Example Programs Guide

Agilent Technologies
8712ET/ES and 8714ET/ES
RF Network Analyzers

For the most up-to-date versions of each example program as well as new programs that may not be listed in this manual, see Web site **http://www.agilent.com**. Use the search function to find Web pages related to 8712 example programs.

Agilent Technologies

Part No. 08714-90016

Printed in USA

Print Date: June 2000

Supersedes: October 1999

Notice

Key Conventions

This manual uses the following conventions:

FRONT PANEL KEY: This represents a key physically located on the analyzer (a “hardkey”).

Softkey : This indicates a “softkey”-- a key whose label is determined by the instrument’s firmware, and is displayed on the right side of the instrument’s screen next to the eight unlabeled keys.

Firmware Revision

This manual documents analyzers with firmware revisions E.06.00 and above.

Example Programs

This document provides a number of useful programs illustrating fundamental operations and techniques for control of the analyzer over the *General Purpose Interface Bus* (GPIB). Programs shown here are chosen to illustrate commonly-needed tasks, important analyzer commands, and fundamental techniques. The programs are also provided on a disk (included with the analyzer) and on Web site **http://www.agilent.com**. Use the search function to find Web pages related to 8712 example programs. The example programs can run on the analyzer's internal controller or on an external controller.

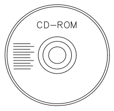
- *Example Programs Disk — DOS Format* : part number 08714-10003
- *Example Programs Disk — LIF Format* : part number 08714-10004

You should become familiar with the operation of your network analyzer before controlling it over GPIB. This document is not intended to teach programming or to discuss GPIB theory. Related information can be found in the following references:

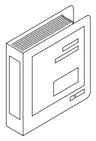
- Information on programming the analyzer is available in the analyzer's *Programmer's Guide*.
- Information on HP Instrument BASIC is available in the *HP Instrument BASIC User's Handbook* and the *HP Instrument BASIC User's Guide Handbook Supplement*.
- Information on HP BASIC programming is available in the manual set for the BASIC revision being used. For example: *BASIC 7.0 Programming Techniques* and *BASIC 7.0 Language Reference*.
- Information on using the GPIB is available in the *Tutorial Description of the General Purpose Interface Bus* (literature no. 5021-1927).
- Information on using the analyzer with a Local Area Network (LAN) is available in *The LAN Interface User's Guide Supplement*.
- Information on making measurements with the analyzer is available in the analyzer's *User's Guide*.

Contact the nearest Agilent sales office for ordering information. A list of sales and service offices can be found in the *User's Guide*.

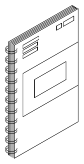
8712ET/ES and 8714ET/ES Network Analyzer Documentation Map



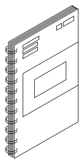
The **CDROM** provides the contents of all of the documents listed below.



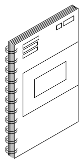
The **User's Guide** shows how to make measurements, explains commonly-used features, and tells you how to get the most performance from the analyzer.



The **LAN Interface User's Guide Supplement** shows how to use a local area network (LAN) for programming and remote operation of the analyzer.



The **Automating Measurements User's Guide Supplement** provides information on how to configure and control test systems for automation of test processes.



The **Programmer's Guide** provides programming information including GPIB and SCPI command references, as well as short programming examples.

The **Example Programs Guide** provides a tutorial introduction using BASIC programming examples to demonstrate the remote operation of the analyzer.

The **Service Guide** provides the information needed to adjust, troubleshoot, repair, and verify analyzer conformance to published specifications.

The **HP Instrument BASIC User's Handbook** describes programming and interfacing techniques using HP Instrument BASIC, and includes a language reference.

The **HP Instrument BASIC User's Handbook Supplement** shows how to use HP Instrument BASIC to program the analyzer.

The **Option 100 Fault Location and Structural Return Loss Measurements User's Guide Supplement** provides theory and measurement examples for making fault location and SRL measurements. (Shipped only with Option 100 analyzers.)

The **CATV Quick Start Guide** provides abbreviated instructions for testing the quality of coaxial cables. (Shipped only with Option 100 analyzers.)

The **Cellular Antenna Quick Start Guide** provides abbreviated instructions for verifying the performance of cellular antenna systems. (Shipped only with Option 100 analyzers.)

Contents

1 Introduction to the Example Programs

What's in This Book

This book contains source code listings of example programs written in HP BASIC. Each program demonstrates one or more fundamental concepts. All of the programs in this book are included on the *Example Programs Disk*, which comes with your analyzer. The programs can also be found at Web site **http://www.agilent.com**. Use the search function to find Web pages related to 8712 example programs.

NOTE The example programs listed in this book may not appear exactly as those programs on the Web site or on the disk that was shipped with your analyzer. The Web site and the disk contain the most up-to-date versions of each program as well as new programs that may not be listed here.

How This Book Is Organized

Chapter 1 Introduction to the Example Programs	Provides an introduction to this book, to the IBASIC programs, and provides a quick-reference guide to the programs.
Chapter 2 Example Programs	Contains the source code listings for the example programs, grouped by programming category.

Operating Environments

Because the example programs may run in different environments, the setup at the beginning of each program must determine the operating environment and properly set the analyzer's GPIB address. In the example programs, the internal IBASIC controller uses the address 800 when communicating with the analyzer (the internal GPIB is at select code 8). The default address of 716 is used when the programs are being run on an external controller.

A version of the following lines is included in most of the example programs. The use of the `Internal` (internal-controller) flag varies due to differences in the programs needs.

`10 IF POS(SYSTEM$("SYSTEM ID"),"HP 87") THEN`	*Identify the operating system.*
`20 ASSIGN @Hp8712 TO 800`	*If internal, set address to 800.*
`30 Internal=1`	*Set internal-control flag to 1.*
`40 ELSE`	
`50 ASSIGN @Hp8712 TO 716`	*If external, set address to 716.*
`60 Internal=0`	*Set internal-control flag to 0.*
`70 ABORT 7`	*Abort all bus transactions and give active control of the bus to the computer.*
`80 CLEAR 716`	*Send a selected device clear (SDC) to the analyzer – this clears all GPIB errors, resets the GPIB interface and clears syntax errors. (It does not affect the status reporting system.)*
`90 END IF`	

Guide to the Example Programs

This book groups the example programs into the following programming categories, and gives example implementations of the following techniques:

Table 1-1 Example Programs Listed by Programming Category (1 of 6)

Programming Category	Program Name	Page	Featured Techniques
annotating	USERANOT	2-4	using user-defined annotation
	FREQBLNK	2-6	limiting the display of information
	KEYCODES	2-8	reading user input from the front panel
creating and using bar codes	BARCODE	2-12	using bar codes to configure a test; using graphics
	STATS	2-16	using bar codes to configure a test; transferring trace data using a built-in subprogram
	DATALOG	2-19	reading bar code; storing trace data in internal RAM
calibrating[1]	TRANCAL	2-25	performing a transmission calibration
	REFLCAL	2-27	performing a reflection calibration
	LOADCALS	2-30	uploading and downloading correction arrays
	CALKIT	2-34	cal kit definition file (not executable)
	TWOPCAL[2]	2-35	performing a guided two-port calibration

1. See “making multiport measurements” on page 1-7 for multiport calibration.
2. For use with 8712ES and 8714ES models only.

Table 1-1 Example Programs Listed by Programming Category (2 of 6)

Programming Category	Program Name	Page	Featured Technique
configuring measurements	SETUP	2-51	setting up a basic measurement; using `*WAI`
	LIMITEST	2-53	performing an automatic pass/fail test
	POWERSWP	2-56	performing a power sweep
	FOURPARM[1]	2-58	measuring four parameters using a single sweep
customizing the display/using graphics	BARCODE	2-12	drawing a representation of the analyzer to the screen (subprogram)
	DRAW871X	2-61	drawing test diagrams on the screen
	GRAPHICS	2-64	creating customized graphics for procedures
	GRAPH2	2-70	drawing customized graphics
	GETPLOT	2-74	reading an HPGL file
using an external controller	DUALCTRL	2-78	downloading and running an IBASIC program
	TRICTRL	2-80	controlling multiple analyzers with a single controller
making fault location measurements[2]	FAULT	2-84	finding the effects of fault location frequency modes on cable measurements
	USR_FLOC	2-87	simplifying fault location measurements by using the **User BEGIN** key

1. For use with 8712ES and 8714ES models only.
2. For use with Option 100 (SRL and fault location).

Table 1-1 Example Programs Listed by Programming Category (3 of 6)

Programming Category	Program Name	Page	Featured Technique
controlling hardcopy	FAST_PRT	2-90	printing quickly to PCL5 printers (parallel port)
	PASSCTRL	2-92	printing to an GPIB printer using pass control
	PRINTPLT	2-95	printing to the serial and parallel ports
	REPORT	2-97	generating and printing a report (parallel port)
saving and recalling instrument states	LEARNSTR	2-101	uploading and downloading instrument states using the learn string
	SAVERCL	2-103	saving and recalling instrument states, instrument calibrations, and trace data
using the LAN[1]	lanio.c		controlling the analyzer with a C program over the LAN
	LAN_SEND		communicating with a computer with an IBASIC program over the LAN
	lan_serv		controlling multiple analyzers with a Perl script over the LAN
	ScpiDemo.java		sending SCPI commands over the LAN using JAVA socket programming
using marker functions	MKR_MATH	2-106	programming marker-math functions, including statistical, flatness testing, and filter statistics functions
using marker limit tests	LIM_FLAT	2-110	testing for a marker flatness limit
	LIM_PEAK	2-112	testing for a marker peak-to-peak ripple limit
	LIM_MEAN	2-115	testing for a marker mean limit

1. These programs are listed in the *LAN Interface User's Guide Supplement*. For additional information about the LAN example programs, see the README file on the *Example Programs Disk*.

Table 1-1 Example Programs Listed by Programming Category (4 of 6)

Programming Category	Program Name	Page	Featured Technique
making multi-port measurements[1]	PORT_SEL	2-119	using custom graphics to show the status of the multiport test set
	TSET_CAL	2-128	recalling the test set calibration file from memory and calibrating the test set
transferring programs	DOWNLOAD	2-138	downloading a program to the analyzer
	UPLOAD	2-140	uploading a program from the analyzer
using service requests	SRQ	2-143	generating a service request
	SRQ_INT	2-146	monitoring the status report from the analyzer
making SRL measurements[2]	MEAS_SRL	2-159	finding the effects of connector modeling on structural return loss measurements
	SRL_SRQ	2-162	initiating a cable scan and triggering an SRQ on completion

1. For use with compatible multiport test sets, such as the 87075C.
2. For use with Option 100 (SRL and Fault Location).

Table 1-1 Example Programs Listed by Programming Category (5 of 6)

Programming Category	Program Name	Page	Featured Technique
transferring data	STATS	2-16	reading and writing trace data (built-in subroutines)
	DATALOG	2-19	storing trace data in internal RAM; reading bar code
	SAVERCL	2-103	saving and recalling instrument states, instrument calibrations, and trace data
	MARKERS	2-167	transferring data using markers
	SMITHMKR	2-170	measuring reflection of a filter in Smith chart and polar formats
	ASCDATA	2-175	transferring data in ASCII format
	REALDATA	2-177	transferring data using IEEE 64-bit floating-point REAL format
	INTDATA	2-180	transferring data using 16-bit INTEGER format
	FAST_CW	2-183	transferring marker data in CW sweep mode
	DATA_EXT	2-185	transferring data between an internal and external program
	DATA_INT	2-187	transferring data to and from an external controller
transferring files over the GPIB	GETFILE	2-189	transferring files from the analyzer to an external controller
	PUTFILE	2-191	transferring files from an external controller to the analyzer
using TTL input and output	TTL_IO	2-193	monitoring the user TTL bit

Table 1-1 Example Programs Listed by Programming Category (6 of 6)

Programming Category	Program Name	Page	Featured Technique
creating user interface items	USERBEG	2-196	creating **User BEGIN** softkeys
	USERBEG1	2-198	(default **User BEGIN** program)
	USERBEG2	2-200	creating **User BEGIN** softkeys for fast recall of instrument states
	USER_BIT	2-202	reading and writing the USER bit
	USERKEYS	2-204	customizing the softkeys
	USR_FLOC[1]	2-87	creating **User BEGIN** softkeys for fault location measurements

1. For use with Option 100 (SRL and Fault Location).

Table 1-2 Example Programs Listed by Program Name (1 of 6)

Program Name	Page	Featured Techniques	Programming Category
ASCDATA	2-175	transferring data in ASCII format	transferring data
BARCODE	2-12	using bar codes to configure a test	creating and using bar codes
		using graphics	customizing the display/using graphics
		drawing a representation of the analyzer to the screen (subprogram)	customizing the display/using graphics
CALKIT	2-34	cal kit definition file (not executable)	calibrating
DATALOG	2-19	reading bar codes	creating and using bar codes
		storing trace data in internal RAM	transferring data
DATA_EXT	2-185	transferring data between an internal and an external program	transferring data
DATA_INT	2-187	transferring data to and from an external controller	transferring data
DOWNLOAD	2-138	downloading a program to the analyzer	transferring programs
DRAW871X	2-61	drawing test diagrams on the screen	customizing the display/using graphics
DUALCTRL	2-78	downloading and running an IBASIC program	using an external controller
FAST_CW	2-183	transferring marker data in CW sweep mode	transferring data

Table 1-2 **Example Programs Listed by Program Name (2 of 6)**

Program Name	Page	Featured Techniques	Programming Category
FAST_PRT	2-90	printing quickly to PCL5 printers (parallel port)	controlling hardcopy
FAULT[1]	2-84	finding the effects of fault location frequency modes on cable measurements	making fault location measurements
FOURPARM[2]	2-58	measuring four parameters using a single sweep	configuring measurements
FREQBLNK	2-6	limiting the display of information	annotating
GETFILE	2-189	transferring files from the analyzer to an external controller	transferring files over the GPIB
GETPLOT	2-74	reading an HPGL file	customizing the display/using graphics
GRAPH2	2-70	drawing customized graphics	customizing the display/using graphics
GRAPHICS	2-64	creating customized graphics for the screen	customizing the display/using graphics
INTDATA	2-180	transferring data using 16-bit integer format	transferring data
KEYCODES	2-8	reading user input from the front panel	annotating

1. For use with Option 100 (SRL and Fault Location).
2. For use with 8712ES and 8714ES models only.

Table 1-2 Example Programs Listed by Program Name (3 of 6)

Program Name	Page	Featured Techniques	Programming Category
lanio.c[1]		controlling the analyzer with a C program over the LAN	using the LAN
LAN_SEND[1]		communicating with a computer with an IBASIC program over the LAN	using the LAN
lan_serv[1]		controlling multiple analyzers with a Perl script over the LAN	using the LAN
LEARNSTR	2-101	uploading and downloading instrument states using the learn string	saving and recalling instrument states
LIMITEST	2-53	performing an automatic pass/fail test	configuring measurements
LIM_FLAT	2-110	testing for a marker flatness limit	using marker limit tests
LIM_MEAN	2-115	testing for a marker mean limit	using marker limit tests
LIM_PEAK	2-112	testing for a marker peak-to-peak ripple limit	using marker limit tests
LOADCALS	2-30	uploading and downloading correction arrays	calibrating
MARKERS	2-167	transferring data using markers	transferring data
MEAS_SRL[2]	2-159	finding the effects of connector modeling on structural return loss measurements	making SRL measurements

1. This program is listed in the *LAN Interface User's Guide Supplement*.
2. For use with Option 100 (SRL and Fault Location).

Table 1-2 Example Programs Listed by Program Name (4 of 6)

Program Name	Page	Featured Techniques	Programming Category
MKR_MATH	2-106	programming marker math functions, including statistical, flatness testing, and filter statistics functions	using marker functions
PASSCTRL	2-92	printing to an GPIB printer using pass control	controlling hardcopy
PORT_SEL[1]	2-119	getting the status of the multiport test set	making multiport measurements
		using custom graphics	customizing the display/using graphics
POWERSWP	2-56	performing a power sweep	configuring measurements
PRINTPLT	2-95	printing to serial and parallel ports	controlling hardcopy
PUTFILE	2-191	transferring files from an external controller to the analyzer	transferring files over the GPIB
REALDATA	2-177	transferring data using IEEE 64-bit floating-point real format	transferring data
REFLCAL	2-27	performing a reflection calibration	calibration
REPORT	2-97	generating and printing a report (parallel port)	controlling hardcopy
SAVERCL	2-103	saving and recalling instrument states	saving and recalling instrument states
		saving and recalling instrument calibrations	calibrating
		saving and recalling trace data	transferring data

1. For use with compatible multiport test sets, such as the 87075C.

Table 1-2 Example Programs Listed by Program Name (5 of 6)

Program Name	Page	Featured Techniques	Programming Category
ScpiDemo.java[1]		sending SCPI commands over the LAN using Java socket programming	using the LAN
SETUP	2-51	setting up a basic measurement	configuring measurements
		using *WAI	
SMITHMKR	2-170	measuring reflection of a filter in Smith chart and polar formats	transferring data
SRL_SRQ[2]	2-162	initiating a cable scan and triggering an SRQ on completion	making SRL measurements
SRQ	2-143	generating a service request	using service requests
SRQ_INT	2-146	monitoring the status report from the analyzer	using service requests
STATS	2-16	using bar codes to configure a test	creating and using bar codes
		using graphics	customizing the display/using graphics
		reading and writing trace data (built-in subroutines)	transferring data
TRANCAL	2-25	performing a transmission calibration	calibrating
TRICTRL	2-80	controlling multiple analyzers with a single controller	using an external controller
TSET_CAL[3]	2-128	recalling test set calibration files	making multiport measurements
		calibrating the test set	calibrating

1. This program is listed in the *LAN interface User's Guide Supplement*.
2. For use with Option 100 (SRL and Fault Location).
3. For use with compatible multiport test sets, such as the 87075C.

Table 1-2 Example Programs Listed by Program Name (6 of 6)

Program Name	Page	Featured Techniques	Programming Category
TTL_IO	2-193	monitoring the user TTL bit	using TTL input and output
TWOPCAL[1]	2-35	performing a guided two-port calibration	calibrating
UPLOAD	2-140	uploading a program to the analyzer	transferring programs
USERANOT	2-4	using user-defined annotation	annotating
USERBEG	2-196	creating **User BEGIN** softkeys	creating user interface items
USERBEG1	2-198	(default User BEGIN program)	creating user interface items
USERBEG2	2-200	creating **User BEGIN** softkeys for fast recall of instrument states	creating user interface items
USER_BIT	2-202	reading and writing the user bit	creating user interface items
USERKEYS	2-204	customizing the softkeys	creating user interface items
USR_FLOC[2]	2-87	simplifying fault location measurements by using the **User BEGIN** key	making fault location measurements

1. For use with 8712ES and 8714ES models only.
2. For use with Option 100 (SRL and fault location).

2 Example Programs Listings

Example Programs

Most of the example programs listed in this chapter are written in HP BASIC. They are also compatible with IBASIC (HP Instrument BASIC). An IBASIC program runs as a controller directly on the analyzer. It controls the analyzer over an internal interface bus that operates the same way as the external GPIB interface. For more information about IBASIC refer to the *HP Instrument BASIC User's Handbook*, and the *HP Instrument BASIC User's Guide Supplement*.

NOTE All of the progams in this book are tested for use with 8712ET/ES and 8714ET/ES analyzers. Programs in this book with comments that refer to the Agilent 8711 or 8711C model analyzers are compatible with your 8712ET/ES and 8714ET/ES analyzer.

Example Programs on Disk and the Web

All of the programs in this book are included on the *Example Programs Disk*, which comes with your analyzer:

ExamplePrograms Disk – DOS Format part number 08714-10003

A LIF version of the *Example Programs Disk* is available, but is not shipped with your analyzer:

ExamplePrograms Disk – LIF Format part number 08714-10004

NOTE The example programs on the disk that was shipped with your analyzer may not appear exactly as listed in this book. The programs on the disk are the most up-to-date versions of each program. Also, check your disk listings for new programs that may not be listed here.

For the most up-to-date versions of each example program as well as new programs that may not be listed in this manual or the disk, see Web site **http://www.agilent.com**. Use the search function to find Web pages related to 8712 example programs.

Annotating

USERANOT	Using user-defined annotation.
FREQBLNK	Concealing sensitive frequency information.
KEYCODES	Reading key presses and knob positions from the analyzer.

USERANOT Example Program

This program demonstrates how to use user-defined x-axis annotation. With this feature, you can set the analyzer to convert all x-axis information into a user-defined scale. In this program, the measurement channel x-axis is modified to display antenna angle in degrees while the measurement channel x-axis displays antenna height in feet.

```
100 !RE-SAVE "USERANOT"
110 !BASIC program: An example program to draw user-defined annotation
120 !$Revision: 1.1 $
130 !$Date: 97/09/02 13:14:22 $
140 !
150 ! Demonstrate these SCPI commands:
160 !
170 !   DISPlay:ANNotation:CHANnel[1|2]:USER[:STATe] {OFF|0|ON|1}
180 !   DISPlay:ANNotation:CHANnel[1|2]:USER:LABel[:DATA] <STRING>
190 !
200 !   DISPlay:ANNotation:FREQuency[1|2]:USER[:STATe] {OFF|0|ON|1}
210 !   DISPlay:ANNotation:FREQuency[1|2]:USER:STARt #-10000~10000#
220 !   DISPlay:ANNotation:FREQuency[1|2]:USER:STOP #-10000~10000#
230 !   DISPlay:ANNotation:FREQuency[1|2]:USER:SUFFix[:DATA] <STRING>
240 !   DISPlay:ANNotation:FREQuency[1|2]:USER:LABel[:DATA] <STRING>
250 !
260 !-
270 !
280 ! Determine select code (800 for IBASIC, 716 for external computer)
290 !
300   IF POS(SYSTEM$("SYSTEM ID"),"HP 87") THEN
310     ASSIGN @Hp8711 TO 800
320   ELSE
330     ASSIGN @Hp8711 TO 716
340     ABORT 7
350     CLEAR 716
360   END IF
370 !
380 ! Preset
390 OUTPUT @Hp8711;"SYST:PRES; *WAI"
400 OUTPUT @Hp8711;"SENS2:STAT ON"  ! So we can see markers
410 OUTPUT @Hp8711;"*OPC?"
420 ENTER @Hp8711;Opc
430 !
440 ! Set up channel annotation using:
450 !   DISPlay:ANNotation:CHANnel[1|2]:USER[:STATe] {OFF|0|ON|1}
460 !   DISPlay:ANNotation:CHANnel[1|2]:USER:LABel[:DATA] <STRING>
470 !
480 DISP "Setting up channel annotation..."
500 OUTPUT @Hp8711;"DISP:ANN:CHAN1:USER:STAT 1"
510 OUTPUT @Hp8711;"DISP:ANN:CHAN1:USER:LABEL 'Ch 1 Custom Annot: Signal vs.
    Antenna angle'"
520 OUTPUT @Hp8711;"DISP:ANN:CHAN2:USER:STAT 1"
530 OUTPUT @Hp8711;"DISP:ANN:CHAN2:USER:LABEL 'Ch 2 Custom Annot: Signal vs.
    Antenna height'"
540 DISP "Setting up channel annotation... Done."
550 WAIT 3
560 !
570 ! Set up frequency annotation using:
```

```
580 !   DISPlay:ANNotation:FREQuency[1|2]:USER[:STATe] {OFF|0|ON|1}
590 !   DISPlay:ANNotation:FREQuency[1|2]:USER:STARt #-10000~10000#
600 !   DISPlay:ANNotation:FREQuency[1|2]:USER:STOP #-10000~10000#
610 !   DISPlay:ANNotation:FREQuency[1|2]:USER:SUFFix[:DATA] <STRING>
620 !
630 DISP "Setting up frequency annotation..."
650 !
660 OUTPUT @Hp8711;"DISP:ANN:FREQ1:USER:LABEL 'Antenna Angle'"
670 OUTPUT @Hp8711;"DISP:ANN:FREQ1:USER:STAT 1"
680 OUTPUT @Hp8711;"DISP:ANN:FREQ1:USER:START -180.0"
690 OUTPUT @Hp8711;"DISP:ANN:FREQ1:USER:STOP   180.0"
700 OUTPUT @Hp8711;"DISP:ANN:FREQ1:USER:SUFFIX 'Deg'"
720 !
730 OUTPUT @Hp8711;"DISP:ANN:FREQ2:USER:LABEL 'Height'"
740 OUTPUT @Hp8711;"DISP:ANN:FREQ2:USER:STAT 1"
750 OUTPUT @Hp8711;"DISP:ANN:FREQ2:USER:START 5280.0"
760 OUTPUT @Hp8711;"DISP:ANN:FREQ2:USER:STOP -1760.0"
770 OUTPUT @Hp8711;"DISP:ANN:FREQ2:USER:SUFFIX 'Ft'"
780 !
790 DISP "Done.  Markers will read out using new units!"
800 OUTPUT @Hp8711;"CALC1:MARK1 ON"
810 OUTPUT @Hp8711;"CALC2:MARK1 ON"
820 !
830 LOCAL @Hp8711
840 !
850 END                                     !End of this program
```

FREQBLNK Example Program

This program demonstrates how to use user-defined x-axis annotation to conceal (or "blank") sensitive frequency information.

```
100 !RE-SAVE "FREQBLNK"
110 !BASIC program: An example program to overwrite frequency annotation
120 !$Revision: 1.2 $
130 !$Date: 97/09/02 13:17:25 $
140 !
150 ! Demonstrate using these SCPI commands to blank freq annotation.
160 !
170 !   DISPlay:ANNotation:FREQuency[1|2]:USER[:STATe] {OFF|0|ON|1}
180 !   DISPlay:ANNotation:FREQuency[1|2]:USER:START #-10000~10000#
190 !   DISPlay:ANNotation:FREQuency[1|2]:USER:STOP #-10000~10000#
200 !   DISPlay:ANNotation:FREQuency[1|2]:USER:SUFFix[:DATA] <STRING>
210 !   DISPlay:ANNotation:FREQuency[1|2]:USER:LABel[:DATA] <STRING>
220 !
230 !-
240 !
250 ! Determine select code (800 for IBASIC, 716 for external computer)
260 !
270   IF POS(SYSTEM$("SYSTEM ID"),"HP 87") THEN
280     ASSIGN @Hp8711 TO 800
290   ELSE
300     ASSIGN @Hp8711 TO 716
310     ABORT 7
320     CLEAR 716
330   END IF
340 !
350 ! Preset
360 OUTPUT @Hp8711;"SYST:PRES; *WAI"
370 OUTPUT @Hp8711;"SENS2:STAT ON"  ! So we can see markers
380 OUTPUT @Hp8711;"*OPC?"
390 ENTER @Hp8711;Opc
400 !
410 ! Set up frequency annotation using:
420 !   DISPlay:ANNotation:FREQuency[1|2]:USER[:STATe] {OFF|0|ON|1}
430 !   DISPlay:ANNotation:FREQuency[1|2]:USER:START #-10000~10000#
440 !   DISPlay:ANNotation:FREQuency[1|2]:USER:STOP #-10000~10000#
450 !   DISPlay:ANNotation:FREQuency[1|2]:USER:SUFFix[:DATA] <STRING>
460 !
470 DISP "Setting up frequency annotation..."
480 !
490 OUTPUT @Hp8711;"DISP:ANN:FREQ1:USER:LABEL 'Blank'"
500 OUTPUT @Hp8711;"DISP:ANN:FREQ1:USER:STAT 1"
510 OUTPUT @Hp8711;"DISP:ANN:FREQ1:USER:START 0.0"
520 OUTPUT @Hp8711;"DISP:ANN:FREQ1:USER:STOP  100.0"
530 OUTPUT @Hp8711;"DISP:ANN:FREQ1:USER:SUFFIX ''"
540 !
550 OUTPUT @Hp8711;"DISP:ANN:FREQ2:USER:LABEL 'Blank'"
560 OUTPUT @Hp8711;"DISP:ANN:FREQ2:USER:STAT 1"
570 OUTPUT @Hp8711;"DISP:ANN:FREQ2:USER:START 0.0"
580 OUTPUT @Hp8711;"DISP:ANN:FREQ2:USER:STOP  100.0"
590 OUTPUT @Hp8711;"DISP:ANN:FREQ2:USER:SUFFIX ''"
600 !
610 DISP "Done.  Markers will read out using new units!"
620 OUTPUT @Hp8711;"CALC1:MARK1 ON"
```

```
630 OUTPUT @Hp8711;"CALC2:MARK1 ON"
640 !
650 LOCAL @Hp8711
660 !
670 END                                       !End of this program
```

KEYCODES Example Program

This program will detect any front panel input, determine if it is from a keystroke or the knob, and display the corresponding keycode or value. Each key has a unique keycode associated with it. The knob will return either a positive or negative number depending upon direction of rotation (clockwise is positive). The program can be exited by pressing PRESET.

Lines `1290 - 1520` are continuously repeated to look for any front panel activity.

Lines `1370 - 1430` read the key type to determine if the activity came from a key press or from the knob. Also read is the value "Key_code." If the activity came from the knob, then the value "Key_code" represents how far the knob has been turned, and in which direction. If the activity came from a key stroke, the value represents the key's keycode.

Line `1510` determines if the PRESET key was pressed. If so, the program exits.

```
1000 ! Filename: KEYCODES
1010 !_
1020 !
1030 !  Demonstration of how to read the analyzer's
1040 !  front panel keys and knob, as well as external
1050 !  PC keyboard, using the SCPI SYST:KEY commands.
1060 !  This program reads key presses and knob turn
1070 !  ticks and displays them on the screen.
1080 !_
1090 !
1100 DIM Msg$[40]
1110 !
1120 !
1130 COM /Sys_state/ @Hp87xx,Scode
1140 ! Identify I/O Port
1150 CALL Iden_port
1160 !
1170 !
1180 ! Clear the key queue to ignore
1190 ! previous key presses.
1200 OUTPUT @Hp87xx;"SYST:KEY:QUE:CLE"
1210 !
1220 ! Turn on the key queue off to limit
1230 ! maximum que size to one.
1240 OUTPUT @Hp87xx;"SYST:KEY:QUE OFF"
1250 !
1260 Msg$="'Press keys or turn knob.  PRESET ends.'"
1270 OUTPUT @Hp87xx;"DISP:ANN:MESS:DATA ";Msg$
1280 !
1290 LOOP
1300 ! Query device status condition register
1310     OUTPUT @Hp87xx;"STAT:DEV:COND?"
1320     ENTER @Hp87xx;Dev_cond
```

```
1330 !
1340 ! Check the bit that indicates a key press.
1350     IF BIT(Dev_cond,0)=1 THEN
1360 ! Read the key type first
1370         OUTPUT @Hp87xx;"SYST:KEY:TYPE?"
1380         ENTER @Hp87xx;Key_type$
1390 ! Read the key code last.
1400 ! This removes it from the queue
1410         OUTPUT @Hp87xx;"SYST:KEY?"
1420         ENTER @Hp87xx;Key_code
1430         DISP "Keycode ";Key_code;" Type ";Key_type$;
1440 ! See how many more keys are in the queue
1450         OUTPUT @Hp87xx;"SYST:KEY:QUEUE:COUNT?"
1460         ENTER @Hp87xx;Key_count
1470         DISP ". Keys in queue:";Key_count
1480     END IF
1490 !
1500 ! Stop looping if the PRESET key was pressed.
1510 EXIT IF Key_code=56 AND Key_type$="KEY"
1520 END LOOP
1530 DISP "The end."
1540 OUTPUT @Hp87xx;"DISP:ANN:MESS:DATA 'The End.'"
1550 END
1560 !
1570 !************************************************************
1580 ! Iden_port:   Identify io port to use.
1590 ! Description: This routines sets up the I/O port address for
1600 !              the SCPI interface.  For "HP 87xx" instruments,
1610 !              the address assigned to @Hp87xx = 800 otherwise,
1620 !              716.
1630 !************************************************************
1640 SUB Iden_port
1650     COM /Sys_state/ @Hp87xx,Scode
1660 !
1670     IF POS(SYSTEM$("SYSTEM ID"),"HP 87")<>0 THEN
1680         ASSIGN @Hp87xx TO 800
1690         Scode=8
1700     ELSE
1710         ASSIGN @Hp87xx TO 716
1720         Scode=7
1730     END IF
1740 !
1750 SUBEND !Iden_port
1760 !
```

Creating and Using Bar Codes

You may use bar code readers to simplify your measurement setups. The HPKW-1220 KeyWand scanner or compatible bar code scanner will work with the analyzer. Connect your bar code scanner to the DIN keyboard connector. You may connect a keyboard or other DIN key input device in parallel with the bar code scanner. The bar code scanner will work in place of, or in addition to, your keyboard.

The `INPUT` statement is used to read the bar code from the scanner. When the input statement is encountered, the program will wait until the user has completed an input. The input is completed whenever a carriage return is received from the keyboard or a bar code has been successfully scanned by the bar code scanner. The following three programs, designed to run on the analyzer's internal IBASIC controller, demonstrate the use of bar code scanner applications as well as other useful applications. While a bar code scanner is useful in demonstrating these programs, it is not required.

BARCODE This program demonstrates basic bar code scanning to select one of three filter setups depending upon what is scanned. RF stimulus is set and response limits are read, set and tested for each device. Depending upon the result, the program prints "PASS" or "FAIL" on the CRT. Most useful in this program is a subprogram to draw a representation of an analyzer on the CRT. This code can be re-used in any user application that requires a guided setup. The analyzer image (and DUT image) can be scaled to any size, and can be offset in the X or Y axis as required. This is an excellent program to familiarize yourself with graphic routines using IBASIC graphics commands. Although most of the other example programs can be used on an external computer, this program is intended to be used in the analyzer's internal IBASIC environment only.

STATS This program performs a reflection calibration. The calibration is full band (performed over the instrument's preset source settings). This example also demonstrates the detection of front panel key presses, the use of softkeys, and the use of the `*OPC?` command. The STATS program also illustrates the use of the built-in, high-speed subprograms available in the analyzer.

DATALOG This program uploads and downloads correction arrays. The data transfer is performed in the 16-bit integer format. The arrays must be dimensioned properly for both the number of data points and the format of the data being transferred.

NOTE These three programs use `DIAG:PRES:SERV` to preset the analyzer, rather than `SYST:PRES`. This is because the preset power level can be user defined, and the normal preset command will not guarantee a 0 dBm level. This service command will force the power level to be set to 0 dBm. Failure to use `DIAG:PRES:SERV` will prevent proper operation of these programs.

BARCODE Example Program

```
10    !---------------------------------------------
20    !
30    ! IBASIC program:  BARCODE - Using barcode reader
40    ! For 871xC and 871xE
50    !
60    ! This IBASIC program was written for a barcode
70    ! reader, but it is not required.  Sets the 871x’s
80    ! state depending on model # of DUT being measured.
90    ! Expects to see BARCODE with the following format:
100   ! Model Number (6 char), space, Serial Number (5 char)
110   ! Valid Models:  BPF175, BPF200, SAW134
120   ! REV C.03.02   981009.JVV  Fixed limit fail problem
130   !
140   ! ----------------------------------------------
150   !
160   COM /Hpib/ @Rfna
170   COM /Scale/ Sc,INTEGER X,Y
180   DIM Name$[50],Stat$[50],Scan$[90],Lim$(1:3,1:5)[30],Test$(0:1)[4]
190   INTEGER Tab,Fail_flg,G(1:4)
200   !
210 Init:!
220   Test$(0)=”PASS”
230   Test$(1)=”FAIL”
240   ASSIGN @Rfna TO 800
250   Sc=1       ! Scales the 871x drawing and DUT
260   X=3        ! Starting X posn of 871x plot
270   Y=44       !    “      Y  “
280   Tab=38     ! Tab position for text
290   OUTPUT @Rfna;”SYST:PRES;*OPC?” !Preset
300   ENTER @Rfna;Opc
310   OUTPUT @Rfna;”SOUR:POW 0” !Set power if needed
320   OUTPUT @Rfna;”DISP:PROG UPP”
330   GINIT
340   GCLEAR
350   GESCAPE 1,3;G(*)
360   WINDOW G(1),G(3),G(2),G(4)
370   OUTPUT @Rfna;”SENS1:STAT OFF;:SENS2:STAT ON”
380   OUTPUT @Rfna;”SENS2:CORR:CSET DEF; *WAI”
390   OUTPUT @Rfna;”DISP:WIND2:TRAC:Y:RPOS 9”
400   WAIT .1
410   OUTPUT @Rfna;”ABOR;:INIT:CONT OFF”
420 Setup: !
430   BEEP 500,.1
440   INPUT “Enter Operator’s Name:”,Name$
450   BEEP 3000,.03
460   INPUT “Enter Station Number:”,Stat$
470   BEEP 3000,.03
480   OUTPUT @Rfna;”SYST:DATE?”
490   ENTER @Rfna;Year,Month,Day
500   CALL Draw_na              ! Draw HP871x
510   Box(418,44,212,163)       ! Draw text box
520   PRINT TABXY(Tab,3);”Oper: “;Name$[1,15]
530   PRINT TABXY(Tab,4);”Station: “;Stat$[1,11]
540   PRINT TABXY(Tab,5);”Date: “;Year;Month;Day
550 Meas_dev: !
560   LOOP
570      CALL Draw_dut(1)
```

```
580      CALL Scan_dut(Scan$,Cent$,Span$,Loss$,Lim$(*))
590      PRINT TABXY(Tab,7);”Model:  “&Scan$[1,6]
600      PRINT TABXY(Tab,8);”Serial: “&Scan$[8,12]
610      GOSUB Set_stim
620      DISP “MEASURING THE DEVICE”
630      OUTPUT @Rfna;”*CLS”
640      OUTPUT @Rfna;”ABOR;:INIT2:CONT OFF;:INIT2;*OPC?”
650      ENTER @Rfna;Opc ! wait for end of sweep
660      OUTPUT @Rfna;”CALC2:MARK1 ON;MARK:FUNC MAX”
670      OUTPUT @Rfna;”CALC2:MARK1:Y?”
680      ENTER @Rfna;Loss
690      OUTPUT @Rfna;”STAT:QUES:LIM:COND?”
700      ENTER @Rfna;Fail_flg
710 Disp_result:   !
720      PRINT TABXY(Tab,9);”Loss (dB): “;Loss
730      Fail_flg=BIT(Fail_flg,1) ! Bit 1 is for Ch2
740      IF Fail_flg THEN BEEP 2100,.5
750      Label(Test$(Fail_flg),78,58,22,5,0,1)
760 Continue:  !
770      CALL Draw_dut(0)
780      BEEP 300,.05
790      INPUT “Disconnect DUT. Measure another? (Y/n)”,Ans$
800   EXIT IF UPC$(Ans$[1,1])=”N”
810      Label(Test$(Fail_flg),78,58,22,5,0,0)
820   END LOOP
830   OUTPUT @Rfna;”ABOR;:INIT:CONT ON”
840   STOP
850   !
860 Set_stim: ! Set Freqs and Limit lines
870   OUTPUT @Rfna;”DISP:ANN:FREQ:MODE CSPAN”
880   OUTPUT @Rfna;”SENS:FREQ:CENT “&Cent$&” MHZ;SPAN “&Span$&” MHZ”
890   OUTPUT @Rfna;”DISP:WIND2:TRAC:Y:RLEV “;-PROUND(VAL(Loss$),1);”DB”
900   FOR I=1 TO 3     ! SET LIMIT LINES
910      OUTPUT @Rfna;”CALC2:LIM:SEGM”&VAL$(I)&”:TYPE “&Lim$(I,1)&”;STAT ON”
920      OUTPUT @Rfna;”CALC2:LIM:SEGM”&VAL$(I)&”:FREQ:STAR “&Lim$(I,2)&” MHZ;STOP
“&Lim$(I,3)&” MHZ”
930      OUTPUT @Rfna;”CALC2:LIM:SEGM”&VAL$(I)&”:AMPL:STAR “&Lim$(I,4)&” ;STOP
“&Lim$(I,5)
940   NEXT I
950   OUTPUT @Rfna;”CALC2:LIM:DISP ON;STAT ON”
960   RETURN
970   END
980   ! #########  SUBPROGRAMS  #########
990   !
1000 Draw_na:SUB Draw_na
1010     ! This draws HP 871x at origin X,Y
1020     Box(144,62,287,125)   ! Frame
1030     Box(144,62,289,127)
1040     Box(78,62,112,90)     ! CRT
1050     Box(78,62,114,94)
1060     FOR I=25 TO 102 STEP 11! Keys
1070        Box(146,I,9,6)
1080     NEXT I
1090     Box(178,110,9,9)      ! BEGIN
1100     Box(234,110,65,15)    ! Drive
1110     Box(234,110,47,5)
1120     Circle(228,75,12)     ! Knob
1130     Circle(187,19,7)      ! Out
1140     Circle(256,19,7)      ! in
1150     Box(10,25,4,12)       ! Switch
```

```
1160     Circle(10,41,3)
1170     Label(“RF OUT”,187,35,8,5,0,1)
1180     Label(“RF IN”,256,35,8,5,0,1)
1190  SUBEND
1200  !
1210 Draw_dut:SUB Draw_dut(INTEGER Pen)
1220     ! This connects DUT to HP 871x
1230     PEN Pen
1240     Connect(187,19,200,-25,0)
1250     Box(221,-25,44,19)
1260     Connect(256,19,243,-25,0)
1270     PEN 1
1280  SUBEND
1290  !
1300 Scan_dut:SUB Scan_dut(Scan$,Cent$,Span$,Loss$,Lim$(*))
1310     LOOP
1320        Invalid=0
1330        Scan$=”BPF175 12345”! Default model/serial
1340        BEEP 500,.05
1350        INPUT “Connect and scan the Device.”,Scan$ ! SCAN BARCODE HERE
1360        IF LEN(Scan$)<12 THEN  ! Valid device needs 12 char.
1370           Invalid=1
1380        ELSE
1390           Model$=Scan$[1,6]
1400           SELECT UPC$(TRIM$(Model$))
1410           CASE “BPF175”,”BPF177”
1420              RESTORE F1
1430           CASE “BPF200”
1440              RESTORE F2
1450           CASE “SAW134”
1460              RESTORE F3
1470           CASE ELSE
1480              Invalid=1
1490           END SELECT
1500        END IF
1510     EXIT IF NOT Invalid
1520        DISP Scan$;” <<--is INVALID!  Try again.”
1530        BEEP 1500,.2
1540        WAIT 1
1550     END LOOP
1560     BEEP 3000,.03
1570     READ Cent$,Span$,Loss$,Lim$(*)
1580     ! Limit lines format: Center, Span, Loss, (LIM TYPE, STRT, STP, STRTdB,
STPdB)
1590 F1: DATA 175,250,2    ! 175 MHz BPF
1600     DATA  “LMIN”, 160,190,-5,-5
1610     DATA  “LMAX”, 100,135,-45,-9
1620     DATA  “LMAX”, 215,245,-7,-30
1630 F2: DATA 200,100,1    ! 200 MHz BPF
1640     DATA  “LMIN”, 196,204,-3,-3
1650     DATA  “LMAX”, 180,190,-40,-10
1660     DATA  “LMAX”, 210,220,-10,-40
1670 F3: DATA 134,40,22    ! 134 MHz SAW BPF
1680     DATA  “LMIN”, 128,140,-27,-27
1690     DATA  “LMAX”, 123,125,-65,-30
1700     DATA  “LMAX”, 143,145,-30,-65
1710  SUBEND
1720  !
1730 Box:SUB Box(Xpos,Ypos,Xsize,Ysize)
1740     COM /Scale/ Sc,INTEGER X,Y
```

```
1750     MOVE X+(Xpos-Xsize/2)*Sc,Y+(Ypos-Ysize/2)*Sc
1760     RECTANGLE Xsize*Sc,Ysize*Sc
1770  SUBEND
1780  !
1790 Circle:SUB Circle(Xpos,Ypos,Radius)
1800     COM /Scale/ Sc,INTEGER X,Y
1810     MOVE X+Xpos*Sc,Y+Ypos*Sc
1820     POLYGON Radius*Sc,16,16
1830  SUBEND
1840  !
1850 Connect:SUB Connect(X1,Y1,X2,Y2,How)
1860     COM /Scale/ Sc,INTEGER X,Y
1870     MOVE X+X1*Sc,Y+Y1*Sc
1880     SELECT How
1890     CASE 1      !...diagonal
1900        DRAW X+X2*Sc,Y+Y2*Sc
1910     CASE 0
1920        DRAW X+X1*Sc,Y+Y2*Sc
1930        DRAW X+X2*Sc,Y+Y2*Sc
1940     CASE -1
1950        DRAW X+X2*Sc,Y+Y1*Sc
1960        DRAW X+X2*Sc,Y+Y2*Sc
1970     END SELECT
1980  SUBEND
1990  !
2000 Label:SUB Label(Text$,Xpos,Ypos,Size,Lorg,Ldr,Pen)
2010     COM /Scale/ Sc,INTEGER X,Y
2020     LORG Lorg
2030     LDIR Ldr
2040     CSIZE Size*Sc,.55
2050     MOVE X+Xpos*Sc,Y+Ypos*Sc
2060     PEN Pen
2070     LABEL Text$
2080     PEN 1
2090  SUBEND
2100  !
2110 Amp:SUB Amp(Xpos,Ypos,Size) ! Draws > Triangle
2120     COM /Scale/ Sc,INTEGER X,Y
2130     MOVE X+(Xpos+Size/2)*Sc,Y+Ypos*Sc
2140     POLYGON Size*Sc,3,3
2150  SUBEND
```

STATS Example Program

```
10    ! ----------------------------------------------
20    !
30    ! IBASIC program:  STATS - Collects statistics.
40    !
50    ! This HP 8711 IBASIC program uses a barcode reader
60    ! but it can be bypassed by simply pressing ENTER.
70    ! Displays running average of selected BPF passbands.
80    ! Finds linear avg of log data (ie Avg of 1dB & 5dB=3)
90    ! Expects to see BARCODE with the following format:
100   ! Model Number (6char), space, Serial Number (5char)
110   ! Valid Models:  BPF175, BPF200, SAW134
120   ! For 871xC/E and 8730A only
130   ! REV C.01.00    961009.JVV
140   !
150   ! ----------------------------------------------
160   !
170 Init: !
180   COM /Hpib/ @Rfna
190   COM Csub_loaded
200   DIM A(1:1601),M(1:1601)
210   INTEGER Points,N,I,Chan
220   Points=201 ! # of trace points
230   Chan=2
240   ASSIGN @Rfna TO 800
250   IF NOT Csub_loaded THEN
260      LOADSUB Read_fdata FROM “XFER:MEM 0,0”
270      LOADSUB Write_fmem FROM “XFER:MEM 0,0”
280      Csub_loaded=1
290   END IF
300   OUTPUT @Rfna;”DIAG:PRES:SERV;*OPC?”
310   ENTER @Rfna;Opc
320   OUTPUT @Rfna;”DISP:PROG UPP”
330   GINIT
340   GCLEAR
350   OUTPUT @Rfna;”DISP:ANN:MESS:STAT 0”
360   OUTPUT @Rfna;”SENS1:STAT OFF;:SENS2:STAT ON”
370   OUTPUT @Rfna;”SENS2:SWE:POIN “;Points   ! points
380   OUTPUT @Rfna;”DISP:WIND2:TRAC:Y:RPOS 9;PDIV 1 DB;*OPC?”
390   ENTER @Rfna;Opc
400   N=0
410 Setup: !
420   LOOP
430      GOSUB Scan_next
440      ! Softkey titles formatted for 871x usage
450      ON KEY 1 LABEL “ AVER THIS DATA” GOSUB Avg_this
460      ON KEY 3 LABEL “SCAN      ANOTHER” GOSUB Scan_next
470      ON KEY 5 LABEL “DONE” GOSUB Exit
480      LOOP
490         DISP “SELECT A SOFTKEY.”
500         WAIT 1
510         DISP
520         WAIT .3
530      END LOOP
540   END LOOP
550   !
560 Exit:  !
570   CLEAR SCREEN
```

```
580   DISP “PROGRAM PAUSED”
590   LOCAL @Rfna
600   PAUSE
610   RETURN
620   !
630 Scan_next:  !
640   LOOP
650      Scan_dut(Model$,Serial$,Cent$,Span$,Loss$)
660      IF Model$=”ABORT” THEN GOTO Exit
670      IF NOT N THEN Curr_model$=Model$
680   EXIT IF Model$=Curr_model$
690      DISP “Inconsistent Model #, Try again!”
700      BEEP 2100,.1
710      WAIT 1
720   END LOOP
730   CLEAR SCREEN
740   PRINT TABXY(1,4);”Device currently under test:”
750   PRINT “Model # “;Model$;”   Serial # “;Serial$
760   PRINT TABXY(1,6);”# Avg’d:”;N
770   PRINT TABXY(1,7);”Status of Serial # “&Serial$&”: MEASURING   “
780   GOSUB Set_stim
790   RETURN
800   !
810 Avg_this:  !
820   PRINT TABXY(1,7);”Status of Serial # “&Serial$&”: READING DATA”
830   Read_fdata(Chan,A(*))
840   N=N+1
850   PRINT TABXY(1,7);”Status of Serial # “&Serial$&”: AVERAGING   “
860   IF N=1 THEN
870      MAT M=A
880      OUTPUT @Rfna;”TRAC CH2SMEM,CH2SDATA;*WAI”
890      OUTPUT @Rfna;”CALC2:MATH (IMPL);:DISP:WIND2:TRAC1 ON;TRAC2
ON;*WAI”
900      OUTPUT @Rfna;”ABOR;:INIT2:CONT ON;*WAI”
910   ELSE
920      FOR I=1 TO Points
930         M(I)=(N-1)/N*M(I)+A(I)/N
940      NEXT I
950   END IF
960   PRINT TABXY(1,6);”# Averaged:”;N
970   PRINT TABXY(1,7);”Status of Serial # “&Serial$&”: WRITING DATA”
980   Write_fmem(Chan,M(*))
990   PRINT TABXY(1,7);”Status of Serial # “&Serial$&”: AVG COMPLETE”
1000  GOSUB Scan_next
1010  RETURN
1020  !
1030 Set_stim:! Set Freqs
1040  OUTPUT @Rfna;”DISP:ANN:FREQ:MODE CSPAN”
1050  OUTPUT @Rfna;”SENS:FREQ:CENT “&Cent$&” MHZ;SPAN “&Span$&” MHZ”
1060  OUTPUT @Rfna;”DISP:WIND2:TRAC:Y:RLEV -”&Loss$&” DB;*OPC?”
1070  ENTER @Rfna;Opc
1080  RETURN
1090  !
1100  END
1110  !
1120  ! #########  SUBPROGRAMS  #########
1130  !
1140 Scan_dut:SUB Scan_dut(Model$,Serial$,Cent$,Span$,Loss$)
1150     ALLOCATE Scan$[80]
1160     LOOP
```

```
1170        Invalid=0
1180        Scan$="ABORT"
1190        Scan$="BPF175 12345" !####### These 3 lines for demo only
1200        S$=VAL$(RND*1.E+9)   !####### Generates random S/N
1210        Scan$[8,12]=S$[3,7]  !####### Delete all to enable abort.
1220        BEEP 500,.05
1230        INPUT "Connect & scan DUT or leave blank to exit.",Scan$ !SCAN BARCODE
1240        IF LEN(Scan$)<12 THEN  ! Valid device needs 12 char.
1250           Invalid=1
1260        ELSE
1270           Model$=Scan$[1,6]
1280           SELECT UPC$(TRIM$(Model$))
1290           CASE "BPF175","BPF177"
1300              RESTORE F1
1310           CASE "BPF200"
1320              RESTORE F2
1330           CASE "SAW134"
1340              RESTORE F3
1350           CASE ELSE
1360              Invalid=1
1370           END SELECT
1380        END IF
1390     EXIT IF NOT Invalid
1400        IF POS(UPC$(Scan$),"ABORT") THEN
1410           Model$="ABORT"
1420           SUBEXIT
1430        END IF
1440        DISP Scan$;" <<--is INVALID!  Try again."
1450        BEEP 1500,.2
1460        WAIT 1
1470     END LOOP
1480     BEEP 3000,.03
1490     Serial$=Scan$[8,12]
1500     READ Cent$,Span$,Loss$  ! Data format: Center, Span, Nom Loss
1510 F1: DATA 175,50,2    ! 175 MHz BPF
1520 F2: DATA 200,12,1    ! 200 MHz BPF
1530 F3: DATA 134,15,22   ! 134 MHz SAW BPF
1540  SUBEND
```

DATALOG Example Program

```
1000 ! -----------------------------------------------
1010 !
1020 ! IBASIC program:  DATALOG - Logs trace data
1030 !
1040 ! This HP 8711 IBASIC program uses a barcode reader
1050 ! but it can be bypassed by simply pressing ENTER.
1060 ! Stores ASCII trace data in internal NonVol memory
1070 ! until full, then copies stored files to floppy.
1080 ! Expects to see BARCODE with the following format:
1090 ! Model Number (6char), space, Serial Number (5char)
1100 ! Valid Models:  BPF175, BPF200, SAW134
1110 ! For 871xC/E and 8730A only
1120 ! REV C.01.00    961009.JVV
1130 !
1140 ! -----------------------------------------------
1150 !
1160 Init:  !
1170 !
1180 !
1190 !  Make @Hp87xx common to all subroutines
1200   COM /Sys_state/ @Hp87xx,Scode
1210 !  Identify the computer we are running on
1220 !  and assign the i/o port address to @Hp87xx
1230   CALL Iden_port
1240 !
1250 !
1260   OUTPUT @Hp87xx;”DIAG:PRES:SERV;*OPC?”! Presets even 8730 to 871xC
state
1270   ENTER @Hp87xx;Opc
1280   OUTPUT @Hp87xx;”DISP:PROG UPP”
1290   GINIT
1300   GCLEAR
1310   GOSUB Warning   ! Warning may be deleted if desired
1320   OUTPUT @Hp87xx;”DISP:ANN:MESS:STAT 0”
1330   OUTPUT @Hp87xx;”SENS1:STAT OFF;:SENS2:STAT ON”
1340   OUTPUT @Hp87xx;”SENS2:SWE:POIN 201”! 201 points
1350   OUTPUT @Hp87xx;”DISP:WIND2:TRAC:Y:RPOS 9”
1360   OUTPUT @Hp87xx;”MMEM:MSIS ‘MEM:’”
1370   OUTPUT @Hp87xx;”MMEM:INIT ‘MEM:’,DOS”
1380   OUTPUT @Hp87xx;”MMEM:STOR:STAT:IST OFF;CORR OFF;TRAC OFF;*OPC?”
1390   ENTER @Hp87xx;Opc
1400 Setup:  !
1410   LOOP
1420      GOSUB Scan_next
1430      ! Softkey titles designed for 871x format
1440      ON KEY 1 LABEL “STORE THIS DATA” GOSUB Stor_mem
1450      ON KEY 2 LABEL “TRANSFER  TO FLOPPY” CALL Store_disk
1460      ON KEY 3 LABEL “SCAN      ANOTHER” GOSUB Scan_next
1470      ON KEY 5 LABEL “DONE” GOSUB Exit
1480      LOOP
1490         DISP “SELECT A SOFTKEY”
1500         WAIT 1
1510         DISP
1520         WAIT .3
1530      END LOOP
1540   END LOOP
1550 !
```

```
1560 Exit:   !
1570   Store_disk
1580   CLEAR SCREEN
1590   DISP “PROGRAM PAUSED”
1600   LOCAL @Hp87xx
1610   PAUSE
1620   RETURN
1630 !
1640 Scan_next:   !
1650   Scan_dut(Model$,Serial$,Cent$,Span$,Loss$)
1660   IF Model$=”ABORT” THEN GOTO Exit
1670   CLEAR SCREEN
1680   PRINT TABXY(1,3);”Device currently under test:”
1690   PRINT
1700   PRINT “Model # “;Model$;”   Serial # “;Serial$
1710   PRINT TABXY(1,7);”Status of Serial # “&Serial$&”: MEASURING     “
1720   GOSUB Set_stim
1730   RETURN
1740 !
1750 Stor_mem:   !
1760   PRINT TABXY(1,7);”Status of Serial # “&Serial$&”: STORING TO RAM”
1770   Store_ram(Model$,Serial$)
1780   PRINT TABXY(1,7);”Status of Serial # “&Serial$&”: STORING DONE  “
1790   GOSUB Scan_next
1800   RETURN
1810 !
1820 Set_stim:  ! Set Freqs
1830   OUTPUT @Hp87xx;”DISP:ANN:FREQ:MODE CSPAN”
1840   OUTPUT @Hp87xx;”SENS:FREQ:CENT “&Cent$&” MHZ;SPAN “&Span$&” MHZ”
1850   OUTPUT @Hp87xx;”DISP:WIND2:TRAC:Y:RLEV -”&Loss$&” DB;*OPC?”
1860   ENTER @Hp87xx;Opc
1870   RETURN
1880 !
1890 Warning:  !
1900   BEEP 3000,.3
1910   PRINT TABXY(15,4);”WARNING!”
1920   PRINT “This program will initialize the INTERNAL memory.”
1930   PRINT “All internally saved files will be lost!”
1940   PRINT
1950   PRINT “Do you wish to continue?  (y/N)”
1960   INPUT “Continue?”,Ans$
1970   CLEAR SCREEN
1980   IF UPC$(Ans$[1,1])=”Y” THEN RETURN
1990   END
2000 !
2010 ! #########  SUBPROGRAMS  #########
2020 !
2030 Scan_dut:SUB Scan_dut(Model$,Serial$,Cent$,Span$,Loss$)
2040      ALLOCATE Scan$[80]
2050      LOOP
2060         Invalid=0
2070         Scan$=”ABORT”
2080         Scan$=”BPF175 12345”!####### These 3 lines for demo only
2090         S$=VAL$(RND*1.E+9)!####### Generates random S/N
2100         Scan$[8,12]=S$[3,7]!####### Delete all to enable abort.
2110         BEEP 500,.05
2120         INPUT “Connect & scan DUT or leave blank to exit.”,Scan$!SCAN
BARCODE
2130         IF LEN(Scan$)<12 THEN ! Valid device needs 12 char.
2140            Invalid=1
```

```
2150         ELSE
2160            Model$=Scan$[1,6]
2170            SELECT UPC$(TRIM$(Model$))
2180            CASE “BPF175”,”BPF177”
2190               RESTORE F1
2200            CASE “BPF200”
2210               RESTORE F2
2220            CASE “SAW134”
2230               RESTORE F3
2240            CASE ELSE
2250               Invalid=1
2260            END SELECT
2270         END IF
2280      EXIT IF NOT Invalid
2290         IF POS(UPC$(Scan$),”ABORT”) THEN
2300            Model$=”ABORT”
2310            SUBEXIT
2320         END IF
2330         DISP Scan$;” <<--is INVALID!  Try again.”
2340         BEEP 1500,.2
2350         WAIT 1
2360      END LOOP
2370      BEEP 3000,.03
2380      Serial$=Scan$[8,12]
2390      READ Cent$,Span$,Loss$
2400 ! Data format: Center, Span, Loss
2410 F1:  DATA 175,300,2    ! 175 MHz BPF
2420 F2:  DATA 200,100,1    ! 200 MHz BPF
2430 F3:  DATA 134,30,22    ! 134 MHz SAW BPF
2440   SUBEND
2450 !
2460   SUB Store_ram(Model$,Serial$)
2470      COM /Sys_state/ @Hp87xx,Scode
2480      Id$=Model$[3,4]&”_”&Serial$! 2 unique chars + Ser
2490      ALLOCATE Err$[80]
2500      DISABLE
2510      REPEAT
2520         OUTPUT @Hp87xx;”*CLS”
2530         OUTPUT @Hp87xx;”MMEM:MSIS ‘MEM:’”
2540         OUTPUT @Hp87xx;”MMEM:STOR:TRAC CH2FDATA,’”&Id$&”’;*WAI”
2550         OUTPUT @Hp87xx;”SYST:ERR?”
2560         ENTER @Hp87xx;Err$
2570         SELECT VAL(Err$)
2580         CASE 0! No Problem
2590         CASE -254! Internal Mem full
2600            CALL Store_disk
2610         CASE -257! dupl file name
2620            OUTPUT @Hp87xx;”MMEM:DEL ‘”&Id$&”’;*WAI” !ERASE OLD
2630         CASE ELSE
2640            BEEP 2000,.5
2650            DISP Err$;
2660            INPUT “ Fix, Press ENTER”,Ans$
2670         END SELECT
2680      UNTIL VAL(Err$)=0
2690      ENABLE
2700   SUBEND
2710 !
2720   SUB Store_disk
2730      COM /Sys_state/ @Hp87xx,Scode
2740      ALLOCATE Err$[80]
```

```
2750      BEEP 700,.1
2760      DISP “Standby: Transferring internal files to disk.”
2770      LOOP
2780         OUTPUT @Hp87xx;”*CLS”
2790         OUTPUT @Hp87xx;”MMEM:COPY ‘*.*’ , ‘INT:’;*WAI”
2800         OUTPUT @Hp87xx;”SYST:ERR?”
2810         ENTER @Hp87xx;Err$
2820      EXIT IF NOT VAL(Err$)
2830         GOSUB Trap_err
2840      END LOOP
2850      OUTPUT @Hp87xx;”MMEM:MSIS ‘MEM:’;DEL ‘*.*’”
2860      SUBEXIT
2870 !
2880 Trap_err:   !
2890      IF VAL(Err$)=-250 THEN SUBEXIT! no file to xfer
2900      BEEP 2000,.5
2910      CLEAR SCREEN
2920      PRINT TABXY(1,4);”DISK ERROR DETECTED”
2930      PRINT “*** “&Err$&” ***”
2940      INPUT “Fix above problem, then press ENTER”,Ans$
2950      CLEAR SCREEN
2960   SUBEND
2970 !
2980   SUB Iden_port
2990 Iden_port:   ! Identify IO port to use
3000      COM /Sys_state/ @Hp87xx,Scode
3010      !
3020      IF POS(SYSTEM$(“SYSTEM ID”),”HP 87”)<>0 THEN
3030         ASSIGN @Hp87xx TO 800
3040         Scode=8
3050      ELSE
3060         ASSIGN @Hp87xx TO 716
3070         Scode=7
3080      END IF
3090   SUBEND! Iden_port
```

Calibrating

TRANCAL	This program performs a transmission calibration. The calibration is user defined (performed over the instrument's current source settings). This example also demonstrates the use of the `*OPC?` command.
REFLCAL	This program performs a reflection calibration. The calibration is full band (performed over the instrument's preset source settings). This example also demonstrates the detection of front panel key presses, the use of softkeys, and the use of the `*OPC?` command.
LOADCALS	This program demonstrates uploading and downloading correction arrays. The data transfer is performed in the 16-bit integer format. The arrays must be dimensioned properly for both the number of data points and the format of the data being transferred.
CALKIT	*This example is not a program*. It is an example of an instrument state file, used for downloading an instrument state. This type of file enables the user to calibrate the analyzer for use with connector types that are not in the firmware. See the *User's Guide* for more information on writing and editing your own cal kit file.
TWOPCAL	This program configures the analyzer to perform a two-port calibration.
KIT_THRU	This IBASIC program requires an 8712ET/ES or 8714ET/ES analyzer. The program develops the measurements needed to characterize the loss and delay of a through standard. This information can be stored into a user-defined calibration kit.

MPCKIT	This IBASIC program requires an 8712ET/ES or 8714ET/ES analyzer connected to a multiport test set. The program demonstrates how to create and assign user-defined male and female calibration kits with non-zero length throughs to each of four ports on the multiport test set.
SIMCAL	This program demonstrates how to create 2-port error-correction arrays from measurements of the raw (uncorrected) calibration standards when using the analyzer’s simcal command.

TRANCAL Example Program

This program demonstrates a transmission calibration performed over user-defined source settings (frequency range, power, and number of points). The operation complete query (*OPC?) is used at each step in the process to make sure the steps are taken in the correct order. More information on calibration is available in the *User's Guide*.

```
1000 ! Filename:  TRANCAL
1010 !
1020 ! Guide user through a transmission cal.
1030 !
1040 !
1050 COM /Sys_state/ @Hp87xx,Scode
1060 ! Identify I/O Port
1070 CALL Iden_port
1080 !
1090 !
1100 ! Configure the analyzer to measure transmission
1110 ! on channel 1.
1120 OUTPUT @Hp87xx;"SENS1:FUNC 'XFR:POW:RAT 2,0';DET NBAN;*WAI"
1130 !
1140 ! Select a calibration kit type.
1150 OUTPUT @Hp87xx;"SENS:CORR:COLL:CKIT 'COAX,7MM,TYPE-N,50,FEMALE'"
1160 !
1170 ! Select a transmission calibration for the current
1180 ! analyzer settings.  The "IST:OFF" ensures that
1190 ! the current settings will be used.
1200 OUTPUT @Hp87xx;"SENS1:CORR:COLL:IST OFF;METH TRAN1"
1210 !
1220 ! Prompt the operator to make a through
1230 ! connection.
1240 DISP "Connect THRU - Press Continue"
1250 PAUSE
1260 DISP "Measuring THRU"
1270 !
1280 ! Analyzer measures the through.
1290 OUTPUT @Hp87xx;"SENS1:CORR:COLL STAN1;*OPC?"
1300 !
1310 ! Wait until the measurement is complete.
1320 ENTER @Hp87xx;Opc
1330 DISP "Calculating Error Coefficients"
1340 !
1350 ! Tell the analyzer to calculate the
1360 ! error coefficients after the measurement
1370 ! is made, and then save for use during
1380 ! subsequent transmission measurements.
1390 ! Note that this is not the same as using
1400 ! the SAVE RECALL key functionality.
1410 OUTPUT @Hp87xx;"SENS1:CORR:COLL:SAVE;*OPC?"
1420 !
1430 ! Wait for the calculations and save to be
1440 ! completed.
1450 ENTER @Hp87xx;Opc
1460 DISP "User Defined TRANSMISSION CAL COMPLETED!"
1470 END
1480 !
```

```
1490 !**************************************************************
1500 ! Iden_port:   Identify io port to use.
1510 ! Description: This routines sets up the I/O port address for
1520 !              the SCPI interface.  For "HP 87xx" instruments,
1530 !              the address assigned to @Hp87xx = 800 otherwise,
1540 !              716.
1550 !**************************************************************
1560 SUB Iden_port
1570     COM /Sys_state/ @Hp87xx,Scode
1580 !
1590     IF POS(SYSTEM$("SYSTEM ID"),"HP 87")<>0 THEN
1600         ASSIGN @Hp87xx TO 800
1610         Scode=8
1620     ELSE
1630         ASSIGN @Hp87xx TO 716
1640         Scode=7
1650     END IF
1660 !
1670 SUBEND !Iden_port
1680 !
```

REFLCAL Example Program

This program demonstrates a reflection calibration performed over the preset source settings (frequency range, power, and number of points). The operation-complete query (*OPC?) is used at each step in the process to make sure the steps are taken in the correct order. More information on calibration is available in the *User's Guide*.

```
1000 !Filename:  REFLCAL
1010 !
1020 ! Guide user through a reflection cal.
1030 !
1040 DIM Msg$[50]
1050 !
1060 COM /Sys_state/ @Hp87xx,Scode,Internal
1070 ! Identify I/O Port
1080 CALL Iden_port
1090 !
1100 !
1110 ! Configure the analyzer to measure
1120 ! reflection on channel 1.
1130 OUTPUT @Hp87xx;"SENS1:FUNC 'XFR:POW:RAT 1,0';DET NBAN;*WAI"
1140 !
1150 ! Select Calibration Kit for 50 ohm instruments.
1160 OUTPUT @Hp87xx;"SENS:CORR:COLL:CKIT 'COAX,7MM,TYPE-N,50,FEMALE'"
1170 !
1180 ! Select Calibration Kit for 75 ohm instruments.
1190 ! (Comment out the 50 ohm line above and uncomment the line
1200 ! below.)
1210 ! OUTPUT @Hp87xx;"SENS:CORR:COLL:CKIT 'COAX,7MM,TYPE-N,75,FEMALE'"
1220 !
1230 ! Select a reflection calibration for the current
1240 ! analyzer settings.  The "IST:OFF" ensures that
1250 ! current settings will be used.
1260 OUTPUT @Hp87xx;"SENS1:CORR:COLL:IST OFF;METH REFL3"
1270 !
1280 ! Prompt the operator to connect an open.
1290 Msg$="Connect OPEN"
1300 GOSUB Get_continue
1310 DISP "Measuring OPEN"
1320 !
1330 ! Measure the open.
1340 OUTPUT @Hp87xx;"SENS1:CORR:COLL STAN1;*OPC?"
1350 !
1360 ! Wait until the measurement of the open
1370 ! is complete.
1380 ENTER @Hp87xx;Opc
1390 !
1400 ! Prompt the operator to connect a short.
1410 Msg$="Connect SHORT"
1420 GOSUB Get_continue
1430 DISP "Measuring SHORT"
1440 !
1450 ! Measure the short.
1460 OUTPUT @Hp87xx;"SENS1:CORR:COLL STAN2;*OPC?"
1470 !
1480 ! Wait until measurement of the short
```

```
1490 ! is complete.
1500 ENTER @Hp87xx;Opc
1510 !
1520 ! Prompt operator to connect a load.
1530 Msg$="Connect LOAD"
1540 GOSUB Get_continue
1550 DISP "Measuring LOAD"
1560 !
1570 ! Measure the load.
1580 OUTPUT @Hp87xx;"SENS1:CORR:COLL STAN3;*OPC?"
1590 ! Wait until measurement of the load
1600 ! is complete.
1610 ENTER @Hp87xx;Opc
1620 DISP "Calculating Error Coefficients"
1630 !
1640 ! Tell the analyzer to calculate the
1650 ! error coefficients, and then save
1660 ! for use during subsequent reflection
1670 ! measurements.  Note that this is not
1680 ! the same as using the SAVE RECALL key
1690 ! functionality.
1700 OUTPUT @Hp87xx;"SENS1:CORR:COLL:SAVE;*OPC?"
1710 !
1720 ! Wait for the calculations to be completed
1730 ! and the calibration saved.
1740 ENTER @Hp87xx;Opc
1750 DISP "Full Band REFLECTION CAL COMPLETED!"
1760 STOP
1770 !
1780 Get_continue: ! Subroutine to handle operator prompts.
1790 !
1800 ! "Internal" is determined above based on the
1810 ! controller.
1820 IF Internal=1 THEN
1830 !
1840 ! If internal control, then use the display
1850 ! line for the prompt.
1860     DISP Msg$&" - Press Measure Standard"
1870 !
1880 ! Use the softkey 2 for the response; loop
1890 ! while waiting for it to be pressed.
1900     ON KEY 2 LABEL "Measure    Standard" RECOVER Go_on
1910     LOOP
1920     END LOOP
1930 ELSE
1940 !
1950 ! If external control, clear the key queue
1960 ! so previous key presses will not interfere.
1970     OUTPUT @Hp87xx;"SYST:KEY:QUE:CLE"
1980 !
1990 ! Use the BEGIN key for the response.
2000     DISP Msg$&" - Press BEGIN to continue"
2010 !
2020 ! Turn on the key queue to trap all key
2030 ! presses.
2040     OUTPUT @Hp87xx;"SYST:KEY:QUE ON"
2050 !
2060 ! Loop while waiting for a key to be
2070 ! pressed.
2080     LOOP
```

```
2090 ! Query the device status condition
2100 ! register.
2110         OUTPUT @Hp87xx;"STAT:DEV:COND?"
2120         ENTER @Hp87xx;Dev_cond
2130 !
2140 ! Check the bit that indicates a key press.
2150         IF BIT(Dev_cond,0)=1 THEN
2160             OUTPUT @Hp87xx;"SYST:KEY?"
2170             ENTER @Hp87xx;Key_code
2180         END IF
2190 !
2200 ! Stop looping if the BEGIN key was pressed.
2210     EXIT IF Key_code=40
2220     END LOOP
2230     Key_code=0
2240 END IF
2250 !
2260 Go_on: ! Subroutine to turn off the softkeys
2270 ! on the analyzer and the computer,
2280 ! and return to main body of the
2290 ! program.
2300 OFF KEY
2310 RETURN
2320 END
2330 !
2340 !***************************************************************
2350 ! Iden_port:   Identify io port to use.
2360 ! Description: This routines sets up the I/O port address for
2370 !              the SCPI interface.  For "HP 87xx" instruments,
2380 !              the address assigned to @Hp87xx = 800 otherwise,
2390 !              716.
2400 !***************************************************************

2410 SUB Iden_port
2420     COM /Sys_state/ @Hp87xx,Scode,Internal
2430 !
2440     IF POS(SYSTEM$("SYSTEM ID"),"HP 87")<>0 THEN
2450         ASSIGN @Hp87xx TO 800
2460         Scode=8
2470         Internal=1
2480     ELSE
2490         ASSIGN @Hp87xx TO 716
2500         Scode=7
2510         Internal=0
2520     END IF
2530 !
2540 SUBEND !Iden_port
2550 !
```

LOADCALS Example Program

This program demonstrates how to read and write correction arrays to and from the analyzer. The `INTeger,16` data format is used because the data does not need to be interpreted; it only needs to be stored and retrieved. More information about calibration is available in the *User's Guide*.

The size of the arrays into which the data is read is important. If they are not dimensioned correctly, the program will not work. Most correction arrays, including the factory default (`DEF`) and the full band (`FULL`, preset source settings) arrays have 801 points. For user defined calibrations (`USER`), the number of points must be determined. If the number of points is other than 801, lines `1110` and `1790` will need to be changed to allocate arrays for the correct number of points. The number of points can be found by reading the correction array's header and determining the size as shown in the example below.

```
1000 !Filename:  LOADCALS
1010 !
1020 ! Description:
1030 !  1. Query the calibration arrays, based on
1040 !     the current measurement (trans/refl).
1050 !  2. Change number of points to 801
1060 !  3. Download the calibration arrays back
1070 !     into the analyzer.
1080 !
1090 DIM Func$[20],A$[10]
1100 INTEGER Swap,Arrays,Digits,Bytes,Points
1110 INTEGER Corr1(1:801,1:4),Corr2(1:801,1:4),Corr3(1:801,1:4)
1120 !
1130 COM /Sys_state/ @Hp87xx,Scode
1140 ! Identify I/O Port
1150 CALL Iden_port
1160 !
1170 !
1180 ! Query the measurement parameter.
1190 OUTPUT @Hp87xx;”SENS1:FUNC?”
1200 !
1210 ! Read the analyzer’s response.
1220 ENTER @Hp87xx;Func$
1230 !
1240 ! Set up a SELECT/CASE depending on the
1250 ! response.
1260 SELECT Func$
1270 !
1280 ! This is the transmission case, a ratio of
1290 ! the powers measured by detector 2 (B) and
1300 ! detector 0 (R).
1310 CASE “““XFR:POW:RAT 2, 0”””
1320 !
1330 ! The transmission calibration has only one
1340 ! correction array.
1350    Arrays=1
```

```
1360 !
1370 ! This is the reflection case, a ratio of
1380 ! the powers measured by detector 1 (A) and
1390 ! detector 0 (R).
1400 CASE """XFR:POW:RAT 1, 0"""
1410 !
1420 ! The reflection calibration has 3 correction
1430 ! arrays.
1440     Arrays=3
1450 END SELECT
1460 !
1470 ! Select the 16 bit integer binary data format.
1480 OUTPUT @Hp87xx;"FORM:DATA INT,16"
1490 !
1500 ! Select normal byte order.
1510 OUTPUT @Hp87xx;"FORM:BORD NORM"
1520 !
1530 ! Request the first correction array from the a
1540 ! analyzer.
1550 OUTPUT @Hp87xx;"TRAC? CH1SCORR1"
1560 !
1570 ! Turn on ASCII formatting on the I/O path
1580 ! to read the header information.
1590 ASSIGN @Hp87xx;FORMAT ON
1600 !
1610 ! Get the header, including the number of
1620 ! of characters that will hold the number
1630 ! of bytes value which follows.
1640 ENTER @Hp87xx USING "%,A,D";A$,Digits
1650 !
1660 ! Get the rest of the header.  The number
1670 ! of bytes to capture in the correction
1680 ! array will be placed in "Bytes".  Note
1690 ! the use of "Digits" in the IMAGE string.
1700 ENTER @Hp87xx USING "%,"&VAL$(Digits)&"D";Bytes
1710 !
1720 ! Determine the number of points from the
1730 ! number of bytes (8 bytes per point).
1740 Points=Bytes/8
1750 !
1760 ! This example was set up in line 1110 above
1770 ! for 801 points.  Edit this line and line 1110
1780 ! to allow other dimensions.
1790 IF Points<>801 THEN
1800     DISP "Arrays are not dimensioned for this calibration"
1810     STOP
1820 END IF
1830 DISP "Uploading (querying) calibration arrays . . . ."
1840 !
1850 ! Turn off ASCII formatting on the I/O path.
1860 ASSIGN @Hp87xx;FORMAT OFF
1870 !
1880 ! Get the first error correction array.
1890 ENTER @Hp87xx;Corr1(*)
1900 !
1910 ! Turn on ASCII formatting.
1920 ASSIGN @Hp87xx;FORMAT ON
1930 !
1940 ! Get the "end of data" character.
1950 ENTER @Hp87xx;A$
```

```
1960 !
1970 ! For the reflection there are two more
1980 ! arrays to read.
1990 IF Arrays=3 THEN
2000 !
2010 ! Request and read in the second
2020 ! correction array.
2030     OUTPUT @Hp87xx;”TRAC? CH1SCORR2”
2040     Read_array(@Hp87xx,Corr2(*))
2050 !
2060 ! Request and read in the third
2070 ! correction array.
2080     OUTPUT @Hp87xx;”TRAC? CH1SCORR3”
2090     Read_array(@Hp87xx,Corr3(*))
2100 END IF
2110 DISP “Calibration arrays have been uploaded.”
2120 WAIT 5
2130 DISP “Downloading (setting) calibration arrays . . . .”
2140 !
2150 ! Turn off correction before writing a
2160 ! calibration back into the analyzer.
2170 OUTPUT @Hp87xx;”SENS1:CORR:STAT OFF”
2180 !
2190 ! Set the number of points for the correction
2200 ! arrays.  (Not necessary in this example,
2210 ! but shown for emphasis.)
2220 OUTPUT @Hp87xx;”SENS1:SWE:POIN”;Points
2230 !
2240 ! Prepare the analyzer to receive the first
2250 ! correction array in the indefinite block
2260 ! length format.
2270 OUTPUT @Hp87xx;”TRAC CH1SCORR1, #0”;
2280 !
2290 ! Turn off ASCII formatting.
2300 ASSIGN @Hp87xx;FORMAT OFF
2310 !
2320 ! Send the first correction array to the
2330 ! analyzer.  The array transfer is
2340 ! terminated with the “END” signal.
2350 OUTPUT @Hp87xx;Corr1(*),END
2360 !
2370 ! Turn on ASCII formatting.
2380 ASSIGN @Hp87xx;FORMAT ON
2390 !
2400 ! For a reflection array download, there
2410 ! are two more arrays.
2420 IF Arrays=3 THEN
2430 !
2440 ! Prepare the analyzer to receive the
2450 ! 2nd array, then output it.
2460     OUTPUT @Hp87xx;”TRAC CH1SCORR2, “;
2470     Write_array(@Hp87xx,Corr2(*))
2480 !
2490 ! Prepare the analyzer to receive the
2500 ! 3rd array, then output it.
2510     OUTPUT @Hp87xx;”TRAC CH1SCORR3, “;
2520     Write_array(@Hp87xx,Corr3(*))
2530 END IF
2540 !
2550 ! Turn on the calibration just downloaded.
```

```
2560 OUTPUT @Hp87xx;”SENS1:CORR:STAT ON;*WAI”
2570 DISP “Calibration arrays have been downloaded.”
2580 END
2590 !
2600 ! Subprogram for reading binary data array from
2610 ! the analyzer.  The command requesting a specific
2620 ! data array has already been sent prior to
2630 ! calling this subprogram.
2640 !
2650 SUB Read_array(@Hp87xx,INTEGER Array(*))
2660     DIM A$[10]
2670     INTEGER Digits,Bytes
2680     ASSIGN @Hp87xx;FORMAT ON
2690     ENTER @Hp87xx USING “%,A,D”;A$,Digits
2700     ENTER @Hp87xx USING “%,”&VAL$(Digits)&”D”;Bytes
2710     ASSIGN @Hp87xx;FORMAT OFF
2720     ENTER @Hp87xx;Array(*)
2730     ASSIGN @Hp87xx;FORMAT ON
2740     ENTER @Hp87xx;A$
2750 SUBEND
2760 !
2770 ! Subprogram for writing binary data array to
2780 ! the analyzer.  The command requesting a specific
2790 ! data array has already been sent prior to
2800 ! calling this subprogram.
2810 !
2820 SUB Write_array(@Hp87xx,INTEGER Array(*))
2830     OUTPUT @Hp87xx;”#0”;
2840     ASSIGN @Hp87xx;FORMAT OFF
2850     OUTPUT @Hp87xx;Array(*),END
2860     ASSIGN @Hp87xx;FORMAT ON
2870 SUBEND
2880 !
2890 !**************************************************************
2900 ! Iden_port:   Identify io port to use
2910 ! Description: This routines sets up the I/O port address for
2920 !              the SCPI interface.  For “HP 87xx” instruments,
2930 !              the address assigned to @Hp87xx = 800 otherwise,
2940 !              716.
2950 !**************************************************************
2960 SUB Iden_port
2970     COM /Sys_state/ @Hp87xx,Scode
2980 !
2990     IF POS(SYSTEM$(“SYSTEM ID”),”HP 87”)<>0 THEN
3000         ASSIGN @Hp87xx TO 800
3010         Scode=8
3020     ELSE
3030         ASSIGN @Hp87xx TO 716
3040         Scode=7
3050     END IF
3060 !
3070 SUBEND !Iden_port
3080 !
```

CALKIT Example File

This instrument state file demonstrates the type of file required to download user-defined calibration kits. Refer to the *User's Guide* for information on user-defined cal kits.

```
10  !$  Standard Definitions for HP 85054B Precision
        Type-N Cal Kit.
11  !
12  !$  This is a Cal Kit definition file, which
13  !$  uses the same format as a BASIC program.
14  !$  Lines that contain "!$" are comments.
15  !$
16  !$  Put your Cal Kit file on a disk, and use the
17  !$  analyzer's [SAVE/RECALL] [Recall State] keys
18  !$  to load your custom Cal Kit into the analyzer.
20  !
30  !$  Definitions for 50 Ohm jack (FEMALE center
        contact) test
40  !$  ports, plug (MALE center contact) standards.
50  !
60  ! OPEN:  $ HP 85054-60027 Open Circuit Plug
70  !    Z0  50.0 $ Ohms
80  !    DELAY  57.993E-12 $ Sec
90  !    LOSS  0.8E+9 $ Ohms/Sec
100 !    C0  88.308E-15 $ Farads
110 !    C1  1667.2E-27 $ Farads/Hz
120 !    C2 -146.61E-36 $ Farads/Hz^2
130 !    C3  9.7531E-45 $ Farads/Hz^3
140 !
150 ! SHORT:  $ HP 85054-60025 Short Circuit Plug
160 !    Z0  50.0 $ Ohms
170 !    DELAY  63.078E-12 $ Sec
180 !    LOSS  8.E+8 $ Ohms/Sec
190 !
200 ! LOAD:  $ HP 00909-60011 Broadband Load Plug
210 !    Z0  50.0 $ Ohms
220 !    DELAY  0.0 $ Sec
230 !    LOSS   0.0 $ Ohms/Sec
240 !
250 ! THRU:  $ HP 85054-60038 Plug to Plug Adapter
260 !    Z0  50.0 $ Ohms
270 !    DELAY  196.0E-12 $ Sec
280 !    LOSS   2.2E+9 $ Ohms/Sec
290 !
300 END
```

TWOPCAL Example Program

This program configures the analyzer to perform a two-port calibration.

```
1 !Filename:  TWOPCAL
2 !
3 ! Guide user through a Two Port User cal.
4 !
5 DIM Msg$[50]
6 COM /Sys_state/ @Hp87xx,Scode,Internal
7 !
8 ! Identify I/O Port
9 CALL Iden_port
10 !
11 ! Configure the analyzer to measure reflection on channel 1.
12 OUTPUT @Hp87xx;"SENS1:FUNC 'XFR:S 1,1';DET NBAN;*WAI"
13 !
14 ! Select Calibration Kit for 50 ohm instruments.
15 ! Assumes that user device ports are as follows:
16 !
17 !           Device Port     Test (Instrument) Port
18 !           -----------     ---------------------
19 !  Port 1      MALE             FEMALE
20 !  Port 2    FEMALE               MALE
21 !
22 OUTPUT @Hp87xx;"SENS:CORR:COLL:CKIT:PORT1 'COAX,7MM,TYPE-N,50,FEMALE'"
23 OUTPUT @Hp87xx;"SENS:CORR:COLL:CKIT:PORT2 'COAX,7MM,TYPE-N,50,MALE'"
24 !
25 ! Select Calibration Kit for 75 ohm instruments.
26 ! (Comment out the 50 ohm line above and uncomment the line below.)
27 ! OUTPUT @Hp87xx;"SENS:CORR:COLL:CKIT:PORT1 'COAX,7MM,TYPE-N,75,FEMALE'"
28 ! OUTPUT @Hp87xx;"SENS:CORR:COLL:CKIT:PORT2 'COAX,7MM,TYPE-N,75,MALE'"
29 !
30 ! Select a Two Port calibration for the current analyzer settings.
31 ! The "IST:OFF" ensures that current settings will be used.
32 OUTPUT @Hp87xx;"SENS1:CORR:COLL:IST OFF;METH TWOPort"
33 !
34 ! Prompt the operator to connect a Thru and measure it
35 !
36 Msg$="Connect THRU from Port 1 to Port 2"
37 GOSUB Get_continue
38 DISP "Measuring THRU"
39 OUTPUT @Hp87xx;"SENS1:CORR:COLL STAN1;*OPC?"
40 ENTER @Hp87xx;Opc
41 !
42 ! Prompt the operator to connect an Open and measure it
43 !
44 Msg$="Connect male OPEN to Port 1"
45 GOSUB Get_continue
46 DISP "Measuring Open"
47 OUTPUT @Hp87xx;"SENS1:CORR:COLL STAN2;*OPC?"
48 ENTER @Hp87xx;Opc
49 !
50 ! Prompt the operator to connect a Short and measure it
51 !
52 Msg$="Connect male SHORT to Port 1"
53 GOSUB Get_continue
54 DISP "Measuring SHORT"
55 OUTPUT @Hp87xx;"SENS1:CORR:COLL STAN3;*OPC?"
```

```
56 ENTER @Hp87xx;Opc
57 !
58 ! Prompt the operator to connect a Load and measure it
59 !
60 Msg$="Connect male LOAD to Port 1"
61 GOSUB Get_continue
62 DISP "Measuring LOAD"
63 OUTPUT @Hp87xx;"SENS1:CORR:COLL STAN4;*OPC?"
64 ENTER @Hp87xx;Opc
65 !
66 ! Prompt the operator to connect an Open and measure it
67 !
68 Msg$="Connect female OPEN to Port 2"
69 GOSUB Get_continue
70 DISP "Measuring Open"
71 OUTPUT @Hp87xx;"SENS1:CORR:COLL STAN5;*OPC?"
72 ENTER @Hp87xx;Opc
73 !
74 ! Prompt the operator to connect a Short and measure it
75 !
76 Msg$="Connect female SHORT to Port 2"
77 GOSUB Get_continue
78 DISP "Measuring SHORT"
79 OUTPUT @Hp87xx;"SENS1:CORR:COLL STAN6;*OPC?"
80 ENTER @Hp87xx;Opc
81 !
82 ! Prompt the operator to connect a Load and measure it
83 !
84 Msg$="Connect female LOAD to Port 2"
85 GOSUB Get_continue
86 DISP "Measuring LOAD"
87 OUTPUT @Hp87xx;"SENS1:CORR:COLL STAN7;*OPC?"
88 ENTER @Hp87xx;Opc
89 !
90 !
91 !
92 DISP "Calculating Error Coefficients"
93 !
94 ! Tell the analyzer to calculate the
95 ! error coefficients, and then save
96 ! for use during subsequent reflection
97 ! measurements.  Note that this is not
98 ! the same as using the SAVE RECALL key
99 ! functionality.
100 OUTPUT @Hp87xx;"SENS1:CORR:COLL:SAVE;*OPC?"
101 !
102 ! Wait for the calculations to be completed
103 ! and the calibration saved.
104 ENTER @Hp87xx;Opc
105 DISP "Full Band TWO PORT CAL COMPLETED!"
106 STOP
107 !
108 Get_continue: ! Subroutine to handle operator prompts.
109 !
110 ! "Internal" is determined above based on the
111 ! controller.
112 IF Internal=1 THEN
113 !
114 ! If internal control, then use the display
115 ! line for the prompt.
```

```
116 DISP Msg$&” - Press Measure Standard”
117 !
118 ! Use the softkey 2 for the response; loop
119 ! while waiting for it to be pressed.
120 ON KEY 2 LABEL “Measure    Standard” RECOVER Go_on
121 LOOP
122 END LOOP
123 ELSE
124 !
125 ! If external control, clear the key queue
126 ! so previous key presses will not interfere.
127 OUTPUT @Hp87xx;”SYST:KEY:QUE:CLE”
128 !
129 ! Use the BEGIN key for the response.
130 DISP Msg$&” - Press BEGIN to continue”
131 !
132 ! Turn on the key queue to trap all key
133 ! presses.
134 OUTPUT @Hp87xx;”SYST:KEY:QUE ON”
135 !
136 ! Loop while waiting for a key to be
137 ! pressed.
138 LOOP
139 ! Query the device status condition
140 ! register.
141 OUTPUT @Hp87xx;”STAT:DEV:COND?”
142 ENTER @Hp87xx;Dev_cond
143 !
144 ! Check the bit that indicates a key press.
145 IF BIT(Dev_cond,0)=1 THEN
146 OUTPUT @Hp87xx;”SYST:KEY?”
147 ENTER @Hp87xx;Key_code
148 END IF
149 !
150 ! Stop looping if the BEGIN key was pressed.
151 EXIT IF Key_code=40
152 END LOOP
153 Key_code=0
154 END IF
155 !
156 !
157 Go_on: ! Subroutine to turn off the softkeys
158 ! on the analyzer and the computer,
159 ! and return to main body of the
160 ! program.
161 OFF KEY
162 RETURN
163 END
164 !
165 !**************************************************************
166 ! Iden_port:   Identify io port to use
167 ! Description: This routines sets up the I/O port address for
168 !              the SCPI interface.  For “HP 87xx” instruments,
169 !              the address assigned to @Hp87xx = 800 otherwise,
170 !              716.
171 !**************************************************************
172 SUB Iden_port
173 COM /Sys_state/ @Hp87xx,Scode,Internal
174 !
175 IF POS(SYSTEM$(“SYSTEM ID”),”HP 87”)<>0 THEN
```

```
176 ASSIGN @Hp87xx TO 800
177 Scode=8
178 Internal=1
179 ELSE
180 ASSIGN @Hp87xx TO 716
181 Scode=7
182 Internal=0
183 END IF
184 SUBEND !Iden_port
```

KIT_THRU Example Program

This IBASIC program requires an 8712ET/ES or 8714ET/ES analyzer. The program develops the measurements needed to characterize the loss and delay of a through standard. This information can be stored into a user-defined calibration kit.

```
10    ! RE-SAVE “KIT_THRU.BAS”

20    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
30    !
40    ! This program can be used to characterize a THRU standard and install
50    ! it’s charecterzation into a user defined cal kit.
60    !
70    ! A 1-Port calibration is first performed.  Several measurements are
80    ! are then made.
90    !
100   ! The results are used to compute the LOSS and DELAY terms needed
110   ! to characterize the THRU standard.
120   !
130   ! The results are stored into a User Defined Cal Kit.
140   !
150   !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
160   !
170     ASSIGN @Hp8714 TO 800
180   !
190     OUTPUT @Hp8714;”SYST:PRES; *opc?”
200     ENTER @Hp8714;Opc
210   !
220   ! Setup an IBASIC Window
230     OUTPUT @Hp8714;”DISP:PROG LOW”
240   !
250   ! Query the system impedance.  The system impedance is needed to
260   ! compute the LOSS term.
270     OUTPUT @Hp8714;”sens:corr:imp:inp:magn:sel?”
280     ENTER @Hp8714;Z_sel
290     IF Z_sel=0 THEN
300       Z0=50
310     ELSE
320       Z0=75
330     END IF
340   !
350   ! The LOSS term is computed from measurements at 1 GHz.  Setup
360   ! the analyzer for 1 GHz.
370     OUTPUT @Hp8714;”SENS1:STAT ON; *WAI”
380     OUTPUT @Hp8714;”SENS1:FUNC ‘XFR:S 1,1’;DET NBAN; *WAI”
390     OUTPUT @Hp8714;”SENS1:FREQ:SPAN 100 MHZ;;*WAI”
400     OUTPUT @Hp8714;”SENS1:FREQ:CENT 1000000000 HZ;*WAI”
410     OUTPUT @Hp8714;”disp:ann:message:clear”
420   !
430   ! Select a Type-N cal kit for modification
440     OUTPUT @Hp8714;”SENS:CORR:CKIT:MODIFY TYPENF”
450   !
460   ! Query the DELAY for the OPEN standard.
470     OUTPUT @Hp8714;”SENS1:CORR:CKIT:OPEN:mod:del?”
480     ENTER @Hp8714;Open_delay
490   !
500     PRINT
```

```
510    PRINT “>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>”
520    PRINT “Program to measure a THROUGH standard.”
530    PRINT “>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>”
540    PRINT
550    BEEP
560  !
570  ! Initiate a 1-Port calibration
580    OUTPUT @Hp8714;”disp:ann:message:clear”
590    OUTPUT @Hp8714;”disp:ann:message ‘Perform a 1-Port calibration now.’”
600    PRINT “Perform a 1-Port calibration.”
610    DISP “Press [Continue] when done.”
620    PRINT “Press [System Options] [IBASIC] [Continue] when done.”
630    PRINT
640    PAUSE
650    BEEP
660  !
670  !Prepare to measure the THROUGH standard.
680    OUTPUT @Hp8714;”disp:ann:message:clear”
690    OUTPUT @Hp8714;”disp:ann:message ‘Connect the THROUGH 
standard.”&CHR$(10)&”Terminate it with an OPEN.’”
700    DISP “Press [Continue] when done.”
710    PRINT “Connect the THROUGH standard.  Terminate it with an OPEN.”
720    PRINT “Press [System Options] [IBASIC] [Continue] when done.”
730    PRINT
740    PAUSE
750    DISP ““
760  !
770  ! Measure the THRU LOSS using a marker
780    OUTPUT @Hp8714;”CALC1:MARK1 ON”
790  !
800    OUTPUT @Hp8714;”calc1:mark1:y?”
810    ENTER @Hp8714;Loss
820  !
830  ! Measure the THRU delay
840    OUTPUT @Hp8714;”CALC1:FORM GDEL”
850    OUTPUT @Hp8714;”*opc?”
860    ENTER @Hp8714;Opc
870  !
880    OUTPUT @Hp8714;”calc1:mark1:y?”
890    ENTER @Hp8714;Delay
900  !
910  ! Compute the delay contributed by the THRU
920    Delay=Delay-Open_delay
930  !
940  ! Compute 1-way loss of THROUGH at 1 GHz
950    Loss=Loss/2
960  ! Loss = (dB Loss) x Z0/ (4.3429 x (Delay))
970    Loss=ABS(Loss*Z0/(4.3429*Delay))
980  !
990  ! Store the DELAY into the modified kit definition.
1000 ! Store the LOSS into
1010   OUTPUT @Hp8714;”SENS1:CORR:CKIT:THRU:mod:del “;Delay
1020   OUTPUT @Hp8714;”SENS1:CORR:CKIT:THRU:mod:loss “;Loss
1030 !
1040   PRINT “Done characterizing the THROUGH!”
1050   PRINT “>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>”
1060   PRINT
1070   PRINT “THROUGH standard:”
1080   PRINT “-----------------”
1090   PRINT “Delay =”,Delay
```

```
1100   PRINT “Loss =”,Loss
1110   PRINT “Z0 =”,Z0
1120   PRINT
1130   PRINT “-> Modified kit stored into USER KIT A”
1140   DISP “Done”
1150   OUTPUT @Hp8714;”SENS:CORR:CKIT:SAVE KIT1;:SENS:CORR:COLL:CKIT:PORT1
‘USER1,IMPLIED,IMPLIED,IMPLIED,IMPLIED’”
1160   BEEP
1170 !
1180   END
```

MPCKIT Example Program

This IBASIC program requires an 8712ET/ES or 8714ET/ES analyzer connected to a multiport test set. The program demonstrates how to create and assign user-defined male and female calibration kits with non-zero length throughs to each of four ports on the multiport test set.

```
10   !

20   ! Filename:  MPCKIT
30   !
40   ! Description:
50   !
60   ! This program creates TYPE-N cal kits with non-zero length THRUs
70   ! of 244 pico-seconds.  The kits are assigned to ports 1 thru 4.
80   ! Ports 1,3 are Type-N (f). Ports 2,4 are Type-N(m)
90   !
100  !-------------------------------------------------------------------------
110  COM /Sys_state/ @Hp87xx,Scode
120  CALL Iden_port
130  !
140  ! Select the Type-N (f) mm cal kit for modification
150  OUTPUT @Hp87xx;"SENS:CORR:CKIT:MODIFY TYPENF"
160  OUTPUT @Hp87xx;"SENS1:CORR:CKIT:THRU:MOD:DEL 2.44e-10"
170  !
180  ! Save the modifid kit into USER KIT A, 'KIT1'
190  OUTPUT @Hp87xx;"SENS:CORR:CKIT:SAVE KIT1"
200  !
210  ! Select the Type-N (m) mm cal kit for modification
220  OUTPUT @Hp87xx;"SENS:CORR:CKIT:MODIFY TYPENM"
230  OUTPUT @Hp87xx;"SENS1:CORR:CKIT:THRU:MOD:DEL 2.44e-10"
240  !
250  ! Save the modifid kit into USER KIT B, 'KIT2'
260  OUTPUT @Hp87xx;"SENS:CORR:CKIT:SAVE KIT2"
270  !
280  ! Select 'USER1' kit for ports 1 thru 4
290  OUTPUT @Hp87xx;"SENS:CORR:COLL:CKIT:PORT1
'USER1,IMPLIED,IMPLIED,IMPLIED,IMPLIED'"
300  OUTPUT @Hp87xx;"SENS:CORR:COLL:CKIT:PORT2
'USER2,IMPLIED,IMPLIED,IMPLIED,IMPLIED'"
310  OUTPUT @Hp87xx;"SENS:CORR:COLL:CKIT:PORT3
'USER1,IMPLIED,IMPLIED,IMPLIED,IMPLIED'"
320  OUTPUT @Hp87xx;"SENS:CORR:COLL:CKIT:PORT4
'USER2,IMPLIED,IMPLIED,IMPLIED,IMPLIED'"
330  !
340  DISP "Done ports 1-4: TYPE-N m/f, 244 ps thru"
350  END
360  !
370  !***************************************************************
380  ! Iden_port:   Identify io port to use
390  ! Description: This routines sets up the I/O port address for
400  !              the SCPI interface.  For "HP 87xx" instruments,
410  !              the address assigned to @Hp87xx = 800 otherwise,
420  !              716.
430  !***************************************************************
440  SUB Iden_port
450  COM /Sys_state/ @Hp87xx,Scode
460  !
```

```
470  IF POS(SYSTEM$(“SYSTEM ID”),”HP 87”)<>0 THEN
480  ASSIGN @Hp87xx TO 800
490  Scode=8
500  ELSE
510  ASSIGN @Hp87xx TO 716
520  Scode=7
530  END IF
540  !
550  SUBEND   !Iden_port
```

SIMCAL Example Program

This program demonstrates how to create 2-port error-correction arrays from measurements of the raw (uncorrected) calibration standards when using the analyzer's simcal command.

```
10   ! RE-SAVE “SIMCAL”

20   !-------------------------------------------------
30   ! This example program demonstrates how to use
40   ! raw measurement data to create a 2-Port
50   ! calibration using “simcal” commands.
60   !
70   ! Twelve, (12), raw measurement cal standards are read
80   ! from the analyzer and stored into the arrays
90   ! namded “Std1” to “Std12”.
100  !
110  ! The ordering of raw (uncorrected) stds is:
120  !
130  !  Std1     s11 (THROUGH)
140  !  Std2     s22 (THROUGH)
150  !  Std3     s21 (THROUGH)
160  !  Std4     s12 (THROUGH)
170  !  Std5     s11 (OPEN)
180  !  Std6     s11 (SHORT)
190  !  Std7     s11 (LOAD)
200  !  Std8     s21 isolation (LOAD on both ports)
210  !  Std9     s22 (OPEN)
220  !  Std10    s22 (SHORT)
230  !  Std11    s22 (LOAD)
240  !  Std12    s12 isolation (LOAD on both ports)
250  !
260  !
270  ! These raw measurements are then downloaded back
280  ! to the analyzer.  (In this example, the arrays
290  ! are not modified before being returned to the
300  ! analyzer).
310  !
320  ! The newly downloaded arrays are used to compute
330  ! the 12 2-Port error correction terms.
340  !
350  ! $Revision: $
360  ! $Date:  $
370  !
380  !-------------------------------------------------
390   COM /Testcom/ Pr_flag,@Hp87xx,Is_ibasic
400   REAL Data_201(1:201,1:2)
410   REAL Std1_201(1:201,1:2),Std2_201(1:201,1:2),Std3_201(1:201,1:2)
420   REAL Std4_201(1:201,1:2),Std5_201(1:201,1:2),Std6_201(1:201,1:2)
430   REAL Std7_201(1:201,1:2),Std8_201(1:201,1:2),Std9_201(1:201,1:2)
440   REAL Std10_201(1:201,1:2),Std11_201(1:201,1:2),Std12_201(1:201,1:2)
450  !
460   REAL Std1_801(1:801,1:2),Std2_801(1:801,1:2),Std3_801(1:801,1:2)
470   REAL Std4_801(1:801,1:2),Std5_801(1:801,1:2),Std6_801(1:801,1:2)
480   REAL Std7_801(1:801,1:2),Std8_801(1:801,1:2),Std9_801(1:801,1:2)
490   REAL Std10_801(1:801,1:2),Std11_801(1:801,1:2),Std12_801(1:801,1:2)
500  !
510   DIM Cset$[4]
```

```
520 Start:                                 !
530  !-------------------------------------------------
540  ! FIRST SEE IF WE ARE IBASIC OR RMB
550  ! SET I/O PORT ACCORDINGLY
560  !-------------------------------------------------
570   IF POS(SYSTEM$("SYSTEM ID"),"HP 87")<>0 THEN
580     ASSIGN @Hp87xx TO 800
590     Is_ibasic=1
600     Scode=8
610   ELSE
620     ASSIGN @Hp87xx TO 716
630     CLEAR @Hp87xx
640   END IF
650  !
660   OUTPUT @Hp87xx;"SYST:PRES;*WAI"
670   IF Is_ibasic THEN OUTPUT @Hp87xx;"DISP:PROG LOW"
680   OUTPUT @Hp87xx;"*OPC?"
690   ENTER @Hp87xx;Opc
700   OUTPUT @Hp87xx;"sens:corr:class def2;*opc?"
710   ENTER @Hp87xx;Opc
720  !
730  !-------------------------------------------------
740  ! Setup up Transmission measurement and do a
750  ! full 2-Port calibration.
760  !-------------------------------------------------
770  !
780   OUTPUT @Hp87xx;"SENS1:STAT ON;*WAI"
790   OUTPUT @Hp87xx;"SENS1:FUNC 'XFR:POW:RAT 2,0';DET NBAN;*WAI"
800   OUTPUT @Hp87xx;"SENS1:SWE:POIN 201;*WAI"
810   OUTPUT @Hp87xx;"INIT1:CONT ON;*WAI"
820   DISP "Do a Two Port CAL then press [Continue]"
830   IF Is_ibasic=0 THEN LOCAL @Hp87xx
840   PAUSE
850   OUTPUT @Hp87xx;"*opc?"
860   ENTER @Hp87xx;Opc
870  !
880  !-------------------------------------------------
890  ! Put current data trace into memory and
900  ! then retrieve it to compare when we restore cal
910  ! arrays
920  !-------------------------------------------------
930  !
940   DISP "Putting current data into memory"
950   DISP "getting data trace ..."
960   CALL Autoscale(@Hp87xx,1)
970   OUTPUT @Hp87xx;"TRAC CH1SMEM, CH1SDATA"
980   OUTPUT @Hp87xx;"DISP:WIND1:TRAC1 ON;TRAC2 ON"
990   CALL Get_trace(@Hp87xx,Data_201(*),"TRAC? CH1SDATA")
1000 !
1010 !-------------------------------------------------
1020 ! See what type of cal is now in place
1030 ! DEF = Factory Default
1040 ! FULL = Full Band
1050 ! USER = User Defined
1060 !-------------------------------------------------
1070 !
1080  OUTPUT @Hp87xx;"SENS1:CORR:CSET?"
1090  ENTER @Hp87xx;Cset$
1100  PRINT "CSET? reports you have a "&Cset$&" Transmission Cal"
1110 !-------------------------------------------------
```

```
1120 ! Now get the raw measured standards from the 2-Port cal.
1130 !-------------------------------------------------
1140  DISP “Getting Two-Port cal arrays”
1150  OUTPUT @Hp87xx;”INIT:CONT OFF;*WAI”
1160  Cal_points=201
1170  PRINT Cal_points
1180  SELECT Cal_points
1190  CASE 201
1200    CALL Get_trace(@Hp87xx,Std1_201(*),”trac:corr:sim? std1”)
1210    CALL Get_trace(@Hp87xx,Std2_201(*),”trac:corr:sim? std2”)
1220    CALL Get_trace(@Hp87xx,Std3_201(*),”trac:corr:sim? std3”)
1230    CALL Get_trace(@Hp87xx,Std4_201(*),”trac:corr:sim? std4”)
1240    CALL Get_trace(@Hp87xx,Std5_201(*),”trac:corr:sim? std5”)
1250    CALL Get_trace(@Hp87xx,Std6_201(*),”trac:corr:sim? std6”)
1260    CALL Get_trace(@Hp87xx,Std7_201(*),”trac:corr:sim? std7”)
1270    CALL Get_trace(@Hp87xx,Std8_201(*),”trac:corr:sim? std8”)
1280    CALL Get_trace(@Hp87xx,Std9_201(*),”trac:corr:sim? std9”)
1290    CALL Get_trace(@Hp87xx,Std10_201(*),”trac:corr:sim? std10”)
1300    CALL Get_trace(@Hp87xx,Std11_201(*),”trac:corr:sim? std11”)
1310    CALL Get_trace(@Hp87xx,Std12_201(*),”trac:corr:sim? std12”)
1320  CASE 801
1330    CALL Get_trace(@Hp87xx,Std1_801(*),”trac:corr:sim? std1”)
1340    CALL Get_trace(@Hp87xx,Std2_801(*),”trac:corr:sim? std1”)
1350    CALL Get_trace(@Hp87xx,Std3_801(*),”trac:corr:sim? std2”)
1360    CALL Get_trace(@Hp87xx,Std4_801(*),”trac:corr:sim? std3”)
1370    CALL Get_trace(@Hp87xx,Std5_801(*),”trac:corr:sim? std4”)
1380    CALL Get_trace(@Hp87xx,Std6_801(*),”trac:corr:sim? std5”)
1390    CALL Get_trace(@Hp87xx,Std7_801(*),”trac:corr:sim? std6”)
1400    CALL Get_trace(@Hp87xx,Std8_801(*),”trac:corr:sim? std7”)
1410    CALL Get_trace(@Hp87xx,Std9_801(*),”trac:corr:sim? std8”)
1420    CALL Get_trace(@Hp87xx,Std10_801(*),”trac:corr:sim? std9”)
1430    CALL Get_trace(@Hp87xx,Std11_801(*),”trac:corr:sim? std10”)
1440    CALL Get_trace(@Hp87xx,Std12_801(*),”trac:corr:sim? std11”)
1450  END SELECT
1460  PRINT “Query of 1 Transmission array complete”
1470  PRINT “Number of points per array (real/imag pairs) = “;Cal_points
1480 !
1490 !-------------------------------------------------
1500 ! Restore the default cal display data and memory
1510 ! Just to show the difference, if any.
1520 ! Wait for user to press continue
1530 !-------------------------------------------------
1540  DISP “Press [Continue] to restore default cal”
1550  PAUSE
1560  OUTPUT @Hp87xx;”SENS1:CORR:class DEF2;*WAI”
1570  CALL Autoscale(@Hp87xx,1)
1580  OUTPUT @Hp87xx;”INIT:CONT ON;*WAI”
1590  OUTPUT @Hp87xx;”DISP:WIND1:TRAC1 ON;TRAC2 ON;*WAI”
1600  DISP “Trace with default cal restored - press [Continue]”
1610  PAUSE
1620 !
1630 !-------------------------------------------------
1640 ! Now put the cal standards back
1650 !-------------------------------------------------
1660 !
1670  DISP “Putting back the cal standards ...”
1680  OUTPUT @Hp87xx;”SENS1:SWE:POINTS “&VAL$(Cal_points)&”;*WAI”
1690  SELECT Cal_points
1700  CASE 201
1710    CALL Put_trace(@Hp87xx,Std1_201(*),”trac:corr:sim std1”)
```

```
1720    CALL Put_trace(@Hp87xx,Std2_201(*),”trac:corr:sim std2”)
1730    CALL Put_trace(@Hp87xx,Std3_201(*),”trac:corr:sim std3”)
1740    CALL Put_trace(@Hp87xx,Std4_201(*),”trac:corr:sim std4”)
1750    CALL Put_trace(@Hp87xx,Std5_201(*),”trac:corr:sim std5”)
1760    CALL Put_trace(@Hp87xx,Std6_201(*),”trac:corr:sim std6”)
1770    CALL Put_trace(@Hp87xx,Std7_201(*),”trac:corr:sim std7”)
1780    CALL Put_trace(@Hp87xx,Std8_201(*),”trac:corr:sim std8”)
1790    CALL Put_trace(@Hp87xx,Std9_201(*),”trac:corr:sim std9”)
1800    CALL Put_trace(@Hp87xx,Std10_201(*),”trac:corr:sim std10”)
1810    CALL Put_trace(@Hp87xx,Std11_201(*),”trac:corr:sim std11”)
1820    CALL Put_trace(@Hp87xx,Std12_201(*),”trac:corr:sim std12”)
1830  CASE 801
1840    CALL Put_trace(@Hp87xx,Std1_801(*),”trac:corr:sim std1”)
1850    CALL Put_trace(@Hp87xx,Std2_801(*),”trac:corr:sim std2”)
1860    CALL Put_trace(@Hp87xx,Std3_801(*),”trac:corr:sim std3”)
1870    CALL Put_trace(@Hp87xx,Std4_801(*),”trac:corr:sim std4”)
1880    CALL Put_trace(@Hp87xx,Std5_801(*),”trac:corr:sim std5”)
1890    CALL Put_trace(@Hp87xx,Std6_801(*),”trac:corr:sim std6”)
1900    CALL Put_trace(@Hp87xx,Std7_801(*),”trac:corr:sim std7”)
1910    CALL Put_trace(@Hp87xx,Std8_801(*),”trac:corr:sim std8”)
1920    CALL Put_trace(@Hp87xx,Std9_801(*),”trac:corr:sim std9”)
1930    CALL Put_trace(@Hp87xx,Std10_801(*),”trac:corr:sim std10”)
1940    CALL Put_trace(@Hp87xx,Std11_801(*),”trac:corr:sim std11”)
1950    CALL Put_trace(@Hp87xx,Std12_801(*),”trac:corr:sim std12”)
1960  END SELECT
1970  OUTPUT @Hp87xx;”SENS1:CORR:STATE ON;*WAI”
1980  CALL Se
1990 !-------------------------------------------------
2000 ! Now execute sim cal.  This command calculates
2010 ! the 2-Port error correction arrays.
2020 !-------------------------------------------------
2030  DISP “Trace Data from Sim Cal”
2040  OUTPUT @Hp87xx;”trac:corr:sim:save twoport”
2050 !
2060 !-------------------------------------------------
2070 ! For comparison, we’ll download the original data trace
2080 ! into the memory trace.
2090 ! We’ll restore the number of points to the original
2100 ! number of points.
2110 !
2120 !-------------------------------------------------
2130  OUTPUT @Hp87xx;”SENS1:SWE:POINTS 201;*WAI”
2140  DISP “Restoring orig. data trace to memory ...”
2150  CALL Put_trace(@Hp87xx,Data_201(*),”TRAC CH1SMEM”)
2160 !-------------------------------------------------
2170 ! Now take a sweep with the restored transmission arrays
2180 ! and look at the trace.  Should be the same as the
2190 ! original
2200 !-------------------------------------------------
2210  CALL Autoscale(@Hp87xx,1)
2220  OUTPUT @Hp87xx;”DISP:WIND1:TRAC1 ON;TRAC2 ON”
2230  OUTPUT @Hp87xx;”INIT:CONT ON;*WAI”
2240  DISP “***** Done ****”
2250  END
2260 !
2270 Autoscale:SUB Autoscale(@Na,Ch)
2280 !-------------------------------------------------
2290 ! TAKE A SWEEP AND AUTOSCALE TRACE
2300 !-------------------------------------------------
2310    DIM Disp$[20],Init$[10]
```

```
2320    Disp$=”DISP:WIND”&VAL$(Ch)
2330    Init$=”INIT”&VAL$(Ch)
2340    OUTPUT @Na;Init$&”:CONT OFF;*WAI;:”&Init$&”;*OPC?”
2350    ENTER @Na;Opc
2360    OUTPUT @Na;Disp$&”:TRAC:Y:SCAL:AUTO ONCE”
2370  SUBEND
2380 !------------------------------------------------
2390 Get_trace:SUB Get_trace(@Rfna,REAL Trace_data(*),Command$)
2400 !-------------------------------------------------
2410 ! Get a raw 64 bit float trace from instrument
2420 ! INPUTS:
2430 !   @Rfna    : I/O path of hpib
2440 !   Command$ : SCPI mnemonic used to query trace
2450 ! OUTPUTS:
2460 !   Trace_data : REAL Array of data received from instrument.
2470 !-------------------------------------------------
2480    DIM A$[10]
2490    INTEGER Dig_cnt
2500    OUTPUT @Rfna;”FORM:DATA REAL,64”
2510    OUTPUT @Rfna;”FORM:BORD NORM”
2520    OUTPUT @Rfna;Command$
2530    ASSIGN @Rfna;FORMAT ON
2540    Dig_cnt=-1
2550    ENTER @Rfna USING “%,A,D”;A$,Dig_cnt
2560    IF (A$<>”#”) OR (Dig_cnt<=0) THEN
2570      DISP “Get_trace: Bad block data - aborting”
2580      CLEAR @Rfna
2590    ELSE
2600      ENTER @Rfna USING “%,”&VAL$(Dig_cnt)&”D”;Num_pts
2610      ASSIGN @Rfna;FORMAT OFF
2620      ENTER @Rfna;Trace_data(*)
2630      ASSIGN @Rfna;FORMAT ON
2640      ENTER @Rfna;A$! Read CR/LF
2650    END IF
2660  SUBEND
2670 !-------------------------------------------------
2680 Put_trace:SUB Put_trace(@Rfna,REAL Trace_data(*),Command$)
2690 !-------------------------------------------------
2700 ! Put a raw 64 bit float trace back into  instrument
2710 ! INPUTS:
2720 !   @Rfna    : I/O path of hpib
2730 !   Command$ : SCPI mnemonic output before trace data
2740 !   Trace_data : REAL Array of data to output as REAL,64 format
2750 !-------------------------------------------------
2760    OUTPUT @Rfna;”FORM REAL,64”
2770    OUTPUT @Rfna;”FORM:BORD NORM”
2780    OUTPUT @Rfna;Command$;” , #0”;
2790    ASSIGN @Rfna;FORMAT OFF
2800    OUTPUT @Rfna;Trace_data(*),END
2810    ASSIGN @Rfna;FORMAT ON
2820  SUBEND
2830 Se:SUB Se
2840 !-------------------------------------------------
2850 ! SHOW ERROR, DUMP OUT SCPI ERROR QUEUE
2860 !-------------------------------------------------
2870    COM /Testcom/ Pr_flag,@Hp87xx,Is_ibasic
2880    DIM Errmsg$[200]
2890    INTEGER Errnum
2900    LOOP
2910      OUTPUT @Hp87xx;”SYST:ERR?”
```

```
2920      ENTER @Hp87xx;Errnum,Errmsg$
2930      PRINT Errnum;Errmsg$
2940    EXIT IF Errnum=0
2950    END LOOP
2960  SUBEND
2970 Gq:SUB Gq(Msg$)
2980 !-------------------------------------------------
2990 ! QUICK AND DIRTY SEND QUERY AND GET RESULT
3000 !-------------------------------------------------
3010    COM /Testcom/ Pr_flag,@Hp87xx,Is_ibasic
3020    DIM R$[200]
3030    OUTPUT @Hp87xx;Msg$
3040    ENTER @Hp87xx;R$
3050    PRINT Msg$;” = “;R$
3060  SUBEND
```

Configuring Measurements

SETUP	This program sets up a basic measurement. The example also demonstrates the use of the `*WAI` command.
LIMITEST	This program performs an automatic PASS/FAIL testing with limit lines. The example also demonstrates some methods of combining mnemonics for more efficient programming.
POWERSWP	This program sets up a power sweep measurement.
FOURPARM	This program sets up a four-parameter measurement. It demonstrates how the analyzer can be used to measure four S-parameters with a single sweep. All four S-parameters are displayed. The analyzer calibration is set to two-port calibration.

SETUP Example Program

This program demonstrates how to set up the analyzer to make a basic measurement. The *WAI command is used extensively throughout this program. This has the effect of making sure that the commands are executed in the order they are received. More information about making measurements with the analyzer is available in your analyzer's *User's Guide.*

```
1000 !Filename:  SETUP
1010 !
1020 ! Description:
1030 !   Set Channel 1 to measure filter's transmission.
1040 !   Set Channel 2 to measure filter's reflection
1050 !   Prompt user for start and stop freq, and set them.
1060 !   Take a sweep.
1070 !   Set Scale and Reference levels.
1080 !
1090 !
1100 COM /Sys_state/ @Hp87xx,Scode
1110 ! Identify I/O Port
1120 CALL Iden_port
1130 !
1140 !
1150 ! Preset the instrument.
1160 OUTPUT @Hp87xx;"SYST:PRES;*WAI"
1170 !
1180 ! Configure the analyzer to measure transmission
1190 ! of a filter on channel 1.  This is the command
1200 ! for the BEGIN Filter Transmissn key sequence.
1210 OUTPUT @Hp87xx;"CONF 'FILT:TRAN';*WAI"
1220 !
1230 ! Put the instrument in trigger hold mode.
1240 OUTPUT @Hp87xx;"ABOR;:INIT:CONT OFF;*WAI"
1250 !
1260 ! Turn on channel 2.
1270 OUTPUT @Hp87xx;"SENS2:STAT ON;*WAI"
1280 !
1290 ! Configure channel 2 to measure reflection.  This
1300 ! is the command for the CHAN 2 Reflection key sequence.
1310 OUTPUT @Hp87xx;"SENS2:FUNC 'XFR:POW:RAT 1,0';DET NBAN"
1320 !
1330 ! Wait for the previous commands to complete execution
1340 ! (respond to the *OPC?).
1350 OUTPUT @Hp87xx;"*OPC?"
1360 ENTER @Hp87xx;Opc
1370 !
1380 ! Input a start frequency.
1390 INPUT "Enter Start Frequency (MHz):",Start_f
1400 !
1410 ! Input a stop frequency.
1420 INPUT "Enter Stop Frequency (MHz):",Stop_f
1430 !
1440 ! Set the start and stop frequencies of the analyzer
1450 ! to the values entered.
1460 OUTPUT @Hp87xx;"SENS2:FREQ:STAR";Start_f;"MHz;STOP";Stop_f;"MHz;*WAI"
1470 !
```

```
1480 ! Trigger a single sweep.
1490 OUTPUT @Hp87xx;"INIT;*OPC?"
1500 !
1510 ! Wait for the sweep to be completed.
1520 ENTER @Hp87xx;Opc
1530 !
1540 ! Set up the scale and reference parameters for channel 1.
1550 OUTPUT @Hp87xx;"DISP:WIND1:TRAC:Y:PDIV 10 DB;RLEV 0 DB;RPOS 8"
1560 !
1570 ! Now for channel 2.
1580 OUTPUT @Hp87xx;"DISP:WIND2:TRAC:Y:PDIV 5 DB;RLEV 0 DB;RPOS 8"
1590 !
1600 ! Make channel 1  active (transmission)
1610 OUTPUT @Hp87xx;"SENS1:STAT ON"
1620 !
1630 ! Display the current start and stop frequencies.
1640 DISP "Done measuring.  Start =";Start_f;"MHz     Stop =";Stop_f;"MHz"
1650 END
1660 !

1670 !**************************************************************
1680 ! Iden_port:   Identify io port to use.
1690 ! Description: This routines sets up the I/O port address for
1700 !              the SCPI interface.  For "HP 87xx" instruments,
1710 !              the address assigned to @Hp87xx = 800 otherwise,
1720 !              716.
1730 !**************************************************************
1740 SUB Iden_port
1750     COM /Sys_state/ @Hp87xx,Scode
1760 !
1770     IF POS(SYSTEM$("SYSTEM ID"),"HP 87")<>0 THEN
1780         ASSIGN @Hp87xx TO 800
1790         Scode=8
1800     ELSE
1810         ASSIGN @Hp87xx TO 716
1820         Scode=7
1830     END IF
1840 !
1850 SUBEND !Iden_port
1860 !
```

LIMITEST Example Program

This program demonstrates how to set up and use limit lines over the GPIB. The example device used in this program is the demonstration filter that is shipped with the analyzer. The program sets up the basic measurement, downloads the limit lines, and uses the status registers to determine if the device passes its specifications. For more information about limit lines, refer to the *User's Guide*. For information about using the status registers, refer to “Using the Status Registers” in the *Programmer’s Guide*.

This example also demonstrates how multiple command mnemonics can be combined together. The easiest commands to combine are ones that are closely related on the command tree (such as the start and stop frequency of a limit segment). For more information of command mnemonics, refer to “Introduction to SCPI” in the *Programmer’s Guide*.

```
1000 !Filename:  LIMITEST
1010 !
1020 DIM Title$[30]
1030 !
1040 !
1050 COM /Sys_state/ @Hp87xx,Scode
1060 ! Identify I/O Port
1070 CALL Iden_port
1080 !
1090 ! Perform a system preset; this clears the limit table.
1100 OUTPUT @Hp87xx;"SYST:PRES;*WAI"
1110 !
1120 ! Set up the source frequencies for the measurement.
1130 OUTPUT @Hp87xx;"SENS1:FREQ:STAR 10 MHZ;STOP 400 MHZ;*WAI"
1140 !
1150 ! Set up the receiver for the measurement parameters
1160 ! (Transmission in this case).
1170 OUTPUT @Hp87xx;"SENS1:FUNC 'XFR:POW:RAT 2,0';DET NBAN;*WAI"
1180 !
1190 ! Configure the display so measurement
1200 ! results are easy to see.
1210 OUTPUT @Hp87xx;"DISP:WIND1:TRAC:Y:PDIV 10 DB;RLEV 0 DB;RPOS 9"
1220 !
1230 ! Reduce the distractions on the display by
1240 ! getting rid of notation that will not be
1250 ! needed in this example.
1260 OUTPUT @Hp87xx;"DISP:ANN:YAX OFF"
1270 !
1280 ! Erase the graticule grid for the same reason.
1290 OUTPUT @Hp87xx;"DISP:WIND1:TRAC:GRAT:GRID OFF"
1300 !
1310 ! Create and turn on the first segment for
1320 ! the new limit lines; this one is a maximum
1330 ! limit.
1340 OUTPUT @Hp87xx;"CALC1:LIM:SEGM1:TYPE LMAX;STAT ON"
1350 !
1360 ! Set the amplitude limits for the first limit
```

```
1370 ! segment.
1380 OUTPUT @Hp87xx;"CALC1:LIM:SEGM1:AMPL:STAR -70;STOP -70"
1390 !
1400 ! Set the frequency of the first limit segment.
1410 OUTPUT @Hp87xx;"CALC1:LIM:SEGM1:FREQ:STAR 10 MHZ;STOP 75 MHZ"
1420 !
1430 ! Create and turn on a second maximum limit
1440 ! segment.
1450 OUTPUT @Hp87xx;"CALC1:LIM:SEGM2:TYPE LMAX;STAT ON"
1460 !
1470 ! Set the amplitude limits for segment 2.
1480 OUTPUT @Hp87xx;"CALC1:LIM:SEGM2:AMPL:STAR 0;STOP 0"
1490 !
1500 ! Set the frequency range for segment 2.
1510 OUTPUT @Hp87xx;"CALC1:LIM:SEGM2:FREQ:STAR 145 MHZ;STOP 200 MHZ"
1520 !
1530 ! Create and turn on a third limit segment;
1540 ! this one is a minimum limit.
1550 OUTPUT @Hp87xx;"CALC1:LIM:SEGM3:TYPE LMIN;STAT ON"
1560 !
1570 ! Set the amplitude limits for segment 3.
1580 OUTPUT @Hp87xx;"CALC1:LIM:SEGM3:AMPL:STAR -6;STOP -6"
1590 !
1600 ! Set the frequency range for segment 3.
1610 OUTPUT @Hp87xx;"CALC1:LIM:SEGM3:FREQ:STAR 150 MHZ;STOP 195 MHZ"
1620 !
1630 ! Create and set parameters for segment 4.
1640 OUTPUT @Hp87xx;"CALC1:LIM:SEGM4:TYPE LMAX;STAT ON"
1650 OUTPUT @Hp87xx;"CALC1:LIM:SEGM4:AMPL:STAR -60;STOP -60"
1660 OUTPUT @Hp87xx;"CALC1:LIM:SEGM4:FREQ:STAR 290 MHZ;STOP 400 MHZ"
1670 !
1680 ! Send an operation complete query to ensure that
1690 ! all overlapped commands have been executed.
1700 OUTPUT @Hp87xx;"*OPC?"
1710 !
1720 ! Wait for the reply.
1730 ENTER @Hp87xx;Opc
1740 !
1750 ! Turn on the display of the limit lines.
1760 OUTPUT @Hp87xx;"CALC1:LIM:DISP ON"
1770 !
1780 ! Turn on the pass/fail testing; watch the
1790 ! analyzer's display for the pass/fail indicator.
1800 OUTPUT @Hp87xx;"CALC1:LIM:STAT ON"
1810 !
1820 ! Take a controlled sweep to ensure that
1830 ! there is real data present for the limit test.
1840 OUTPUT @Hp87xx;"ABOR;:INIT1:CONT OFF;:INIT1;*WAI"
1850 !
1860 ! Query the limit fail condition register to see
1870 ! if there is a failure.
1880 OUTPUT @Hp87xx;"STAT:QUES:LIM:COND?"
1890 !
1900 ! Read the register's contents.
1910 ENTER @Hp87xx;Fail_flag
1920 !
1930 ! Bit 0 is the test result for channel 1 while
1940 ! bit 1 is the results for channel 2 limit testing.
1950 IF BIT(Fail_flag,0)=1 THEN
1960 !
```

```
1970 ! In case of failure, give additional direction
1980 ! to the operator using the title strings.
1990     Title$="Limit Test FAIL - Tune device"
2000 !
2010 ! Turn on the title string.
2020     OUTPUT @Hp87xx;"DISP:ANN:TITL1:DATA '"&Title$&"';STAT ON"
2030 !
2040 ! Turn on continuous sweep mode for tuning.
2050     OUTPUT @Hp87xx;"INIT1:CONT ON;*WAI"
2060 !
2070 ! Loop while the tuning is taking place.
2080     LOOP
2090 !
2100 ! Monitor the status of the limit fail
2110 ! condition register.
2120         OUTPUT @Hp87xx;"STAT:QUES:LIM:COND?"
2130         ENTER @Hp87xx;Fail_flag
2140 !
2150 ! Check the limit fail bit.  Exit if the
2160 ! device has been tuned to pass the test.
2170     EXIT IF BIT(Fail_flag,0)=0
2180     END LOOP
2190 END IF
2200 !
2210 ! Turn off the prompt to the operator and
2220 ! return the analyzer to the continuously
2230 ! sweeping mode.
2240 OUTPUT @Hp87xx;"DISP:ANN:TITL1 OFF;:INIT:CONT ON;*WAI"
2250 END
2260 !
2270 !***************************************************************
2280 ! Iden_port:   Identify io port to use.
2290 ! Description: This routines sets up the I/O port address for
2300 !              the SCPI interface.  For "HP 87xx" instruments,
2310 !              the address assigned to @Hp87xx = 800 otherwise,
2320 !              716.
2330 !***************************************************************

2340 SUB Iden_port
2350     COM /Sys_state/ @Hp87xx,Scode
2360 !
2370     IF POS(SYSTEM$("SYSTEM ID"),"HP 87")<>0 THEN
2380         ASSIGN @Hp87xx TO 800
2390         Scode=8
2400     ELSE
2410         ASSIGN @Hp87xx TO 716
2420         Scode=7
2430     END IF
2440 !
2450 SUBEND !Iden_port
2460 !
```

POWERSWP Example Program

This program demonstrates how to set up a power sweep. It shows how to query the instrument to determine its power sweep ranges, how to place a marker at a given stimulus power value (x-axis), how to read the measured power at a marker (y-axis), and how to read an entire trace of power sweep data.

```
1000 ! Filename:  POWERSWEEP
1010 !_
1020 !
1030 !  Description: Query the power sweep ranges,
1040 !  take a power sweep, and use markers to read
1050 !  gain at marker.  Then query the trace.
1060 !_
1070 DIM Trace_pwr(1:201)
1080 !
1090 COM /Sys_state/ @Hp87xx,Scode
1100 ! Identify I/O Port
1110 CALL Iden_port
1120 !
1130 !
1140 !
1150 ! Initialize the 871x; set to power sweep mode.
1160 ! Set CW mode and freq.
1170 !
1175 OUTPUT @Hp87xx;"SYST:PRESET;*OPC?"
1176 ENTER @Hp87xx;Opc
1180 OUTPUT @Hp87xx;"SENS1:FUNC 'XFR:POW 2';DET BBAN;*WAI"
1190 OUTPUT @Hp87xx;"SENS1:FREQ:CENT 1 GHZ;*WAI"
1200 ! Note that CW mode is set before Power-sweep mode
1210 OUTPUT @Hp87xx;"DISP:ANN:FREQ1:MODE CW;:SENS:FREQ:SPAN 0 HZ;*WAI"
1220 OUTPUT @Hp87xx;"POWER:MODE SWEEP;*WAI"
1230 !
1240 !-
1250 ! Determine the Min/Max power settings for each
1260 ! attenuator range.
1270 !
1280 FOR Atten=0 TO 60 STEP 10
1290     OUTPUT @Hp87xx;"SOUR:POW:RANG ATT"&VAL$(Atten)&";*WAI"
1300     OUTPUT @Hp87xx;"SOUR:POW:STAR? MIN"
1310     ENTER @Hp87xx;Pwr_min
1320     OUTPUT @Hp87xx;"SOUR:POW:STAR? MAX"
1330     ENTER @Hp87xx;Pwr_max
1340     PRINT "Atten: ";Atten;" Min: ";Pwr_min;" Max: ";Pwr_max
1350 NEXT Atten
1360 !
1370 !-
1380 ! Find the optimum power sweep range,
1390 ! defined as being that range for which either:
1400 ! 1) Both the desired Start and Stop Power levels may be set,
1410 ! 2) The desired Start Power may be set and the
1420 !    Power Range is maximized.
1425 ! Then, modify next 3 lines of code to get desired settings.
1430 !
1440 ! Set Start and Stop power levels for power sweep.
1450 !
```

```
1460 OUTPUT @Hp87xx;"SOUR:POW:RANG ATT0;*WAI"
1470 OUTPUT @Hp87xx;"SOUR:POW:STAR -2 DBM;*WAI"
1480 OUTPUT @Hp87xx;"SOUR:POW:STOP 4 DBM;*WAI"
1490 ! Take one sweep, wait till done
1500 OUTPUT @Hp87xx;"ABOR;:INIT1:CONT OFF;:INIT1;*OPC?"
1510 ENTER @Hp87xx;Opc
1520 !-
1530 ! Read marker, display power in, power out, gain.
1540 ! Note that the
1550 !   X-axis is swept output power from the source,
1560 !   Y-axis is power measured by the receiver.
1570 !
1580 OUTPUT @Hp87xx;"CALC1:MARK1 ON"
1590 ! Set marker to start power, wait till done.
1600 OUTPUT @Hp87xx;"CALC1:MARK:X -2;*OPC?"
1610 ENTER @Hp87xx;Opc
1620 ! Read Marker Source power level and measured power.
1630 OUTPUT @Hp87xx;"CALC1:MARK1:X?"
1640 ENTER @Hp87xx;Pwr_src
1650 OUTPUT @Hp87xx;"CALC1:MARK1:Y?"
1660 ENTER @Hp87xx;Pwr_meas
1670 ! Read entire trace array.
1680 OUTPUT @Hp87xx;"FORM:DATA ASC,3"
1690 OUTPUT @Hp87xx;"TRAC? CH1FDATA"
1700 ENTER @Hp87xx;Trace_pwr(*)
1710 !
1720 PRINT "Source Power @ Marker = "&VAL$(Pwr_src)&"dBm"
1730 PRINT "Received Power @ Marker = "&VAL$(Pwr_meas)&"dBm"
1740 PRINT "Gain @ Marker = "&VAL$(Pwr_meas-Pwr_src)&"dB"
1750 PRINT "Power Sweep Trace Point #1:  "&VAL$(Trace_pwr(1))&"dBm"
1760 END
1770 !
1780 !**************************************************************
1790 ! Iden_port:   Identify io port to use.
1800 ! Description: This routines sets up the I/O port address for
1810 !              the SCPI interface.  For "HP 87xx" instruments,
1820 !              the address assigned to @Hp87xx = 800 otherwise,
1830 !              716.
1840 !**************************************************************
1850 SUB Iden_port
1860     COM /Sys_state/ @Hp87xx,Scode
1870 !
1880     IF POS(SYSTEM$("SYSTEM ID"),"HP 87")<>0 THEN
1890         ASSIGN @Hp87xx TO 800
1900         Scode=8
1910         OUTPUT @Hp87xx;"DISP:PROG LOWer"
1920     ELSE
1930         ASSIGN @Hp87xx TO 716
1940         Scode=7
1950     END IF
1960 !
1970 SUBEND !Iden_port
1980 !
```

FOURPARM Example Program

```
1000 !Filename:  FOURPARM
1010 !
1020 ! Demonstrates that all four S-parameters can be computed by
1030 ! the instrument after a single sweep.
1040 !
1050 ! Outline:
1060 !  1. Select Default 2-Port Cal
1070 !  2. Select S11 on MEAS 1.  S21 on MEAS 2.
1080 !     These are Port 1 Reflection and forward transmission.
1090 !     Take a sweep.
1100 !  3. Do Data->Mem on MEAS 1 and on MEAS 2.
1110 !  4. Without taking another sweep, select
1120 !     S22 on MEAS 1 and S12 on MEAS 2.
1130 !     These are Port 2 Reflection and reverse transmission.
1140 !
1150 !  The display will show:
1160 !
1170 !  MEAS 1 : Port 1 and Port 2 REFLECTION      (S11 and S22)
1180 !  MEAS 2 : forward and reverse TRANSMISSION  (S21 and S12)
1190 !
1200 !  Note that a single MEAS channel could be used to gather all four
1210 !  S-parameters after a single sweep.  For display convenience
1220 !  however, this program uses both MEAS channels.
1230 !
1240 DIM A$[10],Data1(1:201)
1250 INTEGER Digits,Bytes
1260 COM /Sys_state/ @Hp87xx,Scode
1270 !
1280 CALL Iden_port                                ! Identify I/O Port
1290 !
1300 OUTPUT @Hp87xx;”SYST:PRES”                    ! System Preset
1310 OUTPUT @Hp87xx;”*OPC?”
1320 ENTER @Hp87xx;Opc_flag                        ! Operation complete
1330 !
1340 ! Select Default Two Port Calibration on MEAS 1 and 2
1350 !
1360 DISP “Enabling 2-Port Calibration...”
1370 OUTPUT @Hp87xx;”SENS2:STAT ON”                 ! Turn on MEAS 2
1380 OUTPUT @Hp87xx;”SENS1:FUNC ‘XFR:S 1,1’”        ! MEAS 1 = S11
1390 OUTPUT @Hp87xx;”SENS2:FUNC ‘XFR:S 2,1’”        ! MEAS 2 = S21
1400 OUTPUT @Hp87xx;”SENS1:CORR:CLASS:SEL DEFAULT2” ! Default 2-Port
1410 OUTPUT @Hp87xx;”SENS2:CORR:CLASS:SEL DEFAULT2” ! Default 2-Port
1420 OUTPUT @Hp87xx;”DISP:FORM ULOW”                ! Split Display
1430 OUTPUT @Hp87xx;”*OPC?”
1440 ENTER @Hp87xx;Opc_flag                         ! Operation complete
1450 DISP ““
1460 !
1470 ! Take a single sweep, leaving the analyzer
1480 ! in trigger hold mode.
1490 !
1500 OUTPUT @Hp87xx;”ABOR;:INIT1:CONT OFF;:INIT1;*WAI”
1510 !
1520 ! Copy MEAS 1 Data (S11) into MEAS 1 Memory trace
1530 ! Copy MEAS 2 Data (S21) into MEAS 2 Memory trace
1540 !
1550 DISP “Copy S11 to Mem1”
1560 OUTPUT @Hp87xx;”TRAC CH1SMEM,CH1SDATA”         ! Ch1 Data->Mem
```

```
1570 DISP “Copy S21 to Mem2”
1580 OUTPUT @Hp87xx;”TRAC CH2SMEM,CH2SDATA”         ! Ch2 Data-> Mem
1590 OUTPUT @Hp87xx;”*OPC?”
1600 ENTER @Hp87xx;Opc_flag
1610 !
1620 ! Now select S22 on MEAS 1 and S12 on MEAS 2.
1630 !
1640 DISP “Meas S22 on ch1”
1650 OUTPUT @Hp87xx;”SENS1:FUNC ‘XFR:S 2,2’”  ! MEAS 1 = S22
1660 OUTPUT @Hp87xx;”*OPC?”
1670 ENTER @Hp87xx;Opc_flag                   ! Opeation complete.
1680 !
1690 DISP “Meas S12 on ch2”
1700 OUTPUT @Hp87xx;”SENS2:FUNC ‘XFR:S 1,2’”  ! MEAS 2 = S12
1710 OUTPUT @Hp87xx;”*OPC?”
1720 ENTER @Hp87xx;Opc_flag                   ! Opeation complete.
1730 !
1740 ! Display Data and Memory on both MEAS channels.  Autoscale result.
1750 !
1760 OUTPUT @Hp87xx;”CALC1:MATH (IMPL);:DISP:WIND1:TRAC1 ON;TRAC2 ON”
1770 OUTPUT @Hp87xx;”DISP:WIND1:TRAC:Y:AUTO ONCE”
1780 OUTPUT @Hp87xx;”CALC2:MATH (IMPL);:DISP:WIND2:TRAC1 ON;TRAC2 ON”
1790 OUTPUT @Hp87xx;”DISP:WIND2:TRAC:Y:AUTO ONCE”
1800 !
1810 DISP “Ch1: S11 (M:S22)   Ch2: S12 (M:S21)”     ! Done
1820 BEEP
1830 !
1840 END
1850 !
1860 !
1870 !**************************************************************
1880 ! Iden_port:   Identify io port to use
1890 ! Description: This routines sets up the I/O port address for
1900 !              the SCPI interface.  For “HP 87xx” instruments,
1910 !              the address assigned to @Hp87xx = 800 otherwise,
1920 !              716.
1930 !**************************************************************
1940 SUB Iden_port
1950   COM /Sys_state/ @Hp87xx,Scode
1960 !
1970   IF POS(SYSTEM$(“SYSTEM ID”),”HP 87”)<>0 THEN
1980       ASSIGN @Hp87xx TO 800
1990       Scode=8
2000   ELSE
2010       ASSIGN @Hp87xx TO 716
2020       Scode=7
2030   END IF
2040 !
2050 SUBEND !Iden_port
2060 !
```

Customizing the Display and Using Graphics

DRAW871X	This program draws the analyzer and a DUT to the full screen IBASIC display partition. The drawing can be scaled to fit the application. This program uses the analyzer's graphics commands for drawing. To see another example program using the IBASIC drawing commands, see the section called "BARCODE Example Program" on page 2-12.
GRAPHICS	This program uses graphics and softkeys to create customized procedures. The example demonstrates the use of some of the user-graphics commands including the one to erase a previously drawn line. It also demonstrates use of the softkeys and detecting a front panel keypress with the service request interrupt process.
GRAPH2	This program uses graphics to draw an instrument and DUT onto the display.
GETPLOT	This program reads an HPGL graphics file.

DRAW871X Example Program

```
 1 !-----------------------------------------------------
 2 !
 3 ! IBASIC program:  DRAW871X - Drawing setup diagrams
 4 !
 5 ! This program draws the HP 871X network analyzer
 6 ! and a device under test to the full screen IBASIC
 7 ! display partition.  The drawing can be scaled to
 8 ! fit the application.  Setting the scale factor to
 9 ! 1.0 creates a drawing of about 400 pixels wide
10 ! (1/2 screen width) and 100 pixels high (1/3 screen
11 ! height).
12 !
13 !-----------------------------------------------------
14 !
15 ! Setup an I/O path name for the internal bus and
16 ! declare variables.
17 !
18 INTEGER X0,Y0
19 REAL Scale
20 !  Make @Hp87xx common to all subroutines
21 COM /Sys_state/ @Hp87xx,Scode
22 ! Identify the computer we are running on
23 ! and assign the i/o port address to @Hp87xx
24 CALL Iden_port
25 !
26 !
27 ! Preset the analyzer and wait until it is done.
28 !
29 OUTPUT @Hp87xx;”SYST:PRES;*OPC?”
30 ENTER @Hp87xx;Opc
31 ! Allocate the full screen as an IBASIC display
32 ! and clear the graphics buffer.
33 !
34 OUTPUT @Hp87xx;”DISP:PROG FULL”
35 OUTPUT @Hp87xx;”DISP:WIND10:GRAP:CLEAR”
36 !
37 ! Rescale the display window for the new VGA display
38 output @Hp87xx;”DISP:WIND10:GRAP:SCAL 0,1023,0,383”
39 !
40 ! Setup the origin and scale parameters for the
41 ! drawing.  Draw the network analyzer and dut.
42 !
43 X0=100
44 Y0=100
45 Scale=1.
46 CALL Draw_na(X0,Y0,Scale)
47 CALL Draw_dut(X0,Y0,Scale)
48 END
49 !-----------------------------------------------------
50 SUB Draw_na(INTEGER X0,INTEGER Y0,REAL Sc)
51 !---------------------------------------------------
52 !
53 ! This subroutine draws the HP 8711 at origin X0,Y0
54 ! and scale Sc.  The drawing is done to the IBASIC
55 ! display (window 10) using the HP 8711’s user
56 ! graphics commands.
57 !
```

```
  58 !-----------------------------------------------------
  59 COM /Sys_state/ @Hp87xx,Scode
  60 OUTPUT @Hp87xx;"DISP:WIND10:GRAP:MOVE "&VAL$(X0)&","&VAL$(Y0)
  61 OUTPUT @Hp87xx;"DISP:WIND10:GRAP:RECT "&VAL$(INT(Sc*350))&","&VAL$(INT(Sc*100))
  62 OUTPUT @Hp87xx;"DISP:WIND10:GRAP:MOVE
"&VAL$(X0+INT(Sc*10))&","&VAL$(Y0+INT(Sc*10))
  63 OUTPUT @Hp87xx;"DISP:WIND10:GRAP:RECT "&VAL$(INT(Sc*180))&","&VAL$(INT(Sc*80))
  64 OUTPUT @Hp87xx;"DISP:WIND10:GRAP:MOVE
"&VAL$(X0+INT(Sc*200))&","&VAL$(Y0+INT(Sc*80))
  65 OUTPUT @Hp87xx;"DISP:WIND10:GRAP:RECT "&VAL$(INT(Sc*15))&","&VAL$(INT(Sc*8))
  66 OUTPUT @Hp87xx;"DISP:WIND10:GRAP:MOVE
"&VAL$(X0+INT(Sc*200))&","&VAL$(Y0+INT(Sc*70))
  67 OUTPUT @Hp87xx;"DISP:WIND10:GRAP:RECT "&VAL$(INT(Sc*15))&","&VAL$(INT(Sc*8))
  68 OUTPUT @Hp87xx;"DISP:WIND10:GRAP:MOVE
"&VAL$(X0+INT(Sc*200))&","&VAL$(Y0+INT(Sc*60))
  69 OUTPUT @Hp87xx;"DISP:WIND10:GRAP:RECT "&VAL$(INT(Sc*15))&","&VAL$(INT(Sc*8))
  70 OUTPUT @Hp87xx;"DISP:WIND10:GRAP:MOVE
"&VAL$(X0+INT(Sc*200))&","&VAL$(Y0+INT(Sc*50))
  71 OUTPUT @Hp87xx;"DISP:WIND10:GRAP:RECT "&VAL$(INT(Sc*15))&","&VAL$(INT(Sc*8))
  72 OUTPUT @Hp87xx;"DISP:WIND10:GRAP:MOVE
"&VAL$(X0+INT(Sc*200))&","&VAL$(Y0+INT(Sc*40))
  73 OUTPUT @Hp87xx;"DISP:WIND10:GRAP:RECT "&VAL$(INT(Sc*15))&","&VAL$(INT(Sc*8))
  74 OUTPUT @Hp87xx;"DISP:WIND10:GRAP:MOVE
"&VAL$(X0+INT(Sc*200))&","&VAL$(Y0+INT(Sc*30))
  75 OUTPUT @Hp87xx;"DISP:WIND10:GRAP:RECT "&VAL$(INT(Sc*15))&","&VAL$(INT(Sc*8))
  76 OUTPUT @Hp87xx;"DISP:WIND10:GRAP:MOVE
"&VAL$(X0+INT(Sc*200))&","&VAL$(Y0+INT(Sc*20))
  77 OUTPUT @Hp87xx;"DISP:WIND10:GRAP:RECT "&VAL$(INT(Sc*15))&","&VAL$(INT(Sc*8))
  78 OUTPUT @Hp87xx;"DISP:WIND10:GRAP:MOVE
"&VAL$(X0+INT(Sc*200))&","&VAL$(Y0+INT(Sc*10))
  79 OUTPUT @Hp87xx;"DISP:WIND10:GRAP:RECT "&VAL$(INT(Sc*15))&","&VAL$(INT(Sc*8))
  80 OUTPUT @Hp87xx;"DISP:WIND10:GRAP:MOVE
"&VAL$(X0+INT(Sc*265))&","&VAL$(Y0+INT(Sc*80))
  81 OUTPUT @Hp87xx;"DISP:WIND10:GRAP:RECT "&VAL$(INT(Sc*70))&","&VAL$(INT(Sc*13))
  82 OUTPUT @Hp87xx;"DISP:WIND10:GRAP:MOVE
"&VAL$(X0+INT(Sc*230))&","&VAL$(Y0+INT(Sc*81))
  83 OUTPUT @Hp87xx;"DISP:WIND10:GRAP:RECT "&VAL$(INT(Sc*20))&","&VAL$(INT(Sc*10))
  84 OUTPUT @Hp87xx;"DISP:WIND10:GRAP:MOVE
"&VAL$(X0+INT(Sc*275))&","&VAL$(Y0+INT(Sc*85))
  85 OUTPUT @Hp87xx;"DISP:WIND10:GRAP:RECT "&VAL$(INT(Sc*50))&","&VAL$(INT(Sc*3))
  86 OUTPUT @Hp87xx;"DISP:WIND10:GRAP:MOVE
"&VAL$(X0+INT(Sc*295))&","&VAL$(Y0+INT(Sc*50))
  87 OUTPUT @Hp87xx;"DISP:WIND10:GRAP:CIRC "&VAL$(INT(Sc*8))
  88 OUTPUT @Hp87xx;"DISP:WIND10:GRAP:MOVE
"&VAL$(X0+INT(Sc*245))&","&VAL$(Y0+INT(Sc*15))
  89 OUTPUT @Hp87xx;"DISP:WIND10:GRAP:CIRC "&VAL$(INT(Sc*4))
  90 OUTPUT @Hp87xx;"DISP:WIND10:GRAP:MOVE
"&VAL$(X0+INT(Sc*325))&","&VAL$(Y0+INT(Sc*15))
  91 OUTPUT @Hp87xx;"DISP:WIND10:GRAP:CIRC "&VAL$(INT(Sc*4))
  92 SUBEND
  93 !-------------------------------------------------------
  94 SUB Draw_dut(INTEGER X0,INTEGER Y0,REAL Sc)
  95 COM /Sys_state/ @Hp87xx,Scode
  96 !-----------------------------------------------------
  97 !
  98 ! This subprogram draws a device under test (dut)
  99 ! and connects it to the HP 8711 that was drawn
 100 ! with an origin at X0,Y0 and a scale of Sc.
 101 !
 102 !-----------------------------------------------------
 103 OUTPUT @Hp87xx;"DISP:WIND10:GRAP:MOVE
"&VAL$(X0+INT(Sc*245))&","&VAL$(Y0+INT(Sc*15))
 104 OUTPUT @Hp87xx;"DISP:WIND10:GRAP:DRAW
"&VAL$(X0+INT(Sc*245))&","&VAL$(Y0-INT(Sc*20))
 105 OUTPUT @Hp87xx;"DISP:WIND10:GRAP:DRAW
"&VAL$(X0+INT(Sc*265))&","&VAL$(Y0-INT(Sc*20))
 106 OUTPUT @Hp87xx;"DISP:WIND10:GRAP:MOVE
"&VAL$(X0+INT(Sc*265))&","&VAL$(Y0-INT(Sc*22))
 107 OUTPUT @Hp87xx;"DISP:WIND10:GRAP:RECT "&VAL$(INT(Sc*40))&","&VAL$(INT(Sc*4))
 108 OUTPUT @Hp87xx;"DISP:WIND10:GRAP:MOVE
"&VAL$(X0+INT(Sc*305))&","&VAL$(Y0-INT(Sc*20))
 109 OUTPUT @Hp87xx;"DISP:WIND10:GRAP:DRAW
"&VAL$(X0+INT(Sc*325))&","&VAL$(Y0-INT(Sc*20))
 110 OUTPUT @Hp87xx;"DISP:WIND10:GRAP:DRAW
"&VAL$(X0+INT(Sc*325))&","&VAL$(Y0+INT(Sc*15))
 111 SUBEND
 112 !
 113 !**************************************************************
 114 ! Iden_port:  Identify io port to use
 115 !**************************************************************
 116 SUB Iden_port
 117 COM /Sys_state/ @Hp87xx,Scode
 118 !
 119 IF POS(SYSTEM$("SYSTEM ID"),"HP 87")<>0 THEN
 120 ASSIGN @Hp87xx TO 800
 121 Scode=8
 122 ELSE
 123 ASSIGN @Hp87xx TO 716
 124 Scode=7
 125 END IF
 126 !
 127 SUBEND !Iden_port
 128 !
```

GRAPHICS Example Program

This program demonstrates how to use the analyzer's user graphics commands to draw setup diagrams. It also demonstrates how to generate a service request in response to a keyboard interrupt. More information on user-graphics commands is available in the *Programmer's Guide*.

Note that this program uses the analyzer's user graphics commands. Graphics are easily implemented using BASIC commands such as `POLYGON` and `RECTANGLE`. For an additional example, see the section called “BARCODE Example Program” on page 2-12.

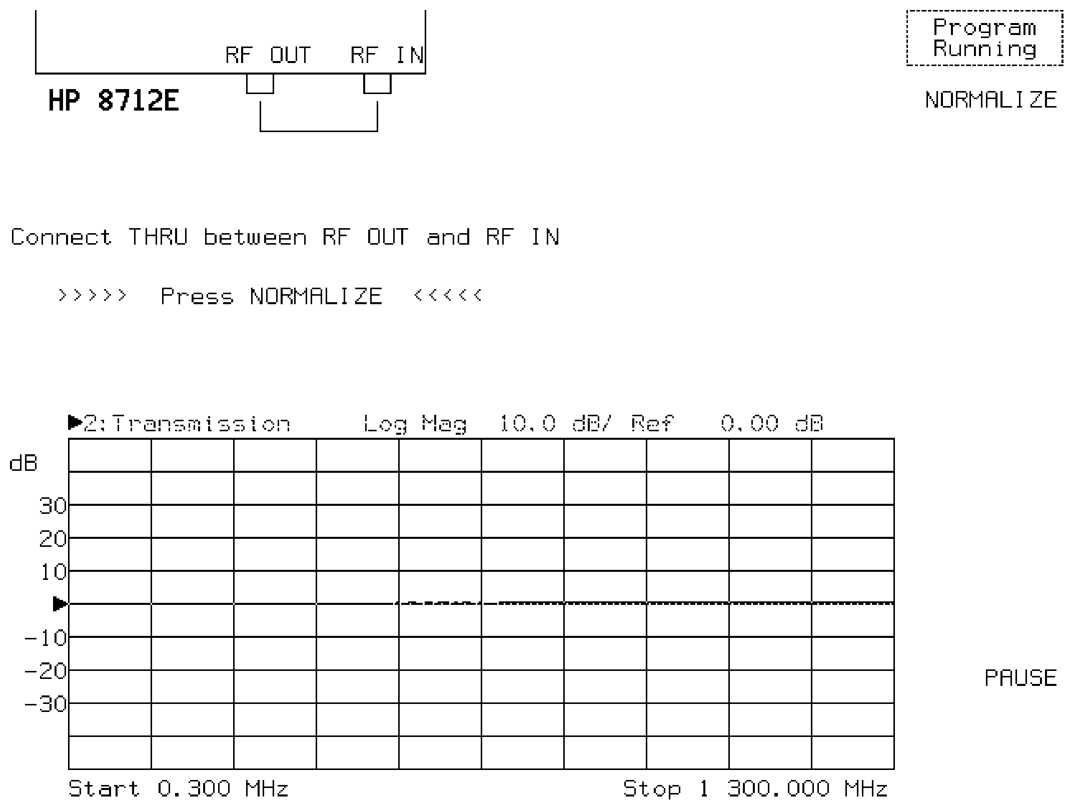
Lines `170-240` draw and label a representation of an analyzer for a connection diagram. This example is a simple front view from the top.

Lines `250-450` draw the connection needed for a normalization. The operator is prompted to make this connection and to press a softkey on the instrument. A flashing message is used to attract attention.

Lines `460-580` perform the normalization, erase the prompts (without erasing the whole screen) and prepare for the test.

Lines `590-730` are a branching routine that handles the service request generated interrupts used by the external controller.

Figure 2-1 GRAPHICS example connection diagram



```
HP 8712E          RF OUT    RF IN                 Program
                                                  Running

                                                  NORMALIZE

Connect THRU between RF OUT and RF IN

    >>>>>   Press NORMALIZE   <<<<<

▶2:Transmission     Log Mag   10.0 dB/  Ref    0.00 dB
```

```
1      ! Filename:  GRAPHICS
2      !
3      ! Description: Draws a simple connection diagram
4      ! in the IBASIC window, and displays a softkey.
5      !
6      ! NOTE:  This program works properly ONLY
7      ! when option 1C2, IBASIC, has been installed.
8      ! Refer to program GRAPH2 if no IBASIC option.
9      !
10     IF POS(SYSTEM$("SYSTEM ID"),"HP 871") THEN
20       ASSIGN @Hp8711 TO 800
30       Internal=1
40       Isc=8
50     ELSE
60       ASSIGN @Hp8711 TO 716
70       Internal=0
80       Isc=7
90       ABORT 7
100      CLEAR 716
110    END IF
111    !
112    ! Allocate an IBASIC display partition
113    ! to show the graphics.
120    OUTPUT @Hp8711;"DISP:PROG UPP"
121    !
122    ! Clear the IBASIC display partition.
130    OUTPUT @Hp8711;"DISP:WIND10:GRAP:CLE"
131    !
132    ! Turn on channel 2 for measurements.  The
133    ! lower part of the display is
134    ! devoted to display of measurements.
140    OUTPUT @Hp8711;"SENS2:STAT ON;*WAI"
141    !
142    ! Take a single controlled sweep to ensure
143    ! a valid measurement using *OPC query.
150    OUTPUT @Hp8711;"ABOR;:INIT2:CONT OFF;:INIT2;*OPC?"
160    ENTER @Hp8711;Opc
161    !
162    ! Select the bright "pen" and bold font.
170    OUTPUT @Hp8711;"DISP:WIND10:GRAP:COL 1;LAB:FONT BOLD"
171    !
172    ! Draw a label reading "HP 8711C" at 45 pixels
173    ! to the right and 120 pixels above the origin.
174    ! The origin is the lower left corner of the
175    ! current graphics window (upper half).
180    OUTPUT @Hp8711;"DISP:WIND10:GRAP:MOVE 45,120
       ;LAB 'HP 8711C'"
181    !
182    ! Draw a box to represent the analyzer.
190    OUTPUT @Hp8711;"DISP:WIND10:GRAP:MOVE 30,175
       ;DRAW 30,140;DRAW 480,140;DRAW 480,175"
191    !
192    ! Draw a box to represent the REFLECTION RF OUT port.
200    OUTPUT @Hp8711;"DISP:WIND10:GRAP:MOVE 275,140
        ;DRAW 275,130;DRAW 305,130;DRAW 305,140"
201    ! Draw a box to represent the TRANSMISSION RF IN port.
210    OUTPUT @Hp8711;"DISP:WIND10:GRAP:MOVE 410,140
       ;DRAW 410,130;DRAW 440,130;DRAW 440,140"
211    ! Change the text font to small, which is the
212    ! same as that used for PRINT or DISP statements.
```

```
220   OUTPUT @Hp8711;"DISP:WIND10:GRAP:LAB:FONT SMAL"
221   !
222   ! Label the RF OUT port.
230   OUTPUT @Hp8711;"DISP:WIND10:GRAP:MOVE 250,145
      ;LAB 'RF OUT'"
231   !
232   ! Label the RF IN port.
240   OUTPUT @Hp8711;"DISP:WIND10:GRAP:MOVE 395,145
      ;LAB 'RF IN'"
241   !
250 Normalize: !
251   !
252   ! Draw a through connection between the RF OUT
253   ! and RF IN ports.
260   OUTPUT @Hp8711;"DISP:WIND10:GRAP:MOVE 290,125
      ;DRAW 290,110;DRAW 425,110;DRAW 425,125"
261   ! Prompt the operator to connect the through.
270   OUTPUT @Hp8711;"DISP:WIND10:GRAP:MOVE 1,50
      ;LAB 'Connect THRU between RF OUT and RF IN'"
280   IF Internal=1 THEN
281     ! If using the IBASIC (internal) controller,
282     ! then use the "ON KEY" method to handle
283     ! user interface.
290     ON KEY 1 LABEL "NORMALIZE" RECOVER Norm
300   ELSE
301     ! If using an external controller...
302     !
303     ! Initialize flag for checking on keyboard
304     ! interrupts.
310     Keycode=-1
311     !
312     ! Label softkey 1.
320     OUTPUT @Hp8711;"DISP:MENU:KEY1 'NORMALIZE'"
321     !
322     ! Clear the status register and event status
323     ! register.
330     OUTPUT @Hp8711;"*CLS;*ESE 0"
331     !
332     ! Preset the other status registers.
333     ! Enable the Device Status register to report
334     ! to the Status Byte on positive transition
335     ! of bit 0 (key press).  Enable the Status
336     ! Byte to generate an interrupt when the
337     ! Device Status register's summary bit
338     ! changes.
340     OUTPUT @Hp8711;"STAT:PRES;DEV:ENAB 1;*SRE 4"
341     !
342     ! Clear the key queue to ensure that previous
343     ! key presses  do not generate an interrupt.
350     OUTPUT @Hp8711;"SYST:KEY:QUE:CLE"
351     !
352     ! Set up and enable the interrupt on the HP-IB
353     ! when a service request is received.
360     ON INTR Isc,5 RECOVER Srq
370     ENABLE INTR Isc;2
380   END IF
381   !
382   ! Turn off the graphics buffer.
390   OUTPUT @Hp8711;"DISP:WIND10:GRAP:BUFF OFF"
391   !
```

```
392   ! Loop for waiting for press of the NORMALIZE key.
393   ! The two different output statements along with
394   ! the wait statements create a blinking effect.
395   ! There is not exit from this loop other than
396   ! a keyboard interrupt.
400   LOOP
410     OUTPUT @Hp8711;"DISP:WIND10:GRAP:MOVE 55,18
        ;LAB '  Press NORMALIZE  '"
420     WAIT .2
430     OUTPUT @Hp8711;"DISP:WIND10:GRAP:MOVE 55,18
        ;LAB '       Press NORMALIZE       '"
440     WAIT .2
450   END LOOP
451   !
460 Norm: ! Entry point to wait for a key press.
461   !
462   ! If wrong key pressed, return to Normalize.
470   IF Keycode&

0 THEN GOTO Normalize
480   OFF KEY
481   !
482   ! The through should now be connected and
483   ! ready to measure.
484   !
485   ! Turn the graphics buffer back on.
490   OUTPUT @Hp8711;"DISP:WIND10:GRAP:BUFF ON"
491   !
492   ! Select the "erase" pen (pen color 0) and
493   ! erase the prompts.
500   OUTPUT @Hp8711;"DISP:WIND10:GRAP:COL 0;MOVE 55,18
      ;LAB '  Press NORMALIZE  '"
510   OUTPUT @Hp8711;"DISP:WIND10:GRAP:MOVE 1,50
      ;LAB 'Connect THRU between RF OUT and RF IN'"
520   OUTPUT @Hp8711;"DISP:MENU:KEY1 '         '"
521   !
522   ! Display the active data trace only.  Turn off
523   ! any previous normalization.
530   OUTPUT @Hp8711;"CALC2:MATH (IMPL)"
531   !
532   ! Take a single sweep on channel 2.
540   OUTPUT @Hp8711;"INIT2;*WAI"
541   !
542   ! Copy the new data trace into the memory array.
550   OUTPUT @Hp8711;"TRAC CH2SMEM,CH2SDATA"
551   !
552   ! Normalize; that is, display the active data
553   ! relative to the memory trace.
560   OUTPUT @Hp8711;"CALC2:MATH (IMPL/CH2SMEM)"
561   !
562   ! Display only one of the traces (the normalized
563   ! trace).
570   OUTPUT @Hp8711;"DISP:WIND2:TRAC1 ON;TRAC2 OFF"
571   !
572   ! Erase the through connect and select pen color 1 again.
580   OUTPUT @Hp8711;"DISP:WIND10:GRAP:MOVE 290,110
      ;DRAW 425,110;DRAW 425,125;COL 1"
590   STOP
600   !
610 Srq: ! This is the branching routine that handles
```

```
           service request
611       ! generated interrupts.
612     !
613     ! Do a serial poll to find out if analyzer generated the
614     ! interrupt.
620     Stb=SPOLL(@Hp8711)
621     !
622     ! Determine if the Device Status register's summary
623     ! bit (bit 2 of the Status Byte) has been set.
630     IF BINAND(Stb,4)&

0 THEN
631       !
632       ! If so, then get the Device Status Register contents.
640       OUTPUT @Hp8711;"STAT:DEV:EVEN?"
650       ENTER @Hp8711;Dev_event
651       !
652       ! Check for key press...
660       IF BINAND(Dev_event,1)&

0 THEN
661         ! If so, then determine which key.
670         OUTPUT @Hp8711;"SYST:KEY?"
680         ENTER @Hp8711;Keycode
690       END IF
700     END IF
701     !
702     ! Reenable the interrupt in case wrong key
703     ! was pressed.
710     ENABLE INTR Isc
720     GOTO Norm
730     END
```

GRAPH2 Example Program

This program demonstrates simple graphics and softkey handling. If the program is run from an external computer, it also demonstrates basic interrupts (SRQ) and status register handling. The program displays a hookup diagram and prompts the user to connect a cable. Once the cable is connected, the user is prompted to press a "NORMALIZE" key. The analyzer then performs the normalization and erases the hookup diagram.

```
1000 ! Filename:  GRAPH2
1010 !
1020 ! Description: Draws a simple connection diagram
1030 ! in the IBASIC window, and displays a softkey.
1040 !
1050 ! NOTE:  This program works properly ONLY
1060 ! when option 1C2, IBASIC, has been installed.
1070 ! Modify to use DISP:WIND1 if no IBASIC option.
1080 !
1090 !
1100 COM /Sys_state/ @Hp87xx,Scode
1110 ! Identify I/O Port
1120 CALL Iden_port
1130 !
1135   output @Hp87xx;"DISP:WIND1:GRAP:SCAL 0,1023,0,383"
1140 !
1150 ! Allocate an IBASIC display partition
1160 ! to show the graphics.
1170 OUTPUT @Hp87xx;"DISP:FORM ULOW"
1180 OUTPUT @Hp87xx;"DISP:WIND1:TRAC:GRAT:GRID OFF"
1190 !
1200 ! Clear the IBASIC display partition.
1210 OUTPUT @Hp87xx;"DISP:WIND1:GRAP:CLE"
1220 !
1230 ! Turn on channel 2 for measurements.  The
1240 ! lower part of the display is
1250 ! devoted to display of measurements.
1260 OUTPUT @Hp87xx;"SENS2:STAT ON;*WAI"
1270 !
1280 ! Take a single controlled sweep to ensure
1290 ! a valid measurement using *OPC query.
1300 OUTPUT @Hp87xx;"ABOR;:INIT2:CONT OFF;:INIT2;*OPC?"
1310 ENTER @Hp87xx;Opc
1320 !
1330 ! Select the bright "pen" and bold font.
1340 OUTPUT @Hp87xx;"DISP:WIND1:GRAP:COL 1;LAB:FONT BOLD"
1350 !
1360 ! Draw a label reading "HP 8711C" at 45 pixels
1370 ! to the right and 120 pixels above the origin.
1380 ! The origin is the lower left corner of the
1390 ! current graphics window (upper half).
1400 OUTPUT @Hp87xx;"DISP:WIND1:GRAP:MOVE 45,120;LAB 'HP 8711C'"
1410 !
1420 ! Draw a box to represent the analyzer.
1430 OUTPUT @Hp87xx;"DISP:WIND1:GRAP:MOVE 30,175;DRAW 30,140;DRAW
     480,140;DRAW 480,175"
```

```
1440 !
1450 ! Draw a box to represent the REFLECTION RF OUT port.
1460 OUTPUT @Hp87xx;"DISP:WIND1:GRAP:MOVE 275,140;DRAW 275,130;DRAW
     305,130;DRAW 305,140"
1470 ! Draw a box to represent the TRANSMISSION RF IN port.
1480 OUTPUT @Hp87xx;"DISP:WIND1:GRAP:MOVE 410,140;DRAW 410,130;DRAW
     440,130;DRAW 440,140"
1490 ! Change the text font to small, which is the
1500 ! same as that used for PRINT or DISP statements.
1510 OUTPUT @Hp87xx;"DISP:WIND1:GRAP:LAB:FONT SMAL"
1520 !
1530 ! Label the RF OUT port.
1540 OUTPUT @Hp87xx;"DISP:WIND1:GRAP:MOVE 250,145;LAB 'RF OUT'"
1550 !
1560 ! Label the RF IN port.
1570 OUTPUT @Hp87xx;"DISP:WIND1:GRAP:MOVE 380,145;LAB 'RF IN'"
1580 !
1590 Normalize: !
1600 !
1610 ! Draw a through connection between the RF OUT
1620 ! and RF IN ports.
1630 OUTPUT @Hp87xx;"DISP:WIND1:GRAP:MOVE 290,125;DRAW 290,110;DRAW
     425,110;DRAW 425,125"
1640 ! Prompt the operator to connect the through.
1650 OUTPUT @Hp87xx;"DISP:WIND1:GRAP:MOVE 1,50;LAB 'Connect THRU between RF
     OUT and RF IN'"
1660 IF Internal=1 THEN
1670 ! If using the IBASIC (internal) controller,
1680 ! then use the "ON KEY" method to handle
1690 ! user interface.
1700     ON KEY 1 LABEL "NORMALIZE" RECOVER Norm
1710 ELSE
1720 ! If using an external controller...
1730 !
1740 ! Initialize flag for checking on keyboard
1750 ! interrupts.
1760     Keycode=-1
1770 !
1780 ! Label softkey 1.
1790     OUTPUT @Hp87xx;"DISP:MENU:KEY1 'NORMALIZE'"
1800 !
1810 ! Clear the status register and event status
1820 ! register.
1830     OUTPUT @Hp87xx;"*CLS;*ESE 0"
1840 !
1850 ! Preset the other status registers.
1860 ! Enable the Device Status register to report
1870 ! to the Status Byte on positive transition
1880 ! of bit 0 (key press).  Enable the Status
1890 ! Byte to generate an interrupt when the
1900 ! Device Status register's summary bit
1910 ! changes.
1920     OUTPUT @Hp87xx;"STAT:PRES;DEV:ENAB 1;*SRE 4"
1930 !
1940 ! Clear the key queue to ensure that previous
1950 ! key presses  do not generate an interrupt.
1960     OUTPUT @Hp87xx;"SYST:KEY:QUE:CLE"
1970 !
1980 ! Set up and enable the interrupt on the HP-IB
1990 ! when a service request is received.
```

```
2000     ON INTR Scode,5 RECOVER Srq
2010     ENABLE INTR Scode;2
2020 END IF
2030 !
2040 ! Turn off the graphics buffer.
2050 OUTPUT @Hp87xx;"DISP:WIND1:GRAP:BUFF OFF"
2060 !
2070 ! Loop for waiting for press of the NORMALIZE key.
2080 ! The two different output statements along with
2090 ! the wait statements create a blinking effect.
2100 ! There is not exit from this loop other than
2110 ! a keyboard interrupt.
2120 LOOP
2130     OUTPUT @Hp87xx;"DISP:WIND1:GRAP:MOVE 55,18;LAB '>>>>>  Press
         NORMALIZE  <<<<<'"
2140     WAIT .2
2150     OUTPUT @Hp87xx;"DISP:WIND1:GRAP:MOVE 55,18;LAB '       Press
         NORMALIZE       '"
2160     WAIT .2
2170 END LOOP
2180 !
2190 Norm: ! Entry point to wait for a key press.
2200 !
2210 ! If wrong key pressed, return to Normalize.
2220 IF Keycode<>0 THEN GOTO Normalize
2230 OFF KEY
2240 !
2250 ! The through should now be connected and
2260 ! ready to measure.
2270 !
2280 ! Turn the graphics buffer back on.
2290 OUTPUT @Hp87xx;"DISP:WIND1:GRAP:BUFF ON"
2300 !
2310 ! Select the "erase" pen (pen color 0) and
2320 ! erase the prompts.
2330 OUTPUT @Hp87xx;"DISP:WIND1:GRAP:COL 0;MOVE 55,18;LAB '>>>>>  Press
     NORMALIZE  <<<<<'"
2340 OUTPUT @Hp87xx;"DISP:WIND1:GRAP:MOVE 1,50;LAB 'Connect THRU between RF
     OUT and RF IN'"
2350 OUTPUT @Hp87xx;"DISP:MENU:KEY1 '          '"
2360 !
2370 ! Display the active data trace only.  Turn off
2380 ! any previous normalization.
2390 OUTPUT @Hp87xx;"CALC2:MATH (IMPL)"
2400 !
2410 ! Take a single sweep on channel 2.
2420 OUTPUT @Hp87xx;"INIT2;*WAI"
2430 !
2440 ! Copy the new data trace into the memory array.
2450 OUTPUT @Hp87xx;"TRAC CH2SMEM,CH2SDATA"
2460 !
2470 ! Normalize; that is, display the active data
2480 ! relative to the memory trace.
2490 OUTPUT @Hp87xx;"CALC2:MATH (IMPL/CH2SMEM)"
2500 !
2510 ! Display only one of the traces (the normalized
2520 ! trace).
2530 OUTPUT @Hp87xx;"DISP:WIND2:TRAC1 ON;TRAC2 OFF"
2540 !
2550 ! Erase the through connect and select pen color 1 again.
```

```
2560 OUTPUT @Hp87xx;"DISP:WIND1:GRAP:MOVE 290,125;DRAW 290,110;DRAW
     425,110;DRAW 425,125"
2570 STOP
2580 !
2590 Srq: ! This is the branching routine that handles service request
2600 ! generated interrupts.
2610 !
2620 ! Do a serial poll to find out if analyzer generated the
2630 ! interrupt.
2640 Stb=SPOLL(@Hp87xx)
2650 !
2660 ! Determine if the Device Status register's summary
2670 ! bit (bit 2 of the Status Byte) has been set.
2680 IF BINAND(Stb,4)<>0 THEN
2690 !
2700 ! If so, then get the Device Status Register contents.
2710     OUTPUT @Hp87xx;"STAT:DEV:EVEN?"
2720     ENTER @Hp87xx;Dev_event
2730 !
2740 ! Check for key press...
2750     IF BINAND(Dev_event,1)<>0 THEN
2760 ! If so, then determine which key.
2770         OUTPUT @Hp87xx;"SYST:KEY?"
2780         ENTER @Hp87xx;Keycode
2790     END IF
2800 END IF
2810 !
2820 ! Reenable the interrupt in case wrong key.
2830 ! was pressed.
2840 ENABLE INTR Scode
2850 GOTO Norm
2860 END
2870 !
2880 !**************************************************************
2890 ! Iden_port:   Identify io port to use.
2900 ! Description: This routines sets up the I/O port address for
2910 !              the SCPI interface.  For "HP 87xx" instruments,
2920 !              the address assigned to @Hp87xx = 800 otherwise,
2930 !              716.
2940 !**************************************************************
2950 SUB Iden_port
2960     COM /Sys_state/ @Hp87xx,Scode
2970 !
2980     IF POS(SYSTEM$("SYSTEM ID"),"HP 87")<>0 THEN
2990         ASSIGN @Hp87xx TO 800
3000         Scode=8
3010     ELSE
3020         ASSIGN @Hp87xx TO 716
3030         Scode=7
3040     END IF
3050 !
3060 SUBEND !Iden_port
3070 !
```

GETPLOT Example Program

This program shows how to capture a screen plot in HP-GL (.hgl) format and transfer it to the analyzer's internal 3.5" disk. Although this capability now exists in firmware, this program is still useful in demonstrating file manipulation and storage. This program also allows the user to specify any filename, whereas the firmware will always choose a predefined name. This can also be used for .pcx format by un-commenting line 280.

Because of BASIC limitations in any single array size, a four element array is used in line 170 to store the complete file. This allows a file size up to 128,000 bytes.

Lines 320-340 enter the screen capture data into the array Blk$.

Lines 360-380 determine the total number of bytes to be saved.

Lines 410-480 save the file. In line 420, any previous file of the same name is erased. If no file exists, this line is ignored due to the ON ERROR statement in line 410. The file is created in line 450 and the data stored in line 470.

```
100 !GETPLOT
110 !
120 !  This program will get a hardcopy screen dump in HP-GL form
at from
130 !  the 8711, and store it locally.
140 !  The user specifies the local filename (default = Plot871x)
150 !
160 !
170 DIM Blk$(1:4)[32000]  ! Max file size = 4 * 32000 = 128000 by
tes
180 !
190 DIM Filename$[64],Dest$[64]
200 INTEGER Word1
210 !
220 COM /Sys_state/ @Hp87xx,Scode
230 ! Identify I/O Port
240 CALL Iden_port
250 !
260 BEEP
270 Filename$="DATA:screen.hgl"   ! HP-GL format
280 ! Filename$="DATA:screen.pcx"  ! PCX format
290 Dest$="Plot871x"
300 INPUT "Enter host filename (default='Plot871x')",Dest$
310 DISP "READING FILE "&Filename$&" ..."
320 OUTPUT @Hp87xx;"MMEM:TRANSFER? '"&Filename$&"'"
330 ENTER @Hp87xx USING "#,W";Word1   ! Assume indefinite block:
#0 header
340 ENTER @Hp87xx USING "%,-K";Blk$(*)
350 ! Compute length of data we just ENTERed
360 FOR I=1 TO 4
370     Filelength=LEN(Blk$(I))+Filelength
```

```
380 NEXT I
390 ! Save data to local file
400 DISP "Creating new file..."
410 ON ERROR GOTO Save_file
420 PURGE Dest$
430 Save_file:     !
440 OFF ERROR
450 CREATE Dest$,Filelength
460 ASSIGN @File TO Dest$;FORMAT ON
470 OUTPUT @File;Blk$(*);
480 ASSIGN @File TO *
490 DISP "File "&Dest$&" created."
500 BEEP
510 END
520 !
530 !************************************************************
**
540 ! Iden_port:   Identify io port to use.
550 ! Description: This routines sets up the I/O port address for
560 !              the SCPI interface.  For "HP 87xx" instruments
,
570 !              the address assigned to @Hp87xx = 800 otherwis
e,
580 !              716.
590 !************************************************************
**
```

```
600 SUB Iden_port
610     COM /Sys_state/ @Hp87xx,Scode
620 !
630     IF POS(SYSTEM$("SYSTEM ID"),"HP 87")<>0 THEN
640         ASSIGN @Hp87xx TO 800
650         Scode=8
660     ELSE
670         ASSIGN @Hp87xx TO 716
680         Scode=7
690     END IF
700 !
710 SUBEND !Iden_port
720 !
```

Using an External Controller

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

TRICTRL This example program demonstrates how an external controller can be used with two instruments running IBASIC. Run this program on an external controller. Connect two analyzers via GPIB cables to the external controller. Set one instrument to address 16, set the other instrument to address 18.

This program insures that only the analyzer (acting as a controller) or the local IBASIC is sending SCPI commands at one time. This is one possible implementation of synchronizing the analyzer and a controller. Refer to the *Automating Measurements User's Guide Supplement* for more information.

The external controller is responsible for downloading the IBASIC program to each analyzer. The external controller sets the status reporting to send an SRQ whenever a user requested service request occurs.

When all instrument configuration has completed, the external controller sends a “run program” command to each analyzer and then goes into an idle loop. The external controller remains in the idle loop until either instrument sends an SRQ.

While the external controller is idle, each instrument can freely send various SCPI commands. Each instrument may ask for service by triggering an SRQ. Once an SRQ has been triggered, the instrument must remain in an idle loop, until the external controller indicates it is done servicing the SRQ. This is done using the program variable “Ctlr_flag”. The flag is cleared when the external controller is done and has returned to its idle loop.

DUALCTRL Example Program

```
10    !----------------------------------------------------------
20    !
30    ! BASIC program:  DUALCTRL - Two controller operation
40    !
50    ! This program is designed to run on an external
60    ! controller.  It demonstrates how the external
70    ! controller and HP IBASIC can work together.  The
80    ! program downloads an IBASIC program to the HP 871X
90    ! and runs it twice.  After each run, two program
100   ! variables are read from the analyzer and displayed.
110   !
120   !----------------------------------------------------------
130   !
140   ! Initialize the variables for the interface select
150   ! code and the HP-IB address of the HP 871X.
160   !
170   Scode=7
180   Address=16
190   Na=Scode*100+Address
200   !
210   ! Prepare the analyzer for remote operation, clear
220   ! the analyzer’s input/output queues, the display
230   ! and scratch any program in the buffer.
240   !
250   CLEAR Na
260   CLEAR SCREEN
270   OUTPUT Na;”PROG:DEL:ALL”
280   !
290   !  Download the program as an indefinite block length
300   !  data transfer, terminate the data transfer by
310   !  sending a carriage return and EOI.
320   !
330   DISP “Downloading the program...”
340   ASSIGN @Prog TO Na
350   OUTPUT @Prog;”PROG:DEF #0”;
360   OUTPUT @Prog;”10 COM INTEGER Times_run,Test$[10]”
370   OUTPUT @Prog;”20 Times_run=Times_run+1”
380   OUTPUT @Prog;”30 IF Times_run=1 THEN Test$=””PASS”””
390   OUTPUT @Prog;”40 IF Times_run=2 THEN Test$=””FAIL”””
400   OUTPUT @Prog;”50 FOR I= 1 TO 20”
410   OUTPUT @Prog;”60 BEEP”
420   OUTPUT @Prog;”70 NEXT I”
430   OUTPUT @Prog;”80 END”
440   OUTPUT @Prog;CHR$(10) END
450   !
460   ! Initialize interrupt registers - clear the status byte,
470   ! the service request enable register, the standard event
480   ! enable register, and preset the other status registers.
490   !
500   OUTPUT Na;”*CLS”
510   OUTPUT Na;”*SRE 0”
520   OUTPUT Na;”*ESE 0”
530   OUTPUT Na;”STAT:PRES”
540   !
550   ! Set up the status registers to generate an interrupt
560   ! on negative transition of the Program Running bit
570   ! (bit 14 in the Operational Status register).
```

```
580   !
590   OUTPUT Na;”STAT:OPER:NTR #HFFFF”
600   OUTPUT Na;”STAT:OPER:ENAB 16384”
610   OUTPUT Na;”*CLS”
620   OUTPUT Na;”*SRE 128”
630   !
640   ! Run the program, read and display the variables.
650   !
660   DISP “Running the program...”
670   OUTPUT Na;”PROG:EXEC ‘RUN’”
680   Display_res(Na,Scode)
690   OUTPUT Na;”PROG:EXEC ‘RUN’”
700   Display_res(Na,Scode)
710   !
720   ! Return the analyzer to front panel control, this
730   ! is the end of the program.
740   !
750   LOCAL Na
760   DISP “DONE !”
770   END
780   !
790   !-------------------------------------------------------
800   !
810   SUB Display_res(Na,Scode)
820     !-----------------------------------------------------
830     !
840     ! This subprogram waits for an SRQ interrupt to
850     ! signal that an IBASIC program running on the
860     ! analyzer has finished.  It then reads and clears
870     ! the HP-IB status registers.  The values of two
880     ! program variables are then read and displayed.
890     !
900     !-----------------------------------------------------
910     !
920     ! Setup branching to an interrupt handling routine,
930     ! enable the interrupts and wait until one occurs.
940     !
950     ON INTR Scode GOTO Read_results
960     ENABLE INTR Scode;2
970 Idle: GOTO Idle
980 Read_results: !
990     !
1000    ! The program has finished running - read and clear
1010    ! the operational status register and status byte.
1020    !
1030    A=SPOLL(Na)
1040    OUTPUT Na;”STAT:OPER:EVEN?”
1050    ENTER Na;Event
1060    OUTPUT Na;”*CLS”
1070    !
1080    ! Read a numeric variable (Times_run) and a string
1090    ! variable (Test$) and display the values.
1100    !
1110    OUTPUT Na;”PROG:NUMB? ‘Times_run’”
1120    ENTER Na USING “X,K”;Times_run
1130    OUTPUT Na;”PROG:STR? ‘Test$’”
1140    ENTER Na USING “X,K”;Test$
1150    DISP “Times_run: “;Times_run,”Test$: “;Test$
1160    PRINT “Times_run: “;Times_run,”Test$: “;Test$
1170  SUBEND
```

TRICTRL Example Program

```
10   !-----------------------------------------------------
20   !
30   ! BASIC program:  TRICTRL - Three controller operation
40   !   One controller, Two IBASIC instruments
50   !
60   ! This program is designed to run on an external
70   ! controller.  It demonstrates how the external
80   ! controller and multiple instruments running IBASIC
90   ! programs can be synchronized to work together.
100  !
110  ! Run this program on an external controller. Two HP871x
120  ! are needed.  Set one HP871x to address 16.  Set the
130  ! other to address 18.  Connect HP-IB cables between
140  ! the controller and the two analyzers.
150  !
160  ! The program downloads IBASIC programs to two HP 871Xs,
170  ! then runs each program.  Pressing softkey 1 on either
180  ! intstrument triggers a sweep.  Pressing softkey 3 on
190  ! either instrument will trigger an SRQ.  The controller
200  ! will poll the instrument over the HP-IB bus, determine
210  ! which instrument has requested service, log the SRQ,
220  ! and release the instrument for more measurements by
230  ! setting the IBASIC variable Ctrl_flag.
240  !-----------------------------------------------------
250  !
260  ! Initialize the variables for the interface select
270  ! code and the HP-IB address of the HP 871X.
280  !
290   Scode=7
300   Address1=16
310   Address2=18
320   Na1=Scode*100+Address1
330   Na2=Scode*100+Address2
340   Dev_count1=0
350   Dev_count2=0
360  !
370  ! Prepare the analyzer for remote operation, clear
380  ! the analyzer’s input/output queues and scratch
390  ! any program in the buffer.
400  !
410   ABORT 7
420   CLEAR Na1
430   CLEAR Na2
440   CLEAR SCREEN
450  !
460   OUTPUT Na1;”SYST:PRES;*OPC?”           ! Preset analyzer #1
470   ENTER Na1;Opc
480   OUTPUT Na2;”SYST:PRES;*OPC?”           ! Preset analyzer #2
490   ENTER Na2;Opc
500  !
510   OUTPUT Na1;”PROG:STAT STOP”            ! Stop all programs
520   REMOTE Na1
530   OUTPUT Na1;”PROG:DEL:ALL”              ! Scratch the programs
540   OUTPUT Na2;”PROG:STAT STOP”
550   REMOTE Na2
560   OUTPUT Na2;”PROG:DEL:ALL”
570  !
```

```
580   ! Initialize interrupt registers - clear the status byte,
590   ! the service request enable register, the standard event
600   ! enable register, and preset the other status registers.
610   !
620    OUTPUT Na1;"*CLS"
630    OUTPUT Na1;"*SRE 0"
640    OUTPUT Na1;"*ESE 0"
650    OUTPUT Na1;"STAT:PRES;*OPC?"
660    ENTER Na1;Opc
670   !
680    OUTPUT Na2;"*CLS"
690    OUTPUT Na2;"*SRE 0"
700    OUTPUT Na2;"*ESE 0"
710    OUTPUT Na2;"STAT:PRES;*OPC?"
720    ENTER Na2;Opc
730   !
740    ON INTR 7,2 GOSUB User_srq        ! Define the SRQ service routine
750    GOSUB Usermask                    ! Enable the user SRQ
760    ENABLE INTR 7;2
770   !
780   !  Download the program as an indefinite block length
790   !  data transfer, terminate the data transfer by
800   !  sending a carriage return and EOI.
810   !
820    DISP "Downloading the programs..."
830    ASSIGN @Prog TO Na1
840    GOSUB Dnld
850    ASSIGN @Prog TO Na2
860    GOSUB Dnld
870   !
880   ! Run the programs
890    DISP "Running the programs..."
900    OUTPUT Na1;"PROG:STAT RUN;*OPC?"
910    ENTER Na1;Opc
920   !
930    OUTPUT Na2;"PROG:STAT RUN;*OPC?"
940    ENTER Na2;Opc
950   !
960    BEEP
970    DISP "Waiting for srq..."
980   !
990    LOCAL Na1
1000   LOCAL Na2
1010  Idle:GOTO Idle
1020   STOP
1030  ! Enable SRQs to occur when the user_srq bit is set
1040  Usermask:        !
1050   OUTPUT Na1;"*ESE 64;*SRE 32"
1060   OUTPUT Na1;"*OPC?"
1070   ENTER Na1;Opc
1080   OUTPUT Na2;"*ESE 64;*SRE 32"
1090   OUTPUT Na2;"*OPC?"
1100   ENTER Na2;Opc
1110   RETURN
1120  !
1130  User_srq:     ! This routine is called to service SRQs
1140   Stb=SPOLL(Na1)        ! Poll the first instrument
1150   IF (BINAND(Stb,64)<>0) THEN
1160     OUTPUT Na1;"*ESR?"
1170     ENTER Na1;Stat
```

```
1180    Dev_count1=Dev_count1+1
1190    PRINT “Inst:”,Na1,”Dev:”,Dev_count1
1200    OUTPUT Na1;”PROG:NUMB ‘Ctlr_flag’,0”  ! Clear the IBASIC flag
1210    LOCAL Na1
1220  ELSE
1230 !
1240    Stb=SPOLL(Na2)      ! Poll the second instrument
1250    IF (BINAND(Stb,64)<>0) THEN
1260      OUTPUT Na2;”*ESR?”
1270      ENTER Na2;Stat
1280      Dev_count2=Dev_count2+1
1290      PRINT “Inst:”,Na2,”Dev:”,Dev_count2
1300      OUTPUT Na2;”PROG:NUMB ‘Ctlr_flag’,0”  ! Clear the IBASIC flag
1310      LOCAL Na2
1320    END IF
1330  END IF
1340 !
1350  ENABLE INTR 7
1360  RETURN
1370 !
1380 Dnld: ! Download example program to analyzer
1390  OUTPUT @Prog;”PROG:DEF #0”;
1400  OUTPUT @Prog;”10 COM INTEGER Ctlr_flag”
1410  OUTPUT @Prog;”20 OUTPUT 800;””ABOR;:INIT1:CONT OFF”””
1420  OUTPUT @Prog;”30 ON KEY 1 LABEL ““Test 1”” GOSUB Do_test”
1430  OUTPUT @Prog;”40 ON KEY 3 LABEL ““Done Test”” GOSUB Send_srq”
1440  OUTPUT @Prog;”50 Idle:GOTO Idle”
1450  OUTPUT @Prog;”60 STOP”
1460  OUTPUT @Prog;”70 Send_srq: !”
1470  OUTPUT @Prog;”80 BEEP”
1480  OUTPUT @Prog;”90 Ctlr_flag=1”
1490  OUTPUT @Prog;”100 OUTPUT 800;””SYST:KEY:USER”””
1500  OUTPUT @Prog;”110 DISP ““Waiting for CTLR...”””
1510  OUTPUT @Prog;”120 Stall: IF Ctlr_flag=1 THEN GOTO Stall”
1520  OUTPUT @Prog;”130 DISP ““ ”””
1530  OUTPUT @Prog;”140 RETURN”
1540  OUTPUT @Prog;”150 DO_TEST: OUTPUT 800;””INIT1;*OPC?”””
1550  OUTPUT @Prog;”160 ENTER 800;Opc”
1560  OUTPUT @Prog;”170 RETURN”
1570  OUTPUT @Prog;”180 END”
1580  OUTPUT @Prog;CHR$(10) END
1590  OUTPUT @Prog;”*opc?”
1600  ENTER @Prog;Opc
1610  RETURN
1620  END
1630 !
```

Making Fault Location Measurements (Option 100 Only)

FAULT	This programs shows the effects of various fault location frequency modes on a cable measurement.
USR_FLOC	This program shows how to simplify fault location measurements by using the **User BEGIN** key.

FAULT Example Program

```
1000 ! Filename: FAULT (Option 100 only)
1010 !
1020 ! This program is designed to show the affects of the various
1030 ! fault location frequency modes on a cable measurement.
1040 !
1050 ! Connect a 50 m. (150ft) cable to the RF out of the analyzer.
1060 ! (if available).
1070 !
1080 ! The program steps through various settings.
1090 !
1100 ! Set Feet/Meters
1110 ! Start Distance   0   meters
1120 ! Stop Distance    100 meters
1130 ! Low Pass mode
1140 ! Band Pass mode   CF  = 600 MHz
1150 ! Band Pass mode   CF  = 900 MHz
1160 ! Low Pass mode
1170 ! Cable Loss
1180 ! Cable Velocity Factor
1190 !
1200 ! The commands which cause changes to frequency settings will
1210 ! cause the analyzer to automatically display a caution message
1220 ! to verify Cable Loss and Velocity Factor.
1230 !
1240 !
1250 COM /Sys_state/ @Hp87xx,Scode
1260 ! Identify I/O Port
1270 CALL Iden_port
1280 !
1290 ! Preset the analyzer
1300 OUTPUT @Hp87xx;"SYST:PRES; *OPC?"
1310 ENTER @Hp87xx;Opc
1320 !
1330 ! Enable fault location measurment on channel 1
1340 OUTPUT @Hp87xx;"SENS1:STAT ON; *WAI"
1350 OUTPUT @Hp87xx;"SENS1:FUNC 'FLOC 1,0';DET NBAN; *OPC?"
1360 ENTER @Hp87xx;Opc
1370 WAIT 2
1380 !
1390 !  Autoscale the fault measurment
1400 OUTPUT @Hp87xx;"DISP:WIND1:TRAC:Y:AUTO ONCE"
1410 !
1420 Clear_disp
1430 Disp_mess("Fault Location Demo...")
1440 WAIT 3
1450 !
1460 ! Reset the cable loss and velocity factor
1470 OUTPUT @Hp87xx;"SENS1:CORR:LOSS:COAX .0"
1480 OUTPUT @Hp87xx;"SENS1:CORR:RVEL:COAX 1."
1490 !
1500 Clear_disp
1510 Disp_mess("Setting units to Meters")
1520 !
1530 ! Set the units to read in METERS
1540 OUTPUT @Hp87xx;"SENS:DIST:UNIT MET"
1550 !OUTPUT @Hp87xx;"SENS:DIST:UNIT FEET"  ! Display units in feet
1560 WAIT 5
```

```
1570 !
1580 !
1590 Clear_disp
1600 Disp_mess("Setting Start and Stop Distance")
1610 !
1620 ! Set the start distance to 0.
1630 OUTPUT @Hp87xx;"SENS1:DIST:STAR 0; *WAI"
1640 !
1650 ! Set the stop distance to 100.
1660 OUTPUT @Hp87xx;"SENS1:DIST:STOP 100; *WAI"
1670 !
1680 ! Send an operation complete query to ensure that
1690 ! all overlapped commands have been executed.
1700 OUTPUT @Hp87xx;"*OPC?"
1710 !
1720 ! Wait for the reply.
1730 ENTER @Hp87xx;Opc
1740 !
1750 WAIT 10
1760 !
1770 ! Change to Band pass mode
1780 OUTPUT @Hp87xx;"SENS:FREQ:MODE CENT; *WAI"
1790 !
1800 Clear_disp
1810 Disp_mess("Setting CF to 600 MHz. Band Pass")
1820 !
1830 ! Set Center Frequency to 600 MHz
1840 OUTPUT @Hp87xx;"SENS1:FREQ:CENT 600000000 HZ;*WAI"
1850 WAIT 10
1860 !
1870 Clear_disp
1880 Disp_mess("Setting CF to 900 MHz. Band Pass")
1890 !
1900 ! Set Center Frequency to 900 MHz
1910 OUTPUT @Hp87xx;"SENS1:FREQ:CENT 900000000 HZ;*WAI"
1920 WAIT 10
1930 !
1940 Clear_disp
1950 Disp_mess("Return to Low Pass Mode")
1960 !
1970 ! Return to Low Pass Mode
1980 OUTPUT @Hp87xx;"SENS:FREQ:MODE LOWP; *WAI"
1990 WAIT 10
2000 !
2010 Clear_disp
2020 Disp_mess("Set Cable Loss to 10dB/100 ft")
2030 OUTPUT @Hp87xx;"SENS1:CORR:LOSS:COAX 10.0"
2040 WAIT 10
2050 !
2060 Clear_disp
2070 Disp_mess("Set Velocity factor to .8")
2080 OUTPUT @Hp87xx;"SENS1:CORR:RVEL:COAX .8"
2090 WAIT 10
2100 !
2110 Clear_disp
2120 Disp_mess("Set   Cable Loss=0., VF=1.0")
2130 OUTPUT @Hp87xx;"SENS1:CORR:LOSS:COAX .0"
2140 OUTPUT @Hp87xx;"SENS1:CORR:RVEL:COAX 1."
2150 WAIT 10
2160 !
```

```
2170 DISP "Done"
2180 BEEP
2190 END
2200 !
2210 SUB Disp_mess(Message$)
2220     COM /Sys_state/ @Hp87xx,Scode
2230     OUTPUT @Hp87xx;"DISP:ANN:MESS:DATA '"&Message$&"'"
2240 SUBEND
2250 !
2260 SUB Clear_disp
2270     COM /Sys_state/ @Hp87xx,Scode
2280     DIM Command$[40]
2290     OUTPUT @Hp87xx;"DISP:ANN:MESS:CLE"
2300 SUBEND
2310 !
2320 !**************************************************************
2330 ! Iden_port:   Identify io port to use.
2340 ! Description: This routines sets up the I/O port address for
2350 !              the SCPI interface.  For "HP 87xx" instruments,
2360 !              the address assigned to @Hp87xx = 800 otherwise,
2370 !              716.
2380 !**************************************************************
2390 SUB Iden_port
2400     COM /Sys_state/ @Hp87xx,Scode
2410 !
2420     IF POS(SYSTEM$("SYSTEM ID"),"HP 87")<>0 THEN
2430         ASSIGN @Hp87xx TO 800
2440         Scode=8
2450     ELSE
2460         ASSIGN @Hp87xx TO 716
2470         Scode=7
2480     END IF
2490 !
2500 SUBEND !Iden_port
2510 !
```

USR_FLOC Example Program

```
10   ! -
20   !
30   ! BASIC program: USR_FLOC
40   !
50   ! Fault Location measurements require option 100.
60   ! User BEGIN requires option 1C2, IBASIC.
70   !
80   !
90   ! This is an example user BEGIN program for fault location.
100  !
110  ! Load this program into the analyzer.  Then press [BEGIN]
120  ! [User BEGIN ON].
130  !
140  ! The following line is required. DO NOT REMOVE!
150 User_begin:ASSIGN @Rfna TO 800       ![User Begin] Program
160   ASSIGN @Hp8712 TO 800
170  !
180  ! To Modify:
190  ! Use [IBASIC][EDIT] or [IBASIC][Key Record]
200  !
210  !
220  ! Declare storage for variables.
230   DIM Name$[60],Str1$[60],Str2$[60],Str3$[60]
240  !
250  ! Clear the softkey labels
260   OUTPUT @Rfna;"DISP:MENU2:KEY8 '';*WAI"
270  !
280  ! Re-define softkey labels here.
290   OUTPUT @Rfna;"DISP:MENU2:KEY1 'Test End  of Cable';*WAI"
300   OUTPUT @Rfna;"DISP:MENU2:KEY2 '*';*WAI"
310   OUTPUT @Rfna;"DISP:MENU2:KEY3 'Mkr -> Max';*WAI"
320   OUTPUT @Rfna;"DISP:MENU2:KEY4 'Next Peak Left';*WAI"
330   OUTPUT @Rfna;"DISP:MENU2:KEY5 'Next Peak Right';*WAI"
340   OUTPUT @Rfna;"DISP:MENU2:KEY6 'Zoom on   Marker';*WAI"
350   OUTPUT @Rfna;"DISP:MENU2:KEY7 '*';*WAI"
360  !
370  !The following 2 lines are required. DO NOT REMOVE!
380 User_pause:PAUSE
390   GOTO User_pause
400  !
410 User_key1:        ! Example Set Stop Distance to 1100ft
420   OUTPUT @Hp8712;"SENS1:STAT ON; *WAI"
430   OUTPUT @Hp8712;"SENS1:FUNC 'FLOC 1,0';DET NBAN; *WAI"
440   OUTPUT @Hp8712;"SENS1:DIST:STOP 1100; *opc?"
450   ENTER @Hp8712;Opc
460   OUTPUT @Hp8712;"SENS1:CORR:RVEL:COAX  0.89"
470   OUTPUT @Hp8712;"DISP:WIND1:TRAC:Y:AUTO ONCE"
480   GOTO User_pause
490  !
500 User_key2:        ! Define softkey 2 here.
510   GOSUB Message ! Remove this line
520   GOTO User_pause
530  !
540 User_key3:        ! Example Marker Function
550   OUTPUT @Rfna;"CALC1:MARK1 ON"
560   OUTPUT @Rfna;"CALC1:MARK:FUNC MAX"
570   GOTO User_pause
```

```
580  !
590 User_key4:        ! Define softkey 6 here.
600   OUTPUT @Rfna;"CALC1:MARK1 ON"
610   OUTPUT @Hp8712;"CALC1:MARK:MAX:LEFT"
620   GOTO User_pause
630  !
640 User_key5:        ! Define softkey 5 here.
650   OUTPUT @Hp8712;"CALC1:MARK1 ON"
660   OUTPUT @Hp8712;"CALC1:MARK:MAX:RIGHT"
670   GOTO User_pause
680  !
690 User_key6:        ! Zoom on Cable
700   OUTPUT @Hp8712;"SENS1:STAT ON; *WAI"
710   OUTPUT @Hp8712;"SENS1:FUNC 'FLOC 1,0';DET NBAN; *WAI"
720   OUTPUT @Hp8712;"calc1:mark1:x?"
730   ENTER @Hp8712;Distance
740   New_start=Distance-20
750   IF (New_start<0) THEN New_start=0
760   OUTPUT @Hp8712;"sens1:dist:start "&VAL$(New_start)
770   OUTPUT @Hp8712;"sens1:dist:stop "&VAL$(Distance+20)
780   OUTPUT @Hp8712;"*opc?"
790   ENTER @Hp8712;Opc
800   GOTO User_pause
810  !
820 User_key7:        ! Define softkey 7 here.
830   GOSUB Message ! Remove this line.
840   GOTO User_pause
850  !
860 Message:     !
870   Str1$="This key is programmable."
880   Str2$="To modify, select"
890   Str3$="[System Options], [IBASIC], [Edit]."
900   OUTPUT @Rfna;"DISP:ANN:MESS
      '"&Str1$&CHR$(10)&Str2$&CHR$(10)&Str3$&"', MEDIUM"
910   RETURN
920  !
930   END
```

Controlling Hardcopy

FAST_PRT	This program provides fast graph dumps to PCL5 printers.
PASSCTRL	This program uses pass control and the GPIB for hardcopy output. The example uses a GPIB printer.
PRINTPLT	This program uses the serial and parallel ports for hardcopy output. The example also demonstrates plotting test results to an HP-GL file.
REPORT	This program uses the analyzer to generate a report, and sends the result to a printer connected to the parallel port. It uses a subprogram to send the output to the parallel port one line at a time. Before using this program, be sure that your printer is configured to ignore the Printer_select Centronics signal, since the `WRITEIO` command does not assert this signal.

FAST_PRT Example Program

This program configures a PCL5 printer to accept HP-GL graphics commands from the analyzer. The program executes a hardcopy which causes the analyzer to send HP-GL commands to the parallel port PCL5 printer. Provides up to 10× speed improvement of some hardcopies.

```
1000 ! FAST_PRT
1010 !
1020 ! This program is designed to set up a PCL5 printer
1030 ! connected to the parallel port of the analyzer to
1040 ! accept HP-GL syntax.  HP-GL gives fast graph dumps.
1050 !
1060 ! Connect your PCL5 printer to the parallel printer of the
1070 ! analyzer, then run the program.
1075 !
1076 ! Note: Firmware hardcopy support for PCL5 for 871xCs can
1077 ! can be enabled by selecting a PCL5 harcopy device.
1078 ! This program may still be needed for the 871xBs.
1080 !
1090 ! Once the parallel printer has been configured to accept
1100 ! HPGL commands, a hardcopy is done, the printer is
1110 ! reset to normal mode, and the page is ejected.
1120 !
1130 DIM A$[50]
1140 !
1150 !
1160 COM /Sys_state/ @Hp87xx,Scode
1170 ! Identify I/O Port
1180 CALL Iden_port
1190 !
1200 ! Define the hardcopy device
1210 OUTPUT @Hp87xx;"HCOP:DEV:LANG HPGL;PORT CENT"
1220 !
1230 ! Define PCL5 escape codes needed to set up HPGL commands:
1240 DATA @E                  ! Reset, Eject page
1250 DATA &@12A              ! Page size 8.5 x 11
1260 DATA &@a0L&@a4000M&@l0E ! No margins
1270 DATA @*c7400x5650y      ! 10.28 x 7.85 size 720/in
1280 !DATA @*c5500x5650y      ! if Marker table included
1290 !DATA @*c4255x3283y      ! portrait,remove Landscape Mode
1300 DATA &@l1O              ! Landscape Mode
1310 DATA @*p50x50y          ! Cursor to anchor point
1320 DATA @*c0T              ! Set picture anchor point
1330 DATA @*r-3U             ! CMY Palette
1340 !DATA @*r1U              ! Monochrome optional
1350 DATA @%1B               ! HPGL Mode
1360 DATA $                  ! dump plot
1370 DATA @%0A               ! Exit HPGL Mode
1380 DATA @E                 ! Eject page
1390 DATA DONE               ! End of defined escape codes
1400 !
1410 ! Send the defined escape codes to the printer
1420 LOOP
1430     READ A$
1440 EXIT IF A$="DONE"
1450     FOR I=1 TO LEN(A$)
1460         SELECT A$[I;1]
```

```
1470         CASE "@"! Escape Character
1480             OUTPUT @Hp87xx;"DIAG:PORT:WRITE 15,0,27"
1490         CASE "$"! Dump the plot
1500             OUTPUT @Hp87xx;"HCOP;*WAI"
1510         CASE ELSE! Send Character
1520             OUTPUT @Hp87xx;"DIAG:PORT:WRITE 15,0,";NUM(A$[I;1])
1530         END SELECT
1540     NEXT I
1550 END LOOP
1560 !
1570 END
1580 !
1590 !**************************************************************
1600 ! Iden_port:   Identify io port to use.
1610 ! Description: This routines sets up the I/O port address for
1620 !              the SCPI interface.  For "HP 87xx" instruments,
1630 !              the address assigned to @Hp87xx = 800 otherwise,
1640 !              716.
1650 !**************************************************************

1660 SUB Iden_port
1670     COM /Sys_state/ @Hp87xx,Scode
1680 !
1690     IF POS(SYSTEM$("SYSTEM ID"),"HP 87")<>0 THEN
1700         ASSIGN @Hp87xx TO 800
1710         Scode=8
1720     ELSE
1730         ASSIGN @Hp87xx TO 716
1740         Scode=7
1750     END IF
1760 !
1770 SUBEND !Iden_port
1780 !
```

PASSCTRL Example Program

This program demonstrates how to send a hardcopy to a GPIB printer. This is done by passing active control of the bus to the analyzer so it can control the printer. More information about passing control to the analyzer is available in the *Programmer's Guide*.

```
1000 !Filename:  PASSCTRL
1010 !
1020 ! Description:
1030 !   External controller runs this program, which
1040 !   instructs the analyzer to perform a hardcopy
1050 !   and then passes control to the analyzer.
1060 !   Analyzer performs hardcopy over HP-IB
1070 !   to printer at 701, then passes control back.
1080 !
1090 !   This program only works on controllers which
1100 !   implement pass control properly.  HP s700
1110 !   computers running BASIC-UX 7.0x will need
1120 !   to upgrade to a newer BASIC-UX version.
1130 !
1140 !
1150 COM /Sys_state/ @Hp87xx,Scode,Internal
1160 ! Identify I/O Port
1170 CALL Iden_port
1180 !
1190 !
1200 ! Select the language to PCL (Printer
1210 ! Control Language) and the output port
1220 ! to HP-IB.
1230 OUTPUT @Hp87xx;"HCOP:DEV:LANG PCL;PORT GPIB"
1240 !
1250 ! Select the HP-IB address for the hardcopy
1260 ! device on the HP-IB.
1270 OUTPUT @Hp87xx;"SYST:COMM:GPIB:HCOP:ADDR 1"
1280 !
1290 ! Set the output to graph only.
1300 OUTPUT @Hp87xx;"HCOP:DEV:MODE GRAP"
1310 !
1320 ! If the internal controller is being used...
1330 IF Internal=1 THEN
1340 !
1350 ! then make it System Controller of HP-IB
1360     OUTPUT @Hp87xx;"SYST:COMM:GPIB:CONT ON"
1370 END IF
1380 !
1390 ! Clear Status Registers
1400 OUTPUT @Hp87xx;"*CLS"
1410 !
1420 ! Enable the Request Control bit in the Event
1430 ! Status Register.
1440 OUTPUT @Hp87xx;"*ESE 2"
1450 !
1460 ! Clear the Service Request enable register;
1470 ! SRQ is not being used.
1480 OUTPUT @Hp87xx;"*SRE 0"
1490 !
1500 ! Send the hardcopy command to start the
```

```
1510 ! print.
1520 OUTPUT @Hp87xx;"HCOP"
1530 LOOP
1540 !
1550 ! Read the status byte using Serial Poll.
1560     Stat=SPOLL(@Hp87xx)
1570 !
1580 ! Exit when the analyzer requests active control
1590 ! of HP-IB from the system controller.
1600 EXIT IF BIT(Stat,5)=1
1610 END LOOP
1620 !
1630 ! Now system controller passes control to
1640 ! the analyzer.
1650 PASS CONTROL @Hp87xx
1660 DISP "Hardcopy in Progress...";
1670 IF Internal=1 THEN
1680 ! If using the internal IBASIC controller,
1690 ! then use the *OPC query method to wait
1700 ! for hardcopy completion.
1710     OUTPUT @Hp87xx;"*OPC?"
1720     ENTER @Hp87xx;Opc
1730 ELSE
1740 ! If external computer control, then...
1750     LOOP
1760 !
1770 ! Monitor the HP-IB status in the
1780 ! external computer's HP-IB status
1790 ! register.  Here, the HP-IB interface
1800 ! code 7 register 6 status is requested
1810 ! and put into "Hpib".
1820         DISP ".";
1830         WAIT 1! No need to poll rapidly
1840         STATUS 7,6;Hpib
1850 !
1860 ! When active control is returned to the
1870 ! system controller (bit 6 set), then exit.
1880 ! (This fails on s700s running BASIC 7.0x)
1890     EXIT IF BIT(Hpib,6)=1
1900     END LOOP
1910 END IF
1920 DISP "HARDCOPY COMPLETE!"
1930 END
1940 !
1950 !***************************************************************
1960 ! Iden_port:   Identify io port to use.
1970 ! Description: This routines sets up the I/O port address for
1980 !              the SCPI interface.  For "HP 87xx" instruments,
1990 !              the address assigned to @Hp87xx = 800 otherwise,
2000 !              716.
2010 !***************************************************************

2020 SUB Iden_port
2030     COM /Sys_state/ @Hp87xx,Scode,Internal
2040 !
2050     IF POS(SYSTEM$("SYSTEM ID"),"HP 87")<>0 THEN
2060         ASSIGN @Hp87xx TO 800
2070         Scode=8
2080         Internal=1
2090     ELSE
```

```
2100          ASSIGN @Hp87xx TO 716
2110          Scode=7
2120          Internal=0
2130      END IF
2140 !
2150 SUBEND !Iden_port
2160 !
```

PRINTPLT Example Program

This program demonstrates how to send a hardcopy to a printer on the serial interface. This is done by selecting the appropriate device, setting up the baud rate and hardware handshaking, and sending the command to print or plot. The `*OPC?` query is used in this example to indicate when the printout is complete. Another method of obtaining the same results is to monitor the Hardcopy in Progress bit (bit 9 in the Operational Status Register). More information on printing or plotting is available in the *User's Guide*.

Lines `1170-1400` demonstrate sending a hardcopy output to a printer connected to the serial port. The same program could be used to send hardcopy output to a device on the parallel port. The only changes would be deleting lines `1230-1280` and changing line `1200` to read `HCOP:DEV:PORT PAR`.

Lines `1430-1680` demonstrate how to create an HP-GL file (plotter language) and send it to the disk in the internal 3.5” disk drive.

```
1000 !Filename:  PRINTPLT
1010 !
1020 ! Description:
1030 !   1. Select serial port.  Configure it.
1040 !   2. Dump table of trace values.
1050 !   3. Re-configure hardcopy items to dump.
1060 !   4. Dump HP-GL file to internal floppy.
1070 !
1080 !
1090 COM /Sys_state/ @Hp87xx,Scode
1100 ! Identify I/O Port
1110 CALL Iden_port
1120 !
1130 !
1140 ! Select the output language (PCL-Printer
1150 ! Control Language) and the hardcopy port
1160 ! to serial.
1170 OUTPUT @Hp87xx;"HCOP:DEV:LANG PCL;PORT SER"
1180 !
1190 ! Select baud rate to 19200.
1200 OUTPUT @Hp87xx;"SYST:COMM:SER:TRAN:BAUD 19200"
1210 !
1220 ! Select the handshaking protocol to Xon/Xoff.
1230 OUTPUT @Hp87xx;"SYST:COMM:SER:TRAN:HAND XON"
1240 !
1250 ! Select the type of output to table, which
1260 ! is the same as the softkey List Trace
1270 ! Values under the Define Hardcopy menu.
1280 OUTPUT @Hp87xx;"HCOP:DEV:MODE TABL"
1290 !
1300 ! Send the command to start a hardcopy, and
1310 ! use *OPC query to make sure the hardcopy is
1320 ! complete before continuing.
1330 OUTPUT @Hp87xx;"HCOP;*OPC?"
```

```
1340 ENTER @Hp87xx;Opc
1350 DISP "Hardcopy to serial printer - COMPLETE!"
1360 !
1370 ! Select the HPGL language and the hardcopy
1380 ! port to be the currently selected mass memory
1390 ! device.
1400 OUTPUT @Hp87xx;"HCOP:DEV:LANG HPGL;PORT MMEM"
1410 !
1420 ! Include trace data in the plot.
1430 OUTPUT @Hp87xx;"HCOP:ITEM:TRAC:STAT ON"
1440 !
1450 ! Turn graticule off in the hardcopy dump.
1460 OUTPUT @Hp87xx;"HCOP:ITEM:GRAT:STAT OFF"
1470 !
1480 ! Include frequency and measurement
1490 ! annotation.
1500 OUTPUT @Hp87xx;"HCOP:ITEM:ANN:STAT ON"
1510 !
1520 ! Include marker symbols.
1530 OUTPUT @Hp87xx;"HCOP:ITEM:MARK:STAT ON"
1540 !
1550 ! Include title (and/or time/date if
1560 ! already selected).
1570 OUTPUT @Hp87xx;"HCOP:ITEM:TITL:STAT ON"
1580 !
1590 ! Define the hardcopy to be both the graph
1600 ! and a marker table.
1610 OUTPUT @Hp87xx;"HCOP:DEV:MODE GMAR"
1620 !
1630 ! Send the command to plot and use *OPC
1640 ! query to wait for finish.
1650 OUTPUT @Hp87xx;"HCOP;*OPC?"
1660 ENTER @Hp87xx;Opc
1670 DISP "Plot to floppy disk - COMPLETE!"
1680 END
1690 !
1700 !**************************************************************
1710 ! Iden_port:   Identify io port to use.
1720 ! Description: This routines sets up the I/O port address for
1730 !              the SCPI interface.  For "HP 87xx" instruments,
1740 !              the address assigned to @Hp87xx = 800 otherwise,
1750 !              716.
1760 !**************************************************************
1770 SUB Iden_port
1780     COM /Sys_state/ @Hp87xx,Scode
1790 !
1800     IF POS(SYSTEM$("SYSTEM ID"),"HP 87")<>0 THEN
1810         ASSIGN @Hp87xx TO 800
1820         Scode=8
1830     ELSE
1840         ASSIGN @Hp87xx TO 716
1850         Scode=7
1860     END IF
1870 !
1880 SUBEND !Iden_port
1890 !
```

REPORT Example Program

```
10     !-------------------------------------------------------
20     !
30     ! IBASIC program:  REPORT - Using the parallel port
40     !
50     ! This program uses the 871X to generate a report,
60     ! making a hardcopy on a printer connected to the
70     ! parallel port.  It uses a subprogram to send the
80     ! output to the parallel port one line at a time.
90     !
100    ! This example uses five different font types that
110    ! may or may not be supported for your printer.
120    ! These character fonts are available for HP LaserJet
130    ! printers.  Refer to your printer manual to modify
140    ! the example fonts for your printer.
150    !
160    !-------------------------------------------------------
170    !
180    ! Assign an I/O path name for the internal bus and
190    ! declare and initialize variables.
200    !
210    COM /Cset/ Block$[50],Title$[50],Slant$[50],Banner$[50],Medium$[50]
220    ASSIGN @Rfna TO 800
230    Esc$=CHR$(27)
240    !
250    ! Preset the analyzer, put it in Trigger HOLD mode,
260    ! allocate the full IBASIC display and clear the
270    ! screen.
280    !
290    OUTPUT @Rfna;”SYST:PRES;*WAI”
300    OUTPUT @Rfna;”ABOR;:INIT:CONT OFF;*WAI”
310    OUTPUT @Rfna;”DISP:PROG FULL”
320    CLEAR SCREEN
330    !
340    ! Define the escape sequence for each font that is
350    ! used.  Refer to your printer manual.
360    !
370    Block$=Esc$&”&l0O”&Esc$&”(8U”&Esc$&”(s1p10h12v0s0b0T”
380    Title$=Esc$&”&l0O”&Esc$&”(8U”&Esc$&”(s1p8h12v0s0b0T”
390    Slant$=Esc$&”&l0O”&Esc$&”(7J”&Esc$&”(s0p6h14v1s0b0T”
400    Banner$=Esc$&”&l0O”&Esc$&”(7J”&Esc$&”(s0p4h24v0s0b0T”
410    Medium$=Esc$&”&l0O”&Esc$&”(7J”&Esc$&”(s0p8h14v0s0b0T”
420    !
430    ! Select the font to use writing the company name
440    ! and address, send the company name and address.
450    !
460    CALL Send_line(Title$,1)
470    CALL Send_line(“COMPANY NAME”,1)
480    CALL Send_line(“CITY, STATE, COUNTRY”,1)
490    CALL Send_line(“ “,1)
500    !
510    ! Select the font to use writing the device name,
520    ! send the device name.
530    !
540    CALL Send_line(Banner$,1)
550    CALL Send_line(“_____________________________”,0)
560    CALL Send_line(“_____________________________”,1)
570    CALL Send_line(“ “,1)
```

```
580   CALL Send_line(“ “,1)
590   CALL Send_line(“  BPF-175 Bandpass Filter”,1)
600   CALL Send_line(“ “,1)
610   CALL Send_line(“ “,1)
620   CALL Send_line(“_____________________________”,0)
630   CALL Send_line(“_____________________________”,1)
640   CALL Send_line(“  “,1)
650   !
660   ! Select the font to use writing the device
670   ! specifications, send the information.
680   !
690   CALL Send_line(Slant$,1)
700   CALL Send_line(“  “,1)
710   CALL Send_line(“PASS BAND (MHZ)        3 dB                  60 +/- 5”,1)
720   CALL Send_line(“ “,1)
730   CALL Send_line(“                      20 dB                  90 +/- 5”,1)
740   CALL Send_line(“ “,1)
750   CALL Send_line(“                      40 dB                 120 +/- 5”,1)
760   CALL Send_line(“ “,1)
770   CALL Send_line(“SWR PASSBAND (typical)     1.8:1”,1)
780   CALL Send_line(“ “,1)
790   CALL Send_line(“SWR STOPBAND (typical)     1.8:1”,1)
800   CALL Send_line(“ “,1)
810   CALL Send_line(“Cost per unit:            36.95”,1)
820   !
830   ! Select the font to use for the performance data
840   ! title, send the title.
850   !
860   CALL Send_line(Block$,1)
870   CALL Send_line(“                               “,0)
880   CALL Send_line(“                   Transmission Characteristics”,1)
890   !
900   ! Return the display to the analyzer.
910   !
920   OUTPUT @Rfna;”DISP:PROG OFF”
930   !
940   ! Setup the device measurement.  This example
950   ! measures the transmission response of a
960   ! bandpass filter at 175 MHz.
970   !
980   OUTPUT @Rfna;”DISP:ANN:FREQ1:MODE SSTOP”
990   OUTPUT @Rfna;”SENS1:FREQ:STAR 10 MHz;STOP 400 MHz;*WAI”
1000  OUTPUT @Rfna;”DISP:WIND1:TRAC:Y:PDIV 20 dB;RLEV -50 dB;RPOS 5”
1010  OUTPUT @Rfna;”DISP:ANN:TITL ON;TITL1:DATA ‘HP 8711 RF NETWORK ANALYZER’”
1020  !
1030  ! Take a measurement sweep and wait for it to
1040  ! complete.  Perform a -3 dB bandwidth search.
1050  !
1060  OUTPUT @Rfna;”INIT1;*OPC?”
1070  ENTER @Rfna;Opc
1080  OUTPUT @Rfna;”CALC1:MARK1 ON;MARK:BWID -3”
1090  !
1100  ! Select the parallel port and the printer’s
1110  ! control language as the hardcopy device.
1120  ! Set the printer resolution and margins -
1130  ! turn off automatic form feed.
1140  !
1150  OUTPUT @Rfna;”HCOP:DEV:LANG PCL;PORT CENT”
1160  OUTPUT @Rfna;”HCOP:DEV:RES 300”
1170  OUTPUT @Rfna;”HCOP:PAGE:MARG:LEFT  40”
```

```
1180  OUTPUT @Rfna;"HCOP:PAGE:WIDT  110"
1190  OUTPUT @Rfna;"HCOP:ITEM1:FFE:STAT OFF"
1200  !
1210  ! Send the measurement data (graph and marker
1220  ! values) to the printer.
1230  !
1240  OUTPUT @Rfna;"HCOP"
1250  !
1260  ! Select the fonts and send the "footer"
1270  ! information for the report.
1280  !
1290  CALL Send_line(Banner$,1)
1300  CALL Send_line(" ",1)
1310  CALL Send_line(" ",1)
1320  CALL Send_line("IN STOCK ____ IMMEDIATE DELIVERY!",1)
1330  CALL Send_line(Medium$,1)
1340  CALL Send_line("         ",0)
1350  CALL Send_line("For more information: Call 1-800-Filter",1)
1360  !
1370  ! Send a form feed to the printer.
1380  !
1390  WRITEIO 15,0;12
1400  END
1410  !-----------------------------------------------------
1420  SUB Send_line(String$,INTEGER Crlf)
1430    !---------------------------------------------------
1440    !
1450    ! The subprogram sends a string to the parallel port
1460    ! (I/O port 15).  The Crlf flag determines whether
1470    ! a carriage return (ASCII 13) and line feed (ASCII
1480    ! 10) are needed at the end of the string.
1490    !
1500    !---------------------------------------------------
1510    INTEGER Length
1520    Length=LEN(String$)
1530    FOR I=1 TO Length
1540      WRITEIO 15,0;NUM(String$[I;1])
1550    NEXT I
1560    IF Crlf=1 THEN
1570      WRITEIO 15,0;10
1580      WRITEIO 15,0;13
1590    END IF
1600  SUBEND
```

Saving and Recalling Instrument States

LEARNSTR	This program uses the learn string to upload and download instrument states.
SAVERCL	This program saves and recalls instrument states, calibrations and data. The example also demonstrates saving data in an ASCII file that includes both magnitude and frequency information.

LEARNSTR Example Program

This program demonstrates how to upload and download instrument states using the learn string. The learn string is a fast and easy way to read an instrument state. It is read out using the `*LRN?` query (an IEEE 488.2 common commands). To restore the learn string, simply output the string to the analyzer.

The learn string contains a mnemonic at the beginning that tells the analyzer to restore the instrument state.

The learn string is transferred as a block. The header is ASCII formatted and the data is in the instrument's internal binary format. The number of bytes in the block of data is determined by the instrument state (no more than 20000 bytes).

```
"SYST:SET #<digits><bytes><learn string data>"
```

The "long" learnstring will include the instrument state like the normal learnstring, and will also include data and calibration arrays if they are selected using the Define Save function under SAVE RECALL. The SCPI equivalent command for saving the calibration arrays is added before the "long" learnstring query.

```
1000 !Filename:  LEARNSTR
1010 !
1020 ! Description:
1030 !   1. Query the learn string.
1040 !   2. Preset the analyzer.
1050 !   3. Send the learn string,
1060 !      restoring the previous state.
1070 !
1080 DIM Learnstr$[20000]
1090 !
1100 COM /Sys_state/ @Hp87xx,Scode
1110 ! Identify I/O Port
1120 CALL Iden_port
1130 !
1140 !
1150 ! Request the learnstring.  If the "long"
1160 ! learnstring is desired, comment the line
1170 ! below, and uncomment the line after it.
1180 ! The "long" learnstring, in addition to
1190 ! the instrument state like the normal
1200 ! learnstring, will include data and
1210 ! calibration arrays IF they are selected
1220 ! using the Define Save function under
1230 ! SAVE RECALL.  The SCPI equivalent command
1240 ! for saving the calibration arrays is
1250 ! added before the "long" learnstring query.
1260 OUTPUT @Hp87xx;"*LRN?"
1270 ! OUTPUT @Hp87xx;"MMEM:STOR:STAT:CORR ON;:SYST:SET:LRNL?"
1280 !
1290 ! Read the learnstring from the analyzer.
```

```
1300 ! The USING "-K" format allows the data
1310 ! being transmitted to include characters
1320 ! (such as the line feed character) that
1330 ! would otherwise terminate the learnstring
1340 ! request prematurely.
1350 ENTER @Hp87xx USING "-K";Learnstr$
1360 DISP "Learn string has been read"
1370 WAIT 5
1380 !
1390 ! Preset the analyzer.
1400 OUTPUT @Hp87xx;"SYST:PRES;*OPC?"
1410 !
1420 ! Wait for the preset operation to complete.
1430 ENTER @Hp87xx;Opc
1440 DISP "Instrument has been PRESET"
1450 WAIT 5
1460 !
1470 ! Output the learnstring to the analyzer.
1480 ! The mnemonic is included in the string,
1490 ! so no command preceding "Learnstr$" is
1500 ! necessary.
1510 OUTPUT @Hp87xx;Learnstr$
1520 DISP "Instrument state has been restored"
1530 END
1540 !
1550 !**************************************************************
1560 ! Iden_port:   Identify io port to use.
1570 ! Description: This routines sets up the I/O port address for
1580 !              the SCPI interface.  For "HP 87xx" instruments,
1590 !              the address assigned to @Hp87xx = 800 otherwise,
1600 !              716.
1610 !**************************************************************
1620 SUB Iden_port
1630     COM /Sys_state/ @Hp87xx,Scode
1640 !
1650     IF POS(SYSTEM$("SYSTEM ID"),"HP 87")<>0 THEN
1660         ASSIGN @Hp87xx TO 800
1670         Scode=8
1680     ELSE
1690         ASSIGN @Hp87xx TO 716
1700         Scode=7
1710     END IF
1720 !
1730 SUBEND !Iden_port
1740 !
```

SAVERCL Example Program

This program demonstrates how to save instrument states, calibrations and data to a mass storage device. The device used in this example is the analyzer's internal 3.5” disk drive. To use this program with the internal non-volatile memory, change the mass storage unit specifier.

The three choices are the internal 3.5” disk drive (`INT:`), the internal non-volatile memory, (`MEM:`), and the internal volatile memory, (`RAM:`).

Lines `1110-1320` are an example of saving an instrument state and calibration on the internal floppy disk drive.

Lines `1460-1470` are an example of recalling that instrument state and calibration.

Lines `1510-1560` are an example of saving a data trace (magnitude and frequency values) to an ASCII formatted file on the internal 3.5” disk drive. This file cannot be recalled into the instrument. It can, however, be imported directly into spreadsheets and word processors.

```
1000 !Filename:  SAVERCL
1010 !
1020 !
1030 COM /Sys_state/ @Hp87xx,Scode
1040 ! Identify I/O Port
1050 CALL Iden_port
1060 !
1070 !
1080 ! Select the internal floppy disk drive
1090 ! as the mass storage device.
1100 OUTPUT @Hp87xx;"MMEM:MSIS 'INT:'"
1110 !
1120 ! Turn on the saving of the instrument state
1130 ! as part of the "Define Save" function under
1140 ! SAVE RECALL.
1150 OUTPUT @Hp87xx;"MMEM:STOR:STAT:IST ON"
1160 !
1170 ! Turn on the saving of the calibration
1180 ! as part of the "Define Save" function under
1190 ! SAVE RECALL.
1200 OUTPUT @Hp87xx;"MMEM:STOR:STAT:CORR ON"
1210 !
1220 ! Turn off the saving of the data
1230 ! as part of the "Define Save" function under
1240 ! SAVE RECALL.
1250 OUTPUT @Hp87xx;"MMEM:STOR:STAT:TRAC OFF"
1260 !
1270 ! Save the current defined state (STAT 1) into
1280 ! a file named "FILTER".  Use *OPC? to make
1290 ! sure the operation is completed before any
1300 ! other operation begins.
1310 OUTPUT @Hp87xx;"MMEM:STOR:STAT 1,'FILTER';*OPC?"
1320 ENTER @Hp87xx;Opc
1330 DISP "Instrument state and calibration have been saved"
```

```
1340 !
1350 ! Preset the instrument so that the change in state
1360 ! is easy to see when it is recalled.
1370 OUTPUT @Hp87xx;"SYST:PRES;*OPC?"
1380 ENTER @Hp87xx;Opc
1390 DISP "Instrument has been PRESET"
1400 WAIT 5
1410 !
1420 ! Recall the file "FILTER" from the internal
1430 ! floppy disk drive.  This becomes the new instrument
1440 ! state.  Use of the *OPC query allows hold off of
1450 ! further commands until the analyzer is reconfigured.
1460 OUTPUT @Hp87xx;"MMEM:LOAD:STAT 1,'INT:FILTER';*OPC?"
1470 ENTER @Hp87xx;Opc
1480 !
1490 ! Take a single sweep to ensure that valid measurement
1500 ! data is acquired.
1510 OUTPUT @Hp87xx;"ABOR;:INIT:CONT OFF;:INIT;*WAI"
1520 DISP "Instrument state and calibration have been recalled"
1530 !
1540 ! Save that measurement data into an ASCII file.
1550 ! called "DATA0001" on the internal floppy disk drive.
1560 OUTPUT @Hp87xx;"MMEM:STOR:TRAC CH1FDATA,'INT:DATA0001'"
1570 DISP "Data has been saved (ASCII format)"
1580 END
1590 !
1600 !***************************************************************
1610 ! Iden_port:   Identify io port to use.
1620 ! Description: This routines sets up the I/O port address for
1630 !              the SCPI interface.  For "HP 87xx" instruments,
1640 !              the address assigned to @Hp87xx = 800 otherwise,
1650 !              716.
1660 !***************************************************************
1670 SUB Iden_port
1680     COM /Sys_state/ @Hp87xx,Scode
1690 !
1700     IF POS(SYSTEM$("SYSTEM ID"),"HP 87")<>0 THEN
1710         ASSIGN @Hp87xx TO 800
1720         Scode=8
1730     ELSE
1740         ASSIGN @Hp87xx TO 716
1750         Scode=7
1760     END IF
1770 !
1780 SUBEND !Iden_port
1790 !
```

Using Marker Functions

MKR_MATH

Marker math functions are used to calculate different parameters on a user-defined measurement trace segment. Frequency span, mean amplitude, amplitude standard deviation, and peak-to-peak amplitude are calculated with the Statistics function. Span, gain, slope and flatness are calculated with the Flatness function. Insertion loss and peak-to-peak ripple of the passband, and maximum signal amplitude in the stopband are calculated with the RF Filter Stats function. This example program steps through the marker math functions, and then reads and reports the results.

MKR_MATH Example Program

```
1000 !Filename:  MKR_MATH
1010 !
1020 ! This example program demonstrates how to program marker math
1030 ! functions.  Marker Statistics, Marker Flatness, and RF Filter Stats.
1040 !
1050 ! Connect the demo filter between the RF out and RF in of the analyzer.
1060 !
1070 ! The program will step through various marker math measurements, then
1080 ! read and report the results.
1090 !
1100 !
1110 COM /Sys_state/ @Hp87xx,Scode
1120 ! Identify I/O Port
1130 CALL Iden_port
1140 !
1150 !
1160 ! Perform a system preset;
1170 OUTPUT @Hp87xx;"SYST:PRES;*WAI"
1180 !
1190 ! Set up the source frequencies for the measurement.
1200 OUTPUT @Hp87xx;"SENS1:FREQ:STAR 10 MHZ;STOP 400 MHZ;*WAI"
1210 !
1220 ! Set up the receiver for the measurement parameters
1230 ! (Transmission in this case).
1240 OUTPUT @Hp87xx;"SENS1:FUNC 'XFR:POW:RAT 2,0';DET NBAN;*WAI"
1250 !
1260 ! Configure the display so measurement
1270 ! results are easy to see.
1280 OUTPUT @Hp87xx;"DISP:WIND1:TRAC:Y:PDIV 10 DB;RLEV 0 DB;RPOS 9"
1290 !
1300 ! Reduce the distractions on the display by
1310 ! getting rid of notation that will not be
1320 ! needed in this example.
1330 OUTPUT @Hp87xx;"DISP:ANN:YAX OFF"
1340 !
1350 ! Erase the graticule grid for the same reason.
1360 OUTPUT @Hp87xx;"DISP:WIND1:TRAC:GRAT:GRID OFF"
1370 !
1380 ! Set the markers for channel 1
1390 OUTPUT @Hp87xx;"CALC1:MARK1 ON"
1400 OUTPUT @Hp87xx;"CALC1:MARK1:X 152000000.000000"
1410 OUTPUT @Hp87xx;"CALC1:MARK2 ON"
1420 OUTPUT @Hp87xx;"CALC1:MARK2:X 200000000.000000"
1430 OUTPUT @Hp87xx;"CALC1:MARK3 ON"
1440 OUTPUT @Hp87xx;"CALC1:MARK3:X 279000000.000000"
1450 OUTPUT @Hp87xx;"CALC1:MARK4 ON"
1460 OUTPUT @Hp87xx;"CALC1:MARK4:X 388000000.000000"
1470 !
1480 ! Turn on marker flatness
1490 OUTPUT @Hp87xx;"CALC1:MARK:FUNC FLATNESS"
1500 DISP "Marker Flatness"
1510 !
1520 WAIT 5
1530 OUTPUT @Hp87xx;"CALC1:MARK:FUNC:RES?"
1540 ! Read the four values:  the span, gain
1550 ! the slope, and the flatness.
1560 ENTER @Hp87xx;Span,Gain,Slope,Flatness
```

```
1570 !
1580 ! Display the results.
1590 BEEP
1600 DISP "Span ";Span
1610 !
1620 WAIT 5
1630 BEEP
1640 DISP "Gain ";Gain
1650 !
1660 WAIT 5
1670 BEEP
1680 DISP "Slope ";Slope
1690 !
1700 WAIT 5
1710 BEEP
1720 DISP "Flatness ";Flatness
1730 !
1740 WAIT 5
1750 ! Turn on marker statistics
1760 OUTPUT @Hp87xx;"CALC1:MARK:FUNC STATISTICS"
1770 DISP "Marker Statistics"
1780 !
1790 WAIT 5
1800 OUTPUT @Hp87xx;"CALC1:MARK:FUNC:RES?"
1810 ! Read the four values:  the span,
1820 ! the mean, the sdev, peak to peak.
1830 ENTER @Hp87xx;Span,Mean,Sdev,Peak
1840 !
1850 ! Display the results.
1860 BEEP
1870 DISP "Span ";Span
1880 !
1890 WAIT 5
1900 BEEP
1910 DISP "Mean ";Mean
1920 !
1930 WAIT 5
1940 BEEP
1950 DISP "Sdev ";Sdev
1960 !
1970 WAIT 5
1980 BEEP
1990 DISP "Peak ";Peak
2000 !
2010 WAIT 5
2020 ! Turn on RF Filter Stats
2030 OUTPUT @Hp87xx;"CALC1:MARK:FUNC FST"
2040 DISP "RF Filter Stats"
2050 !
2060 WAIT 5
2070 OUTPUT @Hp87xx;"CALC1:MARK:FUNC:RES?"
2080 ! Read the three values:  the loss,
2090 ! the peak to peak, and the reject
2100 ENTER @Hp87xx;Loss,Peak,Reject
2110 !
2120 ! Display the results.
2130 BEEP
2140 DISP "Loss ";Loss
2150 !
2160 WAIT 5
```

```
2170 BEEP
2180 DISP "Peak ";Peak
2190 !
2200 WAIT 5
2210 BEEP
2220 DISP "Reject ";Reject
2230 !
2240 WAIT 5
2250 DISP "Done"
2260 END
2270 !
2280 !**************************************************************
2290 ! Iden_port:   Identify io port to use.
2300 ! Description: This routines sets up the I/O port address for
2310 !              the SCPI interface.  For "HP 87xx" instruments,
2320 !              the address assigned to @Hp87xx = 800 otherwise,
2330 !              716.
2340 !**************************************************************
2350 SUB Iden_port
2360     COM /Sys_state/ @Hp87xx,Scode
2370 !
2380     IF POS(SYSTEM$("SYSTEM ID"),"HP 87")<>0 THEN
2390         ASSIGN @Hp87xx TO 800
2400         Scode=8
2410     ELSE
2420         ASSIGN @Hp87xx TO 716
2430         Scode=7
2440     END IF
2450 !
2460 SUBEND !Iden_port
2470 !
```

Using Marker Limit Tests

LIM_FLAT	Limit testing can be performed on the flatness of a user-defined measurement trace segment. This example program sets various flatness limits, then queries the status to determine if the limit test passes or fails.
LIM_PEAK	Limit testing can be performed on the peak-to-peak ripple of a user-defined measurement trace segment. This example program sets various peak-to-peak limits, then queries the status to determine if the limit test passes or fails.
LIM_MEAN	Limit testing can be performed on the mean amplitude of a user-defined measurement trace segment. This example program sets various mean limits, then queries the status to determine if the limit test passes or fails.

LIM_FLAT Example Program

```
10   !Filename:  LIM_FLAT
20   !
30   ! This example program demonstrates how to test for a marker
40   ! flatness limit.
50   !
60   ! Connect the demo filter to the analyzer RF out and RF in.
70   ! The analyzer will set-up a transmission measurement.
80   !
90   ! The program will set various flatness limits, then query the
100  ! status to determine if the specification PASSES or FAILS.
110  !
120  !
130   IF POS(SYSTEM$("SYSTEM ID"),"HP 871") THEN
140     ASSIGN @Hp8711 TO 800
150   ELSE
160     ASSIGN @Hp8711 TO 716
170     ABORT 7
180     CLEAR 716
190   END IF
200  !
210  ! Perform a system preset; this clears the limit table.
220   OUTPUT @Hp8711;"SYST:PRES;*WAI"
230  !
240  ! Set up the source frequencies for the measurement.
250   OUTPUT @Hp8711;"SENS1:FREQ:STAR 10 MHZ;STOP 400 MHZ;*WAI"
260  !
270  ! Set up the receiver for the measurement parameters
280  ! (Transmission in this case).
290   OUTPUT @Hp8711;"SENS1:FUNC 'XFR:POW:RAT 2,0';DET NBAN;*WAI"
300  !
310  ! Configure the display so measurement
320  ! results are easy to see.
330   OUTPUT @Hp8711;"DISP:WIND1:TRAC:Y:PDIV 10 DB;RLEV 0 DB;RPOS 9"
340  !
350  ! Reduce the distractions on the display by
360  ! getting rid of notation that will not be
370  ! needed in this example.
380   OUTPUT @Hp8711;"DISP:ANN:YAX OFF"
390  !
400  ! Erase the graticule grid for the same reason.
410   OUTPUT @Hp8711;"DISP:WIND1:TRAC:GRAT:GRID OFF"
420  !
430  ! Set the markers for channel 1
440   OUTPUT @Hp8711;"CALC1:MARK1 ON"
450   OUTPUT @Hp8711;"CALC1:MARK1:X 152000000.000000"
460   OUTPUT @Hp8711;"CALC1:MARK2 ON"
470   OUTPUT @Hp8711;"CALC1:MARK2:X 200000000.000000"
480  !
490  ! Turn on marker flatness
500   OUTPUT @Hp8711;"CALC1:MARK:FUNC FLATNESS"
510  !
520   OUTPUT @Hp8711;"CALC1:MARK2 ON"
530   OUTPUT @Hp8711;"CALC1:LIM:DISP ON"
540   OUTPUT @Hp8711;"CALC1:LIM:MARK:FLATNESS ON"
550  !
560  ! Turn on the pass/fail testing; watch the
570  ! analyzer's display for the pass/fail indicator.
```

```
580   OUTPUT @Hp8711;"CALC1:LIM:STAT ON"
590  !
600  !  Set sweep hold mode
610   OUTPUT @Hp8711;"ABOR;:INIT1:CONT OFF;:INIT1;*WAI"
620  !
630  ! Send an operation complete query to ensure that
640  ! all overlapped commands have been executed.
650   OUTPUT @Hp8711;"*OPC?"
660  !
670  ! Wait for the reply.
680   ENTER @Hp8711;Opc
690  !
700  ! Turn on a limit to be tested
710   FOR Flatness=0. TO 3 STEP .1
720     DISP "Flatness limit test =",VAL$(Flatness)&" dB"
730     OUTPUT @Hp8711;"CALC1:LIM:MARK:FLAT:MAX "&VAL$(Flatness)
740  !
750  ! Take a controlled sweep to ensure that
760  ! there is real data present for the limit test.
770     OUTPUT @Hp8711;"INIT1;*OPC?"
780     ENTER @Hp8711;Opc
790  !
800  ! Query the limit fail condition register to see
810  ! if there is a failure.
820     OUTPUT @Hp8711;"STAT:QUES:LIM:COND?"
830  !
840  ! Read the register's contents.
850     ENTER @Hp8711;Fail_flag
860  !
870  ! Bit 0 is the test result for channel 1 while
880  ! Bit 1 is the results for channel 2 limit testing.
890  ! Bit 2 is the result for channel 1 mkr limit testing.
900  ! Bit 3 is the result for channel 2 mkr limit testing.
910     IF BIT(Fail_flag,2)=1 THEN
920  ! This limit test failed
930     ELSE
940       DISP "Flatness passed at "&VAL$(Flatness)&" dB"
950       BEEP
960       GOTO Done
970     END IF
980  !
990   NEXT Flatness
1000 Done:OUTPUT @Hp8711;"INIT:CONT ON;*WAI"
1010  END
```

LIM_PEAK Example Program

```
1000 !Filename: LIM_PEAK
1010 !
1020 ! This example program demonstrates how to test for a marker
1030 ! statistics peak to peak ripple limit.
1040 !
1050 ! Connect the demo filter to the analyzer RF out and RF in.
1060 ! The analyzer will set-up a transmission measurement.
1070 !
1080 ! The program will set various statistics peak to peak limits, then
1090 ! query the status to determine if the specification PASSES or FAILS.
1100 !
1110 !
1120 !
1130 COM /Sys_state/ @Hp87xx,Scode
1140 ! Identify I/O Port
1150 CALL Iden_port
1160 !
1170 !
1180 ! Perform a system preset; this clears the limit table.
1190 OUTPUT @Hp87xx;"SYST:PRES;*WAI"
1200 !
1210 ! Set up the source frequencies for the measurement.
1220 OUTPUT @Hp87xx;"SENS1:FREQ:STAR 10 MHZ;STOP 400 MHZ;*WAI"
1230 !
1240 ! Set up the receiver for the measurement parameters
1250 ! (Transmission in this case).
1260 OUTPUT @Hp87xx;"SENS1:FUNC 'XFR:POW:RAT 2,0';DET NBAN;*WAI"
1270 !
1280 ! Configure the display so measurement
1290 ! results are easy to see.
1300 OUTPUT @Hp87xx;"DISP:WIND1:TRAC:Y:PDIV 10 DB;RLEV 0 DB;RPOS 9"
1310 !
1320 ! Reduce the distractions on the display by
1330 ! getting rid of notation that will not be
1340 ! needed in this example.
1350 OUTPUT @Hp87xx;"DISP:ANN:YAX OFF"
1360 !
1370 ! Erase the graticule grid for the same reason.
1380 OUTPUT @Hp87xx;"DISP:WIND1:TRAC:GRAT:GRID OFF"
1390 !
1400 ! Set the markers for channel 1
1410 OUTPUT @Hp87xx;"CALC1:MARK1 ON"
1420 OUTPUT @Hp87xx;"CALC1:MARK1:X 152000000.000000"
1430 OUTPUT @Hp87xx;"CALC1:MARK2 ON"
1440 OUTPUT @Hp87xx;"CALC1:MARK2:X 200000000.000000"
1450 !
1460 ! Turn on marker statistics
1470 OUTPUT @Hp87xx;"CALC1:MARK:FUNC STATISTICS"
1480 !
1490 OUTPUT @Hp87xx;"CALC1:MARK2 ON"
1500 OUTPUT @Hp87xx;"CALC1:LIM:DISP ON"
1510 OUTPUT @Hp87xx;"CALC1:LIM:MARK:STAT:PEAK ON"
1520 !
1530 ! Turn on the pass/fail testing; watch the
1540 ! analyzer's display for the pass/fail indicator.
1550 OUTPUT @Hp87xx;"CALC1:LIM:STAT ON"
1560 !
```

```
1570 !  Set sweep hold mode
1580 OUTPUT @Hp87xx;"ABOR;:INIT1:CONT OFF;:INIT1;*WAI"
1590 !
1600 ! Send an operation complete query to ensure that
1610 ! all overlapped commands have been executed.
1620 OUTPUT @Hp87xx;"*OPC?"
1630 !
1640 ! Wait for the reply.
1650 ENTER @Hp87xx;Opc
1660 !
1670 ! Turn on a limit to be tested
1680 FOR Peak_limit=0. TO 3 STEP .1
1690     DISP "Peak limit test =",VAL$(Peak_limit)&" dB"
1700     OUTPUT @Hp87xx;"CALC1:LIM:MARK:STAT:PEAK:MAX "&VAL$(Peak_limit)
1710 !
1720 ! Send an operation complete query to ensure that
1730 ! all overlapped commands have been executed.
1740     OUTPUT @Hp87xx;"*OPC?"
1750 !
1760 ! Wait for the reply.
1770     ENTER @Hp87xx;Opc
1780 !
1790 ! Take a controlled sweep to ensure that
1800 ! there is real data present for the limit test.
1810     OUTPUT @Hp87xx;"INIT1;*OPC?"
1820     ENTER @Hp87xx;Opc
1830 !
1840 ! Query the limit fail condition register to see
1850 ! if there is a failure.
1860     OUTPUT @Hp87xx;"STAT:QUES:LIM:COND?"
1870 !
1880 ! Read the register's contents.
1890     ENTER @Hp87xx;Fail_flag
1900 !
1910 ! Bit 0 is the test result for channel 1 while
1920 ! Bit 1 is the results for channel 2 limit testing.
1930 ! Bit 2 is the result for channel 1 mkr limit testing.
1940 ! Bit 3 is the result for channel 2 mkr limit testing.
1950     IF BIT(Fail_flag,2)=1 THEN
1960 ! This limit test failed
1970     ELSE
1980         DISP "Passed at "&VAL$(Peak_limit)&" dB"
1990         BEEP
2000         GOTO Done
2010     END IF
2020 !
2030 NEXT Peak_limit
2040 Done:OUTPUT @Hp87xx;"INIT:CONT ON;*WAI"
2050 END
2060 !
2070 !**************************************************************
2080 ! Iden_port:   Identify io port to use.
2090 ! Description: This routines sets up the I/O port address for
2100 !              the SCPI interface.  For "HP 87xx" instruments,
2110 !              the address assigned to @Hp87xx = 800 otherwise,
2120 !              716.
2130 !**************************************************************
```

```
2140 SUB Iden_port
2150     COM /Sys_state/ @Hp87xx,Scode
2160 !
2170     IF POS(SYSTEM$("SYSTEM ID"),"HP 87")<>0 THEN
2180         ASSIGN @Hp87xx TO 800
2190         Scode=8
2200     ELSE
2210         ASSIGN @Hp87xx TO 716
2220         Scode=7
2230     END IF
2240 !
2250 SUBEND !Iden_port
2260 !
```

LIM_MEAN Example Program

```
1000 !Filename:  LIM_MEAN
1010 !
1020 ! This example program demonstrates how to test for a marker
1030 ! statistics mean limit.
1040 !
1050 ! Connect the demo filter to the analyzer RF out and RF in.
1060 ! The analyzer will set-up a transmission measurement.
1070 !
1080 ! The program will set various statistics mean limits, then query
1090 ! the status to determine if the specification PASSES or FAILS.
1100 !
1110 !
1120 !
1130 COM /Sys_state/ @Hp87xx,Scode
1140 ! Identify I/O Port
1150 CALL Iden_port
1160 !
1170 !
1180 ! Perform a system preset; this clears the limit table.
1190 OUTPUT @Hp87xx;"SYST:PRES;*WAI"
1200 !
1210 ! Set up the source frequencies for the measurement.
1220 OUTPUT @Hp87xx;"SENS1:FREQ:STAR 10 MHZ;STOP 400 MHZ;*WAI"
1230 !
1240 ! Set up the receiver for the measurement parameters
1250 ! (Transmission in this case).
1260 OUTPUT @Hp87xx;"SENS1:FUNC 'XFR:POW:RAT 2,0';DET NBAN;*WAI"
1270 !
1280 ! Configure the display so measurement
1290 ! results are easy to see.
1300 OUTPUT @Hp87xx;"DISP:WIND1:TRAC:Y:PDIV 10 DB;RLEV 0 DB;RPOS 9"
1310 !
1320 ! Reduce the distractions on the display by
1330 ! getting rid of notation that will not be
1340 ! needed in this example.
1350 OUTPUT @Hp87xx;"DISP:ANN:YAX OFF"
1360 !
1370 ! Erase the graticule grid for the same reason.
1380 OUTPUT @Hp87xx;"DISP:WIND1:TRAC:GRAT:GRID OFF"
1390 !
1400 ! Set the markers for channel 1
1410 OUTPUT @Hp87xx;"CALC1:MARK1 ON"
1420 OUTPUT @Hp87xx;"CALC1:MARK1:X 152000000.000000"
1430 OUTPUT @Hp87xx;"CALC1:MARK2 ON"
1440 OUTPUT @Hp87xx;"CALC1:MARK2:X 200000000.000000"
1450 !
1460 ! Turn on marker statistics
1470 OUTPUT @Hp87xx;"CALC1:MARK:FUNC STATISTICS"
1480 !
1490 OUTPUT @Hp87xx;"CALC1:MARK2 ON"
1500 OUTPUT @Hp87xx;"CALC1:LIM:DISP ON"
1510 OUTPUT @Hp87xx;"CALC1:LIM:MARK:STAT:MEAN ON"
1520 !
1530 ! Turn on the pass/fail testing; watch the
1540 ! analyzer's display for the pass/fail indicator.
1550 OUTPUT @Hp87xx;"CALC1:LIM:STAT ON"
1560 !
```

```
1570 !  Set sweep hold mode
1580 OUTPUT @Hp87xx;"ABOR;:INIT1:CONT OFF;:INIT1;*WAI"
1590 !
1600 ! Send an operation complete query to ensure that
1610 ! all overlapped commands have been executed.
1620 OUTPUT @Hp87xx;"*OPC?"
1630 !
1640 ! Wait for the reply.
1650 ENTER @Hp87xx;Opc
1660 !
1670 ! Turn on a limit to be tested
1680 FOR Mean_limit=0. TO -5 STEP -.1
1690     DISP "Mean limit test =",VAL$(Mean_limit)&" dB"
1700     OUTPUT @Hp87xx;"CALC1:LIM:MARK:STAT:MEAN:MIN "&VAL$(Mean_limit)
1710 !
1720 ! Send an operation complete query to ensure that
1730 ! all overlapped commands have been executed.
1740     OUTPUT @Hp87xx;"*OPC?"
1750 !
1760 ! Wait for the reply.
1770     ENTER @Hp87xx;Opc
1780 !
1790 ! Take a controlled sweep to ensure that
1800 ! there is real data present for the limit test.
1810     OUTPUT @Hp87xx;"INIT1;*OPC?"
1820     ENTER @Hp87xx;Opc
1830 !
1840 ! Query the limit fail condition register to see
1850 ! if there is a failure.
1860     OUTPUT @Hp87xx;"STAT:QUES:LIM:COND?"
1870 !
1880 ! Read the register's contents.
1890     ENTER @Hp87xx;Fail_flag
1900 !
1910 ! Bit 0 is the test result for channel 1 while
1920 ! Bit 1 is the results for channel 2 limit testing.
1930 ! Bit 2 is the result for channel 1 mkr limit testing.
1940 ! Bit 3 is the result for channel 2 mkr limit testing.
1950     IF BIT(Fail_flag,2)=1 THEN
1960 ! This limit test failed
1970     ELSE
1980         DISP "Passed at "&VAL$(Mean_limit)&" dB"
1990         BEEP
2000         GOTO Done
2010     END IF
2020 !
2030 NEXT Mean_limit
2040 Done:OUTPUT @Hp87xx;"INIT:CONT ON;*WAI"
2050 END
2060 !
2070 !**************************************************************
2080 ! Iden_port:   Identify io port to use.
2090 ! Description: This routines sets up the I/O port address for
2100 !              the SCPI interface.  For "HP 87xx" instruments,
2110 !              the address assigned to @Hp87xx = 800 otherwise,
2120 !              716.
2130 !**************************************************************
```

```
2140 SUB Iden_port
2150     COM /Sys_state/ @Hp87xx,Scode
2160 !
2170     IF POS(SYSTEM$("SYSTEM ID"),"HP 87")<>0 THEN
2180         ASSIGN @Hp87xx TO 800
2190         Scode=8
2200     ELSE
2210         ASSIGN @Hp87xx TO 716
2220         Scode=7
2230     END IF
2240 !
2250 SUBEND !Iden_port
2260 !
```

Making Multiport Test Set Measurements

PORT_SEL	This program uses graphics to show internal connections of the multiport test set when different ports are selected.
TSET_CAL	This program recalls "TSET_CAL.CAL" and performs a test set calibration.
MPCALSRQ	This IBASIC/RMB program requires an 8712ET/ES or 8714ET/ES analyzer. The program configures the analyzer's status reporting so that an SRQ is issued whenever a SelfCal is initiated. The SelfCal can be initiated by the automatic timer or by a front panel keypress.
MPSVRCL	This IBASIC/RMB program requires an 8712ET/ES or 8714ET/ES analyzer. The program saves three instrument states that measure various ports in ALT sweep mode. Fast recall is enabled so that subsequent recalls require a single keypress.

PORT_SEL Example Program

This program displays the internal connections of the multiport test set when different ports are selected. The internal connections of the multiport test set are drawn on the IBASIC display. Whenever the user selects a different port on the multiport test set, the program will redraw the internal connections.

This program also demonstrates how to use IBASIC to draw a fairly complicated drawing on the network analyzer.

NOTE This program requires a multiport test set.

```
1000 ! Filename:  PORT_SEL, 87050/87075 Port Selection Example
1010 !
1020 ! Description:
1030 !      This program demonstrate how the internal connections of a
1040 !      multiport testset are carried out when the different ports for
1050 !      Reflection and Transmission are selected.  It intends to show
1060 !      as an example of how to select the testset ports and how to draw
1070 !      draw graphics on the Ibasic window.
1080 !
1090 !
1100 ! NOTE:  This program works properly on analyzers installed with
1110 ! IBASIC.
1120 ! Modify to use DISP:WIND”;VAL$(Wind);” if IBASIC is not installed.
1130 !
1140 !
1150 !------------------------------------------------------------------
1160 ! Common Variables
1170 COM /Sys_state/ @Hp87xx,Scode,Keycode,Wind
1180 COM /Hp8711_coord/
Orig_x_8711,Orig_y_8711,Refl_x_8711,Refl_y_8711,Tran_x_8711,Tran_y_8711
1190 COM /Hp87075_coord/
Orig_x_87075,Orig_y_87075,Refl_x_87075,Refl_y_87075,Tran_x_87075,Tran_y_87075
1200 COM /Hp87075_ports/ Port_x(1:12),Port_y(1:12)
1210 COM /Color/ Erase,Bright,Dim
1220 COM /Sys_var/ Refl_port,Tran_port
1230 !------------------------------------------------------------------
1240 ! Identify I/O Port
1250 CALL Iden_port
1260 !
1270 OUTPUT @Hp87xx;”SYST:PRES; *WAI”       ! Preset the system
1280 CALL Setup_constant
1290 !
1300 ! Allocate an IBASIC display partition to show the graphics
1310 !
1320 OUTPUT @Hp87xx;”CONT1:MULT:STATE ON”   ! Make sure 87075 mode is enabled
1330 OUTPUT @Hp87xx;”DISP:FORM SING”
1340 OUTPUT @Hp87xx;”DISP:PROG:MODE FULL”
1350 OUTPUT @Hp87xx;”DISP:WIND”;VAL$(Wind);”:GRAP:SCAL 0,1023,0,383”
1360 !
1370 ! Clear the IBASIC display partition.
1380 OUTPUT @Hp87xx;”DISP:WIND”;VAL$(Wind);”:GRAP:CLE”
1390 !
```

```
1400 CALL Draw_analyzer
1410 CALL Draw_87075
1420 !
1430 ! Connect HP8711 to HP87075 for the Refl and Tran ports
1440 CALL Connect(Refl_x_8711,Refl_y_8711,Refl_x_87075,Refl_y_87075,Bright)
1450 CALL Connect(Tran_x_8711,Tran_y_8711,Tran_x_87075,Tran_y_87075,Bright)
1460 CALL Set_refl(1)
1470 CALL Set_tran(2)
1480 !
1490 ! Infinite loop to wait for softkey requests
1500 Do_loop:!
1510 GOSUB Setup_srq
1520 !
1530 GOTO Do_loop
1540 STOP
1550 !
1560 !-------------------------------------------------------------------
1570 ! Setup interrupts
1580 !
1590 Setup_srq:!
1600 ! If using an external controller...
1610 !
1620 ! Initialize flag for checking on keyboard
1630 ! interrupts.
1640 Keycode=-1
1650 !
1660 ! Label softkey 1.
1670 OUTPUT @Hp87xx;"DISP:MENU:KEY1 'Reflection to Port #'"
1680 OUTPUT @Hp87xx;"DISP:MENU:KEY2 'Transmissn to Port #'"
1690 OUTPUT @Hp87xx;"DISP:MENU:KEY5 'Done'"
1700 !
1710 ! Clear the status register and event status
1720 ! register.
1730 OUTPUT @Hp87xx;"*CLS;*ESE 0"
1740 ! Preset the other status registers.
1750 ! Enable the Device Status register to report
1760 ! to the Status Byte on positive transition
1770 ! of bit 0 (key press).  Enable the Status
1780 ! Byte to generate an interrupt when the
1790 ! Device Status register's summary bit
1800 ! changes.
1810 OUTPUT @Hp87xx;"STAT:PRES;DEV:ENAB 1;*SRE 4"
1820 !
1830 ! Clear the key queue to ensure that previous
1840 ! key presses  do not generate an interrupt.
1850 OUTPUT @Hp87xx;"SYST:KEY:QUE:CLE"
1860 !
1870 ! Set up and enable the interrupt on the HP-IB
1880 ! when a service request is received.
1890 ON INTR Scode,5 RECOVER Srq
1900 ENABLE INTR Scode;2
1910 Suspend: !WAIT 5                      ! Use WAIT 'n' to suspend IBASIC
1920 GOTO Suspend
1930 !
1940 !----------------------------------------------------------------
1950 ! Interrupt Handler
1960 !
1970 Srq:   !
1980 !
1990 ! Do a serial poll to find out if analyzer generated the
```

```
2000 ! interrupt.
2010 Stb=SPOLL(@Hp87xx)
2020 !
2030 ! Determine if the Device Status register’s summary
2040 ! bit (bit 2 of the Status Byte) has been set.
2050 IF BINAND(Stb,4)<>0 THEN
2060 !
2070 ! If so, then get the Device Status Register contents.
2080     OUTPUT @Hp87xx;”STAT:DEV:EVEN?”
2090     ENTER @Hp87xx;Dev_event
2100 !
2110 ! Check for key press...
2120     IF BINAND(Dev_event,1)<>0 THEN
2130 ! If so, then determine which key.
2140         OUTPUT @Hp87xx;”SYST:KEY?”
2150         ENTER @Hp87xx;Keycode
2160     END IF
2170 END IF
2180 !
2190 ! Reenable the interrupt in case wrong key
2200 ! was pressed.
2210 CALL Softkey_handler
2220 ENABLE INTR Scode
2230 !
2240 RETURN
2250 END
2260 !
2270 !------------------------------------------------------------------
2280 ! Subroutines
2290 !
2300 !
2310 ! Setup_constant
2320 !   Setup all global constants
2330 SUB Setup_constant
2340     COM /Sys_state/ @Hp87xx,Scode,Keycode,Wind
2350     COM /Hp8711_coord/
Orig_x_8711,Orig_y_8711,Refl_x_8711,Refl_y_8711,Tran_x_8711,Tran_y_8711
2360     COM /Hp87075_coord/
Orig_x_87075,Orig_y_87075,Refl_x_87075,Refl_y_87075,Tran_x_87075,Tran_y_87075
2370     COM /Hp87075_ports/ Port_x(1:12),Port_y(1:12)
2380     COM /Color/ Erase,Bright,Dim
2390     COM /Sys_var/ Refl_port,Tran_port
2400     Orig_x_8711=30
2410     Orig_y_8711=170
2420     Refl_x_8711=Orig_x_8711+300
2430     Refl_y_8711=Orig_y_8711+20
2440     Tran_x_8711=Orig_x_8711+410
2450     Tran_y_8711=Orig_y_8711+20
2460     Orig_x_87075=30
2470     Orig_y_87075=30
2480     Refl_x_87075=Orig_x_87075+300
2490     Refl_y_87075=Orig_y_87075+80
2500     Tran_x_87075=Orig_x_87075+410
2510     Tran_y_87075=Orig_y_87075+80
2520     FOR I=1 TO 11 STEP 2
2530         Port_x(I)=Orig_x_87075+100+(((I-1)/2)*60)
2540         Port_y(I)=Orig_y_87075+45
2550     NEXT I
2560     FOR I=2 TO 12 STEP 2
2570         Port_x(I)=Orig_x_87075+130+(((I-2)/2)*60)
```

```
2580        Port_y(I)=Orig_y_87075+25
2590     NEXT I
2600     Erase=0
2610     Bright=1
2620     Dim=2
2630     Wind=10
2640     Refl_port=0
2650     Tran_port=0
2660 SUBEND
2670 !
2680 !--------------------------------------------------------------
2690 !  Drawing routines
2700 !
2710 !  Draw_analyzer
2720 !   Draw an HP8711 Analyzer on the Ibasic window
2730 SUB Draw_analyzer
2740     COM /Sys_state/ @Hp87xx,Scode,Keycode,Wind
2750     COM /Hp8711_coord/
Orig_x_8711,Orig_y_8711,Refl_x_8711,Refl_y_8711,Tran_x_8711,Tran_y_8711
2760 ! Select the bright “pen” and bold font.
2770     OUTPUT @Hp87xx;”DISP:WIND”;VAL$(Wind);”:GRAP:COL 1;LAB:FONT BOLD”
2780 !
2790 ! Draw a label reading “HP 8711” at 30 pixels
2800 ! to the right and 270 pixels above the origin.
2810 ! The origin is the lower left corner of the
2820 ! current graphics window
2830     OUTPUT @Hp87xx;”DISP:WIND”;VAL$(Wind);”:GRAP:MOVE
“;Orig_x_8711;”,”;Orig_y_8711+120+10;”;LAB ‘HP 8711C’”
2840 !
2850 ! Draw a box to represent the analyzer.
2860     OUTPUT @Hp87xx;”DISP:WIND”;VAL$(Wind);”:GRAP:MOVE
“;Orig_x_8711;”,”;Orig_y_8711
2870     OUTPUT @Hp87xx;”DISP:WIND”;VAL$(Wind);”:GRAP:RECT 480,120”
2880     OUTPUT @Hp87xx;”DISP:WIND”;VAL$(Wind);”:GRAP:MOVE
“;Orig_x_8711+20;”,”;Orig_y_8711+10
2890     OUTPUT @Hp87xx;”DISP:WIND”;VAL$(Wind);”:GRAP:RECT 210,100”
2900     OUTPUT @Hp87xx;”DISP:WIND”;VAL$(Wind);”:GRAP:MOVE
“;Refl_x_8711;”,”;Refl_y_8711
2910     OUTPUT @Hp87xx;”DISP:WIND”;VAL$(Wind);”:GRAP:CIRC 4”
2920     OUTPUT @Hp87xx;”DISP:WIND”;VAL$(Wind);”:GRAP:MOVE
“;Tran_x_8711;”,”;Tran_y_8711
2930     OUTPUT @Hp87xx;”DISP:WIND”;VAL$(Wind);”:GRAP:CIRC 4”
2940     OUTPUT @Hp87xx;”DISP:WIND”;VAL$(Wind);”:GRAP:COL 1;LAB:FONT SLAN”
2950     OUTPUT @Hp87xx;”DISP:WIND”;VAL$(Wind);”:GRAP:MOVE
“;Refl_x_8711-40;”,”;Refl_y_8711+10
2960     OUTPUT @Hp87xx;”DISP:WIND”;VAL$(Wind);”:GRAP:LAB ‘RF OUT’”
2970     OUTPUT @Hp87xx;”DISP:WIND”;VAL$(Wind);”:GRAP:COL 1;LAB:FONT SLAN”
2980     OUTPUT @Hp87xx;”DISP:WIND”;VAL$(Wind);”:GRAP:MOVE
“;Tran_x_8711-20;”,”;Tran_y_8711+10
2990     OUTPUT @Hp87xx;”DISP:WIND”;VAL$(Wind);”:GRAP:LAB ‘RF IN’”
3000 SUBEND
3010 !
3020 ! Draw_87075
3030 !   Draw an 87075 Multiport test set with twelve port setups
3040 SUB Draw_87075
3050     COM /Sys_state/ @Hp87xx,Scode,Keycode,Wind
3060     COM /Hp87075_coord/
Orig_x_87075,Orig_y_87075,Refl_x_87075,Refl_y_87075,Tran_x_87075,Tran_y_87075
3070     COM /Hp87075_ports/ Port_x(1:12),Port_y(1:12)
3080 ! Select the bright “pen” and bold font.
```

```
3090     OUTPUT @Hp87xx;”DISP:WIND”;VAL$(Wind);”:GRAP:COL 1;LAB:FONT BOLD”
3100 !
3110 !
3120 ! Draw a label reading “HP 87075” at 30 pixels
3130 ! to the right and 110 pixels above the origin.
3140 ! The origin is the lower left corner of the
3150 ! current graphics window
3160     OUTPUT @Hp87xx;”DISP:WIND”;VAL$(Wind);”:GRAP:MOVE
“;Orig_x_87075;”,”;Orig_y_87075+100+10;”;LAB ‘HP 87075’”
3170 !
3180     OUTPUT @Hp87xx;”DISP:WIND”;VAL$(Wind);”:GRAP:MOVE
“;Orig_x_87075;”,”;Orig_y_87075
3190     OUTPUT @Hp87xx;”DISP:WIND”;VAL$(Wind);”:GRAP:RECT 480,110”
3200     OUTPUT @Hp87xx;”DISP:WIND”;VAL$(Wind);”:GRAP:MOVE
“;Refl_x_87075;”,”;Refl_y_87075
3210     OUTPUT @Hp87xx;”DISP:WIND”;VAL$(Wind);”:GRAP:CIRC 4”
3220     OUTPUT @Hp87xx;”DISP:WIND”;VAL$(Wind);”:GRAP:MOVE
“;Tran_x_87075;”,”;Tran_y_87075
3230     OUTPUT @Hp87xx;”DISP:WIND”;VAL$(Wind);”:GRAP:CIRC 4”
3240     OUTPUT @Hp87xx;”DISP:WIND”;VAL$(Wind);”:GRAP:COL 1;LAB:FONT SLAN”
3250     OUTPUT @Hp87xx;”DISP:WIND”;VAL$(Wind);”:GRAP:MOVE
“;Refl_x_87075-40;”,”;Refl_y_87075+10
3260     OUTPUT @Hp87xx;”DISP:WIND”;VAL$(Wind);”:GRAP:LAB ‘REFL’”
3270     OUTPUT @Hp87xx;”DISP:WIND”;VAL$(Wind);”:GRAP:COL 1;LAB:FONT SLAN”
3280     OUTPUT @Hp87xx;”DISP:WIND”;VAL$(Wind);”:GRAP:MOVE
“;Tran_x_87075-20;”,”;Tran_y_87075+10
3290     OUTPUT @Hp87xx;”DISP:WIND”;VAL$(Wind);”:GRAP:LAB ‘TRAN’”
3300     FOR I=1 TO 12
3310         OUTPUT @Hp87xx;”DISP:WIND”;VAL$(Wind);”:GRAP:MOVE
“;Port_x(I);”,”;Port_y(I)
3320         OUTPUT @Hp87xx;”DISP:WIND”;VAL$(Wind);”:GRAP:CIRC 4”
3330         OUTPUT @Hp87xx;”DISP:WIND”;VAL$(Wind);”:GRAP:COL 1;LAB:FONT SLAN”
3340         OUTPUT @Hp87xx;”DISP:WIND”;VAL$(Wind);”:GRAP:MOVE
“;Port_x(I)-8;”,”;Port_y(I)-18
3350         OUTPUT @Hp87xx;”DISP:WIND”;VAL$(Wind);”:GRAP:LAB ‘”;VAL$(I);”’”
3360     NEXT I
3370 SUBEND
3380 !
3390 !------------------------------------------------------------------
3400 ! Connection routines
3410 !
3420 ! Connect
3430 !   Connect (x1,y1) to (x2,y2) with the specied color ‘Col’
3440 !   If Color = 0, it will be an erase command instead.
3450 SUB Connect(X1,Y1,X2,Y2,Col)
3460     COM /Sys_state/ @Hp87xx,Scode,Keycode,Wind
3470     OUTPUT @Hp87xx;”DISP:WIND”;VAL$(Wind);”:GRAP:COL “;Col
3480     OUTPUT @Hp87xx;”DISP:WIND”;VAL$(Wind);”:GRAP:MOVE “;X1;”,”;Y1
3490     OUTPUT @Hp87xx;”DISP:WIND”;VAL$(Wind);”:GRAP:DRAW “;X2;”,”;Y2
3500 SUBEND
3510 !
3520 ! Connect_refl
3530 !   Connect the reflection port to the specified port index I with
3540 !   color ‘Col’.  Use Col=0 to erase the connection.  This routine
3550 !   uses the port coordinates from Port_x(1:12) and Port_y(1:12)
3560 SUB Connect_refl(I,Col)
3570     COM /Sys_state/ @Hp87xx,Scode,Keycode,Wind
3580     COM /Hp8711_coord/
Orig_x_8711,Orig_y_8711,Refl_x_8711,Refl_y_8711,Tran_x_8711,Tran_y_8711
```

```
3590     COM /Hp87075_coord/
Orig_x_87075,Orig_y_87075,Refl_x_87075,Refl_y_87075,Tran_x_87075,Tran_y_87075
3600     COM /Hp87075_ports/ Port_x(1:12),Port_y(1:12)
3610 !
3620     Temp_y=Refl_y_87075-10
3630     Connect(Refl_x_87075,Refl_y_87075,Refl_x_87075,Temp_y,Col)
3640     Connect(Refl_x_87075,Temp_y,Port_x(I),Temp_y,Col)
3650     Connect(Port_x(I),Temp_y,Port_x(I),Port_y(I),Col)
3660 SUBEND
3670 !
3680 !
3690 ! Connect_tran
3700 !   Connect the transmission port to the specified port index I with
3710 !   color ‘Col’.  Use Col=0 to erase the connection.  This routine
3720 !   uses the port coordinates from Port_x(1:12) and Port_y(1:12)
3730 SUB Connect_tran(I,Col)
3740     COM /Sys_state/ @Hp87xx,Scode,Keycode,Wind
3750     COM /Hp8711_coord/
Orig_x_8711,Orig_y_8711,Refl_x_8711,Refl_y_8711,Tran_x_8711,Tran_y_8711
3760     COM /Hp87075_coord/
Orig_x_87075,Orig_y_87075,Refl_x_87075,Refl_y_87075,Tran_x_87075,Tran_y_87075
3770     COM /Hp87075_ports/ Port_x(1:12),Port_y(1:12)
3780 !
3790     Temp_y=Refl_y_87075-20
3800     Connect(Tran_x_87075,Tran_y_87075,Tran_x_87075,Temp_y,Col)
3810     Connect(Tran_x_87075,Temp_y,Port_x(I),Temp_y,Col)
3820     Connect(Port_x(I),Temp_y,Port_x(I),Port_y(I),Col)
3830 SUBEND
3840 !
3850 !------------------------------------------------------------------
3860 ! Softkey handle routines
3870 !
3880 ! Select_refl
3890 !   Select the reflection port by requesting a valid port number from
3900 !   the user.  The input port number is used to select the reflection
3910 !   port accordingly.  This routine will also update the drawing
3920 !   connections on the Ibasic window.  Any invalid number will be
3930 !   ignored.
3940 SUB Select_refl
3950     COM /Sys_state/ @Hp87xx,Scode,Keycode,Wind
3960     COM /Hp8711_coord/
Orig_x_8711,Orig_y_8711,Refl_x_8711,Refl_y_8711,Tran_x_8711,Tran_y_8711
3970     COM /Hp87075_coord/
Orig_x_87075,Orig_y_87075,Refl_x_87075,Refl_y_87075,Tran_x_87075,Tran_y_87075
3980     COM /Color/ Erase,Bright,Dim
3990     COM /Sys_var/ Refl_port,Tran_port
4000 !
4010     OUTPUT @Hp87xx;”DISP:WIND”;VAL$(Wind);”:GRAP:COL “;Bright
4020     OUTPUT @Hp87xx;”*OPC?”
4030     ENTER @Hp87xx;Opc
4040     INPUT “Connect Reflection to Port #:”,P
4050     CALL Set_refl(P)
4060 SUBEND
4070 !
4080 !
4090 ! Set_refl
4100 !   Update the currently selected reflection port with the specified
4110 !   port ‘P’.  Update the connection drawing on the Ibasic window.
4120 SUB Set_refl(P)
4130     COM /Sys_state/ @Hp87xx,Scode,Keycode,Wind
```

```
4140     COM /Hp8711_coord/
Orig_x_8711,Orig_y_8711,Refl_x_8711,Refl_y_8711,Tran_x_8711,Tran_y_8711
4150     COM /Hp87075_coord/
Orig_x_87075,Orig_y_87075,Refl_x_87075,Refl_y_87075,Tran_x_87075,Tran_y_87075
4160     COM /Color/ Erase,Bright,Dim
4170     COM /Sys_var/ Refl_port,Tran_port
4180 !
4190     OUTPUT @Hp87xx;"ROUT:REFL:PATH:DEFine:PORT ";P
4200     OUTPUT @Hp87xx;"ROUT:REFL:PATH:DEFine:PORT?"
4210     ENTER @Hp87xx;New_refl
4220     OUTPUT @Hp87xx;"ROUT:TRAN:PATH:DEFine:PORT?"
4230     ENTER @Hp87xx;New_tran
4240     Update_ports(New_refl,New_tran)
4250 SUBEND
4260 !
4270 !
4280 !
4290 ! Select_tran
4300 !   Select the transmission port by requesting a valid port number from
4310 !   the user.  The input port number is used to select the transmission
4320 !   port accordingly.  This routine will also update the drawing
4330 !   connections on the Ibasic window.  Any invalid number will be
4340 !   ignored.
4350 SUB Select_tran
4360     COM /Sys_state/ @Hp87xx,Scode,Keycode,Wind
4370     COM /Hp8711_coord/
Orig_x_8711,Orig_y_8711,Refl_x_8711,Refl_y_8711,Tran_x_8711,Tran_y_8711
4380     COM /Hp87075_coord/
Orig_x_87075,Orig_y_87075,Refl_x_87075,Refl_y_87075,Tran_x_87075,Tran_y_87075
4390     COM /Color/ Erase,Bright,Dim
4400     COM /Sys_var/ Refl_port,Tran_port
4410 !
4420     OUTPUT @Hp87xx;"DISP:WIND";VAL$(Wind);":GRAP:COL ";Bright
4430     OUTPUT @Hp87xx;"*OPC?"
4440     ENTER @Hp87xx;Opc
4450     INPUT "Connect Transmission to Port #:",P
4460     CALL Set_tran(P)
4470 SUBEND
4480 !
4490 !
4500 ! Set_tran
4510 !   Update the currently selected transmission port with the specified
4520 !   port 'P'.  Update the connection drawing on the Ibasic window.
4530 SUB Set_tran(P)
4540     COM /Sys_state/ @Hp87xx,Scode,Keycode,Wind
4550     COM /Hp8711_coord/
Orig_x_8711,Orig_y_8711,Refl_x_8711,Refl_y_8711,Tran_x_8711,Tran_y_8711
4560     COM /Hp87075_coord/
Orig_x_87075,Orig_y_87075,Refl_x_87075,Refl_y_87075,Tran_x_87075,Tran_y_87075
4570     COM /Color/ Erase,Bright,Dim
4580     COM /Sys_var/ Refl_port,Tran_port
4590 !
4600     OUTPUT @Hp87xx;"ROUT:TRAN:PATH:DEFine:PORT ";P
4610     OUTPUT @Hp87xx;"ROUT:TRAN:PATH:DEFine:PORT?"
4620     ENTER @Hp87xx;New_tran
4630     OUTPUT @Hp87xx;"ROUT:REFL:PATH:DEFine:PORT?"
4640     ENTER @Hp87xx;New_refl
4650     Update_ports(New_refl,New_tran)
4660 SUBEND
4670 !
```

```
4680 ! Update_ports
4690 !   Update the currently selected ports.  Erase old connections.
4700 !   Draw new connections.
4710 SUB Update_ports(Refl,Tran)
4720     COM /Color/ Erase,Bright,Dim
4730     COM /Sys_var/ Refl_port,Tran_port
4740     IF Tran_port=0 THEN
4750         Tran_port=Tran
4760     ELSE
4770         IF Tran<>Tran_port THEN
4780             Connect_tran(Tran_port,Erase)
4790             Tran_port=Tran
4800         END IF
4810     END IF
4820     IF Refl_port=0 THEN
4830         Refl_port=Refl
4840     ELSE
4850         IF Refl<>Refl_port THEN
4860             Connect_refl(Refl_port,Erase)
4870             Refl_port=Refl
4880         END IF
4890     END IF
4900     Connect_tran(Tran_port,Dim)
4910     Connect_refl(Refl_port,Dim)
4920 SUBEND
4930 !
4940 ! Softkey_handler
4950 !   Call from Srq to handler all softkey requests.  Terminate program
4960 !   when ‘Done’ is pressed.
4970 SUB Softkey_handler
4980     COM /Sys_state/ @Hp87xx,Scode,Keycode,Wind
4990 !
5000     IF Keycode=0 THEN
5010         CALL Select_refl
5020     ELSE
5030         IF Keycode=1 THEN
5040             CALL Select_tran
5050         ELSE
5060             IF Keycode=4 THEN
5070                 OUTPUT @Hp87xx;”SYST:PRES; *WAI”       ! Preset the system
5080                 STOP
5090             END IF
5100         END IF
5110     END IF
5120 SUBEND
5130 !
5140 !-------------------------------------------------------------------
5150 ! Misc routines
5160 !
5170 !*************************************************************
5180 ! Iden_port:   Identify io port to use
5190 ! Description: This routines sets up the I/O port address for
5200 !              the SCPI interface.  For “HP 87xx” instruments,
5210 !              the address assigned to @Hp87xx = 800 otherwise,
5220 !              716.
5230 !*************************************************************
5240 SUB Iden_port
5250     COM /Sys_state/ @Hp87xx,Scode,Keycode,Wind
5260 !
5270     IF POS(SYSTEM$(“SYSTEM ID”),”HP 87”)<>0 THEN
```

```
5280         ASSIGN @Hp87xx TO 800
5290         Scode=8
5300     ELSE
5310         ASSIGN @Hp87xx TO 716
5320         Scode=7
5330     END IF
5340 !
5350 SUBEND!Iden_port
5360 !
```

TSET_CAL Example Program

This program automates the process of recalling a multiport test set cal. This program first attempts to recall TSET_CAL.CAL from non-volatile RAM, then the internal 3.5” disk drive. If the recall is successful, it invokes the recalled test set cal for transmission and reflection of measurement channels 1 and 2.

Complete a test set calibration over the appropriate frequency range before using TSET_CAL.

```
1000 ! Filename: TSET_CAL, recall test set cal
1010 !
1020 ! Description:
1030 !    This program will try to recall tset_cal.cal from NVRAM if file
1040 !    is present.  If not, it will then try to recall tset_cal.cal from
1050 !    INT device instead.  If recall successful, it will then do test
1060 !    set cal for Transmission and Reflection of Channel 1 and Channel
         2.
1070
!
1080 ! Common Variables
1090 COM /Sys_state/ @Hp87xx,Scode,Errnum
1100 ! Identify I/O Port
1110 CALL Iden_port
1120 OUTPUT @Hp87xx;"syst:pres;*wai"      ! Reset the instrument
1130 CALL Recall_tset_cal
1140 IF (Errnum=0) THEN
1150     PRINT "Doing Test set cal................."
1160     CALL Tsetcal
1170     PRINT "Test Set cal complete"
1180 END IF
1190 !
1200 OUTPUT @Hp87xx;"syst:pres;*wai"      ! Reset the instrument
1210 !
1220 STOP
1230 END
1240 !
1250 !**************************************************************
1260 ! TsetCal:   Do Test set Cal.
1270 ! Description:  This routine will do test set cal for both
1280 !               Transmission and Reflection of Channel 1 and
1290 !               Channel 2.
1300 !**************************************************************
1310 SUB Tsetcal
1320     COM /Sys_state/ @Hp87xx,Scode,Errnum
1330 !
1340 !   Do Test set cal for channel 1 Transmission
1350     OUTPUT @Hp87xx;"sens1:stat ON; *wai"
1360     OUTPUT @Hp87xx;"sens1:func 'xfr:pow:rat 2,0';det nban; *wai"
1370     OUTPUT @Hp87xx;"sens1:corr:testset; *wai"
1380 !
1390 !   Do Test Set cal for channel 1 Reflection
1400     OUTPUT @Hp87xx;"sens1:stat ON; *wai"
1410     OUTPUT @Hp87xx;"sens1:func 'xfr:pow:rat 1,0';det nban; *wai"
1420     OUTPUT @Hp87xx;"sens1:corr:testset; *wai"
1430 !
```

```
1440 !   Do Test set cal for channel 2 Transmission
1450     OUTPUT @Hp87xx;"sens2:stat ON; *wai"
1460     OUTPUT @Hp87xx;"sens2:func 'xfr:pow:rat 2,0';det nban; *wai"
1470     OUTPUT @Hp87xx;"sens2:corr:testset; *wai"
1480 !
1490 !   Do Test Set cal for channel 2 Reflection
1500     OUTPUT @Hp87xx;"sens2:stat ON; *wai"
1510     OUTPUT @Hp87xx;"sens2:func 'xfr:pow:rat 1,0';det nban; *wai"
1520     OUTPUT @Hp87xx;"sens2:corr:testset; *wai"
1530 SUBEND
1540 !

1550 !**************************************************************
1560 ! Recall_tset_cal:   Recall Test Set Cal.
1570 ! Description:  This routine will try to recall tset_cal.cal
1580 !               from NVRAM if available.  If file not found,
1590 !               it will then try to recall tset_cal.cal from
1600 !               INT device instead.
1610 !**************************************************************
1620 SUB Recall_tset_cal
1630     COM /Sys_state/ @Hp87xx,Scode,Errnum
1640     DIM E$[120]
1650 !
1660 !   Clear the status register and event status
1670 !   register.
1680     OUTPUT @Hp87xx;"*cls;*ese 0"
1690     PRINT "Recalling tset_cal.cal from NVRAM device.........."
1700     OUTPUT @Hp87xx;"mmem:load:stat 1,'mem:tset_cal.cal'"
1710     OUTPUT @Hp87xx;"syst:err?"
1720     ENTER @Hp87xx;Errnum,E$
1730     IF (Errnum<>0) THEN
1740         PRINT E$
1750         PRINT "Recalling tset_cal.cal from INT device.........."
1760         OUTPUT @Hp87xx;"mmem:load:stat 1,'int:tset_cal.cal'"
1770         OUTPUT @Hp87xx;"syst:err?"
1780         ENTER @Hp87xx;Errnum,E$
1790         IF (Errnum=0) THEN
1800             PRINT "Recall complete"
1810         ELSE
1820             PRINT "TSET_CAL.CAL is not present, recall aborted"
1830         END IF
1840     ELSE
1850         PRINT "Recall complete"
1860     END IF
1870 !
1880 SUBEND!Recall_tset_cal
1890 !
1900 !
1910 !

1920 !**************************************************************
1930 ! Iden_port:   Identify io port to use.
1940 ! Description: This routines sets up the I/O port address for
1950 !              the SCPI interface.  For "HP 87xx" instruments,
1960 !              the address assigned to @Hp87xx = 800 otherwise,
1970 !              716.
1980 !**************************************************************
1990 SUB Iden_port
2000     COM /Sys_state/ @Hp87xx,Scode,Errnum
2010 !
```

```
2020     IF POS(SYSTEM$("SYSTEM ID"),"HP 87")<>0 THEN
2030         ASSIGN @Hp87xx TO 800
2040         Scode=8
2050     ELSE
2060         ASSIGN @Hp87xx TO 716
2070         Scode=7
2080     END IF
2090 !
2100 SUBEND!Iden_port
2110 !
```

MPCALSRQ Example Program

This IBASIC/RMB program requires an 8712ET/ES or 8714ET/ES analyzer. The program configures the analyzer’s status reporting so that an SRQ is issued whenever a SelfCal is initiated. The SelfCal can be initiated by the automatic timer or by a front panel keypress.

```
1000 !Filename:  MPCALSRQ
1010 !
1020 !-----------------------------------------------------------------------
1030 ! Description:
1040 !
1050 !   Before running this program, enable a TESTSET cal for your
1060 !   87075C or 87050E multiport testset.  Select valid ports
1070 !   and frequency ranges.
1080 !
1090 !   Enable [Periodic Selfcal] and set the Selfcal Timer
1100 !   so that the analyzer self calibration will be
1110 !   triggered periodically on your measurements.
1120 !
1130 !   Run the program.
1140 !
1150 !   The program configures an SRQ to occur when multiport
1160 !   selfcal happens.  After the instrument is configured the program
1170 !   will wait until an SRQ is recieved.  The service routine will poll
1180 !   the device to look for the selfcal condition.
1190 !
1200 !   A message is displayed while a selfcal is in progress and after
1210 !   the selfcal has completed.
1220 !
1230 !------------------------------------------------------------------------
1240 !
1250 COM /Sys_state/ @Hp87xx,Scode
1260 ! Identify I/O Port
1270 CALL Iden_port
1280 !
1290 !
1300 ! Clear status registers.
1310 Start:OUTPUT @Hp87xx;”*CLS”
1320 !
1330 ! Clear the Service Request Enable register.
1340 OUTPUT @Hp87xx;”*SRE 0”
1350 !
1360 ! Clear the Standard Event Status Enable register.
1370 OUTPUT @Hp87xx;”*ESE 0”
1380 !
1390 ! Preset the remaining status registers.
1400 OUTPUT @Hp87xx;”STAT:PRES”
1410 !
1420 ! Set operation status register to report
1430 ! to the status byte on POSITIVE transition of
1440 ! the correcting bit.
1450 OUTPUT @Hp87xx;”STAT:OPER:NTR #H0000”
1460 OUTPUT @Hp87xx;”STAT:OPER:PTR #HFFFF”
1470 OUTPUT @Hp87xx;”STAT:OPER:ENAB 128”
1480 !
1490 ! Enable the operational status bit in the status
1500 ! byte to generate an SRQ.
```

```
1510 OUTPUT @Hp87xx;”*SRE 128”
1520 !
1530 ! On an interrupt from HP-IB “Scode” (Interface
1540 ! Select Code) SRQ bit (2), branch to the interrupt
1550 ! service routine “Srq_handler”.
1560 ON INTR Scode,2 GOSUB Srq_handler
1570 !
1580 ! Now enable the interrupt on SRQ (Service Request).
1590 ENABLE INTR Scode;2
1600 !
1610 !
1620 ! Initialize flag indicating when selfcal done
1630 ! to 0.  Then loop continuously until the
1640 ! interrupt is detected, and the interrupt
1650 ! service routine acknowledges the
1660 ! interrupt and sets the flag to 1.
1670 Selfcal_done=0
1680 DISP “Waiting for SRQ on selfcal”
1690 LOOP
1700 EXIT IF Selfcal_done=1
1710 END LOOP
1720 !
1730 ! Display desired completion message.
1740 !DISP
1750 WAIT 10
1760 GOTO Start
1770 !
1780 Srq_handler: ! Interrupt Service Routine
1790 !
1800 ! Determine that the analyzer was actually
1810 ! the instrument that generated the
1820 ! interrupt.
1830 Stb=SPOLL(@Hp87xx)
1840 !
1850 ! Determine if the operation status register
1860 ! caused the interrupt by looking at bit 7
1870 ! of the result of the serial poll.
1880 DISP “Checking SRQ...”
1890 IF BINAND(Stb,128)<>0 THEN
1900 !
1910 ! Read the operational status event register.
1920 OUTPUT @Hp87xx;”STAT:OPER:EVEN?”
1930 ENTER @Hp87xx;Op_event
1940 !
1950 ! Determine if the correcting status register
1960 ! bit 7 is set.
1970 DISP “Got SRQ.  Selfcal complete!”,Op_event
1980 BEEP
1990 IF BINAND(Op_event,128)<>0 THEN
2000 !
2010 ! If so, then set flag indicating
2020 ! selfcal done.
2030 Selfcal_done=1
2040 END IF
2050 END IF
2060 RETURN
2070 END
2080 !
2090 !**************************************************************
2100 ! Iden_port:   Identify io port to use
```

```
2110 ! Description: This routines sets up the I/O port address for
2120 !              the SCPI interface.  For “HP 87xx” instruments,
2130 !              the address assigned to @Hp87xx = 800 otherwise,
2140 !              716.
2150 !*************************************************************
2160 SUB Iden_port
2170 COM /Sys_state/ @Hp87xx,Scode
2180 !
2190 IF POS(SYSTEM$(“SYSTEM ID”),”HP 87”)<>0 THEN
2200 ASSIGN @Hp87xx TO 800
2210 Scode=8
2220 ELSE
2230 ASSIGN @Hp87xx TO 716
2240 Scode=7
2250 END IF
2260 !
2270 SUBEND!Iden_port
2280 !
```

MPSVRCL Example Program

This IBASIC/RMB program requires an 8712ET/ES or 8714ET/ES analyzer. The program saves three instrument states that measure various ports in ALT sweep mode. Fast recall is enabled so that subsequent recalls require only a single keypress.

```
1000 !Filename:  MPCALSRQ
1010 !
1020 !-----------------------------------------------------------------------
1030 ! Description:
1040 !
1050 !   Before running this program, enable a TESTSET cal for your
1060 !   87075C or 87050E multiport testset.  Select valid ports
1070 !   and frequency ranges.
1080 !
1090 !   Enable [Periodic Selfcal] and set the Selfcal Timer
1100 !   so that the analyzer self calibration will be
1110 !   triggered periodically on your measurements.
1120 !
1130 !   Run the program.
1140 !
1150 !   The program configures an SRQ to occur when multiport
1160 !   selfcal happens.  After the instrument is configured the program
1170 !   will wait until an SRQ is recieved.  The service routine will poll
1180 !   the device to look for the selfcal condition.
1190 !
1200 !   A message is displayed while a selfcal is in progress and after
1210 !   the selfcal has completed.
1220 !
1230 !------------------------------------------------------------------------
1240 !
1250 COM /Sys_state/ @Hp87xx,Scode
1260 ! Identify I/O Port
1270 CALL Iden_port
1280 !
1290 !
1300 ! Clear status registers.
1310 Start:OUTPUT @Hp87xx;"*CLS"
1320 !
1330 ! Clear the Service Request Enable register.
1340 OUTPUT @Hp87xx;"*SRE 0"
1350 !
1360 ! Clear the Standard Event Status Enable register.
1370 OUTPUT @Hp87xx;"*ESE 0"
1380 !
1390 ! Preset the remaining status registers.
1400 OUTPUT @Hp87xx;"STAT:PRES"
1410 !
1420 ! Set operation status register to report
1430 ! to the status byte on POSITIVE transition of
1440 ! the correcting bit.
1450 OUTPUT @Hp87xx;"STAT:OPER:NTR #H0000"
1460 OUTPUT @Hp87xx;"STAT:OPER:PTR #HFFFF"
1470 OUTPUT @Hp87xx;"STAT:OPER:ENAB 128"
1480 !
1490 ! Enable the operational status bit in the status
1500 ! byte to generate an SRQ.
```

```
1510 OUTPUT @Hp87xx;"*SRE 128"
1520 !
1530 ! On an interrupt from HP-IB "Scode" (Interface
1540 ! Select Code) SRQ bit (2), branch to the interrupt
1550 ! service routine "Srq_handler".
1560 ON INTR Scode,2 GOSUB Srq_handler
1570 !
1580 ! Now enable the interrupt on SRQ (Service Request).
1590 ENABLE INTR Scode;2
1600 !
1610 !
1620 ! Initialize flag indicating when selfcal done
1630 ! to 0.  Then loop continuously until the
1640 ! interrupt is detected, and the interrupt
1650 ! service routine acknowledges the
1660 ! interrupt and sets the flag to 1.
1670 Selfcal_done=0
1680 DISP "Waiting for SRQ on selfcal"
1690 LOOP
1700 EXIT IF Selfcal_done=1
1710 END LOOP
1720 !
1730 ! Display desired completion message.
1740 !DISP
1750 WAIT 10
1760 GOTO Start
1770 !
1780 Srq_handler: ! Interrupt Service Routine
1790 !
1800 ! Determine that the analyzer was actually
1810 ! the instrument that generated the
1820 ! interrupt.
1830 Stb=SPOLL(@Hp87xx)
1840 !
1850 ! Determine if the operation status register
1860 ! caused the interrupt by looking at bit 7
1870 ! of the result of the serial poll.
1880 DISP "Checking SRQ..."
1890 IF BINAND(Stb,128)<>0 THEN
1900 !
1910 ! Read the operational status event register.
1920 OUTPUT @Hp87xx;"STAT:OPER:EVEN?"
1930 ENTER @Hp87xx;Op_event
1940 !
1950 ! Determine if the correcting status register
1960 ! bit 7 is set.
1970 DISP "Got SRQ.  Selfcal complete!",Op_event
1980 BEEP
1990 IF BINAND(Op_event,128)<>0 THEN
2000 !
2010 ! If so, then set flag indicating
2020 ! selfcal done.
2030 Selfcal_done=1
2040 END IF
2050 END IF
2060 RETURN
2070 END
2080 !
2090 !************************************************************
2100 ! Iden_port:   Identify io port to use
```

```
2110 ! Description: This routines sets up the I/O port address for
2120 !              the SCPI interface.  For “HP 87xx” instruments,
2130 !              the address assigned to @Hp87xx = 800 otherwise,
2140 !              716.
2150 !**************************************************************
2160 SUB Iden_port
2170 COM /Sys_state/ @Hp87xx,Scode
2180 !
2190 IF POS(SYSTEM$(“SYSTEM ID”),”HP 87”)<>0 THEN
2200 ASSIGN @Hp87xx TO 800
2210 Scode=8
2220 ELSE
2230 ASSIGN @Hp87xx TO 716
2240 Scode=7
2250 END IF
2260 !
2270 SUBEND!Iden_port
2280 !
```

Transferring Programs

DOWNLOAD	This program demonstrates how to download an IBASIC program to the analyzer. It is designed (in HP BASIC or HP BASIC for Windows) to run on an external workstation or PC.
UPLOAD	This program uploads the IBASIC program in the analyzer's program buffer to an ASCII file on the external controller's current mass storage device.

DOWNLOAD Example Program

```
10    !-----------------------------------------------------
20    !
30    ! BASIC program:  DOWNLOAD - Download program to Rfna
40    !
50    ! This program demonstrates how to download an IBASIC
60    ! program to the 871x.  This program is designed to
70    ! run on an external controller.
80    !
90    !-----------------------------------------------------
100   !
110   ! Initialize variables for the interface select code
120   ! and the HP-IB address of the HP 8711.
130   !
140   Scode=7
150   Address=16
160   Na=Scode*100+Address
170   !
180   ! Initialize variables, abort any bus traffic and
190   ! clear the input/output queues of the analyzer.
200   !
210   DIM Line$[255]
220   ABORT Scode
230   CLEAR Na
240   !
250   ! Get the program’s filename and open the file.
260   !
270 Get_filename: INPUT “Program to be transferred?”,Filename$
280   ON ERROR GOTO No_file
290   DISP “Checking file . . .”
300   ASSIGN @Basic_prog TO Filename$;FORMAT ON
310   OFF ERROR
320   !
330   ! Clear the contents of the analyzer’s program buffer.
340   !
350   OUTPUT Na;”PROG:DEL:ALL”
360   !
370   ! Change the EOL (end of line) character to line feed
380   ! and initialize the line counter.
390   !
400 Transfer: ASSIGN @Prog TO Na;EOL CHR$(10)
410   Line_count=0
420   !
430   ! Initiate the program transfer (an indefinite length
440   ! block data transfer).
450   !
460   OUTPUT @Prog;”PROG:DEF #0”;
470   !
480   ! Read each program line from the file and send it to
490   ! the HP 8711.  Loop until the end of file is reached.
500   !
510   ON ERROR GOSUB End_file
520   LOOP
530     ENTER @Basic_prog;Line$
540     OUTPUT @Prog;Line$
550     Line_count=Line_count+1
560     DISP “Lines transferred: “;Line_count
570   END LOOP
```

```
580   !
590   ! End the data transfer (output a line feed with EOI)
600   ! and close the file.  Return the analyzer to LOCAL
610   ! control and stop this program.
620   !
630 End_block: OUTPUT @Prog;CHR$(10) END
640   ASSIGN @Basic_prog TO *
650   DISP “Transfer complete”
660   LOCAL Na
670   STOP
680   !
690   ! This subroutine is the error handler for opening
700   ! the file - if the file won’t open it returns to
710   ! get a new file name.
720   !
730 No_file: BEEP
740   DISP “CAN’T OPEN: “““;Filename$;””” -- “;
750   GOTO Get_filename
760   RETURN
770   !
780   ! This subroutine is the error handler for the
790   ! data transfer.  When the end of file is reached
800   ! it generates an error.  Execution is resumed
810   ! outside of the transfer loop.
820   !
830 End_file: IF ERRN=59 THEN GOTO End_block
840   DISP ERRM$;” occurred during data transfer”
850   STOP
860   RETURN
870   END
```

UPLOAD Example Program

```
10    !------------------------------------------------------
20    !
30    ! BASIC program: UPLOAD - Upload program from HP 871X
40    !
50    ! This program uploads the current IBASIC program
60    ! in the HP 871X’s program buffer to an ASCII file
70    ! on the controller’s current mass storage device.
80    !
90    !------------------------------------------------------
100   !
110   ! Assign an I/O path name to the HP 8711, initialize
120   ! the variables, and clear the analyzer’s input/output
130   ! queues.
140   !
150    ASSIGN @Rfna TO 716
160    DIM Prog_line$[256]
170    CLEAR @Rfna
180   !
190   ! Enter the name of the file to be created.
200   !
210    INPUT “ENTER NAME OF FILE TO UPLOAD PROGRAM TO “,Filename$
220    PRINT Filename$
230   !
240   ! Query the HP 8711 for the contents of its
250   ! program buffer.
260   !
270    OUTPUT @Rfna;”PROG:DEF?”
280   !
290   ! Read the block header, the number of digits in
300   ! the file size, and the file size.
310   !
320    ENTER @Rfna USING “#,A,D”;Prog_line$,Ndigits
330    ENTER @Rfna USING “#,”&VAL$(Ndigits)&”D”;Nbytes
340   !
350   ! Create the target ASCII file on the current mass
360   ! storage device and assign it an I/O path name.
370   !
380    Openfile(@File,Filename$,Nbytes)
390    ASSIGN @File TO Filename$;FORMAT ON
400   !
410   ! Read the program one line at a time, and write
420   ! it to the new file.  Print each line on the
430   ! display as it is read.
440   !
450    LOOP
460      ENTER @Rfna;Prog_line$
470    EXIT IF LEN(Prog_line$)=0
480      PRINT Prog_line$
490      OUTPUT @File;Prog_line$
500    END LOOP
510   !
520   ! Close the new file.
530   !
540    ASSIGN @File TO *
550    END
560   !------------------------------------------------------
570    SUB Openfile(@File,Filename$,Fsize)
```

```
580  !---------------------------------------------------
590  !
600  ! This subprogram creates an ASCII file with the
610  ! name ‘Filename$’ of the specified size ‘Fsize’.
620  ! Error trapping is used to detect any errors in
630  ! opening the file.  If the controller is HP IBASIC
640  ! for Windows a DOS file is created, otherwise the
650  ! LIF format is used.
660  !
670  !---------------------------------------------------
680     ON ERROR GOTO Openerr
690     IF SYSTEM$(“SYSTEM ID”)=”IBASIC/WINDOWS” THEN
700       CREATE Filename$,1
710     ELSE
720       IF Fsize MOD 256>0 THEN Fsize=Fsize+256
730       CREATE ASCII Filename$,Fsize DIV 256
740     END IF
750  !
760 Openerr:IF ERRN<>54 THEN PRINT ERRM$
770   SUBEND
```

Using Service Requests

SRQ	This program generates a service request interrupt. The example uses the status reporting structure to generate an interrupt as soon as averaging is complete.
SRQ_INT	Monitoring the status report of the analyzer.

SRQ Example Program

This program demonstrates generating a service request interrupt. The SRQ is used to indicate when averaging is complete. More information on service requests and the status registers is available in the *Programmer's Guide*, "Using Status Registers."

In this program, the `STATus:PRESet` executed in line `1250` has the effect of setting all bits in the averaging status transition registers (positive transitions to `0`, negative transitions to `1`). It also sets up the operational status transition registers (positive transitions to `1`, negative transitions to `0`). These are the states needed to generate an interrupt when averaging is complete.

```
1000 !Filename:  SRQ
1010 !
1020 ! Description:
1030 !   Set an SRQ to occur when averaging is complete.
1040 !   Turn on averaging, and set to 8 averages.
1050 !   Initiate sweeps.  SRQ will occur after 8 sweeps.
1060 !   Wait in a do-nothing loop, checking SRQ flag.
1070 !   Display message after SRQ flag is set.
1080 !
1090 !
1100 COM /Sys_state/ @Hp87xx,Scode
1110 ! Identify I/O Port
1120 CALL Iden_port
1130 !
1140 !
1150 ! Clear status registers.
1160 OUTPUT @Hp87xx;"*CLS"
1170 !
1180 ! Clear the Service Request Enable register.
1190 OUTPUT @Hp87xx;"*SRE 0"
1200 !
1210 ! Clear the Standard Event Status Enable register.
1220 OUTPUT @Hp87xx;"*ESE 0"
1230 !
1240 ! Preset the remaining status registers.
1250 OUTPUT @Hp87xx;"STAT:PRES"
1260 !
1270 ! Set operation status register to report
1280 ! to the status byte on POSITIVE transition of
1290 ! the averaging bit.
1300 OUTPUT @Hp87xx;"STAT:OPER:ENAB 256"
1310 !
1320 ! Set averaging status register to report to
1330 ! operational status register on NEGATIVE transition
1340 ! of the averaging done bits.  The NEGATIVE
1350 ! transition needs to be detected because the
1360 ! averaging bit 0 is set to 1 while the analyzer
1370 ! is sweeping on channel 1 and the number of
1380 ! sweeps completed since averaging restart is
1390 ! less than the averaging factor.  When the bit
1400 ! goes back to 0, the averaging is done.
```

```
1410 OUTPUT @Hp87xx;"STAT:OPER:AVER:ENAB 1"
1420 !
1430 ! Enable the operational status bit in the status
1440 ! byte to generate an SRQ.
1450 OUTPUT @Hp87xx;"*SRE 128"
1460 !
1470 ! On an interrupt from HP-IB "Scode" (Interface
1480 ! Select Code) SRQ bit (2), branch to the interrupt
1490 ! service routine "Srq_handler".
1500 ON INTR Scode,2 GOSUB Srq_handler
1510 !
1520 ! Now enable the interrupt on SRQ (Service Request).
1530 ENABLE INTR Scode;2
1540 !
1550 ! Set averaging factor to 8.
1560 OUTPUT @Hp87xx;"SENS1:AVER:COUN 8;*WAI"
1570 !
1580 ! Turn on averaging and restart.
1590 OUTPUT @Hp87xx;"SENS1:AVER ON;AVER:CLE;*WAI"
1600 !
1610 ! Turn on continuous sweep trigger mode.
1620 OUTPUT @Hp87xx;"ABOR;:INIT1:CONT ON;*WAI"
1630 !
1640 ! Initialize flag indicating when averaging done
1650 ! to 0.  Then loop continuously until the
1660 ! interrupt is detected, and the interrupt
1670 ! service routine acknowledges the
1680 ! interrupt and sets the flag to 1.
1690 Avg_done=0
1700 DISP "Waiting for SRQ on averaging complete.";
1710 LOOP
1720     DISP ".";
1730     WAIT .1! Slow down dots
1740 EXIT IF Avg_done=1
1750 END LOOP
1760 !
1770 ! Display desired completion message.
1780 DISP
1790 DISP "Got SRQ.  Averaging Complete!"
1800 STOP
1810 !
1820 Srq_handler: ! Interrupt Service Routine
1830 !
1840 ! Determine that the analyzer was actually
1850 ! the instrument that generated the
1860 ! interrupt.
1870 Stb=SPOLL(@Hp87xx)
1880 !
1890 ! Determine if the operation status register
1900 ! caused the interrupt by looking at bit 7
1910 ! of the result of the serial poll.
1920 IF BINAND(Stb,128)<>0 THEN
1930 !
1940 ! Read the operational status event register.
1950     OUTPUT @Hp87xx;"STAT:OPER:EVEN?"
1960     ENTER @Hp87xx;Op_event
1970 !
1980 ! Determine if the averaging status register
1990 ! bit 8 is set.
2000     IF BINAND(Op_event,256)<>0 THEN
```

```
2010 !
2020 ! If so, then set flag indicating
2030 ! averaging done.
2040         Avg_done=1
2050     END IF
2060 END IF
2070 RETURN
2080 END
2090 !
2100 !**************************************************************
2110 ! Iden_port:   Identify io port to use.
2120 ! Description: This routines sets up the I/O port address for
2130 !              the SCPI interface.  For "HP 87xx" instruments,
2140 !              the address assigned to @Hp87xx = 800 otherwise,
2150 !              716.
2160 !**************************************************************
2170 SUB Iden_port
2180     COM /Sys_state/ @Hp87xx,Scode
2190 !
2200     IF POS(SYSTEM$("SYSTEM ID"),"HP 87")<>0 THEN
2210         ASSIGN @Hp87xx TO 800
2220         Scode=8
2230     ELSE
2240         ASSIGN @Hp87xx TO 716
2250         Scode=7
2260     END IF
2270 !
2280 SUBEND !Iden_port
2290 !
```

SRQ_INT Example Program

This program demonstrates how to monitor the status report of the network analyzer via an interrupt handler. It monitors all the status or error bits of the status register. Whenever an error or an event occurs, the analyzer will interrupt this program. This program will then decode the error bits and display the appropriate messages. For a detailed status report register map, refer to Chapter 5, "Using Status Registers," in the *Programmer's Guide.*

```
1000 !  SRQ_INT
1010 !  BASIC program:
1020 !
1030 !***************************************************************
1040 !  Description:  This program implements the interrupt handler
1050 !  routine for SRQ status reports.  It tries to decode all the
1060 !  possible error bits by rippling it through the registers
1070 !  and print out the appropriate messages for the status report.
1080 !  Refer to the status model diagram for registers mapping.
1090 !
1100 !  Note:  To setup additional states other than the status
1110 !         model, add code to the subroutine "Setup_states".
1120 !***************************************************************
1130 !
1140 !************ Main program **************
1150 !
1160 !  Make @Hp87xx common to all subroutines
1170 COM /Sys_state/ @Hp87xx,Scode
1180 ! Identify the computer we are running on
1190 ! and assign the i/o port address to @Hp87xx
1200 CALL Iden_port
1210 !
1220 ! Setup all required SRQ registers
1230 CALL Setup_srq_regs
1240 !
1250 ! This is required if user wants to detect either
1260 ! "Any Ext. Keybd. Pressed" or "Front Panel Knob Turned"
1270 ! of the Device Status Register.
1280 OUTPUT @Hp87xx;"SYST:KEY:QUE:STATE ON"
1290 !
1300 ! Go to subroutine to setup any neccessary states
1310 CALL Setup_states
1320 !
1330 Report_count=0
1340 ! Forever loop to wait for any failed events to happened
1350 DISP "Waiting for any Failed Events.........Report Count =
     ";Report_count
1360 Do_loop: !
1370 GOSUB Set_userbit                             ! Read and display
     variables
1380 !
1390 GOTO Do_loop
1400 STOP
1410 !
1420 !************ Subroutine Blocks ********************
1430 !
```

```
1440 !**************************************************************
1450 ! Set_userbit:  Setup user bit.
1460 ! Description:  This subroutine waits for an SRQ interrupt to
1470 ! signal that a sweep has finished.  It then clears the HP-IB
1480 ! registers by reading them.  Once that is done, the user bit
1490 ! is toggled.
1500 !**************************************************************
1510 !
1520 Set_userbit:!
1530 !
1540 ON INTR Scode GOTO Read_results       ! Set up interrupt branching
1550 ENABLE INTR Scode;2                   ! Allow interrupt on SRQ
1560 Suspend: !WAIT 5                      ! Use WAIT 'n' to suspend IBASIC
1570 GOTO Suspend
1580 !
1590 ! Interrupt Service Routine
1600 Read_results:                         ! Program has finished
1610 A=SPOLL(@Hp87xx)                      !  and clear the SRQ
1620 !
1630 ! This CLEAR command is for clearing out the bus just in case
1640 ! this is a Query error.  Without this CLEAR command, the previous
1650 ! Query would screw up the state of the instrument and the next
1660 ! Query will get an error.
1670 CLEAR @Hp87xx
1680 OUTPUT @Hp87xx;"*STB?"
1690 ENTER @Hp87xx;Stbr
1700 WHILE Stbr<>0
1710     CALL Decode_srq(Stbr)
1720     Stb=Stbr
1730     OUTPUT @Hp87xx;"*STB?"
1740     ENTER @Hp87xx;Stbr
1750 END WHILE
1760 OUTPUT @Hp87xx;"*CLS"   ! Clear status byte
1770 BEEP
1780 Report_count=Report_count+1
1790 DISP "Waiting for any Failed Events.........Report Count =
     ";Report_count
1800 !
1810 RETURN
1820 END
1830 !
1840 !***********   Status Report Decode Block ********************
1850 !
1860 !
1870 !******************************************************************
1880 ! Decode_srq:  Decode status byte.
1890 ! Description: Decode the Srq register and ripple through the
1900 !              registers and decode the necessary failed registers.
1910 !              The decoding is done with the Event registers.  The
1920 !              corresponding Condition Registers are only read and
1930 !              display.  The numbers are display in hex numbers
1940 !              with a leading "0x".
1950 !              If any Event has failed, a message will be display
1960 !              with the "Meas" Channel number and "Segment" number.
1970 !******************************************************************
1980 SUB Decode_srq(Reg)
1990     COM /Sys_state/ @Hp87xx,Scode
2000     DIM Reg_name$(1:8)[40]
2010 !
2020 !
```

```
2030 !   Print out the Date of Time of this report
2040 !
2050     PRINT
2060     PRINT
2070     PRINT "Status Report:  ";DATE$(TIMEDATE);"  ";TIME$(TIMEDATE)
2080     PRINT
2090     PRINT "Status Byte = 0x";IVAL$(Reg,16)
2100     IF (BIT(Reg,6))=1 THEN
2110         RESTORE
2120         READ Reg_name$(*)
2130 !
2140 !       For each of the bit set in the status register,
2150 !       call Decode_reg() to decode the appropriate
2160 !       second level registers.
2170 !
2180         FOR I=2 TO 7
2190 !
2200             IF (BIT(Reg,I)=1) THEN
2210                 CALL Decode_reg(Reg_name$(I+1))
2220             END IF
2230 !
2240         NEXT I
2250 !
2260     ELSE
2270         PRINT "Bogus Interrupt??? Bit 6 of Status byte is not set???"
2280     END IF
2290 !
2300     PRINT
2310     PRINT ".......END OF REPORT"
2320     PRINT
2330 !
2340 !
2350 !   Status Register
2360     DATA  "Unknown Register"        ! Bit 0
2370     DATA  "Unknown Register"        ! Bit 1
2380     DATA  "Device Status"           ! Bit 2
2390     DATA  "Questionable"            ! Bit 3
2400     DATA  "Output Queue"            ! Bit 4
2410     DATA  "Standard Event"          ! Bit 5
2420     DATA  "Status Fail"             ! Bit 6
2430     DATA  "Operational"             ! Bit 7
2440 !
2450 SUBEND !DECODE_SRQ
2460 !
2470 !******************************************************************
2480 ! Decode_reg:  Decode the second level registers.
2490 ! Description: The Cases in the SELECT statements corresponds
2500 !              to the bits supported by the instrument.  Refer
2510 !              to the menu for the bit positions of these bits.
2520 !              For each failed event, the Event and Condition
2530 !              registers are read and Display.  The Event register
2540 !              is further used to decode the third level registers.
2550 !
2560 !******************************************************************
2570 !
2580 SUB Decode_reg(Reg_name$)
2590     COM /Sys_state/ @Hp87xx,Scode
2600 !
2610     SELECT Reg_name$
2620 !
```

```
2630 !   Device Status register
2640     CASE "Device Status"! Bit 2
2650         OUTPUT @Hp87xx;"STAT:DEV:EVEN?"
2660         ENTER @Hp87xx;Dev_event
2670         PRINT "   Device Status Event Reg = 0x";IVAL$(Dev_event,16)
2680         OUTPUT @Hp87xx;"STAT:DEV:COND?"
2690         ENTER @Hp87xx;Dev_cond
2700         PRINT "   Device Status Condition Reg = 0x";IVAL$(Dev_cond,16)
2710         CALL Decode_dev(Dev_event)
2720 !
2730 !   Questionable status register
2740     CASE "Questionable"! Bit 3
2750 !
2760         OUTPUT @Hp87xx;"STAT:QUES:EVEN?"! Read and clear Questional
             STATUS reg.
2770         ENTER @Hp87xx;Ques_event
2780         PRINT "   Questionable Event Reg = 0x";IVAL$(Ques_event,16)
2790         OUTPUT @Hp87xx;"STAT:QUES:COND?"
2800         ENTER @Hp87xx;Ques_cond
2810         PRINT "   Questionable Condition Reg = 0x";IVAL$(Ques_cond,16)
2820         CALL Decode_ques(Ques_event)
2830 !
2840 !
2850     CASE "Standard Event"! Bit 4
2860 !
2870         OUTPUT @Hp87xx;"*ESR?"
2880         ENTER @Hp87xx;Stand_event
2890         PRINT "   Standard Event Reg = 0x";IVAL$(Stand_event,16)
2900         CALL Decode_esr(Stand_event)
2910 !
2920     CASE "Output Queue"! Bit 5
2930 !
2940         PRINT "   Message Available"
2950 !
2960 !   Latch bit of Status Byte register
2970     CASE "Status Fail"! Bit 6
2980 !     Do Nothing
2990 !
3000 !   Operational Status register
3010     CASE "Operational"! Bit 7
3020 !
3030         OUTPUT @Hp87xx;"STAT:OPER:EVEN?"!Read and clear Operational
             STATUS reg.
3040         ENTER @Hp87xx;Oper_event
3050         PRINT "   Operational Event Reg = 0x";IVAL$(Oper_event,16)
3060         OUTPUT @Hp87xx;"STAT:OPER:COND?"
3070         ENTER @Hp87xx;Oper_cond
3080         PRINT "   Operational Condition Reg = 0x";IVAL$(Oper_cond,16)
3090 !
3100         CALL Decode_oper(Oper_event)
3110 !
3120 !
3130     CASE ELSE
3140         PRINT "   Unsupported Bit set in Status Byte or ";
3150         PRINT "   Bogus interrupt. "
3160 !
3170     END SELECT
3180 SUBEND !DECODE_REG
3190 !
3200 !
```

```
3210 !*************************************************************
3220 ! Decode_Ques:  Decode Questionable Fail register.
3230 ! Description:  Decode Questionable Fail register and Print out
3240 !              appropriate messages.
3250 !*************************************************************
3260 SUB Decode_ques(Reg)
3270     COM /Sys_state/ @Hp87xx,Scode
3280     DIM Message$(0:15,0:1)[120]
3290     DIM Segment_event(4:7)
3300 !
3310     READ Message$(*)
3320     FOR I=0 TO 15
3330         IF Message$(I,0)="Enable" THEN
3340             IF BIT(Reg,I)=1 THEN
3350 !
3360                 PRINT Message$(I,1)
3370                 IF (I=9) THEN !  Check Limit Fail Register
3380                     OUTPUT @Hp87xx;"STAT:QUES:LIM:EVEN?"
3390                     ENTER @Hp87xx;Lim_event
3400                     PRINT "      Limit Fail Event Reg =
                         0x";IVAL$(Lim_event,16)
3410                     OUTPUT @Hp87xx;"STAT:QUES:LIM:COND?"
3420                     ENTER @Hp87xx;Lim_cond
3430                     PRINT "      Limit Fail Condition Reg =
                         0x";IVAL$(Lim_cond,16)
3440                     CALL Decode_lim(Lim_event)
3450                 END IF
3460 !
3470                 IF (I=4) OR (I=5) OR (I=6) OR (I=7) THEN
3480                     OUTPUT @Hp87xx;"STAT:QUES:SEGM";VAL$(I-3);":EVEN?"
3490                     ENTER @Hp87xx;Segment_event(I)
3500                     PRINT "      Segment ";VAL$(I-3);" Event Reg =
                         0x";IVAL$(Segment_event(I),16)
3510                     OUTPUT @Hp87xx;"STAT:QUES:SEGM";VAL$(I-3);":COND?"
3520                     ENTER @Hp87xx;Segment_cond
3530                     PRINT "      Segment ";VAL$(I-3);" Condition Reg =
                         0x";IVAL$(Segment_cond,16)
3540                     CALL Decode_seg(Segment_event(I),I-3)
3550                 END IF
3560             END IF
3570         END IF
3580     NEXT I
3590 !
3600 !  This array has two fields:
3610 !  First Field - If Enable, Display the next string message
3620 !                Else, ignore the next string message.
3630 !  Second Field - String message for the corresponding bits of
3640 !                 the Questionable register.
3650     DATA "Enable","      ALC UNLEVELED..............."       ! Bit 0
3660     DATA "Enable","      FREQ_ERROR.................."       ! Bit 1
3670     DATA "Disable","      Bit 2 Unsupported"                 ! Bit 2
3680     DATA "Disable","      Bit 3 Unsupported"                 ! Bit 3
3690     DATA "Enable","      Segment 1 Limit Fail................"
         ! Bit 4
3700     DATA "Enable","      Segment 2 Limit Fail................"
         ! Bit 5
3710     DATA "Enable","      Segment 3 Limit Fail................"
         ! Bit 6
3720     DATA "Enable","      Segment 4 Limit Fail................"
         ! Bit 7
```

```
3730     DATA "Disable","      Bit 8 Unsupported"              ! Bit 8
3740     DATA "Enable","      Limit Fail................."      ! Bit 9
3750     DATA "Enable","      Stale Data(Data?).........."      ! Bit 10
3760     DATA "Disable","      Bit 11 Unsupported"             ! Bit 11
3770     DATA "Disable","      Bit 12 Unsupported"             ! Bit 12
3780     DATA "Disable","      Bit 13 Unsupported"             ! Bit 13
3790     DATA "Disable","      Bit 15 Unsupported"             ! Bit 14
3800     DATA "Disable","      Bit 15 Unsupported"             ! Bit 15
3810 SUBEND !Decode_Ques
3820 !
3830 !
3840 !
3850 !**************************************************************
3860 ! Decode_lim:  Decode Limit Fail register.
3870 ! Description: Decode Limit Fail register and Print out the
3880 !              appropriate messages.
3890 !**************************************************************
3900 !
3910 SUB Decode_lim(Reg)
3920     COM /Sys_state/ @Hp87xx,Scode
3930     DIM Message$(0:3)[120]
3940 !
3950     READ Message$(*)
3960 !
3970     FOR I=0 TO 3
3980         IF BIT(Reg,I)=1 THEN
3990             PRINT Message$(I)
4000         END IF
4010     NEXT I
4020 !
4030 !   Displaying message
4040     DATA "         Limit Line Failed   on  Meas 1"        ! Bit 0
4050     DATA "         Limit Line Failed   on  Meas 2"        ! Bit 1
4060     DATA "         Marker Limit Failed on  Meas 1"        ! Bit 2
4070     DATA "         Marker Limit Failed on  Meas 2"        ! Bit 3
4080 SUBEND !Decode_lim
4090 !
4100 !
4110 !**************************************************************
4120 ! Decode_seg:  Decode Segment status registers.
4130 ! Description: Decode Segment status registers and Print out the
4140 !              appropriate messages.
4150 !**************************************************************
4160 !
4170 SUB Decode_seg(Reg,Segment)
4180     COM /Sys_state/ @Hp87xx,Scode
4190     DIM Message$(0:9)[120]
4200 !
4210     READ Message$(*)
4220 !
4230 !   Check to see if bias regs need to be decoded.
4240     FOR I=0 TO 9
4250         IF BIT(Reg,I)=1 THEN
4260             PRINT Message$(I);Segment
4270             IF I=4 THEN ! Meas 1 has failed?
4280                 CALL Decode_bias("Meas 1",Segment)
4290             END IF
4300             IF I=5 THEN ! Meas 2 has failed?
4310                 CALL Decode_bias("Meas 2",Segment)
4320             END IF
```

```
4330         END IF
4340     NEXT I
4350 !
4360 !   Displaying message
4370     DATA "         Limit Line Failed      on Meas 1, Segment"        !
     Bit 0
4380     DATA "         Limit Line Failed      on Meas 2, Segment"        !
     Bit 1
4390     DATA "         Marker Limit Failed    on Meas 1, Segment"        !
     Bit 2
4400     DATA "         Marker Limit Failed    on Meas 2, Segment"        !
     Bit 3
4410     DATA "         Bias Limit Failed      on Meas 1, Segment"        !
     Bit 4
4420     DATA "         Bias Limit Failed      on Meas 2, Segment"        !
     Bit 5
4430     DATA "         Gain Bar Limit Failed  on Meas 1, Segment"        !
     Bit 6
4440     DATA "         Gain Bar Limit Failed  on Meas 2, Segment"        !
     Bit 7
4450     DATA "         IIc Ack Test Failed    on Meas 1, Segment"        !
     Bit 8
4460     DATA "         IIc Ack Test Failed    on Meas 2, Segment"        !
     Bit 9
4470 SUBEND !Decode_seg
4480 !
4490 !
4500 !**************************************************************
4510 ! Decode_dev:  Decode Device status register.
4520 ! Description: Decode Device status register and Print out the
4530 !              appropriate messages.
4540 !**************************************************************
4550 !
4560 SUB Decode_dev(Reg)
4570     COM /Sys_state/ @Hp87xx,Scode
4580     DIM Message$(0:3)[120]
4590 !
4600     READ Message$(*)
4610     FOR I=0 TO 3
4620         IF BIT(Reg,I)=1 THEN
4630             PRINT Message$(I)
4640         END IF
4650     NEXT I
4660 !
4670     DATA "      Any Key Pressed"
4680     DATA "      Any Softkey Pressed"
4690     DATA "      Any Ext. Keybd. Pressed"
4700     DATA "      Front Panel Knob Turned"
4710 SUBEND !Decode_dev_
4720 !
4730 !
4740 !
4750 !
4760 !**************************************************************
4770 ! Decode_esr:  Decode Standard Event register
4780 ! Description: Decode Standard Event register and Print out the
4790 !              appropriate messages
4800 !**************************************************************
4810 !
4820 SUB Decode_esr(Reg)
```

```
4830     COM /Sys_state/ @Hp87xx,Scode
4840     DIM Message$(0:7)[120]
4850 !
4860     READ Message$(*)
4870     FOR I=0 TO 7
4880         IF BIT(Reg,I)=1 THEN
4890             PRINT Message$(I)
4900             IF I=4 THEN
4910                 CALL Disp_err
4920             END IF
4930         END IF
4940     NEXT I
4950 !
4960     DATA "      Operation Complete"                ! Bit 0
4970     DATA "      Request Control"                   ! Bit 1
4980     DATA "      Qeury Error"                       ! Bit 2
4990     DATA "      Device-Dependent Error"            ! Bit 3
5000     DATA "      Execution Error"                   ! Bit 4
5010     DATA "      Command Error"                     ! Bit 5
5020     DATA "      User Request"                      ! Bit 6
5030     DATA "      Power On"                          ! Bit 7
5040 SUBEND !Decode_esr
5050 !
5060 !
5070 !**************************************************************
5080 ! Decode_oper:  Decode Operational register.
5090 ! Description:  Decode Operational register and Print out the
5100 !               appropriate messages
5110 !**************************************************************
5120 !
5130 SUB Decode_oper(Reg)
5140     COM /Sys_state/ @Hp87xx,Scode
5150     DIM Message$(0:15,0:1)[120]
5160 !
5170     READ Message$(*)
5180     FOR I=0 TO 15
5190         IF Message$(I,0)="Enable" THEN
5200             IF BIT(Reg,I)=1 THEN
5210 !
5220                 PRINT Message$(I,1)
5230                 IF I=4 THEN
5240 !
5250                     OUTPUT @Hp87xx;"STAT:OPER:MEAS:EVEN?"
5260                     ENTER @Hp87xx;Meas_event
5270                     PRINT "Measuring Event Reg                =
                         0x";IVAL$(Meas_event,16)
5280                     OUTPUT @Hp87xx;"STAT:OPER:MEAS:COND?"
5290                     ENTER @Hp87xx;Meas_cond
5300                     PRINT "Measuring Condition Reg            =
                         0x";IVAL$(Meas_cond,16)
5310 !
5320                     CALL Decode_meas(Meas_event)
5330                 ELSE
5340                     IF I=8 THEN
5350 !
5360                         OUTPUT @Hp87xx;"STAT:OPER:AVER:EVEN?"
5370                         ENTER @Hp87xx;Aver_event
5380                         PRINT "Averaging Event Reg                =
                             0x";IVAL$(Aver_event,16)
5390                         OUTPUT @Hp87xx;"STAT:OPER:AVER:COND?"
```

```
5400                         ENTER @Hp87xx;Aver_cond
5410                         PRINT "Averaging Condition Reg            =
                             0x";IVAL$(Aver_cond,16)
5420 !
5430                         CALL Decode_avg(Aver_event)
5440                     END IF
5450                 END IF
5460 !
5470             END IF
5480         END IF
5490     NEXT I
5500 !
5510 !  This array has two fields:
5520 !  First Field - If Enable, Display the next string message
5530 !                Else, ignore the next string message.
5540 !  Second Field - String message for the corresponding bits of
5550 !                 the Questionable register.
5560     DATA "Enable","      Calibrating..............."        ! Bit 0
5570     DATA "Enable","      Settling.................."        ! Bit 1
5580     DATA "Disable","      Bit 2 Unsupported"                ! Bit 2
5590     DATA "Disable","      Bit 3 Unsupported"                ! Bit 3
5600     DATA "Enable","      Measuring................."        ! Bit 4
5610     DATA "Disable","      Bit 5 Unsupported"                ! Bit 5
5620     DATA "Disable","      Bit 6 Unsupported"                ! Bit 6
5630     DATA "Enable","      Correcting................"        ! Bit 7
5640     DATA "Enable","      Averaging................."        ! Bit 8
5650     DATA "Enable","      Hardcopy In Progress......"        ! Bit 9
5660     DATA "Enable","      Service Test In Progress.."        ! Bit 10
5670     DATA "Disable","      Bit 11 Unsupported"               ! Bit 11
5680     DATA "Disable","      Bit 12 Unsupported"               ! Bit 12
5690     DATA "Disable","      Bit 13 Unsupported"               ! Bit 13
5700     DATA "Enable","      Program Running..........."        ! Bit 14
5710     DATA "Disable","      Bit 15 Unsupported"               ! Bit 15
5720 SUBEND !Decode_oper
5730 !
5740 !
5750 !**************************************************************
5760 ! Decode_meas:  Decode Measuring register.
5770 ! Description:  Decode Measuring register and Print out the
5780 !               appropriate messages.
5790 !**************************************************************
5800 !
5810 SUB Decode_meas(Reg)
5820     COM /Sys_state/ @Hp87xx,Scode
5830     DIM Message$(0:1)[120]
5840 !
5850     READ Message$(*)
5860     FOR I=0 TO 1
5870         IF BIT(Reg,I)=1 THEN
5880             PRINT Message$(I)
5890         END IF
5900     NEXT I
5910 !
5920 !   Displaying message
5930     DATA "        Meas 1 Measuring........."
5940     DATA "        Meas 2 Measuring........."
5950 SUBEND !Decode_meas
5960 !
5970 !
5980 !**************************************************************
```

```
5990 ! Decode_avg:  Decode Averaging registe.
6000 ! Description: Decode Averaging register and Print out the
6010 !              appropriate messages.
6020 !*************************************************************
6030 !
6040 SUB Decode_avg(Reg)
6050     COM /Sys_state/ @Hp87xx,Scode
6060     DIM Message$(0:1)[120]
6070 !
6080     READ Message$(*)
6090     FOR I=0 TO 1
6100         IF BIT(Reg,I)=1 THEN
6110             PRINT Message$(I)
6120         END IF
6130     NEXT I
6140 !
6150 !   Displaying message
6160     DATA "         Meas 1 Averaging........."
6170     DATA "         Meas 2 Averaging........."
6180 SUBEND !Decode_avg
6190 !
6200 !
6210 !*************************************************************
6220 ! Decode_bias:  Decode Bias register.
6230 ! Description:  Decode Bias register and Print out the
6240 !               appropriate messages.
6250 !*************************************************************
6260 !
6270 SUB Decode_bias(Meas$,Segment)
6280     COM /Sys_state/ @Hp87xx,Scode
6290     DIM Message$(0:11)[120]
6300 !
6310     READ Message$(*)
6320     SELECT Meas$
6330     CASE "Meas 1"
6340         Chan=1
6350     CASE "Meas 2"
6360         Chan=2
6370     END SELECT
6380     OUTPUT
         @Hp87xx;"CALC";VAL$(Chan);":BIAS:LIM:SEGM";VAL$(Segment);":COND?"
6390     ENTER @Hp87xx;Node
6400     FOR I=0 TO 11
6410         IF BIT(Node,I)=1 THEN
6420             PRINT Message$(I);" of ";Meas$;", Segment ";VAL$(Segment)
6430         END IF
6440     NEXT I
6450 !
6460 !   Displaying message
6470     DATA "            Current Limit Failed on  Bias 1"
6480     DATA "            Current Limit Failed on  Bias 2"
6490     DATA "            Current Limit Failed on  Bias 3"
6500     DATA "            Current Limit Failed on  Bias 4"
6510     DATA "            Current Limit Failed on  Bias 5"
6520     DATA "            Current Limit Failed on  Bias 6"
6530     DATA "            Current Limit Failed on  Bias 7"
6540     DATA "            Current Limit Failed on  Bias 8"
6550     DATA "            Current Limit Failed on  VTune "
6560     DATA "            Current Limit Failed on  VTotal"
6570     DATA "            Voltage Limit Failed on  VAux1 "
```

```
6580     DATA "            Voltage Limit Failed on  VAux2 "
6590 SUBEND !Decode_bias
6600 !
6610 !
6620 SUB Disp_err
6630 !-
6640 ! SHOW ERROR, DUMP OUT SCPI ERROR QUEUE
6650 !-
6660     COM /Sys_state/ @Hp87xx,Scode
6670 !
6680     DIM Errmsg$[400]
6690     INTEGER Errnum
6700     LOOP
6710         OUTPUT @Hp87xx;"SYST:ERR?"
6720         ENTER @Hp87xx;Errnum,Errmsg$
6730     EXIT IF Errnum=0
6740         PRINT "         ";Errnum;Errmsg$
6750     END LOOP
6760 SUBEND ! Disp_err
6770 !
6780 !
6790 !***************************************************************
6800 ! Iden_port:  Identify io port to use.
6810 !***************************************************************
6820 SUB Iden_port
6830     COM /Sys_state/ @Hp87xx,Scode
6840 !
6850     IF POS(SYSTEM$("SYSTEM ID"),"HP 87")<>0 THEN
6860         ASSIGN @Hp87xx TO 800
6870         Scode=8
6880     ELSE
6890         ASSIGN @Hp87xx TO 716
6900         Scode=7
6910     END IF
6920 !
6930 SUBEND ! Iden_port
6940 !
6950 !***************************************************************
6960 ! Setup_states:
6970 ! Note:  Insert any setup routines or statements in here.....
6980 !        This routine is execute before interrupt is enabled.
6990 !***************************************************************
7000 !
7010 SUB Setup_states
7020 !
7030 !
7040 SUBEND !Setup_states
7050 !
7060 !
7070 !***************************************************************
7080 ! Setup_srq_regs:  Set up SRQ interrupt registers.
7090 !***************************************************************
7100 SUB Setup_srq_regs
7110     COM /Sys_state/ @Hp87xx,Scode
7120 !
7130 ! Initialize interrupt registers
7140 !
7150     OUTPUT @Hp87xx;"*CLS"                 ! Clear the STATUS BYTE
         register
7160     OUTPUT @Hp87xx;"*SRE 0"               ! Clear the service request
```

```
         enable register
7170     OUTPUT @Hp87xx;"*ESE 0"              ! Clear the std event enable
         register
7180     OUTPUT @Hp87xx;"STAT:PRES"           ! Clears all other registers
7190 !
7200 ! Set up registers for interrupt on Measuring going false
7210 !
7220     OUTPUT @Hp87xx;"STAT:QUES:PTR #H06F3" ! Positive Transition
7230     OUTPUT @Hp87xx;"STAT:QUES:ENAB #H06F3"! Enable Questionable
         Register
7240     OUTPUT @Hp87xx;"STAT:OPER:PTR #HFFFF" ! Positive Transition
7250     OUTPUT @Hp87xx;"STAT:OPER:ENAB #H0600"! Enable Operational Register
7260     OUTPUT @Hp87xx;"STAT:QUES:SEGM1:PTR #H03FF"! Positive Transition
7270     OUTPUT @Hp87xx;"STAT:QUES:SEGM1:ENAB #H03FF"! Enable Segment 1
         Register
7280     OUTPUT @Hp87xx;"STAT:QUES:SEGM2:PTR #H03FF"! Positive Transition
7290     OUTPUT @Hp87xx;"STAT:QUES:SEGM2:ENAB #H03FF"! Enable Segment 2
         Register
7300     OUTPUT @Hp87xx;"STAT:QUES:SEGM3:PTR #H03FF"! Positive Transition
7310     OUTPUT @Hp87xx;"STAT:QUES:SEGM3:ENAB #H03FF"! Enable Segment 3
         Register
7320     OUTPUT @Hp87xx;"STAT:QUES:SEGM4:PTR #H03FF"! Positive Transition
7330     OUTPUT @Hp87xx;"STAT:QUES:SEGM4:ENAB #H03FF"! Enable Segment 4
         Register
7340     OUTPUT @Hp87xx;"STAT:DEV:PTR #HFF"
7350     OUTPUT @Hp87xx;"STAT:DEV:ENAB #HFF"
7360 !
7370 !
7380     OUTPUT @Hp87xx;"*CLS"                ! Clear the STATUS BYTE
         register
7390     OUTPUT @Hp87xx;"*SRE #HFF"           ! Allow SRQ on all Registers
7400     OUTPUT @Hp87xx;"*ESE #HFF"           ! Enable standard event
         registers
7410     OUTPUT @Hp87xx;"*PSC #HFFFF"
7420 !
7430 SUBEND ! Setup_srq_regs
```

Making SRL Measurements (Option 100 Only)

MEAS_SRL	This programs shows the effects of various connector modeling parameters on an SRL measurement.
SRL_SRQ	This program initiates an SRL cable scan and sets up the analyzer to send an SRQ interrupt when the scan is completed.

MEAS_SRL Example Program

```
1000 ! Filename: MEAS_SRL (option 100 only)
1010 !
1020 ! This program is designed to show the effects of the various
1030 ! connector modeling on an SRL measurement.
1040 !
1050 ! For this measurement:  Users can change the following
1060 ! parameters.  Each parameter can be adjusted either
1070 ! manually or can be determined automatically by the
1080 ! analyzer.
1090 !
1100 ! To measure SRL of a cable, connect a long cable terminated
1110 ! with a load standard.  (50 or 75 ohm).
1120 ! The program steps through various settings.
1130 !
1140 ! Cable Z           -  Cable impedance
1150 ! Cable Zstop       -  The max freq in which Z average is measrued
1160 ! Connector C       -  Connector Capacitance
1170 ! Connector Length  -  Connector Length
1180 !
1190 ! After several values have been tried, the command is sent to
1200 ! measure the connector and automatically determine the optimum
1210 ! connector model values.
1220 !
1230 !
1240 COM /Sys_state/ @Hp87xx,Scode
1250 ! Identify I/O Port
1260 CALL Iden_port
1270 !
1280 OUTPUT @Hp87xx;"SYST:PRES; *OPC?"
1290 ENTER @Hp87xx;Opc
1300 !
1310 ! Select the SRL measurement on channel 1
1320 OUTPUT @Hp87xx;"SENS1:STAT ON; *WAI"
1330 OUTPUT @Hp87xx;"SENS1:FUNC 'SRL 1,0';DET NBAN; *WAI"
1340 !
1350 ! Sweep Hold mode
1360 OUTPUT @Hp87xx;"ABOR;:INIT1:CONT OFF;*OPC?"
1370 ENTER @Hp87xx;Opc
1380 !
1390 ! Take a sweep
1400 OUTPUT @Hp87xx;"ABOR;:INIT1:CONT OFF;:INIT1;*WAI"
1410 BEEP
1420 !
1430 Clear_disp
1440 Disp_mess("SRL connector model test...")
1450 WAIT 5
1460 !
1470 CALL Meas_srl(0.,0,0.,2.10E+8)
1480 ! Change srl parameters and re-measure
1490 Clear_disp
1500 Disp_mess("Setting default settings...")
1510 CALL Meas_srl(0.,0,0.,2.10E+8)
1520 WAIT 4
1530 !
1540 Clear_disp
1550 Disp_mess("Setting C = -1 pF...")
1560 CALL Meas_srl(0.,-1.E-12,0.,2.10E+8)
```

```
1570 WAIT 4
1580 !
1590 Clear_disp
1600 Disp_mess("Setting L = 50 mm...")
1610 CALL Meas_srl(0.,-1.E-12,.050,2.10E+8)
1620 WAIT 4
1630 !
1640 Clear_disp
1650 Disp_mess("Setting manual Z to 76 Ohm...")
1660 CALL Meas_srl(76.,-1.E-12,.050,2.10E+8)
1670 WAIT 4
1680 !
1690 Clear_disp
1700 Disp_mess("Auto Z with z_cutoff = 1E9 Hz")
1710 CALL Meas_srl(0.,-1.E-12,.050,1.E+9)
1720 WAIT 4
1730 !
1740 Clear_disp
1750 Disp_mess("Optimize connector model...")
1760 CALL Meas_srl(-1.,-1.E-12,0.,2.1E+6)
1770 WAIT 4
1780 !
1790 Clear_disp
1800 BEEP
1810 END
1820 !
1830 SUB Meas_srl(REAL Z,REAL Cap,REAL Length,REAL Zstop)
1840     COM /Sys_state/ @Hp87xx,Scode
1850     DIM Msg$[80]
1860     IF Z>=0 THEN
1870         WAIT 1
1880         OUTPUT @Hp87xx;"SENS1:CORR:LENG:CONN "&VAL$(Length)
1890         OUTPUT @Hp87xx;"SENS1:CORR:CAP:CONN "&VAL$(Cap)
1900         OUTPUT @Hp87xx;"SENS:FREQ:ZST "&VAL$(Zstop)
1910         IF Z>1. THEN
1920 ! Set manual impedance mode
1930             OUTPUT @Hp87xx;"SENS1:FUNC:SRL:MODE MANUAL"
1940             OUTPUT @Hp87xx;"SENS1:FUNC:SRL:IMP "&VAL$(Z)
1950         ELSE
1960 ! Automatically measure the impedance
1970             OUTPUT @Hp87xx;"SENS1:FUNC:SRL:MODE AUTO"
1980         END IF
1990 ! Take a sweep
2000         OUTPUT @Hp87xx;"ABOR;:INIT1:CONT OFF;:INIT1;*OPC?"
2010         ENTER @Hp87xx;Opc
2020     ELSE
2030 ! Automatically determine the srl connector model
2040         OUTPUT @Hp87xx;"SENS1:CORR:MODEL:CONN;*OPC?"
2050         ENTER @Hp87xx;Opc
2060     END IF
2070     BEEP
2080     OUTPUT @Hp87xx;"SENS1:CORR:LENG:CONN?"
2090     ENTER @Hp87xx;L
2100     OUTPUT @Hp87xx;"SENS1:CORR:CAP:CONN?"
2110     ENTER @Hp87xx;C
2120 ! Read the impedance.
2130 ! In AUTOMATIC_Z mode, the returned impedance is the measured Z.
2140 ! In MANUAL Z mode, the returned impedance is the manually entered Z.
2150     OUTPUT @Hp87xx;"SENS1:FUNC:SRL:IMP?"
2160     ENTER @Hp87xx;Zmeas
```

```
2170     Zmeas=DROUND(Zmeas,3)
2180     Cnew=DROUND(C,3)
2190     Lnew=DROUND(L,3)
2200     Msg$="C="&VAL$(Cnew)&" F,  L="&VAL$(Lnew)&" m, Z="&VAL$(Zmeas)&"
         Ohm"
2210     Clear_disp
2220     Disp_mess(Msg$)
2230 SUBEND
2240 !
2250 SUB Disp_mess(Message$)
2260     COM /Sys_state/ @Hp87xx,Scode
2270     OUTPUT @Hp87xx;"DISP:ANN:MESS:DATA '"&Message$&"'"
2280 SUBEND
2290 !
2300 SUB Clear_disp
2310     COM /Sys_state/ @Hp87xx,Scode
2320     DIM Command$[40]
2330     OUTPUT @Hp87xx;"DISP:ANN:MESS:CLE"
2340 SUBEND
2350 !
2360 !***************************************************************
2370 ! Iden_port:   Identify io port to use.
2380 ! Description: This routines sets up the I/O port address for
2390 !              the SCPI interface.  For "HP 87xx" instruments,
2400 !              the address assigned to @Hp87xx = 800 otherwise,
2410 !              716.
2420 !***************************************************************
2430 SUB Iden_port
2440     COM /Sys_state/ @Hp87xx,Scode
2450 !
2460     IF POS(SYSTEM$("SYSTEM ID"),"HP 87")<>0 THEN
2470         ASSIGN @Hp87xx TO 800
2480         Scode=8
2490     ELSE
2500         ASSIGN @Hp87xx TO 716
2510         Scode=7
2520     END IF
2530 !
2540 SUBEND !Iden_port
2550 !
```

SRL_SRQ Example Program

```
1000 !Filename:  SRL_SRQ (option 100 only)
1010 !
1020 ! Description:
1030 !
1040 ! This example program demonstrates how to initiate an SRL
1050 ! cable scan.  The instrument is set-up to send a
1060 ! SRQ interrupt when the scan has been completed.
1070 !
1080 ! Connect the cable to be tested to the RF out port on the
1090 ! analyzer.
1100 !
1110 ! Set an SRQ to occur when the SRL scan is complete.
1120 !
1130 !
1140 COM /Sys_state/ @Hp87xx,Scode
1150 ! Identify I/O Port
1160 CALL Iden_port
1170 !
1180 !
1190 ! Preset the instrument
1200 OUTPUT @Hp87xx;"SYST:PRES;*OPC?"
1210 ENTER @Hp87xx;Opc
1220 !
1230 ! Turn on SRL measurement
1240 OUTPUT @Hp87xx;"SENS1:FUNC 'SRL 1,0';DET NBAN;*OPC?"
1250 ENTER @Hp87xx;Opc
1260 !
1270 ! Clear status registers.
1280 OUTPUT @Hp87xx;"*CLS"
1290 !
1300 ! Clear the Service Request Enable register.
1310 OUTPUT @Hp87xx;"*SRE 0"
1320 !
1330 ! Clear the Standard Event Status Enable register.
1340 OUTPUT @Hp87xx;"*ESE 0"
1350 !
1360 ! Preset the remaining status registers.
1370 OUTPUT @Hp87xx;"STAT:PRES"
1380 !
1390 ! Set operation status register to report
1400 ! to the status byte on NEGATIVE transition
1410 ! the srl bit.
1420 OUTPUT @Hp87xx;"STAT:OPER:ENAB 16"
1430 OUTPUT @Hp87xx;"STAT:OPER:MEAS:PTR #H0000"
1440 OUTPUT @Hp87xx;"STAT:OPER:MEAS:NTR #HFFFF"
1450 !
1460 ! Set measuring status register to report to
1470 ! operational status register on NEGATIVE transition
1480 ! of the srl scan done bits.  The NEGATIVE
1490 ! transition needs to be detected because the
1500 ! srl= bit 3 is set to 1 while the analyzer
1510 ! is sweeping on channel 1  When the bit
1520 ! goes back to 0, the srl scan is done.
1530 OUTPUT @Hp87xx;"STAT:OPER:MEAS:ENAB 4"
1540 !
1550 ! Enable the operational status bit in the status
1560 ! byte to generate an SRQ.
```

```
1570 OUTPUT @Hp87xx;"*SRE 128"
1580 !
1590 ! On an interrupt from HP-IB "Scode" (Interface
1600 ! Select Code) SRQ bit (2), branch to the interrupt
1610 ! service routine "Srq_handler".
1620 ON INTR Scode,2 GOSUB Srq_handler
1630 !
1640 ! Initialize flag indicating when srl scan done
1650 ! to 0.  Then loop continuously until the
1660 ! interrupt is detected, and the interrupt
1670 ! service routine acknowledges the
1680 ! interrupt and sets the flag to 1.
1690 !
1700 Srl_done=0
1710 ! Now enable the interrupt on SRQ (Service Request).
1720 ENABLE INTR Scode;2
1730 !
1740 ! Initiate the SRL sweep
1750 OUTPUT @Hp87xx;"SENS1:FUNC:SRL:SCAN;*WAI"
1760 !
1770 DISP "Waiting for SRQ on srl scan done.";
1780 LOOP
1790     DISP ".";
1800     WAIT 1! Slow down dots
1810 EXIT IF Srl_done=1
1820 END LOOP
1830 !
1840 ! Display desired completion message.
1850 DISP
1860 DISP "Got SRQ.  SRL Scan!"
1870 STOP
1880 !
1890 Srq_handler:  ! Interrupt Service Routine
1900 !
1910 ! Determine that the analyzer was actually
1920 ! the instrument that generated the
1930 ! interrupt.
1940 Stb=SPOLL(@Hp87xx)
1950 !
1960 ! Determine if the operation status register
1970 ! caused the interrupt by looking at bit 7
1980 ! of the result of the serial poll.
1990 IF BINAND(Stb,128)<>0 THEN
2000 !
2010 ! Read the operational status event register.
2020     OUTPUT @Hp87xx;"STAT:OPER:EVEN?"
2030     ENTER @Hp87xx;Op_event
2040 !
2050 ! Determine if the srl status register
2060 ! bit 4 is set.
2070     IF BINAND(Op_event,16)<>0 THEN
2080 !
2090 ! If so, then set flag indicating
2100 ! srl scan done.
2110         Srl_done=1
2120     END IF
2130 END IF
2140 RETURN
2150 END
2160 !
```

```
2170 !**************************************************************
2180 ! Iden_port:   Identify io port to use.
2190 ! Description: This routines sets up the I/O port address for
2200 !              the SCPI interface.  For "HP 87xx" instruments,
2210 !              the address assigned to @Hp87xx = 800 otherwise,
2220 !              716.
2230 !**************************************************************
2240 SUB Iden_port
2250     COM /Sys_state/ @Hp87xx,Scode
2260 !
2270     IF POS(SYSTEM$("SYSTEM ID"),"HP 87")<>0 THEN
2280         ASSIGN @Hp87xx TO 800
2290         Scode=8
2300     ELSE
2310         ASSIGN @Hp87xx TO 716
2320         Scode=7
2330     END IF
2340 !
2350 SUBEND !Iden_port
2360 !
```

Transferring Data

MARKERS	This program transfers data using markers. The example also demonstrates the use of the query form of command mnemonics.
SMITHMKR	This program measures the reflection of a filter, and displays the measurements in Smith chart and polar formats.
ASCDATA	This program transfers data using the ASCII format.
REALDATA	This program transfers data using the IEEE 64-bit floating point REAL format. The example also demonstrates block data transfers of both indefinite length and definite length syntax. Also demonstrated is access to the swapped-byte data format designed for PCs.
INTDATA	This program transfers data using the 16-bit INTEGER format.
FAST_CW	This program transfers marker data in CW measurement mode.
DATA_EXT	This program is designed to run on an external controller. This controller can be HP BASIC for Windows running on a PC or HP BASIC running on an Agilent workstation. This program demonstrates how to transfer data from an IBASIC program running on the analyzer to an HP BASIC or IBASIC program running externally. It loads a program into the analyzer, runs it, sets a variable, and then gives it control of the bus. This program then acts as a device on the bus (sending and receiving data).
DATA_INT	This program is designed to run on the analyzer's internal IBASIC controller. This program demonstrates how to transfer data to and from an external controller. In this example a catalog listing is transferred from the analyzer to the external

controller. A numeric variable value is also downloaded from the external controller to the analyzer’s program.

MARKERS Example Program

This program demonstrates how to transfer measurement data by using the markers. Before any data is read over the GPIB, a controlled sweep should be taken. The analyzer has the ability to process and execute commands very quickly when they are received over the GPIB. This speed can lead to commands (such as marker searches) being executed before any data has been taken. To ensure that the sweep has completed and the data is present before it is read, the command for a single sweep is used before data is requested. Note that `*WAI` is sent with that command. More information about making measurements with the analyzer is available in the *User's Guide*.

```
1000 !Filename:  MARKERS
1010 !
1020 ! Description:
1030 !   1. Take sweep
1040 !   2. Set marker to 175 MHz, and query Y value
1050 !   3. Execute Marker -> Max, and query X and Y
1060 !   4. Turn on marker tracking
1070 !   5. Execute a 3 dB bandwidth search
1080 !   6. Query the result
1090 !
1100 COM /Sys_state/ @Hp87xx,Scode
1110 ! Identify I/O Port
1120 CALL Iden_port
1130 !
1140 !
1150 ! Turn on channel 1 and set up start and stop
1160 ! frequencies for the example.  These frequencies
1170 ! were chosen for the demonstration filter that is
1180 ! shipped with the analyzer.
1190 OUTPUT @Hp87xx;"SENS1:STAT ON;FREQ:STAR 10 MHZ;STOP 400 MHZ;*WAI"
1200 !
1210 ! Configure a transmission measurement on channel 1
1220 ! using the narrowband detection mode.
1230 OUTPUT @Hp87xx;"SENS1:FUNC 'XFR:POW:RAT 2,0';DET NBAN;*WAI"
1240 !
1250 ! Take a single controlled sweep and have the
1260 ! analyzer wait until it has completed before
1270 ! executing the next command.
1280 OUTPUT @Hp87xx;"ABOR;:INIT:CONT OFF;:INIT;*WAI"
1290 !
1300 ! Turn on the first marker.
1310 OUTPUT @Hp87xx;"CALC1:MARK1 ON"
1320 !
1330 ! Set marker 1 to a frequency of 175 MHz.
1340 OUTPUT @Hp87xx;"CALC1:MARK1:X 175 MHZ"
1350 !
1360 ! Query the amplitude of the signal at 175 MHz.
1370 OUTPUT @Hp87xx;"CALC1:MARK1:Y?"
1380 !
1390 ! Read the data; the data is in the NR3 format.
1400 ENTER @Hp87xx;Data_1
1410 DISP "Marker 1 (175 MHz) = ";Data_1
```

```
1420 WAIT 5
1430 !
1440 ! Turn on the second marker and use a marker
1450 ! search function to find the maximum point
1460 ! on the data trace.
1470 OUTPUT @Hp87xx;"CALC1:MARK2 ON;MARK2:MAX"
1480 !
1490 ! Query the frequency and amplitude of the
1500 ! maximum point.  Note that the two queries can
1510 ! be combined into one command.
1520 OUTPUT @Hp87xx;"CALC1:MARK2:X?;Y?"
1530 !
1540 ! Read the data.
1550 ENTER @Hp87xx;Freq2,Data2
1560 !
1570 ! Display the results of the marker search.
1580 DISP "Max = ";Data2;"dB at";Freq2/1.E+6;"MHz"
1590 !
1600 ! Put the analyzer into its continuously
1610 ! sweeping mode.  This mode works well for
1620 ! tuning applications.
1630 OUTPUT @Hp87xx;"INIT:CONT ON;*WAI"
1640 !

1650 ! Turn on the marker search tracking function.
1660 ! This function causes the marker 2 to track
1670 ! the maximum value each time the analyzer takes
1680 ! a sweep.
1690 OUTPUT @Hp87xx;"CALC1:MARK2:FUNC:TRAC ON"
1700 WAIT 5
1710 !
1720 ! Turn off marker 2.
1730 OUTPUT @Hp87xx;"CALC1:MARK2 OFF"
1740 !
1750 ! Take a single controlled sweep.
1760 OUTPUT @Hp87xx;"ABOR;:INIT:CONT OFF;:INIT;*WAI"
1770 !
1780 ! Perform a search for the -3 dB bandwidth of
1790 ! the filter.  This function uses several
1800 ! markers to find four key values.
1810 OUTPUT @Hp87xx;"CALC1:MARK:BWID -3;FUNC:RES?"
1820 !
1830 ! Read the four values:  the bandwidth, center
1840 ! frequency, Q and the insertion loss.
1850 ENTER @Hp87xx;Bwid,Center_f,Q,Loss
1860 !
1870 ! Display the results.
1880 DISP "BW: ";Bwid
1890 WAIT 5
1900 DISP "Center Freq: ";Center_f
1910 WAIT 5
1920 DISP "Q: ";Q
1930 WAIT 5
1940 DISP "Loss: ";Loss
1950 !
1960 ! Turn off all the markers.
1970 OUTPUT @Hp87xx;"CALC1:MARK:AOFF"
1980 END
1990 !
2000 !**************************************************************
```

```
2010 ! Iden_port:   Identify io port to use.
2020 ! Description: This routines sets up the I/O port address for
2030 !              the SCPI interface.  For "HP 87xx" instruments,
2040 !              the address assigned to @Hp87xx = 800 otherwise,
2050 !              716.
2060 !**************************************************************
2070 SUB Iden_port
2080     COM /Sys_state/ @Hp87xx,Scode
2090 !
2100     IF POS(SYSTEM$("SYSTEM ID"),"HP 87")<>0 THEN
2110         ASSIGN @Hp87xx TO 800
2120         Scode=8
2130     ELSE
2140         ASSIGN @Hp87xx TO 716
2150         Scode=7
2160     END IF
2170 !
2180 SUBEND !Iden_port
2190 !
```

SMITHMKR Example Program

```
1000 !Filename:  SMITHCHART
1010 !
1020 !
1030 ! Description:  Measures a 175MHz BPF using the
1040 !   Smith and Polar plot formats.  User must connect
1050 !   the 175MHz filter between the reflection and transmission
1060 !   ports.  The program will do the following:
1070 !     1) Set analyzer to sweep over the filter's passband (50MHz).
1080 !     2) Set analyzer to Smith Chart format; make a marker
1090 !          reading (Frequency, Real Impedance in ohms, Imaginary
               Impedance
1100 !          in ohms, Impedance Capacitance or Inductance); dump the
1110 !          trace and print S11 Real and Imaginary values for the
1120 !          first data point.
1130 !     3) Set analyzer to Polar Chart format; make a marker
1140 !          reading (Frequency, Linear Magnitude in "units",
1150 !          Phase in degrees); dump the
1160 !          trace and print S11 Real and Imaginary values for the
1170 !          first data point.
1180 !
1190 !**************************************************
1200 ! DEFINITIONS
1210 !
1220 REAL Opc,Freq_center,Freq_span,Freq_start,Bpf_q,Bpf_loss
1230 REAL Mrkr_freq,Mrkr_res,Mrkr_reac,Mrkr_ind
1240 REAL Trace_s11(1:201,1:2),Mrkr_mag,Mrkr_phas
1250 !
1260 !***************************************************
1270 ! Determine computer type
1280 !
1290 CLEAR SCREEN
1300 !
1310 !
1320 COM /Sys_state/ @Hp87xx,Scode
1330 ! Identify I/O Port
1340 CALL Iden_port
1350 !
1360 !
1370 !-
1380 ! Preset analyzer, set Center and Span frequencies
1390 !
1400 OUTPUT @Hp87xx;"SYST:PRES;*OPC?"               !preset instrument
1410 ENTER @Hp87xx;Opc           !waits for PRESET to finish before
     proceeding.
1420 !
1430 ! Center the filter's frequency response (to get an accurate Bandwidth
       measurement).
1440 !
1450 DISP "Setting analyzer frequencies..."          !message to user
1460 OUTPUT @Hp87xx;"ABOR;:INIT:CONT OFF;:INIT;*OPC?" !take a single sweep
1470 ENTER @Hp87xx;Opc                               !wait for sweep to
     finish
1480 OUTPUT @Hp87xx;"CALC1:MARK:FUNC MAX;*WAI"      !set Marker 1 to max
1490 OUTPUT @Hp87xx;"CALC1:MARK:X?;*WAI"            !get Marker frequency
     setting
1500 ENTER @Hp87xx;Mrkr_freq                        !read frequency of max
     marker
```

```
1510 OUTPUT @Hp87xx;"SENS1:FREQ:CENT "&VAL$(Mrkr_freq)&" HZ;*WAI"  !set
     Center Freq
1520 OUTPUT @Hp87xx;"SENS1:FREQ:SPAN 200 MHZ;*WAI"  !set Span Freq = 200MHz
1530 !
1540 ! Measure Bandwidth, set Center to band center, Span to 50MHz
1550 !
1560 OUTPUT @Hp87xx;"ABOR;:INIT:CONT OFF;:INIT;*OPC?" !take a single sweep
1570 ENTER @Hp87xx;Opc                              !wait for sweep to
     finish
1580 OUTPUT @Hp87xx;"CALC1:MARK:FUNC BWID;*OPC?"    !search filter for -3dB
     bandwidth
1590 ENTER @Hp87xx;Opc                              !wait for bandwidth to
     be found
1600 OUTPUT @Hp87xx;"CALC1:MARK:FUNC:RES?"          !read the bandwidth data
1610 ENTER @Hp87xx;Freq_span,Freq_center,Bpf_q,Bpf_loss             !read
     in data
1620 OUTPUT @Hp87xx;"SENS1:FREQ:CENT "&VAL$(Freq_center)&" HZ;*WAI"  !set
     Center Freq

1630 OUTPUT @Hp87xx;"SENS1:FREQ:SPAN 50 MHZ;*WAI"   !set Span Freq to 50MHz
     (passband)
1640 !
1650 !-
1660 ! Set marker 1 to beginning of trace.
1670 !
1680 OUTPUT @Hp87xx;"CALC1:MARK:AOFF;*WAI"          !clear all markers
1690 OUTPUT @Hp87xx;"CALC1:MARK1 ON"                !turn on marker 1
1700 OUTPUT @Hp87xx;"SENS1:FREQ:STAR?"              !get start frequency
1710 ENTER @Hp87xx;Freq_start                       !enter start freq
1720 OUTPUT @Hp87xx;"CALC1:MARK1:X "&VAL$(Freq_start)&";*OPC?"  !set marker
     to start freq
1730 ENTER @Hp87xx;Opc                              !wait for all previous
     commands to finish
1740 !
1750 !
1760 ! Set to Reflection mode & Smith Chart format.
1770 !
1780 DISP "Setting to Smith Chart format..."
1790 OUTPUT @Hp87xx;"ABOR;:INIT1:CONT ON;*WAI" !set to Cont Sweep mode so
     can select reflection
1800 OUTPUT @Hp87xx;"SENS1:FUNC 'XFR:POW:RAT 1,0';DET NBAN;*WAI"
         !CHAN1=reflection
1810 OUTPUT @Hp87xx;"CALC1:FORM SMIT;*WAI"          !set smith chart format
1820 !
1830 !-
1840 ! Read marker information from Smith Chart.
1850 !
1860 OUTPUT @Hp87xx;"ABOR;:INIT:CONT OFF;:INIT;*OPC?" !force one sweep
     before read markers
1870 ENTER @Hp87xx;Opc                              !wait for sweep to
     finish
1880 OUTPUT @Hp87xx;"ABOR;:INIT1:CONT ON;*WAI"      !set to Continuous Sweep
     mode
1890 OUTPUT @Hp87xx;"CALC1:MARK:X?"                 !read marker frequency
1900 ENTER @Hp87xx;Mrkr_freq                        !units are in Hz
1910 OUTPUT @Hp87xx;"CALC1:MARK:Y:RES?"             !read real part of
     marker impedance
1920 ENTER @Hp87xx;Mrkr_res                         !units are in ohms
1930 OUTPUT @Hp87xx;"CALC1:MARK:Y:REAC?"            !read imaginary part of
     marker impedance
```

```
1940 ENTER @Hp87xx;Mrkr_reac                        !units are in ohms
1950 OUTPUT @Hp87xx;"CALC1:MARK:Y:IND?"             !read inductance (or
     capacitance)
1960 ENTER @Hp87xx;Mrkr_ind    !units are Henries if positive value, Farads
     if negative
1970 !
1980 !-
1990 ! Display Smith Marker data.
2000 !
2010 Mrkr_freq=DROUND(Mrkr_freq,3)                        !round frequency
     to 3 digits
2020 DISP "Smith Marker Frequency = "&VAL$(Mrkr_freq)&"Hz"      !display
     frequency
2030 WAIT 3
2040 !
2050 Mrkr_res=DROUND(Mrkr_res,3)                          !round resistance
     to 3 digits
2060 DISP "Smith Marker Resistance = "&VAL$(Mrkr_res)&" ohms"
2070 WAIT 3
2080 !
2090 Mrkr_reac=DROUND(Mrkr_reac,3)                        !round reactance
     to 3 digits
2100 DISP "Smith Marker Reactance = "&VAL$(Mrkr_reac)&" ohms"
2110 WAIT 3
2120 !
2130 Mrkr_ind=DROUND(Mrkr_ind,3)                          !round inductance
     to 3 digits
2140 IF Mrkr_ind<0 THEN                                   !label as
     capacitance if negative
2150     DISP "Smith Marker Capacitance = "&VAL$(-Mrkr_ind)&"F"!label
         capacitance
2160 ELSE                                                 !label as
     inductance if positive
2170     DISP "Smith Marker Inductance = "&VAL$(Mrkr_ind)&"H"  !label
         inductance
2180 END IF
2190 WAIT 3
2200 !

2210 !
2220 ! Read Smith Chart formatted trace data, display first data point.
2230 !   Data is transferred in ASCII format with 3 significant digits.
2240 !   S11 trace data is read out as:  Real data for point #1, Imaginary
         data
2250 !   for point #1, Real data for point #2, Imaginary data for point
         #2...
2260 !   Since instrument was preset, number of trace data points
2270 !   defaults to 201.
2280 !
2290 OUTPUT @Hp87xx;"FORM:DATA ASC,3;:TRAC? CH1FDATA"      !set up to read
     ASCII data, 3 digits
2300 ENTER @Hp87xx;Trace_s11(*)                            !read trace data,
     real & imaginary pairs
2310 !
2320 ! Display data.
2330 !
2340 DISP "Smith Trace Point #1:  S11(REAL) = "&VAL$(Trace_s11(1,1))&"
     Units"  !display Real data
2350 WAIT 3
2360 DISP "Smith Trace Point #1:  S11(IMAGINARY) = "&VAL$(Trace_s11(1,2))&"
```

```
     Units"  !display Imaginary data
2370 WAIT 3
2380 !
2390 !+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
2400 ! Set to Polar Chart Format, read Polar Markers.
2410 !
2420 DISP "Setting to Polar Format..."
2430 OUTPUT @Hp87xx;"CALC1:FORM POL;*WAI"           !set polar chart format
2440 OUTPUT @Hp87xx;"CALC1:MARK:X?"                 !read marker frequency
2450 ENTER @Hp87xx;Mrkr_freq                        !units are in Hz
2460 OUTPUT @Hp87xx;"CALC1:MARK:Y:MAGN?"            !read magnitude marker
     reflection coefficient
2470 ENTER @Hp87xx;Mrkr_mag                         !magnitude in "units"
2480 OUTPUT @Hp87xx;"CALC1:MARK:Y:PHAS?"            !read phase of marker
     reflection coefficient
2490 ENTER @Hp87xx;Mrkr_phas                        !units are in degrees
2500 !

2510 !-
2520 ! Display Polar Marker data.
2530 !
2540 Mrkr_freq=DROUND(Mrkr_freq,3)                         !round frequency
     to 3 digits
2550 DISP "Polar Marker Frequency = "&VAL$(Mrkr_freq)&"Hz"       !display
     frequency
2560 WAIT 3
2570 !
2580 Mrkr_mag=DROUND(Mrkr_mag,3)                           !round magnitude
     to 3 digits
2590 DISP "Polar Marker Magnitude = "&VAL$(Mrkr_mag)&" Units"    !display
     magnitude
2600 WAIT 3
2610 !
2620 Mrkr_phas=DROUND(Mrkr_phas,3)                         !round phase to 3
     digits
2630 DISP "Polar Marker Phase = "&VAL$(Mrkr_phas)&" Degrees"     !display
     phase
2640 WAIT 3
2650 !
2660 !
2670 ! Read Polar Chart trace data, display first data point.
2680 !   S11 trace data is read out as:  Real data for point #1, Imaginary
         data
2690 !   for point #1, Real data for point #2, Imaginary data for point
         #2...
2700 !
2710 OUTPUT @Hp87xx;"FORM:DATA ASC,3;:TRAC? CH1FDATA"      !set up to read
     ASCII data, 3 digits
2720 ENTER @Hp87xx;Trace_s11(*)                            !read trace data,
     real & imaginary pairs
2730 !
2740 ! Display data
2750 !
2760 DISP "Polar Trace Point #1:  S11(REAL) = "&VAL$(Trace_s11(1,1))&"
     Units"  !display Real data
2770 WAIT 3
2780 DISP "Polar Trace Point #1:  S11(IMAGINARY) = "&VAL$(Trace_s11(1,2))&"
     Units"  !display Imaginary data
2790 WAIT 3
```

```
2800 DISP ""                                                   !clear display
     line
2810 !
2820 STOP
2830 END
2840 !
2850 !***************************************************************
2860 ! Iden_port:    Identify io port to use.
2870 ! Description: This routines sets up the I/O port address for
2880 !              the SCPI interface.  For "HP 87xx" instruments,
2890 !              the address assigned to @Hp87xx = 800 otherwise,
2900 !              716.
2910 !***************************************************************
2920 SUB Iden_port
2930     COM /Sys_state/ @Hp87xx,Scode
2940 !
2950     IF POS(SYSTEM$("SYSTEM ID"),"HP 87")<>0 THEN
2960         ASSIGN @Hp87xx TO 800
2970         Scode=8
2980     ELSE
2990         ASSIGN @Hp87xx TO 716
3000         Scode=7
3010     END IF
3020 !
3030 SUBEND !Iden_port
3040 !
```

ASCDATA Example Program

This program demonstrates how to read data arrays from the analyzer and write them back again. The ASCii data format is being used with a resolution of 5 digits. More information about data transfer is available in "Data Types and Encoding," and "Trace Data Transfers" in the *Programmer's Guide*.

In addition to the channel 1 formatted data array used in this example, there are a number of arrays that can be accessed inside the instrument. These arrays and their corresponding mnemonics are listed in the *Programmer's Guide*.

```
1000 !Filename:  ASCDATA
1010 !
1020 ! Description:
1030 !  1. Takes a sweep, and reads the formatted
1040 !     data trace into an array.  The trace
1050 !     is read as a definite length block.
1060 !  2. Instructs you to remove DUT.
1070 !  3. Downloads the trace back to the analyzer
1080 !     as an indefinite length block.
1090 REAL Data1(1:51)
1100 !
1110 COM /Sys_state/ @Hp87xx,Scode
1120 ! Identify I/O Port
1130 CALL Iden_port
1140 !
1150 !
1160 ! Set the analyzer to measure 51 data points.
1170 OUTPUT @Hp87xx;"SENS1:SWE:POIN 51;*WAI"
1180 !
1190 ! Take a single sweep, leaving the analyzer
1200 ! in trigger hold mode.
1210 OUTPUT @Hp87xx;"ABOR;:INIT1:CONT OFF;:INIT1;*WAI"
1220 !
1230 ! Set up the ASCII data format with 5
1240 ! significant digits
1250 OUTPUT @Hp87xx;"FORM:DATA ASC,5"
1260 !
1270 ! request the channel 1 formatted data array
1280 ! from the instrument.
1290 OUTPUT @Hp87xx;"TRAC? CH1FDATA"
1300 !
1310 ! Get the data and put into data array Data1.
1320 ENTER @Hp87xx;Data1(*)
1330 !
1340 ! Display the first 3 numbers in the array.
1350 DISP "Trace: ";Data1(1);Data1(2);Data1(3);"..."
1360 !
1370 ! Use the wait time to visually compare the
1380 ! numbers to the visible data trace.
1390 WAIT 5
1400 !
1410 ! Prompt the operator to disconnect the test
1420 ! device and then how to continue the program.
```

```
1430 DISP "Disconnect the test device  Press Continue"
1440 PAUSE
1450 !
1460 ! Update the display line.
1470 DISP "Taking a new sweep...";
1480 !
1490 ! Take a sweep so the display shows new data.
1500 OUTPUT @Hp87xx;":INIT1;*WAI"
1510 DISP " Done."
1520 WAIT 5
1530 !
1540 ! Prepare the analyzer to receive the data.
1550 ! Suppress the "end" character by using a
1560 ! semicolon at end of output statement.
1570 DISP "Downloading saved trace...";
1580 OUTPUT @Hp87xx;"TRAC CH1FDATA";
1590 !
1600 ! Send the data array one point at a time,
1610 ! using the semicolon at the end of the
1620 ! output statement to suppress the
1630 ! end character.
1640 FOR I=1 TO 51
1650     OUTPUT @Hp87xx;", ";Data1(I);
1660 NEXT I
1670 !
1680 ! Now send the end character.
1690 OUTPUT @Hp87xx;""
1700 DISP " Done!"
1710 END
1720 !
1730 !**************************************************************
1740 ! Iden_port:   Identify io port to use.
1750 ! Description: This routines sets up the I/O port address for
1760 !              the SCPI interface.  For "HP 87xx" instruments,
1770 !              the address assigned to @Hp87xx = 800 otherwise,
1780 !              716.
1790 !**************************************************************
1800 SUB Iden_port
1810     COM /Sys_state/ @Hp87xx,Scode
1820 !
1830     IF POS(SYSTEM$("SYSTEM ID"),"HP 87")<>0 THEN
1840         ASSIGN @Hp87xx TO 800
1850         Scode=8
1860     ELSE
1870         ASSIGN @Hp87xx TO 716
1880         Scode=7
1890     END IF
1900 !
1910 SUBEND !Iden_port
1920 !
```

REALDATA Example Program

This program demonstrates how to read data arrays from the analyzer and write them back again. The `REAL,64` data format is being used. Note that the analyzer outputs the data using the definite length block syntax. This example uses the indefinite length block syntax when data is being written back to the analyzer. More information about data transfer is available in "Data Types and Encoding" in the *Programmer's Guide*. All of the arrays listed in the `ASCDATA` example section can also be accessed using this data format.

```
1000 !Filename:  REALDATA
1010 !
1020 ! Description:
1030 !  1. Takes a sweep, and reads the formatted
1040 !     data trace into an array.  The trace
1050 !     is read as a definite length block.
1060 !  2. Instructs you to remove DUT.
1070 !  3. Downloads the trace back to the analyzer
1080 !     as an indefinite length block.
1090 DIM A$[10],Data1(1:51)
1100 INTEGER Digits,Bytes
1110 !
1120 COM /Sys_state/ @Hp87xx,Scode
1130 ! Identify I/O Port
1140 CALL Iden_port
1150 !
1160 !
1170 ! Set up the analyzer to measure 51 data points.
1180 OUTPUT @Hp87xx;"SENS1:SWE:POIN 51;*WAI"
1190 !
1200 ! Take a single sweep, leaving the analyzer
1210 ! in trigger hold mode.
1220 OUTPUT @Hp87xx;"ABOR;:INIT1:CONT OFF;:INIT1;*WAI"
1230 !
1240 ! Select binary block transfer.
1250 OUTPUT @Hp87xx;"FORM:DATA REAL,64"
1260 !
1270 ! Request the channel 1 formatted data array
1280 ! from the analyzer.
1290 OUTPUT @Hp87xx;"TRAC? CH1FDATA"
1300 !
1310 ! Turn on ASCII formatting on the I/O path.
1320 ! It is needed for reading the header
1330 ! information.
1340 ASSIGN @Hp87xx;FORMAT ON
1350 !
1360 ! Get the data header.  "A$" will contain the
1370 ! "#" character indicating a block data transfer.
1380 ! "Digits" will contain the number of characters
1390 ! for the number of bytes value which follows.
1400 ENTER @Hp87xx USING "%,A,D";A$,Digits
1410 !
1420 ! Get the rest of the header.  The number of
1430 ! bytes to capture in the data array will be
1440 ! placed in "Bytes".  Note the use of "Digits"
```

```
1450 ! in the IMAGE string.
1460 ENTER @Hp87xx USING "%,"&VAL$(Digits)&"D";Bytes
1470 !
1480 ! Turn off ASCII formatting on the I/O path;
1490 ! it is not needed for transferring binary
1500 ! formatted data.
1510 ASSIGN @Hp87xx;FORMAT OFF
1520 !
1530 ! Get the data.
1540 ENTER @Hp87xx;Data1(*)
1550 !
1560 ! Turn on ASCII formatting again.
1570 ASSIGN @Hp87xx;FORMAT ON
1580 !
1590 ! Get the "end of data" character.
1600 ENTER @Hp87xx;A$
1610 !
1620 ! Display the first three numbers in the array.
1630 DISP "Trace: ";Data1(1);Data1(2);Data1(3);"..."
1640 !
1650 ! Use this time to visually compare the
1660 ! numbers to the visible data trace.
1670 WAIT 5
1680 !
1690 ! Prompt the operator to disconnect the test
1700 ! device and how to continue the program.
1710 DISP "Disconnect the test device  Press Continue"
1720 PAUSE
1730 !
1740 ! Update the display line.
1750 DISP "Taking a new sweep...";
1760 !
1770 ! Take a sweep so the display shows new data.
1780 OUTPUT @Hp87xx;":INIT1;*WAI"
1790 DISP " Done."
1800 WAIT 5
1810 !
1820 ! Send the header for an indefinite block length
1830 ! data transfer.
1840 DISP "Downloading saved trace...";
1850 OUTPUT @Hp87xx;"TRAC CH1FDATA, #0";
1860 !
1870 ! Turn off ASCII formatting.
1880 ASSIGN @Hp87xx;FORMAT OFF
1890 !
1900 ! Send the data array back to the analyzer.
1910 OUTPUT @Hp87xx;Data1(*),END
1920 !
1930 ! Turn on ASCII formatting again.
1940 ASSIGN @Hp87xx;FORMAT ON
1950 DISP " Done!"
1960 END
1970 !
1980 !**************************************************************
1990 ! Iden_port:   Identify io port to use.
2000 ! Description: This routines sets up the I/O port address for
2010 !              the SCPI interface.  For "HP 87xx" instruments,
2020 !              the address assigned to @Hp87xx = 800 otherwise,
2030 !              716.
2040 !**************************************************************
```

```
2050 SUB Iden_port
2060     COM /Sys_state/ @Hp87xx,Scode
2070 !
2080     IF POS(SYSTEM$("SYSTEM ID"),"HP 87")<>0 THEN
2090         ASSIGN @Hp87xx TO 800
2100         Scode=8
2110     ELSE
2120         ASSIGN @Hp87xx TO 716
2130         Scode=7
2140     END IF
2150 !
2160 SUBEND !Iden_port
2170 !
```

INTDATA Example Program

This program demonstrates how to read data arrays from the analyzer and write them back again. The INTeger,16 data format is used. This data format is the instrument's internal format. It should only be used to read data that will later be returned to the instrument.

The data array dimensioned in line 1100 is different from the arrays in either REAL,64 or ASCii examples. This is because each data point is represented by a set of three 16-bit integers. Another difference in using this data format is that all arrays cannot be accessed with it. The formatted data arrays CH1FDATA and CH2FDATA cannot be read using the INTEGER format.

Note that the analyzer outputs the data using the definite length block syntax. This example uses the indefinite length block syntax when data is being written back to the analyzer. More information about data transfer is available in "Data Types and Encoding" in the *Programmer's Guide*.

```
1000 !Filename:  INTDATA
1010 !
1020 ! Description:
1030 !  1. Takes a sweep, and reads the formatted
1040 !     data trace into an array.  The trace
1050 !     is read as a definite length block.
1060 !  2. Instructs you to remove DUT.
1070 !  3. Downloads the trace back to the analyzer
1080 !     as an indefinite length block.
1090 DIM A$[10]
1100 INTEGER Digits,Bytes,Data1(1:51,1:3)
1110 !
1120 COM /Sys_state/ @Hp87xx,Scode
1130 ! Identify I/O Port
1140 CALL Iden_port
1150 !
1160 !
1170 ! Set up the analyzer to measure 51 data points.
1180 OUTPUT @Hp87xx;"SENS1:SWE:POIN 51;*WAI"
1190 !
1200 ! Take a single sweep, leaving the analyzer
1210 ! in trigger hold mode.
1220 OUTPUT @Hp87xx;"ABOR;:INIT1:CONT OFF;:INIT1;*WAI"
1230 !
1240 ! Select binary block transfer
1250 OUTPUT @Hp87xx;"FORM:DATA INT,16"
1260 !
1270 ! Request the channel 1 unformatted data array
1280 ! from the analyzer.
1290 OUTPUT @Hp87xx;"TRAC? CH1SDATA"
1300 !
1310 ! Turn on ASCII formatting on the I/O path;
1320 ! it is needed for reading the header information.
1330 ASSIGN @Hp87xx;FORMAT ON
```

```
1340 !
1350 ! Get the data header.  "A$" will contain the
1360 ! "#" character indicating a block data transfer.
1370 ! "Digits" will contain the number of characters
1380 ! for the number of bytes value which follows.
1390 ENTER @Hp87xx USING "%,A,D";A$,Digits
1400 !
1410 ! Get the rest of the header.  The number of
1420 ! bytes to capture in the data array will be
1430 ! placed in "Bytes".  Note the use of "Digits"
1440 ! in the IMAGE string.
1450 ENTER @Hp87xx USING "%,"&VAL$(Digits)&"D";Bytes
1460 !
1470 ! Turn off ASCII formatting on the I/O path;
1480 ! it is not needed for transferring binary
1490 ! formatted data.
1500 ASSIGN @Hp87xx;FORMAT OFF
1510 !
1520 ! Get the data.
1530 ENTER @Hp87xx;Data1(*)
1540 !
1550 ! Turn on ASCII formatting again.
1560 ASSIGN @Hp87xx;FORMAT ON
1570 !
1580 ! Get the "end of data" character.
1590 ENTER @Hp87xx;A$
1600 !
1610 ! Display the first 3 numbers; there will
1620 ! be no visible similarity between these
1630 ! numbers and the data displayed on the
1640 ! analyzer.
1650 DISP "Trace: ";Data1(1,1);Data1(1,2);Data1(1,3);"..."
1660 WAIT 5
1670 !
1680 ! Prompt the operator to disconnect the test
1690 ! device and how to continue the program.
1700 DISP "Disconnect the test device  Press Continue"
1710 PAUSE
1720 !
1730 ! Update the display line.
1740 DISP "Taking a new sweep...";
1750 !
1760 ! Take a sweep so the display shows new data.
1770 OUTPUT @Hp87xx;":INIT1;*WAI"
1780 DISP " Done."
1790 WAIT 5
1800 !
1810 ! Send the header for an indefinite block length
1820 ! data transfer.
1830 DISP "Downloading saved trace...";
1840 OUTPUT @Hp87xx;"TRAC CH1SDATA, #0";
1850 !
1860 ! Turn off ASCII formatting.
1870 ASSIGN @Hp87xx;FORMAT OFF
1880 !
1890 ! Send the data back to the analyzer.
1900 OUTPUT @Hp87xx;Data1(*),END
1910 !
1920 ! Turn on ASCII formatting.
1930 ASSIGN @Hp87xx;FORMAT ON
```

```
1940 DISP "Done!"
1950 END
1960 !
1970 !***************************************************************
1980 ! Iden_port:   Identify io port to use.
1990 ! Description: This routines sets up the I/O port address for
2000 !              the SCPI interface.  For "HP 87xx" instruments,
2010 !              the address assigned to @Hp87xx = 800 otherwise,
2020 !              716.
2030 !***************************************************************
2040 SUB Iden_port
2050     COM /Sys_state/ @Hp87xx,Scode
2060 !
2070     IF POS(SYSTEM$("SYSTEM ID"),"HP 87")<>0 THEN
2080         ASSIGN @Hp87xx TO 800
2090         Scode=8
2100     ELSE
2110         ASSIGN @Hp87xx TO 716
2120         Scode=7
2130     END IF
2140 !
2150 SUBEND !Iden_port
2160 !
```

FAST_CW Example Program

This program demonstrates how to set up a CW (fixed frequency) sweep with the minimum number of trace points. Such a sweep allows measurements to be made very rapidly. The program also shows how to set up a loop which uses a fast CW sweep, reads a marker value on the measurement trace, then changes the CW frequency.

```
1000 ! Filename: FAST_CW
1010 !
1020 ! Description:
1030 !   Set sweep to CW, and select the
1040 !   fewest number of points.
1050 !   Change the frequency, take a sweep,
1060 !   and use a marker to read the trace.
1070 !   Repeat as quickly as possible.
1080 !
1090 DIM Freq_str$[20]
1100 DIM Msg$[100]
1110 !
1120 !
1130 COM /Sys_state/ @Hp87xx,Scode
1140 ! Identify I/O Port.
1150 CALL Iden_port
1160 !
1170 !
1180 ! PRESET, to ensure known state.
1190 OUTPUT @Hp87xx;"SYST:PRES;*WAI"
1200 CLEAR SCREEN
1210 !
1220 ! Set up the analyzer to measure 3 data points.
1230 OUTPUT @Hp87xx;"SENS1:SWE:POIN 3;*WAI"
1240 !
1250 ! Select CW display and sweep.
1260 OUTPUT @Hp87xx;"DISP:ANN:FREQ1:MODE CW"
1270 OUTPUT @Hp87xx;"SENS1:FREQ:SPAN 0 Hz;*WAI"
1280 !
1290 ! Take a single sweep, leaving the analyzer
1300 ! in trigger hold mode.
1310 OUTPUT @Hp87xx;"ABOR;:INIT1:CONT OFF;*WAI"
1320 !
1330 ! Turn on Marker 1
1340 OUTPUT @Hp87xx;"CALC:MARK1 ON"
1350 !
1360 Count=0
1370 T0=TIMEDATE
1380 ! Step from 175 MHz 463 MHz by 6 MHz
1390 FOR Freq=175 TO 463 STEP 6
1400   ! Take a sweep
1410     Freq_str$=VAL$(Freq)&" MHz"
1420     OUTPUT @Hp87xx;"SENS1:FREQ:CENT ";Freq_str$
1430     OUTPUT @Hp87xx;"INIT1;*WAI"
1440   !
1450   ! Set marker to frequency
1460     OUTPUT @Hp87xx;"CALC:MARK:X ";Freq_str$
1470   !
1480   ! Query the marker value
```

```
1490     OUTPUT @Hp87xx;"CALC:MARK:Y?"
1500     ENTER @Hp87xx;Response
1510   !
1520   ! Display the first three numbers in the array.
1530     Msg$="'"&Freq_str$&": "&VAL$(Response)&"'"
1540     OUTPUT @Hp87xx;"DISP:ANN:MESS ";Msg$
1550     PRINT Msg$
1560     Count=Count+1
1570 NEXT Freq
1580 T1=TIMEDATE
1590 PRINT "Sweeps per second: ";Count/(T1-T0)
1600 DISP "Sweeps per second: ";Count/(T1-T0)
1610 END
1620 !
1630 !**************************************************************
1640 ! Iden_port:    Identify io port to use.
1650 ! Description: This routines sets up the I/O port address for
1660 !              the SCPI interface.  For "HP 87xx" instruments,
1670 !              the address assigned to @Hp87xx = 800 otherwise,
1680 !              716.
1690 !**************************************************************
1700 SUB Iden_port
1710     COM /Sys_state/ @Hp87xx,Scode
1720 !
1730     IF POS(SYSTEM$("SYSTEM ID"),"HP 87")<>0 THEN
1740         ASSIGN @Hp87xx TO 800
1750         Scode=8
1760     ELSE
1770         ASSIGN @Hp87xx TO 716
1780         Scode=7
1790     END IF
1800 !
1810 SUBEND !Iden_port
1820 !
```

DATA_EXT Example Program

```
10    !-----------------------------------------------------
20    !
30    ! BASIC program:  DATA_EXT - Data transfer (external)
40    !
50    ! This program demonstrates how to transfer data from
60    ! an IBASIC program running on the HP 8711 to an
70    ! HP BASIC program (or an IBASIC program running
80    ! externally).  This program was designed to run on a
90    ! computer or PC.  It loads a program into the HP 8711,
100   ! runs it, and then gives it control of the bus.
110   ! This program then acts as a device on the bus;
120   ! sending and receiving data.
130   !
140   ! Before running this program, a disc with the program
150   ! ‘DATA_INT’ should be in the HP 8711’s internal drive.
160   !
170   !-----------------------------------------------------
180   !
190   ! Initialize variables for the interface select code
200   ! and the HP-IB address of the HP 8711.
210   !
220   Scode=7
230   Address=16
240   Na=Scode*100+Address
250   !
260   ! Abort any bus traffic, clear the input/output queues
270   ! of the analyzer, clear the analyzer’s status
280   ! registers and the display.
290   !
300   ABORT Scode
310   CLEAR Na
320   OUTPUT Na;”*CLS”
330   CLEAR SCREEN
340   !
350   ! Dimension an array to hold the catalog listing.
360   !
370   DIM Directory$(1:100)[85]
380   !
390   ! Prompt the operator to insert the disk in the
400   ! HP 8711, load the program and wait until done.
410   !
420   INPUT “Put disc with program ‘DATA_INT’ into the HP 8711. Press <ENTER>”,A$
430   DISP “Loading program on HP 8711 . . .”
440   OUTPUT Na;”PROG:EXEC ‘GET ““DATA_INT:INTERNAL”“’”
450   OUTPUT Na;”*OPC?”
460   ENTER Na;Opc
470   !
480   ! Read the analyzer’s event status register and
490   ! check for any errors when loading file.
500   !
510   OUTPUT Na;”*ESR?”
520   ENTER Na;Esr
530   IF Esr>0 THEN
540     BEEP
550     DISP “Error occurred while loading ‘DATA_INT’ . . . Program stopped.”
560     STOP
570   END IF
```

```
580    !
590    ! Determine the HP-IB address of the controller
600    ! and set the pass control back address.
610    !
620    INTEGER Stat,Addr
630    STATUS 7,3;Stat
640    Addr=BINAND(Stat,31)
650    OUTPUT Na;"*PCB ";Addr
660    !
670    ! Send the command to run the DATA_INT program.
680    !
690    DISP "Running the program..."
700    OUTPUT Na;"PROG:STAT RUN"
710    !
720    ! Monitor the program's status.  When it has
730    ! paused, set the variable for the controller's
740    ! HP-IB address.
750    !
760    OUTPUT Na;"PROG:STAT?"
770    ENTER Na;Prog$
780    IF Prog$<>"PAUS" THEN GOTO 760
790    OUTPUT Na;"PROG:NUMB 'Host',";Scode*100+Addr
800    OUTPUT Na;"PROG:STAT CONT"
810    !
820    ! Pass control of the bus to the HP 8711.
830    !
840    PASS CONTROL Na
850    !
860    ! Wait until addressed to talk by the HP 8711,
870    ! then send the name of the disk to catalog.
880    !
890    OUTPUT Scode;":INTERNAL"
900    !
910    ! Wait until addressed to listen by the HP 8711,
920    ! then read the directory from the analyzer.
930    !
940    DISP "Reading data . . ."
950    ENTER Scode;Directory$(*)
960    !
970    ! Print the catalog to the controller's display.
980    !
990    FOR I=1 TO 100
1000     IF LEN(Directory$(I))>0 THEN PRINT Directory$(I)
1010   NEXT I
1020   !
1030   ! Try to return the HP 8711 to LOCAL control.
1040   ! If the analyzer is still the active controller
1050   ! an error will be generated and the program
1060   ! will loop until control of the bus is received.
1070   !
1080   ON ERROR GOTO 1090
1090   LOCAL Na
1100   DISP ""
1110   END
```

DATA_INT Example Program

```
10    !----------------------------------------------------
20    !
30    ! IBASIC program:  DATA_INT - Data transfer (internal)
40    !
50    ! This program demonstrates how to transfer data to
60    ! and from an external controller.  In this example a
70    ! catalog listing is transferred from the HP 8711 to
80    ! the external controller.  For more information look
90    ! at the program listing for ‘DATA_EXT’.
100   !
110   ! This IBASIC program is intended to run on the
120   ! HP 8711’s internal controller.
130   !
140   !----------------------------------------------------
150   !
160   ! Dimension an array to hold the catalog listing.
170   !
180   DIM Directory$(1:100)[85]
190   !
200   ! Pause the program and wait for the controller to
210   ! set the ‘Host’ variable with its’ HP-IB address.
220   ! The controller continues this program after the
230   ! variable has been passed.
240   !
250   Host=0
260   PAUSE
270   !
280   ! Address the external controller to talk, read
290   ! the device to catalog.  If the HP 8711 is not
300   ! active controller on the bus an error will occur
310   ! and the program will loop until control is
320   ! received.
330   !
340   ON ERROR GOTO 340
350   ENTER Host;Stor_dev$
360   OFF ERROR
370   !
380   ! Catalog the requested storage device into
390   ! the string array.
400   !
410   DISP “Reading catalog...”
420   CAT Stor_dev$ TO Directory$(*)
430   !
440   ! Address the external controller to listen,
450   ! send the catalog array to the controller.
460   !
470   DISP “Transferring data...”
480   OUTPUT Host;Directory$(*)
490   !
500   ! Pass control back to the external controller.
510   !
520   PASS CONTROL Host
530   DISP “DONE”
540   END
```

Transferring Files over the GPIB

Two example programs demonstrate how to transfer files from the analyzer's mass memory to and from mass memory of an external controller via GPIB. Instrument states and program files may be transferred to or from the analyzer’s internal non-volatile memory, (`MEM:`), internal-volatile memory, (`RAM:`), and the internal 3.5” disk drive, (`INT:`).

This can be a convenient method to archive data and programs to a central large mass storage hard drive.

To run these programs, connect an external controller to the analyzer with a GPIB cable.

GETFILE	This program transfers a file from the analyzer to an external controller.
PUTFILE	This program transfers a file from an external controller to the analyzer.

GETFILE Example Program

This program transfers files from the analyzer to an external BASIC controller. The program runs on an external BASIC controller. The program prompts you to specify which analyzer program to transfer, the mass storage unit `MEM:` internal non-volatile memory, `RAM:` internal volatile memory, or `INT:` internal 3.5” disk drive, and the name of the file to be created on your external controller mass storage. GETFILE transfers instrument state files or program files.

```
1000 !GETFILE
1010 !
1020 !  This program will get files from 871X specified mass storage to a host
1030 !  mass storage.  The user specifies the mass storage unit, the filename
1040 !  of the 871X and the file on the host controller to be created.
1050 !
1060 !
1070 DIM Blk$(1:10)[32000]  ! Max file size = 10 * 32000 = 320000 bytes
1080 !
1090 DIM Filename$[15],Mass$[15],Dest$[15]
1100 INTEGER Dig_cnt
1110 !
1120 COM /Sys_state/ @Hp87xx,Scode
1130 ! Identify I/O Port
1140 CALL Iden_port
1150 !
1160 BEEP
1170 Mass$=”INT”
1180 Dest$=”File871X”
1190 INPUT “Enter the name of the 871X file to get.”,Filename$
1200 INPUT “Enter 871X Mass Storage (mem,INT,ram)”,Mass$
1210 INPUT “Enter host filename (default=’File871X’)”,Dest$
1220 DISP “READING FILE “&Mass$&”:”&Filename$&” ...”
1230 OUTPUT @Hp87xx;”MMEM:TRANSFER? ‘”&Mass$&”:”&Filename$&”’”
1240 ENTER @Hp87xx USING “%,A,D”;A$,Dig_cnt
1250 ENTER @Hp87xx USING “%,-K”;Blk$(*)
1260 FOR I=1 TO 6
1270     Filelength=LEN(Blk$(I))+Filelength
1280 NEXT I
1290 BEEP
1300 PRINT “Length”,Filelength
1310 DISP “Creating new file...”
1320 ON ERROR GOTO Save_file
1330 PURGE Dest$
1340 Save_file:     !
1350 OFF ERROR
1360 CREATE Dest$,Filelength
1370 ASSIGN @File TO Dest$;FORMAT ON
1380 OUTPUT @File;Blk$(*);
1390 ASSIGN @File TO *
1400 DISP “File “&Dest$&” created.”
1410 BEEP
1420 END
1430 !
1440 !***************************************************************
1450 ! Iden_port:   Identify io port to use
```

```
1460 ! Description: This routines sets up the I/O port address for
1470 !              the SCPI interface.  For “HP 87xx” instruments,
1480 !              the address assigned to @Hp87xx = 800 otherwise,
1490 !              716.
1500 !**************************************************************
1510 SUB Iden_port
1520     COM /Sys_state/ @Hp87xx,Scode
1530 !
1540     IF POS(SYSTEM$(“SYSTEM ID”),”HP 87”)<>0 THEN
1550         ASSIGN @Hp87xx TO 800
1560         Scode=8
1570     ELSE
1580         ASSIGN @Hp87xx TO 716
1590         Scode=7
1600     END IF
1610 !
1620 SUBEND !Iden_port
1630 !
```

PUTFILE Example Program

This program transfers files from the BASIC mass storage to the analyzer. Run this program on an external BASIC controller. The program will prompt you to specify the file to transfer and where to transfer the file. BDATA or ASCII files may be transferred to the analyzer's internal non-volatile memory (`MEM:`), the internal volatile memory (`RAM:`), or the internal built in 3.5” disk drive (`INT:`).

```
1000 !  PUTFILE
1010 !
1020 !  This program will transfer files from RMB mass mem to the specified
1030 !  871X mass storage.  The user specifies the 871X mass storage unit,
1040 !  the 871X file to be created, file type, and file to be transferred.
1050 !
1060 !
1070 DIM A$(1:4)[32000]
1080 DIM Filename$[15],Mass$[15],Source$[15]
1090 INTEGER Word1
1100 !
1110 COM /Sys_state/ @Hp87xx,Scode
1120 ! Identify I/O Port
1130 CALL Iden_port
1140 !
1150 Bdat$="n"
1160 BEEP
1170 Mass$="INT"
1180 INPUT "Enter the name of the 871X file to create",Filename$
1190 INPUT "File type BDAT? (y,n) [n]",Bdat$
1200 INPUT "Enter the 871X Mass Storage (mem,INT,ram)",Mass$
1210 INPUT "Enter source filename",Source$
1220 DISP "READING FILE "&Source$&" ..."
1230 ASSIGN @File TO Source$;FORMAT OFF
1240 ENTER @File USING "%,-K";A$(*)
1250 ASSIGN @File TO *
1260 !PRINT A$
1270 BEEP
1280 Length=0
1290 FOR I=1 TO 4
1300     Length=LEN(A$(I))+Length
1310 NEXT I
1320 DISP "TRANSFERRING FILE = ",Length
1330 IF Bdat$="y" OR Bdat$="Y" THEN
1340     IF Length<10 THEN
1350         Blk$="1"&VAL$(Length)
1360     ELSE
1370         IF Length<100 THEN
1380             Blk$="2"&VAL$(Length)
1390         ELSE
1400             IF Length<1000 THEN
1410                 Blk$="3"&VAL$(Length)
1420             ELSE
1430                 IF Length<10000 THEN
1440                     Blk$="4"&VAL$(Length)
1450                 ELSE
1460                     IF Length<100000 THEN
1470                         Blk$="5"&VAL$(Length)
```

```
1480                      ELSE
1490                          Blk$="6"&VAL$(Length)
1500                      END IF
1510                  END IF
1520              END IF
1530          END IF
1540      END IF
1550      OUTPUT @Hp87xx;"MMEM:TRANSFER:BDAT
          '"&Mass$&":"&Filename$&"',#"&Blk$;
1560 ELSE
1570      OUTPUT @Hp87xx;"MMEM:TRANSFER '"&Mass$&":"&Filename$&"',#0";
1580 END IF
1590 OUTPUT @Hp87xx;A$(*);END
1600 DISP "871X file "&Mass$&":"&Filename$&" created."
1610 BEEP
1620 END
1630 !
1640 !***************************************************************
1650 ! Iden_port:   Identify io port to use.
1660 ! Description: This routines sets up the I/O port address for
1670 !              the SCPI interface.  For "HP 87xx" instruments,
1680 !              the address assigned to @Hp87xx = 800 otherwise,
1690 !              716.
1700 !***************************************************************
1710 SUB Iden_port
1720      COM /Sys_state/ @Hp87xx,Scode
1730 !
1740      IF POS(SYSTEM$("SYSTEM ID"),"HP 87")<>0 THEN
1750          ASSIGN @Hp87xx TO 800
1760          Scode=8
1770      ELSE
1780          ASSIGN @Hp87xx TO 716
1790          Scode=7
1800      END IF
1810 !
1820 SUBEND !Iden_port
1830 !
```

Using TTL Input and Output

TTL_IO Example Program

This program continuously reads the user TTL port and reports the number of times the port has detected a closure (short) via an external switch. This program is useful in a production environment where a device must be properly connected, either manually or by automated means, where the analyzer must wait for a signal from the operator that the DUT is in place and is ready to be tested.

This program reads the user TTL port continuously until a short (0) is detected. Once this has been detected, a message is displayed. It then waits for the switch to open (1) and displays another message. At this point, code can be added to take a sweep and measure the DUT. The total number of cycles is counted and is displayed.

```
1000 ! Filename: TTL_IO
1010 !
1020 ! This program reads the USER TTL IO
1030 ! port, and counts how many times a
1040 ! switch connected to the port is pressed.
1050 !
1060 DIM Msg$[200]
1070 INTEGER X
1080 !
1090 !
1100 COM /Sys_state/ @Hp87xx,Scode
1110 ! Identify I/O Port
1120 CALL Iden_port
1130 !
1140 !
1150 Pass_count=0
1160 Start: !
1170 LOOP
1180 ! Display message
1190     Msg$="'DUTs passed: "&VAL$(Pass_count)&CHR$(10)
1200     Msg$=Msg$&"Press button to measure next DUT.'"
1210     OUTPUT @Hp87xx;"DISP:ANN:MESS ";Msg$
1220 !
1230 ! Wait for button to be pressed
1240     REPEAT
1250         OUTPUT @Hp87xx;"DIAG:PORT:READ? 15,1"
1260         ENTER @Hp87xx;X
1270     UNTIL X=0
1280     DISP "Button is now pressed."
1290     OUTPUT @Hp87xx;"DISP:ANN:MESS:CLEAR"
1300 !
1310 ! Wait for button to be released
1320     REPEAT
```

```
1330         OUTPUT @Hp87xx;"DIAG:PORT:READ? 15,1"
1340         ENTER @Hp87xx;X
1350     UNTIL X=1
1360     DISP "Button is now released."
1370 !
1380     OUTPUT @Hp87xx;"DISP:ANN:MESS 'Measuring...'"
1390 ! Add code here to take sweep
1400 ! and measure DUT.
1410     WAIT 1
1420     Pass_count=Pass_count+1
1430 END LOOP
1440 END
1450 !
1460 !**************************************************************
1470 ! Iden_port:   Identify io port to use.
1480 ! Description: This routines sets up the I/O port address for
1490 !              the SCPI interface.  For "HP 87xx" instruments,
1500 !              the address assigned to @Hp87xx = 800 otherwise,
1510 !              716.
1520 !**************************************************************

1530 SUB Iden_port
1540     COM /Sys_state/ @Hp87xx,Scode
1550 !
1560     IF POS(SYSTEM$("SYSTEM ID"),"HP 87")<>0 THEN
1570         ASSIGN @Hp87xx TO 800
1580         Scode=8
1590     ELSE
1600         ASSIGN @Hp87xx TO 716
1610         Scode=7
1620     END IF
1630 !
1640 SUBEND !Iden_port
1650 !
```

Creating User Interface Items

USERBEG	This program creates **User BEGIN** softkeys which allow the user to save or recall one of two instrument states, set the marker to maximum, set the scale/div, and compute some measurement statistics at the marker.
USERBEG1	The default User BEGIN program is created automatically when there is no IBASIC program installed. In this default program, softkey 3 is defined to be the marker-to-max function; softkey 4 prompts the user for a title, and also enables the clock. You may edit this program to change the functions you need.
USERBEG2	This program demonstrates the fast recall of previously defined instrument states. The instrument states SETUP1, SETUP2, and SETUP3 must have been previously saved to the analyzer's internal non-volatile RAM disk. Load the program into the analyzer, and press the BEGIN key. Enable **User BEGIN** by pressing **User BEGIN ON**. When **User BEGIN** is enabled, the **BEGIN** softkeys will appear. To save or recall each setup, select the appropriate softkey.
USER_BIT	This program demonstrates how to read and write to the USER bit. The USER bit is a TTL signal accessible by a BNC connector on the analyzer's rear panel. IBASIC's graphics commands are used to draw the USER bit value to the display.
USERKEYS	This program provides an example of how the analyzer's softkeys can be customized. The example demonstrates how to set up six instrument states, store them to the analyzer's internal memory, and set up two interactive softkey menus to choose between them.

USERBEG Example Program

```
0 !Filename:  USERBEG
20 !
70    !
80    ! Description:  Program to set up the User Begin softkeys.
90    !
100   !  A) This program creates User Begin softkeys which allow
110   !      the user to:  Save or Recall one of two instrument
120   !      states, set the marker to maximum, set the scale/div,
130   !      and compute some measurement statistics at the marker.
140   !
150   !  B) In order to run this program, do the following -
160   !    1) Load this program into the 871x
170   !    2) Press the “BEGIN” (hardkey) and the “User Begin on/OFF” (softkey).
180   !    3) The “User Begin” function is now enabled (which runs this
190   !      program).  This program re-defines the softkeys displayed
200   !      whenever the BEGIN hardkey is pressed.  The functions
210   !      performed by these softkeys are defined by this
220   !      program.  Note that all front panel keys in the analyzer are
230   !      active (as if there were no program running).
240   !    4) Use the instrument as you normally would.  However, when
250   !      the BEGIN hardkey is pressed, the menu defined
260   !      by this program will be displayed instead of the usual
270   !      BEGIN softkeys, until the “User Begin ON/off” (softkey)
280   !      is pressed, turning off the “User Begin” mode.
290   !
300   !*******************************************************
310   ! Initialize
320   !
330 User_begin:ASSIGN @Hp871x TO 800    !REQUIRED - first line for User Begin program
340   !
350   REAL Vert_scale,Mrkr_data(1:30),Mrkr_mean,Mrkr_sdev
360   REAL Mrkr_max,Mrkr_min,I
370   DIM Message$[124]
380   !
390   !-------------------------------------------------------
400   ! Write the softkey labels.  Maximum label length=20characters
410   !
420   OUTPUT @Hp871x;”DISP:MENU2:KEY8 ‘’;*WAI”     !clear all labels
430   OUTPUT @Hp871x;”DISP:MENU2:KEY1 ‘      Save   State 1’;*WAI”
440   OUTPUT @Hp871x;”DISP:MENU2:KEY2 ‘    Recall   State 1’;*WAI”
450   OUTPUT @Hp871x;”DISP:MENU2:KEY3 ‘      Save   State 2’;*WAI”
460   OUTPUT @Hp871x;”DISP:MENU2:KEY4 ‘    Recall   State 2’;*WAI”
470   OUTPUT @Hp871x;”DISP:MENU2:KEY5 ‘Mkr -> Max’;*WAI”
480   OUTPUT @Hp871x;”DISP:MENU2:KEY6 ‘Scale/Div’;*WAI”
490   OUTPUT @Hp871x;”DISP:MENU2:KEY7 ‘    MarkerStatistics’;*WAI”
500   !
510 User_pause:PAUSE    !pause the program until a softkey is pressed
520   GOTO User_pause   !return to program pause after a softkey press
530   !
540   !--------------------------------------------------------
550   ! Define softkey routines
560   !
570 User_key1:           ! Define softkey 1:  save state 1
580   OUTPUT @Hp871x;”MMEM:STOR:STAT 1,’MEM:UBEGN1.STA’”  !save state 1
590   GOTO User_pause    !return to softkey loop
600   !
610 User_key2:           ! Define softkey 2:  recall state 1
```

```
620   OUTPUT @Hp871x;"MMEM:LOAD:STAT 1,'MEM:UBEGN1.STA'"  !recall state 1
630   GOTO User_pause    !return to softkey loop
640   !
650 User_key3:           ! Define softkey 3:  save state 2
660   OUTPUT @Hp871x;"MMEM:STOR:STAT 1,'MEM:UBEGN2.STA'"  !save state 2
670   GOTO User_pause    !return to softkey loop
680   !
690 User_key4:           ! Define softkey 4:  recall state 2
700   OUTPUT @Hp871x;"MMEM:LOAD:STAT 1,'MEM:UBEGN2.STA'"  !recall state 2
710   GOTO User_pause    !return to softkey loop
720   !
730 User_key5:      ! Define softkey 5:  set marker to max
740   OUTPUT @Hp871x;"CALC1:MARK:FUNC MAX"      !marker -> max
750   GOTO User_pause
760   !
770 User_key6:      ! Define softkey 6:  adjust the scale, dB/Div, of the trace
780   INPUT "Enter the scale (dB/Div)",Vert_scale  !ask user for scale
790   OUTPUT @Hp871x;"DISP:WIND1:TRAC:Y:PDIV "&VAL$(Vert_scale)  !set the scale
800   GOTO User_pause
810   !
820 User_key7:      ! Define softkey 7:  compute statistics for marker.
830   OUTPUT @Hp871x;"DISP:ANN:MESS:DATA 'Computing marker statistics...'"
840   OUTPUT @Hp871x;"CALC1:MARK1 ON"       !ensure marker is on
850   FOR I=1 TO 30                         !read marker 30 times
860     OUTPUT @Hp871x;"CALC1:MARK1:Y?"     !get marker reading
870     ENTER @Hp871x;Mrkr_data(I)
880   NEXT I
890   Mrkr_mean=SUM(Mrkr_data)/30           !compute mean
900   !
910   Mrkr_sdev=0                           !initialize standard deviation
920   Mrkr_min=Mrkr_data(1)                 !initialize min
930   Mrkr_max=Mrkr_data(1)                 !initialize max
940   FOR I=1 TO 30                         !compute std dev, min, max
950     Mrkr_sdev=Mrkr_sdev+(Mrkr_data(I)-Mrkr_mean)^2  !sum squares of deviation
960     Mrkr_min=MIN(Mrkr_min,Mrkr_data(I))             !find min
970     Mrkr_max=MAX(Mrkr_max,Mrkr_data(I))             !find max
980   NEXT I
990   Mrkr_sdev=SQRT(Mrkr_sdev/29)          !finish computation of std dev
1000  !
1010  Message$="Marker Statistics:"&CHR$(10)                !1st line of message
1020  Message$=Message$&"  Mean ="&VAL$(Mrkr_mean)&CHR$(10)  !2nd line of message
1030  Message$=Message$&"  Min ="&VAL$(Mrkr_min)&CHR$(10)    !3rd line of message
1040  Message$=Message$&"  Max ="&VAL$(Mrkr_max)&CHR$(10)    !4th line of message
1050  Message$=Message$&"  Standard Deviation = "&VAL$(Mrkr_sdev) !5th line of
message
1060  OUTPUT @Hp871x;"DISP:ANN:MESS:DATA '"&Message$&"', MEDIUM" !display message
1070  GOTO User_pause                       !return to softkey loop
1080  !
1090  END
```

USERBEG1 Example Program

```
10    ! ----------------------------------------------------------
20    !
30    ! BASIC program: USERBEG1
40    !
50    ! This is the default User Defined BEGIN program.  This program
60    ! will automatically install if the [User BEGIN] key is
70    ! selected, and a program has not been previously loaded.
80    !
90    ! The following line is required. DO NOT REMOVE!
100 User_begin:ASSIGN @Rfna TO 800    ![User Begin] Program
110   !
120   ! To Modify:
130   ! Use [IBASIC][EDIT] or [IBASIC][Key Record]
140   !
150   !
160   ! Delclare storage for variables.
170   DIM Name$[60],Str1$[60],Str2$[60],Str3$[60]
180   !
190   ! Clear the softkey labels
200   OUTPUT @Rfna;"DISP:MENU2:KEY8 '';*WAI"
210   !
220   ! Re-define softkey labels here.
230   OUTPUT @Rfna;"DISP:MENU2:KEY1 '*';*WAI"
240   OUTPUT @Rfna;"DISP:MENU2:KEY2 '*';*WAI"
250   OUTPUT @Rfna;"DISP:MENU2:KEY3 'Mkr -> Max';*WAI"
260   OUTPUT @Rfna;"DISP:MENU2:KEY4 'Title and Clock';*WAI"
270   OUTPUT @Rfna;"DISP:MENU2:KEY5 '*';*WAI"
280   OUTPUT @Rfna;"DISP:MENU2:KEY6 '*';*WAI"
290   OUTPUT @Rfna;"DISP:MENU2:KEY7 '*';*WAI"
300   !
310   !The following 2 lines are required. DO NOT REMOVE!
320 User_pause:PAUSE
330   GOTO User_pause
340   !
350 User_key1:      ! Define softkey 1 here.
360   GOSUB Message ! Remove this line.
370   GOTO User_pause
380   !
390 User_key2:      ! Define softkey 2 here.
400   GOSUB Message ! Remove this line
410   GOTO User_pause
420   !
430 User_key3:      ! Example Marker Function
440   OUTPUT @Rfna;"CALC1:MARK1 ON"
450   OUTPUT @Rfna;"CALC1:MARK:FUNC MAX"
460   GOTO User_pause
470   !
480 User_key4:      ! Example Title Entry
490   INPUT "Enter Title Line 1.  Press [Enter] when done.",Name$
500   OUTPUT @Rfna;"DISP:ANN:TITL1:DATA '"&Name$&"'"
510   OUTPUT @Rfna;"DISP:ANN:TITL ON"
520   GOTO User_pause
530   !
540 User_key5:      ! Define softkey 5 here.
550   GOSUB Message ! Remove this line.
560   GOTO User_pause
570   !
```

```
580 User_key6:      ! Define softkey 6 here.
590   GOSUB Message ! Remove this line.
600   GOTO User_pause
610   !
620 User_key7:      ! Define softkey 7 here.
630   GOSUB Message ! Remove this line.
640   GOTO User_pause
650   !
660 Message:   !
670   Str1$=”This key is programmable.”
680   Str2$=”To modify, select”
690   Str3$=”[System Options], [IBASIC], [Edit].”
700   OUTPUT @Rfna;”DISP:ANN:MESS ‘”&Str1$&CHR$(10)&Str2$&CHR$(10)&Str3$&”’, MEDIUM”
710   RETURN
720   !
730   END
```

USERBEG2 Example Program

```
10    ! -----------------------------------------------------------
20    !
30    ! BASIC program: USERBEG2
40    !
50    ! This is an example User Defined BEGIN program.  This program
60    ! will recall the named file.  Demonstrates fast recall
70    ! of a previously defined files SETUP1, SETUP2, and SETUP3.
80    !
90    ! The following line is required. DO NOT REMOVE!
100 User_begin:ASSIGN @Rfna TO 800    ![User Begin] Program
110   !
120   ! To Modify:
130   ! Use [IBASIC][EDIT] or [IBASIC][Key Record]
140   !
150   !
160   ! Delclare storage for variables.
170   DIM Name$[60],Str1$[60],Str2$[60],Str3$[60]
180   !
190   ! Clear the softkey labels
200   OUTPUT @Rfna;"DISP:MENU2:KEY8 '';*WAI"
210   !
220   ! Re-define softkey labels here.
230   OUTPUT @Rfna;"DISP:MENU2:KEY1 'Setup1';*WAI"
240   OUTPUT @Rfna;"DISP:MENU2:KEY2 'Setup2';*WAI"
250   OUTPUT @Rfna;"DISP:MENU2:KEY3 'Setup3';*WAI"
260   OUTPUT @Rfna;"DISP:MENU2:KEY4 '*';*WAI"
270   OUTPUT @Rfna;"DISP:MENU2:KEY5 '*';*WAI"
280   OUTPUT @Rfna;"DISP:MENU2:KEY6 '*';*WAI"
290   OUTPUT @Rfna;"DISP:MENU2:KEY7 '*';*WAI"
300   !
310   !The following 2 lines are required. DO NOT REMOVE!
320 User_pause:PAUSE
330   GOTO User_pause
340   !
350 User_key1:      ! Define softkey 1 here.
360   OUTPUT @Rfna;"MMEM:LOAD:STAT 1,'MEM:SETUP1'"
370   GOTO User_pause
380   !
390 User_key2:      ! Define softkey 2 here.
400   OUTPUT @Rfna;"MMEM:LOAD:STAT 1,'MEM:SETUP2'"
410   GOTO User_pause
420   !
430 User_key3:      ! Example Marker Function
440   OUTPUT @Rfna;"MMEM:LOAD:STAT 1,'MEM:SETUP3'"
450   GOTO User_pause
460   !
470 User_key4:      ! Example Title Entry
480   GOTO User_pause
490   !
500 User_key5:      ! Define softkey 5 here.
510   GOTO User_pause
520   !
530 User_key6:      ! Define softkey 6 here.
540   GOTO User_pause
550   !
560 User_key7:      ! Define softkey 7 here.
```

```
570   GOTO User_pause
580   !
590   END
```

USER_BIT Example Program

```
10    !-------------------------------------------------------
20    !
30    ! IBASIC program:  USER_BIT - Using the USER bit
40    !
50    ! This program reads and writes to the USER bit.
60    ! IBASIC’s graphics commands are used to draw the
70    ! USER bit value to the display.
80    !
90    !-------------------------------------------------------
100   !
110   ! Assign an I/O path name to the internal bus and
120   ! initialize variables.
130   !
140   ASSIGN @Rfna TO 800
150   INTEGER Beeper,Count
160   Count=0
170   Beeper=0
180   !
190   ! Preset the analyzer, setup measurement and display
200   ! parameters for a measurement and put the analyzer
210   ! in Trigger HOLD mode.
220   !
230   OUTPUT @Rfna;”SYST:PRES;*WAI”
240   OUTPUT @Rfna;”DISP:ANN:FREQ1:MODE SSTOP”
250   OUTPUT @Rfna;”SENS1:FREQ:STAR 100 MHz;STOP 400 MHz;*WAI”
260   OUTPUT @Rfna;”DISP:WIND1:TRAC:Y:PDIV 20 dB;RLEV -60 dB;RPOS 5”
270   OUTPUT @Rfna;”SENS1:SWE:POIN 101;TIME .1 s;*WAI”
280   OUTPUT @Rfna;”ABOR;:INIT1:CONT OFF;*WAI”
290   !
300   ! Wait for all the setup operations to be complete
310   ! before continuing the program.
320   !
330   OUTPUT @Rfna;”*OPC?”
340   ENTER @Rfna;Opc
350   !
360   ! Allocate the lower display partition.
370   !
380   OUTPUT @Rfna;”DISP:PROG LOW”
390   !
400   ! Setup a softkey menu to enable and disable the
410   ! beeper.  Clear the analyzer’s input/output queues.
420   !
430   ON KEY 1 LABEL “Beep      Enable” GOSUB Beep_on
440   ON KEY 2 LABEL “Beep      Disable” GOSUB Beep_off
450   CLEAR @Rfna
460   !
470   ! Trigger 100 sweeps.  Beep (if the beeper flag is set)
480   ! and toggle the USER bit after each sweep.
490   !
500   DISP “USER bit example program. End of sweep toggles USER bit.”
510   PRINT “Draw the end of sweep USER bit value...”
520   MOVE 0,20
530   FOR I=1 TO 100
540     OUTPUT @Rfna;”INIT1;*OPC?”
550     ENTER @Rfna;Opc
560     GOSUB Toggle
570   NEXT I
```

```
580   DISP “End program”
590   STOP
600   !
610   ! The Odd flag’s value alternates between 1 and 0
620   ! depending on the number of sweeps that have been
630   ! taken.  It is the value that is written to the
640   ! USER bit.
650   !
660 Toggle: !
670   IF Odd=0 THEN
680     WRITEIO 15,1;0
690     Odd=1
700   ELSE
710     WRITEIO 15,1;1
720     Odd=0
730   END IF
740   IF Beeper=1 THEN
750     BEEP
760   END IF
770   !
780   ! Read the value of the USER bit and draw it to the
790   ! IBASIC display.
800   !
810   Val=READIO(15,1)
820   Val=Val*12
830   DRAW 6*(I-1),Val+20
840   DRAW 6*I,Val+20
850   RETURN
860   !
870   ! These two subroutines set a flag that is used
880   ! to turn on or off the beeper.
890   !
900 Beep_on: Beeper=1
910   RETURN
920   !
930 Beep_off: Beeper=0
940   RETURN
950   END
```

USERKEYS Example Program

```
10    !-----------------------------------------------------
20    !
30    ! IBASIC program:  USERKEYS - Customized softkeys
40    !
50    ! This program provides an example template for use
60    ! in customizing the HP 871X’s softkeys.  The example
70    ! demonstrates how to set up six instrument states,
80    ! store them to the analyzer’s internal memory, and
90    ! setup two interactive softkey menus to choose
100   ! between them.
110   !
120   !-----------------------------------------------------
130   !
140   ! Assign an I/O path name to the internal bus, preset
150   ! the analyzer, wait until the preset is complete,
160   ! turn on Trigger HOLD mode and set the display scale
170   ! and reference values.
180   !
190   ASSIGN @Rfna TO 800
200   OUTPUT @Rfna;”SYST:PRES;*OPC?”
210   ENTER @Rfna;Opc
220   OUTPUT @Rfna;”ABOR;:INIT1:CONT OFF;*WAI”
230   OUTPUT @Rfna;”DISP:WIND1:TRAC:Y:PDIV 20 dB;RLEV -60 dB;RPOS 5”
240   !
250   ! Setup six instrument states and store them to the
260   ! internal memory.
270   !
280   GOSUB Save_1
290   GOSUB Save_2
300   GOSUB Save_3
310   GOSUB Save_4
320   GOSUB Save_5
330   GOSUB Save_6
340   !
350   ! Setup the Main Menu keys.
360   !
370   GOSUB Menu_1
380   !
390   ! Wait until a softkey is pressed.
400   !
410 Suspend: !
420   WAIT 100000
430   GOTO Suspend
440   STOP
450   !
460   ! This subroutine sets up the softkey menus -
470   ! Menu1 sets up the main menu, Menu2 sets up
480   ! the second level menu.
490   !
500 Menu_1: BEEP
510   DISP “MAIN MENU”
520   ON KEY 1 LABEL “Setup #1” GOSUB Load_1
530   ON KEY 2 LABEL “Setup #2” GOSUB Load_2
540   ON KEY 3 LABEL “Setup #3” GOSUB Load_3
550   ON KEY 5 LABEL “Autoscale” GOSUB Autoscale
560   ON KEY 6 LABEL “ Next Menu” GOSUB Menu_2
570   RETURN
```

```
580   !
590 Menu_2:  BEEP
600   DISP “MORE MENU”
610   ON KEY 1 LABEL “Setup #4” GOSUB Load_4
620   ON KEY 2 LABEL “Setup #5” GOSUB Load_5
630   ON KEY 3 LABEL “Setup #6” GOSUB Load_6
640   ON KEY 5 LABEL “Autoscale” GOSUB Autoscale
650   ON KEY 6 LABEL “Prior Menu” GOSUB Menu_1
660   RETURN
670   !
680   ! This subroutine automatically sets the scale and
690   ! reference values of the display.
700   !
710 Autoscale: OUTPUT @Rfna;”DISP:WIND1:TRAC:Y:AUTO ONCE”
720   OUTPUT @Rfna;”DISP:WIND2:TRAC:Y:AUTO ONCE”
730   RETURN
740   !
750   ! These six subroutines each set up the analyzer to
760   ! make a different measurement and store that setup
770   ! to the instrument’s internal memory.
780   !
790 Save_1: OUTPUT @Rfna;”SENS1:STAT ON;*WAI”
800   OUTPUT @Rfna;”DISP:ANN:FREQ1:MODE SSTOP”
810   OUTPUT @Rfna;”SENS1:FREQ:STAR 100 MHz;STOP 400 MHz;*WAI”
820   OUTPUT @Rfna;”INIT1;*WAI”
830   OUTPUT @Rfna;”MMEM:STOR:STAT 1,’MEM:STATE1.STA’”
840   RETURN
850   !
860 Save_2: OUTPUT @Rfna;”SENS2:STAT ON;*WAI”
870   OUTPUT @Rfna;”SENS2:FUNC ‘XFR:POW:RAT 1,0’;DET NBAN;*WAI”
880   OUTPUT @Rfna;”DISP:ANN:FREQ2:MODE CSPAN”
890   OUTPUT @Rfna;”SENS2:FREQ:CENT 200 MHz;SPAN 300 MHz;*WAI”
900   OUTPUT @Rfna;”INIT2;*WAI”
910   OUTPUT @Rfna;”MMEM:STOR:STAT 1,’MEM:STATE2.STA’”
920   RETURN
930   !
940 Save_3: OUTPUT @Rfna;”CALC2:FORM SWR”
950   OUTPUT @Rfna;”INIT2;*WAI”
960   OUTPUT @Rfna;”MMEM:STOR:STAT 1,’MEM:STATE3.STA’”
970   OUTPUT @Rfna;”CALC2:FORM MLOG”
980   RETURN
990   !
1000 Save_4:OUTPUT @Rfna;”SENS2:STAT OFF”
1010  OUTPUT @Rfna;”SENS1:SWE:POIN 1601;*WAI”
1020  OUTPUT @Rfna;”INIT1;*WAI”
1030  OUTPUT @Rfna;”MMEM:STOR:STAT 1,’MEM:STATE4.STA’”
1040  RETURN
1050  !
1060 Save_5:OUTPUT @Rfna;”CALC1:MARK:BWID -3;FUNC:TRAC ON”
1070  OUTPUT @Rfna;”INIT1;*WAI”
1080  OUTPUT @Rfna;”MMEM:STOR:STAT 1,’MEM:STATE5.STA’”
1090  RETURN
1100  !
1110 Save_6:OUTPUT @Rfna;”SENS1:BWID 250 Hz;*WAI”
1120  OUTPUT @Rfna;”SENS1:SWE:POIN 101;*WAI”
1130  OUTPUT @Rfna;”INIT1;*WAI”
1140  OUTPUT @Rfna;”MMEM:STOR:STAT 1,’MEM:STATE6.STA’”
1150  RETURN
1160  !
1170  ! These six subroutines each recall one of the
```

```
1180  ! measurement setups that were stored earlier.
1190  !
1200 Load_1:DISP “Setup 1”
1210  OUTPUT @Rfna;”MMEM:LOAD:STAT 1,’MEM:STATE1.STA’;*WAI”
1220  OUTPUT @Rfna;”INIT1;*WAI”
1230  RETURN
1240  !
1250 Load_2:DISP “Setup 2”
1260  OUTPUT @Rfna;”MMEM:LOAD:STAT 1,’MEM:STATE2.STA’;*WAI”
1270  OUTPUT @Rfna;”INIT2;*WAI”
1280  RETURN
1290  !
1300 Load_3:DISP “Setup 3”
1310  OUTPUT @Rfna;”MMEM:LOAD:STAT 1,’MEM:STATE3.STA’;*WAI”
1320  OUTPUT @Rfna;”INIT2;*WAI”
1330  RETURN
1340  !
1350 Load_4:DISP “Setup 4”
1360  OUTPUT @Rfna;”MMEM:LOAD:STAT 1,’MEM:STATE4.STA’;*WAI”
1370  OUTPUT @Rfna;”INIT1;*WAI”
1380  RETURN
1390  !
1400 Load_5:DISP “Setup 5”
1410  OUTPUT @Rfna;”MMEM:LOAD:STAT 1,’MEM:STATE5.STA’;*WAI”
1420  OUTPUT @Rfna;”INIT1;*WAI”
1430  RETURN
1440  !
1450 Load_6:DISP “Setup 6”
1460  OUTPUT @Rfna;”MMEM:LOAD:STAT 1,’MEM:STATE6.STA’;*WAI”
1470  OUTPUT @Rfna;”INIT1;*WAI”
1480  RETURN
1490  END
```

Index

Index

Index