant stored in trace1; now copy to D1
4480     OUTPUT @Agilent35670a;"TRAC D1, TRAC1"
4490     OUTPUT @Agilent35670a;"CALC:MATH:STATE OFF"
4500   SUBEND
```

TWO_CTLR

```
10     ! BASIC program:  TWO_CTLR - Two controller operation
20     !-----------------------------------
30     !This program demonstrates how an external controller
40     !and Instrument Basic can work together.  This program
50     !will download a BASIC program to the Agilent 35670A and run it two
60     !times.  After each run, two BASIC program variables will
70     !will be read from the Agilent 35670A and displayed.
80     !-----------------------------------
90     !
100    Scode=7                          !Select code for interface
110    Address=11                       !Address for Agilent 35670A
120    Agilent35670a=Scode*100+Address
130    !
140    CLEAR Agilent35670a
150    OUTPUT Agilent35670a;"PROG:DEL:ALL"    !Scratch the program space
160    !
170    DISP "Downloading the program..."
180    ASSIGN @Prog TO Agilent35670a;EOL CHR$(10) !Change EOL character
190    OUTPUT @Prog;"PROG:DEF #0";           !Send program
200    OUTPUT @Prog;"10 COM INTEGER Times_run,Test$[10]"
210    OUTPUT @Prog;"20 Times_run=Times_run +1"
220    OUTPUT @Prog;"30 IF Times_run=1 THEN Test$=""PASS"""
230    OUTPUT @Prog;"40 IF Times_run=2 THEN Test$=""FAIL"""
240    OUTPUT @Prog;"50 BEEP"
250    OUTPUT @Prog;"60 END"
260    OUTPUT @Prog;CHR$(10) END         !Terminate the data block
270    !
280    !Set up registers for interrupt on PROGRAM_RUNNING going false
290    OUTPUT Agilent35670a;"*CLS"            !Clear the STATUS register
300    !Program NTR reg and OPERATION ENABLE reg for PROGRAM_RUNNING bit
310    OUTPUT Agilent35670a;"STAT:OPER:NTR 16384"
320    OUTPUT Agilent35670a;"STAT:OPER:ENAB 16384"
330    OUTPUT Agilent35670a;"*SRE 128"    !Allow SRQ on bit 7 of STATUS reg
340    !
350    DISP "Running the program..."
360    OUTPUT Agilent35670a;"PROG:STAT RUN"  !Run Program
370    Display_res(Agilent35670a,Scode)      !Read and display variables
380    OUTPUT Agilent35670a;"PROG:STAT RUN"  !Run Program  again
390    Display_res(Agilent35670a,Scode)      !Read and display variables
400    !
410    END                             !End of this program
420    !
430    SUB Display_res(Agilent35670a,Scode)
440    ! This subprogram waits for an SRQ interrupt to signal that a
450    ! BASIC program has finished.  It then clears the GPIB registers
460    ! by reading them.  Once that is done, the values of two IBASIC
470    ! variables are read and displayed.
480    !
490     ON INTR Scode GOTO Read_results     !Set up interrupt branching
500     ENABLE INTR Scode;2                 !Allow interrupt on SRQ
```

```
510     Idle:GOTO Idle
520     !
530     Read_results:                           !Program has finished
540     A=SPOLL(Agilent35670a)                   !Read and clear the SRQ
550     OUTPUT Agilent35670a;"STAT:OPER?" !Read and clear OPERATION STATUS reg.
560     ENTER Agilent35670a;Event
570     WAIT .5
580     !
590     OUTPUT Agilent35670a;"FORM:DATA ASCII,3"
600     OUTPUT Agilent35670a;"PROG:NUMB? 'Times_run'"!Read the first variable
610     ENTER Agilent35670a;Times_run
620     !
630     OUTPUT Agilent35670a;"PROG:STR? 'Test$'"     !Read the second variable
640     ENTER Agilent35670a;Test$
650     !
660     PRINT "Times_run: ";Times_run,"Test$: ";Test$
670    SUBEND
```

WAI_SYNC

```
10     ! Instrument Basic program: WAI_SYNC - Measurement synchronization
20     ! ————————————————————————————
30     ! This program demonstrates how to use the *WAI command to
40     ! prevent execution of an GPIB command until all previous
50     ! commands have finished.  In this example, the trace display
60     ! measurement has finished.
70     ! The *WAI command does not affect program operation. The
80     ! program will run to completion, sending all of the commands to
90     ! to the Agilent35670A without waiting for them to be executed.
100    ! ————————————————————————————-
110    Scode=8                          !Interface select code
120    Address=0
130    Agilent35670a=Scode*100+Address
140    !
150    DISP "Sending GPIB commands..."
160    OUTPUT Agilent35670a;"SYST:PRES"
170    OUTPUT Agilent35670a;"AVER:COUN 1"
180    OUTPUT Agilent35670a;"AVER ON"
190    OUTPUT Agilent35670a;"FREQ:SPAN 50 HZ"!Set narrow span
200    OUTPUT Agilent35670a;"ABORT; INIT"    !Start the measurement
210    OUTPUT Agilent35670a;"*WAI"           !Tell analyzer to wait here until
220                                !all GPIB commands have finished
230    OUTPUT Agilent35670a;"DISP:FORM ULOW" !Go to upper/lower after waiting
240    BEEP
250    DISP "Finished. Display will go to UPPER/LOWER when meas. Done"
260    END
```

12

Instrument-Specific Instrument Basic Features

12

Instrument-Specific Instrument Basic Features

Introduction

The Instrument Basic Users Handbook that accompanies this manual is divided into the following sections:

- "Instrument Basic Programming Techniques"
- "Instrument Basic Interfacing Techniques"
- "Instrument Basic Language Reference"

The Instrument Basic Users Handbook is included with all Agilent Technologies instruments that use Instrument Basic. Since each instrument is different, the way that Instrument Basic interfaces and interacts with its host often changes from one instrument to another.

For example, some instruments employ editors, while others do not, and front panel interfaces often vary a great deal from one instrument to another. For this reason, many parts of the Instrument Basic Users Handbook are either generic in nature, or apply to only one of many possible instrument interfaces.

This chapter describes how to use the Instrument Basic Users Handbook for the Agilent 35670A. Global exceptions apply throughout the handbook. These differences are discussed by category. Specific differences for each command are listed in table 12-4.

Supported Interfaces

The Instrument BASIC Users Handbook refers to various interfaces, particularly in chapter 2 of " Instrument BASIC Interfacing Techniques." Instrument BASIC in the Agilent 35670A supports the following interfaces and the select codes that provide access to them:

- the analyzer's display (select code 1)
- the keyboard (select code 2)
- the external bus (select code 7)
- the internal bus (select code 8)
- the RS-232-C port (select code 9)
- the parallel port (select code 26)

Instrument BASIC in the Agilent 35670A does not support the GPIO interface (select code 12).

Display and Keyboard Interfaces

The following section describes the differences between the standard interface assumed by the Instrument Basic Users Handbook and the Agilent 35670A's display and keyboard interface. In addition, the Agilent 35670A's keys that emulate command line execution of Instrument Basic keywords are listed.

Display Differences

Most references to the display (CRT) in the Instrument Basic Users Handbook assume a standard 80 column terminal. The Agilent 35670A has a 58 column display for text. This affects references to the width of the default PRINTER IS device (the display) in the LIST, PRINT and PRINTER IS commands.

You must allocate a display partition to view output to the display because the instrument shares the display with Instrument BASIC. This affects both the text commands listed above, as well as the graphics commands, MOVE and DRAW. Three different display partitions, UPPER, LOWER or FULL, may be allocated. The text width for all three is the same. The only change for the text commands is how much text is displayed at one time.

In the Agilent 35670A, PEN 0 does not perform a pixel complement function. Instead, it causes subsequent graphics statements to erase those portions of graphic elements that lie along the drawing path.

Keyboard Differences

The Instrument Basic Users Handbook assumes the use of a standard BASIC Series 200 workstation keyboard. It also assumes that Instrument Basic works in "command line execution mode," where individual commands may be entered and executed from the keyboard.

Instrument Basic in the Agilent 35670A works with an Agilent approved PC keyboard and does not use command line execution mode. When using the Instrument Basic editor, the Agilent 35670A's front-panel hardkeys become alpha keys. Softkey menus emulate many of the keywords that are executable from the command line on an BASIC workstation (such as RUN, CONTINUE, and SCRATCH).

The following keypath of Agilent 35670A hardkeys and softkeys correspond to Instrument BASIC keywords:

CONTINUE

[**BASIC**] [CONTINUE]

[**BASIC**] [INSTRUMNT BASIC] [DEBUG] [CONTINUE]

DEL

[**BASIC**] [INSTRUMNT BASIC] [EDIT] [DELETE LINE]

EDIT

[**BASIC**] [INSTRUMNT BASIC] [EDIT]

LIST

[**BASIC**] [INSTRUMNT BASIC] [PRINT PROGRAM]

PAUSE

[**BASIC**]

RUN

[**BASIC**] [RUN PROGRAM1] . . . [RUN PROGRAM5]

[**BASIC**] [INSTRUMNT BASIC] [RUN PROGRAM]

[**BASIC**] [INSTRUMNT BASIC] [DEBUG] [RUN]

SCRATCH

[**BASIC**] [INSTRUMNT BASIC] [UTILITIES] [SCRATCH]

SECURE

[**BASIC**] [INSTRUMNT BASIC] [UTILITIES] [SECURE]

STOP

[**Local/GPIB**]

REN

[**BASIC**] [INSTRUMNT BASIC] [UTILITIES] [RENUMBER]

The following Instrument Basic keywords are described in the "Language Reference" section of the Instrument Basic Users Handbook in terms of a standard workstation keyboard. Their use with the Agilent 35670A is described below:

EDIT

Ignore all documentation in the "Instrument Basic Language Reference" for the EDIT command. See chapter 5, "Developing Programs," for information on using the Instrument Basic editor in the Agilent 35670A.

ON KEY and OFF KEY

Nine softkeys are available for use in the Agilent 35670A. They appear on the right side of the display where instrument softkeys normally appear. Key selector values range from 0 through 8. In addition, the softkeys are loaded into the function keys ([F1] - [F9]) with the keyboard.

INPUT

When an INPUT statement is encountered in an Instrument Basic program, the alpha entry menu appears on the display. You can use your keyboard, or the front panel alpha keys, the numeric keypad and the symbol softkeys to enter a response.

To enter an input response press the [ENTER] softkey in the alpha entry menu or the [Enter] key on the keyboard. Disregard all keys mentioned in the "Instrument Basic Language Reference" description of this key.

You have two options for terminating an INPUT command:

Press [ENTER] (either the softkey or the [Enter] key on the keyboard)
Press the [PAUSE] softkey ([F9] on the keyboard).

To continue the program after pressing the [PAUSE] softkey, press the [CONTINUE] softkey in the [**BASIC**] menu or the [SINGLE STEP] softkey in the [DEBUG] menu. The INPUT statement is re-executed.

You cannot press the [**BASIC**] hardkey to pause the program nor the [**Local/GPIB**] hardkey to stop the program because these are redefined as alpha keys whenever the alpha entry menu appears.

To enter an input response while a program is under remote control—that is, an external controller is executing the program—the program must be returned to local (front panel) control. Press the [**Local/GPIB**] hardkey to return the program to local control. Enter the input response as described above. The instrument remains in local control after terminating the input command. Pressing the [**Local/GPIB**] hardkey again, resets the program. It is recommended that you specify the exact key sequence expected in your input prompt.

ENTER

For a description of using ENTER with a keyboard (ENTER KBD) see the previous description for the INPUT statement.

Disk I/O

The following section specifies the Agilent 35670A's implementation of disk I/O functions.

Disk I/O Commands

Many Instrument Basic commands that pertain to the disk I/O (SAVE, RE-SAVE, COPY, MSI, etc.) have similar functions executed by normal Agilent 35670A front-panel operations. These front-panel operations are not considered to be Instrument Basic functions.

For example, the MASS STORAGE IS command when executed in a program is totally independent of the current storage device found under the [**Save/Recall**] [DEFAULT DISK] key. Conversely, using the [DEFAULT DISK] key to change the default storage device has no impact on any MSI statements within an Instrument Basic program.

Volume Specifiers

Instrument Basic in the Agilent 35670A supports four mass storage devices; the internal disk drive, volatile RAM disk (memory unit 0) and non-volatile RAM disk (memory unit 1) and an external disk drive (Agilent Technologies Subset/80). This affects the volume specifier parameter in the following commands:

- ASSIGN
- CAT
- COPY
- CREATE
- CREATE ASCII
- CREATE BDAT
- CREATE DIR
- GET
- INITIALIZE
- MASS STORAGE IS
- PRINTER IS
- PURGE
- RENAME
- RE-SAVE
- SAVE

Valid volume specifiers for each mass storage device are shown in table 12-1.

Table 12-1. Mass Storage Volume Specifiers

Disk	Instrument Basic Volume Specifier	Equivalent GPIB MSI Specifier
External Disk (connected via GPIB)	:EXTERNAL,7xx,uu	EXT,7xx,uu:
Front-Panel Disk (3.5 inch floppy)	:INTERNAL,4	INT:
Volatile RAM Disk	:MEMORY,0,0	RAM:
Nonvolatile RAM Disk	:MEMORY,0,1	NVRAM:

xx GPIB address
uu unit number

Disk Format

Instrument Basic in the Agilent 35670A recognizes two types of disk and file formats; LIF (Logical Interface Format) and DOS (Disk Operating System). Although the Agilent 35670A recognizes DOS format and directories, it does not support HFS (Hierarchical File System). The Instrument Basic Users Handbook addresses LIF, DOS, and HFS file systems. In general, disregard all references to HFS throughout the Instrument Basic Users Handbook.

File Types

Instrument Basic can create three types of files: ASCII, BDAT, and untyped. These files can exist on either a DOS or a LIF formatted disk.

If you catalog a DOS disk, these three types show up as "ASCII," "BDAT," and "DOS." If you catalog a LIF disk, these three types show up as "ASCII," "BDAT," and "HP-UX." Table 12-2 indicates these configurations.

Table 12-2. Instrument Basic File Types

File Type	Appears on a LIF disk as	Appears on a DOS disk as
ASCII	ASCII	ASCII
BDAT	BDAT	BDAT
untyped	HP-UX	DOS

Instrument Basic supports LIF protect codes only on BDAT files. An error is generated if a LIF protect code is encountered on an ASCII file. Instrument Basic ignores a LIF protect code on an untyped file.

A special note about file types and file systems:

The "Instrument Basic Language Reference" sometimes uses the terms HP-UX file and HFS interchangeably or refers to HP-UX files only in context of HFS volumes. In fact, HP-UX files can exist on a LIF volume, which the Agilent 35670A supports. Be careful when reading the descriptions in the "Instrument Basic Language Reference." The Agilent 35670A supports HP-UX files on LIF volumes only. The Agilent 35670A does not support HP-UX files on HFS volumes.

The ASCII file type described in this section is specific to Agilent's LIF file system and is not the same as the standard "ASCII" file type in a DOS environment. If you copy an ASCII file from a LIF disk to a DOS disk, the file appears as an "ASCII" file. However, the file is not usable with DOS-system editors. Untyped files are the only files you can edit with a DOS ASCII editor on a PC.

An untyped file is automatically generated whenever an Instrument Basic program is SAVEd from the Agilent 35670A to a DOS-formatted disk. A RE-SAVE maintains the original file type if a file exists, otherwise it performs the same action as SAVE.

Formatting Disks

Formatting a disk prepares it for use. The Agilent 35670A recognizes both LIF and DOS formats and has the capability to format either type in the [FORMAT DISK] menu.

You can also use the INITIALIZE statement to format a disk.

Caution Existing files on the media are destroyed when formatting or executing the INITIALIZE command.

The [FORMAT DISK] Menu

The [FORMAT DISK] menu under the [**Disk Utility**] hardkey allows you to define format parameters and to format a disk using these parameters.

The [DISK TYPE LIF **DOS**] softkey allows you to select the type of format. The default type is DOS. Press this key to toggle the selection to LIF.

The [RAM DISK SIZE] softkey specifies the storage capacity of the Volatile RAM disk. The default value is 64 KBytes of storage. Use the numeric keypad or the arrow keys to enter a new value. Values are rounded up to the next highest KByte (1024 byte) increment. The entry window appears at the top of the display when you press the softkey.

Note You can use the [FORMAT DISK] menu to format non-volatile RAM (NVRAM), the internal disk drive or the external disk drive. However, the [RAM DISK SIZE] specification is ignored.

The [INTRLEAVE FACTOR] softkey defines the ordering of the sectors on the 3.5 inch flexible disks. To specify the default value for the disk, specify 0. This value is ignored when formatting the non-volatile RAM (NVRAM).

Press [PERFORM FORMAT] to start the format process. An entry window at the top of the screen displays the specifier for the default disk. See table 12-1 for the MSI specifiers. Use the alpha entry keys or the keyboard to modify the field. The current values for [DISK TYPE], [RAM DISK SIZE] and [INTRLEAVE FACTOR] are used.

The INITIALIZE Statement

The disk format, LIF or DOS, is specified when the media is initialized. The INITIALIZE statement takes the following form:

```
INITIALIZE "<disk format>: <volume specifier>" <interleave factor>,
           <format option>
```

If <disk format> is not specified, the default format is LIF.

The <volume specifier> for the volatile RAM disk includes a <unit size> parameter,that specifies the number of 256-byte sectors. The actual size is memory dependent and ranges from 4 thru 32767.

The <format option> parameter specifies the capacity of the flexible disk drive (internal or external). See table 12-3 for the valid format options.

Table 12-3. Flexible Disk Format Options

Media	Format Option	Bytes/Sector	Sectors/Track	Tracks/Surface	Maximum Capacity (bytes)
1-MByte	0	256	16	77	630,784
	1*	256	16	77	630,784
	2	512	9	77	709,632
	3	1,024	5	77	788,480
	4**	256	16	77	270,336
	16	512	9	80	737,280
2-MByte	0	256	32	77	1,261,568
	1***	256	32	77	1,261,568
	2	512	18	77	1,419,264
	3	1,024	10	77	1,576,960
	4***	256	32	77	1,261,568
	16	512	18	80	1,474,560

*Same as Format Option 0 (default) when using 1-MByte media.
**Format Option 4 (singled sided disk format) is not supported in internal disk drive (INT:).
***Same as Format Option 0 (default) when using 2-MByte media.

Note Table 12-3 specifies the maximum capacity for each format option. Actual capacity is dependent upon the file system type, LIF or DOS.

For example, to INITIALIZE a flexible disk in the internal disk drive in LIF format, use the following:

```
INITIALIZE "LIF:,4,0",1
```

To INITIALIZE a DOS disk in an external disk drive, use the following:

```
INITIALIZE "DOS:,700,0",1,16
```

In this example, the format option, 16, is important when initializing a DOS disk. An incorrect format option results in a disk that the Agilent 35670A can use, but other DOS systems cannot use. This potential problem can be avoided by formatting the disk on a DOS system, rather than the Agilent 35670A.

Caution An incorrect format option may prevent other DOS systems from using the DOS disk.

You can use the INITIALIZE statement to format the non-volatile disk (NVRAM). However, the size of NVRAM is fixed and the interleave factor is ignored. You can only change the file format.

Once initialization is complete, file format, "LIF" or "DOS", is not specified in any other file operations. Instrument Basic automatically determines the format of the disk.

Miscellaneous Command Differences

COS

The range of the COS command is all absolute values less than 1.7083127722 e+10 degrees.

SYSTEM$

The Agilent 35670A does not support the topic specifier, (SYSTEM VERSION:).

Note Since the Instrument Basic Users Handbook is continually revised to support all implementations of Instrument Basic, there may be other commands that appear in that documentation that are not supported in the Agilent 35670A. Table 12-4 in the following section lists all Instrument Basic keywords supported by Instrument Basic in the Agilent 35670A.

Keyword Exceptions

Table 12-4 summarizes the Instrument Basic keyword implementation in the Agilent 35670A. The table indicates if the keyword has front panel support. (If it does, the key path is given.) Table 12-4 also lists the major differences between the descriptions of these keywords in the "Instrument Basic Language Reference" and the way they are implemented in the Agilent 35670A. Where differences are too extensive to be summarized, references to their explanation in the "Global Exceptions" section are given.

Any keywords or functions found in the "Instrument Basic Language Reference" that do not appear in this table, do not apply to Instrument Basic in the Agilent 35670A and should be ignored.

Table 12-4. Agilent 35670A Keyword Implementation

Command	Front Panel Support	Exceptions
ABORT	None	Interface Select Code = 7 or 8
ABS	None	None
ACS	None	None
ALLOCATE	None	None
ALPHA ON | OFF	[BASIC] [DISPLAY SETUP] [ALPHA ON OFF]	None
AND	None	None
AREA	None	Not supported
ASN	None	None
ASSIGN	None	One GPIB device per ASSIGN statement LIF protect code supported in BDAT files only Does not support HFS volumes See "Disk I/O"
ATN	None	None
AXES	None	None
BASE	None	None
BEEP	None	None
BINAND	None	None
BINCMP	None	None
BINEOR	None	None
BINIOR	None	None
BIT	None	None
CALL	None	None

Table 12-4. Agilent 35670A Keyword Implementation (Continued)

Command	Front Panel Support	Exceptions
CAT	None (Independent of [**Disk Utility**] and [**Save/Recall**] functions)	Does not support HFS catalogs See "Volume Specifiers" in "Disk I/O
CAUSE ERROR	None	None
CHR$	None	None
CLEAR	None	None
CLEAR ERROR	None	None
CLEAR SCREEN	None	None
CLIP	None	None
COM	None	None
CONT	[**BASIC**] [CONTINUE] or [**BASIC**] [DEBUG] [CONTINUE]	No line number or label support
CONTROL	None	Not supported
COPY	None (Independent of[**Disk Utility**] functions)	LIF protect code in BDAT files only Does not support HFS volumes See "Volume Specifiers" in "Disk I/O
COPYLINES	None	Not supported
COS	None	Absolute range values less than 1.7083127722 e+10
CREATE	None	Does not support HFS volumesSee "Disk I/O"
CREATE ASCII	None	Does not support HFS volumes See "Disk I/O"
CREATE BDAT	None	LIF protect code allowed Does not support HFS volumes See "Disk I/O"
CREATE DIR	None	Does not support HFS volumesSee "Volume Specifiers" in "Disk I/O
CRT	None	ENTER CRT (ENTER 1) not supported
CSIZE	None	None
DATA	None	None
DATE	None	None
DATE$	None	None
DEALLOCATE	None	None
DEF FN	None	None
DEG	None	None

Table 12-4. Agilent 35670A Keyword Implementation (Continued)

Command	Front Panel Support	Exceptions
DEL	[**BASIC**] [EDIT][DELETE LINE]	Deletes only the current line
DELSUB	[BASIC] [INSTRUMENT BASIC] [UTILITIES] [DELSUB]	None
DET	None	None
DIM	None	None
DISABLE	None	None
DISABLE INTR	None	Interface Select Code = 7 or 8
DISP	None	None
DIV	None	None
DOT	None	None
DRAW	None	Maximum x,y coordinates: Full partition (474,345) Upper partition (474,171) Lower partition (474,171)
DROUND	None	None
DUMP	None	Not supported
DVAL	None	None
DVAL$	None	None
EDIT	[**BASIC**][EDIT]	Editing functions described in chapter 5.
ENABLE	None	None
ENABLE INTR	None	Interface Select Code = 7 or 8 Must not precede an ON INTR statement.
END	None	None
END IF	None	None
END LOOP	None	None
END SELECT	None	None
END WHILE	None	None
ENTER	None	Select Code = 2, 7, 8, 9, or 26 only
ERRL	None	None
ERRLN	None	None
ERRM$	None	None
ERRN	None	None

Table 12-4. Agilent 35670A Keyword Implementation (Continued)

Command	Front Panel Support	Exceptions
EXIT IF	None	None
EXOR	None	None
EXP	None	None
FN	None	None
FNEND	None	None
FOR...NEXT	None	None
FRACT	None	None
FRAME	None	None
GCLEAR	None	None
GESCAPE	None	None
GET	None (Independent of [**Save/Recall**] functions)	Does not support HFS volumesSee “Volume Specifiers” in “Disk I/O
GINIT	None	None
GLOAD	None	None
GOSUB	None	None
GOTO	None	None
GRAPHICS ON | OFF	[BASIC] [DISPLAY SETUP] [GRAPHICS ON OFF]	None
GRID	None	None
GSTORE	None	None
IDRAW	None	None
IF...THEN	None	None
IMAGE	None	None
IMOVE	None	None
INDENT	None	None
INITIALIZE	None (Independent of [**Disk Utility**] functions)	Does not support HFS volumesSee “Disk I/O”
INPUT	None	See INPUT command in “Keyboard Differences” section
INT	None	None
INTEGER	None	None

Table 12-4. Agilent 35670A Keyword Implementation (Continued)

Command	Front Panel Support	Exceptions
IPLOT	None	None
IVAL	None	None
IVAL$	None	None
KBD	None	External Keyboard (2) or front-panel alpha keys
LABEL	None	None
LDIR	None	None
LEN	None	None
LET	None	None
LGT	None	None
LINE TYPE	None	Line types are different from those listed in the Language Reference
LIST	[**BASIC**][PRINT PROGRAM]	Default width = 58 (see PRINTER IS)
LOAD	None	None
LOADSUB	None	None
LOCAL	None	None
LOCAL LOCKOUT	None	Interface Select Code = 7 only
LOG	None	None
LOOP	None	None
LORG	None	None
LWC$	None	None
MASS STORAGE IS	None (Independent of [**Save/Recall**] and **Disk Utility**] functions)	Does not support HFS volumesSee “Disk I/O”External disks must be online
MAT	None	None
MAX	None	None
MAXREAL	None	None
MERGE ALPHA WITH GRAPHICS	None	Not supported
MIN	None	None
MINREAL	None	None
MOD	None	None

Table 12-4. Agilent 35670A Keyword Implementation (Continued)

Command	Front Panel Support	Exceptions
MODULO	None	None
MOVE	None	Maximum x, y coordinates: Full partition (474,345) Upper partition (474,171) Lower partition (474,171)
MOVELINES	None	Not supported
NOT	None	None
NUM	None	None
OFF CYCLE	None	None
OFF ERROR	None	None
OFF INTR	None	Interface Select Code = 7 or 8 Must precede ENABLE INTR statement.
OFF KEY	None	Key selectors are 0 thru 9
OFF TIMEOUT	None	Interface Select Code = 7, 8 or 9
ON	None	None
ON CYCLE	None	None
ON ERROR	None	None
ON INTR	None	Interface Select Code = 7 or 8
ON KEY	None	Key selectors are 0 thru 9
ON TIMEOUT	None	Interface Select Code = 7, 8 or 9
OPTION BASE	None	None
OR	None	None
OUTPUT	None	Select Code = 1, 7, 8, 9, or 26 only
PASS CONTROL	None	Interface Select Code 8 (pass control of external bus to analyzer)
PAUSE	None	None
PDIR	None	None
PEN	None	0 = erase nonzero = draw
PENUP	None	None
PI	None	None
PIVOT	None	None
PLOT	None	None

Table 12-4. Agilent 35670A Keyword Implementation (Continued)

Command	Front Panel Support	Exceptions
PLOTTER IS	None	Not supported
POLYGON	None	Solid fill only
POLYLINE	None	None
POS	None	None
PRINT	None	PRINTER IS default width = 58
PRINTER IS	None	default width = 58 LIF protect code in BDAT files only Does not support HFS volumes See "Volume Specifiers" in "Disk I/O"
PROUND	None	None
PRT	None	None
PURGE	None (Independent of [**Disk Utility**] functions)	LIF protect code in BDAT files only Does not support HFS volumes See "Volume Specifiers" in "Disk I/O"
RAD	None	None
RANDOMIZE	None	None
RANK	None	None
RATIO	None	None
READ	None	None
REAL	None	None
RECTANGLE	None	Solid fill only
REDIM	None	None
REM	None	None
REMOTE	None	Does not support Interface Select Code 8, 9, or 26
REN	[**BASIC**] [UTILITIES] [RENUMBER]	No line label support
RENAME	None (Independent of[**Disk Utility**] functions)	LIF protect code in BDAT files only Does not support HFS volumes See "Volume Specifiers" in "Disk I/O"
REPEAT...UNTIL	None	None
RE-SAVE	None (Independent of [**Save/Recall**] functions)	Does not support HFS volumes See "Disk I/O"
RESTORE	None	None

Table 12-4. Agilent 35670A Keyword Implementation (Continued)

Command	Front Panel Support	Exceptions
RE-STORE	None	None
RETURN	None	None
RETURN . . .	None	None
REV$	None	None
RND	None	None
ROTATE	None	None
RPLOT	None	None
RPT$	None	None
RUN	[**BASIC**] [RUN] or [**BASIC**] [DEBUG] [RUN]	None
SAVE	None (Independent of [**Save/Recall**] functions)	Does not support HFS volumes See "Disk I/O"
SCRATCH	[**BASIC**] [UTILITIES] [SCRATCH]	Does not support HFS volumes
SECURE	[**BASIC**] [UTILITIES] [SECURE]	None
SELECT...CASE	None	None
SEPARATE ALPHA	None	Not supported
SET PEN	None	Not supported
SET TIME	None	None
SET TIMEDATE	None	None
SGN	None	None
SHIFT	None	None
SHOW	None	None
SIN	None	None
SIZE	None	None
SPOLL	None	None
SQR	None	None
SQRT	None	None
STATUS	None	Not Supported
STOP	None	None
STORE	None	None
SUB	None	None

Table 12-4. Agilent 35670A Keyword Implementation (Continued)

Command	Front Panel Support	Exceptions
SUM	None	None
SYSTEM PRIORITY	None	None
SYSTEM$	None	Does not support “VERSION:”
TAB	None	None
TABXY	None	None
TAN	None	None
TIME	None	None
TIME$	None	None
TIMEDATE	None	None
TRIGGER	None	None
TRIM$	None	None
UPC$	None	None
VAL	None	None
VAL$	None	None
VIEWPORT	None	None
WAIT	None	None
WHERE	None	None
WHILE	None	None
WILDCARDS	None	Does not support UX
WINDOW	None	None

Index

Index

E

F

G

H

P

R

S

T

U

X

Need Assistance?

If you need assistance, contact your nearest Agilent Technologies Sales and Service Office listed in the Agilent Catalog. You can also find a list of local service representatives on the Web at:
http://www.agilent.com/find/assist *or contact your nearest regional office listed below.*

If you are contacting Agilent Technologies about a problem with your Agilent 35670 Dynamic Signal Analyzer, please provide the following information:

- ❑ Model number: Agilent 35670A
- ❑ Serial number:
- ❑ Options:
- ❑ Date the problem was first encountered:
- ❑ Circumstances in which the problem was encountered:
- ❑ Can you reproduce the problem?
- ❑ What effect does this problem have on you?

You may find the serial number and options from the front panel of your analyzer by executing the following:

Press [**System Utility**], [more], [serial number].

Press [**System Utility**], [options setup].

If you do not have access to the Internet, one of these centers can direct you to your nearest representative:

United States	Test and Measurement Call Center (800) 452-4844 (Toll free in US)
Canada	(905) 206-4725
Europe	(31 20) 547 9900
Japan	Measurement Assistance Center (81) 426 56 7832 (81) 426 56 7840 (FAX)
Latin America	(305) 267 4245 (305) 267 4288 (FAX)
Australia/New Zealand	1 800 629 485 (Australia) 0800 738 378 (New Zealand)
Asia-Pacific	(852) 2599 7777 (FAX) (852) 2506 9285