Errata

Document Title: Using the 3585A Spectrum Analyzer With the 9825A Computing Controller (AN 246)

Part Number: 5952-8765

Revision Date: April 1978

HP References in this Application Note

This application note 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. We have made no changes to this application note copy. The HP XXXX referred to in this document is now the Agilent XXXX. For example, model number HP8648A is now model number Agilent 8648A.

About this Application Note

We've added this application note to the Agilent website in an effort to help you support your product. This manual provides the best information we could find. It may be incomplete or contain dated information, and the scan quality may not be ideal. If we find a better copy in the future, we will add it to the Agilent website.

Support for Your Product

Agilent no longer sells or supports this product. You will find any other available product information on the Agilent website:

www.agilent.com

Search for the model number of this product, and the resulting product page will guide you to any available information. Our service centers may be able to perform calibration if no repair parts are needed, but no other support from Agilent is available.

Agilent Technologies

TABLE OF CONTENTS

Introduction

This application note is written to help you understand the ease with which the HP 3585A and a computing controller, in this case the HP 9825A, can solve measurement problems not possible with analyzers alone. The instruments are interconnected using the HP-IB.* A companion note on the instrument, "Understanding the 3585A," is recommended for your reading.

Several programs are included which will probably relate closely to your own measurement situations. These are, of course, but examples and the 3585A/9825A combination offers a powerful solution to many additional problems. More detailed information on the programming codes to solve these problems can be obtained from either the 3585A Operating and Service Manual or HP 3585A Spectrum Analyzer Remote Operation.

Hardware

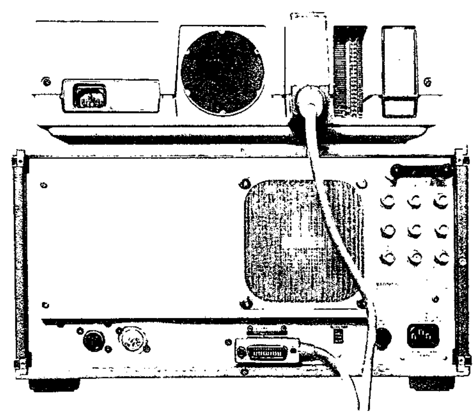
The standard 3585A is already equipped for HP-IB operation. You will need the following options for the 9825A:

HP 98034A—HP-IB interface card set to select code 7,

HP 98210A—String—Advanced Programming ROMs,

Any one of the General I/O-Extended I/O ROMs.

The system interconnections are shown below.



*The HP Interface Bus (HP-IB) is Hewlett-Packard's implementation of IEEE Standard 488-1975 and identical ANSI Standard MC1.1, "Digital Interface for Programmable Instrumentation."

Programming Basics

If you are not familiar with the HP 9825A, we suggest you obtain the following documents:

Hewlett-Packard 9825A Calculator Operating and Programming Manual—HP Part No. 09825-90000

Hewlett-Packard 9825A Calculator General I/O Programming Manual—HP Part No. 09825-90024

Hewlett-Packard 9825A Calculator Extended I/O Programming Manual—HP Part No. 09825-90025

It is helpful to have had some experience of calculator programming; however, with some background and the flow charts provided with each subroutine, the reader should have little difficulty in understanding these programs. They are written in HPL, the language of the 9825 Controller. Perhaps the least familiar things to the average reader are the I/O operations. The first thing to note about I/O operations is that the HP-IB address of the analyzer is 711. The device statement (dev) is used to make this the more recognizable code "3585" in most of the programs, so,

wrt "3585", "A"

means write to the 3585 the ASCII character "A" and,

red "3585", X

means read a number from the 3585 and put it in variable X. A variation you will see is,

wrt "3585.2",X

which means write to the 3585 the contents of the variable X in a form specified by format statement #2. (fmt 2)

The rest of the calculator language is quite easily readable, particularly with the aid of the flow charts.

Analyzer HP-IB Operation

There are four modes of HP-IB operation with the analyzer. These are:

1. Keyboard Programming — Every front panel key on the 3585A can be programmed by the HP-IB. The only controls which can not be programmed are the power switch, the CRT controls (intensity, gradicule illumination, astigmatism, focus) and the tracking generator output level.
2. Analyzer as a Terminal — The controller can place text on the analyzer CRT and can use the analyzer keys as a remote input. Thus the operator of an HP-IB controlled analyzer does not have to be near the controller.
3. Display Manipulation — The B trace can be transferred to the controller and both the A and B traces can be loaded from the controller. This means that the controller can process the analyzer results and replace them on the analyzer screen.
4. Service — Through special codes, many diagnostic messages and self-tests are available through the HP-IB. These, in conjunction with diagnostic routines in the controller, can easily isolate the faulty section of the instrument.

Of these four modes of HP-IB operation, only the first three will be covered. Diagnostic software is covered in the service manual. These HP-IB operations are implemented by the programming codes on the fold-out sheet in the appendix. It is recommended that the reader use this fold-out when studying the programs.

All the programs in this note are presented as subroutines. This is done to make it easy for the user to combine as many of these subroutines together as needed to solve his own measurement problems. However, one must realize that more is needed in an operating program than a series of subroutines. There must be a loader routine to initialize the calculator and analyzer and a user's program to connect the subroutines together in the desired order. Therefore, all the subroutines are presented as part of a program, complete with a loader routine and an example user's program.

The reader is reminded that these programs are not restricted to usage under their chapter heading. For instance, the use of the analyzer as a terminal is described under the Production Test Chapter, where it probably has the most usage. However, this capability could certainly be useful in other environments.

Most of all, the reader is encouraged to study these programs, understand how the analyzer is programmed to implement a desired measurement, and then write his own subroutines to solve his own measurement problem. To aid this, many suggestions on variations to these subroutines are given throughout the text.

Signal Analysis Subroutines

Level and Frequency Measurements

gsb "Measure"

This is a simple subroutine to illustrate how to program the spectrum analyzer and take readings back to the calculator. When calling the subroutine, the user places in the variable F the approximate frequency of the signal he wants to measure. The subroutine will return with the measured frequency and amplitude of the signal in the variables F and A respectively.

The flow chart describes in detail how the measurements are made under calculator control. The only major differences between this procedure and an operator's front panel keystrokes are that the calibration is disabled and the manual sweep mode is used. These are both done to speed up the subroutine. Under HP-IB control, the analyzer immediately recalibrates itself every time a parameter is changed which could affect the accuracy. While this insures that no accidentally uncalibrated readings are taken, it also means that unnecessary calibrations are done. For instance, unless the calibration were disabled, a calibration would occur between lines 18 and 19. Since we have not finished setting up the test conditions in the analyzer, we are not going to take a reading and calibration would be a waste of time.

The manual sweep mode is used because we are interested only in the amplitude at the center frequency. Allowing the analyzer to sweep the other frequencies would once again waste time.

A typical loader or start up routine precedes the subroutine and is also flow charted. It initializes the calculator and the analyzer as described in the flow chart.

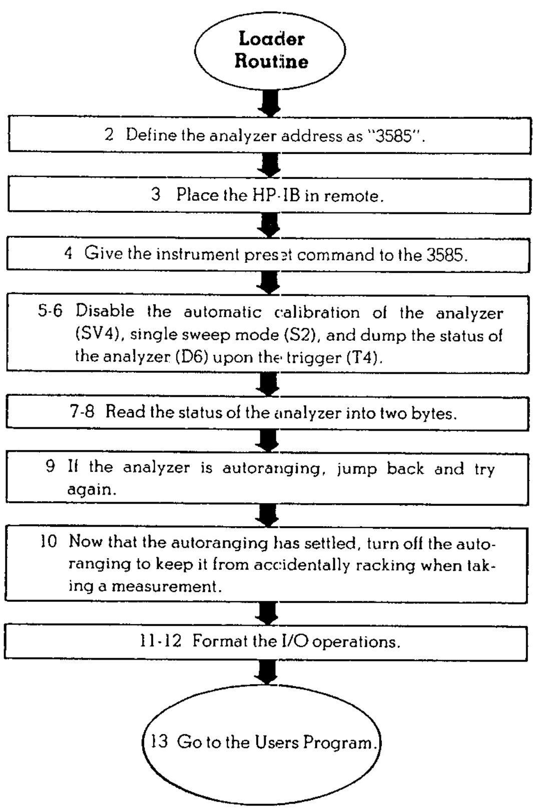
In addition, a typical user's program is shown in lines 29-33. A frequency synthesizer set to 1.0015 MHz was connected to the analyzer 50Ω input and the program was run. As one can see from the example output on the next page, the analyzer read the frequency to within 0.1 Hz.

While this routine is simple and straight forward, if the frequency of the desired signal isn't known very accurately, or if the unknown is too low in amplitude, then the routine may not work properly. This can happen because the counter in the analyzer doesn't get sufficient signal from the IF. The measure routine used in the "Distortion" subroutine in the next section minimizes these problems at the price of added complexity.

Program*

```
0: "3585 System's Application Note Software":
1:
2: dev "3585",711
3: rem 7
4: clr "3585"
5: wrt "3585","SV4S2D6"
6: wrt "3585","T4"
7: rdb("3585")→X
8: rdb("3585")→Y
9: if X#0 and X#2;jmp -3
10: wrt "3585","AR0"
11: fmt 1,"CF",f.1,"HZ"
12: fmt 2,"FS",f.1,"HZ"
13: gto "Start"
14:
15:
16: "Measure":
17: wrt "3585","SV4"
18: wrt "3585.1",F
19: wrt "3585.2",.2F
20: wrt "3585","S3,MK501,CN1,D1,T6"
21: red "3585",X
22: wrt "3585","MC,RC4,CN0,T6"
23: red "3585",X
24: wrt "3585","MR,D2,T6"
25: red "3585",F,A
26: ret
27:
28:
29: "Start":
30: 1e6→F
31: gsb "Measure"
32: prt "Frequency (Hz)",F,"Amplitude (dBm)",A
33: spc 2
*23362
```

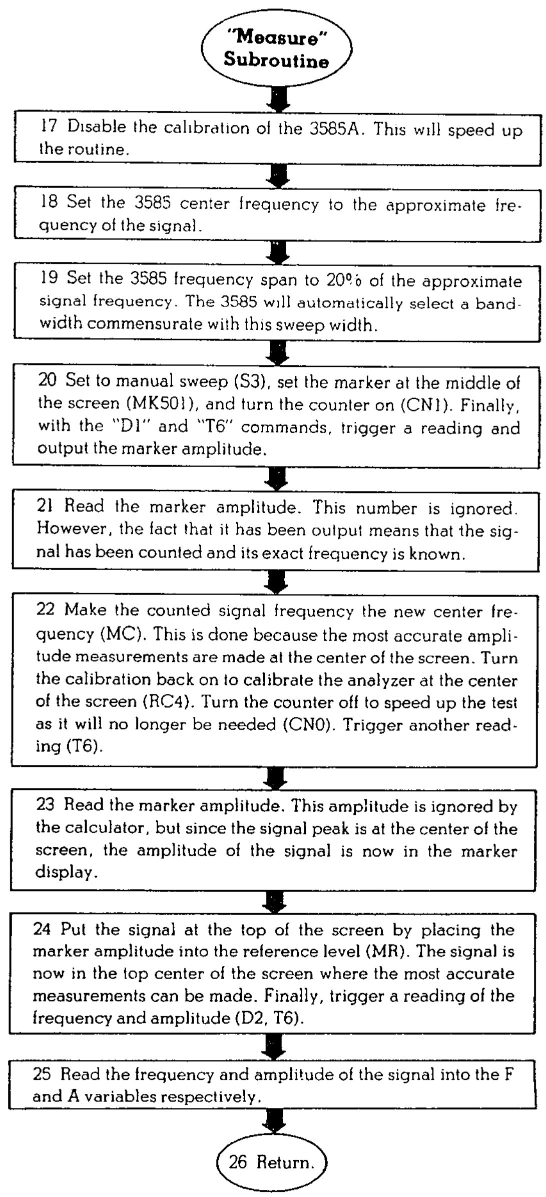
*The commas inside the quotation marks in the write statement are not needed for proper analyzer operation, but are inserted for readability.



User's Program Output

```
Frequency (Hz)
     1001500.10
Amplitude (dBm)
         -15.50
```

Variables Used	
A	Amplitude
F	Frequency
X	Working register
Y	Working register



Harmonic Distortion and THD Measurement

cll 'Distortion' (frequency, # of harmonics, % THD, amplitude of fundamental, amp of 2nd harmonic, amp of 3rd harmonic, . . .)

Here is a routine which uses the computing power of the controller to generate a result not normally directly available from a spectrum analyzer, total harmonic distortion (THD). When called, the programmer gives the subroutine the approximate frequency of the fundamental and the number of harmonics he wishes to measure. The subroutine returns the measured frequency and amplitude of the fundamental, percent THD, and the amplitude of each harmonic in dB below the fundamental.

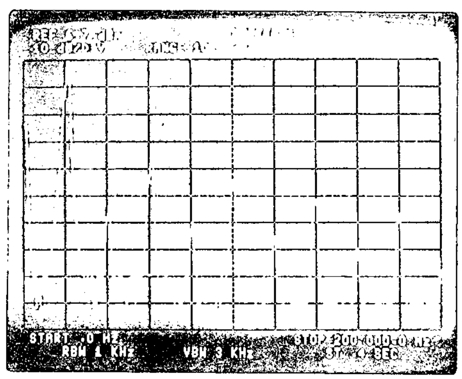
The program is straightforward and explained in the flow chart. THD is computed by the formula:

$$\%\,THD = 100\sqrt{\left(\frac{A_2}{A}\right)^2 + \left(\frac{A_3}{A}\right)^2 + \ldots + \left(\frac{An}{A}\right)^2}$$

where A is the fundamental amplitude
and An is the amplitude of the nth harmonic.

The start-up routine has been compressed into fewer lines than in the "Measure" example in the last section, but is otherwise the same. The "Measure" subroutine has a changed line, #30. If the amplitude measured is below -40 dBm, then the bandwidth of the analyzer is widened to allow for the fact that the signal may be far from the frequency the analyzer is tuned to. Enough of the signal may then get through the IF to allow the counter to work properly. If not, the bandwidth is again widened. Obviously, this routine will not work with signals below -40 dBm.

The two printouts from the example user's program show the harmonic distortion in a low cost audio oscillator and a high quantity synthesizer. The picture below shows a frequency sweep of the low cost oscillator and its actual harmonic content.



Audio Oscillator Spectrum

Program

```
0: "3585 System's Application Note Software":
1:
2: dev "3585",711;rem 7
3: clr "3585";wrt "3585","SV4S2I1D6"
4: wrt "3585","T4";rdb("3585")→X;rdb("3585")→Y;if X#0
   and X#2;jmp 0
5: wrt "3585","AR0"
6: fmt 1,"CF",f.1,"HZ"
7: fmt 2,"FS",f.1,"HZ"
8: gto "Start"
9:
10:
11: "Distortion":
12: p1→F
13: gsb "Measure"
14: F→p1;A→p4
15: wrt "3585","MS,OF1,MO,CN0"
16: 0→Y
17: for I=2 to p2
18: wrt "3585","CF,UP,D1,T6"
19: red "3585",X
20: 10^(X/10)+Y→Y
21: X→p(I+3)
22: next I
23: 100√Y→p3
24: ret
25: "Measure":
26: wrt "3585","SV4"
27: wrt "3585.1",F
28: wrt "3585.2",.2F
29: wrt "3585","S3,MK501,CN1,D1,T6"
30: red "3585",X;if X<-40;wrt "3585","RB,UP,D1,T6";gto -0
31: wrt "3585","MC,PR,RC4,T6"
32: red "3585",X
33: wrt "3585","MR,D2,T6"
34: red "3585",F,A
35: ret
36:
37:
38: "Start":
39: 1e6→F
40: cll 'Distortion'(F,5,T,A,r2,r3,r4,r5)
41: fxd 1;prt "Fundamental Freq",F,"Fund. Amp (dBm)",A
42: prt "THD (dB)      ",20log(T/100)
43: fxd 2;prt "% THD          ",T
44: spc 2
*3122
```

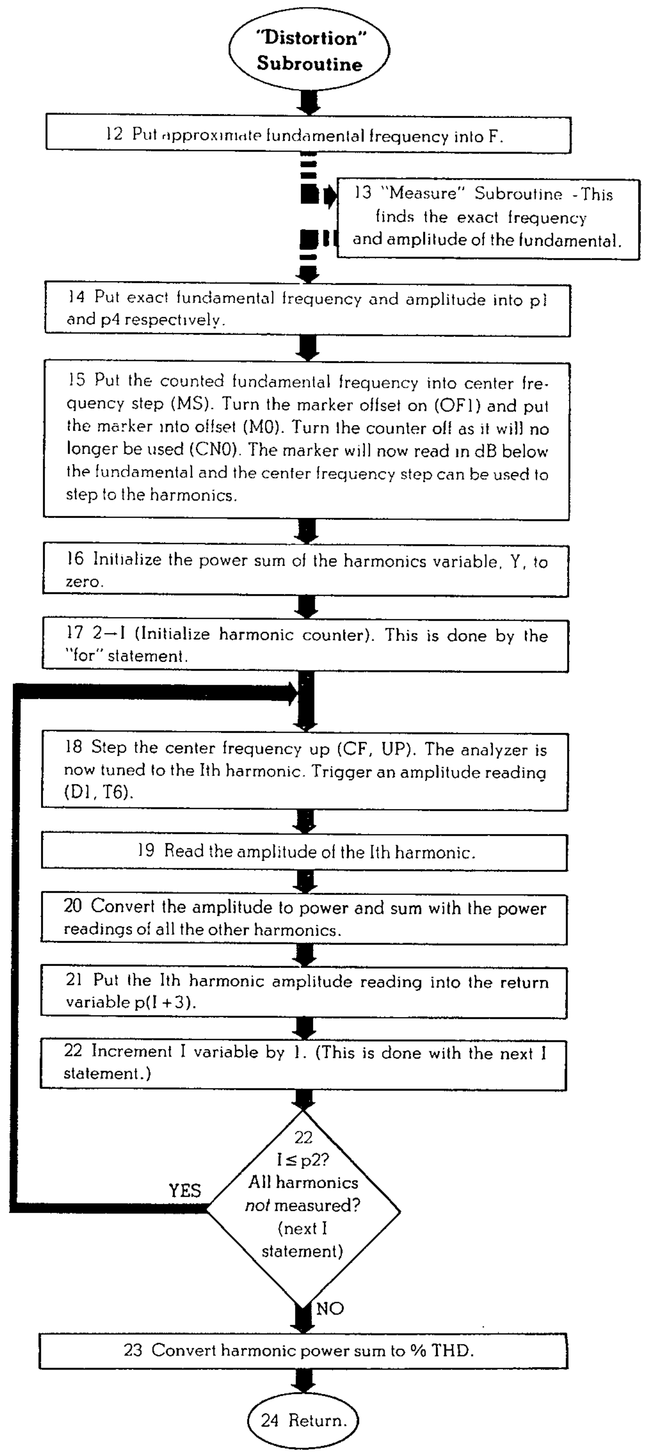
User's Program Output

```
Fundamental Freq
         20003.9
Fund. Amp (dBm)
             6.9
THD (dB)
           -33.8
% THD
            2.04


Fundamental Freq
       1001500.1
Fund. Amp (dBm)
           -16.1
THD (dB)
           -54.8
% THD
            0.18
```

Variables Used

A	Amplitude of fundamental
F	Frequency of fundamental
I	Harmonic number index
X	Working register
Y	Sum of harmonic powers



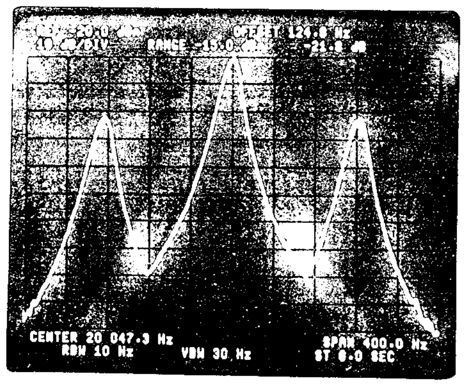
Amplitude Modulation Measurements

cll 'AM' (carrier frequency, modulation frequency, carrier amplitude, % modulation)

Here is yet another subroutine for making measurements beyond the scope of other spectrum analyzers that cannot be calculator controlled. The programmer gives the subroutine the approximate carrier and modulating frequencies and the subroutine returns the measured carrier and modulating frequencies, the carrier amplitude and the percent amplitude modulation.

The flow chart describes the subroutine operation which is very similar to the front panel manual operations one would do. The center frequency step command is used to tune the center frequency once the frequency of the carrier and upper sideband are measured. The percent modulation is calculated from the sideband levels in dB below the carrier.

There are many useful variations on this simple program which can be easily implemented. For instance, the levels of both sidebands (variables Y and Z) could be compared and if they were unequal, one would know that there was PM as well as AM present in the modulation. Another example would be to step beyond the modulation sideband and measure the second and third harmonic distortion of the modulation. Also, one could measure by using the center frequency step, the many sidebands of a high modulation index FM signal and compute the modulation index. The reader is encouraged to adapt this routine to these or your own measurement problems.



AM Signal Measured by Program

Program

```
0: "3585 System's Application Note Software":
1:
2: dev "3585",711
3: rem 7;clr "3585";wrt "3585","SV4S2I1D6"
4: wrt "3585","T4";rdb("3585")→X;rdb("3585")→Y;if X#0
   and X#2;jmp 0
5: wrt "3585","AR0"
6: fmt 1,"CF",f.1,"HZ"
7: fmt 2,"FS",f.1,"HZ"
8: fmt 3,"CS",f.1,"HZ"
9: gto "Start"
10:
11:
12: "AM":
13: wrt "3585","SV4,S3,BH1,RB30KZ"
14: wrt "3585.1",p1
15: wrt "3585","CN1,D2,T6"
16: red "3585",p1,X
17: wrt "3585","MC"
18: wrt "3585.2",p2
19: wrt "3585","BH0,PR,RC4,MR,D1,T6"
20: red "3585",p3
21: wrt "3585","OF1,M0"
22: wrt "3585.3",p2
23: wrt "3585","CF,UP,D2,T6"
24: red "3585",p2,X
25: wrt "3585","MC,D1,T6"
26: red "3585",Y
27: wrt "3585","MS,CF,DN,DN,D1,T6"
28: red "3585",Z
29: 100(10^(Y/20)+10^(Z/20))→p4
30: ret
31:
32:
33: "Start":
34: 2e4→C
35: 135→M
36: cll 'AM'(C,M,A,P)
37: fxd 1;prt "Carrier Freq",C,"Modulation Freq",M
38: prt "Carrier Amp (dB)",A,"% Modulation",P
39: spc 2
*19491
```

User's Program Output

```
Carrier Freq
         20004.0
Modulation Freq
           136.3
Carrier Amp (dB)
             6.9
% Modulation
            16.6
```

Variables Used	
X	Working register
Y	Upper sideband amplitude
Z	Lower sideband amplitude

"AM" Subroutine

13 Disable the calibration of the 3585 to speed up routine (SV4). Put in manual sweep mode (S3). Hold the analyzer in the 30 kHz bandwidth (BH1, RB30KZ).

14 Set the 3585 center frequency to the approximate carrier frequency.

15 Turn the counter function on (CN1) and trigger a frequency and amplitude reading (D2,T6).

16 The exact carrier frequency is put into return variable p1 and the amplitude reading is ignored.

17 Put the counted carrier frequency into the center frequency of the analyzer (MC).

18 Make the sweep width equal to the approximate modulation frequency.

19 Turn the bandwidth hold off and preset the bandwidth coupling (BH0, PR). The analyzer will now be in a bandwidth that can easily resolve the carrier and sidebands. Enable the calibration of the analyzer to improve the accuracy of the results (RC4). Put the marker (carrier) amplitude into the reference level (MR). The carrier can now be measured with the greatest accuracy since it is at the top center of the screen. Trigger an amplitude reading (D1,T6).

20 Read the carrier amplitude and place in return variable p3.

21 Turn the marker offset on and place the marker into the offset (OF1,M0). Now all marker readings will be in Hertz away from the carrier and dB below the carrier.

22 Set the center frequency step of the analyzer to the approximate modulating frequency.

23 Step the center frequency of the analyzer up (CF,UP). The analyzer will now be approximately tuned to the upper sideband frequency. Trigger a frequency and amplitude reading (D2,T6).

24 Read the frequency of the upper sideband and put in return variable p2. Since the marker offset and counter functions are on, this will be the exact value of the modulating frequency. Ignore the amplitude reading.

25 Place the upper sideband in the center of the screen for an accurate amplitude reading (MC). Trigger an amplitude reading (D1,T6).

26 Read the upper sideband level and place in variable Y. Because the marker offset is on, the reading will be in dB below the carrier.

27 Place the exact modulating frequency into the center frequency step of the analyzer (MS). Step the center frequency down twice to the lower sideband frequency (CF, DN, DN). Trigger an amplitude reading (D1,T6).

28 Read the lower sideband frequency in dB below the carrier and store in Z.

29 Convert the dB sideband levels into percent modulation and place the result in return variable p4.

30 Return.

Production Test Subroutines

Messages on Analyzer CRT

cll 'Generate Message'
cll 'Display Message'

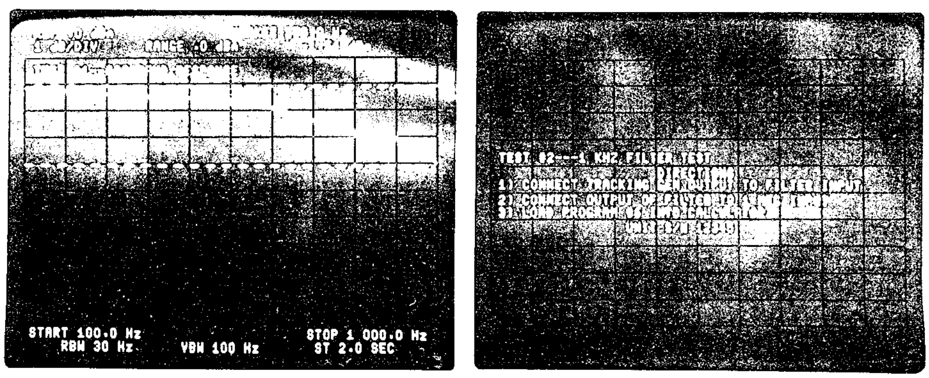
These subroutines are the first in a series that use the analyzer as a remote terminal for the controller. They allow the controller to communicate with the test technician and allow flexibility in test procedures as well as technician promoting. The photographs below show the typical kinds of messages which can be written on the CRT of the analyzer.

Since these programs manipulate alphanumeric data, string variables are used extensively. In the 9825A Calculator, the plug-in 98210A Advanced Programming-String ROM provides this capability. String variables allow the calculator to store and manipulate alphanumeric data much as if they were solely numeric. String variables are readily identified in a 9825A program by the "$" symbol which follows the variable (e.g., D$ is the D string variable).

The flow charts explain the subroutines in detail. One should note that the script display of six lines of up to 50 characters each totally replaces the graphics display on the CRT and both displays are not viewed at the same time. If one wanted to display the graphics and script display at the same time, one would use the "DC" instead of the "DS" command and turn on the B trace. However, the 50-character annotation line can be on while viewing the graphics trace A display as in the lower left photo. The example user's program (lines 29-33) flashes the annotation line on the graphics display with a period of about 1.4 seconds.

Also note that the analyzer will not accept all ASCII characters for text display. It does not accept lower case letters and some special ASCII commands like backspacing. See the HP-IB section of the operating manual for details. Generally this is not a problem for text displays.

The loader or start-up routine for this subroutine initializes only the calculator, not the analyzer. This is done so that whatever instrument settings were in the analyzer are retained and only the annotation and script display lines are changed.



Example Display Messages

Program

```
0: "3585 System's Application Note Software":
1:
2: dim A$[48],D$[100],L$[5,48]
3: dev "3585",711
4: rem 7
5: gto "Start"
6:
7:
8: "Generate Message":
9: ""→D$[1,48]
10: ent "Annotation line message?",D$
11: if len(D$)>48;prt "Line too long";spc 2;gto -1
12: cap(D$)→A$
13: for I=1 to 6
14: ""→L$[I,1,48]→D$[1,48]
15: fxd 0;dsp "Line ",I," of script display?";ent "",D$
16: if len(D$)>48;prt "Line too long.";spc 2;gto -1
17: cap(D$)→L$[I];next I
18: ret
19:
20: "Display Message":
21: fmt "LA",c;wrt "3585",A$
22: fmt c;wrt "3585","DAES"
23: for I=1 to 6
24: fmt "L",f.0,c;wrt "3585",I,L$[I]
25: next I
26: fmt c;wrt "3585","DS";ret
27:
28:
29: "Start":
30: cll 'Generate Message'
31: cll 'Display Message'
32: wrt "3585","DG";wait 700
33: wrt "3585","DA";wait 700;gto -1
*23758
```

"Generate Message" Subroutine

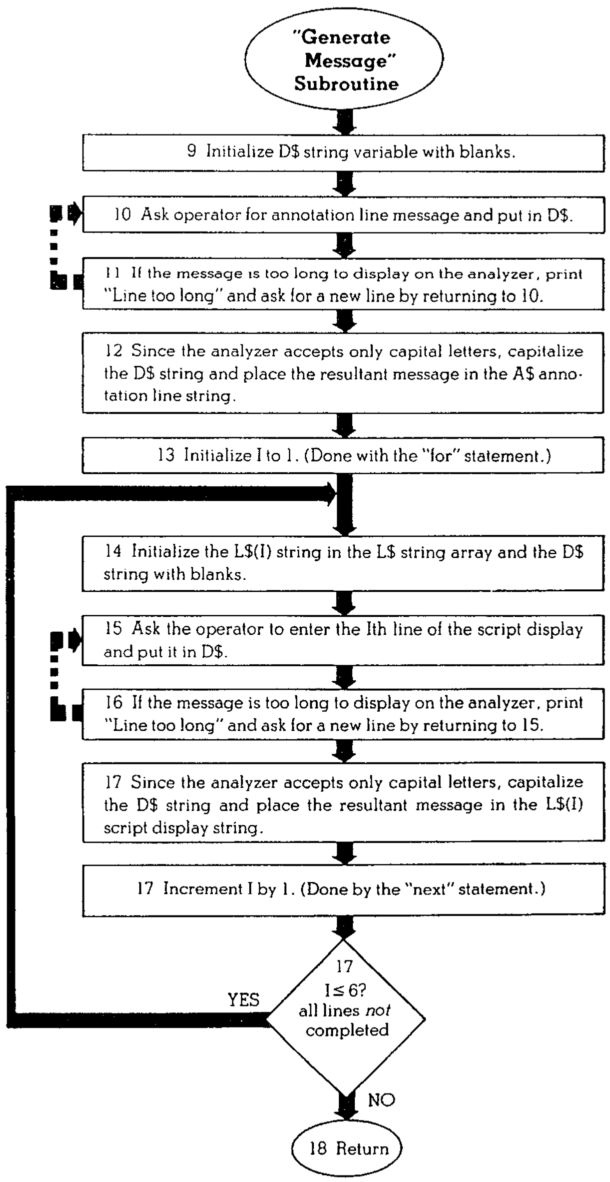
Variables Used

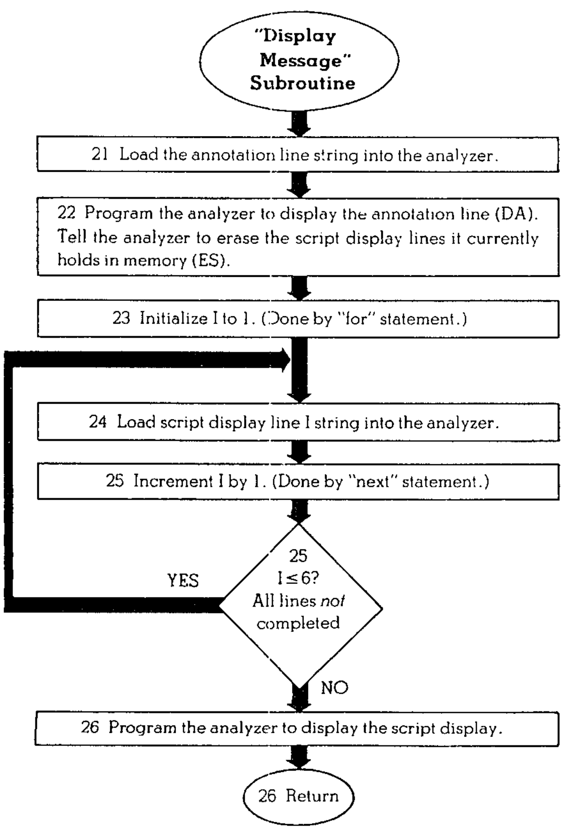
I	Script display line counter
A$	Annotation line string variable
D$	Keyboard entry string variable
L$(6)	Script display string variable array

"Display Message" Subroutine

Variables Used

I	Script display line counter
A$	Annotation line string variable
L$(6)	Script display string variable array





Learn Mode

cll 'Learn Mode'

Like many HP-IB instruments, the HP 3585A has learn mode capability. This means that the complete analyzer settings (i.e., center frequency, bandwidth, input range, etc.) can be output to the controller in 100 bytes of code. If at any time in the future it is desired to return to these instrument settings, the controller then outputs the code back to the instrument.

Thus, the learn mode is used much as the three save/recall registers in the analyzer. But in these subroutines, the 100 byte code is stored on the tape cartridge in the 9825A controller. Thus, the instrument settings are non-volatile, i.e., they can be recalled after the power has been off. This is not true of the save/recall registers in the analyzer. Also, the tape cartridge can hold many hundreds of instrument settings, whereas the analyzer is limited to three settings.

Both the learn mode and the instrument save/recall registers are useful because sweeps can be simply set up and then stored to be recalled with the touch of a button. This is much easier than writing all the HP-IB programming codes to set up a sweep.

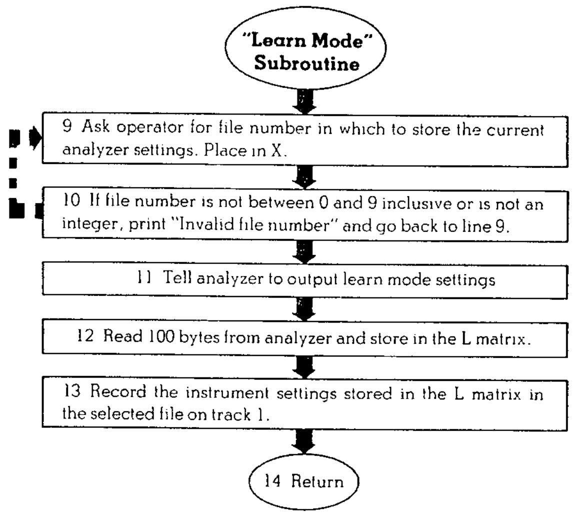
Both subroutines are well explained by their flow charts. The next section provides two subroutines to read back into the analyzer the information stored by these subroutines. The first of both subroutines in each section is slower, but probably easier to understand. The "Learn Mode" subroutine at the top of the facing page simply asks for a learn mode output and reads 100 bytes into an array. This array is then recorded on track 1 of the tape cartridge.

The second subroutine reads the data from the analyzer six times faster. It uses the buffered I/O capability of the Extended I/O ROM to speed the data transfer. The buffer statement in line 3 sets up the B$ string variable as buffer area in the calculator memory. The transfer statement in line 13 then transfers the bytes to this buffer area without translating them to binary or formatting them to the calculator's storage format. The string buffer is then stored on the tape. Because we do not wish to view the data in the calculator but merely want to transfer the data from the analyzer to the tape, this is a viable, fast subroutine.

Note that because both subroutines have the same name, only one can be used in a program. The second one is recommended. Also note that files must be marked on track 1 of the tape for the instrument settings.

Program

```
0: "3585 System's Application Note Software":
1:
2: dim L[100]
3: dev "3585",711
4: lcl 7
5: gto "Start"
6:
7:
8: "Learn Mode":
9: ent "File # to store instr. settings?",X
10: if X<0 or X>9 or X#int(X);prt "Invalid File #";spc 2;gto -1
11: wrt "3585","LO"
12: for I=1 to 100;rdb("3585")→L[I];next I
13: trk 1;rcf X,L[*]
14: ret
15:
16:
17: "Start":
18: dsp "Set up test. Then CONT.";stp
19: rem 7
20: cll 'Learn Mode'
21: lcl 7
*308
```



Variables Used

I	Byte counter
X	File number
L(100)	Learn mode storage array

Faster Program

```
0: "3585 System's Application Note Software":
1:
2: dim B$[116]
3: buf "B",B$,3
4: dev "3585",711
5: lcl 7
6: gto "Start"
7:
8:
9: "Learn Mode":
10: ent "File # to store instr. settings?",X
11: if X<0 or X>9 or X#int(X);prt "Invalid File #";spc 2;gto -1
12: wrt "3585","LO"
13: tfr "3585","B",100,10
14: if rds("B")=-1;jmp 0
15: trk 1;rcf X,B$
16: ret
17:
18:
19: "Start":
20: dsp "Set up test. Then CONT.";stp
21: rem 7
22: cll 'Learn Mode'
23: lcl 7
*26686
```

Variables Used

X	File number
B$	String variable used as buffer for learn mode information

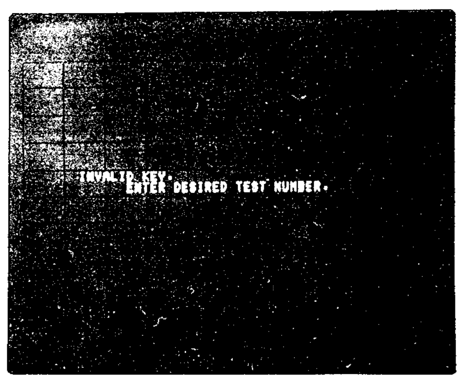
Test Selection from Analyzer

cll 'Test Number?'

These subroutines use the keyboard of the analyzer to load the analyzer with the instrument settings stored by the "Learn Mode" subroutines. The controller places the message "ENTER DESIRED TEST NUMBER" (see lower left photo) on the analyzer CRT and the operator presses the desired key (0-9) on the analyzer keyboard. This outputs the ASCII code for the key pushed to the calculator (decimals 48 to 57 for 0 to 9 respectively). If some other key on the analyzer is pushed, the analyzer will beep and the calculator will wait for a valid entry. Upon a valid entry, the calculator will load the instrument settings from the correct tape file and output these to the analyzer.

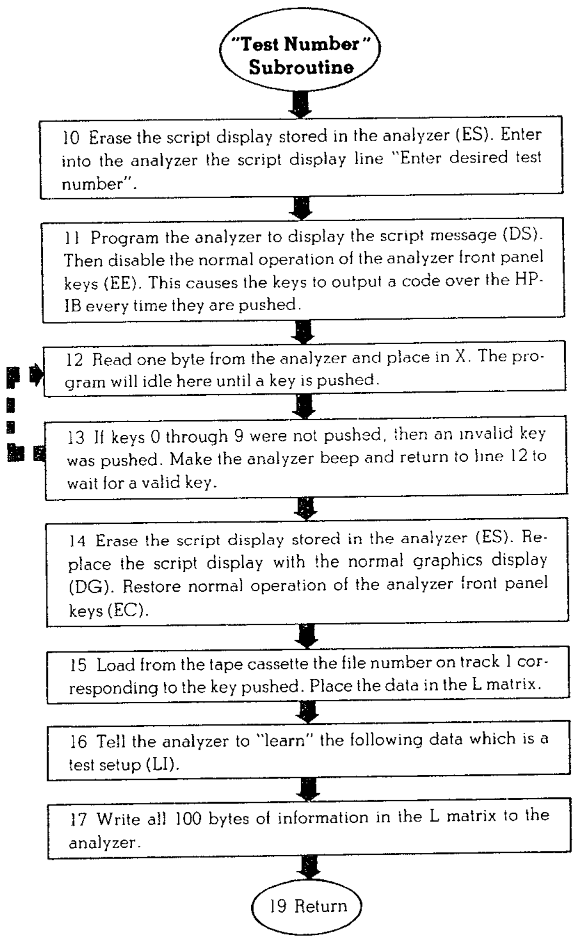
As described in the "Learn Mode" subroutine, these subroutines must be run with their respective "Learn Mode" subroutines because of the form their data is stored in. Once again the second program is faster, this time by about 1.2 seconds. The second program also uses lines 15 and 16 to check and see if the tape file actually has instrument settings recorded in it by looking at the length of the file. If the file doesn't have 124 bytes of data in it, the "EMPTY FILE" error message is displayed and the analyzer asks for a new file number. Also, as shown in the lower right photo, it puts up an error message when an invalid key is pushed.

These displayed error messages could be incorporated into the first subroutine, but the second "Learn Mode" and "Test Number?" subroutines using the buffered I/O are recommended, primarily because of their speed.



Program

```
0: "3585 System's Application Note Software":
1:
2: dim L[100]
3: dev "3585",711
4: rem 7
5: clr "3585"
6: gto "Start"
7:
8:
9: "Test Number?":
10: wrt "3585","ESL3            ENTER DESIRED TEST NUMBER."
11: wrt "3585","DSEE"
12: rdb("3585")→X
13: if X<48 or X>57;wrt "3585","RC5";gto -1
14: wrt "3585","ESDGEC"
15: trk 1;ldf X-48,L[*]
16: wtb "3585","LI"
17: for I=1 to 100;wtb "3585",L[I];next I
18: wtb "3585",13,10
19: ret
20:
21: "Start":
22: cll 'Test Number?'
*30448
```



Variables Used	
I	Byte counter
X	File number (test number)
L(100)	Learn mode storage array

Faster Program

```
0: "3585 System's Application Note Software":
1:
2: dim B$[116]
3: buf "B",B$,3
4: dev "3585",711
5: rem 7
6: clr "3585"
7: gto "Start"
8:
9:
10: "Test Number?":
11: wrt "3585","ESL3            ENTER DESIRED TEST NUMBER."
12: wrt "3585","DSEE"
13: rdb("3585")→X
14: if X<48 or X>57;wrt '3585","RC5,L2        INVALID KEY.";gto -1
15: trk 1;fdf X-48;idf U,Y,Z
16: if Z#124;wrt "3585",'RC5,L2        EMPTY FILE.";gto -3
17: wrt "3585","DGEC"
18: ldf X-48,B$
19: wtb 711,"LI"
20: tfr "B",711,100,10
21: if rds("B")=-1;jmp 0
22: ret
23:
24: "Start":
25: cll 'Test Number?'
*6404
```

Variables Used	
U	Working register
X	File (Test) number
Y	Working register
Z	File length register
B$	String variable used as buffer for learn mode information

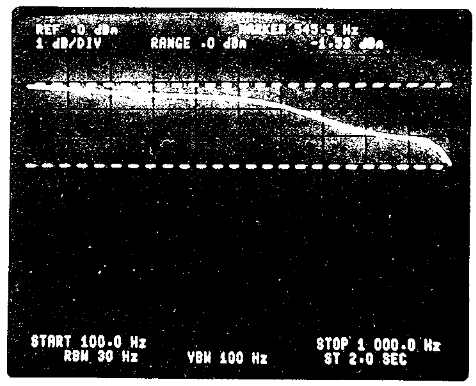
Additional Swept Displays

Limits on Analyzer CRT
cll 'Limits'

These subroutines are the first presented in this note which put graphic data back onto the analyzer CRT. The subroutines place two dotted straight lines in the B trace to indicate limits for the A trace. For example, in the photo below, we are using the tracking generator to sweep the passband of a low pass filter. The filter response falls within the +0, -3 dB limits drawn on the screen by the controller.

The vertical position of each of the 1000 horizontal points is written to the analyzer as a two byte word. The vertical position is scaled from 0 at the bottom of the screen to 1000 at the top. The most significant bits of this position code are sent in the first byte along with a code which indicates blanking or unblanking of the trace. The second byte contains the least significant bits of the position code.

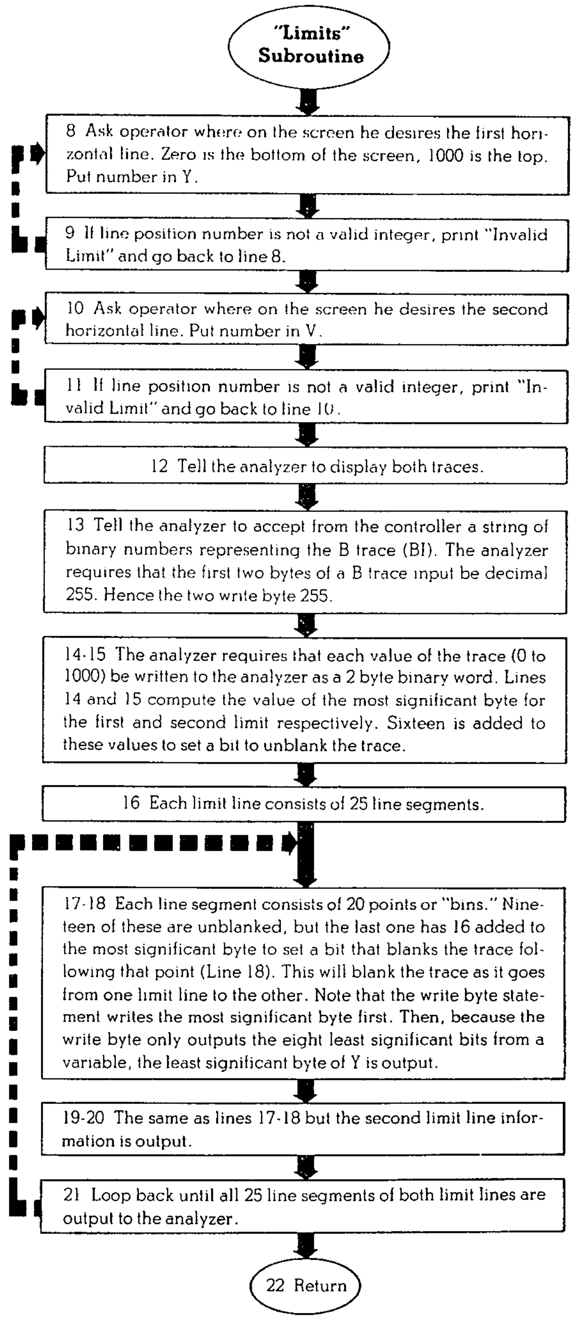
Because we are writing 2002 bytes to the analyzer, even a fraction of a millisecond difference in output time of the controller can make a large difference in the time it takes to write a trace on the analyzer. The second subroutine replaces the for-next loop with a long write byte statement and this reduces the time to write a trace on the analyzer from 2.2 seconds to 1.2 seconds.



Program

```
0: "3585 System's Application Note Software":
1:
2: dev "3585",711
3: rem 7
4: gto "Start"
5:
6:
7: "Limits":
8: ent "1st limit? (0 to 1000)",Y
9: if Y<0 or Y>1000 or Y#int(Y);prt "Invalid Limit";spc 2;gto -1
10: ent "2nd limit? (0 to 1000)",V
11: if V<0 or V>1000 or V#int(V);prt "Invalid Limit";spc 2;gto -1
12: wrt "3585","TA1TB1"
13: wtb "3585","BI",255,255
14: int(Y/256)+16→X
15: int(V/256)+16→U
16: for I=1 to 25
17: for J=1 to 19;wtb 731,X,Y;next J
18: wtb 731,X+16,Y
19: for J=1 to 19;wtb 731,U,V;next J
20: wtb 731,U+16,V
21: next I
22: wtb 731,13,10
23: ret
24:
25:
26: "Start":
27: cll 'Limits'
*25807
```

Variables Used	
I	Line segment counter
J	Point counter within a line segment
U	Most significant byte of 2nd limit
V	Second limit
X	Most significant byte of 1st limit
Y	First limit

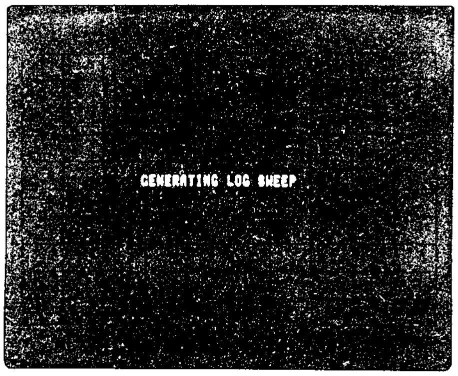


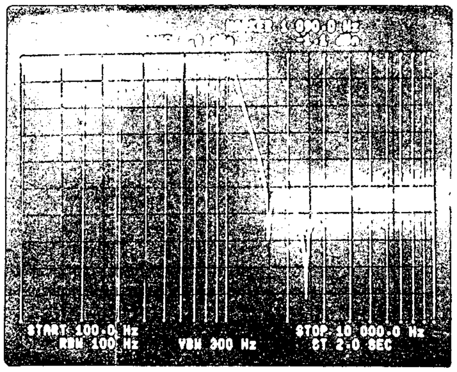
Program

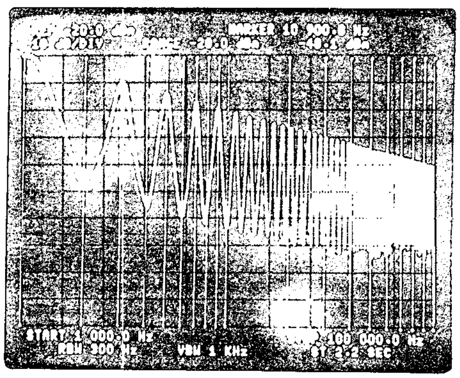
```
0: "3585 System's Application Note Software":
1:
2: dev "3585",711
3: rem 7
4: gto "Start"
5:
6:
7: "Limits":
8: ent "1st limit? (0 to 1000)",Y
9: if Y<0 or Y>1000 or Y#int(Y);prt "Invalid Limit";spc 2;gto -1
10: ent "2nd limit? (0 to 1000)",V
11: if V<0 or V>1000 or V#int(V);prt "Invalid Limit";spc 2;gto -1
12: wrt "3585","TA1TB1"
13: wtb "3585","BI",255,255
14: int(Y/256)+16→X
15: int(V/256)+16→U
16: for I=1 to 25
17: wtb 731,X,Y,X,Y,X,Y,X,Y,X,Y,X,Y,X,Y,X,Y,X,Y,X,Y,X,Y,X,Y,X,Y,
    X,Y,X,Y,X,Y
18: wtb 731,X,Y,X,Y,X,Y,X+16,Y
19: wtb 731,U,V,U,V,U,V,U,V,U,V,U,V,U,V,U,V,U,V,U,V,U,V,U,V,U,V,
    U,V,U,V,U,V
20: wtb 731,U,V,U,V,U,V,U+16,V
21: next I
22: wtb 731,13,10
23: ret
24:
25:
26: "Start":
27: cll 'Limits'
*8881
```

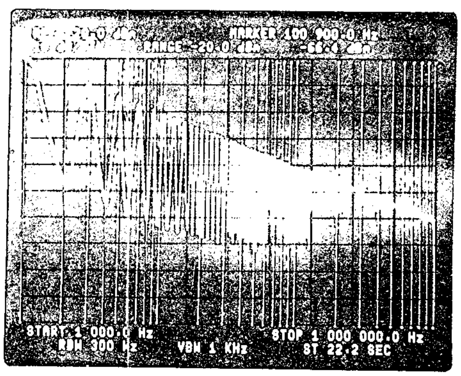
Logarithmic Sweep

cll 'Log' (start freq, stop freq, # of linear segments)









This is the longest subroutine in this note and builds on much of what was presented before. It uses an extended I/O buffer to assemble the complete log trace from linear trace segments and to assemble the electronic log graticule. It puts both the A and B traces back into the analyzer and uses the script display message "GENERATING LOG SWEEP" to mask from the operator the partial traces used in generating the log sweep.

The program operation is detailed in the flow chart, but the basic philosophy will be described here. The number of points desired in each linear segment is

$$N = 1000/p3,$$

where p3 is the number of linear segments. This insures that when all the linear segments are put back into the analyzer, we will get a 1000 point sweep.

To get the desired N points on each linear segment, we place the marker at the Nth point of the sweep where we will stop the sweep. We make the start frequency of the trace equal to the start frequency of the segment. The stop frequency of the trace is greater than the stop frequency of the segment because the segment stop frequency is at the Nth point of the sweep where the marker is. Because of this, the subroutine can try to program the analyzer to above 40 MHz. If this happens, the analyzer will beep and the sweep display will be invalid.

There is one more point of which the user should be careful. It is that all the measurements are made with the calibration disabled. This may mean that the instrument specifications are no longer valid. However, it does mean that the display can be generated faster. This lack of calibration should not be a serious problem since the display can be read only to the CRT resolution and accuracy is reduced anyway. In other words, the logarithmic sweep is a useful display mode, but does not have the high accuracy potential of the linear sweeps of the 3585.

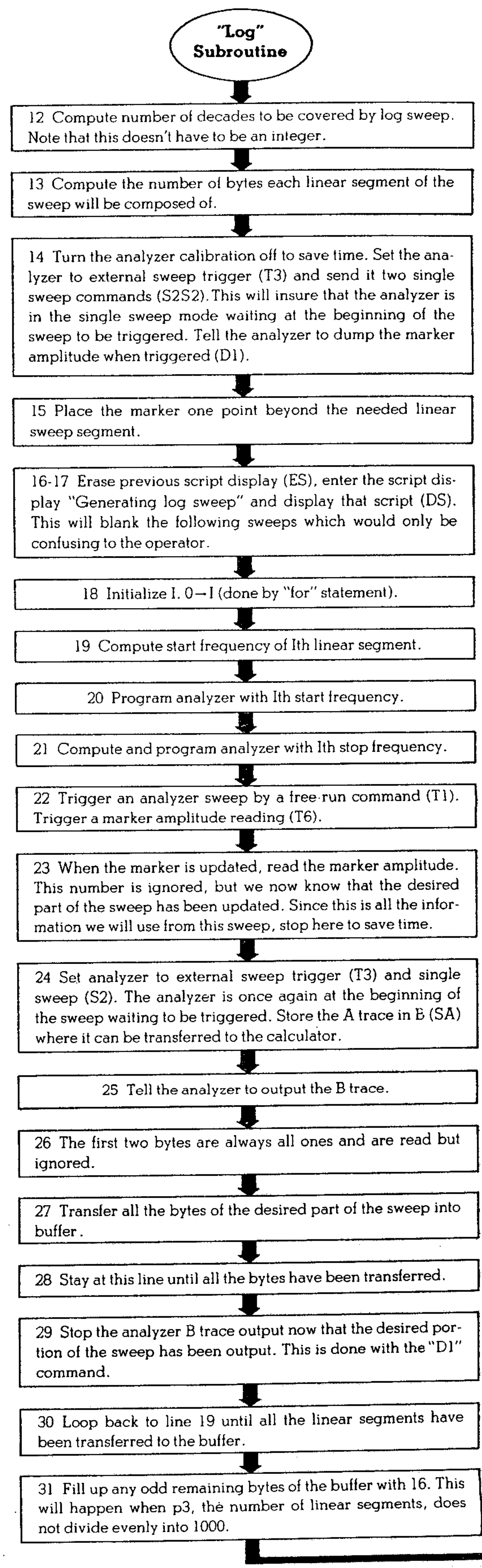
The operator must decide whether the bandwidth hold should be on or off. This decision is based on the kind of signal being analyzed. If a network response is being analyzed, it is better for the bandwidth hold to be off because the bandwidth will widen at higher frequencies and therefore the sweep will be faster. However, if the harmonics of a signal are being viewed, the bandwidth hold must be on. This is because as the bandwidth widens, more than one harmonic will be measured at a given frequency, and the result will be too large.

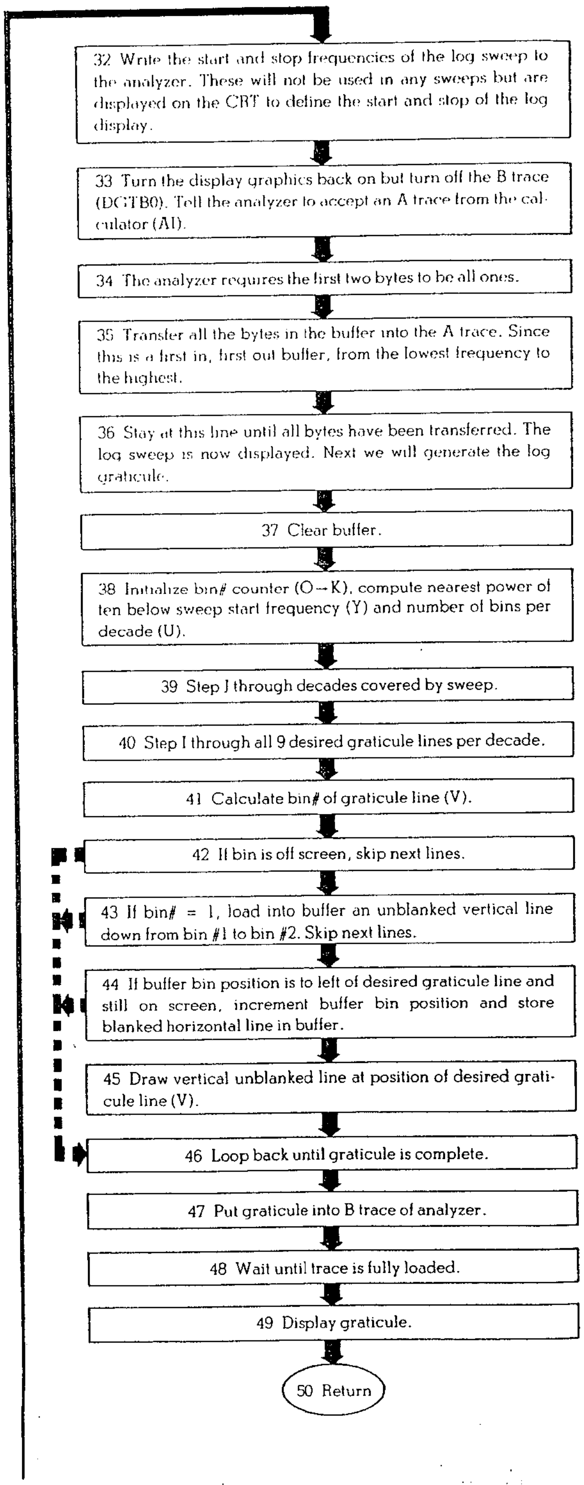
Program

```
0: "3585 System's Application Note Software":
1:
2: rem 7;llo 7
3: dim Z[1002]
4: buf "A",2002,3
5: fmt 4,"FA",f.1,"HZ"
6: fmt 5,"FB",f.1,"HZ"
7: fmt 6,"MK",f.0
8: gto "Start"
9:
10:
11: "Log":
12: log(p2/p1)→Z
13: 2int(1000/p3)→X
14: wrt 711,"SV4,T3,S2,S2,D1"
15: wrt 711.6,X/2+1
16: wrt 711,"ESL3                          GENERATING LOG SWEEP"
17: wrt 711,"DS"
18: for I=0 to p3-1
19: p1*10^(ZI/p3)→W
20: wrt 711.4,W
21: wrt 731.5,p3p1*10^(Z(I+1)/p3)+(1-p3)W
22: wrt 731,"T1T6"
23: red 711,Y
24: wrt 711,"T3S2SA"
25: wrt 731,"BO"
26: rdb(711)→Y;rdb(711)→Y
27: tfr 711,"A",X
28: if rds("A")=-1;jmp 0
29: wrt 711,"D1"
30: next I
31: for I=p3X+1 to 2002;wtb "A",16;next I
32: wrt 711.4,p1;wrt 711.5,p2
33: wrt 711,"DG,TB0,AI"
34: wtb 731,255,255
35: tfr "A",731,2002
36: if rds("A")=-1;jmp 0
37: "Generate Gradicule":buf "A"
38: 0→K;int(log(p1))→Y;1000/Z→U
39: for J=Y to Y+Z+1
40: for I=1 to 9
41: int(U(log(I/p1)+J)+1)→V
42: if V<1;gto +4
43: if V=1;wtb "A",19,232,16,0;K+2→K;gto +3
44: if K<V-4 and K<1001;K+1→K;wtb "A",32,0;jmp 0
45: if K<998;K+4→K;wtb "A",16,0,16,0,19,232,35,232
46: next I;next J
47: wtb 711,"BI",255,255:tfr "A",711
48: if rds("A")=-1;jmp 0
49: wrt 731,"TB1"
50: ret
51:
52:
53: "Start":
54: cll 'Log'(1e2,1e4,10
*19858
```

Variables Used

I	General index
J	Index used in generating graticule
K	Bin number used in generating graticule
V	Position of graticule line
W	Start frequency of linear sweep segments
X	Twice the number of points in each linear sweep segment
Y	Working register
Z	Number of decades covered in log sweep




32 Write the start and stop frequencies of the log sweep to the analyzer. These will not be used in any sweeps but are displayed on the CRT to define the start and stop of the log display.
33 Turn the display graphics back on but turn off the B trace (DGTB0). Tell the analyzer to accept an A trace from the calculator (AI).
34 The analyzer requires the first two bytes to be all ones.
35 Transfer all the bytes in the buffer into the A trace. Since this is a first in, first out buffer, from the lowest frequency to the highest.
36 Stay at this line until all bytes have been transferred. The log sweep is now displayed. Next we will generate the log graticule.
37 Clear buffer.
38 Initialize bin# counter (O→K), compute nearest power of ten below sweep start frequency (Y) and number of bins per decade (U).
39 Step J through decades covered by sweep.
40 Step I through all 9 desired graticule lines per decade.
41 Calculate bin# of graticule line (V).
42 If bin is off screen, skip next lines.
43 If bin# = 1, load into buffer an unblanked vertical line down from bin #1 to bin #2. Skip next lines.
44 If buffer bin position is to left of desired graticule line and still on screen, increment buffer bin position and store blanked horizontal line in buffer.
45 Draw vertical unblanked line at position of desired graticule line (V).
46 Loop back until graticule is complete.
47 Put graticule into B trace of analyzer.
48 Wait until trace is fully loaded.
49 Display graticule.
50 Return

HP 3585A HP-IB Programming Codes

Programming Front Panel	
Center Freq.	— CF
Freq. Span	— FS
Start Freq.	— FA
Stop Freq.	— FB
Cen Freq. Step Size	— CS
Ref Level Log	— RL
dB/DIV	— DD
Ref Level Volt	— RV
Up	— UP
Down	— DN
V/MHz/dBm	— VL,MZ,DM
mV/kHz/dBV	— MV,KZ,DV
μV/Hz/dB	— UV,HZ,DB
sec	— SC
Full Sweep	— FL
Save Register	— SV
Recall Register	— RC
Sweep Mode	
Auto	— S1
Single	— S2
Manual	— S3
Sweep Trigger	
Free Run	— T1
Line	— T2
Ext	— T3
BW/VBW/ST	
BW Coupled	— CP0,CP1
Coupling Preset	— PR
BW Hold	— BH0,BH1
RBW	— RB
Video	— VB
Sweep Time	— ST
Input	
Impedance	
1MΩ	— I1
50Ω	— I2
75Ω	— I3
Range	
Autorange	— AR0,AR1
Range	— RA and R01 through R12
Auto Level	— AL0,AL1
Display	
Clear A	— CA
Store A→B	— SA
View Trace A	— TA0,TA1
View Trace B	— TB0,TB1
Max Hold	— MH0,MH1
A-B	— AB0,AB1
Continuous	
Marker	— C1
Manual	— C2
Ref Level	— C3
Center Freq.	— C4
Level Line	— C5
Off	— C6
Clear Line	— CL
Noise Level	— NL0,NL1
Counter	— CN0,CN1
Offset→Span	— OS
MRK→Center	— MC
MRK→Ref Level	— MR
MRK→Offset	— MO
MRK→Step	— MS
Knob	— IL,IR
Offset	— OF0,OF1

Bus Operation Only

Marker	MK1 to MK1001
Immediate Trigger	T4
Delayed Trigger	T5
Dump Marker Amp only	D1 (default)
Dump Marker Amp & Freq.	D2
Dump A Trace	D3
Dump B Trace	D4
Dump Cal Reg.	D5
Dump Status	D6
Dump Alphas	D7
Dump D1 Continuous	D8
Dump D2 Continuous	D9
Display Graphics	DG
Display Annotation	DA
Display Script	DS
Erase Script	ES
Enter Annotation	LA
Enter Script Line 1	L1
Enter Script Line 2	L2
Enter Script Line 3	L3
Enter Script Line 4	L4
Enter Script Line 5	L5
Enter Script Line 6	L6
Learn mode in from HP-IB	LI
Learn mode out to HP-IB	LO
B trace in (binary) from HP-IB	BI
B trace out (binary) to HP-IB	BO
Memory out to HP-IB	MD
Memory in from HP-IB	ML
Memory Run	JM
Delay trigger without SRQ	T6
A trace in (binary) from HP-IB	AI
Enable key entry	EE
Enable key entry with SRQ	EQ
Clear enable entry	EC

HEWLETT PACKARD

Application Note 246

Printed in U.S.A.

April 1978

5952-8765

For more information, call your local HP Sales Office or East (301) 948-6370 • Midwest (312) 255-9800 • South (404) 955-1500 • West (213) 970-7710. Or write: Hewlett-Packard, 1501 Page Mill Road, Palo Alto, California 94304. In Europe: P.O. Box 85, CH-1217 Meyrin 2, Geneva, Switzerland. In Japan: YHP, 1-59-1, Yoyogi, Shibuya-Ku, Tokyo, 151.