THIRD OCTAVE ANALYSIS WITH THE HP 3582A SPECTRUM ANALYZER

HEWLETT **hp** PACKARD
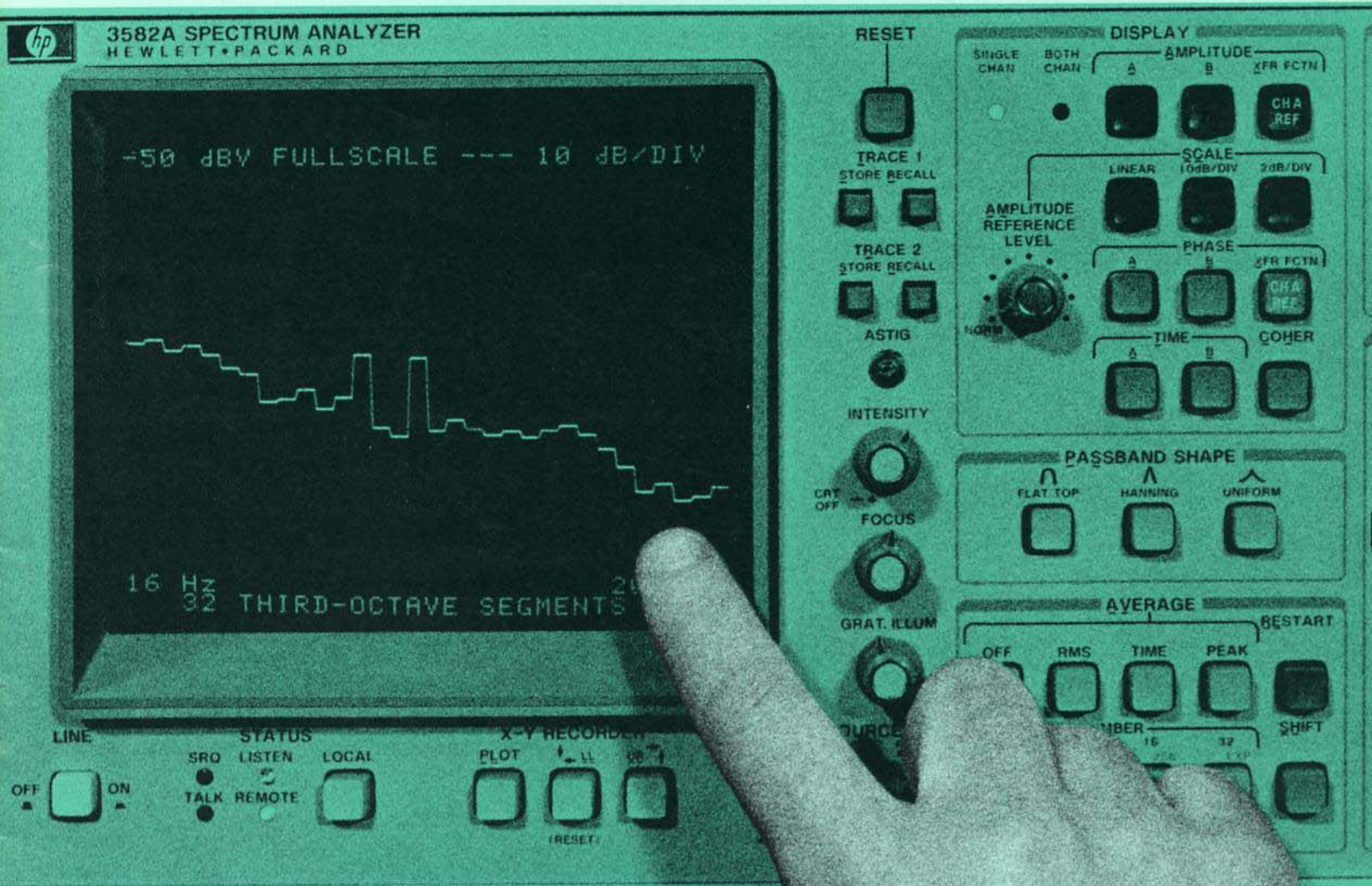
FOREWORD

Low frequency analyzers based on digital signal processing — especially the Fast Fourier Transform algorithm — are rapidly replacing older analog spectrum analyzers for a variety of measurement tasks. However, even the most enthusiastic FFT analyzer users recognize that there are some measurements for which they are not particularly suited. Log frequency sweep and 1/3 octave analysis are examples.

Nevertheless, the combination of an FFT analyzer and a "friendly" (i.e., easily programmed) small computer can perform a greater variety of measurements than the analyzer itself can do. For this to happen, it is essential to have fast, efficient communication between the two. The Hewlett-Packard Interface Bus (HP-IB*) serves this need well.

1/3 octave analysis is the measurement of a frequency spectrum by the use of constant percentage bandwidth filters 1/3 octave wide and spaced 1/3 octave. It has long been popular for audio and acoustic applications, largely because of the relationship between this filtering technique and certain psychoacoustic properties of human hearing.

This application note offers a means for making 1/3 octave measurements with a 3582A Spectrum Analyzer controlled by a 9835A Desktop Computer. Enough information is included (program listing, flowcharts, and description) to enable the reader to use it directly or to modify it as he requires.

*HP-IB is Hewlett-Packard's implementation of IEEE Standard 488 and identical ANSI Standard MC1.1 "Digital interface for programmable instrumentation."

THE HEWLETT-PACKARD MODEL 3582A SPECTRUM ANALYZER
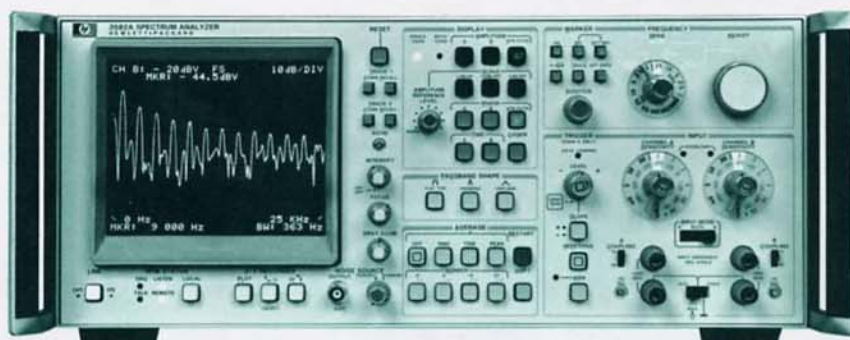
The HP 3582A is a spectrum analyzer covering the frequency range of DC to 25 kHz. Although it is a FFT-based, digital instrument, a special design effort has made it as straightforward to use as a conventional swept analyzer. With dual measurement channels it is possible to measure transfer function gain and phase, as well as the coherence function. A built-in random or pseudo-random noise source, whose spectrum tracks the analysis range, is a useful measurement stimulus. Band Selectable Analysis enables narrowband, high resolution analysis to be applied to any portion of the frequency range. The instrument comes equipped with a flexible HP-IB interface for control and two-way data transfers.

TABLE OF CONTENTS

Section 1:
Third octave analysis

In concept, "third octave" analysis is straightforward: imagine a set of parallel-connected filters being used to examine an audio signal. The center frequencies of the filters are scaled by a factor of 1/3 octave; that is, each filter is located at a frequency $2^{1/3}$ times its lower neighbor. In addition the nominal bandwidth of each filter is $2^{1/3}$ -1 times its center frequency. To cover the audio range of, say, 20 Hz to 20 kHz (9.97 octaves), with this technique requires 30 filters. Because of the multiplicative frequency spacing and bandwidth of the filters, it is convenient to display their characteristics on a log frequency plot, such as illustrated by Figure 1.

Historically, this kind of analyzer has been implemented with an actual parallel bank of filters. The instantaneous signal amplitude in each filter is detected and converted to dB. The data is displayed on an oscilloscope in the form of a bar graph: log amplitude (vertical) versus log frequency (horizontal).

Although parallel analog filters are still being used in some current instruments, there are at least two newer alternative techniques: digital filtering and FFT synthesis. Of these, the digital filter approach is preferable from the point of view of performance. This is primarily because the hardware can be optimized for the 1/3 octave task

and the display is "real time." However, for the many cases for which "real time" operation is not necessary — meaning that the signal to be analyzed has a stationary spectrum — the FFT technique is attractive, especially if you already have an FFT analyzer! Advantages of the FFT synthesis technique include easy modification of the frequency range and the use of frequency weighting functions if desired.

Figure 1.

Representation of a 1/3 octave analyzer composite filter characteristic
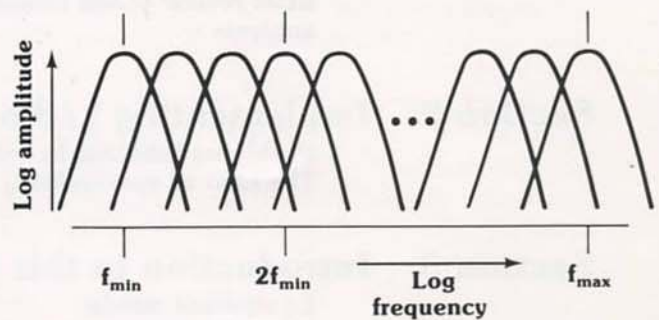
Section 2:
Implementing 1/3 octave analysis with an FFT analyzer

Why can't an FFT analyzer, such as the 3582A, be modified so that it produces a 1/3 octave analysis directly? Primarily because the FFT algorithm generates data on a set of linearly spaced sample points in the frequency domain. Its display has a linear frequency axis, not logarithmic as required by the 1/3 octave data. Also, the individual FFT filters, or "bins," have all the same bandwidth rather than bandwidths proportional to their center frequencies.

The approach used in this application note is to synthesize the frequency characteristics of 1/3 octave filters by combining the signals from several FFT bins. This requires weighting the contribution from each bin so that the composite "filter" is a good approximation to the specified shape of the 1/3 octave filter in question. Figure 2 indicates how this is done.

At lower frequencies the approximation is not as good because only a few FFT bins can be used. In fact, in the 3582A the bin spacing is 100 Hz when using the 0-25 kHz span. If only this span were used, the lowest frequency third octave filter that could be synthesized would be about 500 Hz. This is certainly not satisfactory for audio analysis. Therefore, to adequately cover the audio range, three spans are used: 0-250 Hz, 0-5 kHz, and

0-25 kHz. The result is 32 third octave filters, with center frequencies ranging from 15.85 Hz to 19.95 KHz (Fig. 3).

Figure 2.

Synthesizing a composite 1/3 octave filter by combining the weighted responses of several FFT filters, or "bins"

1
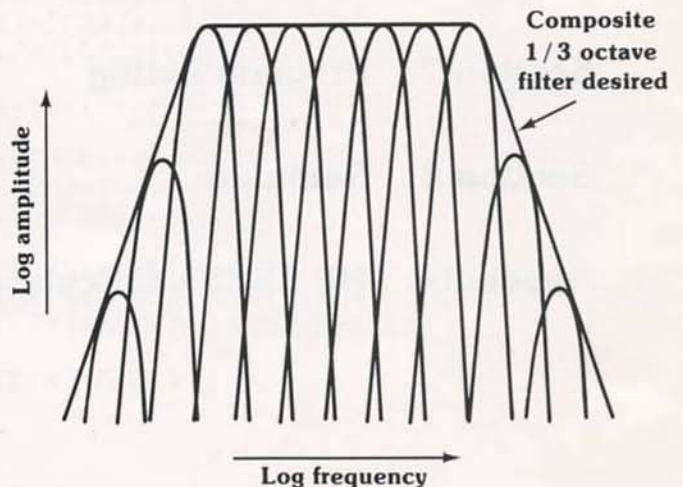
Section 3:
Introduction to this program

The equipment needed to use this program for 1/3 octave analysis is a 3582A Spectrum Analyzer (standard equipment includes HP-IB), a 9835A Desktop Computer with a 98332A I/O ROM installed, and a 98034A HP-IB Interface. The program is written in BASIC; it will run on another language-compatible calculator with the appropriate I/O, such as the 9845A. Memory requirements are approximately 15000 bytes for program and variable storage. Using the program listing and the flow diagrams, one can rewrite the program in another language. See the Appendix for the 9825A Calculator version.

Operating the program is simple. Pressing RUN causes the necessary initialization and then the user is asked, "Do you want RMS averaging?" The reason for this is that many spectra are random in nature, and a better estimate of the spectrum — and thus a better 1/3 octave analysis — is obtained when the 3582A is allowed to average the data. (Application Note 245-1, "Signal Averaging with the 3582A Spectrum Analyzer," deals with averaging in detail.) After the user answers the question, the program proceeds to:

a) set the 3582A to each frequency range in turn
b) bring the amplitude data for each range from the 3582A display into the controller for processing
c) convert the data, apply weights, and combine to form 32 synthesized results as if from 1/3 octave filters
d) format and output these results in the form of a bar-graph display on the 3582A
e) return to the beginning for another analysis, if desired

Step (c) requires some explanation. How exactly should the data from several FFT bins be combined to approximate the result expected from a 1/3 octave filter? It should be done on the basis of power rather than linear addition. This is because the signals in adjacent FFT bins are uncorrelated when the input is a random time signal. And when a coherent signal is analyzed, such as a sinusoid, the sum of signal power remains constant as the signal frequency varies. This means there is no ripple in the synthesized passband.

In the program, line 3090 converts the FFT bin signals to power (that is, volts squared) from dBV, and line 3180 converts the sum of weighted powers back to dBV.

Figure 3.

Center frequencies of 1/3 octave filters synthesized by this program

| FILTER NUMBER	CENTER FREQ	
1	15.85	
2	19.95	
3	25.12	
4	31.62	
5	39.81	
6	50.12	Pass.1
7	63.1	0-250 Hz
8	79.43	
9	100	
10	125.9	
11	158.5	
12	199.5	
13	251.2	
14	316.2	
15	398.1	
16	501.2	
17	631	
18	794.3	
19	1000	Pass.2
20	1259	0-5 kHz
21	1585	
22	1995	
23	2512	
24	3162	
25	3981	
26	5012	
27	6310	
28	7943	
29	10000	Pass.3
30	12590	0-25 kHz
31	15850	
32	19950	

Section 4:
Comparison with ANSI class III 1/3 octave filters

The American National Standards Institute publishes a document* recognized as setting proper standards for these filters. The filters synthesized by this program conform closely to the specifications for "Third-Octave Band Filters – Class III," as defined in this paper. Here are the principal characteristics of the filters:

a) Center frequencies: Strictly speaking, the ANSI document defines these in "tenth decade" intervals, but the difference from third octave is negligible. The greatest deviation from the specified value occurs in filter #20, whose geometric mean frequency is 0.7% below the specified value of 1259 Hz. (±3% is allowable).

b) Transmission loss limits: All filters meet these criteria, although the rolloff rates differ due to the varying number of FFT bins used in the synthesis of individual filters. Filter #13, which uses only 4 bins, reaches -72 dB loss at 1/5 its center frequency, rather than the specified -75 dB. The other extreme is filter #12, which uses 49 bins. The attenuation characteristics of these two filters are shown in Figures 4 and 5, with the specification limits superimposed.

c) Effective bandwidth (noise bandwidth): This specification requires that the power output from a filter, when the input is white noise, be within 10% of the noise passed by an ideal rectangular 1/3 octave filter. Filter #10 has the greatest deviation, with a noise bandwidth 2.9% higher than standard.

d) Passband uniformity (passband ripple): The synthesized filters have no perceptible ripple within the defined band-edge frequencies, and so the ripple specification (0.5 dB) does not apply.

e) Variation of minimum loss among filters: Theoretically, all filters in this program have zero mid-band loss. Some variation will be encountered due to individual 3582A amplitude accuracy characteristics, specifically gain variations between different frequency spans. These will be well within the allowed ±1 dB.

f) Transient response: The program cannot meet this specification, since it is necessary that the signal being analyzed be statistically stationary during the acquisition of data.

Figure 4.
Filter #12
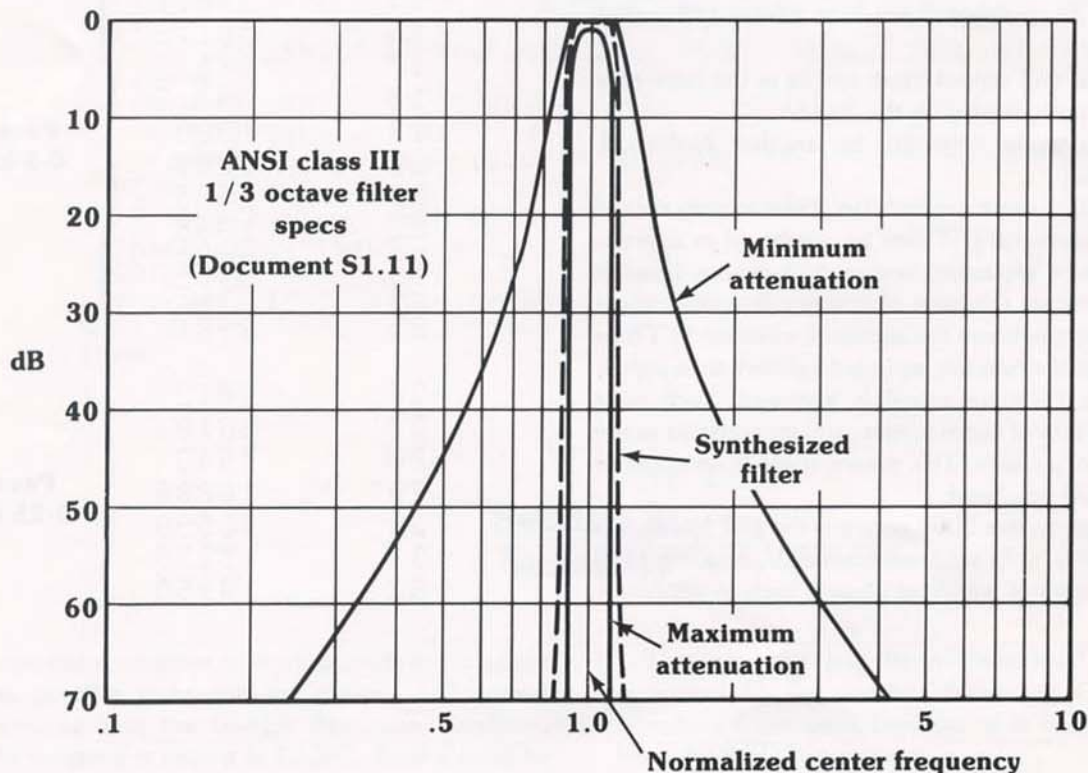
Section 5:
Suggested extensions to the program

Because of the modular structure of the program, it is simple to modify by adding or deleting sections.

"Modular" means that the action portions of the program are written as subroutines called up as needed by a control sequence located in lines 50 to 150.

Here are some possible modifications:

PRINTED OUTPUT. Some users may want a permanent record of the analysis results. The structure of this program makes this an easy job. Simply write an output routine to the desired printer (internal or HP-IB external), listing the contents of "Thirdmag" (dBV) together with the 1/3 octave filter number or center frequency. The center frequency is readily calculated as $10^{((filter\ number\ +11)/10)}$ Hz. Append this routine to the program and call it by a statement like

145   GOSUB Printer

This method also could be used to save the data on a mass storage device, like the tape cassette.

DISPLAY ANNOTATION. It may be desirable to identify individual filters more readily on the 3582A display. Since there are 32 third octave filters and also room for 32 characters on each of the four display lines, you can fill the lower two lines with digits so that they give the filter number when read vertically:

0000. . . .2333
1234. . . .9012

This change would be made in lines 2540 and 2550 of the program.

FREQUENCY WEIGHTING. Some measurements require the application of special shaping to measured spectra. An example is "A" weighting sometimes used in acoustics measurements. It is simple to do this with the present program. What is required is a table of dB loss values for the center frequency of each of the 1/3 octave filters in the program. Then a routine should be written to modify the dBV numbers in "Thirdmag" by these values. The routine should be called up before the display routine by adding, for example, the line

125   GOSUB Spectrum_weight

Figure 5.
Filter #13

4
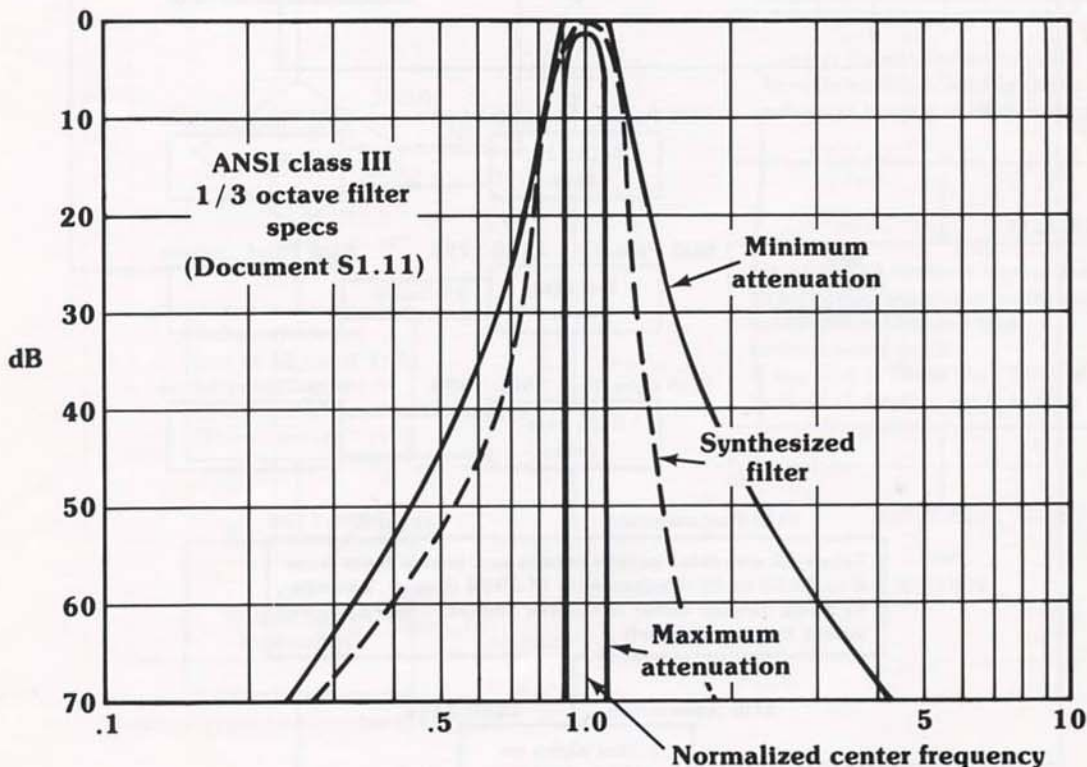
Main Flow Diagram – 1/3 octave program

```
                                        10-40
                                  ┌──────────────┐
                                  │  initialize  │
                                  │  variables   │
      Subroutine names ──→        └──────────────┘        ←── Program line numbers

      SUB Bin.groups                                500-590
      ┌─────────────────────────────────────┐
      │ defines first and last FFT bins      │
      │ to be used to compute each of the    │
      │ 32 one-third octave filter responses │
      └─────────────────────────────────────┘

      SUB Fft.weights                               1000-1230
      ┌─────────────────────────────────────┐
      │ establishes table of weighting factors│
      │ used in combining FFT responses       │
      └─────────────────────────────────────┘

                                         70
                                    (  Start  )

   SUB Set.ave 180-330                  80              SUB No.ave 160-170
   ┌──────────────────┐           ◇──────────◇          ┌──────────────┐
   │ Asks number of   │           │   RMS    │          │ Sends command│
   │ averages. Sends  │←── YES ───│ averaging│── NO ──→ │ to 3582 for  │
   │ commands to 3582 │           │ in 3582? │          │ no averaging │
   │ for #averages,   │           ◇──────────◇          └──────────────┘
   │ RMS averaging    │
   └──────────────────┘

              SUB Pass.1      1600-1690      SUB Third__octave
              ┌──────────┐                   ┌──────────────┐
              │ 0-250 Hz │ ─────────────────→│              │
              │  data    │ ←─────────────────│              │
              └──────────┘                   └──────────────┘

    see       SUB Pass.2      1700-1790      SUB Third__octave
  detailed    ┌──────────┐                   ┌──────────────┐
flow diagrams │ 0-5 kHz  │ ─────────────────→│              │
              │  data    │ ←─────────────────│              │
              └──────────┘                   └──────────────┘

              SUB Pass.3      1800-1890      SUB Third__octave
              ┌──────────┐                   ┌──────────────┐
              │ 0-25 kHz │ ─────────────────→│              │
              │  data    │ ←─────────────────│              │
              └──────────┘                   └──────────────┘

   SUB Plot.response                           2000-2100
   ┌────────────────────────────────────────────────────────┐
   │ Takes 32 one-third octave results and scales them from  │
   │ 0 to 1023 to fit requirements of 3582 display; outputs  │
   │ to 3582 display buffer in reverse sequence since display│
   │ writes from right to left                               │
   └────────────────────────────────────────────────────────┘

   SUB Annotation                              2500-2560
                    ┌──────────────┐
                    │ writes alpha on│
                    │ 3582 display  │
                    └──────────────┘
```
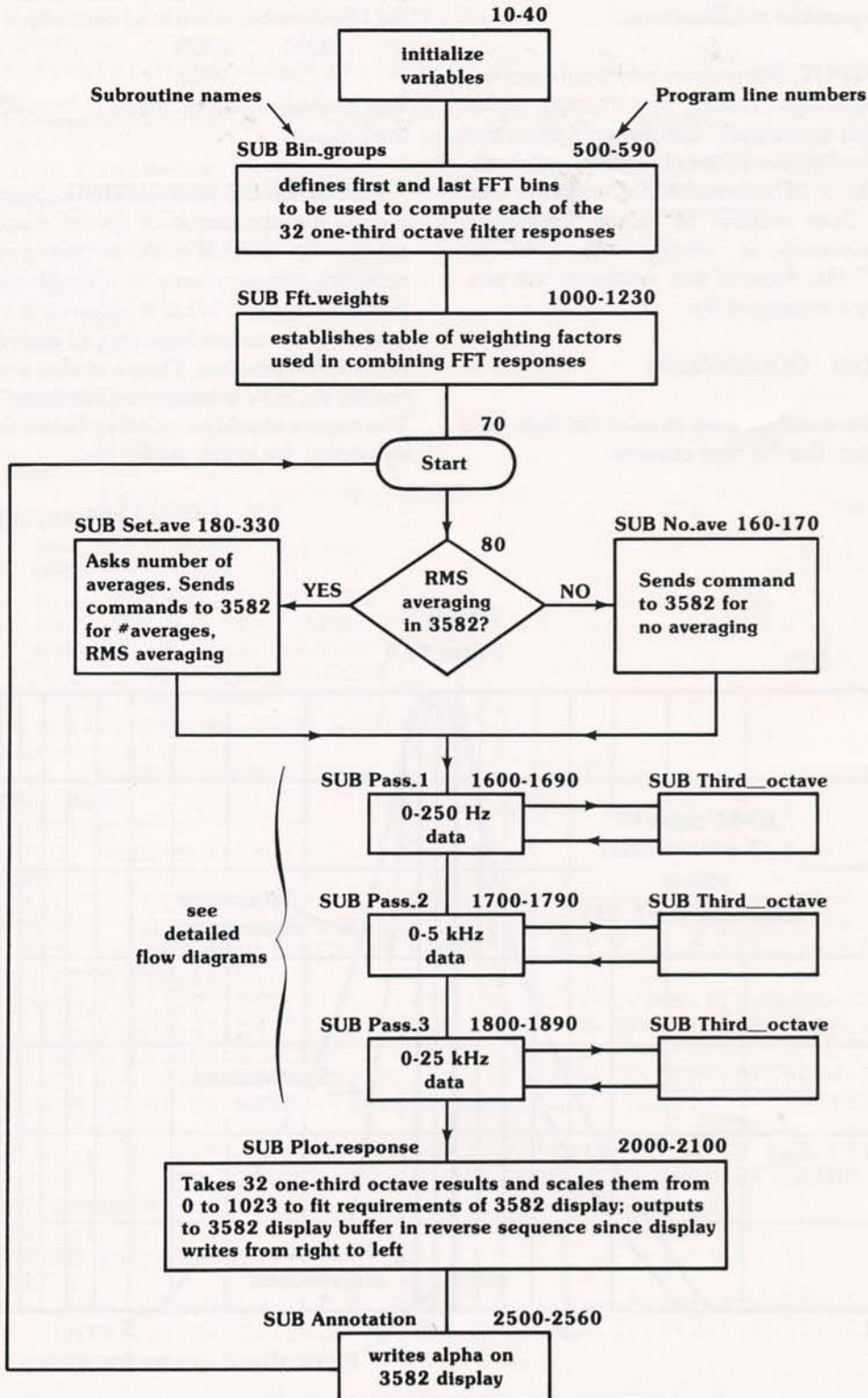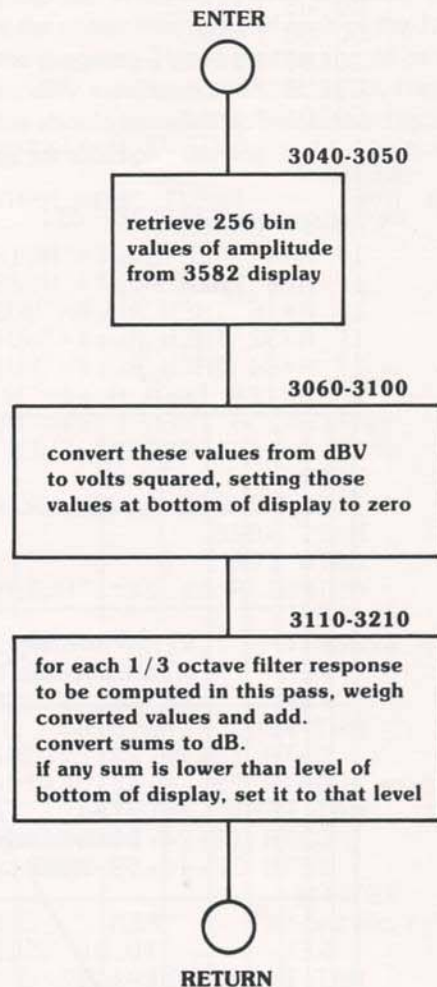
Detailed Flow Diagram
Pass subroutines – 1/3 octave program

Pass.1 begins at 1600

Pass.2 begins at 1700

Pass.3 begins at 1800

ENTER

Program line numbers;
XX are 16, 17, or 18

XX10-20
Set 3582 frequency span; start measurement

XX30
was RMS averaging asked for ?

NO

YES

XX40-50
interrogate 3582 status to determine end of averaging

XX50
averaging complete ?

NO

YES

XX60-70
define which set (out of 32 total) 1/3 octave filters are to be computed by "Third_octave"

XX80
call "Third_octave" subroutine

RETURN
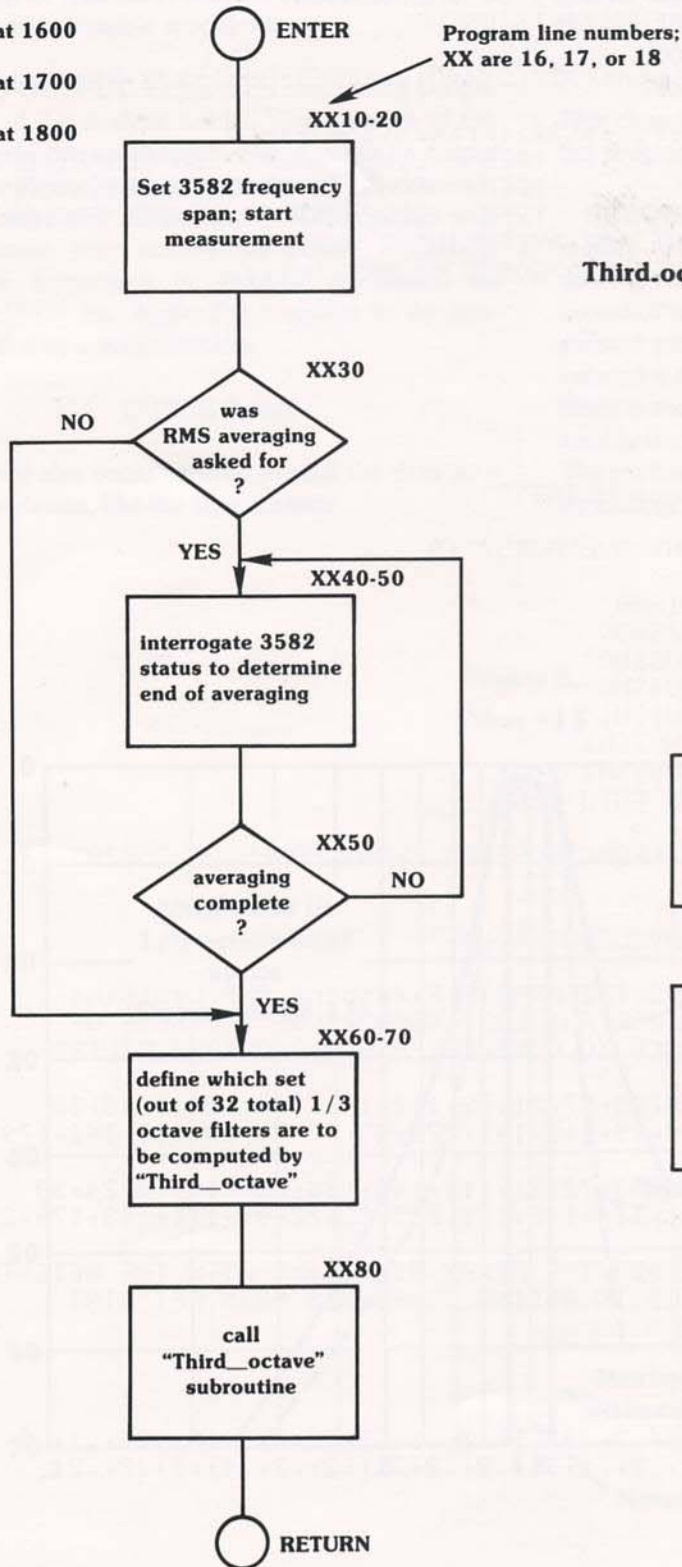
Detailed Flow Diagram
Third.octave subroutine – 1/3 octave program

ENTER

3040-3050
retrieve 256 bin values of amplitude from 3582 display

3060-3100
convert these values from dBV to volts squared, setting those values at bottom of display to zero

3110-3210
for each 1/3 octave filter response to be computed in this pass, weigh converted values and add.
convert sums to dB.
if any sum is lower than level of bottom of display, set it to that level

RETURN

```
1      REM    1/3 OCTAVE ANALYSIS PROGRAM FOR THE H-P 3582A SPECTRUM
2      REM    ANALYZER AND THE 9800 SERIES 35 CALCULATOR.
3      REM    H-P LOVELAND INSTRUMENT DIVISION.
10     OPTION BASE 1
20     INTEGER Firstbins(32),Lastbins(32),Graphics(256),Analyzer,First_third,Last_
third,A,I,J,K
30     SHORT Binweight(32,49),Thirdmag(32),Fftmag(256),Temp
40     Analyzer=711                          !REM    LINES 50 TO 150 COMPRISE THE MAIN
50         GOSUB Bin_groups                  !REM    CONTROL PROGRAM WHICH CALLS
60         GOSUB Fft_weights                 !REM    SUBROUTINES IN THE PROPER SEQUENCE
70 Start:      INPUT "DO YOU WANT RMS AVERAGING?",Averaging$
80             ON (Averaging$="YES")+1 GOSUB No_ave,Set_ave
90         GOSUB Setup
100        GOSUB Pass_1
110        GOSUB Pass_2
120        GOSUB Pass_3
130        GOSUB Plot_response
140        GOSUB Annotation
150        GOTO Start
160 No_ave:       OUTPUT Analyzer;"HLTAV1"
170       RETURN
180 Set_ave:      INPUT "HOW MANY AVERAGES?",A
190       Ave$="ERROR"
200       IF A=4 THEN Ave$="NU1SH0"
210       IF A=8 THEN Ave$="NU2SH0"
220       IF A=16 THEN Ave$="NU3SH0"
230       IF A=32 THEN Ave$="NU4SH0"
240       IF A=64 THEN Ave$="NU1SH1"
250       IF A=128 THEN Ave$="NU2SH1"
260       IF A=256 THEN Ave$="NU3SH1"
270       IF Ave$<>"ERROR" THEN GOTO 320
280       BEEP
290       DISP "CHOSE A NUMBER FROM THOSE ON FRONT PANEL OF 3582A"
300       WAIT 4000
310       GOTO 180
320       OUTPUT Analyzer;"HLTAV2LST0",Ave$
330      RETURN
500 Bin_groups:     REM    LOADS THE ARRAYS Firstbins AND Lastbins
510          REM    WITH THE FIRST AND LAST NUMBERS OF THE SETS OF
520          REM    FFT BINS USED TO COMPUTE EACH 1/3 OCTAVE FILTER
530        MAT READ Firstbins
540         DATA 15,18,23,29,36,45,57,71,89,112,142,178,12,15,18,23
550         DATA 29,36,45,56,71,89,112,141,178,45,57,71,89,112,142,179
560        MAT READ Lastbins
570         DATA 19,24,30,37,46,58,72,91,114,143,180,226,15,19,24,30
580         DATA 37,46,58,72,91,114,143,179,225,58,72,91,114,143,179,225
590      RETURN
1000 Fft_weights:    REM    LOADS THE ARRAY Binweight WITH THE WEIGHTS
1010         REM    TO BE USED IN ADDING RESPONSES FROM FFT BINS
1020        MAT Binweight=(.67)
1030        FOR I=1 TO 32
1040          READ Binweight(I,1)
1050        NEXT I
1060            DATA .25,.1,.3,.3,.3,.2,.2,.2,.1,.2,.3,.2,.15,.25,.1,.2
1070            DATA .3,.3,.2,.2,.2,.1,.1,.2,.2,.1,.2,.2,.1,.1,.3,.2
```

```
1080        FOR I=1 TO 32
1090          READ Binweight(I,2)
1100        NEXT I
1110            DATA .67,.4,.4,.5,.5,.4,.5,.4,.4,.4,.4,.4,.67,.67,.4,.4
1120            DATA .5,.5,.4,.4,.4,.4,.3,.5,.5,.4,.67,.4,.3,.4,.5,.5
1130        FOR I=1 TO 32
1140          READ Binweight(I,Lastbins(I)-Firstbins(I))
1150        NEXT I
1160            DATA .67,.4,.4,.5,.5,.4,.5,.4,.4,.4,.4,.4,.67,.67,.4,.4
1170            DATA .5,.5,.4,.4,.4,.4,.3,.5,.5,.4,.67,.4,.3,.4,.5,.5
1180        FOR I=1 TO 32
1190          READ Binweight(I,Lastbins(I)-Firstbins(I)+1)
1200        NEXT I
1210            DATA .25,.2,.25,.3,.3,.2,.2,.2,.1,.2,.3,.2,.5,.25,.2,.2
1220            DATA .3,.3,.2,.2,.2,.1,.1,.2,.2,.1,.2,.2,.1,.1,.3,.2
1230      RETURN
1500 Setup:       REM    INITIALIZES THE CONTROL SETTINGS OF THE 3582A.
1510          REM    CHAN "A" SENSITIVITY IS OBTAINED FOR PROPER DISPLAY SCALING.
1520          REM    THREE CONSTANTS ARE CALCULATED.
1530        OUTPUT Analyzer;"AA1AB0AX0SC2PA0PB0PX0CH0PS2FR1RP1MD2IM1TR0RR0"
1540        OUTPUT Analyzer;"LAS"
1550        ENTER Analyzer;Sensitivity
1560        Minimum=10↑((Sensitivity-80)/10)
1570        Display_scale=1023/80
1580        Input_scale=8/1023
1590      RETURN
1600 Pass_1:       REM    0 TO 250 Hz
1610        OUTPUT Analyzer;"SP8RUNRE"
1620        WAIT 2000
1630        IF Averaging$<>"YES" THEN GOTO 1660
1640        OUTPUT Analyzer;"LST0"
1650        IF BIT(READBIN(Analyzer),6)=0 THEN GOTO 1640
1660        First_third=1
1670        Last_third=12
1680        GOSUB Third_octave
1690      RETURN
1700 Pass_2:       REM    0 TO 5 KHz
1710        OUTPUT Analyzer;"SP12RUNRE"
1720        WAIT 1000
1730        IF Averaging$<>"YES" THEN GOTO 1760
1740        OUTPUT Analyzer;"LST0"
1750        IF BIT(READBIN(Analyzer),6)=0 THEN GOTO 1740
1760        First_third=13
1770        Last_third=25
1780        GOSUB Third_octave
1790      RETURN
1800 Pass_3:       REM    0 TO 25 KHz
1810        OUTPUT Analyzer;"SP14RUNRE"
1820        WAIT 1000
1830        IF Averaging$<>"YES" THEN GOTO 1860
1840        OUTPUT Analyzer;"LST0"
1850        IF BIT(READBIN(Analyzer),6)=0 THEN GOTO 1840
1860        First_third=26
1870        Last_third=32
1880        GOSUB Third_octave
1890      RETURN
```

```
2000 Plot_response:       REM     OUTPUTS GRAPHIC RESULTS TO 3582A DISPLAY
2010      FOR I=1 TO 32
2020        Temp=Display_scale*(Thirdmag(33-I)+80-Sensitivity)
2030        IF Temp>1023 THEN Temp=1023
2040        IF Temp<0 THEN Temp=0
2050          FOR J=1 TO 8
2060            Graphics(8*I+J-8)=Temp
2070          NEXT J
2080      NEXT I     :
2090      OUTPUT Analyzer USING "13A/256(Y)";"WTM,74400,256",Graphics(*)
2100    RETURN
2500 Annotation:       REM     OUTPUTS 3 LINES OF ALPHA TO 3582A DISPLAY,
2510                   REM     LEAVING THE 2ND LINE BLANK
2520      OUTPUT Analyzer USING "5A,DDD,32A";"WTA1,",Sensitivity," dBV FULLSCALE
 --- 10 dB/DIV "
2530      OUTPUT Analyzer;"WTA2,                                           "
2540      OUTPUT Analyzer;"WTA3,16 HZ                         20 KHZ"
2550      OUTPUT Analyzer;"WTA4,   32 THIRD-OCTAVE SEGMENTS          "
2560    RETURN
3000 Third_octave:      REM     TAKES THE DISPLAY DATA OF THE 3582 IN
3010          REM   BINARY FORM AND CONVERTS THE NUMBERS TO VOLTS SQUARED;
3020          REM   APPLIES THE WEIGHTING FACTORS AND COMBINES THE DATA TO FORM
3030          REM   RESULTS AS IF FROM ANSI CLASS III FILTERS.
3040      OUTPUT Analyzer;"HLTLFM,74400,256"
3050      ENTER Analyzer USING "#,Y";Fftmag(*)
3060      FOR I=1 TO 256               !REM   FORMAT BINARY DATA FROM DISPLAY
3070        Fftmag(I)=Input_scale*BINAND(Fftmag(I),1023)
3080        IF Fftmag(I)=0 THEN GOTO 3100
3090        Fftmag(I)=10↑(Fftmag(I)+Sensitivity/10-8)
3100      NEXT I
3110      FOR I=First_third TO Last_third    !REM   WEIGHT AND ADD FFT RESPONSES
3120        Temp=0
3130        FOR J=Firstbins(I) TO Lastbins(I)
3140          K=J-Firstbins(I)+1
3150          Temp=Temp+Binweight(I,K)*Fftmag(257-J)
3160        NEXT J
3170        IF Temp<Minimum THEN GOTO 3200
3180        Thirdmag(I)=10*LGT(Temp)
3190        GOTO 3210
3200        Thirdmag(I)=Sensitivity-80
3210      NEXT I
3220    RETURN
```

Section 8:
Summary

We hope this application note will provide some insight into one possible technique for making 1/3 octave measurements with the 3582A Spectrum Analyzer. While the program is written in BASIC, there should be enough flowcharts, program annotation, and comments to allow the interested reader to implement the measurement with a controller using another language.

The program has been developed, tested, and evaluated by the Product Marketing Group of HP's Loveland Instrument Division. It is based on a report from the R&D Department.

Appendix

HP 9825A Calculator Program

For the users of the popular 9825A Calculator, the 1/3 octave program is given here in HPL, the language of that machine.

The 9825A version is structurally the same as the 9835A program around which this application note is written. All subroutines have the same labels; this will help you use the flow diagrams of Section 6 to follow the 9825A version. However, most of the variables are different, because HPL allows only single characters for simple, array, and string variables. Listed below is a table showing corresponding variables in both programs.

EQUIVALENT VARIABLES

9835A Program	9825A Program	Comments
Firstbins(32)	F[32]	
Lastbins(32)	L[32]	
Graphics(256)	M[256]	shared by "Plot_response" and
Fftmag(256)	M[256]	"Third_octave" routines.
Thirdmag(32)	T[32]	also used as a temporary in "Fft_weights" routine.
Binweight(32,49)	B$[3136]	weight values times 100, stored in integer format.
Sensitivity	S	
Minimum	C	
Display_scale	D	
Input_scale	E	
First_third	M	
Last_third	L	
Temp	P	
I	I	
J	J	
K	K	
A	A	
Averaging$	A$[3]	
Ave$	D$[6]	
---	C$[2]	used as temporary in "Fft_weights" and "Third_octave".

```
0: "1/3 octave analysis program for the H-P 3582A spectrum":
1: "analyzer and the 9825A calculator":
2: dim F[32],L[32],B$[3136],T[32],M[256],A$[3],D$[6],C$[2]
3: dev "Analyzer",711
4: gsb "Bin_groups"
5: gsb "Fft_weights"
6: "Start":ent "do you want RMS averaging?",A$
7: if A$="yes";gsb "Set_ave"
8: if A$#"yes";gsb "No_ave"
9: gsb "Setup"
10: gsb "Pass_1"
11: gsb "Pass_2"
12: gsb "Pass_3"
13: gsb "Plot_response"
14: gsb "Annotation"
15: gto "Start"
16: "No_ave":wrt "Analyzer","HLTAV1";ret
17: "Set_ave":ent "how many averages?",A
18: "ERROR"→D$
19: if A=4;"NU1SH0"→D$
20: if A=8;"NU2SH0"→D$
21: if A=16;"NU3SH0"→D$
22: if A=32;"NU4SH0"→D$
23: if A=64;"NU1SH1"→D$
24: if A=128;"NU2SH1"→D$
25: if A=256;"NU3SH1"→D$
26: if D$#"ERROR";gto +3
27: beep;dsp "chose a number from front panel"
28: wait 3000;gto "Set_ave"
29: wrt "Analyzer","HLTAV2LST0",D$
30: ret
31: "Bin_groups":15→F[1];18→F[2];23→F[3];29→F[4];36→F[5];45→F[6];57→F[7]
32: 71→F[8];89→F[9];112→F[10];142→F[11];178→F[12];12→F[13];15→F[14];18→F[15]
33: 23→F[16];29→F[17];36→F[18];45→F[19];56→F[20];71→F[21];89→F[22];112→F[23]
34: 141→F[24];178→F[25];45→F[26];57→F[27];71→F[28];89→F[29];112→F[30]
35: 142→F[31];179→F[32];19→L[1];24→L[2];30→L[3];37→L[4];46→L[5];58→L[6]
36: 72→L[7];91→L[8];114→L[9];143→L[10];180→L[11];226→L[12];15→L[13];19→L[14]
37: 24→L[15];30→L[16];37→L[17];46→L[18];58→L[19];72→L[20];91→L[21];114→L[22]
38: 143→L[23];179→L[24];225→L[25];58→L[26];72→L[27];91→L[28];114→L[29]
39: 143→L[30];179→L[31];225→L[32]
40: ret
41: "Fft_weights":fti (67)→C$
42: for I=1 to 1568;C$→B$[2I-1,2I];next I
43: .25→T[1];.1→T[2];.3→T[3]→T[4]→T[5];.2→T[6]→T[7]→T[8]
44: .1→T[9];.2→T[10];.3→T[11];.2→T[12];.15→T[13];.25→T[14]
45: .1→T[15];.2→T[16];.3→T[17]→T[18];.2→T[19]→T[20]→T[21]
46: .1→T[22]→T[23];.2→T[24]→T[25];.1→T[26];.2→T[27]→T[28]
47: .1→T[29]→T[30];.3→T[31];.2→T[32]
48: for I=1 to 32;fti (100T[I])→B$[98(I-1)+1,98(I-1)+2];next I
49: .67→T[1];.4→T[2]→T[3];.5→T[4]
50: .5→T[5];.4→T[6];.5→T[7];.4→T[8]→T[9]→T[10]→T[11]→T[12]
51: .4→T[15]→T[16];.5→T[17]→T[18];.4→T[19]→T[20]→T[21]→T[22]
52: .3→T[23];.5→T[24]→T[25];.4→T[26]→T[28];.3→T[29];.4→T[30]
53: .5→T[31]→T[32];.67→T[13]→T[14]→T[27]
54: for I=1 to 32;fti (100T[I])→B$[98(I-1)+3,98(I-1)+4];next I
55: .4→T[2]→T[3];.5→T[4]→T[5];.4→T[6];.5→T[7];.4→T[8]→T[9]→T[10]→T[11]→T[12]
56: .4→T[15]→T[16];.5→T[17]→T[18];.4→T[19]→T[20]→T[21]→T[22];.3→T[23]
57: .5→T[24]→T[25];.4→T[26]→T[28]→T[30]
58: .3→T[29];.5→T[31]→T[32];.67→T[1]→T[13]→T[14]→T[27]
59: for I=1 to 32;L[I]-F[I]→K
*776
```

```
60: fti (100T[I])→B$[98(I-1)+2K-1,98(I-1)+2K];next I
61: .25→T[1]→T[3]→T[14];.2→T[2]→T[6]→T[7]→T[8]→T[10]→T[12]→T[15]→T[16]→T[19]
62: .2→T[20]→T[21]→T[24]→T[25]→T[27]→T[28]→T[32]
63: .3→T[4]→T[5]→T[11]→T[17]→T[18]→T[31];.5→T[13]
64: .1→T[9]→T[22]→T[23]→T[26]→T[29]→T[30]
65: for I=1 to 32;L[I]-F[I]+1→K
66: fti (100T[I])→B$[98(I-1)+2K-1,98(I-1)+2K];next I
67: ret
68: "Setup":
69: wrt "Analyzer","AA1AB0AX0SC2PA0PB0PX0CH0PS2FR1RP1MD2IM1TR0RR0"
70: wrt "Analyzer","LAS";red "Analyzer",S
71: 10↑((S-80)/10)→C
72: 1023/80→D;8/1023→E
73: ret
74: "Pass_1":wrt "Analyzer","SP8RUNRE"
75: wait 2000
76: if A$#"yes";9to +3
77: wrt "Analyzer","LST0"
78: if bit(6,rdb("Analyzer"))=0;9to -1
79: 1→M;12→L;9sb "Third_octave"
80: ret
81: "Pass_2":wrt "Analyzer","SP12RUNRE"
82: wait 1000
83: if A$#"yes";9to +3
84: wrt "Analyzer","LST0"
85: if bit(6,rdb("Analyzer"))=0;9to -1
86: 13→M;25→L;9sb "Third_octave"
87: ret
88: "Pass_3":wrt "Analyzer","SP14RUNRE"
89: wait 1000
90: if A$#"yes";9to +3
91: wrt "Analyzer","LST0"
92: if bit(6,rdb("Analyzer"))=0;9to -1
93: 26→M;32→L;9sb "Third_octave"
94: ret
95: "Plot_response":for I=1 to 32
96: D(T[33-I]+80-S)→P
97: if P>1023;1023→P
98: if P<0;0→P
99: for J=1 to 8;P→M[8I+J-8];next J;next I
100: wrt "Analyzer","WTM,74400,256"
101: for I=1 to 256;wtb 731,shf(M[I],8);wtb 731,M[I];next I
102: ret
103: "Annotation":fmt 1,c5,f3.0,c29
104: wrt "Analyzer.1","WTA1,",S," dBV FULLSCALE --- 10 dB/DIV "
105: wrt "Analyzer","WTA2,                                    "
106: wrt "Analyzer","WTA3,16 Hz                      20 KHz"
107: wrt "Analyzer","WTA4,   32 THIRD-OCTAVE SEGMENTS      "
108: ret
109: "Third_octave":wrt "Analyzer","HLTLFM,74400,256";red "Analyzer"
110: for I=1 to 256;ior(shf(rdb(731),-8),rdb(731))→M[I];next I
111: for I=1 to 256;Eband(M[I],1023)→M[I]
112: if M[I]=0;9to +2
113: 10↑(M[I]+S/10-8)→M[I]
114: next I
115: for I=M to L;0→P;for J=F[I] to L[I]
116: J-F[I]+1→K;B$[98(I-1)+2K-1,98(I-1)+2K]→C$
117: P+.01itf(C$)M[257-J]→P;next J
118: if P<C;9to +2
119: 10log(P)→T[I];9to +2
120: S-80→T[I]
121: next I
122: ret
*9859
```

HEWLETT **hp** PACKARD

5952-8800