Programming Note

Agilent Technologies
Introductory Programming Guide
For the 8757D/E Scalar Network Analyzer
with the HP Vectra Personal Computer
Using Microsoft® QuickC 2.5

Agilent Technologies

Manufacturing Part Number: 08757-90118

Printed in USA

Print Date: July 1992

08757-90118

Hewlett-Packard to Agilent Technologies Transition

This manual may contain references to HP or Hewlett-Packard. Please note that Hewlett-Packard's former test and measurement, semiconductor products and chemical analysis businesses are now part of Agilent Technologies. To reduce potential confusion, the only change to product numbers and names has been in the company name prefix: where a product number/name was HP XXXX the current name/number is now Agilent XXXX. For example, model number HP8648 is now model number Agilent 8648.

Documentation Warranty

THE MATERIAL CONTAINED IN THIS DOCUMENT IS PROVIDED "AS IS," AND IS SUBJECT TO BEING CHANGED, WITHOUT NOTICE, IN FUTURE EDITIONS. FURTHER, TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, AGILENT DISCLAIMS ALL WARRANTIES, EITHER EXPRESS OR IMPLIED WITH REGARD TO THIS MANUAL AND ANY INFORMATION CONTAINED HEREIN, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. AGILENT SHALL NOT BE LIABLE FOR ERRORS OR FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH THE FURNISHING, USE, OR PERFORMANCE OF THIS DOCUMENT OR ANY INFORMATION CONTAINED HEREIN. SHOULD AGILENT AND THE USER HAVE A SEPARATE WRITTEN AGREEMENT WITH WARRANTY TERMS COVERING THE MATERIAL IN THIS DOCUMENT THAT CONFLICT WITH THESE TERMS, THE WARRANTY TERMS IN THE SEPARATE AGREEMENT WILL CONTROL.

DFARS/Restricted Rights Notice

If software is for use in the performance of a U.S. Government prime contract or subcontract, Software is delivered and licensed as “Commercial computer software” as defined in DFAR 252.227-7014 (June 1995), or as a “commercial item” as defined in FAR 2.101(a) or as “Restricted computer software” as defined in FAR 52.227-19 (June 1987) or any equivalent agency regulation or contract clause. Use, duplication or disclosure of Software is subject to Agilent Technologies’ standard commercial license terms, and non-DOD Departments and Agencies of the U.S. Government will receive no greater than Restricted Rights as defined in FAR 52.227-19(c)(1-2) (June 1987). U.S. Government users will receive no greater than Limited Rights as defined in FAR 52.227-14 (June 1987) or DFAR 252.227-7015 (b)(2) (November 1995), as applicable in any technical data.

Printing Copies of Documentation from the Web

To print copies of documentation from the Web, download the PDF file from the Agilent web site:

- Go to http://www.agilent.com.
- Enter the document's part number (located on the title page) in the **Quick Search** box.
- Click GO.
- Click on the hyperlink for the document.
- Click the printer icon located in the tool bar.

Contacting Agilent

This information supersedes all prior HP contact information.

Online assistance: www.agilent.com/find/assist

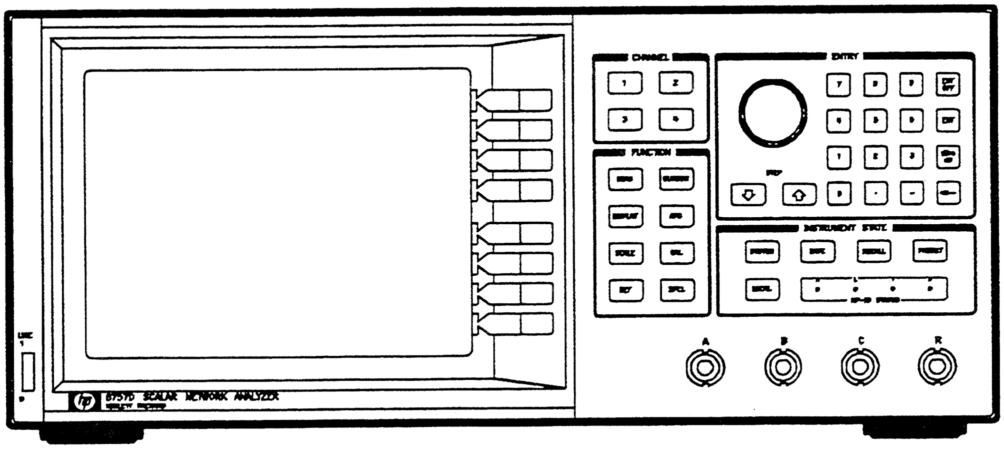
Americas			
Brazil *(tel)* (+55) 11 3351 7012 *(fax)* (+55) 11 3351 7024	**Canada** *(tel)* 888 447 7378 *(fax) 905 282 6495*	**Mexico** *(tel)* 1 800 254 2440 *(fax)* 1 800 254 4222	**United States** *(tel)* 800 829 4444 *(alt)* (+1) 303 662 3998 *(fax)* 800 829 4433
Asia Pacific and Japan			
Australia (*tel*) 1 800 225 574 (*fax*) 1 800 681 776 (*fax*) 1 800 225 539	**China** (*tel*) 800 810 0508 (*alt*) 800 810 0510 (*fax*) 800 810 0507 (*fax*) 800 810 0362	**Hong Kong** (*tel*) 800 933 229 (*fax*) 800 900 701	**India** (*tel*) 1600 112 626 (*fax*) 1600 112 727 (*fax*) 1600 113 040
Japan (Bench) (*tel*) 0120 32 0119 (*alt*) (+81) 426 56 7799 (*fax*) 0120 01 2144	**Japan (On-Site)** (*tel*) 0120 802 363 (*alt*) (+81) 426 56 7498 (*fax*) (+81) 426 60 8953	**Singapore** (*tel*) 1 800 275 0880 (*fax*) (+65) 6755 1235 (*fax*) (+65) 6755 1214	**South Korea** (*tel*) 080 778 0011 (*fax*) 080 778 0013
Taiwan (*tel*) 0800 047 669 (*fax*) 0800 047 667 (*fax*) 886 3492 0779	**Thailand** (*tel*) 1 800 2758 5822 (*alt*) (+66) 2267 5913 (*fax*) 1 800 656 336	**Malaysia** (*tel*) 1800 880 399 (*fax*) 1800 801 054	
Europe			
Austria (*tel*) 0820 87 44 11* (*fax*) 0820 87 44 22	**Belgium** (*tel*) (+32) (0)2 404 9340 (*alt*) (+32) (0)2 404 9000 (*fax*) (+32) (0)2 404 9395	**Denmark** (*tel*) (+45) 7013 1515 (*alt*) (+45) 7013 7313 (*fax*) (+45) 7013 1555	**Finland** (*tel*) (+358) 10 855 2100 (*fax*) (+358) (0) 10 855 2923
France (*tel*) 0825 010 700* (*alt*) (+33) (0)1 6453 5623 (*fax*) 0825 010 701*	**Germany** (*tel*) 01805 24 6333* (*alt*) 01805 24 6330* (*fax*) 01805 24 6336*	**Ireland** (*tel*) (+353) (0)1 890 924 204 (*alt*) (+353) (0)1 890 924 206 (*fax*)(+353) (0)1 890 924 024	**Israel** (*tel*) (+972) 3 9288 500 (*fax*) (+972) 3 9288 501
Italy (*tel*) (+39) (0)2 9260 8484 (*fax*) (+39) (0)2 9544 1175	**Luxemburg** (*tel*) (+32) (0)2 404 9340 (*alt*) (+32) (0)2 404 9000 (*fax*) (+32) (0)2 404 9395	**Netherlands** (*tel*) (+31) (0)20 547 2111 (*alt*) (+31) (0)20 547 2000 (*fax*) (+31) (0)20 547 2190	**Russia** (*tel*) (+7) 095 797 3963 (*alt*) (+7) 095 797 3900 (*fax*) (+7) 095 797 3901
Spain (*tel*) (+34) 91 631 3300 (*alt*) (+34) 91 631 3000 (*fax*) (+34) 91 631 3301	**Sweden** (*tel*) 0200 88 22 55* (*alt*) (+46) (0)8 5064 8686 (*fax*) 020 120 2266*	**Switzerland (French)** (*tel*) 0800 80 5353 opt. 2* (*alt*) (+33) (0)1 6453 5623 (*fax*) (+41) (0)22 567 5313	**Switzerland (German)** (*tel*) 0800 80 5353 opt. 1* (*alt*) (+49) (0)7031 464 6333 (*fax*) (+41) (0)1 272 7373
Switzerland (Italian) (*tel*) 0800 80 5353 opt. 3* (*alt*) (+39) (0)2 9260 8484 (*fax*) (+41) (0)22 567 5314	**United Kingdom** (*tel*) (+44) (0)7004 666666 (*alt*) (+44) (0)7004 123123 (*fax*) (+44) (0)7004 444555		

(tel) = primary telephone number; *(alt)* = alternate telephone number; *(fax)* = FAX number; * = in country number 8/03/04

HP–IB Programming Note

Introductory Programming Guide

for the HP 8757D/E Scalar Network Analyzer with the
HP Vectra Personal Computer using Microsoft® QuickC 2.5



Introduction

This programming note describes the remote operation of the HP 8757D/E Scalar Network Analyzer with the HP Vectra Personal Computer (or IBM compatible) using the HP 82335A HP–IB Command Library and Microsoft QuickC 2.5. Included in this guide are several short programs that demonstrate the use of the HP 8757D/E with HP–IB commands, and a diagram of system connections for remote control.

The HP 8757D/E is a fully programmable analyzer capable of making magnitude–only transmission and reflection measurements over an RF and microwave frequency range of 10 MHz to 110 GHz. When used with an HP–IB computer, the analyzer's front panel may be remotely controlled, along with most softkey functions and some functions accessible only via HP–IB. The analyzer exerts control over a source (HP 8350B, 8340B/41B, or 8360), digital plotter (HP 7440A or 7550A/B), and printer (HP 2225A ThinkJet, 3630A PaintJet, or 2225B QuietJet Plus) connected to the 8757 SYSTEM INTERFACE.

This note assumes you are familiar with local (non–remote) operation of the HP 8757D/E. If not, refer to the operating manual. You should also be familiar with the HP Vectra Personal Computer (or compatible), particularly HP–IB operation using the HP 82335A HP–IB Command Library.

Sample programs included in this guide are:

- Program 1: Remote, Local, and Local Lockout.
- Program 2: Controlling the Front Panel.
- Program 3: Passthru Mode.
- Program 4: Cursor Operations.
- Program 5: Read a Single Value.
- Program 6: Trace Transfer.
- Program 7: Using the TAKE SWEEP Command.
- Program 8: Programming the Softkeys.
- Program 9: CRT Graphics.
- Program10: Learning the Instrument State.
- Program11: Guided Instrument Setup with CRT Graphics.

Reference information

The following texts provide additional information on the HP Interface Bus, the analyzer, the source, or the HP Vectra Personal Computer.

HP 8757D/E literature:

- *HP 8757D Operating Manual*
- *HP 8757D/E Operating Manual.*
- Programming Note: *Quick Reference Guide for the HP 8757D/E Scalar Network Analyzer.*

Source literature:

- Programming Note: *Quick Reference Guide for the HP 8350B Sweep Oscillator.*
- Programming Note: *Quick Reference Guide for the HP 8340B Synthesized Sweeper.*
- *HP 8360 Operating and Programming Reference Manual.*

HP Vectra Personal Computer literature:

- *HP 82335A HP–IB Command Library Manual.*
- *Microsoft QuickC: Up and Running.*
- *Microsoft QuickC: Tool Kit.*
- *C for Yourself.*

Equipment required

1 HP 8757D/E Scalar Network Analyzer.

1 HP 8350B Sweeper with plug–in or
HP 8340B/41B Synthesized Sweeper or
HP 8360 Series Synthesized Sweeper.

1 HP Vectra Personal Computer (or compatible) with Microsoft QuickC 2.5, HP 82335A HP–IB Interface Card, MS–DOS 3.3 or higher, and at least 512K bytes of memory.

1 HP 85027A/B/C/D/E Directional Bridge.

1 HP 11664A/E Detector or HP 85025A/B/D/E Detector. or HP 85037A/B Precision Detector with connector type to match bridge and test device.

1 Shielded open circuit with connector to mate with bridge.

1 Short circuit with connector to mate with bridge.

3 HP 11170C BNC cables,122 cm. (48 inches). (4 are needed with HP 8340B/41B).

2 HP 10833A/B/C/D HP–IB cables.

1 Test device.

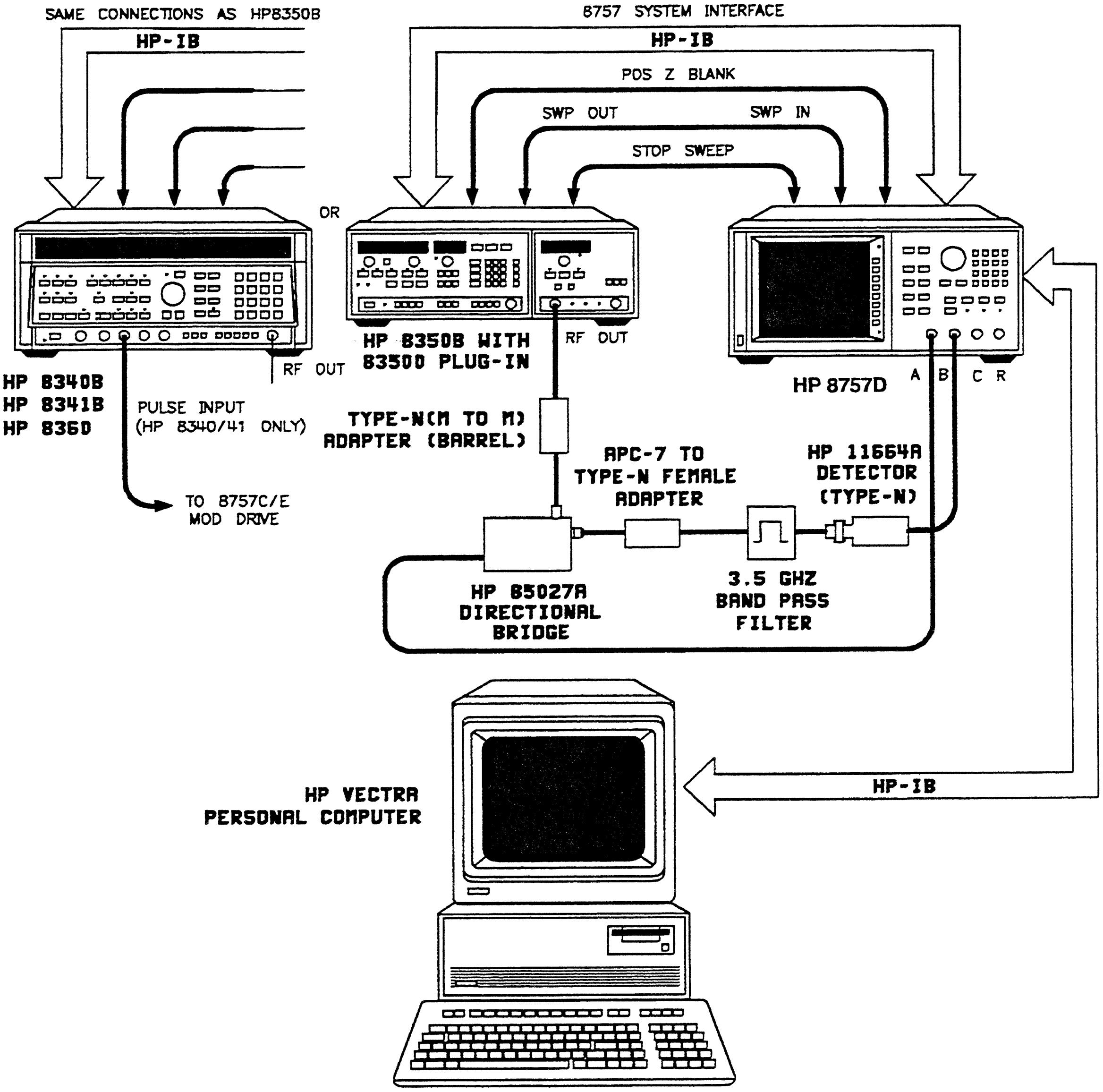


Figure 1. System Connections

Set–up

Connect the instruments as shown in Figure 1. The following procedure sets the HP–IB addresses of the instruments to operate properly with the programs contained in this guide. If the HP 82335A HP–IB interface card is not installed in the HP Vectra PC, follow the instructions in the *HP 82335A HP–IB Command Library Manual* for installation. Before installation, set the interface select code to 7.

1. Turn on the HP 8350B Sweeper. Press **[SHIFT] [LCL]**. The FREQUENCY/TIME display shows the current HP–IB address of the source. If it is not 19, press **[1] [9] [GHz]**. The HP 8340B or 8341B Synthesized Sweeper operates the same, although the address is displayed in the right–hand display area. For the HP 8360, access the HP–IB menu under the **[SYSTEM MENU]** key. Verify that the address is 19 and programming language is "Analyzer".

2. Power on the HP 8757D/E Scalar Network Analyzer. The current HP–IB address is shown in the active entry area of the CRT. If it is not 16, press **[LOCAL] [8757] [1] [6] [ENT]** to set the address to 16.

Check out procedure

Press **[PRESET]** on the analyzer. If the 8757 SYSTEM INTERFACE is properly connected, and the address of the source correctly set, both the analyzer and the source will perform an instrument preset. If either instrument detects a failure during instrument preset, that instrument displays the error encountered. The operating manual of the source gives instructions to help interpret the error message. If the analyzer displays an error message, see "In Case of Difficulty" in the operating manual.

Configuring Microsoft QuickC

It is important to configure Microsoft QuickC properly for operation with the HP 82335A HP–IB Command Library and the following programs. Before running any program, verify the following:

1. When installing Microsoft QuickC, choose either the small or large memory model. More importantly, the graphics library (GRAPHICS.LIB) should be included in the combined standard QuickC library (or libraries if you installed more than one memory model). If you did not do this upon initial installation, you may want to re–install Microsoft QuickC. Refer to the Microsoft QuickC manuals for more information.

2. It is assumed that Microsoft QuickC is installed in the "qc25" directory on the default drive with the following subdirectories:

qc25\bin	for binary and help files
qc25\include	for include files
qc25\lib	for library files

3. Copy the following HP 82335A HP–IB Command Library files to the proper destination:

   CLHPIB.LIB – – > qc25\lib\CLHPIB.LIB
   CHPIB.H – – > qc25\include\CHPIB.H
   CFUNC.H – – > qc25\include\CFUNC.H

4. Load Microsoft QuickC by typing "QC" at the MS–DOS prompt. You may need to change the default directory to "qc25".

5. Activate the Microsoft QuickC Options menu by clicking the mouse on the menu bar or by pressing the **[ALT] [O]** keys. Enable the "Full Menus" option.

6. Again from the Microsoft QuickC Options menu, select the "Environment" menu. Enter the appropriate file directory names using the names in step 2 above. Select <OK> when done.

7. Again from the Microsoft QuickC Options menu, select the "Make" menu. Select the "Linker Flags" option. From this menu, enter the following for `"GLOBAL FLAGS:  Stack Size":`

   ```
   4096
   ```

   Also from this menu, enter the following for `"CUSTOM FLAGS: Global":`

   ```
   qc25\lib\clhpib.lib
   ```

   Select <OK> when done.

   This will allow you to compile and run programs using HP 82335A HP–IB Command Library within the Microsoft QuickC integrated environment and avoid any run–time errors for stack overflow. The default stack size is 2048 bytes and run–time errors will be encountered on programs that have large data arrays used for trace data and/or learn strings.

When you exit Microsoft QuickC, the information entered in steps 5, 6, and 7, will be retained in its startup information file (QC.INI) so that you will not need to be re–enter them later.

If you wish to use the command line compiler instead of compiling within the integrated environment, use QCL with the appropriate switches. For "filename.c", the following compiles the file using the large memory model:

```
qcl/AL filename.c/link\qc25\lib\clhpib.lib+
\qc25\lib\/STACK:4096
```

The flags for the QuickC command line compiler (QCL) are case sensitive so be careful to enter them correctly.

Programming examples

The following example programs introduce the HP–IB capabilities of the analyzer. Each example program consists of these sections:

1. A description of the functions exercised.
2. The program listing.
3. An explanation of each program line.
4. Detailed instructions for operating the program.

When you finish all of the example programs, you will have a good idea of the power of the HP 8757D/E when used in an automatic system. Note that line numbers aren't used in C programs but are included in the program listings for the functional explanations. The HP–IB Command Library function names are shown in upper case for emphasis. Remember that identifier names are case–sensitive in the C language, so you must be consistent in your usage.

Error checking line should be performed after every HP–IB library call. Each HP 82335A HP–IB Command Library call returns a value representing the error status of the operation. An error handler routine (see program 1) can be used to return an appropriate HP–IB error message (timeout, etc.) if an error occurs. For example:

```
error = IOTIMEOUT (isc,10.0);
error_handler (error, "IOTIMEOUT");
```

If an error occurs, the number corresponding to that error is assigned to the variable "error". Within the error_handler routine, "error" is compared to the constant NOERR (=0). If an error occurred, a message appears on the computer screen stating the error number and type of error. The error values and errstr function are contained within the CHPIB.H include file.

Program 1: remote, local, and local lockout

The analyzer may be used with the front panel (local operation) or programmed via HP–IB (remote operation). The programmer has control over the operation of all instruments in the system.

When the computer first addresses an instrument, the instrument is placed in a special remote operating mode, called remote mode. When in remote, the instrument does not respond to its front panel, except for the **[LOCAL]** key. **[LOCAL]** cancels the remote mode and allows the instrument to be used with its front panel.

The computer can also return the instrument to local operation. To do so, the computer sends a special command that forces the instrument to go to local mode.

The programmer of an automatic system may need to prevent the operator from returning the instrument to local operation (via **[LOCAL]**). When the local lockout function of the computer is used, the instruments cannot exit remote mode, even if **[LOCAL]** is pressed.

Frequently, the programmer needs to place the instruments connected to the computer into a known state. When preset, the analyzer defaults to the conditions shown below. The instrument preset function operates the same as the front–panel **[PRESET]** key on the analyzer and the source. When presetting the analyzer and its associated source, send the PRESET command only to the analyzer. The analyzer will preset the source attached to the 8757 SYSTEM INTERFACE.

HP 8757D/E instrument preset conditions

Channels1 and 2 on. The channel menu appears in the softkey area of the CRT.

- Measure power A on channel 1.
- Measure power B on channel 2.
- Measure power C[2] (or B[1]) on channel 3.
- Measure power R on channel 4.
- Display measurement data in log magnitude format.
- Scale = 20 dB/div.
- Reference level 0 dB for all channels.
- Reference level step size = 20 dB.
- Averaging off.
- Averaging factor = 8.
- Cursor off.
- All labels on.
- Channel 1 as the active channel.
- Modulation drive on.
- Number of points = 401.
- Detector mode set for AC detection.
- Smoothing set for 5.0% of span (off).
- Cursor format = log magnitude.
- Search value = –3 dB[1].
- Adaptive normalization off[1].
- Temperature compensation on.
- Repeat autozero off.
- Detector amplitude offset reset to 0.[1]
- Detector frequency offset[3] off, start and stop = 50 MHz.

Source

- Instrument preset.
- Sweep time set to 200 ms.
- HP 8350B square wave modulation on
- HP 8340/41 SHIFT PULSE on; RF Output on.
- HP 8360 Scalar Modulation on; RF Output on; Analyzer mode.

Plotter

- Abort plot if in progress.
- P1 and P2 scaling points unchanged.
- Selection of plotter pens unchanged.

Printer

- Abort print if in progress.

Disk drive[1]

- Abort any data transfers in progress.
- Unit number unchanged.
- Volume number unchanged.
- ASCII or binary mode unchanged.

The following analyzer conditions are not changed during a PRESET (IP) command execution:

- Reference position.
- Trace memory.
- Save/Recall registers.
- HP–IB addresses.
- Request mask.
- Limit lines[1].
- Title.
- Detector offset (HP 8757E only).
- User–defined plot.
- 8757 System Interface control on/off.
- Repeat autozero timer.
- Display intensity.
- Display colors[1].

[1] HP 8757D only.

[2] HP 8757D Option 001 only.

[3] HP 8757D with HP 85037 series precision detector only.

Program 1 listing

```
 10: /*  HP 8757D/E QuickC IPG Program1  */
 20:
 30: #include <graph.h>
 40: #include <stdio.h>
 50: #include <cfunc.h>
 60: #include <chpib.h>
 70:
 80: void disp_prompt (void);
 90: void error_handler (int error_no, char
     *routine);
100:
110: main ()
120: {
130:      long      isc=7,
140:                sna=716;
150:      int       error;
160:
170:    _clearscreen (_GCLEARSCREEN);
180:
190:    error = IOTIMEOUT (isc,10.0);
200:    error_handler (error, "IOTIMEOUT");
210:    error = IOABORT (isc);
220:    error_handler (error, "IOABORT");
230:    error = IOCLEAR (isc);
240:    error_handler (error, "IOCLEAR");
250:    error = IOREMOTE (sna);
260:    error_handler (error, "IOREMOTE");
270:    disp_prompt ();
280:
290:    error = IOREMOTE (sna);
300:    error_handler (error, "IOREMOTE");
310:    error = IOLLOCKOUT (isc);
320:    error_handler (error, "IOLLOCKOUT");
330:    disp_prompt ();
340:
350:    error = IOLOCAL (isc);
360:    error_handler (error, "IOLOCAL");
370:    disp_prompt ();
380:
390:    error = IOOUTPUTS (sna, "IP",2);
400:    error_handler (error, "IOOUTPUTS");
410: }
420:
430: void disp_prompt (void)
440:    {
450:      char   ch;
460:
470:      _settextposition (25,1);
480:      printf ("Press <ENTER> to continue
             \n");
490:      ch = getche ();
500:      _clearscreen (_GCLEARSCREEN);
510:    }
520:
530: void error_handler (int error_no, char
     *routine)
540:    {
550:      char   ch;
560:
570:      if (error_no != NOERR)
580:      {
590:        printf ("Error in call to %s \n",
                routine);
600:        printf ("     Error = %d : %s \n",
                 error_no, errstr (error_no));
610:        printf ("Press <ENTER> to contin
                ue\n");
620:        ch = getche ();
630:        exit (1);
640:      }
650:    }
```

Program 1 explanation

Line 30 Tell the compiler which file includes information on _clearscreen() and _settextposition().

Line 40 Tell the compiler which file includes information on printf().

Line 50 Tell the compiler which file includes information on the HP 82335A HP–IB Command Library I/O functions.

Line 60 Tell the compiler which file includes information on the HP 82335A HP–IB Command Library error constants and errstr().

Line 80 Function prototype for the disp_prompt() routine.

Line 90 Function prototype for the error_handler() routine.

Line 110 Define the beginning of the main() routine.

Line 130 Define a variable and assign it a value for the interface select code of the HP 82335A HP–IB interface card.

Line 140 Define a variable and assign it a value for the HP–IB address of the HP 8757D/E analyzer.

Line 150 Define a variable for the HP–IB Command Library error status.

Line 170 Clear the computer CRT.

Line 190 Define a system timeout of 10 seconds. Timeout allows recovery from I/O operations that aren't completed in less than 10 seconds. The timeout value passed must be a float, so include the decimal point (vs. passing just "10") so it is not passed as an integer.

Line 200 Perform error trapping.

Line 210 Abort any HP–IB transfers.

Line 220 Perform error trapping.

Line 230 Clear the analyzer's HP–IB interface.

Line 240 Perform error trapping.

Line 250 Set the analyzer and source to remote mode.

Line 260 Perform error trapping.

Line 270 Wait until **[ENTER]** is pressed to continue.

Line 290 Set the analyzer and source to remote mode.

Line 300 Perform error trapping.

Line 310 Lock out the **[LOCAL]** key of the analyzer and source.

Line 320 Perform error trapping.

Line 330 Wait until **[ENTER]** is pressed to continue.

Line 350 Set the analyzer and source to local mode.

Line 360 Perform error trapping.

Line 370 Wait until **[ENTER]** is pressed to continue.

Line 390 Preset the analyzer and source.

Line 400 Perform error trapping.

Line 410 The end of main().

Line 430 Define a routine that prints a prompt on the computer CRT and waits for **[ENTER]** to be pressed.

Line 450 Define a variable to hold the keypress.

Line 470 Locate the text cursor at the beginning of row 25.

Line 480 Print a prompt on the computer CRT.

Line 490 Wait for a keypress, then continue.

Line 500 Clear the computer CRT.

Line 510 The end of disp_prompt().

Line 530 Define a routine that checks the HP–IB Command Library error status. Define the types of variables passed to this routine: error_no is the error value, routine is the HP–IB Command Library routine called.

Line 550 Define a variable to hold the keypress.

Line 570 Test if an error actually occurred.

Line 590 Yes, one did. Print on the computer CRT which HP–IB Command Library routine the error occurred in.

Line 600 Print on the computer CRT the error number and a message.

Line 610 Print a prompt on the computer CRT.

Line 620 Wait for a keypress, then continue.

Line 630 Since an error occurred, halt program execution. If you want to trap for specific errors, this "exit" statement could be replaced with some specific error messages to display, error correcting actions to be performed, and then allow program execution to continue.

Line 650 The end of error_handler().

Running program 1

1. Press **[ALT] [F] [N]** on the computer. This clears the QuickC screen.
2. Type in the program.
3. Press **[ALT] [R] [G]** on the computer to run the program.
4. When the program pauses, the analyzer is in remote mode. You can verify this by observing the lights in the INSTRUMENT STATE area of the analyzer. The R (remote) and L (listen) lights should be on. Try pressing any key on the analyzer (except **[LOCAL]**). Nothing happens. The source is also in remote mode. Now press **[LOCAL]** and verify that the keys on the analyzer are active. Also, notice the R light went out when you pressed **[LOCAL]**. The source went into local mode along with the analyzer.
5. Press **[ENTER]** on the computer. The analyzer is again in remote mode. This time, however, the **[LOCAL]** key is locked out. Try pressing **[LOCAL]** and the other keys. None of the keys on the analyzer or the source cause any action.
6. Press **[ENTER]** on the computer. All instruments on the HP–IB interface are returned to local mode, including the analyzer and source. Verify that the R light on the analyzer and the REM light on the source are off.
7. Press **[ENTER]** on the computer. The analyzer and source are both preset. Note that the computer sent the Instrument Preset command only to the analyzer. The analyzer, in turn, presets the source.

Remember, to preset both the analyzer and the source, you only need to send the instrument preset command to the analyzer. Do not send instrument preset to the source by way of passthru mode (discussed in program 3).

Program 2: controlling the front panel

All front panel keys and most of the softkeys of the analyzer may be programmed remotely via HP–IB. For example, you can program the scale per division, reference level, and reference position for each channel.

Program 2 listing

```
 10: /*  HP8757D/E QuickC IPG Program2   */
 20:
 30: #include <string.h>
 40: #include <graph.h>
 50: #include <stdio.h>
 60: #include <cfunc.h>
 70: #include <chpib.h>
 80:
 90: int IOOUTPUTS_CHK (long hpib_adr, char
     *cmd_str);
100: void disp_prompt (void);
110: void error_handler (int error_no, char
     *routine);
120:
130: main ()
140: {
150:    long    isc =7,
160:            sna =716;
170:    int     error;
180:
190:    _clearscreen (_GCLEARSCREEN);
200:
210:    error = IOTIMEOUT (isc,10.0);
220:    error_handler (error, "IOTIMEOUT");
230:    error = IOABORT (isc);
240:    error_handler (error, "IOABORT");
250:    error = IOCLEAR (isc);
260:    error_handler (error, "IOCLEAR");
270:    error = IOOUTPUTS_CHK (sna, "IP");
280:    disp_prompt ();
290:
300:    error = IOOUTPUTS_CHK (sna, "C1C0C2");
310:    disp_prompt ();
320:
330:    error = IOOUTPUTS_CHK (sna, "SD10");
340:    disp_prompt ();
350:
360:    error = IOOUTPUTS_CHK (sna, "RL-10");
370:    disp_prompt ();
380:
390:    error = IOOUTPUTS_CHK (sna, "RP4");
400:    disp_prompt ();
410:
420:    error = IOOUTPUTS_CHK (sna, "IA");
430:    disp_prompt ();
440:
450:    error=IOOUTPUTS_CHK(sna,"C0C1SD5;RP4;RL-5");
460: }
470:
480: int IOOUTPUTS_CHK (long hpib_adr, char
     *cmd_str)
490:    {
500        int    length, error_no;
510:
520:       length = strlen (cmd_str);
530:       error_no = IOOUTPUTS (hpib_adr, cmd_str,
           length);
540:       error_handler (error_no,"IOOUTPUTS_CHK");
550:       return error_no;
560:    }
570:
580: void disp_prompt (void)
590:    {
600:       char  ch;
610:
620        _settextposition (25,1);
630:       printf ("Press <ENTER> to continue\n");
640:       ch = getche ();
650:       _clearscreen (_GCLEARSCREEN);
660:    }
670:
680: void error_handler (int error_no, char
     *routine)
690:    {
700:       char  ch;
710:
720:       if (error_no != NOERR)
730:       {
740:         printf ("Error in call to %s \n",
             routine);
750:         printf (" Error = %d : %s \n",
              error_no,errstr (error_no));
760:         printf ("Press <ENTER> to continue\n");
770:         ch = getche ();
780:         exit (1);
790:       }
800:    }
```

Program 2 explanation

Line 30 Tell the compiler which file includes information on string functions.

Line 40 Tell the compiler which file includes information on _clearscreen() and _settextposition().

Line 50 Tell the compiler which file includes information on printf().

Line 60 Tell the compiler which file includes information on the HP–IB Command Library I/O functions.

Line 70 Tell the compiler which file includes information on the HP–IB Command Library error constants and errstr().

Line 90 Function prototype for the IOOUTPUTS_CHK() routine.

Line 100 Function prototype for the disp_prompt() routine.

Line 110 Function prototype for the error_handler() routine.

Line 130 Define the beginning of the main() routine.

Line 150 Define a variable and assign it a value for the interface select code.

Line 160 Define a variable and assign it a value for the HP–IB address of the analyzer.

Line 170 Define a variable for the HP–IB Command Library error status.

Line 190 Clear the computer CRT.

Line 210 Define a system timeout of 10 seconds.

Line 220 Perform error trapping.

Line 230 Abort any HP–IB transfers.

Line 240 Perform error trapping.

Line 250 Clear the analyzer's HP–IB interface.

Line 260 Perform error trapping.

Line 270 Preset the analyzer and the source.

Line 280 Wait until **[ENTER]** is pressed to continue.

Line 300 Select channel 1 and turn it off. Turn on channel 2.

Line 310 Wait until **[ENTER]** is pressed to continue.

Line 330 Set the scale per division to 10 dB. Note that semicolon (";") terminators are needed after any analyzer command that can have a variable length. However, no terminator is needed here because this is the only command on the line and the linefeed in the End–of–Line string (the HP 82335A default is carriage return/linefeed) will terminate it.

Line 340 Wait until **[ENTER]** is pressed to continue.

Line 360 Set the reference level to −10 dBm. Again, note the absence of a terminator (";").

Line 370 Wait until **[ENTER]** is pressed to continue.

Line 390 Set the reference position line to the center of the screen (graticule 4).

Line 400 Wait until **[ENTER]** is pressed to continue.

Line 420 Program channel 2 to measure input A (reflection) instead of input B (transmission).

Line 430 Wait until **[ENTER]** is pressed to continue.

Line 450 There are many commands on one line, with terminators. Turn channel 2 off and channel 1 on (C0C1). Set the scale per division (SD) to 5 dB, the reference position line (RP) to the center of the screen, and the reference level (RL) to −5 dBm. Semicolon (";") terminators are needed after any analyzer command that can have a variable length. Extra terminators never hurt, so use them liberally.

Line 460 The end of main().

Line 480 Define a routine that outputs string commands and performs error trapping. Define the types of variables passed to this routine: hpib_adr is the HP−IB address, cmd_str is the command string to output.

Line 500 Define variables for the length of the string and the error status.

Line 520 Determine the length of the command string.

Line 530 Output the command string.

Line 540 Perform error trapping.

Line 550 Return the error status as the value of the routine.

Line 560 The end of IOOUTPUTS_CHK().

Line 580 Define a routine that prints a prompt on the computer CRT and waits for **[ENTER]** to be pressed.

Line 600 Define a variable to hold the keypress.

Line 620 Locate the text cursor at the beginning of row 25.

Line 630 Print a prompt on the computer CRT.

Line 640 Wait for a keypress, then continue.

Line 650 Clear the computer CRT.

Line 660 The end of disp_prompt().

Line 680 Define a routine that checks the HP−IB Command Library error status. Define the types of variables passed to this routine: error_no is the error value, routine is the HP−IB Command Library routine called.

Line 700 Define a variable to hold the keypress.

Line 720 Test if an error actually occurred.

Line 740 Yes, one did. Print on the computer CRT which HP−IB Command Library routine the error occurred in.

Line 750 Print on the computer CRT the error number and a message.

Line 760 Print a prompt on the computer CRT.

Line 770 Wait for a keypress, then continue.

Line 780 Since an error occurred, halt program execution.

Line 800 The end of error_handler().

Running program 2

1. Press **[ALT] [F] [N]** on the computer. This clears the previous program.
2. Type in this program and press **[ALT] [R] [G]** on the computer.
3. The computer presets the analyzer and source and pauses. Note the settings of channel 1 and 2, then press **[ENTER]**.
4. Channel 1 is turned off. Channel 2 is now the active channel, as you can see from the highlighted box around the channel 2 mode labels on the analyzer CRT. Press **[ENTER]**.
5. Channel 2 scale per division is now set to 10 dB. It defaulted to 20 dB/div at preset. Press **[ENTER]**.
6. The reference level is set to −10 dBm (it was 0.0 dBm). Press **[ENTER]**.
7. The reference position line is set to the center of the CRT (graticule 4). The top of the CRT is graticule 8 and the bottom is graticule 0. Press **[ENTER]**.
8. Change the measurement to input A (reflection) instead of input B (transmission). At preset, channel 2 defaults to input B. Press **[ENTER]**.
9. In one statement: turn off channel 2, turn on channel 1, set the scale per division to 5 dB, set the reference position line to the center of the CRT, and set the reference level to −5 dBm.

Program 3: passthru mode

In normal operation, the system source, digital plotter, printer, and disk drive (HP 8757D only) are connected to the 8757 SYSTEM INTERFACE. This connection allows the analyzer to control and extract information from the other parts of the measurement system. To control other instruments with the computer, the analyzer has a built–in passthru command that takes a command from the computer and passes it on to one of the instruments connected to the 8757 SYSTEM INTERFACE.

To initiate passthru mode, first tell the analyzer which instrument you wish to command by setting the passthru address. Then, to talk (or listen) to that device, address the analyzer's special passthru HP–IB address (which is different from the analyzer's HP–IB address). While in the passthru mode, the analyzer stops updating its CRT and does not respond to its front panel (because it's in remote mode). To remove the analyzer from passthru mode, simply address it via HP–IB. While in passthru mode, do not press **[LOCAL]** on the analyzer.

The analyzer's passthru address is calculated from its HP–IB address. If the address of the analyzer is even (such as 16 decimal) then the passthru address is the next larger number (17 decimal). If the address of the analyzer is odd (such as 15 decimal), then the passthru address is the next smaller number (14 decimal). Never set the address of the analyzer such that its address conflicts with one of the instruments connected to the 8757 SYSTEM INTERFACE. For instance, if the source is set to 19 decimal, do not set the address of the analyzer to 19.

Data can be sent to or received from any instrument on the 8757 SYSTEM INTERFACE via passthru mode. The IOLOCAL, IOREMOTE, and IOTRIGGER HP–IB messages do not pass through the analyzer.

Program 3 listing

```
 10: /*  HP8757D/E QuickC IPG Program3   */
 20:
 30: #include <string.h>
 40: #include <graph.h>
 50: #include <stdio.h>
 60: #include <cfunc.h>
 70: #include <chpib.h>
 80:
 90: int IOOUTPUTS_CHK (long hpib_adr, char
     *cmd_str);
100: void error_handler (int error_no, char
     *routine);
110:
120: main ()
130: {
140:    char     cmd [80];
150:    long     isc =7,
160:             sna=716,
170:             passthru=717;
180:    int      error;
190:    float    min_freq, max_freq,
200:             start_freq, stop_freq; 210:
220:    _clearscreen (_GCLEARSCREEN);
230:
240:    error = IOTIMEOUT (isc,10.0);
250:    error_handler (error, "IOTIMEOUT");
260:    error = IOABORT (isc);
270:    error_handler (error, "IOABORT");
280:    error = IOCLEAR (isc);
290:    error_handler (error, "IOCLEAR");
300:
310:    error = IOOUTPUTS_CHK (sna, "IP");
320:    error = IOOUTPUTS_CHK (sna, "PT19;");
330:    error = IOOUTPUTS_CHK (passthru, "OPFA");
340:    error = IOENTER (passthru, &min_freq);
350:    error_handler (error, "IOENTER");
360:    min_freq = min_freq /1.0e+9;
370:    error = IOOUTPUTS_CHK (passthru, "OPFB");
380:    error = IOENTER (passthru, &max_freq);
390:    error_handler (error, "IOENTER");
400:    max_freq = max_freq /1.0e+9;
410:    error = IOCLEAR (sna);
420:    error_handler (error, "IOCLEAR");
430:    printf ("Frequency limits: %f to %f
   GHz\n", min_freq, max_freq);
440:
450:    printf ("Start frequency (GHz) ? ");
460:    scanf ("%f", &start_freq);
470:    printf ("Stop frequency (GHz) ? ");
480:    scanf ("%f", &stop_freq);
490:
500:    sprintf (cmd, "FA%fGZ;FB%fGZ;",
            start_freq, stop_freq);
510:    error = IOOUTPUTS_CHK (passthru, cmd);
520:    error = IOCLEAR (sna);
530:    error_handler (error, "IOCLEAR");
540: }
550:
560: int IOOUTPUTS_CHK (long hpib_adr, char
     *cmd_str)
570:    {
580:       int    length, error_no;
590:
600:       length = strlen (cmd_str);
610:       error_no = IOOUTPUTS (hpib_adr,
           cmd_str, length);
620:       error_handler (error_no,"IOOUTPUTS_CHK");
630:       return error_no;
640:    }
650:
660: void error_handler (int error_no, char
     *routine)
670:    {
680:       char  ch;
690:
700:       if (error_no != NOERR)
710:       {
720:          printf ("Error in call to %s \n",
            routine);
730:          printf ("     Error = %d : %s \n",
              error_no, errstr (error_no));
740:          printf ("Press <ENTER> to continue
            \n");
750:          ch = getche ();
760:          exit (1);
770:       }
780:    }
```

Program 3 explanation

Line 30 Tell the compiler which file includes information on string functions.

Line 40 Tell the compiler which file includes information on _clearscreen() and _settextposition().

Line 50 Tell the compiler which file includes information on printf().

Line 60 Tell the compiler which file includes information on the HP–IB Command Library I/O functions.

Line 70 Tell the compiler which file includes information on the HP–IB Command Library error constants and errstr().

Line 90 Function prototype for the IOOUTPUTS_CHK() routine.

Line 100 Function prototype for the error_handler() routine.

Line 120 Define the beginning of the main() routine.

Line 140 Define a string variable for the output commands.

Line 150 Define a variable and assign it a value for the interface select code.

Line 160 Define a variable and assign it a value for the HP–IB address of the analyzer. (This is the analyzer's control address).

Line 170 Define a variable and assign it a value for the analyzer's passthru address. By communicating to this HP–IB address, the computer will control a device connected to the 8757 SYSTEM INTERFACE.

Line 180 Define a variable for the HP–IB Command Library error status.

Line 190 Define variables for the minimum and maximum frequencies of the source.

Line 200 Define variables for the start and stop frequencies of a sweep.

Line 220 Clear the computer CRT.

Line 240 Define a system timeout of 10 seconds.

Line 250 Perform error trapping.

Line 260 Abort any HP–IB transfers.

Line 270 Perform error trapping.

Line 280 Clear the analyzer's HP–IB interface.

Line 290 Perform error trapping.

Line 310 Preset the analyzer and source.

Line 320 Tell the analyzer which device is controlled through the analyzer's passthru address. In this case, the source (device19).

Line 330 Send a command to the source. Command it to output its current start frequency.

Line 340 Read the start frequency from the source.

Line 350 Perform error trapping.

Line 360 Scale the start frequency to display it in GHz.

Line 370 Command the source to output its current stop frequency.

Line 380 Read the stop frequency from the source.

Line 390 Perform error trapping.

Line 400 Scale the stop frequency to display it in GHz.

Line 410 Exit passthru mode by clearing the analyzer's HP–IB interface.

Line 420 Perform error trapping.

Line 430 Print the start and stop frequencies.

Line 450 Print a prompt asking for the start frequency.

Line 460 Get start frequency from user.

Line 470 Print a prompt asking for the stop frequency.

Line 480 Get stop frequency from user.

Line 500 Create a formatted output by printing the start and stop frequencies of the source to a string.

Line 510 Set the start and stop frequencies of the source to those given by the user.

Line 520 Exit passthru mode by clearing the analyzer's HP–IB interface.

Line 530 Perform error trapping.

Line 540 The end of main().

Line 560 Define a routine that outputs string commands and performs error trapping. Define the types of variables passed to this routine: hpib_adr is the HP–IB address, cmd_str is the command string to output.

Line 580 Define variables for the length of the string and the error status.

Line 600 Determine the length of the command string.

Line 610 Output the command string.

Line 620 Perform error trapping.

Line 630 Return the error status as the value of the routine.

Line 640 The end of IOOUTPUTS_CHK().

Line 660 Define a routine that checks the HP–IB Command Library error status. Define the types of variables passed to this routine: error_no is the error value, routine is the HP–IB Command Library routine called.

Line 680 Define a variable to hold the keypress.

Line 700 Test if an error actually occurred.

Line 720 Yes, one did. Print on the computer CRT which HP−IB Command Library routine the error occurred in.

Line 730 Print on the computer CRT the error number and a message.

Line 740 Print a prompt on the computer CRT.

Line 750 Wait for a keypress, then continue.

Line 760 Since an error occurred, halt program execution.

Line 780 The end of error_handler().

Running program 3

1. Clear the computer CRT and type in the program.
2. Press **[ALT] [R] [G]** on the computer to run the program.
3. The computer presets the analyzer and the source, reads the start and stop frequency of the source, and displays it on the CRT of the computer. At preset, the source defaults to the full frequency range of the plug−in. The values read represent the frequency limits of this plug−in. When the computer stops, it displays the prompt:

   ```
   Start frequency (GHz)?
   ```

   Enter a start frequency in the frequency range of the plug−in and press **[ENTER]**.
4. The computer displays the prompt:

   ```
   Stop frequency (GHz)?
   ```

   Enter a stop frequency in the frequency range of the plug−in (but higher than the start frequency) and press **[ENTER]**.
5. The computer sets the start and stop frequency of the source to those you entered. The analyzer immediately begins sweeping the frequency range you defined.

Points to remember: You must address the analyzer after using passthru mode to return it to normal swept operation. Any command can be sent via passthru mode to any instrument on the 8757 SYSTEM INTERFACE, and any data can be read. Service requests and parallel polls do not passthru the analyzer.

Program 4: cursor operations

To enhance the speed and accuracy of measurements, the analyzer contains a built−in cursor that displays the frequency and magnitude of a trace at any given point. To make measurements even more efficient, the cursor may be set to the maximum or minimum point on the trace simply by pressing a softkey. These cursor functions are available via HP−IB commands.

With a computer, the cursor may be turned on and off, its position (0 to n−1, where n is the number of points per trace) set, its value and position read, and set to the maximum or minimum point on the trace. The cursor functions all apply to the active channel (the channel accessed most recently). You have complete control over cursor operations via HP−IB.

Cursor programming is especially useful for measuring parameters like flatness and maximum power, where you are interested in the highest and lowest point on the trace. For measuring parameters such as 3 dB points and other specific points (not a maximum or minimum), it is more efficient to use either the cursor search functions (available on the HP 8757D only) or to read the entire trace and search for the points you need.

Program 4 listing

```
 10: /*   HP 8757D/E QuickC IPG Program 4   */
 20:
 30: #include <string.h>
 40: #include <graph.h>
 50: #include <stdio.h>
 60: #include <cfunc.h>
 70: #include <chpib.h>
 80:
 90: int IOOUTPUTS_CHK (long hpib_adr, char
     *cmd_str);
100: void error_handler (int error_no, char
     *routine);
110:
120: main ()
130: {
140:    char    cmd [80];
150:    long    isc = 7,
160:            sna = 716,
170:            passthru = 717;
180:    int     error, elements,
190:            crsr_posn, new_posn;
200:    float   start_freq = 2.0,
210:               stop_freq = 5.0,
220:               crsr_freq, cur_freq, crsr_vals[2];
230:
240:         _clearscreen (_GCLEARSCREEN);
250:
260:         error = IOTIMEOUT (isc,10.0);
270:         error_handler (error, "IOTIMEOUT");
280:         error = IOABORT (isc);
290:         error_handler (error, "IOABORT");
300:         error = IOCLEAR (isc);
310:         error_handler (error, "IOCLEAR");
320:
330:         error = IOOUTPUTS_CHK (sna, "IP");
340:         error = IOOUTPUTS_CHK (sna, "PT19;");
350:         sprintf (cmd, "FA%fGZ;FB%fGZ;",
             start_freq, stop_freq);
360:         error = IOOUTPUTS_CHK (passthru, cmd);
370:
375:         error = IOOUTPUTS_CHK(sna," ");
380:         error = IOOUTPUTS_CHK (sna, "C2CXOC");
390:         elements = 2;
400:         error = IOENTERA (sna, crsr_vals, &ele
             ments);
410:         error_handler (error, "IOENTERA");
```

```
420:        printf ("Cursor reads %f dB at
            position %4.0f\n\n", crsr_vals[0],
            crsr_vals[1]);
430:
440:        printf ("Desired cursor position
            (0..400)           ? ");
450:        scanf ("%i", &new_posn);
460:        sprintf (cmd, "SC%d;", new_posn);
470:        error = IOOUTPUTS_CHK (sna, cmd);
480:        error = IOOUTPUTS_CHK (sna, "OC");
490:        elements = 2;
500:        error = IOENTERA (sna, crsr_vals, &ele
            ments);
510:        error_handler (error, "IOENTERA");
520:        printf ("Value at position %4.0f is
            %7.3f dB\n\n", crsr_vals[1],
            crsr_vals[0]);
530:
540:        printf ("Cursor frequency (GHz) ? ");
550:        scanf ("%f", &cur_freq);
560:        new_posn = 400 * ((cur_freq -
            start_freq)/(stop_freq - start_freq));
570:        sprintf (cmd, "SC%d;", new_posn);
580:        error = IOOUTPUTS_CHK (sna, cmd);
590:        error = IOOUTPUTS_CHK (sna, "OC");
600:        elements = 2;
610:        error = IOENTERA (sna, crsr_vals, &ele
            ments);
620:        error_handler (error, "IOENTERA");
630:        cur_freq = start_freq + (stop_freq -
            start_freq) * (crsr_vals[1]/ 400);
640:        printf ("Cursor reads %7.3f dB at
            %7.3f GHz\n", crsr_vals[0], cur_freq);
650:     }
660:
670:     int IOOUTPUTS_CHK (long hpib_adr, char
         *cmd_str)
680:        {
690:           int    length, error_no;
700:
710:           length = strlen (cmd_str);
720:           error_no = IOOUTPUTS (hpib_adr,
               cmd_str, length);
730:           error_handler (error_no,
               "IOOUTPUTS_CHK");
740:           return error_no;
750:        }
760:
770:     void error_handler (int error_no, char
         *routine)
780:        {
790:           char   ch;
800:
810:           if (error_no != NOERR)
820:           {
830:              printf ("Error in call to %s \n",
                  routine);
840:              printf ("      Error = %d : %s
                  \n",error_no, errstr (error_no));
850:              printf ("Press <ENTER> to continue
                  \n");
860:              ch = getche ();
870:              exit (1);
880:           }
890:        }
```

Program 4 explanation

Line 30 Tell the compiler which file includes information on string functions.

Line 40 Tell the compiler which file includes information on _clearscreen() and _settextposition().

Line 50 Tell the compiler which file includes information on printf().

Line 60 Tell the compiler which file includes information on the HP–IB Command Library I/O functions.

Line 70 Tell the compiler which file includes information on the HP–IB Command Library error constants and errstr().

Line 90 Function prototype for the IOOUTPUTS_CHK() routine.

Line 100 Function prototype for the error_handler() routine.

Line 120 Define the beginning of the main() routine.

Line 140 Define a string variable for the output commands.

Line 150 Define a variable and assign it a value for the interface select code.

Line 160 Define a variable and assign it a value for the HP–IB address of the analyzer.

Line 170 Define a variable and assign it a value for the analyzer's passthru address.

Line 180 Define variables for the HP–IB Command Library error status and the number of elements to be read into an array.

Line 220 Define variables for the present and new cursor positions.

Line 200 Define a variable and assign it a value (in GHz) for the start frequency of the desired sweep.

Line 210 Define a variable and assign it a value (in GHz) for the stop frequency of the desired sweep.

Line 220 Define variables for the present and new cursor frequencies, and an array variable for reading the cursor values.

Line 240 Clear the computer CRT.

Line 260 Define a system timeout of 10 seconds.

Line 270 Perform error trapping.

Line 280 Abort any HP–IB transfers.

Line 290 Perform error trapping.

Line 300 Clear the analyzer's HP–IB interface.

Line 310 Perform error trapping.

Line 330 Preset the analyzer and source. This sets the number of points per trace to 401.

Line 340 Tell the analyzer which instrument is controlled through the passthru address (19 is the source).

Line 350 Create a formatted output by printing the start and stop frequencies of the source to a string.

Line 360 Command the source to set a start frequency of 2 GHz and a stop frequency of 5 GHz.

Line 380 Set the cursor to the maximum point on channel 2 and command the analyzer to output the cursor's value and position.

Line 375 Exit passthru mode. Allow analyzer to display update.

Line 390 Define the number of elements to be read into an array.

Line 400 Read the value and position of the cursor.

Line 410 Perform error trapping.

Line 420 Print the value and position of the cursor on the computer CRT.

Line 440 Print a prompt asking for the cursor position.

Line 450 Get new cursor position from the user. Input should be between 0 and 400.

Line 460 Create a formatted output by printing the cursor position to a string.

Line 470 Set the cursor to the new cursor position chosen by the user.

Line 480 Command the analyzer to output the cursor's value and position.

Line 490 Define the number of elements to be read into an array.

Line 500 Read the value and position of the cursor at its new position.

Line 510 Perform error trapping.

Line 520 Print the cursor's value and position on the computer CRT.

Line 540 Print a prompt asking for the cursor frequency.

Line 550 Get new cursor frequency from the user. It must be within the frequency range of the sweep selected.

Line 560 Calculate the position of the cursor from its frequency and the start and stop frequencies of the current measurement.

Line 570 Create a formatted output by printing the cursor position to a string.

Line 580 Set the cursor to the desired position.

Line 590 Command the analyzer to output the cursor's value and position.

Line 600 Define the number of elements to be read into an array.

Line 610 Read the cursor's value and position.

Line 620 Perform error trapping.

Line 630 Calculate the cursor's actual frequency from its position and the start and stop frequencies of the current measurement. You can easily program other start and stop frequencies by following the example in program 3.

Line 640 On the computer CRT, print the value and actual frequency of the cursor.

Line 650 The end of main().

Line 670 Define a routine that outputs string commands and performs error trapping. Define the types of variables passed to this routine: hpib_adr is the HP–IB address, cmd_str is the command string to output.

Line 690 Define variables for the length of the string and the error status.

Line 710 Determine the length of the command string.

Line 720 Output the command string.

Line 730 Perform error trapping.

Line 740 Return the error status as the value of the routine.

Line 750 The end of IOOUTPUTS_CHK().

Line 770 Define a routine that checks the HP–IB Command Library error status. Define the types of variables passed to this routine: error_no is the error value, routine is the HP–IB Command Library routine called.

Line 790 Define a variable to hold the keypress.

Line 810 Test if an error actually occurred.

Line 830 Yes, one did. Print on the computer CRT which HP–IB Command Library routine the error occurred in.

Line 840 Print on the computer CRT the error number and a message.

Line 850 Print a prompt on the computer CRT.

Line 860 Wait for a keypress, then continue.

Line 870 Since an error occurred, halt program execution.

Line 890 The end of error_handler().

Running program 4

1. Clear the computer CRT and type in the program.
2. Press **[ALT] [R] [G]** on the computer.
3. The computer turns on both channels and sets channel 1 to reflection (input A) and channel 2 to transmission (input B). The cursor is positioned to the maximum point on the channel 2 trace and its value and position are read and displayed. At preset, the number of points per trace is 401.
4. The computer displays the prompt:

   ```
   Desired cursor position (0..400)?
   ```

   Type in a number between 0 and 400 and press **[ENTER]**. A position of 0 represents the left side of the analyzer's CRT (lowest frequency) and 400 represents the right side of the CRT (highest frequency). The position is set, and the cursor's value is read and printed on the CRT of the computer.
5. The computer displays the prompt:

   ```
   Cursor frequency (GHz)?
   ```

   Enter a frequency within the current start and stop frequencies of the measurement (0.01 to 20 GHz). The nearest cursor position is calculated and set. The value and position of the cursor are read, and the actual cursor frequency is calculated from the cursor's position.

Note: The original desired frequency and the actual cursor frequency are usually different. Because there are only 401 possible cursor positions, some frequencies cannot be set exactly.

To use more points per trace when using the HP 8757D, modify line 330 to be "IP SP801" for 801 points. Then modify the "400" in lines 440, 560, and 630, to "800".

Program 5: read a single value

Measurements often require that a single value be read at a CW frequency, particularly when extremely good frequency accuracy and resolution are required.

The analyzer is able to read and send a single reading of any measurement channel, via HP–IB, to the computer. The OUTPUT VALUE (OV) command operates on the active channel and causes the analyzer to send one reading of measurement data. Even when the analyzer is in normalized mode (MEAS–MEM), the OV command sends the measured, not the normalized, data.

Program 5 listing

```
/*   HP8757D/E QuickC IPG Program5   */

#include <string.h>
#include <graph.h>
#include <stdio.h>
#include <cfunc.h>
#include <chpib.h>

int IOOUTPUTS_CHK (long hpib_adr, char
*cmd_str);
void error_handler (int error_no, char
*routine);

main ()
{
   char    cmd [80];
   long    isc =7,
           sna =716,
           passthru =717;
   int     error, i;
   float   freq, freq_step, value;

   _clearscreen (_GCLEARSCREEN);

   error = IOTIMEOUT (isc,10.0);
   error_handler (error, "IOTIMEOUT");
   error = IOABORT (isc);
   error_handler (error, "IOABORT");
   error = IOCLEAR (isc);
   error_handler (error, "IOCLEAR");

   error  =  IOOUTPUTS_CHK (sna,  "IP");
   error  =  IOOUTPUTS_CHK (sna,  "PT19;");
   error  =  IOOUTPUTS_CHK (sna,  "SW0");

   freq =2.0;
   freq_step = 0.1;
   sprintf (cmd, "CW%fGZ;SF%fGZ;", freq,
   freq_step);
   error = IOOUTPUTS_CHK (passthru, cmd);
   error = IOOUTPUTS_CHK (sna, "C1IA");

   for (i =1; i <=21; i = i +1)
   {
      error  = IOOUTPUTS_CHK (sna, "OV");
      error  = IOENTER (sna, &value);
      error_handler (error, "IOENTER");
      printf ("%4d: %8.3f dB at Freq %7.3f
    GHz\n", i, value, freq);
      error  = IOOUTPUTS_CHK (passthru,
      freq = freq + freq_step;
   }

   error = IOOUTPUTS_CHK (passthru,
   "FA2GZFB4GZ");
   error = IOOUTPUTS_CHK (sna, "SW1");
}

int IOOUTPUTS_CHK (long hpib_adr, char
*cmd_str)
   {
      int     length, error_no;

      length = strlen (cmd_str);
      error_no = IOOUTPUTS (hpib_adr,
      cmd_str, length);
      error_handler (error_no,
      "IOOUTPUTS_CHK");
      return error_no;
   }

void error_handler (int error_no, char
*routine)
   {
      char   ch;

      if (error_no != NOERR)
      {
         printf ("Error in call to %s \n",
        routine);
         printf ("      Error = %d : %s \n",
         error_no, errstr (error_no));
         printf ("Press <ENTER> to continue
        \n");
         ch = getche ();
```

```
740:        exit (1);
750:     }
760:   }
```

Program 5 explanation

Line 30 Tell the compiler which file includes information on string functions.

Line 40 Tell the compiler which file includes information on _clearscreen() and _settextposition().

Line 50 Tell the compiler which file includes information on printf().

Line 60 Tell the compiler which file includes information on the HP−IB Command Library I/O functions.

Line 70 Tell the compiler which file includes information on the HP−IB Command Library error constants and errstr().

Line 90 Function prototype for the IOOUTPUTS_CHK() routine.

Line 100 Function prototype for the error_handler() routine.

Line 120 Define the beginning of the main() routine.

Line 140 Define a string variable for the output commands.

Line 150 Define a variable and assign it a value for the interface select code.

Line 160 Define a variable and assign it a value for the HP−IB address of the analyzer.

Line 170 Define a variable and assign it a value for the analyzer's passthru address.

Line 180 Define variables for the HP−IB Command Library error status and a loop counter.

Line 190 Define variables for the present frequency, frequency step size, and cursor value.

Line 210 Clear the computer CRT.

Line 230 Define a system timeout of 10 seconds.

Line 240 Perform error trapping.

Line 250 Abort any HP−IB transfers.

Line 260 Perform error trapping.

Line 270 Clear the analyzer's HP−IB interface.

Line 280 Perform error trapping.

Line 300 Preset the analyzer and source.

Line 310 Tell the analyzer which instrument is controlled through the passthru address (19 is the source).

Line 320 Put the analyzer in non−swept mode. This step is necessary when you wish to read single values. After receiving this command, the analyzer stops updating its display.

Line 340 Define a start frequency for further measurements (in GHz).

Line 350 Define a frequency increment (in GHz).

Line 360 Create a formatted output by printing the CW frequency and freqency step size to a string.

Line 370 Put the source into CW mode at the start frequency and set its frequency step size to that of the frequency increment.

Line 380 Command the analyzer to measure reflection (input A) on channel 1. This statement also causes the analyzer to exit passthru mode.

Line 400 Make 21 measurements, at equally−spaced CW frequencies.

Line 420 Command the analyzer to send the current reading of channel 1 (the active channel) to the computer. The reading is taken immediately.

Line 430 Read the value. In this instance, no format has been defined so the default format of ASCII is in effect.

Line 440 Print the measurement number, the reading, and the frequency on the computer CRT.

Line 450 Perform error trapping.

Line 460 Command the source to increment the CW frequency by the step size set earlier (line 390). This is a very fast way of setting a series of equally−spaced frequencies.

Line 470 Increment the variable that contains the current frequency. This variable is only used for printing the current frequency at each iteration of the loop.

Line 480 End of the loop.

Line 500 Command the source to sweep from 2 to 4 GHz. The source exits CW mode and returns to start/stop mode.

Line 510 Command the analyzer to return to swept mode. The analyzer again updates the trace information on the display. This command also exits passthru mode.

Line 520 The end of main().

Line 540 Define a routine that outputs string commands and performs error trapping. Define the types of variables passed to this routine: hpib_adr is the HP–IB address, cmd_str is the command string to output.

Line 560 Define variables for the length of the string and the error status.

Line 580 Determine the length of the command string.

Line 590 Output the command string.

Line 600 Perform error trapping.

Line 610 Return the error status as the value of the routine.

Line 620 The end of IOOUTPUTS_CHK().

Line 640 Define a routine that checks the HP–IB Command Library error status. Define the types of variables passed to this routine: error_no is the error value, routine is the HP–IB Command Library routine called.

Line 660 Define a variable to hold the keypress.

Line 680 Test if an error actually occurred.

Line 700 Yes, one did. Print on the computer CRT which HP–IB Command Library routine the error occurred in.

Line 710 Print on the computer CRT the error number and a message.

Line 720 Print a prompt on the computer CRT.

Line 730 Wait for a keypress, then continue.

Line 740 Since an error occurred, halt program execution.

Line 760 The end of error_handler().

Running program 5

1. Clear the computer CRT and type in the program.
2. Press **[ALT]** **[R]** **[G]** on the computer.
3. The source frequency is set immediately to 2 GHz and the computer begins reading input A (reflection) of the analyzer and printing the measurements. After 21 readings, the program ends.

Program 6: trace transfer

One feature that sets the HP 8757D/E apart is its ability to transfer an entire measurement trace to a computer at very high speed. A complete, high–resolution (0.01 dB) 401–point measurement can be sent to the computer in 35 milliseconds (binary format) or 800 milliseconds (ASCII format). Transfer time will be less for fewer points per trace, and greater for more points per trace.

The analyzer gives you complete flexibility when reading measurement traces via HP–IB. You can read from the active channel and you can read the stored memory trace, the current measurement trace, or the normalized trace (measurement–minus–memory). In addition, the memory trace may be written back to the analyzer, allowing you to save and restore calibration traces via HP–IB.

With trace transfer measurements, some frequency resolution is sacrificed for measurement speed. The number of points per trace can be programmed to control the resolution across the frequency range being swept. If you are measuring a device that changes very rapidly with frequency, it is possible to miss very narrowband responses that occur between measurement points if the resolution is low. For these cases, the measurement should be made at a higher resolution. The Trace Transfer method of measurement is much faster than CW point–by–point measurements.

Program 6 listing

```
  10: /*  HP8757D/E QuickC IPG Program6   */
  20:
  30: #include <time.h>
  40: #include <string.h>
  50: #include <graph.h>
  60: #include <stdio.h>
  70: #include <cfunc.h>
  80: #include <chpib.h>
  90:
 100: int IOOUTPUTS_CHK (long hpib_adr, char
      *cmd_str);
 110: void disp_prompt (void);
 120: void error_handler (int error_no, char
      *routine);
 130:
 140: main ()
 150: {
 160:    char    endline [2],
 170:            cmd [80];
 180:    long    isc =7,
 190:            sna =716,
 200:            start_time, stop_time, the_time;
 210:    int     error, elements, i,
 220:            num_pts =401,
 230:            binary_dat [401];
 240:    float   ascii_dat [401];
 250:
 260:    endline [0] =13;     /*    cr    */
 270:    endline [1] =10;     /*    lf    */
 280:
 290:    _clearscreen (_GCLEARSCREEN);
 300:
 310:    error = IOTIMEOUT (isc,10.0);
 320:    error_handler (error, "IOTIMEOUT");
 330:    error = IOABORT (isc);
 340:    error_handler (error, "IOABORT");
 350:    error = IOCLEAR (isc);
 360:    error_handler (error, "IOCLEAR");
 370:
 380:    error = IOOUTPUTS_CHK (sna, "IP");
 390:    error = IOOUTPUTS_CHK (sna, "C1IA;C2IB");
 400:    start_time = time (&the_time);
 410:    do
 420:       {
 430:          stop_time = time (&the_time);
 440:       }
 450:    while (stop_time - start_time <2);
 460:
 470:    error = IOOUTPUTS_CHK (sna, "FD2;C1OD");
 480:    elements = num_pts;
 490:    error = IOENTERA (sna, ascii_dat,
         &elements);
 500:    error_handler (error, "IOENTERA");
 510:    error = IOEOL (isc, endline, 0);
 520:    error_handler (error, "IOEOL");
 530:    error = IOOUTPUTS_CHK (sna, "C1WM");
 540:    error = IOEOL (isc, endline,2);
 550:    error_handler (error, "IOEOL");
 560:    elements = num_pts;
 570:    error = IOOUTPUTA (sna, ascii_dat,elements);
 580:    error_handler (error, "IOOUTPUTA");
 590:    error = IOOUTPUTS_CHK (sna, "C1MY");
 600:    disp_prompt ();
 610:
 620:    error = IOOUTPUTS_CHK (sna, "C1C0;C2MY");
 630:    error = IOOUTPUTS_CHK (sna, "FD3;C2OD");
 640:    elements =2 * num_pts;
 650:    error = IOENTERB (sna, binary_dat,
         &elements,1);
 660:    error_handler (error, "IOENTERB");
 670:    error = IOEOL (isc, endline, 0);
 680:    error_handler (error, "IOEOL");
 690:    error = IOOUTPUTS_CHK (sna, "C2WM");
 700:    error = IOEOL (isc, endline,2);
 710:    error_handler (error, "IOEOL");
 720:    elements =2 * num_pts;
 730:    error = IOOUTPUTB (sna, binary_dat,
         elements,1);
 740:    error_handler (error, "IOOUTPUTB");
 750:    disp_prompt ();
 760:
 770:    for (i = 0; i < num_pts; i = i+1)
 780:       {
 790:          binary_dat [i] = i %100;
 800:       }
 810:    error = IOOUTPUTS_CHK (sna, "C2C0;C1MY");
 820:    error = IOEOL (isc, endline, 0);
 830:    error_handler (error, "IOEOL");
 840:    error = IOOUTPUTS_CHK (sna, "FD3;C1WM");
 850:    error = IOEOL (isc, endline,2);
 860:    error_handler (error, "IOEOL");
 870:    elements =2 * num_pts;
 880:    error = IOOUTPUTB (sna, binary_dat,
         elements,1);
 890:    error_handler (error, "IOOUTPUTB");
 900:    error = IOOUTPUTS_CHK (sna, "AS");
 910: }
 920:
 930: int IOOUTPUTS_CHK (long hpib_adr, char
      *cmd_str)
 940:    {
 950:       int    length, error_no;
 960:
 970:       length = strlen (cmd_str);
 980:       error_no = IOOUTPUTS (hpib_adr,
            cmd_str, length);
 990:       error_handler (error_no,
            "IOOUTPUTS_CHK");
1000:       return error_no;
1010:    }
1020:
1030: void disp_prompt (void)
1040:    {
1050:       char   ch;
1060:
1070:       settextposition (25,1);
1080:       printf ("Press <ENTER> to continue
            \n");
1090:       ch = getche ();
1100:       _clearscreen (_GCLEARSCREEN);
1110:    }
1120:
1130: void error_handler (int error_no, char
      *routine)
1140:    {
1150:       char   ch;
1160:
1170:       if (error_no != NOERR)
1180:       {
1190:        printf ("Error in call to %s\n",
             routine);

1200:        printf ("     Error = %d : %s
             \n", error_no, errstr (error_no));

1210:        printf ("Press <ENTER> to
             continue\n");
1220:        ch = getche ();
1230:        exit (1);
1240:       }
1250:    }
```

Program 6 explanation

Line 30 Tell the compiler which file includes information on time functions.

Line 40 Tell the compiler which file includes information on string functions.

Line 50 Tell the compiler which file includes information on _clearscreen() and _settextposition().

Line 60 Tell the compiler which file includes information on printf().

Line 70 Tell the compiler which file includes information on the HP–IB Command Library I/O functions.

Line 80 Tell the compiler which file includes information on the HP–IB Command Library error constants and errstr().

Line 100 Function prototype for the IOOUTPUTS_CHK() routine.

Line 110 Function prototype for the disp_prompt() routine.

Line 120 Function prototype for the error_handler() routine.

Line 140 Define the beginning of the main() routine.

Line 160 Define a string variable for the HP–IB command end–of–line string.

Line 170 Define a string variable for the output commands.

Line 180 Define a variable and assign it a value for the interface select code.

Line 190 Define a variable and assign it a value for the HP–IB address of the analyzer.

Line 200 Define variables for the start, stop, and present time.

Line 210 Define variables for the HP–IB Command Library error status, the number of elements in an array, and a loop counter.

Line 220 Define a variable and assign it a value for the number of trace points on the analyzer. By using a variable here it helps to make the program easily adaptable to different numbers of trace points.

Line 230 Define an array to hold a trace of 401 points in binary format.

Line 240 Define an array to hold a trace of 401 points in ASCII format.

Line 260 Define the end–of–line string as a carriage return and linefeed.

Line 290 Clear the computer CRT.

Line 310 Define a system timeout of 10 seconds.

Line 320 Perform error trapping.

Line 330 Abort any HP–IB transfers.

Line 340 Perform error trapping.

Line 350 Clear the analyzer's HP–IB interface.

Line 360 Perform error trapping.

Line 380 Preset the analyzer and the source. This sets the number of points per trace to 401.

Line 390 Set channel 1 to reflection (input A) and channel 2 to transmission (input B).

Line 400 Set the start time using the time function.

Line 410 Start of do loop.

Line 430 Get the present time.

Line 450 Loop until 2 seconds have elapsed from the start time.

Line 470 Set the data format to Extended ASCII and command the analyzer to output the channel 1 measurement data.

Line 480 Determine the number of elements to be read into the array.

Line 490 Read the measurement trace data from channel 1.

Line 500 Perform error trapping.

Line 510 Disable the end–of–line string (carriage return/linefeed) that is sent after any IOOUTPUT command.

Line 520 Perform error trapping.

Line 530 Command the analyzer to input data into the trace memory of channel 1.

Line 540 Enable the end–of–line string (carriage return/linefeed) that is sent after any IOOUTPUT command.

Line 550 Perform error trapping.

Line 560 Determine the number of elements in the array to be sent.

Line 570 Write the measured trace data back to the trace memory of channel 1. Reading the measurement trace and storing it back into trace memory is equivalent to executing the MEAS ––> MEM function (HP–IB command SM).

Line 580 Perform error trapping.

Line 590 Command channel 1 to display the trace memory data.

Line 600 Wait until **[ENTER]** is pressed to continue.

Line 620 Turn channel 1 off and channel 2 on. Command the analyzer to display the trace memory from channel 2.

Line 630 Set the data format to PC binary format. Command the analyzer to output its channel 2 measurement trace data.

Line 640 Determine the number of bytes used in the binary trace transfer.

Line 650 Read the binary measurement data from channel 2.

Line 660 Perform error trapping.

Line 670 Disable the end–of–line string (carriage return/linefeed) that is sent after any IOOUTPUT command.

Line 680 Perform error trapping.

Line 690 Command the analyzer to input data into the trace memory of channel 2.

Line 700 Enable the end–of–line string (carriage return/linefeed) that is sent after any IOOUTPUT command.

Line 710 Perform error trapping.

Line 720 Determine the number of bytes used in the binary trace transfer.

Line 730 Write the binary data array back to the trace memory of channel 2.

Line 740 Perform error trapping.

Line 750 Wait until **[ENTER]** is pressed to continue.

Line 770 Set up a loop to create 401 measurement points.

Line 790 Calculate some arbitrary function and fill the binary data array. This function has no particular meaning, but represents some special calibration data (such as an open/short average).

Line 800 End of the loop.

Line 810 Turn channel 2 off and display the channel 1 trace memory.

Line 820 Disable the end–of–line string (carriage return/linefeed) that is sent after any IOOUTPUT command.

Line 830 Perform error trapping.

Line 840 Command the analyzer to input data into the trace memory of channel 1.

Line 850 Enable the end–of–line string (carriage return/linefeed) that is sent after any IOOUTPUT command.

Line 860 Perform error trapping.

Line 870 Determine the number of bytes used in the binary trace transfer.

Line 880 Write the binary data array to the trace memory of channel 1.

Line 890 Perform error trapping.

Line 900 Autoscale the display on channel 1.

Line 910 The end of main().

Line 930 Define a routine that outputs string commands and performs error trapping. Define the types of variables passed to this routine: hpib_adr is the HP–IB address, cmd_str is the command string to output.

Line 950 Define variables for the length of the string and the error status.

Line 970 Determine the length of the command string.

Line 980 Output the command string.

Line 990 Perform error trapping.

Line 1000 Return the error status as the value of the routine.

Line 1010 The end of IOOUTPUTS_CHK().

Line 1030 Define a routine that prints a prompt on the computer CRT and waits for **[ENTER]** to be pressed.

Line 1050 Define a variable to hold the keypress.

Line 1070 Locate the text cursor at the beginning of row 25.

Line 1080 Print a prompt on the computer CRT.

Line 1090 Wait for a keypress, then continue.

Line 1100 Clear the computer CRT.

Line 1110 The end of disp_prompt().

Line 1130 Define a routine that checks the HP–IB Command Library error status. Define the types of variables passed to this routine: error_no is the error value, routine is the HP–IB Command Library routine called.

Line 1150 Define a variable to hold the keypress.

Line 1170 Test if an error actually occurred.

Line 1190 Yes, one did. Print on the computer CRT which HP–IB Command Library routine the error occurred in.

Line 1200 Print on the computer CRT the error number and a message.

Line 1210 Print a prompt on the computer CRT.

Line 1220 Wait for a keypress, then continue.

Line 1230 Since an error occurred, halt program execution.

Line 1250 The end of error_handler().

Running program 6

1. Clear the computer CRT and type in the program.
2. Press [ALT] [R] [G] on the computer.
3. Watching the analyzer CRT, you will see `DATA DUMP TO HP-IB` when it begins sending trace data to the computer, and `DATA DUMP TO TRACE MEMORY` when the computer sends the data back.
4. Watching the analyzer CRT, press **[ENTER]** on the computer. The computer again reads and writes a trace of data. The analyzer displays the same messages. This time the transfer occurs much more rapidly. A binary transfer takes about 35 milliseconds to be completed each way, while an ASCII transfer requires about 800 milliseconds each way.
5. Press **[ENTER]** on the computer. The computer calculates an arbitrary function and sends it to a trace memory of the analyzer, where it is autoscaled and displayed. This function has no significance. It represents a special calibration trace, such as an open/short average. With a computer, the analyzer measurement system may be calibrated over several different frequency ranges and changed from one to another very quickly, without recalibration.

If you wish to transfer a higher resolution trace with the HP 8757D, modify line 380 to be "IP SP801" for 801 points. Then modify the "401" in lines 220, 230, and 240 to "801".

Program 7: using the TAKE SWEEP command

The computer can detect this event in two ways:

- Monitor the status byte continuously until the bit is set (polling).
- Let the analyzer generate a service request (SRQ) and interrupt the computer.

Table1 is a diagram of the status bytes of the analyzer. It shows all of the bits that can be used to either monitor or interrupt the computer. Unfortunately, Microsoft QuickC is unable to automatically detect SRQ interrupts so the only approach available is to monitor the status byte. In this program, bit 4 (decimal value16) is used to signal "operation complete" (all of the sweeps specified by the TAKE SWEEP command have been completed).

When you follow the take sweep command with an output statement, such as OUTPUT DATA (OD), the data is sent immediately, not after the instructed number of sweeps. The approach mentioned overcomes this by letting us send the data at the end of the specified number of sweeps, not immediately. Another approach is to use the sweep hold mode (SW2) instead of the non−swept mode (SW0). In this mode the analyzer will prevent any HP−IB operations until the completion of the TAKE SWEEP command.

Table 1. HP 8757D/E Status Byte Descriptions

STATUS BYTE (#1)								
BIT #	7	6	5	4	3	2	1	0
Decimal Value	128	64	32	16	8	4	2	1
Function	N/A	Request Service (SRQ)	SRQ on HP–IB Syntax Error	SRQ on Operation Complete (Sweep, Plot or Print)	SRQ on Softkey Only Pressed	SRQ on Change in Extended Status Byte	SRQ on Numeric Entry Completed (HP–IB or Front Panel)	SRQ on Any Front Panel Key Pressed
EXTENDED STATUS BYTE (#2)								
BIT #	7	6	5	4	3	2	1	0
Decimal Value	128	64	32	16	8	4	2	1
Function	N/A	SRQ on Detector Uncal	SRQ on Front Panel Preset or Power–on	SRQ on Limit Test Failed[1]	SRQ on Action Requested not possible	SRQ on Knob Activity	SRQ on Operation Failed[1]	SRQ on Self Test Failure

[1] HP 8757D only.

Program 7 listing

```
 10: /*  HP8757D/E QuickC IPG Program7   */
 20:
 30: #include <string.h>
 40: #include <graph.h>
 50: #include <stdio.h>
 60: #include <cfunc.h>
 70: #include <chpib.h>
 80:
 90: int IOOUTPUTS_CHK (long hpib_adr, char
     *cmd_str);
100: void error_handler (int error_no, char
     *routine);
110:
120: main ()
130: {
140:    char    cmd [80];
150:    long    isc =7,
160:            sna =716,
170:            passthru =717;
180:    int     error, elements, status;
190:    float   ascii_dat [401];
200:
210:    _clearscreen (_GCLEARSCREEN);
220:
230:    error = IOTIMEOUT (isc,10.0);
240:    error_handler (error, “IOTIMEOUT”);
250:    error = IOABORT (isc);
260:    error_handler (error, “IOABORT”);
270:    error = IOCLEAR (isc);
280:    error_handler (error, “IOCLEAR”);
290:
300:    error = IOOUTPUTS_CHK (sna, “IP”);
310:    error = IOOUTPUTS_CHK (sna, “PT19;”);
320:    error = IOOUTPUTS_CHK (passthru,
        “ST250MS”);
330:    error = IOOUTPUTS_CHK (sna, “C2C0IB”);
340:    error = IOOUTPUTS_CHK (sna,
        “SW0;CS;RM16;”);
350:    error = IOOUTPUTS_CHK (sna, “TS10;”);
360:    do
370:       {
380:          error = IOSPOLL (sna, &status);
390:          error_handler (error, “IOSPOLL”);
400:       }
410:    while ((status &16) == 0);
420:    error = IOOUTPUTS_CHK (sna, “FD2;C1OD”);
430:    elements =401;
440:    error = IOENTERA (sna, ascii_dat,
        &elements);
450:    error_handler (error, “IOENTERA”);
460:    error = IOOUTPUTS_CHK (sna, “SW1”);
470: }
480:
490: int IOOUTPUTS_CHK (long hpib_adr, char
     *cmd_str)
500:    {
510:       int   length, error_no;
520:
530:       length = strlen (cmd_str);
540:       error_no = IOOUTPUTS (hpib_adr,
           cmd_str, length);
550:       error_handler (error_no,
           “IOOUTPUTS_CHK”);
560:       return error_no;
570:    }
580:
590: void error_handler (int error_no, char
     *routine)
600:    {
610:       char   ch;
620:
630:       if (error_no != NOERR)
640:       {
650:          printf (”Error in call to %s \n”,
            routine);
660:          printf (”     Error = %d : %s \n”,
              error_no, errstr (error_no));
670:          printf (”Press <ENTER> to
            continue\n”);
680:          ch = getche ();
690:          exit (1);
700:       }
710:    }
```

Program 7 explanation

Line 30 Tell the compiler which file includes information on string functions.

Line 40 Tell the compiler which file includes information on _clearscreen() and _settextposition().

Line 50 Tell the compiler which file includes information on printf().

Line 60 Tell the compiler which file includes information on the HP–IB Command Library I/O functions.

Line 70 Tell the compiler which file includes information on the HP–IB Command Library error constants and errstr().

Line 90 Function prototype for the IOOUTPUTS_CHK() routine.

Line 100 Function prototype for the error_handler() routine.

Line 120 Define the beginning of the main() routine.

Line 140 Define a string variable for the output commands.

Line 150 Define a variable and assign it a value for the interface select code.

Line 160 Define a variable and assign it a value for the HP–IB address of the analyzer.

Line 170 Define a variable and assign it a value for the analyzer's passthru address.

Line 180 Define variable for the HP–IB Command Library error status, the number of elements in an array, and the analyzer's status byte.

Line 190 Define an array to hold a trace of 401 points in ASCII format.

Line 210 Clear the computer CRT.

Line 230 Define a system timeout of 10 seconds.

Line 240 Perform error trapping.

Line 250 Abort any HP–IB transfers.

Line 260 Perform error trapping.

Line 270 Clear the analyzer's HP–IB interface.

Line 280 Perform error trapping.

Line 300 Preset the analyzer and source.

Line 310 Tell the analyzer which instrument is controlled through the passthru address (19 is the source).

Line 320 Set the source to 250 milliseconds per sweep.

Line 330 Turn off channel 2 of the analyzer and select transmission (input B) for display on channel 1.

Line 340 Put the analyzer into non–swept mode. Clear the status register of the analyzer. Set the request mask to 16 (bit 4) so that the analyzer will set bit 4 (operation complete) at the completion of the TAKE SWEEP command. Table1 has a description of all bits in the status bytes.

Line 350 Command the analyzer to take 10 sweeps.

Line 360 Start of do loop.

Line 380 Read the analyzer status byte.

Line 390 Perform error trapping.

Line 410 Wait for the 10 sweeps to be completed by testing the status byte to see if bit 4 is set. Remain in the loop until bit 4 (decimal16) is set.

Line 420 Set the data format to Extended ASCII and command the analyzer to output the channel 1 trace data.

Line 430 Define the maximum number of elements to be read into an array.

Line 440 Read the trace data.

Line 450 Perform error trapping.

Line 460 Return the analyzer to swept mode. The display now updates continuously.

Line 470 The end of main().

Line 490 Define a routine that outputs string commands and performs error trapping. Define the types of variables passed to this routine: hpib_adr is the HP–IB address, cmd_str is the command string to output.

Line 510 Define variables for the length of the string and the error status.

Line 530 Determine the length of the command string.

Line 540 Output the command string.

Line 550 Perform error trapping.

Line 560 Return the error status as the value of the routine.

Line 570 The end of IOOUTPUTS_CHK().

Line 590 Define a routine that checks the HP–IB Command Library error status. Define the types of variables passed to this routine: error_no is the error value, routine is the HP–IB Command Library routine called.

Line 610 Define a variable to hold the keypress.

Line 630 Test if an error actually occurred.

Line 650 Yes, one did. Print on the computer CRT which HP–IB Command Library routine the error occurred in.

Line 660 Print on the computer CRT the error number and a message.

Line 670 Print a prompt on the computer CRT.

Line 680 Wait for a keypress, then continue.

Line 690 Since an error occurred, halt program execution.

Line 710 The end of error_handler().

Running program 7

1. Clear the computer CRT and type in the program.
2. Press **[ALT] [R] [G]** on the computer.
3. The computer first presets the analyzer and source, then sets the source to 250 milliseconds per sweep, and the analyzer to display transmission on channel 1.
4. The computer commands the analyzer to take 10 sweeps and polls the analyzer status byte to determine when they were completed. The computer reads a trace from the analyzer. Just before the trace is sent, you should see the display "freeze" as the TAKE SWEEP command is completed.

To use the sweep hold mode, modify line 340 to "SW2;" (instead of "SW0;CS;RM16;") and delete lines 360, 370, 380, 390, 400, and 410. The program will wait at line 420 until the 10 sweeps are completed.

Program 8: programming the softkeys

The HP 8757D/E has eight screen–labeled softkeys that make measurements faster and easier for users. Under HP–IB control, you can re–label the softkeys with any annotation and sense when they are pressed.

Use the softkeys to branch to special measurement programs. By making full use of the softkeys, your automatic system may not need a normal computer keyboard at all, making it as easy to use as a manual instrument.

Program 8 listing

```
 10: /*  HP8757D/E QuickC IPG Program8   */
 20:
 30: #include <string.h>
 40: #include <graph.h>
 50: #include <stdio.h>
 60: #include <cfunc.h>
 70: #include <chpib.h>
 80:
 90: int IOOUTPUTS_CHK (long hpib_adr, char
     *cmd_str);
100: void error_handler (int error_no, char
     *routine);
110:
120: main ()
130: {
140:    char   cmd [80];
150:    long   isc =7,
160:           sna =716;
170:    int    error, elements,
180:           status, keycode;
190:    float  value;
200:
210:    _clearscreen (_GCLEARSCREEN);
220:
230:    error = IOTIMEOUT (isc,10.0);
240:    error_handler (error, "IOTIMEOUT");
250:    error = IOABORT (isc);
260:    error_handler (error, "IOABORT");
270:    error = IOCLEAR (isc);
280:    error_handler (error, "IOCLEAR");
290:
300:    error = IOOUTPUTS_CHK (sna, "IP");
310:    error = IOOUTPUTS_CHK (sna, "CS;RM8;");
320:    error = IOOUTPUTS_CHK (sna, "WK1 CAL1");
330:    error = IOOUTPUTS_CHK (sna, "WK2 TEST1");
340:    error = IOOUTPUTS_CHK (sna, "WK3 CAL2");
350:    error = IOOUTPUTS_CHK (sna, "WK4 TEST2");
360:    error = IOOUTPUTS_CHK (sna, "WK8 ABORT");
370:    printf ("SOFT KEYS LOADED\n");
380:
390:    do
400:      {
410:        status = 0;
420:        do
430:          {
440:            error = IOSPOLL (sna, &status);
450:            error_handler (error,"IOSPOLL");
460:          }
470:        while ((status &8) == 0);
480:        error = IOOUTPUTS_CHK (sna, "OK");
490:        error = IOENTER (sna, &value);
500:        error_handler (error, "IOENTER");
510:        keycode = value;
520:        _clearscreen (_GCLEARSCREEN);
530:        _settextposition (12,29);
540:        switch (keycode)
550:          {
560:            case32:
570:              printf ("Calibration #1\n");
580:              break;
590:            case8:
600:              printf ("Test #1\n");
610:              break;
620:            case 0:
630:              printf ("Calibration #2\n");
640:              break;
650:            case16:
660:              printf ("Test #2\n");
670:              break;
680:            case41:
690:              printf ("Abort\n");
700:              break;
710:            default:
720:              printf ("*** Undefined ***\n");
730:              break;
740:          }
750:        error = IOOUTPUTS_CHK (sna, "CS");
760:      }
770:    while (keycode !=41);
780: }
790:
800: int IOOUTPUTS_CHK (long hpib_adr, char
     *cmd_str)
810:    {
820:      int   length, error_no;
830:
840:      length = strlen (cmd_str);
850:      error_no = IOOUTPUTS (hpib_adr,
          cmd_str, length);
860:      error_handler (error_no,
          "IOOUTPUTS_CHK");
870:      return error_no;
880:    }
890:
900: void error_handler (int error_no, char
     *routine)
910     {
920:     char  ch;
930:
940:      if (error_no != NOERR)
950:        {
960:          printf ("Error in call to %s \n",
              routine);
970:          printf ("      Error = %d : %s \n",
              error_no, errstr (error_no));
980:          printf ("Press <ENTER> to
              continue\n");
990:          ch = getche ();
1000:         exit (1);
1010:       }
1020:     }
```

Program 8 explanation

Line 30 Tell the compiler which file includes information on string functions.

Line 40 Tell the compiler which file includes information on _clearscreen() and _settextposition().

Line 50 Tell the compiler which file includes information on printf().

Line 60 Tell the compiler which file includes information on the HP−IB Command Library I/O functions.

Line 70 Tell the compiler which file includes information on the HP−IB Command Library error constants and errstr().

Line 90 Function prototype for the IOOUTPUTS_CHK() routine.

Line 100 Function prototype for the error_handler() routine.

Line 120 Define the beginning of the main() routine.

Line 140 Define a string variable for the output commands.

Line 150 Define a variable and assign it a value for the interface select code.

Line 160 Define a variable and assign it a value for the HP−IB address of the analyzer.

Line 170 Define variables for the HP−IB Command Library error status and the number of elements in an array.

Line 180 Define variables for the analyzer's status byte and the keycode of the softkey pressed.

Line 190 Define a variable for reading the keycode value.

Line 210 Clear the computer CRT.

Line 230 Define a system timeout of 10 seconds.

Line 240 Perform error trapping.

Line 250 Abort any HP−IB transfers.

Line 260 Perform error trapping.

Line 270 Clear the analyzer's HP−IB interface.

Line 280 Perform error trapping.

Line 300 Preset the analyzer and source.

Line 310 Set the request mask to 8 (bit 3). See Table1 for the description of the status bytes.

Line 320 Label softkey 1 with "CAL1". Softkey 1 is the softkey at the top of the CRT.

Line 330 Label softkey 2 with "TEST1".

Line 340 Label softkey 3 with "CAL 2".

Line 350 Label softkey 4 with "TEST 2".

Line 360 Label softkey 8 with "ABORT".

Line 370 Print a message to the user.

Line 390 Start of do loop.

Line 410 Set status variable to zero.

Line 420 Start of do loop.

Line 440 Read the analyzer status byte.

Line 450 Perform error trapping.

Line 470 Wait for a softkey to be pressed by testing the status byte to see if bit 3 is set. Remain in the loop until bit 3 (decimal 8) is set.

Line 480 Command the analyzer to output the key code of the last key pressed.

Line 490 Read the key code.

Line 500 Perform error trapping.

Line 510 Make the key code an integer value.

Line 520 Clear the computer CRT.

Line 530 Move the text cursor to row 12, column 29, on the computer CRT.

Line 540 Multi−way branch on key code value.

Line 560 If the key code is 32, then softkey 1 was pressed.

Line 570 Print an appropriate message on the computer CRT.

Line 580 Exit the switch statement.

Line 590 If the key code is 8, then softkey 2 was pressed.

Line 600 Print an appropriate message on the computer CRT.

Line 610 Exit the switch statement.

Line 620 If the key code is 0, then softkey 3 was pressed.

Line 630 Print an appropriate message on the computer CRT.

Line 640 Exit the switch statement.

Line 650 If the key code is 16, then softkey 4 was pressed.

Line 660 Print an appropriate message on the computer CRT.

Line 670 Exit the switch statement.

Line 680 If the key code is 41, then softkey 8 was pressed.

Line 690 Print an appropriate message on the computer CRT.

Line 700 Exit the switch statement.

Line 710 If the key code doesn't match any of the preceding codes, another key was pressed. In this case, the key code has to be for softkey 5, 6, or 7 (key codes 14, 38, or 40) since these are the only other keys that impact the analyzer's status byte.

Line 720 Print an appropriate message on the computer CRT.

Line 730 Exit the switch statement.

Line 740 End of multi–way branch.

Line 750 Command the analyzer to clear the status byte.

Line 770 Wait for the "Abort" softkey to be pressed by testing the key code to see if is 41. Remain in the loop until this is true.

Line 780 The end of main().

Line 800 Define a routine that outputs string commands and performs error trapping. Define the types of variables passed to this routine: hpib_adr is the HP–IB address, cmd_str is the command string to output.

Line 820 Define variables for the length of the string and the error status.

Line 840 Determine the length of the command string.

Line 850 Output the command string.

Line 860 Perform error trapping.

Line 870 Return the error status as the value of the routine.

Line 880 The end of IOOUTPUTS_CHK().

Line 900 Define a routine that checks the HP–IB Command Library error status. Define the types of variables passed to this routine: error_no is the error value, routine is the HP–IB Command Library routine called.

Line 920 Define a variable to hold the keypress.

Line 940 Test if an error actually occurred.

Line 960 Yes, one did. Print on the computer CRT which HP–IB Command Library routine the error occurred in.

Line 970 Print on the computer CRT the error number and a message.

Line 980 Print a prompt on the computer CRT.

Line 990 Wait for a keypress, then continue.

Line 1000 Since an error occurred, halt program execution.

Line 1020 The end of error_handler().

Running program 8

1. Clear the computer CRT and type in the program.
2. Press **[ALT]** **[R]** **[G]** on the computer.
3. After the computer presets the analyzer and the source, it writes the softkey labels on the analyzer CRT. When the first key label is written, the analyzer labels it and blanks the other softkey labels. Since all labels except softkeys 5, 6, and 7 are given new labels, softkeys 5, 6, and 7 remain blank.
4. Press any key on the analyzer. Pressing a softkey causes a message to be printed on the CRT of the computer. Note that softkeys 5, 6, and 7 generate an interrupt, even though they weren't labeled. No other keys of the analyzer generate an interrupt, because of the SRQ mask specified.

Because the analyzer was left in remote mode, it didn't respond to any keys pressed on its front panel. In some applications it is useful to put the analyzer into local operation, so that it can be controlled from the front panel and still generate interrupts whenever a key is pressed.

Program 9: CRT graphics

For applications requiring diagrams, drawings, or special limit lines, the CRT of the analyzer may be used as a plotter.

This program draws a connection diagram for a hypothetical test system measuring an amplifier. It will blank the analyzer's standard display containing the graticule, annotation, and softkeys so that we have a blank CRT. Figure 2 shows what the CRT should look like when the program is done.

For fast, easy–to–use graphics, the graphics memory of the HP 8757D/E is divided into seven "pages" of 500 words. One vector requires two words. Each of the pages may be selected to receive data, and turned on and off independently. You can keep different drawings in each of the graphics memory pages and simply turn on the drawing you need by turning on the appropriate page. Each page may also be erased independently.

To use the graphics capability of the HP 8757D/E, first define the passthru address to be one less than the analyzer's control address. If the analyzer's address is16, its graphics address is 15. To the computer, the CRT of the analyzer looks like a plotter connected to the 8757 SYSTEM INTERFACE.

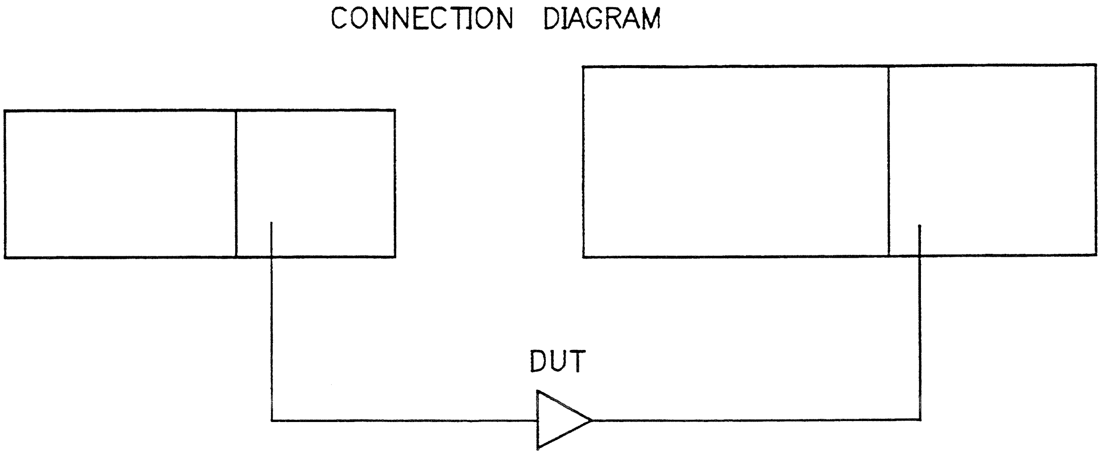


Figure 2. The CRT Graphics Display.

Program 9 listing

```
/*   HP8757D/E QuickC IPG Program9   */

#include <string.h>
#include <graph.h>
#include <stdio.h>
#include <cfunc.h>
#include <chpib.h>

int IOOUTPUTS_CHK (long hpib_adr, char
   *cmd_str);
void error_handler (int error_no, char
   *routine);

main ()
{
  char     cmd [80];
  long     isc =7,
           sna =716,
           passthru =717;
  int      error, col, row;

  _clearscreen (_GCLEARSCREEN);

  error = IOTIMEOUT (isc,10.0);
  error_handler (error, "IOTIMEOUT");
  error = IOABORT (isc);
  error_handler (error, "IOABORT");
  error = IOCLEAR (isc);
  error_handler (error, "IOCLEAR");

  error = IOOUTPUTS_CHK (sna, "IP BL5
 PT15;");
  error = IOOUTPUTS_CHK (passthru,
  "EP;GP1,1;DF;");
  error = IOOUTPUTS_CHK (passthru, "SP9;");

  for (col = 0; col <=29; col = col +1)
    {
       sprintf (cmd,
     "PU;PA%d,0;PD;PA%d,2000;", col *
          100, col *100);
       error = IOOUTPUTS_CHK (passthru, cmd);
    }
  for (row = 0; row <=20; row = row +1)
    {
       sprintf (cmd, "PU;PA 0,%d;PD;PA
       2900,%d;", row *100, row *100);
       error = IOOUTPUTS_CHK (passthru, cmd);
    }
  error = IOOUTPUTS_CHK (passthru, "SP1");
  error = IOOUTPUTS_CHK (passthru, "PU;PA
     600,1600;PD");
  error = IOOUTPUTS_CHK (passthru,
  "SI0.28,0.34;LBCONNECTION DIAGRAM\3");
  error = IOOUTPUTS_CHK (passthru, "PU;PA
 1200,250;PD");
  error = IOOUTPUTS_CHK (passthru,
  "SI0.28,0.34;LBDUT\3");
  error = IOOUTPUTS_CHK (passthru, "PU;PA
 300,800;PD;PA 1100,800,1100,1100,300,
 1100,300,800");
  error = IOOUTPUTS_CHK (passthru, "PU;PA
 800,800;PD;PA800,1100");
  error = IOOUTPUTS_CHK (passthru, "PU;PA
 1500,800;PD;PA2300,800,2300,1200,1500,
 1200,1500,800");
  error = IOOUTPUTS_CHK (passthru, "PU;PA
 1950,800;PD;PA1950,1200");
  error = IOOUTPUTS_CHK (passthru, "PU;PA
 875,850;PD;PA875,500,1200,500");
530:  error = IOOUTPUTS_CHK (passthru, "PU;PA
     1400,500;PD;PA2050,500,2050,850");
540:  error = IOOUTPUTS_CHK (passthru, "PU;PA
     1200,400;PD;PA1400,500,1200,600,1200,400");
550:  error = IOOUTPUTS_CHK (passthru, "PU;PA
     0,0");
560:  error = IOCLEAR (sna);
570:  error_handler (error, "IOCLEAR");
580: }
590:
600: int IOOUTPUTS_CHK (long hpib_adr, char
     *cmd_str)
610:  {
620:     int    length, error_no;
630:
640:     length = strlen (cmd_str);
650:     error_no = IOOUTPUTS (hpib_adr,
         cmd_str, length);
660:     error_handler (error_no, "IOOUTPUTS_CHK");
670:     return error_no;
680:  }
690:
700: void error_handler (int error_no, char
     *routine)
710:  {
720:     char   ch;
730:
740:     if (error_no != NOERR)
750:     {
760:       printf ("Error in call to %s \n",
          routine);
770:       printf ("       Error = %d : %s \n",
           error_no, errstr (error_no));
780:       printf ("Press <ENTER> to
          continue\n");
790:       ch = getche ();
800:       exit (1);
810:     }
820:  }
```

Program 9 explanation

Line 30 Tell the compiler which file includes information on string functions.

Line 40 Tell the compiler which file includes information on _clearscreen() and _settextposition().

Line 50 Tell the compiler which file includes information on printf().

Line 60 Tell the compiler which file includes information on the HP–IB Command Library I/O functions.

Line 70 Tell the compiler which file includes information on the HP–IB Command Library error constants and errstr().

Line 90 Function prototype for the IOOUTPUTS_CHK() routine.

Line 100 Function prototype for the error_handler() routine.

Line 120 Define the beginning of the main() routine.

Line 140 Define a string variable for the output commands.

Line 150 Define a variable and assign it a value for the interface select code.

Line 160 Define a variable and assign it a value for the HP–IB address of the analyzer.

Line 170 Define a variable and assign it a value for the analyzer's passthru address.

Line 180 Define variables for the HP–IB Command Library error status, the CRT column and row.

Line 200 Clear the computer CRT.

Line 220 Define a system timeout of 10 seconds.

Line 230 Perform error trapping.

Line 240 Abort any HP–IB transfers.

Line 250 Perform error trapping.

Line 260 Clear the analyzer's HP–IB interface.

Line 270 Perform error trapping.

Line 290 Preset the analyzer and blank the CRT display. Define the CRT graphics as the target of passthru commands. The CRT graphics address is always one less than the analyzer's HP–IB address.

Line 300 Erase all graphics pages. Turn graphics page 1 on to ensure that the graphics start in it. Set the color selection to default monochrome colors.

Line 310 Select to plot with pen 9, the lowest intensity for the analyzer CRT.

Line 330 Loop 30 times to draw the vertical part of the grid.

Line 350 Create a formatted output by printing the HP–GL plotter commands to a string.

Line 360 Draw a vertical line down the CRT.

Line 370 End of the loop.

Line 380 Loop 21 times to draw the horizontal part of the grid.

Line 400 Create a formatted output by printing the HP–GL plotter commands to a string.

Line 410 Draw a horizontal line across the CRT.

Line 420 End of the loop.

Line 430 Select to plot with pen 1, the brightest intensity for the analyzer CRT.

Line 440 Move the pen to title the display.

Line 450 Specify the width and height of each character, indicate what the title is, terminate the title with an end–of–text character (decimal 3).

Line 460 Move the pen to label the DUT.

Line 470 Specify the width and height of each character, indicate what the title is, terminate the title with an end–of–text character (decimal 3).

Line 480 Move the pen and draw the outline of the source.

Line 490 Draw the plug–in of the source.

Line 500 Move the pen and draw the outline of the analyzer.

Line 510 Draw the CRT of the analyzer.

Line 520 Draw the connections from the source to the DUT.

Line 530 Draw the connections from the DUT to the analyzer.

Line 540 Draw the DUT (an amplifier).

Line 550 Move to the bottom left corner of the CRT.

Line 560 Exit passthru mode by clearing the analyzer's HP–IB interface.

Line 570 Perform error trapping.

Line 580 The end of main().

Line 600 Define a routine that outputs string commands and performs error trapping. Define the types of variables passed to this routine: hpib_adr is the HP–IB address, cmd_str is the command string to output.

Line 620 Define variables for the length of the string and the error status.

Line 640 Determine the length of the command string.

Line 650 Output the command string.

Line 660 Perform error trapping.

Line 670 Return the error status as the value of the routine.

Line 680 The end of IOOUTPUTS_CHK().

Line 700 Define a routine that checks the HP–IB Command Library error status. Define the types of variables passed to this routine: error_no is the error value, routine is the HP–IB Command Library routine called.

Line 720 Define a variable to hold the keypress.

Line 740 Test if an error actually occurred.

Line 760 Yes, one did. Print on the computer CRT which HP–IB Command Library routine the error occurred in.

Line 770 Print on the computer CRT the error number and a message.

Line 780 Print a prompt on the computer CRT.

Line 790 Wait for a keypress, then continue.

Line 800 Since an error occurred, halt program execution.

Line 820 The end of error_handler().

Running program 9

1. Clear the computer CRT and type in the program.
2. Press **[ALT]** **[R]** **[G]** on the computer.
3. After the analyzer and source are preset, the CRT will be blanked. First a grid is plotted on the CRT. While this isn't necessary for our connection diagram, it does give you a good indication of where the X and Y coordinates are on the analyzers' CRT.
4. The labeling is added. The labels "CONNECTION DIAGRAM" and "DUT" are done using the analyzer CRT's internal character set.
5. All of the lines are plotted on the analyzer's CRT. If brighter lines are desired, draw each line twice or, select a different pen number.

In this example, only graphics page 1 was used. You can independently control up to 7 separate pages of graphics information. If you write too much information into one page, it overflows onto the next page.

When a graphics page is selected, the first location of memory that receives information is reset to the beginning of the page. Thus, as information is written into the page, the old information is destroyed. If we were plotting a line, this would appear as a new trace overwriting an old one.

Program 10: learning the instrument state

Being able to save a specific instrument state is helpful when it is needed several times in a test or measurement procedure. You can save the instrument state by manually logging the important analyzer and source parameters, such as start/stop frequency, sweep time, number of trace points, scale per division, and display format, then replace them at the appropriate time. A simpler approach is to save the instrument state in one of the nine internal save/recall registers of the analyzer/source combination, then recall it when needed.

The HP–IB user has two additional options: the interrogate function and the learn string. With the output interrogated parameter function (OP), you can selectively interrogate the values of all functions that have numeric values (such as frequency and number of trace points). This function operates the same way in both the analyzer and the source. It is illustrated in program 3 where the source start and stop frequencies are interrogated in lines 330 through 400.

A more thorough approach is to use the learn string functions of the analyzer and source. The learn string describes the present instrument state and is similar to one of the internal save/recall registers. For the analyzer, the Learn String also includes all of the global parameters, but not limit line information. Once an instrument state is learned, the analyzer and source states can be restored at any time. The following program demonstrates how to learn and restore the instrument states of the analyzer and HP 8350B Sweeper by using their learn string functions. If using the HP 8340B, 8341B, or 8360 Series Synthesized Sweepers, perform the modification described at the end of "Running program 10." If using an HP 8757E, note the program changes to lines 160 and 210 under Program 10 explanation.

Program 10 listing

```
 10: /*   HP8757D/E QuickC IPG Program10    */
 20:
 30: #include <string.h>
 40: #include <graph.h>
 50: #include <stdio.h>
 60: #include <cfunc.h>
 70: #include <chpib.h>
 80:
 90: int IOOUTPUTS_CHK (long hpib_adr, char
          *cmd_str);
100: void error_handler (int error_no, char
          *routine);
110:
120: main ()
130: {
140:    char     ch, match,
150:             cmd [160],
160:             lsna [301], lswpr [91];
170:    long     isc =7,
180:             sna =716,
190:             passthru =717;
200:    int      error, elements,
210:             maxsna =300,
220:             maxswpr =90;
230:
240:    match =10;          /*   lf    */
250:    _clearscreen (_GCLEARSCREEN);
260:
270:    error = IOTIMEOUT (isc,10.0);
280:    error_handler (error, "IOTIMEOUT");
290:    error = IOABORT (isc);
300:    error_handler (error, "IOABORT");
310:    error = IOCLEAR (isc);
320:    error_handler (error, "IOCLEAR");
330:
340:    error = IOOUTPUTS_CHK (sna, "IP");
350:    error = IOOUTPUTS_CHK (sna, "PT19;");
360:    error = IOLOCAL (sna);
370:    error_handler (error, "IOLOCAL");
380:    printf ("Set up system, Press
          <ENTER>\n");
390:    ch = getche ();
400:
410:    error = IOOUTPUTS_CHK (sna, "OL");
420:    error = IOMATCH (isc, match, 0);
430:    error_handler (error, "IOMATCH");
440:    elements = maxsna;
450:    error = IOENTERS (sna, lsna, &elements);
460:    error_handler (error, "IOENTERS");
470:    error = IOOUTPUTS_CHK (passthru, "OL");
480:    elements = maxswpr;
490:    error = IOENTERS (passthru, lswpr,
      &elements);
500:    error_handler (error, "IOENTERS");
510:    error = IOMATCH (isc, match,1);
520:    error_handler (error, "IOMATCH");
530:    error = IOOUTPUTS_CHK (sna, "IP");
540:    printf ("To restore system, Press
        <ENTER>\n");
550:    ch = getche ();
560:
570:    elements = maxsna +2;
580:    strcpy (cmd, "IL");
590:    memcpy (&cmd[2], lsna, maxsna);
600:    error = IOOUTPUTS (sna, cmd, elements);
610:    error_handler (error, "IOOUTPUTS");
620:    elements = maxswpr +2;
630:    strcpy (cmd, "IL");
640:    memcpy (&cmd[2], lswpr, maxswpr);
650:    error = IOOUTPUTS (passthru, cmd,
        elements);
660:    error_handler (error, "IOOUTPUTS");
670:    error = IOCLEAR (sna);
680:    error_handler (error, "IOCLEAR");
690:    error = IOLOCAL (sna);
700:    error_handler (error, "IOLOCAL");
710: }
720:
730: int IOOUTPUTS_CHK (long hpib_adr, char
     *cmd_str)
740:    {
750:      int    length, error_no;
760:
770:      length = strlen (cmd_str);
780:      error_no = IOOUTPUTS (hpib_adr,
          cmd_str, length);
790:      error_handler (error_no,
          "IOOUTPUTS_CHK");
800:      return error_no;
810:    }
820:
830: void error_handler (int error_no, char
     *routine)
840:    {
850:      char   ch;
860:
870:      if (error_no != NOERR)
880:      {
890:        printf ("Error in call to %s \n",
           routine);
900:        printf ("      Error = %d : %s \n",
            error_no, errstr (error_no));
910:        printf ("Press <ENTER> to
           continue\n");
920:        ch = getche ();
930:        exit (1);
940:      }
950:    }
```

Program 10 explanation

Line 30 Tell the compiler which file includes information on string functions.

Line 40 Tell the compiler which file includes information on _clearscreen() and _settextposition().

Line 50 Tell the compiler which file includes information on printf().

Line 60 Tell the compiler which file includes information on the HP–IB Command Library I/O functions.

Line 70 Tell the compiler which file includes information on the HP–IB Command Library error constants and errstr().

Line 90 Function prototype for the IOOUTPUTS_CHK() routine.

Line 100 Function prototype for the error_handler() routine.

Line 120 Define the beginning of the main() routine.

Line 140 Define variables to hold the keypress and the HP–IB Command Library match character.

Line 150 Define a string variable for the output commands.

Line 160 Define string variables for the analyzer and source learn strings. Make sure the dimensioned length is one more than the number of bytes in the learn string to retain the end–of–string null character (decimal 0). If using an HP 8757E, change lsna [301] to lsna [151].

Line 170 Define a variable and assign it a value for the interface select code.

Line 180 Define a variable and assign it a value for the HP–IB address of the analyzer.

Line 190 Define a variable and assign it a value for the analyzer's passthru address.

Line 200 Define variables for the HP–IB Command Library error status and the number of elements in an array.

Line 210 Define a variable and assign it a value for the maximum number of characters in the analyzer learn string. If using an HP 8757E, change maxsna = 300 to maxsna = 150.

Line 220 Define a variable and assign if a value for the maximum number of characters in the source learn string.

Line 240 Define the HP–IB Command Library match character as a linefeed.

Line 250 Clear the computer CRT.

Line 270 Define a system timeout of 10 seconds.

Line 280 Perform error trapping.

Line 290 Abort any HP–IB transfers.

Line 300 Perform error trapping.

Line 310 Clear the analyzer's HP–IB interface.

Line 320 Perform error trapping.

Line 340 Preset the analyzer and the source.

Line 350 Tell the analyzer which device is controlled through the passthru address. Address 19 belongs to the source.

Line 360 Set the analyzer and source to local mode.

Line 370 Perform error trapping.

Line 380 Prompt the user to set up the system.

Line 390 Wait until **[ENTER]** is pressed to continue.

Line 410 Program the analyzer to output its learn string.

Line 420 Disable character matching for the linefeed. The analyzer learn string is 300 contiguous binary bytes (150 for the HP 8757E) that does not end with a cr/lf (since this could actually be part of the learn string information).

Line 430 Perform error trapping.

Line 440 Determine the number of elements to be read.

Line 450 Read the analyzer learn string into the string "lsna".

Line 460 Perform error trapping.

Line 470 Program the source to output its learn string.

Line 480 Determine the number of elements to be read.

Line 490 Read the source learn string into the string "lswpr". The computer must read the entire source learn string which, for the HP 8350B Sweeper, is 90 bytes long.

Line 500 Perform error trapping.

Line 510 Enable character matching; this results in termination on a linefeed when a string is read.

Line 520 Perform error trapping.

Line 530 Preset the analyzer and source to clear the instrument states.

Line 540 Prompt the user to restore the system.

Line 550 Wait until **[ENTER]** is pressed to continue.

Line 570 Determine the number of elements to be sent (add 2 for the "IL" prefix).

Line 580 Start the learn string with the "IL" command.

Line 590 Concatenate the analyzer's binary learn string to the "IL" command. Remember that cmd[2] is the third element in this string (cmd[0] is the first). Since the learn string may contain nulls (decimal 0), "strcpy" cannot be used as it will stop at the first null. "memcpy" does not have this limitation.

Line 600 Program the analyzer to accept its learn string, then send it. Because "strlen" has the same problems as "strcpy", you cannot use the IOOUTPUTS_CHK routine here.

Line 610 Perform error trapping.

Line 620 Determine the number of elements to be sent (add 2 for the "IL" prefix).

Line 630 Start the learn string with the "IL" command.

Line 640 Concatenate the source's binary learn string to the "IL" command. Remember that cmd[2] is the third element in this string (cmd[0] is the first). Since the learn string may contain nulls (decimal 0), "strcpy" cannot be used as it will stop at the first null. "memcpy" does not have this limitation.

Line 650 Program the source to accept its learn string, then send it. Because "strlen" has the same problems as "strcpy", you cannot use the IOOUTPUTS_CHK routine here.

Line 660 Perform error trapping.

Line 670 Exit passthru mode by clearing the analyzer's HP–IB interface and continue sweeping.

Line 680 Perform error trapping.

Line 690 Set the analyzer and source to local mode.

Line 700 Perform error trapping.

Line 710 The end of main().

Line 730 Define a routine that outputs string commands and performs error trapping. Define the types of variables passed to this routine: hpib_adr is the HP–IB address, cmd_str is the command string to output.

Line 750 Define variables for the length of the string and the error status.

Line 770 Determine the length of the command string.

Line 780 Output the command string.

Line 790 Perform error trapping.

Line 800 Return the error status as the value of the routine.

Line 810 The end of IOOUTPUTS_CHK().

Line 830 Define a routine that checks the HP–IB Command Library error status. Define the types of variables passed to this routine: error_no is the error value, routine is the HP–IB Command Library routine called.

Line 850 Define a variable to hold the keypress.

Line 870 Test if an error actually occurred.

Line 890 Yes, one did. Print on the computer CRT which HP–IB Command Library routine the error occurred in.

Line 900 Print on the computer CRT the error number and a message.

Line 910 Print a prompt on the computer CRT.

Line 920 Wait for a keypress, then continue.

Line 930 Since an error occurred, halt program execution.

Line 950 The end of error_handler().

Running program 10

1. Clear the computer CRT and type in the program.
2. Press **[ALT] [R] [G]** on the computer.
3. When the computer stops and displays:

   ```
   SET UP SYSTEM, PRESS <ENTER>
   ```

   Adjust the analyzer and source to a preferred instrument state and press **[ENTER]** on the computer.

4. The computer will save the learn strings of both the analyzer and the source. After completing this, the analyzer and source will be preset to destroy your original instrument state.

5. When the computer stops and displays:

   ```
   TO RESTORE SETUP, PRESS <ENTER>
   ```

   Press the **[ENTER]** key. The computer will restore your original instrument state via the two learn strings. Verify on the displays of the analyzer and the source that your state has been restored.

This example is designed to work with the HP 8350B Sweeper, which has a learn string of 90 bytes. The program can be easily modified to work with the HP 8340B and 8341B Synthesized Sweepers which have learn strings 123 bytes in length. To do this, change the following lines to be:

```
160: lsna [301], lswpr [124];
220: maxswpr =123;
```

To work with the HP 8360 Series Synthesized Sweeper, the modifications are more extensive due to its variable length learn string. To do this, change and/or add the following lines:

```
150:                  cmd [701],
160:                  lsna [301], lswpr [701];
161:   unsigned       char    lswpr0 [4];
220:                  maxswpr = 700;
471:   elements = 3;
472:   error = IOENTERS (passthru, lswpr0,
       &elements);
473:   error_handler (error, "IOENTERS");
474:   maxswpr = 256 * lswpr0 [1] + lswpr0 [2];
620:   elements = maxswpr + 5;
631:   memcpy (&cmd[2], lswpr0, 3);
640:   memcpy (&cmd[5], lswpr, maxswpr);
```

The following should explain the above actions:

Line 150 Define a string variable large enough to hold the HP 8360 learn string.

Line 160 Define another string variable large enough to hold the HP 8360 learn string. Presently, the HP 8360 learn string is 605 bytes long but allow for some potential growth.

Line 161 Define a string variable to hold the header portion of the HP 8360 learn string (3 bytes). Make it an unsigned char array so that the value of each character ranges from 0 to 255 decimal (vs. −128 to +127).

Line 472 Read the 3 header bytes. Bytes 2 and 3 indicate the number of bytes to follow.

Line 473 Perform error trapping.

Line 474 Compute the number of bytes to follow (for the remainder of the HP 8360 learn string) and change maxswpr to reflect this.

Line 620 Determine the number of elements to be sent (add 2 for the "IL" prefix and 3 for the header bytes).

Line 631 Concatenate the HP 8360 header bytes to the "IL" command. Remember that cmd[2] is the third element in this string (cmd[0] is the first). Since the learn string may contain nulls (decimal 0), "strcpy" cannot be used as it will stop at the first null. "memcpy" does not have this limitation.

Line 640 Concatenate the remainder of the HP 8360 learn string. Remember that cmd[5] is the sixth element in this string.

Program 11: guided instrument setup with CRT graphics

As was illustrated by program 9, it is possible to utilize the CRT of the HP 8757D/E as a plotter. This program goes one step further by utilizing the CRT to create a simple connection diagram which may be recalled by the user, at any time, from the front panel of the analyzer.

This program draws the same hypothetical connection diagram as was drawn by program 9. It will blank most of the analyzer's standard display including the graticule and all annotation except the softkeys. In addition it will add one softkey under both the save and the recall hardkey menus. This softkey will allow the user to toggle the state of the CRT graphics off and on.

To use the graphics off/on capability of the HP 8757D/E, change "BL5" in line 310 of program 9 to "BLA", and make the necessary changes in the size of the background grid. These and other changes are illustrated in the following listing.

The same principle can be used to save anything stored to disk on the HP 8757D in the first seven pages of user graphics. By having the softkeys available, the user can store CRT graphics onto a disk for later recall.

Program 11 listing

```
 10: /*   HP8757D/E QuickC IPG Program11   */
 20:
 30: #include <string.h>
 40: #include <graph.h>
 50: #include <stdio.h>
 60: #include <cfunc.h>
 70: #include <chpib.h>
 80:
 90: int IOOUTPUTS_CHK (long hpib_adr, char
     *cmd_str);
100: void error_handler (int error_no, char
     *routine);
110:
120: main ()
130: {
140:    char    cmd [80];
150:    long    isc =7,
160:            sna =716,
170:            passthru =717;
180:    int     error, col, row;
190:
200:    _clearscreen (_GCLEARSCREEN);
210:
```

```
220:   error = IOTIMEOUT (isc,10.0);
230:   error_handler (error, “IOTIMEOUT”);
240:   error = IOABORT (isc);
250:   error_handler (error, “IOABORT”);
260:   error = IOCLEAR (isc);
270:   error_handler (error, “IOCLEAR”);
280:
290:   error = IOOUTPUTS_CHK (sna, “IP BLA
      PT15;”);
300:   error = IOOUTPUTS_CHK (passthru,
       “EP;GP1,1;DEC;”);
310:   error = IOOUTPUTS_CHK (passthru, “SP6;”);
320:
330:   for (col = 0; col <=25; col = col +1)
340:      {
350:        sprintf (cmd, “PU;PA%d,0;PD;
          PA%d,2000;”, col * 100, col *100);
360:        error = IOOUTPUTS_CHK (passthru, cmd);
370:      }
380:   for (row = 0; row <=20; row = row +1)
390:      {
400:        sprintf (cmd, “PU;PA 0,%d;PD;PA
            2500,%d;”, row *100, row *100);
410:        error = IOOUTPUTS_CHK (passthru,cmd);
420:      }
430:   error = IOOUTPUTS_CHK (passthru, “SP8”);
440:   error = IOOUTPUTS_CHK (passthru, “PU;PA
      600,1600;PD”);
450:   error = IOOUTPUTS_CHK (passthru,
       “SI0.28,0.34; LBCONNECTION DIAGRAM\3”);
460:   error = IOOUTPUTS_CHK (passthru, “PU;PA
      1200,250;PD”);
470:   error = IOOUTPUTS_CHK (passthru,
       “SI0.28,0.34;LBDUT\3”);
480:   error = IOOUTPUTS_CHK (passthru, “PU;PA
      300,800;PD;PA 1100,800,1100,1100,300,1100,
     300,800”);
490:   error = IOOUTPUTS_CHK (passthru, “PU;PA
      800,800;PD;PA800,1100”);
500:   error = IOOUTPUTS_CHK (passthru, “PU;PA
      1500,800;PD;PA 2300,800,2300,1200,1500,
      1200,1500,800”);
510:   error = IOOUTPUTS_CHK (passthru, “PU;PA
      1950,800;PD;PA1950,1200”);
520:   error = IOOUTPUTS_CHK (passthru, “PU;PA
      875,850;PD;PA875,500,1200,500”);
530:   error = IOOUTPUTS_CHK (passthru, “PU;PA
      1400,500;PD;PA2050,500,2050,850”);
540:   error = IOOUTPUTS_CHK (passthru, “PU;PA
      1200,400;PD;PA 1400,500,1200,600,1200,400”);
550:   error = IOOUTPUTS_CHK (passthru, “PU;PA
      0,0”);
560:   error = IOCLEAR (sna);
570:   error_handler (error, “IOCLEAR”);
580:
590:   error = IOLOCAL (sna);
600:   error_handler (error, “IOLOCAL”);
610: }
620:
630: int IOOUTPUTS_CHK (long hpib_adr, char
     *cmd_str)
640:    {
650:       int    length, error_no;
660:
670:       length = strlen (cmd_str);
680:       error_no = IOOUTPUTS (hpib_adr,
           cmd_str, length);
690:       error_handler (error_no,
           “IOOUTPUTS_CHK”);
700:       return error_no;
710:    }
720:
730: void error_handler (int error_no, char
     *routine)
740:    {
750:       char   ch;
760:
770:       if (error_no != NOERR)
780:       {
790:          printf (”Error in call to %s \n”,
              routine);
800:          printf (”       Error = %d : %s \n”,
              error_no, errstr (error_no));
810:          printf (”Press <ENTER> to
            continue\n”);
820:          ch = getche ();
830:          exit (1);
840:       }
850:    }
```

Program 11 explanation

Line 30 Tell the compiler which file includes information on string functions.

Line 40 Tell the compiler which file includes information on _clearscreen() and _settextposition().

Line 50 Tell the compiler which file includes information on printf().

Line 60 Tell the compiler which file includes information on the HP–IB Command Library I/O functions.

Line 70 Tell the compiler which file includes information on the HP–IB Command Library error constants and errstr().

Line 90 Function prototype for the IOOUTPUTS_CHK() routine.

Line 100 Function prototype for the error_handler() routine.

Line 120 Define the beginning of the main() routine.

Line 140 Define a string variable for the output commands.

Line 150 Define a variable and assign it a value for the interface select code.

Line 160 Define a variable and assign it a value for the HP–IB address of the analyzer.

Line 170 Define a variable and assign it a value for the analyzer's passthru address.

Line 180 Define variables for the HP–IB Command Library error status, CRT column and row.

Line 200 Clear the computer CRT.

Line 220 Define a system timeout of 10 seconds.

Line 230 Perform error trapping.

Line 240 Abort any HP–IB transfers.

Line 250 Perform error trapping.

Line 260 Clear the analyzer's HP–IB interface.

Line 270 Perform error trapping.

Line 290 Preset the analyzer and blank all the CRT display except the softkeys. Define the CRT graphics as the target of passthru commands. The CRT graphics address is always one less than the analyzer's HP–IB address.

Line 300 Erase all graphics pages. Turn graphics page 1 on to ensure that the graphics start in it. Set the color selection to default colors.

Line 310 Select to plot with pen 6 (white), the lowest intensity for the analyzer CRT.

Line 330 Loop 26 times to draw the vertical part of the grid.

Line 350 Create a formatted output by printing the HP-GL plotter commands to a string.

Line 360 Draw a vertical line down the CRT.

Line 370 End of the loop.

Line 380 Loop 21 times to draw the horizontal part of the grid.

Line 400 Create a formatted output by printing the HP-GL plotter commands to a string.

Line 410 Draw a horizontal line across the CRT.

Line 420 End of the loop.

Line 430 Select to plot with pen 8 (yellow), the brightest intensity for the analyzer CRT.

Line 440 Move the pen to title the display.

Line 450 Specify the width and height of each character, indicate what the title is, terminate the title with an end-of-text character (decimal 3).

Line 460 Move the pen to label the DUT.

Line 470 Specify the width and height of each character, indicate what the title is, terminate the title with an end-of-text character (decimal 3).

Line 480 Move the pen and draw the outline of the source.

Line 490 Draw the plug-in of the source.

Line 500 Move the pen and draw the outline of the analyzer.

Line 510 Draw the CRT of the analyzer.

Line 520 Draw the connections from the source to the DUT.

Line 530 Draw the connections from the DUT to the analyzer.

Line 540 Draw the DUT (an amplifier).

Line 550 Move to the bottom left corner of the CRT.

Line 560 Exit passthru mode by clearing the analyzer's HP-IB interface.

Line 570 Perform error trapping.

Line 590 Place the analyzer and the source in local mode.

Line 600 Perform error trapping.

Line 610 The end of main().

Line 630 Define a routine that outputs string commands and performs error trapping. Define the types of variables passed to this routine: hpib_adr is the HP-IB address, cmd_str is the command string to output.

Line 650 Define variables for the length of the string and the error status.

Line 670 Determine the length of the command string.

Line 680 Output the command string.

Line 690 Perform error trapping.

Line 700 Return the error status as the value of the routine.

Line 710 The end of IOOUTPUTS_CHK().

Line 730 Define a routine that checks the HP-IB Command Library error status. Define the types of variables passed to this routine: error_no is the error value, routine is the HP-IB Command Library routine called.

Line 750 Define a variable to hold the keypress.

Line 770 Test if an error actually occurred.

Line 790 Yes, one did. Print on the computer CRT which HP-IB Command Library routine the error occurred in.

Line 800 Print on the computer CRT the error number and a message.

Line 810 Print a prompt on the computer CRT.

Line 820 Wait for a keypress, then continue.

Line 830 Since an error occurred, halt program execution.

Line 850 The end of error_handler().

Running program 11

1. Clear the computer CRT and type in the program.

2. Press **[ALT] [R]** [G] on the computer.

3. After the analyzer and source are preset, the CRT is blanked, except for softkeys. First a grid is plotted on the CRT. While this isn't necessary for our connection diagram, it does give you a good indication of where the X and Y coordinates are on the analyzers' CRT.

4. The labeling is added. The labels "CONNECTION DIAGRAM" and "DUT" are written using the analyzer CRT's internal character set.

5. All of the lines are plotted on the analyzer's CRT. If brighter lines are desired, draw each line twice or, select different pen numbers.

6. The analyzer is placed in local mode with the front panel and the softkeys active. To access the graphics on/off capability, press [SAVE] on the analyzer to show the save menu. Press the softkey labeled [STORE TO DISK]. Note the [GRAPHIC ON/OFF] softkey, it does not appear unless the "BLA" command is used. Press the [GRAPHIC ON/OFF] softkey so that it is "off". The connection diagram will disappear from the CRT display. Press the [GRAPHIC ON/OFF] softkey again and the diagram will reappear. If you store this setup to the external disk drive at this time, the analyzer will remember this graphics on/off mode later upon recall from disk.

Table 2. Alphabetical Listing of HP 8757D/E Programming Codes (1 of 3)

Code	Action	Code	Action
A0	Averaging off	CLS	Color list, salmon[1]
AB	A/B ratio measurement	CLW	Color list, white[1]
AC	A/C ratio measurement[2]	CLY	Color list, yellow[1]
AFd	Averaging on and factor d	CN	Cursor to minimum
ANm	Adaptive Normalization on/off	COBd	Brightness adjust, one color[1]
AR	A/R ratio measurement	COCd	Color adjust, one color[1]
AS	Autoscale	COTd	Tint adjust, one color[1]
AZ2	Autozero the DC detectors once	CR	C/R ratio measurement[2]
AZm	Autozero repeat on/off of the DC detectors	CS	Clear status bytes
BA	B/A ratio measurement	CTm	Auto system calibration on/off
BC	B/C ratio measurement[2]	CUm	Cursor on/off
BFm	Plotter buffer on/off[3]	CWm	CW mode on/off
BL0	Restore CRT to normal mode	CX	Cursor to maximum
BL1	Blank frequency labels (secure frequency mode, frequency labels cannot be restored)	DAd	Detector A amplitude offset set to d
BL2	Blank all labels	DBd	Detector B amplitude offset set to d
BL3	Blank active channel trace	DCd	Detector C amplitude offset set to d[2]
BL4	Blank softkey labels	DEC	Set default colors[1]
BL5	Blank all (except user CRT graphics)	DFA	Set disk format to ASCII[1]
BL6	Blank title	DFB	Set disk format to binary[1]
BL7	Blank mode labels	DFE	Set Disk format to extended binary[1]
BL8	Blank the active entry area	DHm	Display Hold on/off of the active channel trace
BL9	Blank the limit lines	DIAd	Set disk HP−IB address[1]
BLA	Blank all (except user CRT graphics and softkeys)	DIUd	Set disk unit number[1]
BR	B/R ratio measurement	DIVd	Set disk volume number[1]
BTNd	Overall display brightness	DLF	Delete file from disk[1]
BW	Display the search bandwidth on the CRT[1]	DM0	All inputs set to DC detection
C0	Channel off	DM1	All inputs set to AC detection
C1	Channel 1 on/active	DN	Step down (decrement)
C2	Channel 2 on/active	DOAd	Measure Detector A amplitude offset
C3	Channel 3 on/active[1]	DOBd	Measure Detector B amplitude offset
C4	Channel 4 on/active[1]	DOCd	Measure Detector C amplitude offset[2]
CA	C/A ratio measurement[2]	DORd	Measure Detector R amplitude offset
CB	C/B ratio measurement[2]	DRd	Detector R amplitude offset set to d
CC1	Set channel 1 color[1]	DS0	Display trace data in log magnitude
CC2	Set channel 2 color[1]	DS1	Display trace data in standing wave ratio (SWR) format
CC3	Set channel 3 color[1]	DTSTPAs	Enter stop frequency for detector A
CC4	Set channel 4 color[1]	DTSTPBs	Enter stop frequency for detector B
CDm	Cursor delta on/off	DTSTPCs	Enter stop frequency for detector C[2]
CGL	Set labels color[1]	DTSTPRs	Enter stop frequency for detector R
CGN	Set background color[1]	DTSTRAs	Enter start frequency for detector A
CGR	Set grid color[1]	DTSTRBs	Enter start frequency for detector B
CGW	Set warning label color[1]	DTSTRCs	Enter start frequency for detector C[2]
CL	Perform system configuration of detectors and channels	DTSTRRs	Enter start frequency for detector R
CLB	Color list, black[1]	EO	Enter measured detector amplitude offset
CLG	Color list, green[1]	ER0	Erase all save/recall registers
CLL	Color list, blue[1]	FAs	Start frequency label
CLR	Color list, red[1]	FBs	Stop frequency label
		FD0	Format data ASCII

1. HP 8757D only
2. HP 8757D Option 001 only
3. Revision 3.1 or above for HP 8757E.

Table 2. Alphabetical Listing of HP 8757D/E Programming Codes (2 of 3)

Code	Action
FD1	Format data binary (HP BASIC compatible)
FD2	Format data extended ASCII
FD3	Format data binary (PC compatible)
FD4	Format data extended binary (HP BASIC compatible)
FD5	Format data extended binary (PC compatible)
FR0	Logarithmic (dB) cursor format[3]
FR1	SWR cursor format[3]
FSm	Step sweep on/off[3,4]
FTAm	Detector A frequency on/off
FTBm	Detector B frequency on/off
FTCm	Detector C frequency on/off[2]
FTRm	Detector R frequency on/off
IA	Input A absolute power measurement
IB	Input B absolute power measurement
IC	Input C absolute power measurement[2]
ILs	Input Learn string
IND	Initialize disk format[1]
IP	Instrument preset
IR	Input R absolute power measurement
IX	External ADC input (AUX) voltage measurement1
LE	Erase limit lines for active channel[5]
LFA	Load instrument information file from disk[1]
LFC	Load CRT graphics file from disk[1]
LFD	Load data trace file from disk[1]
LFF	Load measurement file from disk.[1]
LFH	Load instrument information file from disk and place instrument in hold mode.[1]
LFI	Load instrument state file from disk[1]
LFM	Load memory trace file from disk[1]
LFN	Load display trace file from disk.[1]
LFs	Enter limit test flat line data[5]
LL	Store lower limit line into memory[5]
LPs	Enter limit test point data[5]
LSs	Enter limit test sloped line data[5]
LTm	Limit line test on/off[5]
LU	Store upper limit line into memory[5]
M−	Display normalized data (measurement − memory)
MDm	Modulation on/off
ME	Display measurement data
MM	Display the channel menu(main menu)
MN	Display normalized data (same as M−)
MOC	Monochrome display[1]
MR	Marker (or cursor) to reference line
MSm	Manual sweep mode on/off
MU0	Display the measurement menu
MU1	Display the display menu
MU2	Display the scale menu

Code	Action
MU3	Display the reference menu
MU4	Display the cursor menu
MU5	Display the average menu
MU6	Display the calibration menu
MU7	Display the special menu
MU8	Display the system menu
MY	Display memory data
MZ	Manual calibration of DC detectors
NSm	Non−standard sweep mode on/off
OC	Output cursor value
OD	Output trace data
OE1	Output error status of display channel 1
OE2	Output error status of display channel 2
OI	Output identity
OK	Output keycode of last key pressed
OL	Output learn string
OM	Output memory data
ON	Output normalized (measurement —memory) data
OPDO	Output measured detector amplitude offset
OPxx	Output interrogated parameter value xx= AF, BW, DA, DB, DC, DR, RL, RP, SD, SL, SO, SP, SR, SS, ST
OR	Output rotary knob value (—32768 ≤ value ≤ +32767)
OS	Output status bytes
OT1m	Control output #1 on/off
OT2m	Control output #2 on/off
OV	Output CW value
P1	Plot channel 1 trace on external plotter
P2	Plot channel 2 trace on external plotter
P3	Plot channel 3 trace on external plotter[1]
P4	Plot channel 4 trace on external plotter[1]
PA	Plot all on external plotter
PBm	System interface control on/off
PC	Plot labels on external plotter
PD	Plot custom plot
PG	Plot grid on external plotter
PR1	Print all to monochrome printer, except softkeys and CRT graphics
PR2	Print tabular display data in monochrome
PR3	Print tabular marker/cursor data to external printer
PR4	Print all to color printer, except softkeys and CRT graphics[1]
PTd	Passthrough address set to d
PWRA	Execute a detector A power calibration
PWRB	Execute a detector B power calibration
PWRC	Execute a detector C power calibration[2]
PWRR	Execute a detector R power calibration
R1	R/A ratio measurement
R2	R/B ratio measurement

1. HP 8757D only
2. HP 8757D Option 001 only
3. Revision 3.1 or above for HP 8757E.
4. HP 8340, HP 8341, or HP 8360 series synthesized sweeper only with 8757 SYSTEM INTERFACE connected and active.
5. Limit line functions valid only for channels 1 or 2. HP 8757D only.

Table 2. Alphabetical Listing of HP 8757D/E Programming Codes (3 of 3)

Code	Action	Code	Action
R3	R/C ratio measurement[2]	SR	Cursor search right[1]
RCn	Recall register n	SSd	Cursor search value set to d[1]
RLd	Reference level set to d	STd	Reference level step size set to d
RMd	Service request mask set to d	SUd	Specify custom plot according to d
RPq	Reference position set to vertical division q	SVn	Save register n
RS	Restart averaging	SW0	Non−swept mode; non−swept operation
SCd	Set cursor to horizontal position d	SW1	Swept mode; normal swept operation
SDd	Scale per division set to d	SW2	Sweep hold mode; non−swept mode with HP−IB bus hold off until completion of TSd
SFA	Store all instrument information to disk in file[1]	TCm	Continuous Temperature Compensation on/off
SFC	Store CRT graphics to disk in file[1]	TIFs	Title for file[1]
SFD	Store data trace to disk in file[1]	TSd	Take d sweeps, then hold display
SFI	Store instrument state to disk in file[1]	UP	Step up (increment)
SFM	Store memory trace to disk in file[1]	WKs	Write softkey label
SFN	Store normalized trace to disk in file[1]	WMs	Write to channel memory.
SKq	Select softkey q: q =1 to 8	WTs	Write title, s is an ASCII string of up to 50 characters
SL	Cursor search left[1]	XAs	External detector cal value for detector A
SM	Store measurement into memory	XBs	External detector cal value for detector B
SN	Store normalized data (measurement − memory) into memory	XCs	External detector cal value for detector C[2]
SOd	Smoothing set to d % of frequency span	XRs	External detector cal value for detector R
SPd	Number of points set to d: d=101, 201, 401, 801[1], 1601[1]		

[1] HP 8757D only

[2] HP 8757D Option 001 only (detector C)

NOTES:	n	=	decimal integer 1 to 9
	d	=	variable length numeric
	m	=	0 for off/1 for on
	q	=	unique value
	s	=	ASCII or binary string

For more information, call your local HP sales office listed in your telephone directory or an HP regional office listed below for the location of your nearest sales office.

United States:
Hewlett–Packard Company
4 Choke Cherry Road
Rockville, MD 20850
(301) 670–4300

Hewlett–Packard Company
5201 Tollview Drive
Rolling Meadows, IL 60008
(312) 255–9800

Hewlett–Packard Company
5161 Lankershim Blvd.
No. Hollywood, CA 91601
(818) 505–5600

Hewlett–Packard Company
2015 South Park Place
Atlanta, GA 30339
(404) 955–1500

Canada:

Hewlett–Packard Ltd.
6877 Goreway Drive
Mississauga, Ontario L4V1M8
(416) 678–9430

Australia/New Zealand:
Hewlett–Packard Australia Ltd.
31–41 Joseph Street,
Blackburn, Victoria 3130
Melbourne, Australia
(03) 895–2895

Europe/Africa/Middle East:
Hewlett–Packard S.A.
Central Mailing Department,
P.O. Box 529
1180 AM Amstelveen,
The Netherlands
(31) 20/547 9999

Far East:
Hewlett–Packard Asia Ltd.
22/F Bond Centre
West Tower
89 Queensway
Central, Hong Kong
(5) 8487777

Japan:
Yokogawa–Hewlett–Packard Ltd.
29–21, Takaido–Higashi 3–chome
Suginami–ku, Tokyo 168
(03) 331–6111

Latin America:
Latin American Region Headquarters
Monte Pelvoux Nbr. 111
Lomas de Chapultepec
11000 Mexico, D.F. Mexico
(905) 596–79–33

HEWLETT
PACKARD

July 15, 1992

HP Part Number 08757–90118
Printed in U.S.A.