VOLUME 2

APPENDICES

INTRODUCTION

In Volume 1 of this thesis reference is made to five appendices where the details of points which are of particular interest can be found.

The appendices concerned are all included in this volume. However, before attempting to read any of these appendices the reader is advised to consult the relevant parts of the main thesis in Volume 1. This point is particularly important in the case of the readers who wish to either use, or just fully understand the contents of appendices B and E.

CONTENTS

APPENDIX A

Determination of the time constant of the SCALE forward path integrator

The block diagrams shown in Fig. (A1) and (A2) represent Delta-sigma and the linear Deltamodulator, respectively. It can be seen that the Delta-sigma decoder - unlike the linear Deltamodulation decoder - has no RC integrator; but instead the integrator has been removed to the forward path of the encoder.

Johnson[49] shows that for a linear Deltamodulation system - having a sampling frequency f_p, a minimum peak-to-peak feedback step height d, a binary output sequence L(t) with voltage levels ±V and a characteristic frequency $f_1 = 1/2\pi C_1 R_1$ - the total mean square quantization noise voltage N_q^2 (at the decoder output) is

$$N_q^2 = \frac{4\pi^2 V^2 f_1^2 f_a}{3f_p^3} \qquad (1)$$

f_a being the upper cut-off frequency of the decoder output filter F_2. It is assumed that $f_a >> f_b$ (the lower cut-off frequency of F_2).

Johnson arrives at the above relation assuming that the spectral distribution of the total error signal is of the form $(\sin x/x)^2$ with the first null at $f_p H_Z$ as shown in Fig. (A3).

For the linear Deltamodulation system the overload characteristic and total quantization noise power (on dB scale) as functions of the signal frequency are of the form shown in Fig. (A4).

Johnson points out that when the input signal amplitude is less than d/2 the idling pattern (010101...) will not be disturbed and thus no coding will take place. For this reason the situation when the amplitude of the input signal is equal to d/2 is called the threshold

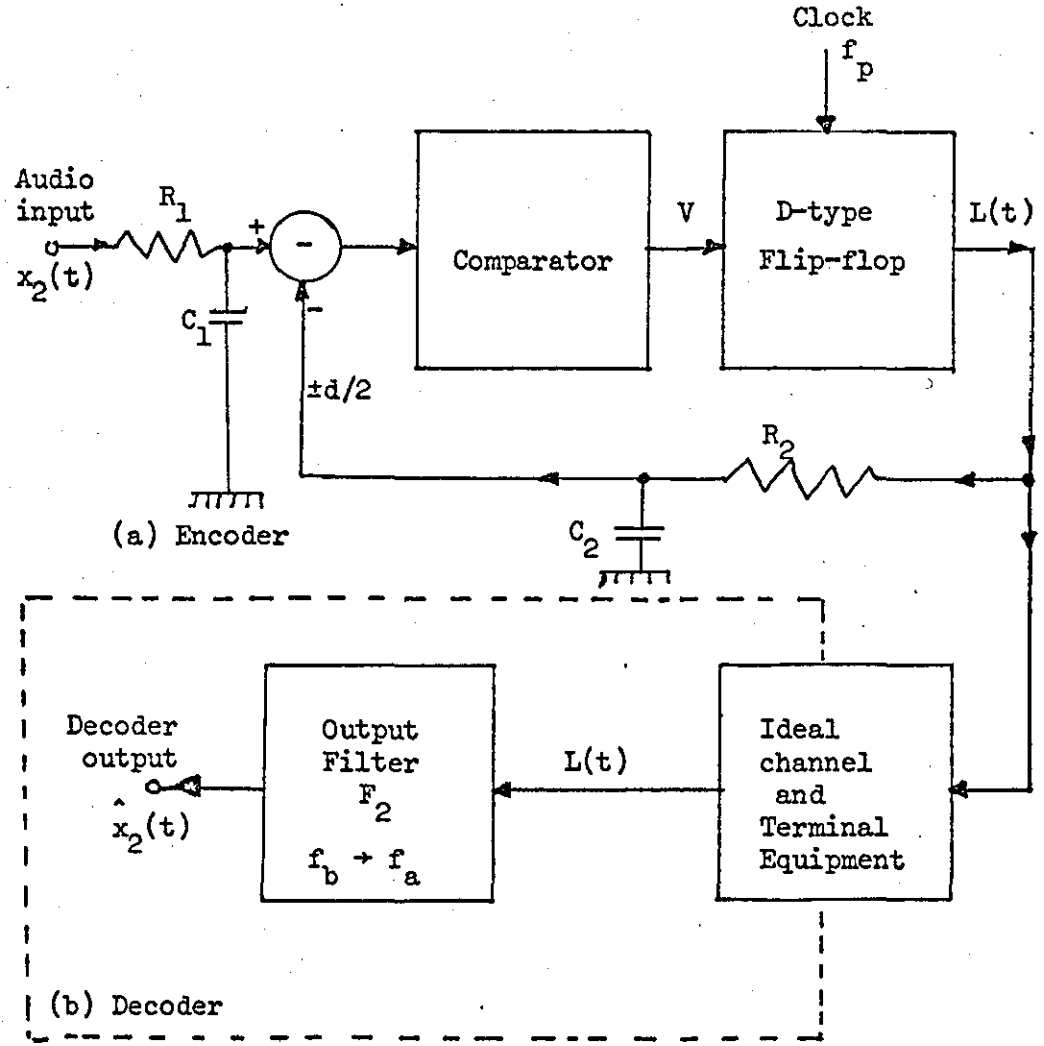


Figure (A1) Delta-sigma modulator system
(a) Encoder, (b) Decoder

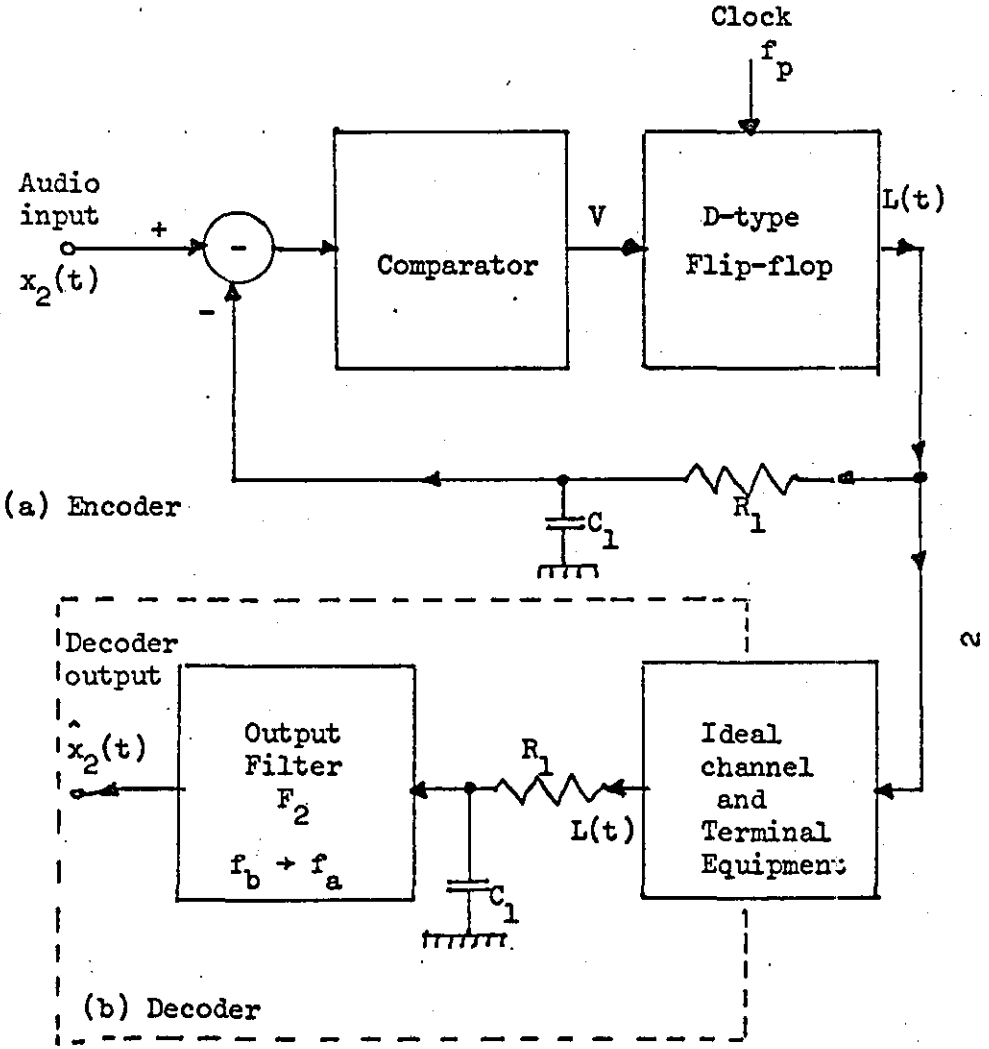


Figure (A2) Linear Deltamodulation system
(a) Encoder, (b) Decoder

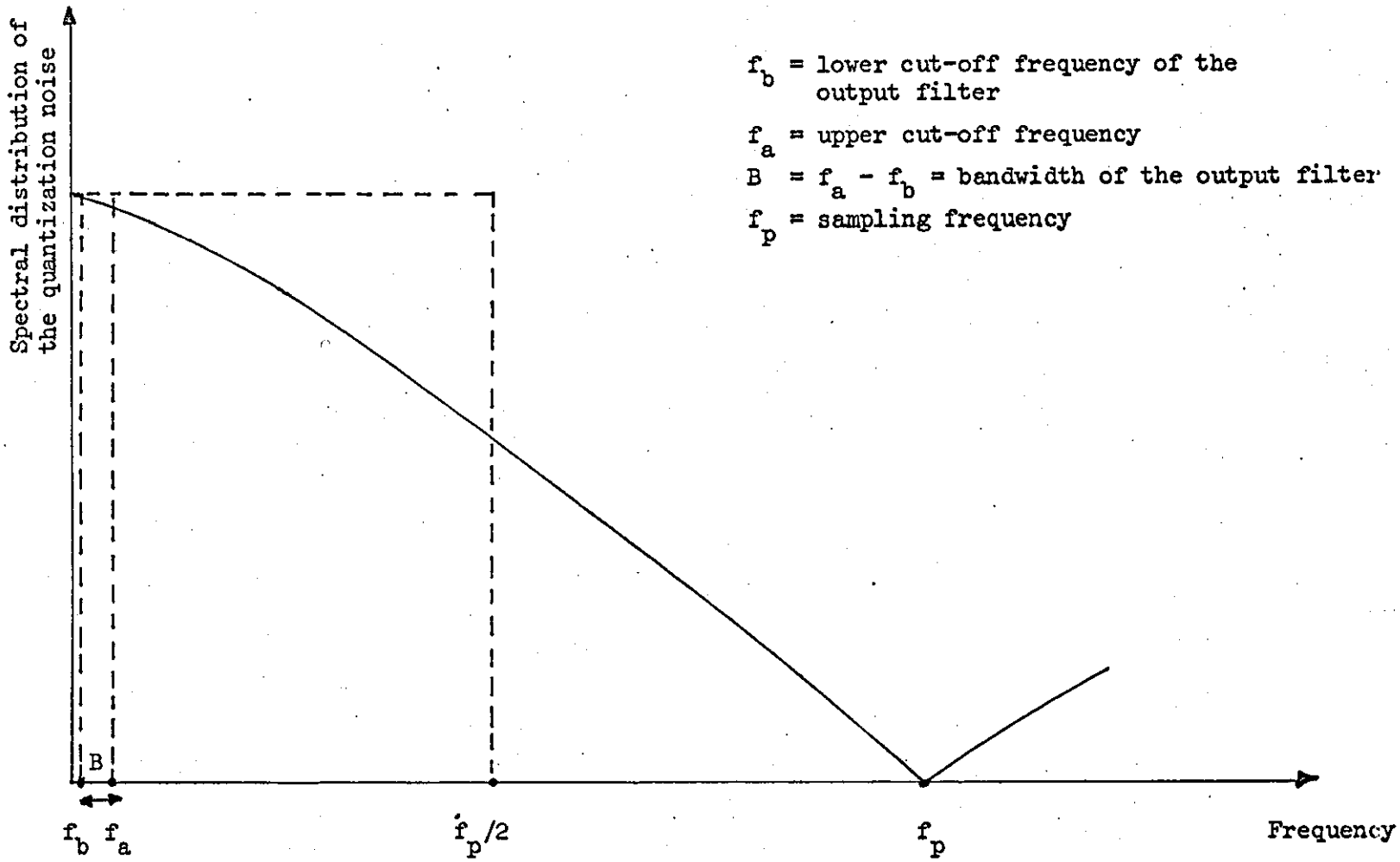


Figure (A3) Assumed frequency distribution of quantization noise energy

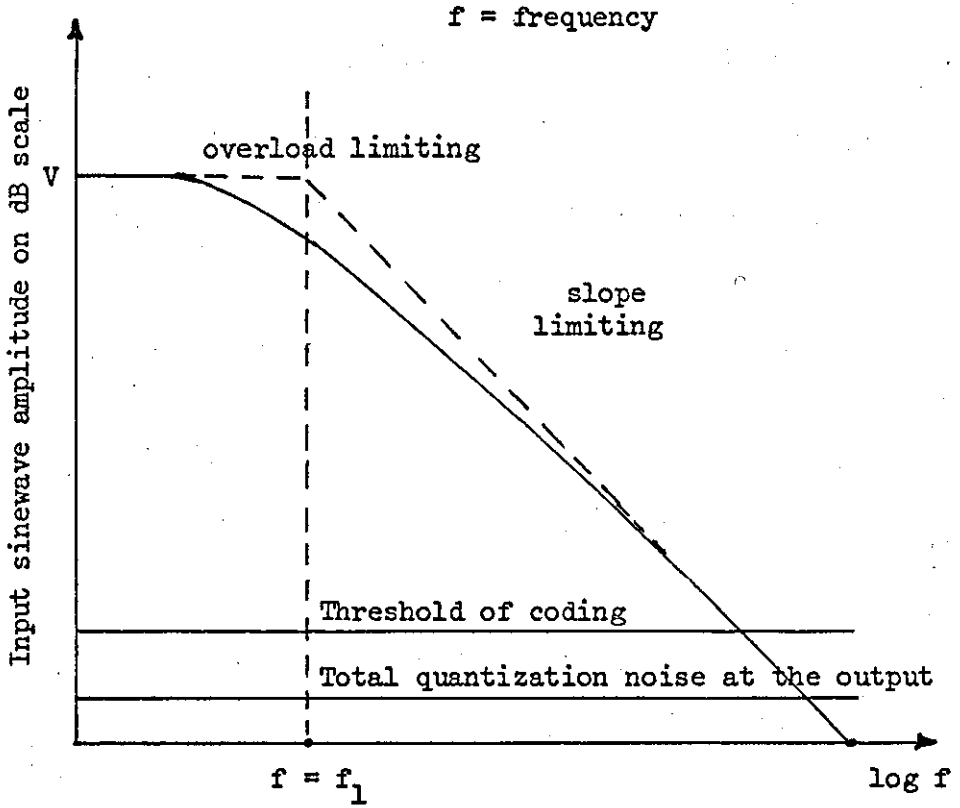


Figure (A4) Linear Deltamodulator characteristic

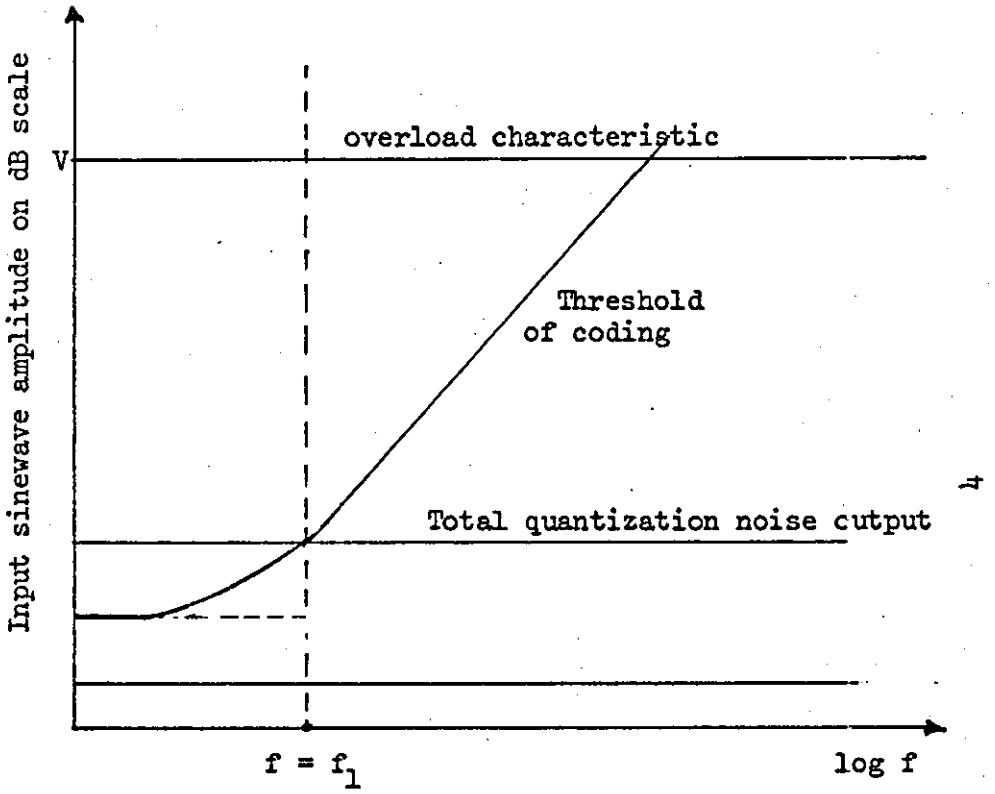


Figure (A5) Delta-sigma modulator characteristic

of coding. The threshold of coding must of course be above the quantization noise power level and is a factor for determining the lower limit of the input signal.

For f_p = 20 Kb/s, and B = $f_a - f_b$ = 2.4 KH_Z Johnson suggests that the minimum threshold of coding should be in the range of 5 dB. This corresponds to $f_p/f_a = 8.5$.

In the case of Delta-sigma modulator the overload and the threshold characteristics, as functions of frequency, are of the form shown in Fig. (A5). The overload level is at V for all input frequencies and independent of all other parameters. This is because the omission of the integrator at the decoder produces a lift of the higher frequencies which compensates for the attenuation caused by the encoder forward path integrator. As a result of achieving flat overload characteristic the other characteristics are also modified. The quantization noise spectrum is no longer flat, but rises at 6 dB per octave above f_1. The threshold of coding is similarly modified. To account for these modifications resulting from the omission of the integrator from the Deltamodulator decoder (and reinserting it in the encoder forward path), the following approach is adopted. This approach was first suggested by Johnson, and later by Cartmale and Steele[35]. It proceeds as follows:

Instead of using $(d^2/3f_p)$ as the energy density of the quantization noise, we use the frequency dependent expression

$$N_q^2 = \left[\frac{d^2(f_1^2 + f^2)}{3f_p f_1^2}\right] = \frac{4\pi^2 V^2}{3f_p^3}\left[f_1^2 f_a + \frac{(f_a^3 - f_b^3)}{3}\right] \qquad (2)$$

(See also equations (2.6), (2.19) and (2.21) in the text.)

Comparison of equations (1) and (2) above shows that when $f_1^2 f_a >> (f_a^3 - f_b^3)/3$ the quantization noise at the output of the Delta and Delta-sigma decoders are identical. When $f_1^2 f_a << (f_a^3 - f_b^3)/3$ the Delta-sigma quantization noise becomes constant and independent of f_1.

Thus for optimum coder operation we should aim for f_1 which is just large enough to cause the quantization noise to begin to rise.

For this Johnson suggests the relation

$$10 f_1^2 f_a = \frac{1}{3}(f_a^3 - f_b^3) \tag{3}$$

This results in a forward integrator time constant given by

$$T_1 = C_1 R_1 = \frac{1}{2\pi f_1} = \frac{1}{2\pi \sqrt{\frac{(f_a^3 - f_b^3)}{30 f_a}}} \tag{4}$$

Wilkinson[16,17] suggests a similar relation for syllabically companded Delta-sigma modulation systems in which

$$T_1 \approx \frac{1}{3\pi \sqrt{\frac{(f_a^3 - f_b^3)}{30 f_a}}} \tag{5}$$

Such time constants result in approximately 1 dB increase in quantization power above its minimum value.

APPENDIX B

This appendix is devoted to the listing of FORTRAN programs and subprograms which were used in the computer simulation experiments discussed in this thesis.

To facilitate the understanding, and the use, of these routines by the reader a flow chart is provided whenever necessary, each routine is briefly described and the meaning of the main parameters employed by the routine is explained. The FORTRAN statements are then listed.

Before reading this appendix the reader should refer to the relevant parts of the computer simulation sections in Chapters III, IV, V, VI and/or VII.

A list of the subprograms included in this appendix is given below.

LIST OF THE SUBPROGRAMS INCLUDED IN APPENDIX B

1. SUBROUTINE SIGNAL
2. SUBROUTINE SPEECH
3. SUBROUTINE GAUSS2
4. SUBROUTINE NLOGN
5. SUBROUTINE CONVOL
6. SUBROUTINE FILTER
7. SUBROUTINE CODEC
8. SUBROUTINE DIRECTION
9. SUBROUTINE HISTORY
10. SUBROUTINE AUCOUNTER
11. SUBROUTINE ADDER
12. SUBROUTINE DAC
13. SUBROUTINE INTERPOLATION
14. SUBROUTINE ACCUMULATOR
15. SUBROUTINE FILFUN
16. SUBROUTINE COMPRESSION
17. SUBROUTINE OPERATION
18. SUBROUTINE ESTIMATION
19. SUBROUTINE DECODERD
20. SUBROUTINE EXPANDER
21. SUBROUTINE ATODCONVERTER
22. SUBROUTINE COMPARITOR
23. SUBROUTINE SUBTRACTOR
24. SUBROUTINE ALAWRX

1. SUBROUTINE SIGNAL

The function of this subprogram is to generate step waveforms of assignable magnitudes and sinusoids of assignable magnitudes and frequencies. The subprogram is executed when the following statement is encountered in the master segment:

CALL SIGNAL (N, XT, A, JF, JA, JB, FP)

where

N = The number of steps or sinewave samples to be generated

XT = A real array where the output samples are stored

A = A real array where the amplitude samples of the sinewaves or step waveforms are stored.

JF, JA and JB are integer variables, the values of which are defined in the master segment before the CALL statement. When JB > 0 a step waveform is generated. The magnitude of the step is decided by the value of JA, e.g. when JA = 1, the step magnitude is 1.5 volts and it is equal to 3 volts when JA is made equal to 6. When JB $\leq$ 0 the waveform generated by the routine is a sinusoid, the frequency and amplitude of which are determined by JF and JB respectively; e.g. when JA = 6 and JF = 1 the waveform generated is a sinusoid with amplitude of 3 volts and frequency of 300 H_Z.

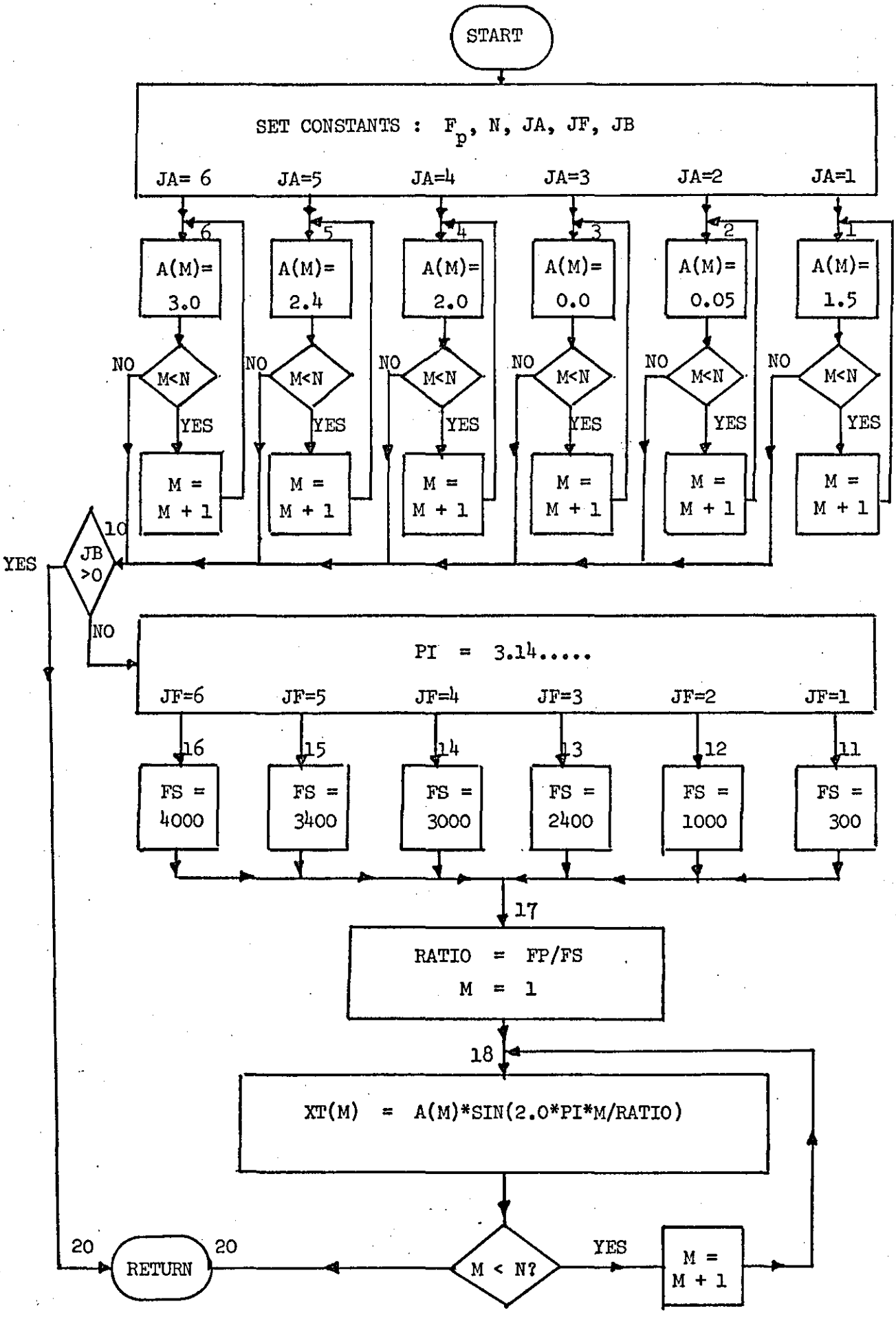


Figure B1 Flow chart of "SUBROUTINE SIGNAL"

```
C
C
C
C
      SUBROUTINE SIGNAL(N,XT,A,JF,JA,JB,FP)
      DIMENSION XT(N),A(N)
      PI=3.141593
      GO TO(1,2,3,4,5,6),JA
    1 DO55M=1,N
   55 A(M)=1.5
      GO TO 10
    2 DO56M=1,N
   56 A(M)=0.05
      GO TO 10
    3 DO57M=1,N
   57 A(M)=0.0
      GO TO 10
    4 DO58M=1,N
   58 A(M)=2.0
      GO TO 10
    5 DO59M=1,N
   59 A(M)=2.40
      GO TO 10
    6 DO60M=1,N
   60 A(M)=3.0
   10 IF(JB.GT.0)GO TO 20
      GO TO (11,12,13,14,15,16),JF
   11 FS=300.0
      GO TO 17
   12 FS=1000.0
      GO TO 17
   13 FS=2400.0
      GO TO 17
   14 FS=3000.0
      GO TO 17
   15 FS=3400.0
```

```
      GO TO 17
   16 FS=4000.0
   17 FRATIO=FP/FS
      DO18M=1,N
      TK M
      XT(M)=A(M)*SIN(2.0*PI*TK/FRATIO)
   18 CONTINUE
   20 RETURN
      END
```

2. SUBROUTINE SPEECH

This subprogram was designed to fetch 200 samples of a speech waveform which has been sampled at a frequency equal to 9.6 Kb/s and stored on a magnetic tape. When these samples are recovered they are operated upon by the subprogram to double the sampling frequency (using interpolation techniques) and subsequently filtered to produce the waveform shown in Figure 3.3. The actual values of the samples in millivolts, as read from the magnetic tape are listed in Table B1.

The subprogram is executed by the computer when the following statement is encountered in the master segment:

```
CALL SPEECH (INPUT, INPUT1, INPUT2, XT, CLP, ISAMPL, NSAMPL,
                    HREAL, NOUT, XTF, NHALF1)
```

where the arguments have the following meanings:

INPUT = An integer array for storing 100 speech samples

INPUT1, INPUT2, XTF } = Arrays which are used as working spaces

CLP = A real array for holding the time in clock periods

ISAMPL = The number of samples to be "returned" to the master segment

NSAMPL = The number of coefficients in the filter impulse response

NHALF1 = NSAMPL/2

NOUT = NSAMPLE + ISAMPL - 1

HREAL = A real array for storing the samples of the filter impulse response.

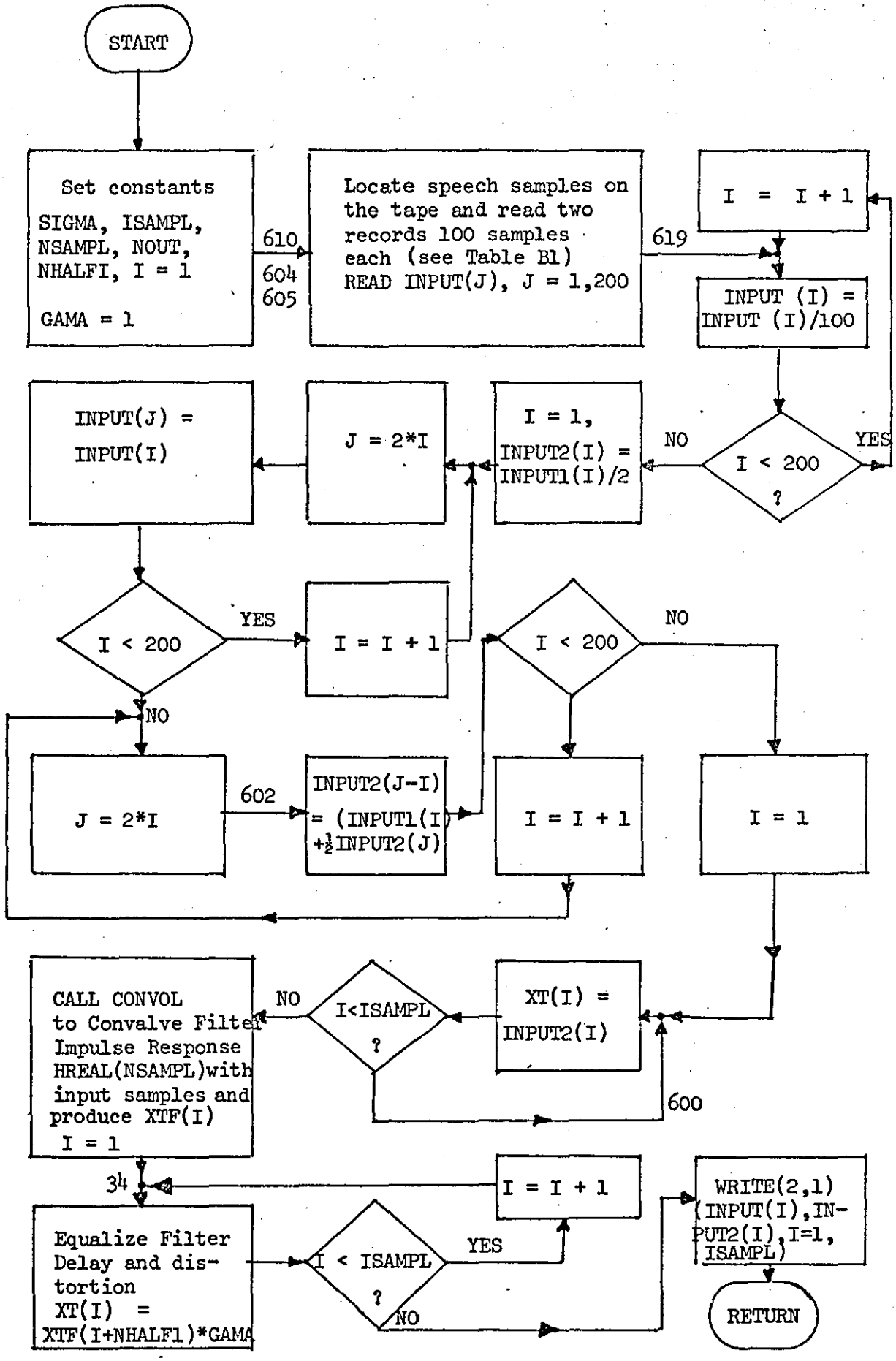


Figure B 2 Flow chart of "SUBROUTINE SPEECH"

```
      SUBROUTINE SPEECH(INPUT,INPUT1,INPUT2,XT,CLP,ISAMPL,NSAMPL,HREA
     1NOUT,XTF,NHALF1)
      REAL INPUT1
      REAL INPUT2
      DIMENSION INPUT(100)
      DIMENSION INPUT1(420)
      DIMENSION INPUT2(420)
      DIMENSION XT(ISAMPL)
      DIMENSION CLP(ISAMPL)
      DIMENSION HREAL(NSAMPL)
      DIMENSION XTF(400)
      DO610I=1,590
      READ(6)INPUT
   15 FORMAT(1H ,I3,I10)
  610 CONTINUE
      READ(6)INPUT
      DO604I=1,100
  604 INPUT1(I)=INPUT(I)
      READ(6)INPUT
      DO605I=1,100
  605 INPUT1(I+100)=INPUT(I)
      DO619I=1,200
      INPUT1(I)=INPUT1(I)/100.0
  619 CONTINUE
      INPUT2(1)=INPUT1(1)/2.0
      DO601I=1,200
      J=2*I
  601 INPUT2(J)=INPUT1(I)
      DO602I=2,200
      J=2*I
  602 INPUT2(J-I)=(INPUT1(I)+INPUT1(J))/2.0
      DO600I=1,ISAMPL
      XT(I)=INPUT2(I)
  600 CONTINUE
      CALL CONVOL(NSAMPL,HREAL,ISAMPL,XT,NOUT,XTF)
       DO34I=1,ISAMPL
       XT(I)=XTF(I+NHALF1)*GAMA
   34 CONTINUE
      REWIND 6
     RETURN
    END
```

Group Number	Sample Number within a group				
	1	2	3	4	5
1	0.19	0.20	0.15	0.09	-0.02
2	-0.07	-0.01	0.04	0.05	-0.02
3	-0.09	0.02	0.19	0.22	0.18
4	0.14	0.00	-0.09	-0.02	0.00
5	-0.16	-0.29	-0.29	-0.11	0.12
6	0.10	-0.06	-0.10	-0.08	-0.06
7	-0.80	-0.16	-0.25	-0.22	-0.14
8	-0.12	-0.07	0.10	0.26	0.24
9	0.06	-0.07	-0.08	-0.10	-0.14
10	-0.03	0.13	0.00	-0.31	-0.49
11	-0.48	-0.40	-0.30	-0.20	-0.10
12	-0.03	-0.06	-0.12	-0.08	0.11
13	0.26	0.24	0.03	-0.17	-0.22
14	-0.16	-0.14	-0.31	-0.37	-0.41
15	-0.45	-0.28	0.07	0.28	0.20
16	0.06	-0.08	-0.22	-0.23	-0.05
17	0.15	0.12	-0.10	-0.20	-0.05
18	0.05	0.00	-0.06	-0.03	0.06
19	0.13	0.16	0.22	0.27	0.20
20	0.06	0.07	0.21	0.19	0.00

Table B1 A record of 100 speech samples on magnetic tape (Record 1)

Group Number	Sample Number within a group				
	1	2	3	4	5
1	-0.18	-0.15	-0.02	0.04	0.04
2	0.02	-0.08	-0.28	-0.32	-0.15
3	0.23	0.47	0.43	0.26	0.13
4	0.12	0.14	0.07	-0.02	-0.07
5	-0.16	-0.23	-0.20	-0.13	-0.13
6	-0.16	-0.23	-0.31	-0.26	-0.03
7	0.39	1.19	2.29	2.64	1.32
8	-1.28	-4.08	-4.69	-3.05	-1.13
9	-0.47	-0.68	-0.90	-0.59	0.55
10	2.07	3.07	2.88	1.28	-0.48
11	-1.03	-0.25	0.55	0.54	-0.30
12	-1.40	-1.83	-1.21	-0.01	0.89
13	0.68	-0.16	-0.33	-0.39	0.91
14	0.75	0.32	-0.08	-0.30	-0.25
15	-0.03	0.38	0.34	-0.16	-0.54
16	-0.57	0.25	0.45	0.24	-0.10
17	-0.22	-0.09	0.14	0.29	0.22
18	-0.10	-0.30	-0.15	-0.03	-0.24
19	-0.33	0.47	2.23	3.81	3.72
20	1.76	-1.12	-3.84	-4.73	-3.25

Table B1 cont — A record of 100 speech samples on magnetic tape (Record 2)

3. SUBROUTINE GAUSS2

This subprogram generates a signal with Gaussian probability distribution. It makes use of the computer library function G05AEF (see Chapter III Section 2.3.c). It generates the wide band Gaussian signal and then bandlimits it by low-pass filtering it. The "call" statement for the routine is:

```
CALL GAUSS2 (NSAMPL, HREAL, ISAMPL, GNOISE, NOUT, BLGNSE, MU,
             DBMNL, JZ, SIGMA, XT, DBMN2, RMSNE)
```

where ISAMPL, NSAMPL, NOUT and HREAL have the meanings given in Section 2 of this appendix, and:

GNOISE = A real array holding the wide band Gaussian signal samples

BLGNSE = A real array holding the filtered Gaussian signal samples

SIGMA = The standard deviation of the distribution

MU = The mean of the distribution

RMSNE = An array for holding the r.m.s. values of the bandlimited waveform

DBMNL, DBMN2 = Arrays for holding the value of the bandlimited waveform in dB when the source impedance is 1 Ohm or 600 Ohm respectively.

JZ = A constant used as a switching parameter (see FORTRAN listing).

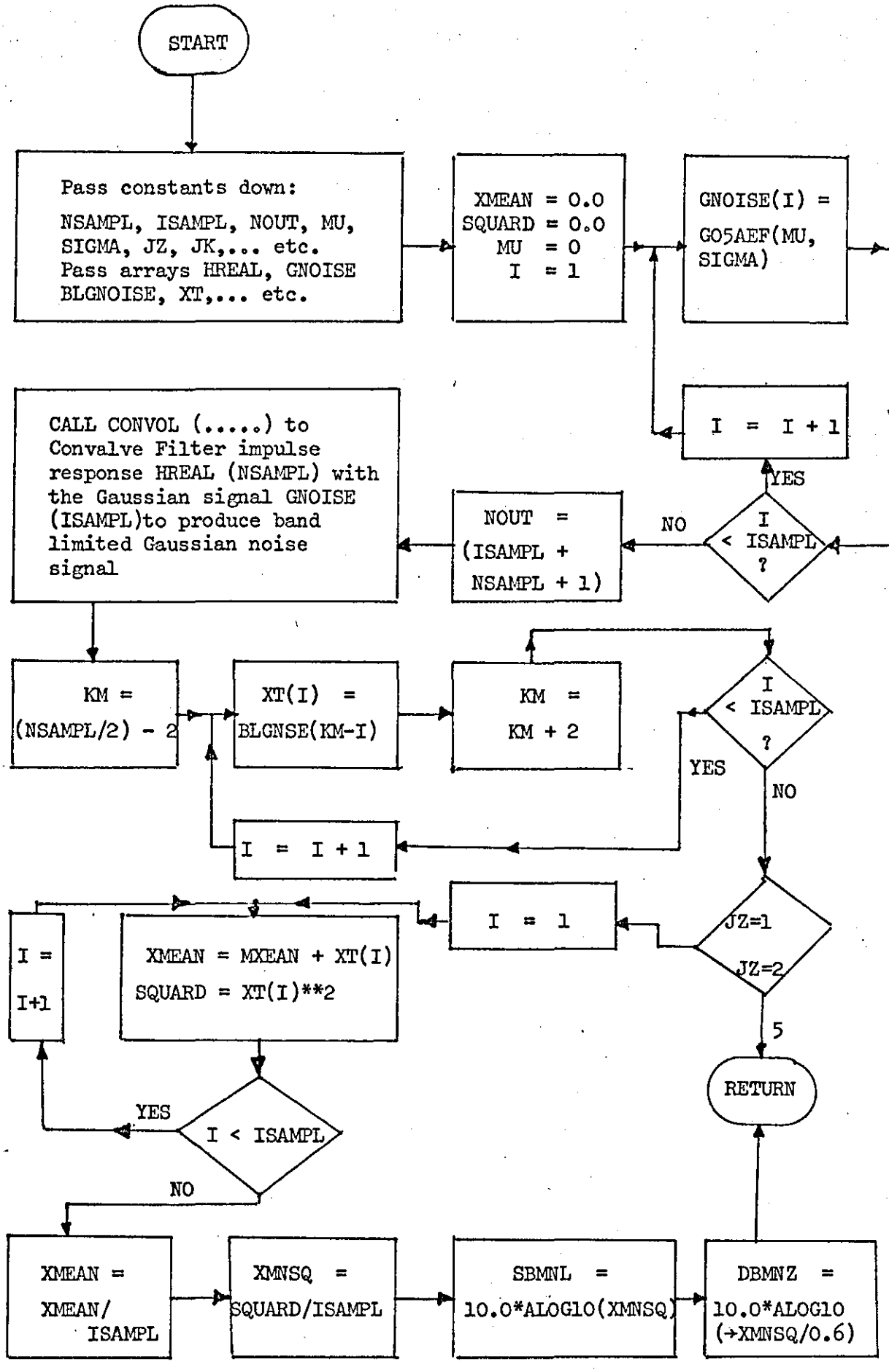


Figure B3 Flow chart of "SUBROUTINE GAUSS2(....)"

```
C
      SUBROUTINE GAUSS2(NSAMPL,HREAL,ISAMPL,GNOISE,NOUT,BLGNSE,
     1DBMNL,
     1JZ,SIGMA,
     1XT,DBMN2,
     1RMSNE)
      REAL MU
      DIMENSION HREAL(NSAMPL)
      DIMENSION XT(ISAMPL)
      DIMENSION BLGNSE(NOUT)
      DIMENSION GNOISE(ISAMPL)
      MU=0.0
      XMEAN=0.0
      SQUARD=0.0
      DO1I=1,ISAMPL
    7 GNOISE(I)=G05AEF(MU,SIGMA)
    1 CONTINUE
      NOUT=NSAMPL+ISAMPL-1
      CALL CONVOL(NSAMPL,HREAL,ISAMPL,GNOISE,NOUT,BLGNSE)
      KM=NSAMPL/2+2
      DO2I=1,ISAMPL
      XT(I)=BLGNSE(KM-I)
      KM=KM+2
    2 CONTINUE
      GO TO(5,6),JZ
    6 DO3I=1,ISAMPL
      XMEAN=XMEAN+XT(I)
      SQUARD=SQUARD+XT(I)**2
    3 CONTINUE
      XMEAN=XMEAN/ISAMPL
      XMNSQ=SQUARD/ISAMPL
      RMSNE=SQRT(XMNSQ)
      DBMNL=10.0*ALOG10(XMNSQ)
      DBMN2=10.0*ALOG10(XMNSQ/0.6)
    5 RETURN
      END
```

4. SUBROUTINE NLOGN

The function of this subprogram is to accept a block of data and perform the Discrete Fast Fourier Transform (DFFT) or the Inverse Discrete Fast Fourier Transform (IDFFT) on it as required. The call for the subprogram is:

CALL NLOGN (N, X, LX, DIR)

where

X = A real array which holds the input data when the subprogram is called. At the end of the execution of the subprogram X holds the transformed data block.

LX = The number of data samples held in X.

N = $\log_2(LX)$

The DFFT or the IDFFT operation is performed depending on whether DIR = -1.0 or 1.0 respectively.

```
C
C
C
C
      SUBROUTINE NLOGN(N,X,LX,DIR)
      COMPLEX X,WK,HOLD,Q
      DIMENSION M(25)
      DIMENSION X(LX)
      DO 1 I=1,N
    1 M(I)=2**(N-I)
      DO 4 L=1,N
      NBLOCK=2**(L-1)
      LBLOCK=LX/NBLOCK
      LBHALF=LBLOCK/2
      K=0
      DO 4 IBLOCK=1,NBLOCK
      FK=K
      FLX=LX
      V=DIR*6.28318531*FK/FLX
      WK=CMPLX(COS(V),SIN(V))
      ISTART=LBLOCK*(IBLOCK-1)
      DO 2 I=1,LBHALF
      J=ISTART+I
      JH=J+LBHALF
      Q=X(JH)*WK
      X(JH)=X(J)-Q
      X(J)=X(J)+Q
    2 CONTINUE
      DO 3 I=2,N
      II=I
      IF(K.LT.M(I)) GO TO 4
    3 K=K-M(I)
    4 K=K+M(II)
      K=0
      DO 7 J=1,LX
      IF(K.LT.J) GO TO 5
      HOLD=X(J)
      X(J)=X(K+1)
      X(K+1)=HOLD
```

```
5 DO 6 I=1,N
  II=I
  IF(K.LT.M(I)) GO TO 7
6 K=K-M(I)
7 K=K+M(II)
  IF(DIR.LT.0.0) RETURN
  DO 8 I=1,LX
8 X(I)=X(I)/FLX
  RETURN
  END
```

5. SUBROUTINE CONVOL

This subprogram accepts two blocks of data and produces their convolution. It is executed when the following statement is encountered in the master segment:

CALL CONVOL (NSAMPL, HREAL, LARRAY, YT, KARRAY, YTFTWO)

where

HREAL = An array holding the filter impulse response samples

NSAMPL = The number of samples in HREAL

YT = An array holding the signal samples to be filtered

LARRAY = The number of data points in YT

YTFTWO = An array for holding the results of the convolution of YT and HREAL

KARRAY = The number of points in YTFTWO.

It must be ensured that the length of the array reserved in the computer for YTFTWO is not less than (NSAMPL + LARRAY - 1).

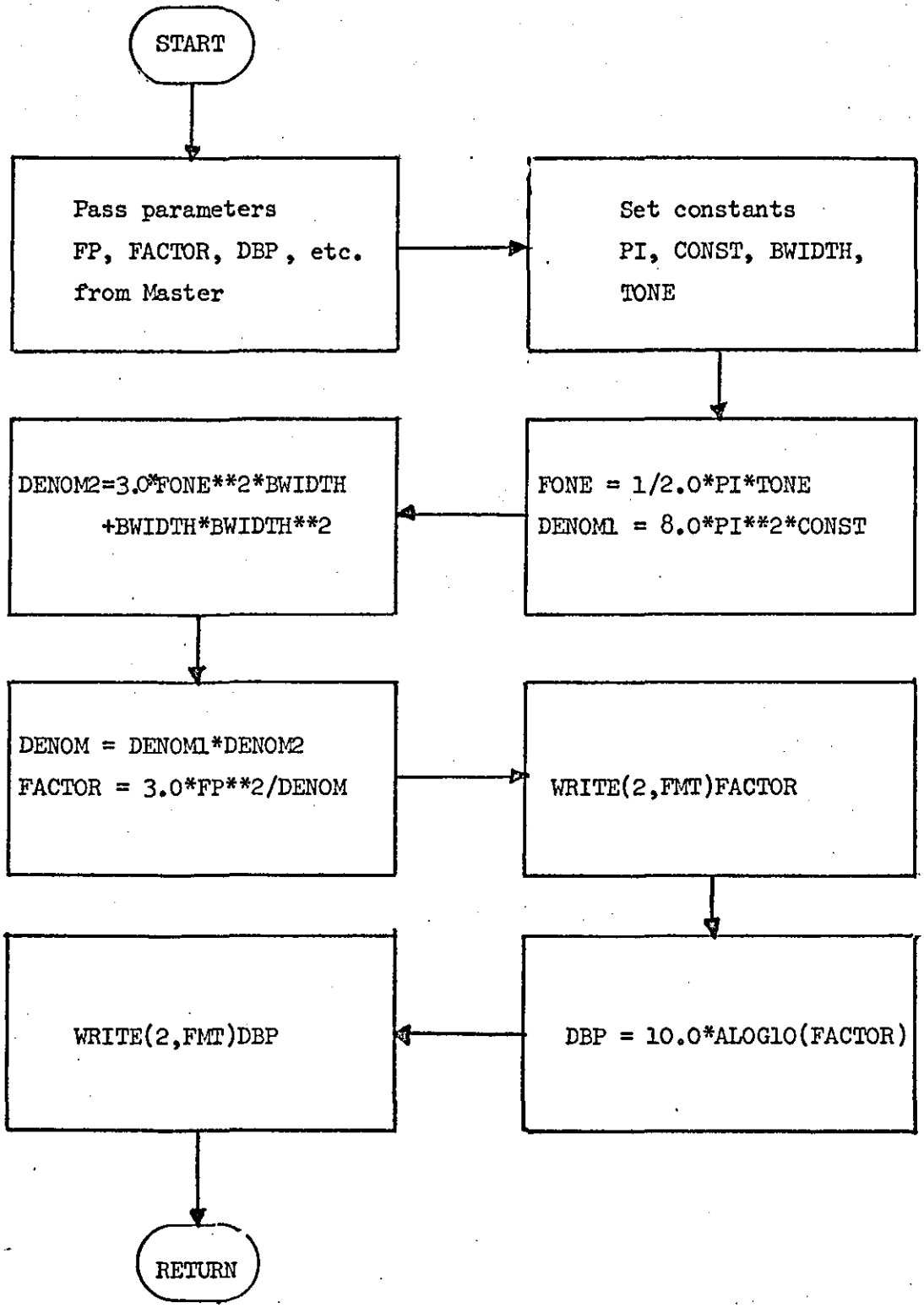


Figure B4 Flow chart of "SUBROUTINE CONVOL"

```
C
C
C
C
      SUBROUTINE CONVOL(NSAMPL,HREAL,LARRAY,YT,KARRAY,YTFTWO)
      DIMENSION HREAL(NSAMPL),YT(LARRAY),YTFTWO(KARRAY)
      KARRAY=NSAMPL+LARRAY-1
      DO 5I=1,KARRAY
    5 YTFTWO(I)=0.0
      DO 10I=1,NSAMPL
      DO10J=1,LARRAY
      M=I+J-1
   10 YTFTWO(M)=YTFTWO(M)+HREAL(I)*YT(J)
      RETURN
      END
```

6. SUBROUTINE FILTER

The function of this routine is to generate the frequency or the impulse response of a low-pass digital filter using the frequency sampling technique (see Chapter III). The "call" for it is:

CALL FILTER (L, NSAMPL, ORDER, HORDER, BWS, HREAL, HCOMLX, FP)

where

HORDER = A real array holding the frequency response samples of the filter in the transition band

ORDER = An integer equal to the number of data points in HORDER

BWS = An integer denoting the number of frequency samples of the filter frequency response in the pass band

FP = The sampling frequency

HREAL = An array holding the real part of the filter impulse response samples

HCOMLX = An array holding the complex samples of the filter frequency response samples

NSAMPL = The number of data points in HREAL

L = A parameter which when less than 2 causes the frequency response to be plotted

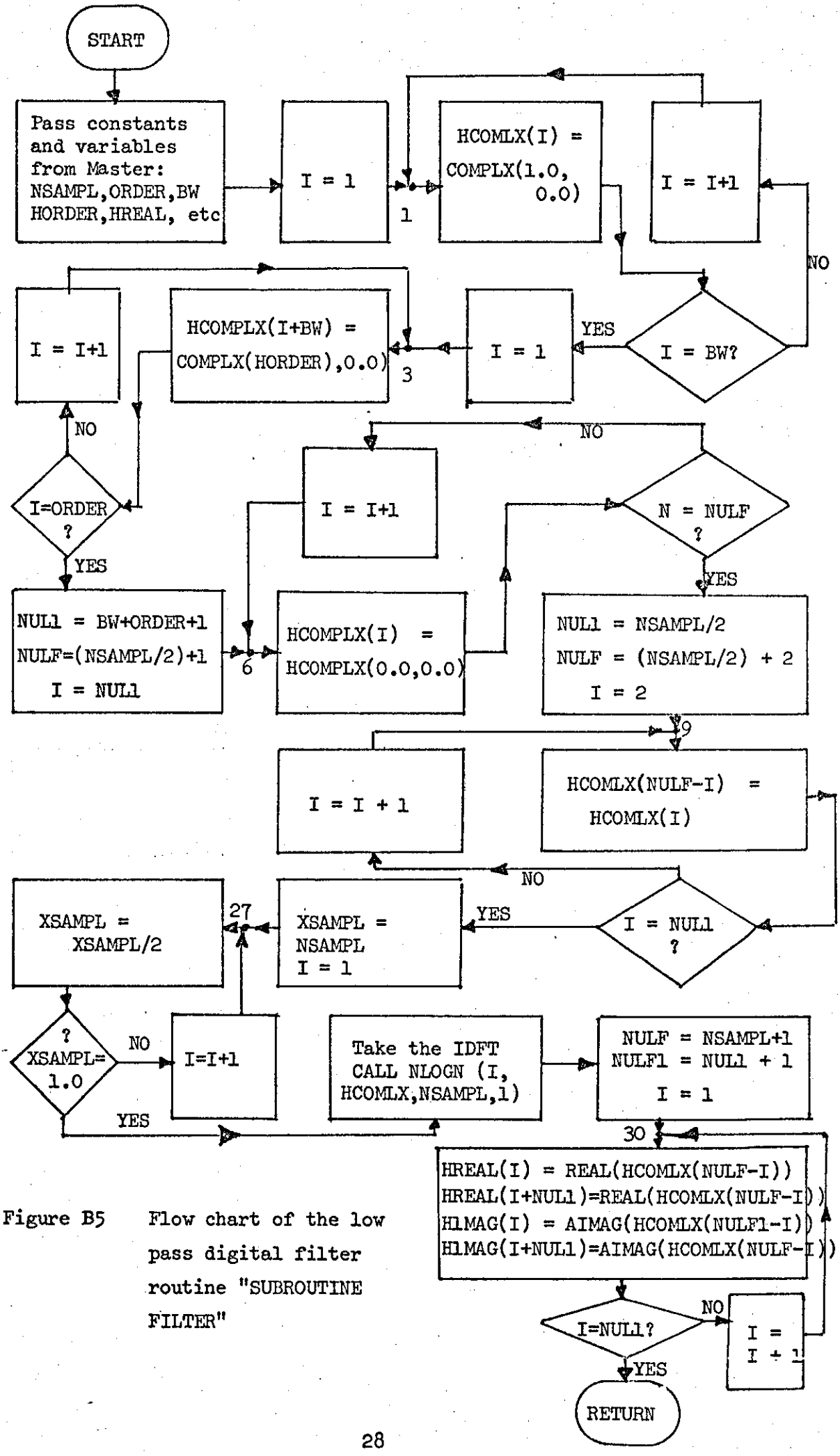


Figure B5 Flow chart of the low pass digital filter routine "SUBROUTINE FILTER"

```
C
C
C
C
      SUBROUTINE FILTER(L,NSAMPL,ORDER,HORDER,BWS,HREAL,HCOMLX,FP)
      COMPLEX HCOMLX
      INTEGER BW,ORDER
      INTEGER BWS
      DIMENSION HCOMLX(NSAMPL),HREAL(NSAMPL),HORDER(ORDER)
      DIMENSION HIMAG(128)
      DIMENSION FREQ(256)
      DIMENSION HHREAL(256)
      L=3
      IF(L.GT.2)GO TO2
      ELEMNT=FP/NSAMPL
      ELEMNT=ELEMNT/1000.0
      XMAX=9.9
      DO99J=1,NSAMPL
      K=J-1
   99 FREQ(J)=K*ELEMNT
    2 BW=BWS
      DO1I=1,BW
    1 HCOMLX(I)=CMPLX(1.0,0.0)
      DO3I=1,ORDER
    3 HCOMLX(BW+I)=CMPLX(HORDER(I),0.0)
      NUL1 BW+ORDER+1
      NUL1=BW+ORDER+1
      NULF=NSAMPL/2+1
      DO6I=NUL1,NULF
    6 HCOMLX(I)=CMPLX(0.0,0.0)
      IF(L.GT.2)GO TO 5
      DO33I=1,NULF
      HHREAL(I)=REAL(HCOMLX(I))
   33 CONTINUE
      CALL UTP4C(0.0,10.0,0.0,1.0,5.0,4.0,
     1'FREQUENCY IN KHZ',2,
     1'FILTER RESPONSE',2)
      CALL UTP4B(FREQ,HHREAL,NULF,3)
```

```
  5 NUL1=NSAMPL/2
    NULF=NSAMPL+2
    DO9I=2,NUL1
  9 HCOMLX(NULF-I)=HCOMLX(I)
  4 XSAMPL=NSAMPL
    DO27I=1,25
    XSAMPL=XSAMPL/2.0
    IF(XSAMPL.EQ.1.0)GO TO 8
 27 CONTINUE
    L=0
    GO TO 5
  8 CALL NLOGN(I,HCOMLX,NSAMPL,1.0)
    NUL1=NSAMPL/2
    NULF=NSAMPL+1
    NULF1=NUL1+1
    DO30I=1,NUL1
    HREAL(I)=REAL(HCOMLX(NULF1-I))
 30 HREAL(I+NUL1)=REAL(HCOMLX(NULF-I))
    DO31I=1,NUL1
    HIMAG(I)=AIMAG(HCOMLX(NULF1-I))
 31 HIMAG(I+NUL1)=AIMAG(HCOMLX(NULF-I))
    GO TO 34
 34 RETURN
    END
```

7. SUBROUTINE CODEC

This subprogram can implement the hybrid SCALE CODEC on the ICL 1904A digital computer in FORTRAN IV. The routine can be used to simulate the encoder or the decoder as required, with the following facilities:

(i) Variable shift register length

(ii) Variable clock frequency

The subprogram is executed when the following statement is encountered in the master segment:

CALL CODEC (NPULSE, KAA, XT, OUT, NS, FP, NNN, IJJ)

where

XT = An array holding the samples of the analogue signal to be encoded

OUT = An array holding the feedback signal samples

KAA = An array holding the SCALE binary data digits

NPULSE = The number of points in each of the above arrays

FP = The sampling frequency

NS, NN = Constant integers which are defined in the master segment to fix the shift register length employed and the operation mode (encoder or decoder), respectively; e.g. when NS = 3 and NNN = 1 the shift register to be employed consists of 3 stages and the system simulated is an encoder.

IJJ = A switching parameter (see FORTRAN listing)

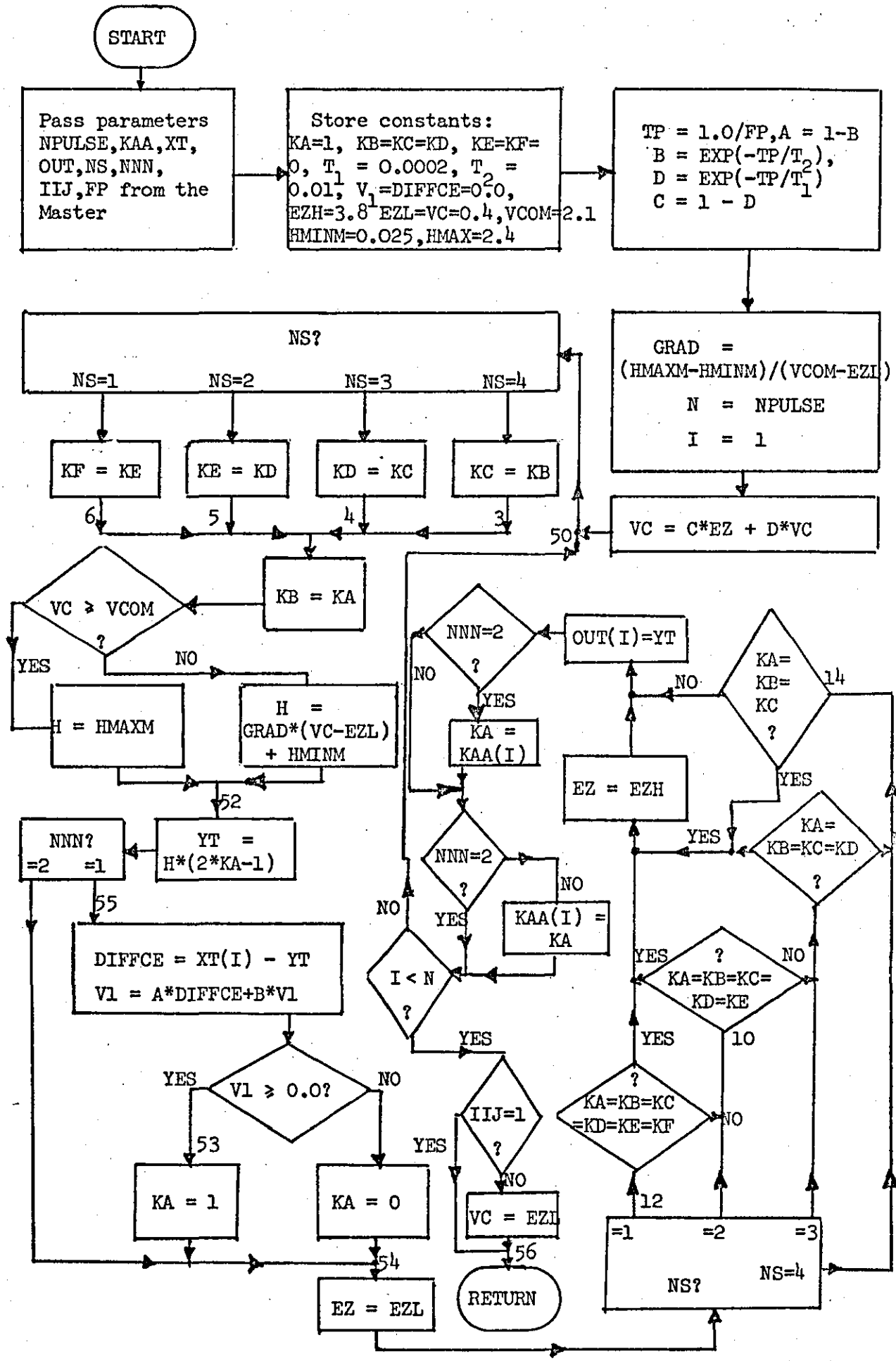


Figure B6 Flow chart of "SUBROUTINE CODEC"

```
C
C
C
C
      SUBROUTINE CODEC(NPULSE,KAA,XT,OUT,NS,FP,
     2NNN,
     1IIJ)
      DIMENSION OUT(NPULSE)
      DIMENSION XT(NPULSE),KAA(NPULSE)
      DATA T1,T2/0.0002,0.010/
      DATA KA,KB,KC,KD,KE,KF/6*0/
      DATA DIFFCE,V1,VC,EZ/2*0.0,2*0.4/
      KA=1
      TP=1.0/FP
      B=1.0/EXP(TP/T1)
      A=1.0-B
      D=1.0/EXP(TP/T2)
      C=1.0-D
      EZL=0.4
      EZH=3.8
      HMAX=2.4
      VCOM=2.1
      HMINM=0.025
      GRAD=(HMAX-HMINM)/(VCOM-EZL)
      N=NPULSE
   49 DO50I=1,N
      VC=C*EZ+D*VC
      GO TO(6,5,4,3),NS
    6 KF=KE
    5 KE=KD
    4 KD=KC
    3 KC=KB
      KB=KA
      IF(VC.GE.VCOM)GO TO 51
      H=GRAD*(VC-EZL)+HMINM
      GO TO 52
   51 H=HMAX
   52 YT=H*(2*KA-1)
      GO TO(55,54),NNN
```

```
55 DIFFCE=XT(I)-YT
   V1=A*DIFFCE+BV1
   IF(V1.GE.0.0)GO TO 53
   KA=0
   GO TO 54
53 KA=1
54 EZ EZL
   GO TO(12,10,8,14),NS
12 IF(KA.EQ.KB.AND.KB.EQ.KC.AND.KC.EQ.KD.AND.KD.EQ.KE.AND.KE.EQ.
  1KF)EZ=EZH
10 IF(KA.EQ.KB.AND.KB.EQ.KC.AND.KC.EQ.KD.AND.KD.EQ.KE)EZ=EZH
 8 IF(KA.EQ.KB.AND.KB.EQ.KC.AND.KC.EQ.KD)EZ=EZH
14 IF(KA.EQ.KB.AND.KB.EQ.KC)EZ=EZH
   OUT(I)=YT
   IF(NNN.EQ.2)KA=KAA(I)
   IF(NNN.EQ.2)GO TO 50
   KAA(I)=KA
50 CONTINUE
   IF(IIJ.EQ.1)GO TO 56
   VC=EZL
56 RETURN
   END
```

8. SUBROUTINE DIRECTION

In the DSCALE discussed in Chapter V the feedback digital signal H_{Di} was generated in the computer program by an adaptive Up-down binary counter AUDBC. The function of "SUBROUTINE DIRECTION" is to simulate the electronic circuit which accepts the SCALE binary data L(t), inspects it, and generates the control signal which governs the direction of counting (up or down) in the AUDBC. It basically simulates a 3-bit shift register followed by an exclusive OR.

The subprogram is executed by the computer when the following statement is encountered in the master segment:

CALL DIRECTION (NPULSE, KAA, IN, DIRECN)

where

NPULSE = A constant which is defined in the master segment and represents the number of SCALE binary digits L(t) to be decoded.

KAA = An integer array holding the SCALE binary data stream.

IN = An integer variable representing the time in clock periods.

DIRECN = An integer array holding the logical values of the Up-down control signal generated by the subprogram and which is returned to the master segment every clock period.

In Chapters V, VI and VII, KAA, IN and DIRECN correspond respectively to L_i, i and Z (see Chapter V Section 5.6.ii).

```
C
C
C
C
      SUBROUTINE DIRECTION(NPULSE,KAA,
     1IN,
     1DIRECN)
      INTEGER DIRECN
      DIMENSION KAA(NPULSE),DIRECN(NPULSE)
      IF(IN.GT.1)GO TO 4
      KA=1
      KB=0
      KC=0
    4 I=IN
      KC=KB
      KB=KA
      IF(KA.EQ.KB.AND.KB.EQ.KC)GO TO 2
      DIRECN(I)=0
      GO TO 3
    2 DIRECN(I)=1
    3 KA=KAA(I)
C
      IF(I.EQ.384)KAA(I)=0
C
      RETURN
      END
```

9. SUBROUTINE HISTORY

This subprogram was written to simulate the adaptive up-down counter described in Chapter V Section 5.6.ii(1). Together with "SUBROUTINE ADDER" and "SUBROUTINE AUCOUNTER"; "SUBROUTINE HISTORY" simulates the digital accumulator (which generates the DSCALE feedback digital code words H_{Di} from the SCALE binary data stream L(t) under the control of "SUBROUTINE DIRECTION").

The subprogram is executed by the computer when the following FORTRAN statement is encountered in the master segment:

CALL HISTORY (MCEL, NPULSE, DIRECN, SUM, IN, TEMSUM, NCEL)

where IN, DIRECN and NPULSE have the same meaning as in subprogram number 8 ("SUBROUTINE DIRECTION" described above), and:

MCEL = An integer array holding the partial contents S_h (see Figure 5.7) of the digital accumulator

NCEL = An integer array holding S_c the contents of the binary up-counter (see Figure 5.7)

TEMSUM = An integer array used as a working space

SUM = An integer array for holding the DSCALE digital code worde H_{Di} which is "returned" to the master segment by "SUBROUTINE ADDER".

```
C
      SUBROUTINE HISTORY(MCEL,NPULSE,DIRECN,
     2SUM,
     1IN,
     1TEMSUM,
     2NCEL)
      INTEGER SUM
      INTEGER DIRECN
      INTEGER TEMSUM
      DIMENSION TEMSUM(15)
      DIMENSION DIRECN(NPULSE)
      DIMENSION NCEL(25),MCEL(25),SUM(25)
      I=IN
      IND=0
      IF(DIRECN(I).EQ.0)GO TO 19
      GO TO 26
   19 DO25II=4,12
      IND=IND+MCEL(II)
   25 CONTINUE
      IF(IND.LT.1)RETURN
      GO TO 1
   26 L=1
      LB=0
   20 IF(MCEL(12).EQ.L)GO TO 2
      IF(MCEL(11).EQ.L)GO TO 3
   16 IF(MCEL(4).EQ.L)GO TO 4
      MCEL(4)=L
      GO TO 18
    4 MCEL(4)=LB
      IF(MCEL(5).EQ.L)GO TO 5
      MCEL(5)=L
      GO TO 18
    5 MCEL(5)=LB
      IF(MCEL(6).EQ.L)GO TO 6
      MCEL(6)=L
      GO TO 18
    6 MCEL(6)=LB
      IF(MCEL(7).EQ.L)GO TO 7
      MCEL(7)=L
      GO TO 18
   7 MCEL(7)=LB
```

```
      IF(MCEL(8).EQ.L)GO TO 8
      MCEL(8)=L
      GO TO 18
    8 MCEL(8)=LB
      IF(MCEL(9).EQ.L)GO TO 9
      MCEL(9)=L
      GO TO 18
    9 MCEL(9)=LB
      IF(MCEL(10).EQ.L)GO TO 10
      MCEL(10)=L
      GO TO 18
   10 MCEL(10)=LB
      IF(MCEL(11).EQ.L)GO TO 11
      MCEL(11)=L
      GO TO 18
   11 MCEL(11)=LB
      IF(MCEL(12).EQ.L)GO TO 12
      MCEL(12)=L
      GO TO 18
   12 WRITE(2,13)
   13 FORMAT(5X,'OVERLOAD HAS BEEN DETECTED')
      GO TO 18
    3 IF(MCEL(10).EQ.L)GO TO 14
C
      NST=2
      CALL AUCOUNTER(NCEL,DIRECN,NPULSE,
     1IN,
     2SUM,
     1NST,
     1MCEL)
C
   17 IF(MCEL(3).EQ.L)GO TO 15
      MCEL(3)=L
      GO TO 18
   15 MCEL(3)=LB
      GO TO 16
```

```
   14 GO TO 17
C
    1 L=0
      LB=1
      IF(MCEL(12).EQ.LB)GO TO 16
      IF(MCEL(11).EQ.LB)GO TO 24
      IF(MCEL(1).EQ.L)GO TO 21
      MCEL(1)=L
      GO TO 18
   21 MCEL(1)=LB
      IF(MCEL(2).EQ.L)GO TO 22
      MCEL(2)=L
      GO TO 18
   22 MCEL(2)=LB
      GO TO 14
   24 IF(MCEL(10).EQ.LB)GO TO 16
      NST=1
      CALL AUCOUNTER(NCEL,DIRECN,NPULSE,IN,
     2SUM,
     1NST,
     1MCEL)
      GO TO 17
    2 GO TO 17
   18 RETURN
      END
```

10. SUBROUTINE AUCOUNTER

This segment of the program simulates the action of the adaptive binary counter ABUC shown in Figure 5.7 and described in Chapter V Section 6.11(1).

In order to execute this subprogram the following statements must be included in the master segment:

CALL AUCOUNTER (NCEL, DIRECN, NPULSE, IN, SUM, SNT, TEMSUM, MCEL) where DIRECN, SUM and MCEL are respectively the outputs of "SUBROUTINE DIRECTION", "SUBROUTINE ADDER" and "SUBROUTINE HISTORY" and represent the digital binary words Z, H_{Di} and S_h respectively (see Figure 5.7). NPULSE and IN have the meanings given in Section 8 of this appendix and TEMSUM is as in Section 9 above. NST is a parameter which has not been used and NCEL is an integer array holding S_c (see Figure 5.7), which is the output of "SUBROUTINE AUCOUNTER".

```
C
C
C
C
      SUBROUTINE AUCOUNTER(NCEL,DIRECN,NPULSE,
     1IN,
     2SUM,
     1NST,
     1TEMSUM,
     1MCEL)
      INTEGER SUM
      INTEGER TEMSUM
      INTEGER DIRECN
      DIMENSION TEMSUM(15)
      DIMENSION DIRECN(NPULSE)
      DIMENSION NCEL(25),MCEL(25),SUM(25)
C
      I=IN
      L=1
      LB=0
   12 IF(NST.EQ.2) GO TO 2
      IF(NCEL(1).EQ.L)GO TO 1
      NCEL(1)=L
      GO TO 20
    1 NCEL(1)=LB
    2 IF(NCEL(2).EQ.L) GO TO 15
      NCEL(2)=L
      GO TO 20
   15 NCEL(2)=LB
      IF(NCEL(3).EQ.L) GO TO 3
      NCEL(3)=L
      GO TO 20
    3 NCEL(3)=LB
      IF(NCEL(4).EQ.L)GO TO 4
      NCEL(4)=L
      GO TO 20
    4 NCEL(4)=LB
```

```
      IF(NCEL(5).EQ.L)GO TO 5
      NCEL(5)=L
      GO TO 20
    5 NCEL(5)=LB
      IF(NCEL(6).EQ.L)GO TO 6
      NCEL(6)=L
      GO TO 20
    6 NCEL(6)=LB
      IF(NCEL(7).EQ.L)GO TO 7
      NCEL(7)=L
      GO TO 20
    7 NCEL(7)=LB
      IF(NCEL(8).EQ.L)GO TO 8
      NCEL(8)=L
      GO TO 20
    8 NCEL(8)=LB
      IF(NCEL(9).EQ.L)GO TO 9
      NCEL(9)=L
      GO TO 20
    9 NCEL(9)=LB
      IF(NCEL(10).EQ.L)GO TO 10
      NCEL(10)=L
      GO TO 20
   10 NCEL(10)=LB
      IF(NCEL(11).EQ.L)GO TO 11
      NCEL(11)=L
      GO TO 20
   11 NCEL(11)=LB
      IF(NCEL(12).EQ.L)GO TO 13
      NCEL(12)=L
      GO TO 20
   13 WRITE(2,14)
   14 FORMAT(5X,'OVER FLOW IN AUX COUNTER HAS BEEB DETECTED')
      GO TO 20
   20 RETURN
      END
```

11. SUBROUTINE ADDER

This subprogram simulates in FORTRAN the action of a binary full adder. It accepts the two binary code words MCEL (generated by "HISTORY") and NCEL (generated by "AUCOUNTER") and produces the sum and the carry digit of these two words stored in the integer array SUM (see Sections 8, 9 and 10 above and Figure 5.7 in the text of Chapter V).

This segment is executed when the following statement is used in the master segment:

CALL ADDER (NCEL, MCEL, SUM, NPULSE, TEMSUM, NBIT)

where

NBIT = The number of digits in the code word stored in the array NCEL)

The rest of the arguments are defined in Sections 8, 9 and 10 of this appendix.

```
C
C
C
C
      SUBROUTINE ADDER(NNCEL,MMCEL,SUM,NPULSE,
     1TEMSUM,
     1NBIT)
      INTEGER TEMSUM
      INTEGER CAR,SUM
      DIMENSION TEMSUM(15)
      DIMENSION NNCEL(25),MMCEL(25),SUM(25)
C CLEAR THE BINARY ADDER
      DO10IC=1,25
      SUM(IC)=0
   10 CONTINUE
      CAR=0
      L=1
      LB=0
      DO1I=1,NBIT
      IF(NNCEL(I).EQ.LB.AND.MMCEL(I).EQ.LB)GO TO 2
      IF(NNCEL(I).EQ.LB.AND.MMCEL(I).EQ.L.OR.
     1NNCEL(I).EQ.L.AND.MMCEL(I).EQ.LB)GO TO 3
      IF(MMCEL(I).EQ.L.AND.NNCEL(I).EQ.L)GO TO 4
    2 IF(CAR.EQ.L)GO TO 6
      SUM(I)=LB
      CAR=LB
      GO TO 1
    6 SUM(I)=L
      CAR=LB
      GO TO 1
    3 IF(CAR.EQ.L)GO TO 7
      SUM(I)=L
      CAR=LB
      GO TO 1
    7 SUM(I)=LB
      CAR=L
      GO TO 1
    4 IF(CAR.EQ.L)GO TO 8
      SUM(I)=LB
```

```
      CAR=L
      GO TO 1
    8 SUM(I)=L
      CAR=L
    1 CONTINUE
      DO9I=1,25
      MMCEL(I)=SUM(I)
    9 CONTINUE
C
      DO5IC=1,15
CRESET (CLEAR) THE ADAPTIVE UP COUNTER
      NNCEL(IC)=0
    5 CONTINUE
      RETURN
      END
```

12. SUBROUTINE DAC

This subprogram simulates the action of a 12-bit digital-to-analogue converter (see Chapter V, Figure 5.8). The input to the subprogram is a 12-bit binary word stored in the integer array ICEL and the output is ANALOG which is the decimal equivalent of the input digital word.

The FORTRAN statement which causes this segment to be executed by the computer is:

CALL DAC (ICEL, ZULU, JBIT, NBIT, ANALOG)

where

ZULU = An integer constant = 2 and is set in the master segment

NBIT = An integer constant representing the number of digits constituting the digital word stored in ICEL

JBIT = NBIT + 3, a parameter which has not been used

Note that in Chapters V, VI and VII ICEL and ANALOG correspond respectively to H_{Di} and $\hat{H}_i$.

```
C
C
C
C
      SUBROUTINE DAC(ICEL,
     1ZULU,
     1JBIT,
     1NBIT,ANALOG)
      INTEGER ZULU
      DIMENSION ICEL(25)
      ANALOG=0.0
      DO1JB=1,JBIT
C
      NUMU=JB-1
      ANALOG=ANALOG+((ZULU)**NUMU)*ICEL(JB)
    1 CONTINUE
      RETURN
      END
```

13. SUBROUTINE INTERPOLATION

The function of this segment is to simulate in FORTRAN the action of the electronic circuit which accepts the sum $\hat{y}$ of 8 consecutive binary code words $H_{D(i+k)}$ (k = 0,1,...7) and produces the average X_L of these words (see Chapter VI). It simulates the operation of binary division of the code word $\hat{y}$ by 8.

The subprogram is executed by the computer when the following statement is encountered:

CALL INTERPOLATION (STOR, DSTOR, NPULSE, PRINSTOR, LSTOR, NBIT)

where NPULSE has the same meaning as in Section 8 of this appendix and NBIT is the number of digits in the integer arrays STOR, LSTOR, PRINSTOR, LSTOR and DSTOR.

STOR = An integer array holding the digital word to be divided. It is an input to this subprogram

DSTOR = An integer array holding the result of the division; i.e. the output of the subprogram

PRINSTOR and LSTOR are integer arrays used as working spaces.

```
C
C
C
C
      SUBROUTINE INTERPOLATION(STOR,DSTOR,NPULSE,
     1PRINSTOR,
     1LSTORE,
     1NBIT)
      INTEGER DSTOR
      INTEGER STOR
      INTEGER PRINSTOR
      DIMENSION STOR(15),DSTOR(15)
      DIMENSION PRINSTOR(20)
      DIMENSION LSTORE(20)
C CLEAR THE INTERPOLATOR
      DO2IE=1,15
      DSTOR(IE)=0
    2 CONTINUE
      NSHIFT=3
      JBIT=NBIT+NSHIFT
      DO1I=1,20
      PRINSTOR(I)=0
    1 CONTINUE
      DO5I=1,JBIT
      PRINSTOR(I)=STOR(I)
    5 CONTINUE
      DO4J=1,NSHIFT
      DO3I=1,JBIT
      PRINSTOR(I)=PRINSTOR(I+1)
      LSTORE(J)=J
    3 CONTINUE
    4 CONTINUE
      DO6KJ=1,NBIT
      DSTOR(KJ)=PRINSTOR(KJ)
    6 CONTINUE
      RETURN
      END
```

14. SUBROUTINE ACCUMULATOR

This segment simulates in FORTRAN the action of a digital accumulator whose contents are reset to zeros at 8 clock period intervals. This, together with "SUBROUTINE INTERPOLATION", simulates the action of the digital interpolator described in Chapter VI.

The subprogram "ACCUMULATOR" accepts the 8 digital code words representing H_{Dk} (k = 1,8) stored in the integer array SUM and produces $\hat{y}$ the sum of these words at 8 clock period intervals. The output of the segment is stored in the integer array STOR.

To execute the subprogram the following statement must be included in the master segment:

CALL ACCUMULATOR (SUM, STOR, NBIT, TEMSUM)

where NBIT is the number of digits in the input code words' storage array SUM and TEMSUM is an integer array used as working space.

```
C
C
C
C
      SUBROUTINE ACCUMULATOR(SUM,STOR,NBIT,
     1TEMSTOR)
      INTEGER STOR,SUM,CARY
      DIMENSION TEMSTOR(15)
      DIMENSION SUM(25),STOR(25)
      NSHIFT=3
      JBIT=NBIT+NSHIFT
      KBIT=JBIT+1
      L=1
      LB=0
      DO1 I=1,JBIT
      IF(SUM(I).EQ.LB.AND.STOR(I).EQ.LB)GO TO 2
      IF(SUM(I).EQ.LB.AND.STOR(I).EQ.L.OR.
     1SUM(I).EQ.L.AND.STOR(I).EQ.LB)GO TO 3
      IF(SUM(I).EQ.L.AND.STOR(I).EQ.L)GO TO 4
    2 IF(CARY.EQ.L)GO TO 5
      TEMSTOR(I)=LB
      CARY=LB
      GO TO 1
    5 TEMSTOR(I)=L
      CARY=LB
      GO TO 1
    3 IF(CARY.EQ.L)GO TO 6
      TEMSTOR(I)=L
      CARY=LB
      GO TO 1
    6 TEMSTOR(I)=LB
      CARY=L
      GO TO 1
    4 IF(CARY.EQ.L)GO TO 7
      TEMSTOR(I)=LB
      CARY=L
      GO TO 1
```

```
 7 TEMSTOR(I)=L
   CARY=L
 1 CONTINUE
   DO8I=1,JBIT
   STOR(I)=TEMSTOR(I)
   TEMSTOR(I)=0
 8 CONTINUE
   INDICATOR=0
   DO9I=1,JBIT
   INDICATOR=INDICATOR+STOR(I)
 9 CONTINUE
   IF(INDICATOR.EQ.NBIT)GO TO 10
   IF(INDICATOR.EQ.JBIT)GO TO 10
   GO TO 12
10 WRITE(2,11)
11 FORMAT(10X,'ABSOLUTE OVERLAOD HAS BEEN DETECTED')
12 RETURN
   END
```

15. SUBROUTINE FILTUN

The function of this subprogram is to evaluate the transfer function of the digital interpolator described in Chapter VI and it subsequently plots the system response in the frequency domain (see the theory leading to the digital interpolator frequency response developed in Chapter VI).

To execute this segment the following statement is inserted in the master segment after the sampling frequency f_p and the averaging period N_r are stored in the addresses FPP and NARE respectively.

CALL FILFUN (FSS, FUN1, FUN2, FIFSS, FPP, NARE)

where FUN1 and FUN2 are internal variables and:

FIFSS = A real array holding the output of the segment representing the samples of the system frequency response.

FSS = A real array holding the frequency points corresponding to the response samples stored in the array FIFSS.

UTP4C and UTP4B are computer library plotting routines.

Note that FIFSS corresponds to $|\phi(f)|$ (see Chapter VI).

```
C
C
C
C
      SUBROUTINE FILFUN(FSS,
     1FUN1,FUN2,FIFSS,FPP,NARE)
      DIMENSION FSS(100),FIFSS(100),FUN1(100),FUN2(100)
      PI=3.141597654
      DO2NEF=1,80
      FSS(NEF)=NEF*0.800
      FUN1(NEF)=SIN(PI*NARE*FSS(NEF)/FPP)
      FUN2(NEF)=SIN(PI*FSS(NEF)/FPP)
      FUN2(NEF)=NARE*FUN2(NEF)
      FIFSS(NEF)=FUN1(NEF)/FUN2(NEF)
      WRITE(2,3)NARE,NEF,FSS(NEF),FUN1(NEF),FUN2(NEF),FIFSS(NEF)
    2 FIFSS(NEF)=ABS(FIFSS(NEF))
      WRITE(2,3)NARE,NEF,FSS(NEF),FUN1(NEF),FUN2(NEF),FIFSS(NEF)
    3 FORMAT(5X,2I10,4F15.3)
      GO TO 5
      CALL UTP4C(0.0,70.0,0.0,1.25,7.0,5.0,
     1'FREQUENCY IN KHZ',2,
     1'FILTER RESPONSE',2)
      CALL UTP4B(FSS,FIFSS,80,0)
    5 RETURN
      END
```

16. SUBROUTINE COMPRESSION

This segment simulates in FORTRAN the action of the linear PCM-to-A-law PCM converter (compressor) described in Chapter VI. It accepts the linear PCM simulated code words X_L stored in the integer array DSTOR and produces A-law PCM code words X_C stored in the integer array FINSTOR. The number of digits constituting the linear PCM code word is set in the master segment and stored in the address NBIT.

When this segment is to be executed the following statement must be inserted in the master segment:

CALL COMPRESSION (DSTOR, NPULSE, NBIT, LCEL, PRELSTOR, FINSTOR)

where NPULSE is a constant which can be set in the master segment if needed (not used here) and LCEL and PRELSTOR are real arrays used as working spaces.

```
C
C
C
C
C
      SUBROUTINE COMPRESSION(DSTOR,NPULSE,NBIT,LCEL,PRELSTOR,
     1FINSTOR)
      INTEGER FINSTOR,PRELSTOR
      INTEGER DSTOR
      DIMENSION DSTOR(15)
      DIMENSION PRELSTOR(20)
      DIMENSION LCEL(25),FINSTOR(25)
      MD=0
      L=1
      LB=0
      DO1I=1,15
      FINSTOR(I)=0
      LCEL(I)=0
    1 CONTINUE
      DO3I=1,20
      PRELSTOR(I)=0
    3 CONTINUE
      GO TO 2
      DO26MS=1,NBIT
   26 DSTOR(MS)=DSTOR(MS+1)
    2 NOSHFT=1
      DO4I=1,4
      PRELSTOR(I)=DSTOR(I)
    4 CONTINUE
      DO5K=5,11
      I=K+7
      PRELSTOR(I)=DSTOR(K)
    5 CONTINUE
      MONITOR=11
      IF(PRELSTOR(MONITOR).EQ.LB)GO TO 8
      DO9N=1,3
    9 LCEL(N)=L
    8 DO6J=1,7
```

```
      DO7K=5,18
      PRELSTOR(K)=PRELSTOR(K+1)
    7 CONTINUE
      IF(PRELSTOR(MONITOR).EQ.L)GO TO 11
      IF(LCEL(1).EQ.L)GO TO 12
      IF(LCEL(2).EQ.L)GO TO 13
      IF(LCEL(3).EQ.L)GO TO 14
      DO15I=1,3
   15 LCEL(I)=0
      GO TO 6
   14 LCEL(3)=LB
      LCEL(2)=L
      LCEL(1)=L
      GO TO 6
   13 LCEL(2)=LB
      LCEL(1)=L
      GO TO 6
   12 LCEL(1)=LB
      GO TO 6
   11 DO25I=1,3
   25 LCEL(I)=L
      GO TO 6
    6 CONTINUE
      DO10I=5,7
      JW=I-4
   10 FINSTOR(I)=LCEL(JW)
   24 IF(LCEL(2).EQ.L.OR.LCEL(3).EQ.L)GO TO 16
      DO17I=1,4
      FINSTOR(I)=PRELSTOR(I)
   17 CONTINUE
      GO TO 18
   16 J=8+MD
   23 DO22K=1,11
C
   22 PRELSTOR(K)=PRELSTOR(K+1)
      IF(LCEL(1).EQ.1)GO TO 19
      IF(LCEL(2).EQ.L)GO TO 20
      IF(LCEL(3).EQ.L)GO TO 21
```

```
19 LCEL(1)=LB
   MD=1
   GO TO 24
20 LCEL(2)=LB
   LCEL(1)=L
   MD=2
   GO TO 24
21 LCEL(3)=LB
   LCEL(2)=L
   LCEL(1)=L
   GO TO 24
18 RETURN
   END
```

17. SUBROUTINE OPERATION

This subprogram interpolates between each 7 consecutive SCALE feedback signal values to generate one PAM sample. The interpolation strategy was described in Chapter VI and is basically as follows:

The subprogram accepts the SCALE feedback signal samples H_i - stored in the real array HDELIN - at a rate f_p = 56 Kb/s. It then inspects these samples and selects the $7n^{th}$ (n=1,2,3,4.....) sample to represent all the previous 6 and the present sample. The output of the subprogram - stored in the real array SAMPLH - therefore consists of constant steps occupying a time interval of 7 clock periods and having a magnitude which is the same as that of the appropriate $7n^{th}$ SCALE feedback signal sample.

The "CALL" statement for this segment is:

```
CALL OPERATION (CLP, HDELIN, NPULSE, SAMPLH, LSAMPL, IIJ, CLPP,
                SAMPL)
```

where NPULSE has the same meaning as in Section 8 of this appendix and IIJ, CLP, CLPP, SAMPL and LSAMPL are internal parameters defined within the subprogram and HEDLIN and SAMPLH are the input and output arrays.

```
C
C
C
C
      SUBROUTINE OPERATION(CLP,HDELIN,NPULSE,SAMPLH,LSAMPL,IIJ,
     1CLPP,SAMPLE)
      DIMENSION CLP(NPULSE),HDELIN(NPULSE),SAMPLH(NPULSE)
      DIMENSION CLPP(100),SAMPLE(100)
      CLPP(1)=CLP(1)
      SAMPLE(1)=HDELIN(1)
      KLEVEL=7
      DO16I=KLEVEL,NPULSE-1,KLEVEL
      J=I/KLEVEL
      K=J+1
      SAMPLE(K)=HDELIN(I)
      CLPP(K)=CLP(I)
   16 CONTINUE
      MLEVEL=7
      IM=1
      K=1
   13 DO12IL=IM,MLEVEL
      SAMPLH(IL)=SAMPLE(K)
   12 CONTINUE
      IF(K.LE.LSAMPL)GO TO 11
      GO TO 10
   11 K=K+1
      MLEVEL=MLEVEL+KLEVEL
      IM=IM+KLEVEL
      GO TO 13
   10 ABC=8.0
      DO8I=1,NPULSE
      SAMPLH(I)=SAMPLH(I)+HDELIN(1)/2.0
      GO TO 8
    8 CONTINUE
      WRITE(2,9)(CLP(I),SAMPLH(I),I=1,NPULSE)
    9 FORMAT(10X,E14.7,10X,E14.7)
  909 ABC=0.0
      RETURN
      END
```

18. SUBROUTINE ESTIMATE

This subprogram can perform the function of "SUBROUTINE ACCUMULATOR", "SUBROUTINE INTERPOLATION" and "SUBROUTINE DAC". It accepts the SCALE PAM samples $\hat{H}_{Di}$ stored in the real array "HDELIN" and produces LPCM PAM samples held in the real array "SAMPLH". The processing is performed in the computer using decimal arithmetic.

The subprogram was written for converting SCALE PAM samples to LPCM levels when the line rates of both systems is 56 Kb/s. However, if the line rate is to be changed to 64 Kb/s instead of 56 Kb/s, both of the integer variables "KLEVEL" and "MLEVEL", and the divisor in the third statement after statement Number 2, should be equated to 8 instead of 7.

The "CALL" statement which enables the subprogram to be executed is:

CALL ESTIMATE (GROUPE, HDELIN, NPULSE, SAMPLH)

where "GROUPE" is a real array used as a working space and "NPULSE" is the number of SCALE samples to be processed.

```
      SUBROUTINE ESTIMATE(GROUPE,HDELIN,NPULSE,SAMPLH)
      DIMENSION HDELIN(NPULSE),GROUPE(NPULSE)
      DIMENSION CLPP(100)
      DIMENSION SAMPLH(NPULSE)
      IK=1
      KLEVEL=7
      LPULSE=(NPULSE-1)/KLEVEL+1
      LPULSE=LPULSE-1
      LSAMPL=LPULSE
      DO1 I=KLEVEL,NPULSE-1,KLEVEL
      J=I/KLEVEL
      CLPP(J)=J
      DO2IN=IK,J*KLEVEL
      GROUP=GROUP+HDELIN(IN)
    2 CONTINUE
      GROUPE(J)=GROUP
      GROUP=0.0
      GROUPE(J)=GROUPE(J)/7.0
      IK=IK+KLEVEL
    1 CONTINUE
      GO TO 6
    6 AB 0.0
      K=1
      IM=1
      MLEVEL=7
   13 DO12IL=IM,MLEVEL
      SAMPLH(IL)=GROUPE(K)
   12 CONTINUE
      IF(K.LT.LSAMPL)GO TO 11
      GO TO 10
   11 K=K+1
      MLEVEL=MLEVEL+KLEVEL
      IM=IM+KLEVEL
      GO TO 13
   10 ABC=8.0
      GO TO 7
      WRITE(2,3)(CLPP(J),GROUPE(J),J=1,LPULSE)
```

```
3 FORMAT(5X,F20.5,10X,F20.5)
  WRITE(2,4)IN,IK,J
4 FORMAT(5X,3I20)
  WRITE(2,5)IK
5 FORMAT(5X,I20)
7 DONTWRIT=1.0
  RETURN
  END
```

19. SUBROUTINE DECODERD

This segment can perform the function of "SUBROUTINE DIRECTION", "SUBROUTINE HISTORY", "SUBROUTINE AUCOUNTER", "SUBROUTINE ADDER" and "SUBROUTINE DAC". It simulates the complete Digital SCALE decoder using decimal arithmetic. The subprogram accepts the SCALE binary data samples stored in the integer array "KAA" and produces DSCALE discrete PAM samples ($\hat{H}_i$) held in the real array "HDELIN".

The "CALL" statement which enables the segment to be executed is:

CALL DECODERD (HDELIN, KAA, NPULSE, HALSTP, IIJ)

where NPULSE is the total number of SCALE samples involved in the processing and "HALSTP" is the address where the maximum step increment $\Delta(J,Z)$ (see Chapter V) is stored.

IIJ is a switching parameter which enables the value of the feedback step height $\hat{H}_i$ to be initialized at the beginning of the decoding operation.

```
C
      SUBROUTINE DECODERD(HDELIN,KAA,NPULSE,HALSTP,IIJ)
      DIMENSION HDELIN(NPULSE),KAA(NPULSE)
      DATA KA,KB,KC,Z/0,0,0,0.0/
      HDELIN(1)=0.0
      HDELIN(1)=HALSTP/2.0
      HDELIN(1)=0.025
      IF(IIJ.EQ.2)HDELIN(1)=HVALUE
      FACTOR=1000.0/9.0
      FACTOR=1.0/HALSTP
      FACTOR=1.0
      HATSTP=HALSTP*FACTOR
      SEGT1=HALSTP*FACTOR*128
      SEGT2=2.0*SEGT1
      SEGT3=2.0*SEGT2
      SEGT3F=SEGT3-SEGT1
      ZH=3.8
      DO21I=2,NPULSE
      KC=KB
      KB=KA
      IF(KA.EQ.KB.AND.KB.EQ.KC)GO TO 24
      STEP=HATSTP
      IF(ABS(HDELIN(I-1)).LT.SEGT3F)STEP=HATSTP*3.0/4.0
      IF(ABS(HDELIN(I-1)).LT.SEGT2)STEP=HATSTP*3.0/8.0
      IF(ABS(HDELIN(I-1)).LT.SEGT1)STEP=HATSTP/8.0
      HDELIN(I)=ABS(HDELIN(I-1))-STEP
      GO TO 23
   24 Z=ZH
      STEP=HATSTP
      BCONST=3.0*HATSTP
      IF(ABS(HDELIN(I-1)).GT.SEGT1)STEP=3.0*HATSTP/4.0
      IF(ABS(HDELIN(I-1)).GT.SEGT2)STEP=HATSTP/2.0
      HDELIN(I)=ABS(HDELIN(I-1))+STEP
   23 HDELIN(I)=HDELIN(I)*(2*KA-1)
      KA=KAA(I)
   21 CONTINUE
      HVALUE=HDELIN(NPULSE)
      RETURN
      END
```

20. SUBROUTINE EXPANDER

The function of this segment is to accept the A-law PCM compressed code words X_c stored in the integer array "FINSTOR" and produce the corresponding LPCM code words stored in the integer array "LINEAR". Also it produces the analogue equivalent of X_L stored in the address "ANALOG".

The subprogram is executed by the computer when the following statement is encountered in the master segment:

CALL EXPANDER (FINSTOR, LCEL, LINEAR, NBIT, JBIT, MD, ZULU, ANALOG)

where "LCEL" and "MD" are internal variables, and "NBIT", "JBIT" and "ZULU" are the arguments of "SUBROUTINE DAC" which is "called" from this segment.

"SUBROUTINE DAC" must be loaded into the computer working file before "EXPANDER" is "called".

```
      SUBROUTINE EXPANDER(FINSTOR,LCEL,LINEAR,
     1NBIT,JBIT,MD,ZULU,ANALOG)
      INTEGER ZULU
      DIMENSION LINEAR(25),FINSTOR(25),LCEL(25)
      LB=0
      L=1
      LCEL(1)=L
      DO1I=1,25
      LCEL(I)=LB
      LINEAR(I)=LB
    1 CONTINUE
      LINEAR(1)=L
      DO2I=1,4
      J=I+1
      LINEAR(J)=FINSTOR(I)
    2 CONTINUE
      DO3I=1,3
      II=I+4
      LCEL(I)=FINSTOR(II)
    3 CONTINUE
   11 IF(LCEL(2).EQ.L.OR.LCEL(3).EQ.L)GO TO 4
      GO TO 8
    4 DO5K=1,NBIT
    5 LINEAR(K+1)=LINEAR(K)
      IF(LCEL(1).EQ.L)GO TO 6
      IF(LCEL(2).EQ.L)GO TO 7
      IF(LCEL(3).EQ.L)GO TO 10
    6 LCEL(1)=LB
      GO TO 11
    7 LCEL(2)=LB
      LCEL(1)=L
      GO TO 11
   10 LCEL(3)=LB
      LCEL(2)=L
      LCEL(1)=L
     GO TO 11
     MD=1
```

```
 8 GO TO(12,13),MD
12 CALL DAC(LINEAR,
  1ZULU,JBIT,NBIT,ANALOG)
   WRITE(2,14)ANALOG
14 FORMAT(10X,E14.7)
13 RETURN
   END
```

21. SUBROUTINE ATODCONV

This segment was written to simulate the action of the Linear PCM encoder (Analogue-to-Digital converter) using the successive approximation technique. It accepts the magnitude and polarity of the analogue signal stored in the addresses "SIGN" and "SIGSAMPL", respectively, at Nyquist rate, and produces 12-bit LPCM words at the same rate. The LPCM words are "returned" to the master segment held in the integer array "DIGITAL".

The subprogram is executed by the computer when the following statement is encountered in the master segment:

```
CALL ATODCONV (DIGITAL, SIGVMAX, NSMAX, JJJ, SIGN, TEMSUM,
               SIGSAMPL)
```

where "SIGVMAX" and NSMAX" are respectively the addresses where the maximum expected analogue signal magnitude and number of digits in the LPCM (less the sign digit) are stored. "TEMSUM" is an integer array used as a working space and JJJ is an integer variable representing the number of the LPCM PAM sample.

```
      SUBROUTINE ATODCONV(DIGITAL,SIGVMAX,NSMAX,JJJ,
     4SIGN,
     3TEMSUM,
     1SIGSAMPL)
      INTEGER TEMSUM
      INTEGER DIGITAL
      INTEGER SIGN
      DIMENSION TEMSUM(15)
      DIMENSION STEP(25)
      DIMENSION DIGITAL(25)
      DIMENSION SIGSAMPL(512)
      Q=0.0
      L=1
      LB=0
      NSMAX=11
      MSMAX=NSMAX+1
      STEP(1)=SIGVMAX
      IN=JJJ
      SIGN=1
      IF(SIGSAMPL(IN).LT.0.0)SIGN=-1
      SIGSAMPL(IN)=ABS(SIGSAMPL(IN))
      DO1I=2,MSMAX
      J=I-1
      IJK=MSMAX-J
      STEP(I)=STEP(J)/2.0
C
      IF(J.LT.2)GO TO 4
      LM=J-1
      LL=DIGITAL(LM)+1
      GO TO(3,4),LL
    4 Q=Q+STEP(I)
      GO TO 5
    3 Q=Q-STEP(I)
```

```
    5 DIGITAL(J)=LB
C
      IF(SIGSAMPL(IN).GE.Q)DIGITAL(J)=L
      TEMSUM(IJK)=DIGITAL(J)
      GO TO 1
      WRITE(2,6)I,J,LM,LL,DIGITAL(J),STEP(I),STEP(J),Q,SIGSAMPL(IN)
    6 FORMAT(10X,5I10,4F10.3)
    1 CONTINUE
C
      DO7IQ=1,NSMAX
    7 DIGITAL(IQ)=TEMSUM(I)
      GO TO 8
      WRITE(2,2)(DIGITAL(K),K=1,NSMAX)
    2 FORMAT(10X,11I8)
    8 RETURN
      END
```

22. SUBROUTINE COMPARITOR

This segment was written to simulate the action of the forward path (subtractor, comparator and time quantizer) of the LPCM to SCALE converter. It accepts as inputs the LPCM digital words (X_L) stored in the integer array "DIGITAL" and the locally generated DSCALE digital words (H_{Di}) stored in the integer array "SUM". Its output stored in the integer array "KAA" is the SCALE binary data stream ($\hat{L}(t)$) representing X_L in the SCALE format.

The "CALL" statement which enables the subprogram to be executed is:

```
CALL COMPARITOR (SUM, DIGITAL, ZULU, JBIT, NBIT, ICEL, SIGN,
                 ANALOG, VALUE, T1, ERROR1, POL, ERROR, IN, KAA)
```

where "SIGN" represents the polarity digit of the LPCM word and "POL" represents the polarity of "ERROR1", the accumulated value of "ERROR". "ERROR" represents in digital form of 12-bit words the instantaneous difference between X_L and H_{Di}. "IN" is the number of the SCALE clock period starting from zero time, and "VALUE" and "ICEL" are internal parameters.

"T1" is a constant representing the time constant of the SCALE decoder which is going to decode the binary data generated by the LPCM-to-SCALE converter.

This subprogram "calls" "SUBROUTINE DAC" which, therefore, must be loaded into the computer working file before this segment is "called". The meaning of the arguments "ZULU", "JBIT", "NBIT" and "ANALOG" have already been explained (see "SUBROUTINE DAC").

```
      SUBROUTINE COMPARITOR(SUM,DIGITAL,
     2ZULU,JBIT,NBIT,
     3ICEL,SIGN,
     4ANALOG,VALUE,
     5T1,ERROR1,POL,
     1ERROR,IN,KAA)
      INTEGER DIGITAL
      INTEGER SIGN
      INTEGER ZULU
      INTEGER SUM
      DIMENSION SUM(25)
      DIMENSION ICEL(25)
      DIMENSION KAA(512)
      DIMENSION DIGITAL(25)
      IIN=IN+1
      FP=64000.0
      IDEC=2
      TP=1.0/FP
      ALPHAE=TP/T1
      AE1=1.0/EXP(ALPHAE)
      CALL DAC(DIGITAL,ZULU,JBIT,NBIT,ANALOG)
      VALUE=ANALOG
      CALL DAC(SUM,ZULU,JBIT,NBIT,ANALOG)
      VALUE=VALUE*SIGN/1000.0
      ANALOG=ANALOG*POL/1000.0
      ERROR=VALUE-ANALOG
      ERROR1=AE1*(ERROR1-ERROR)+ERROR
CDETERMINE THE POLARITY OF L(T),THE DIGIT TO BE SENT TO THE RECEIVER
      KAA(IIN)=0
      GO TO(2,3),IDEC
    2 IF(SIGN.EQ.1.AND.ERROR.GE.0.0.OR.KAA(IN).EQ.
     10.AND.ERROR.LT.0.0)KAA(IIN)=1
```

```
    3 IF(ERROR1.GE.0.0)KAA(IIN)=1
      IPOL=KAA(IIN)+1
      GO TO(4,5),IPOL
    4 POL=-1.0
      GO TO 6
    5 POL=1.0
C
    6 WRITE(2,1)VALUE,ANALOG,ERROR,SIGN,IN,KAA(IN),IIN,KAA(IIN)
    1 FORMAT(10X,3F10.3,5I10)
      RETURN
      END
```

23. SUBROUTINE SUBTRACTOR

This subprogram accepts two 12-bit binary words stored in the integer arrays "DIGITAL" and "SUM" and produces the difference between them (i.e. it subtracts the contents of "SUM" from the contents of "DIGITAL"). The difference is "returned" to the master segment held in the integer array "DIFFRNCE". The subtraction operation is performed using binary arithmetic and the "two's complement" technique.

The segment is executed when the following statement is encountered in the master segment:

```
CALL SUBTRACTOR (DIGITAL, SUM, DIFFRNCE, NPULSE, NBIT, TEMSUM,
                 ADDONE, NEG)
```

where "NPULSE" and "NBIT" are integer variables representing respectively, the total number of subtraction operations involved and number of digits in the digital words to be operated upon (NBIT = 12 for example).

The parameter "NEG" and "ADDONE" are defined in the subprogram and "TEMSUM" is an integer array used as working space.

This subprogram "calls" "SUBROUTINE ADDER" to perform the addition operations involved in the "two's complement" operations. For this reason "SUBROUTINE ADDER" should be loaded into the computer working file before "SUBROUTINE SUBTRACTOR" is "called".

```
      SUBROUTINE SUBTRACTOR(DIGITAL,SUM,DIFFRNCE,NPULSE,NBIT,
     1TEMSUM,
     1ADDONE,NEG)
      INTEGER DIGITAL,SUM
      INTEGER DIFFRNCE,ADDONE
      INTEGER TEMSUM
      DIMENSION SUM(25),DIGITAL(25)
      DIMENSION DIFFRNCE(25),ADDONE(25)
      DIMENSION TEMSUM(15)
      NEG=1
      NPULSE=512
      NBIT=12
      L=1
      LB=0
      NTOT=NBIT+1
      DO1I=1,NBIT
      DIFFRNCE(I)=L
      IF(SUM(I).EQ.L)DIFFRNCE(I)=LB
      ADDONE(I)=LB
    1 CONTINUE
      ADDONE(1)=L
      CALL ADDER(ADDONE,DIFFRNCE,SUM,NPULSE,
     1TEMSUM,
     1NBIT)
      CALL ADDER(DIGITAL,DIFFRNCE,SUM,NPULSE,
     1TEMSUM,
     1NBIT)
      IF(DIFFRNCE(NBIT).EQ.LB)GO TO 3
      ADDONE(1)=L
      NEG=-1
      DO2I=1,NBIT
      DIFFRNCE(I)=L
      IF(SUM(I).EQ.L)DIFFRNCE(I)=LB
```

```
2 CONTINUE
  CALL ADDER(ADDONE,DIFFRNCE,SUM,NPULSE,
 1TEMSUM,
 1NBIT)
3 AVO=1.0
  RETURN
  END
```

24. SUBROUTINE ALAWRX

This subprogram was written to simulate the action of an A-law PCM receiver. The simulation strategy was based on the theory of the segmented A-law PCM discussed in Chapter VI. It accepts the A-law PCM code words X_c stored in the integer array "FINSTOR" and produces the corresponding analogue PAM samples stored in the address labled "WYOUT".

The subprogram is executed by the computer when the following statement is encountered in the master segment:

CALL ALAWRX (WYOUT, FINSTOR)

where the arguments "WYOUT" and "FINSTOR" have the meanings described above.

```
C
C
      SUBROUTINE ALAWRX(WYOUT,
     1FINSTOR)
      INTEGER FINSTOR
      DIMENSION FINSTOR(25)
      LSUM=0
      DO1I=5,7
      LSUM=LSUM+(2**(I-5))*FINSTOR(I)
    1 CONTINUE
      LETA=1
      IF(LSUM.EQ.0)LETA=0
      IVSTEP=0
      DO2I=1,4
      IVSTEP=IVSTEP+(2**(I-1))*FINSTOR(I)
    2 CONTINUE
C
      ICORECT=1
      MAINSTEP=16
      ICORECT=0
      ICORECT=-1
      CORECT=ICORECT
      CORECT=0.5
      MAINSTEP=MAINSTEP*LETA
      LFACTOR1=2**(LSUM-LETA)
      XFACTOR2=(IVSTEP+MAINSTEP+CORECT)
      WYOUT=LFACTOR1*XFACTOR2
      WRITE(2,3)WYOUT
    3 FORMAT(10X,E14.7)
      GO TO 4
    4 RETURN
      END
```

APPENDIX C

Theory of Finite Impulse Response Filters[72,95,97]

The Finite Impulse Response Filter, known as "FIR Filter" is a non-recursive type. This means that in a FIR digital filter the output sequence depends only on the input sequence and is free from feedback effects. This type of filter has linear phase characteristic and is suitable for filtering of speech signals corrupted by noise.

A block diagram of the FIR digital filter is shown in Fig. (C1). It employs N digital delay elements (e.g. D-type flipflops) which hold the successive digits. The digital sequence is shifted by one position (towards the output port) every clock period (T_p seconds, say) and each digit in the sequence is then weighted by the appropriate coefficient h_k. The relationship between the input sequence $x(iT_p)$ and the output sequence $y(iT_p)$ is given by

$$y(iT_p) = \sum_{k=0}^{N-1} h_k x\{(i - k)T_p\} \qquad (1C)$$

Extra noise can be introduced by the filter into the signal which is being filtered. This is mainly due to the representation of h_k by a short binary word. This type of noise is known as rounding off noise and can be reduced by employing a longer binary word to represent h_k.

The impulse response consists of the N weighting coefficient sequence $(h_o \ldots . h_{N-1})$. Because it is always finite the impulse response is limited in time. Hence the name "Finite Impulse Response". The recursive type filters do not have this property.

The frequency response of the filter can be determined from a knowledge of the impulse response sequence.

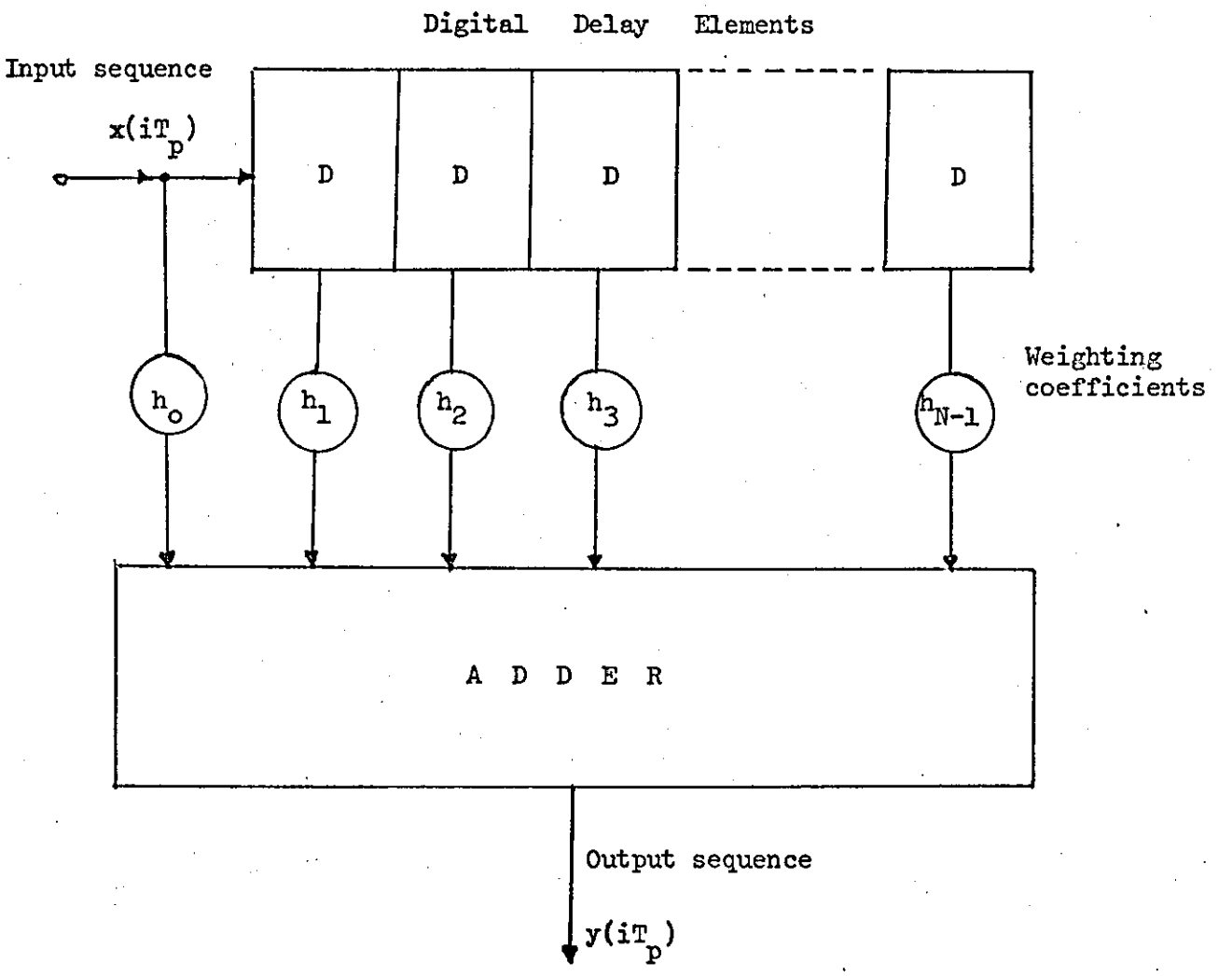


Figure (C1) Block diagram of Finite Impulse Response FIR digital filter

e.g. let the input sequence be a digitized sinusoid with angular frequency ω rads/sec, then substituting in equation (1C)

$$y_\omega(iT_p) = \sum_{k=0}^{N-1} h_k \cos[(i-k)\omega T_p]$$

$$= R_e\left[e^{j\omega iT_p}\right]\sum_{k=0}^{N-1} h_k e^{-j\omega T_p} \qquad (2C)$$

(Note that $2\cos\omega kT_p = e^{j\omega kT_p} + e^{-j\omega kT_p}$

$$= R_e \sum_{k=0}^{N-1} e^{j(i-k)\omega T_p} \quad)$$

where R_e means the real part.

Using De Moivre theorem the output can be rewritten in magnitude and angle representation

$$y_\omega(iT_p) = R_e\left[\sum_{k=0}^{N-1} h_k e^{-j\omega kT_p}\right] e^{j(i\omega T_p+\theta)}$$

$$= \left|\sum_{k=0}^{N-1} h_k e^{-j\omega kT_p}\right| \cos(i\omega T_p + \theta)$$

Thus the response of the filter to a cosine wave with angular frequency ω is a cosine wave of the same frequency. The effect of the filter on the input signal is to introduce amplitude attenuation which is governed by the term

$$\left|\sum_{k=0}^{N-1} h_k e^{-j\omega kT_p}\right|$$

and phase shift θ. The above term is therefore the magnitude of the frequency response of the filter.

Frequency sampling techniques

The frequency sampling technique described by Rabiner et al[72] is a general procedure which is very suitable for digital filtering when employing a digital computer. The procedure is basically as follows:

(1) Given the frequency response of a filter, the response is sampled at intervals of $1/NT_p$ along the frequency axis. ($T_p = 1/f_p$ = the sampling interval.) Let these samples be $(X_o, X_1 \ldots X_{N-1})$ then a sample of the frequency response can be written as

$$X_r = \sum_{k=0}^{N-1} h_k e^{(-jk2\pi r/N)} \qquad (3C)$$

where $n = 0,1 \ldots N-1$.

This equation relates the discrete frequency samples of the filter frequency response to the weighting coefficients of the impulse response. It can also be seen at this stage that equation (3C) expresses the Discrete Fourier Transform DFT of the sequence of the weighting coefficients $(h_o, h_1 \ldots h_{N-1})$. This means that the impulse response can be obtained by taking the Inverse DFT (IDFT) of the frequency samples of the filter frequency response.

(2) Having obtained the samples of the filter frequency response $(X_o, X_1 \ldots X_{N-1})$ the IDFT is computed to obtain the required weighting sequence $(h_o, h_1 \ldots h_{N-1})$ using the relation

$$h_k = 1/N \sum_{r=0}^{N-1} X_r e^{j2\pi kr/N} \qquad (4C)$$

where $k = 0,1 \ldots N-1$.

This can best be achieved using the Fast Fourier Transform FFT technique[73]. This technique reduces the number of multiplication and addition operations (involved in the finding of the IDFT of N samples) from being

proportional to N^2, to a number which is proportional to $N.\log_2 N$. The gain in speed of the calculation due to using the FFT is $N^2/N.\log_2 N = N/\log_2 N$.

(3) With the aid of the impulse response of the filter, filtering can be performed in the time domain.

If a unit input sample, preceded and followed by zeros, is applied to a filter the output of the filter would be $h_0, h_1 \ldots h_{N-1}$; as the input moves along the delay elements, the sequence h_k, $k = 0,1 \ldots N-1$, is the impulse response of the filter. When an input sequence x_r is applied the output of the filter is

$$y_r = \sum_{k=0}^{\infty} h_k x_{r-k} = \sum_{k=-\infty}^{r} h_{r-k} x_k$$

$$= h_r * x_r \qquad (C5)$$

where * designates convolution.

The output sequence can therefore be obtained by performing the discrete convolution of the input sequence with the impulse response sequence.

An alternative approach for obtaining the filtered output is to perform the filtering operation in the frequency domain. In this case the input and impulse response sequences are transformed to the frequency domain by means of the DFT and subsequently multiplied (instead of convolved) to produce the output samples.

APPENDIX D

Theory of Active Filters[69,74,75]

An active filter is cheaper and lighter than a passive filter at low operating frequencies, specifically in the audio range, where an active filter works well. The frequency limitation of an active filter design is the active element (the amplifier used). This latter limitation has now been by the advent of modern integrated circuits and high quality transistors. Most active filters (low pass, high pass, band pass, etc.) operating in the audio range employ resistors and capacitors as passive elements and an amplifier in a positive fixed gain configuration as the active elements.

To fully characterize the type of filter the following information is needed:

(1) Voltage-transfer function H(s)

(2) Circuit configuration

(3) Cut-off or centre frequency ω_o

(4) Damping ratio d.r. or quality factor q.f.

(5) Passive component values

Both ω_o and d.r. (or q.f.) are selected to meet the filter overall requirements and from these the passive components are subsequently calculated.

The stability functions provide a measure of the circuit sensitivity to changes in component values.

The transfer function of this type of active filter can be written

$$H(s) = \frac{N(s)}{D(s)} \tag{D1}$$

$$= \frac{N(s)}{s^2 + 2(\text{d.r.})\omega_o + \omega_o^2} \tag{D2}$$

$\omega_o = 2\pi f_o$ where f_o is the centre frequency of the filter

$s = (\sigma + j\omega)$, (d.r.) is the damping factor.

If the damping factor (d.r.) < 1 then the roots of D(s) will be complex conjugates and will lie along a circle of constant radius in the s plane as shown in Fig. (D1). The angle $\theta = \cos^{-1}$(d.r.) and ω_o determine the coordinates of the roots. As (d.r.) varies the poles move along the root locus (semi-circle).

For frequency selective filters, the quality factor (q.f.) is often used instead of the damping factor. The quality factor is given as

$$(q.f.) = \tfrac{1}{2}(d.r.)$$

$$= \omega_o/\omega_a \qquad (D3)$$

where ω_a is the upper 3 dB frequency.

The simplest form of active filter is the low pass type shown in Fig. (D2). This filter has the characteristics of two isolated RC filter sections and a buffered low impedance output. The attenuation is approximately 12 dB per octave and the ultimate attenuation is 40 dB/decade. A third RC section can be added on the output of the amplifier to achieve a third order filter with an ultimate attenuation of 60 dB/decade.

There are two basic designs for this type of filter. One is the Butterworth filter with maximally flat frequency response. For this characteristic, the components values are determined from:

$$C_1 = \frac{R_1 + R_2}{\sqrt{2}\, R_1 R_2 \omega_a}$$

and

$$C_2 = \frac{\sqrt{2}}{(R_1 + R_2)\omega_a} \qquad (D4)$$

The second kind is the linear-phase filter (Bessel filter)[75] and is only necessary for pulse inputs.

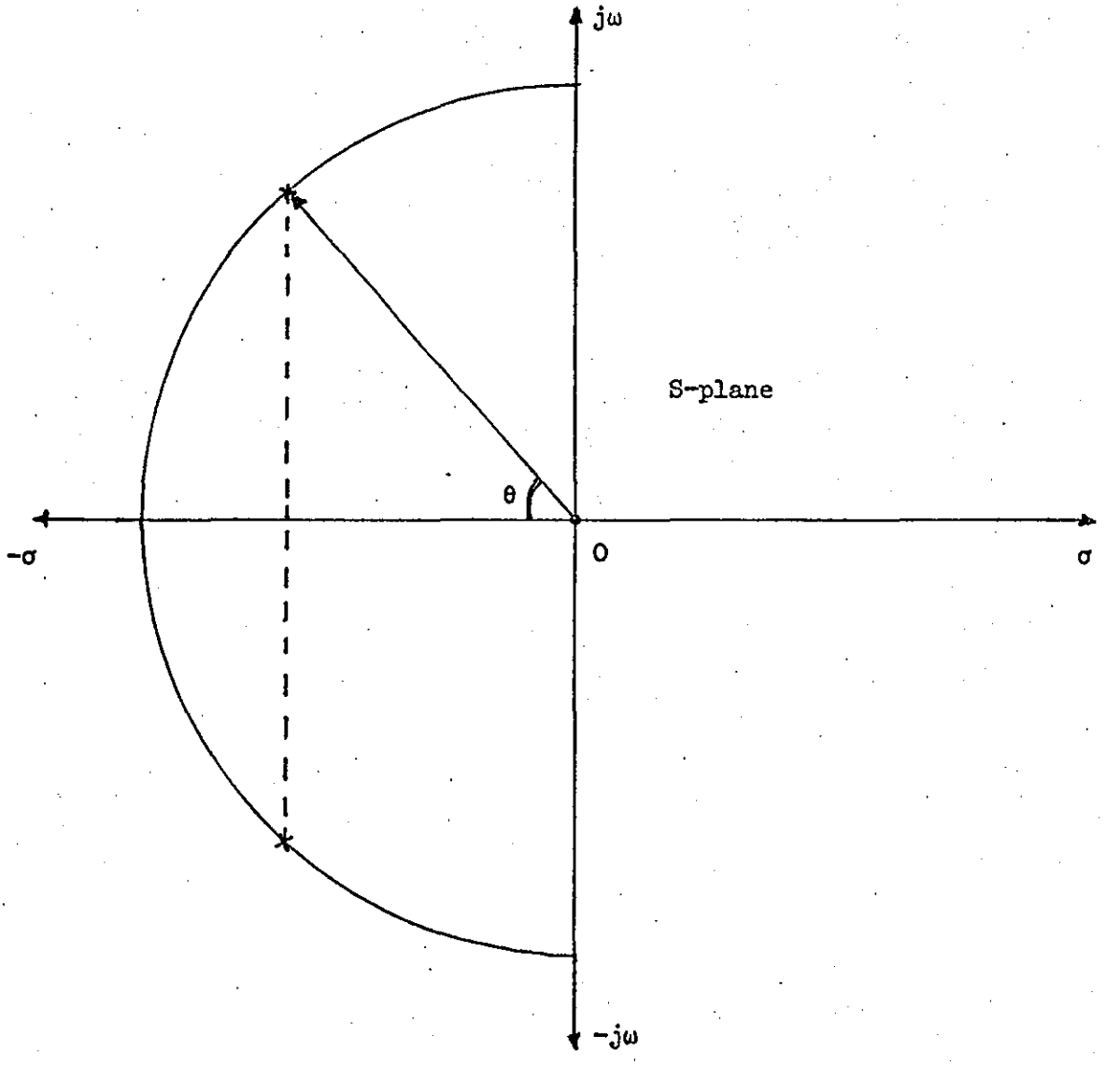


Figure (D1) The poles of D(S) when (d.r.) < 1. The poles can be placed anywhere in the S-plane to achieve the desired damping value.

$\theta = \cos^{-1}$(d.r.) and is determined by the passive components

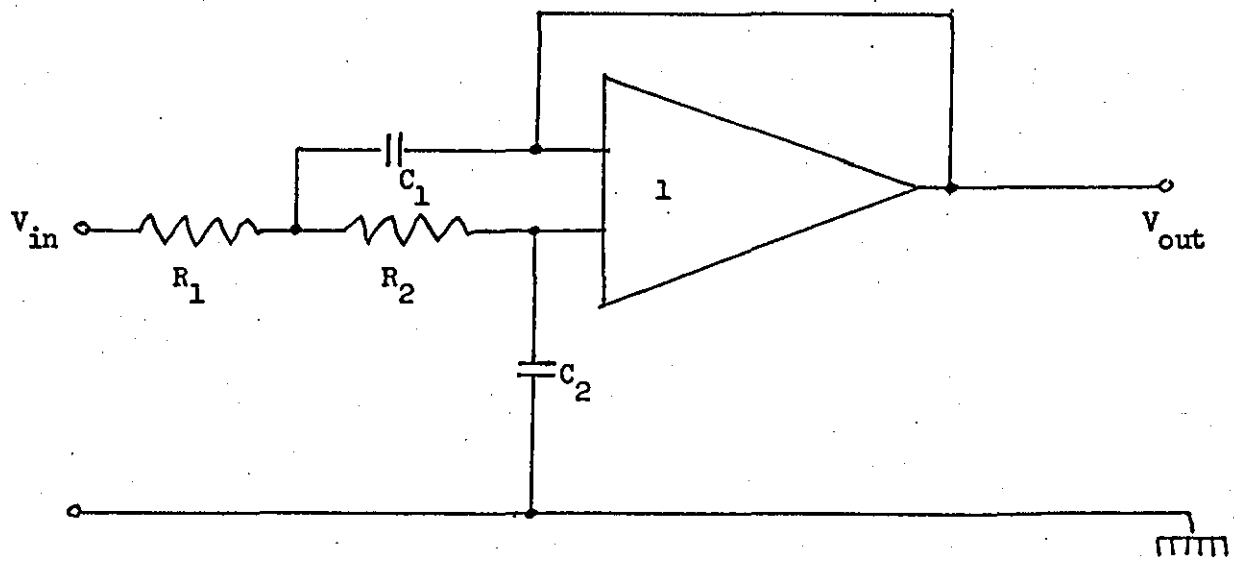


Figure (D2) Second order low pass active filter

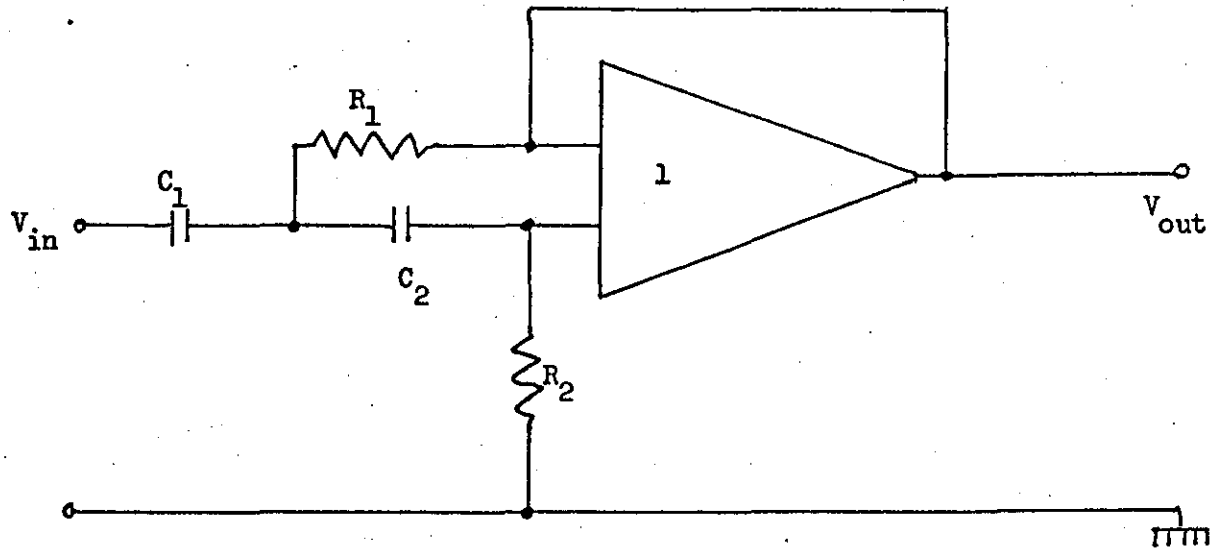
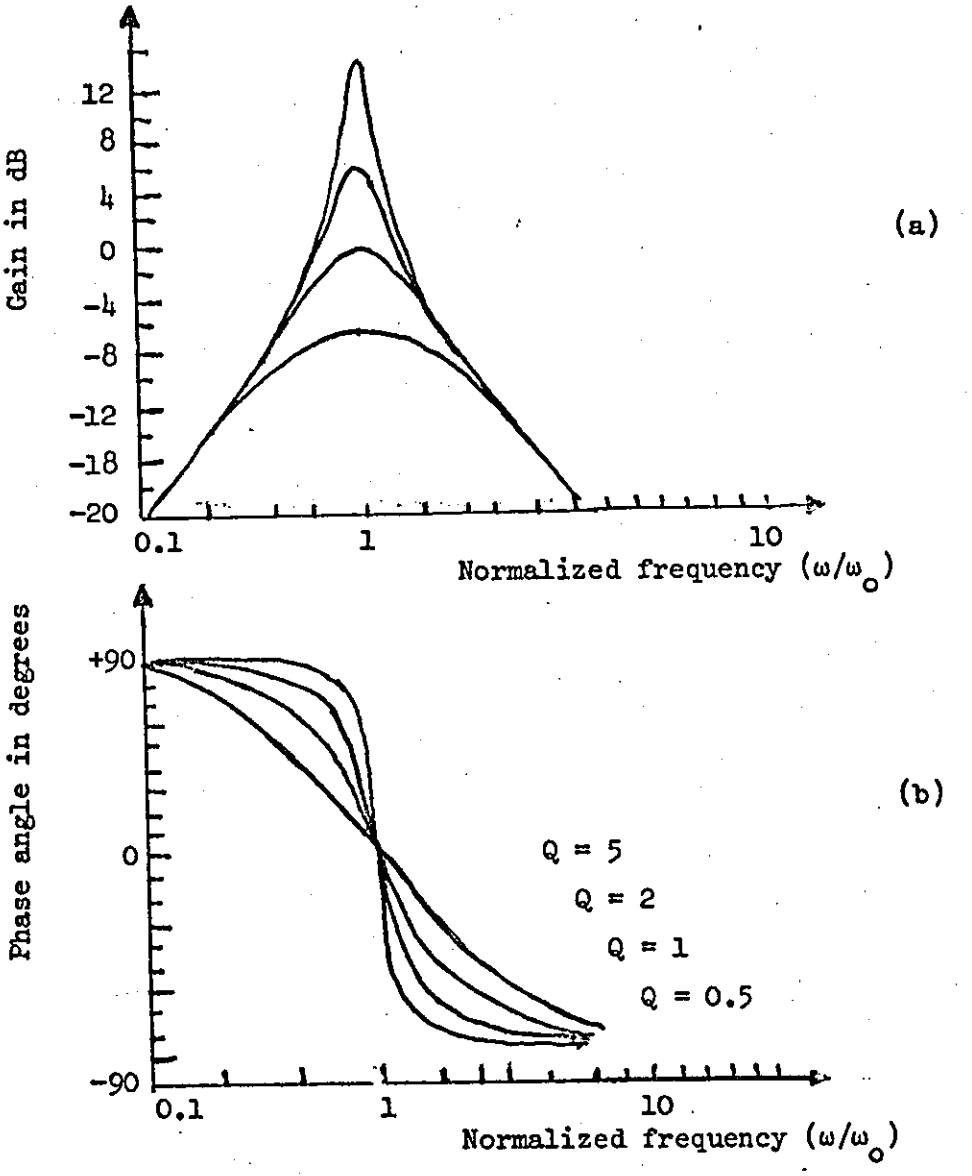


Figure (D3) Second order high pass active filter

Figure (D4) Examples of active bandpass filter characteristics

(a) Gain characteristic

(b) Phase characteristic



Bandpass filtering

Bandpass filters can be made by cascading low and high pass filter sections. These sections can be arranged in any order without affecting the overall filter response. The high pass section can be obtained from the design of a low pass prototype by means of frequency transformation[74,75]. The components of the bypass section can be calculated using the low pass section formulae but substituting capacitors for resistors and vice versa, as shown in Fig. D3 (c.f. Fig. D2).

The magnitude and phase responses of a second order pandpass filter are shown in Fig. (D4).

Since the filter bandwidth, centre frequency and midband voltage gain are usually known, desired specifications (q.f.) can be found from equation (D3). It must however be remembered that for real components (q.f.) > $\sqrt{|A1|/2}$ - for example resistors - must be positive numbers.

If the inequality is true, the capacitors C_1 and C_2 are usually made equal and given a convenient standard value. Finally, the values of the resistors are calculated using equation (D4).

APPENDIX E

COMPUTER PROGRAM USED FOR THE MANIPULATION AND PLOTTING OF THE EXPERIMENTAL RESULTS

(1) Master Segment "SCALEDESIGN"

The function of the program has been discussed in Chapter IV. A summarized flow chart of the program is given in Figure (E1). The parameters (symbols) used in the master segment are:

INPUT = An array holding the measured values of the input power in the practical model (N_e = 3,4,5).

THREE1, THREE2, THREE3, THREE4 — Arrays holding the measured SNR values corresponding to the input power values stored in "INPUT" for the case N_e = 3 and f_p = 19.2 (THREE1), 28, 38.4 and 56 Kb/s respectively.

FOUR1, FOUR2, FOUR3, FOUR4 — Arrays holding the measured SNR for the case N_e = 4 and f_p = 19.2 (FOUR1), 28, 38.4 and 56 Kb/s.

FIVE1, FIVE2, FIVE3, FIVE4 — Arrays holding the measured SNR for the case N_e = 5 and the sampling frequencies stated above.

TRE, FRE, FVE — Arrays holding the maximum values of the measured SNR for the cases N_e = 3,4,5 with f_p = 19.2, 28, 38.4, 56 Kb/s in each case.

XX, VVCC, YTT — Arrays used to hold the simulated values of: the input r.m.s. voltage σ_x, and the corresponding values of the control voltage V_c and the feedback step height H (f_p = 19.2 Kb/s, N_e = 3).

XFIT, VCFIT, HFIT — The same as XX, VVCC, YTT respectively, but the values stored are obtained by calculation.

DBINDS, SNRDS — Arrays holding the values of the measured input power in dBm and the corresponding SNR values for uncompanded Delta-sigma Modulation CODEC.

DBM, SNRCOM, SNRSIM — Arrays holding the input power in dBm measured at the input port of the input amplifier, the corresponding SNR obtained by calculation (theoretically) and the SNR obtained by simulation.

FREQ, FFREQ — Arrays holding the values of the sampling frequency for $N_e = 3,4,5$ and $N_e = 2$.

TWO110, TWO115, TWO120 — Arrays holding the measured values of SNR for the case $N_e = 2$, $f_p = 19.2$ Kb/s and $T_2 = 10,15,20$ ms. The corresponding input power values measured at the input port of the input amplifier in dBm are stored in the array XINPUT2.

SNRTWO10, SNRTWO15, SNRTWO20 — Arrays holding the maximum value of the measured SNRs above for $f_p = 19.2, 32, 38.4, 56, 76.8, 100$ Kb/s.

VCCOM = $Z_\ell = V_{co}$

VCMAXM = Z_h

VCM = V_{cmax}

HMAXM = H_{max}

HMINM = H_{min}

See Chapter II

(2) SUBROUTINE TWOBIT SEGMENT

Symbols used in this segment and are not defined above will be listed below:

TWO210, TWO215, TWO220 — Arrays holding measurement values made under the same conditions as for the case of TWO110, TWO115 and TWO120 above, except for f_p which is in this case 32 Kb/s.

TWO310, TWO315, TWO320 } As above but f_p = 38.4 Kb/s

TWO410, TWO415, TWO420 } As above but f_p = 56 Kb/s

TWO510, TWO515, TWO520 } As above but f_p = 76.8 Kb/s

TWO610, TWO615, TWO620 } As above but f_p = 100 Kb/s

(3) SUBROUTINE RESULT

The main symbols used in this subroutine are:

FMT = "FORMAT" array. (Execution time FORMT)

SRLENG = An array holding information about the shift register length.

XNUM1, XNUM2, XNUM3, XNUM4 } Arrays which are defined in the subroutine.

(4) SUBROUTINES FITTER and COMPARE

The main symbols used in these subroutines are:

ALPHAM = A first approximation value of δ

TONE = T_1

CONST = k_c

BANDWIDTH = B

FONE = f_1

PI = π = 3.14...

FACTOR = R_c

DBP = R_c in dB

} See Chapters II, III and IV

(5) SUBROUTINE SNRACL

The symbols which have not yet been defined are:

KADJUST = An arbitrary switching (JUMP) parameter.

DBGAIN = Gain of the input filter amplifier in dB.

DBSHIFT = dB to dBm conversion factor in dB.

NRM, KILO1, NRP } Number of points to be plotted on the graph.

(6) DATA

The data to be read into the computer memory and used by this program are listed at the end of the FORTRAN segments.

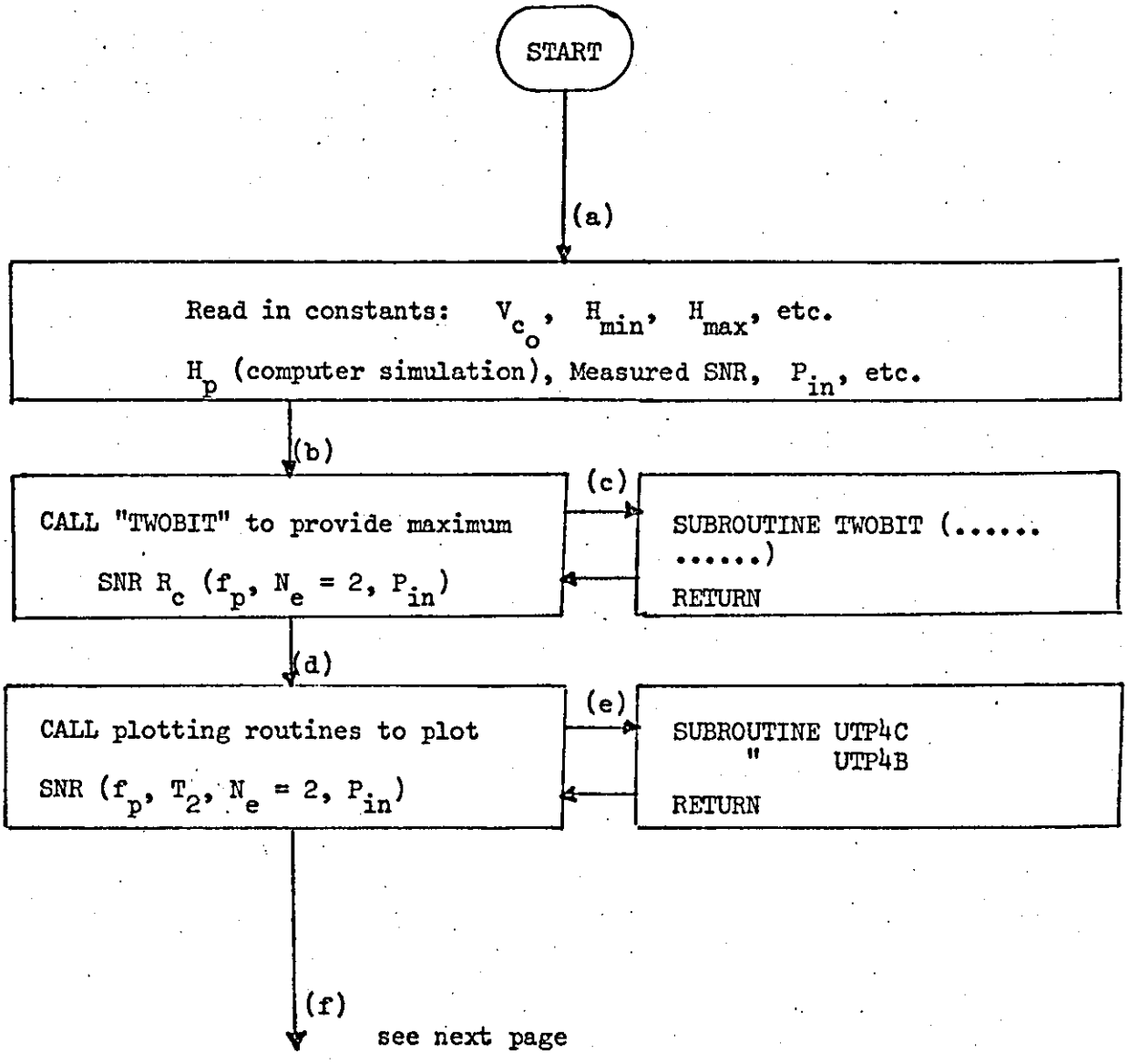
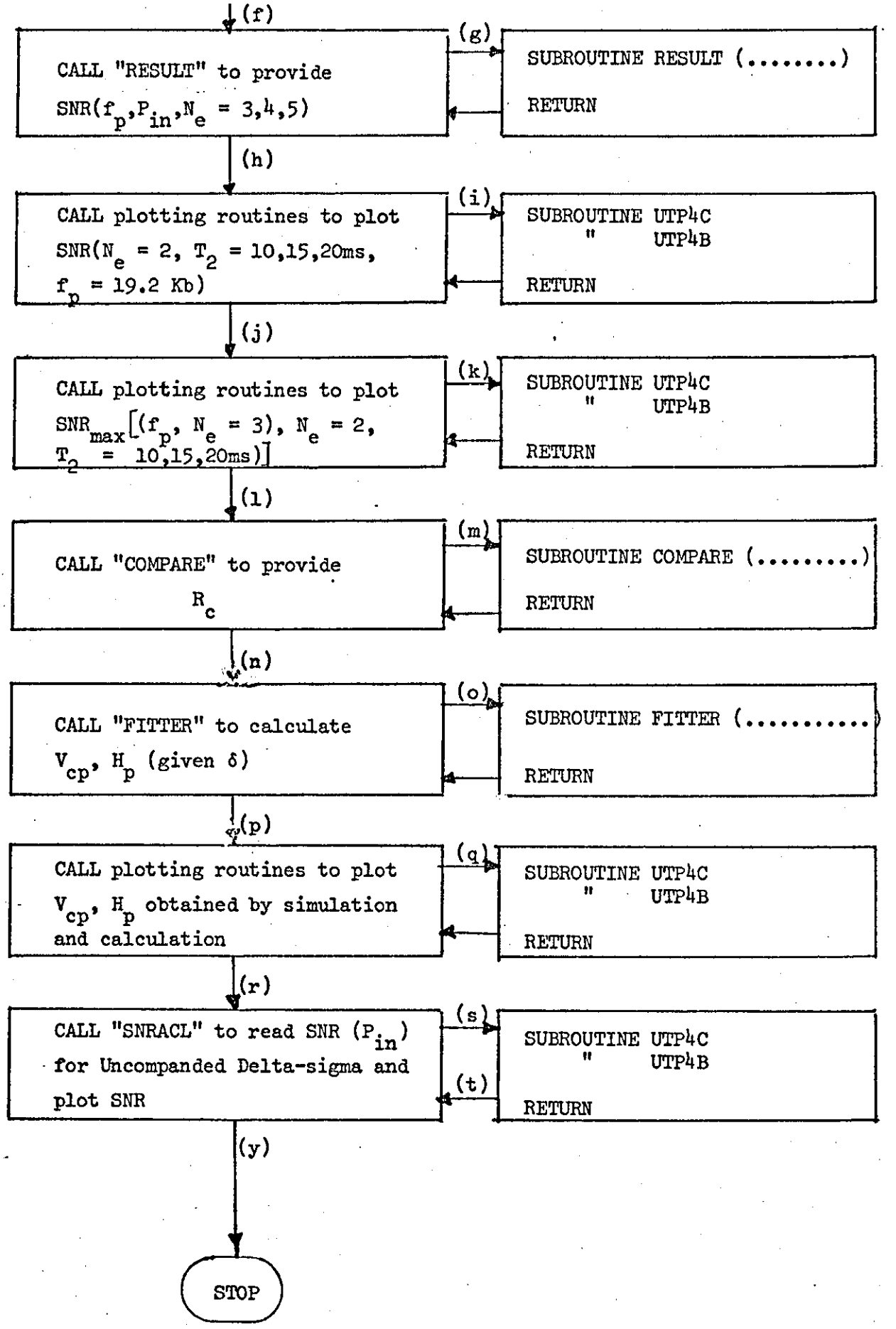


Figure (E1) Flow diagram of the program "SCALE DESIGN" employed to manipulate the measured and simulation results and to calculate SNR, V_c, H_p, etc. using the theory of Chapter II.


(f)
CALL "RESULT" to provide
SNR(f_p,P_{in},N_e = 3,4,5)
(g)
SUBROUTINE RESULT (.....)
RETURN
(h)
CALL plotting routines to plot
SNR(N_e = 2, T_2 = 10,15,20ms,
f_p = 19.2 Kb)
(i)
SUBROUTINE UTP4C
" UTP4B
RETURN
(j)
CALL plotting routines to plot
$SNR_{max}[(f_p, N_e = 3), N_e = 2,$
T_2 = 10,15,20ms)]
(k)
SUBROUTINE UTP4C
" UTP4B
RETURN
(l)
CALL "COMPARE" to provide
R_c
(m)
SUBROUTINE COMPARE (.....)
RETURN
(n)
CALL "FITTER" to calculate
V_{cp}, H_p (given δ)
(o)
SUBROUTINE FITTER (.....)
RETURN
(p)
CALL plotting routines to plot
V_{cp}, H_p obtained by simulation
and calculation
(q)
SUBROUTINE UTP4C
" UTP4B
RETURN
(r)
CALL "SNRACL" to read SNR (P_{in})
for Uncompanded Delta-sigma and
plot SNR
(s)
SUBROUTINE UTP4C
" UTP4B
(t)
RETURN
(y)
STOP

```
JOB MN84PLCC,E,MN1905
CARDLIST
LUFORTRAN
RUN ,,1000
JOBCORE 32K
VOLUME 7500
DOCUMENT SOURCE
      LIBRARY (ED,SUBGROUPGRAF)
      PROGRAM(MN84)
      COMPRESS INTEGER AND LOGICAL
      COMPACT
      INPUT 1 = CRO
      OUTPUT 2 = LPO
      TRACE 2
      END
      MASTER SCALEDESIGN
      REAL INPUT
      INTEGER CFIX
      DIMENSION DBMA(50),SNRSIMA(50),SNRSIMB(50),SNRCOMA(50)
      DIMENSION SNRTHREA(50)
      DIMENSION INPUT(15)
      DIMENSION THREE1(15),FOUR1(15),FIVE1(15)
      DIMENSION THREE2(15),FOUR2(15),FIVE2(15)
      DIMENSION THREE3(15),FOUR3(15),FIVE3(15)
      DIMENSION THREE4(15),FOUR4(15),FIVE4(15)
      DIMENSION TRE(15),FRE(15),FVE(15)
      DIMENSION XX(50),VVCC(50),YTT(50)
      DIMENSION HFIT(50)
      DIMENSION CDBP(4),CFACT(4)
      DIMENSION VCFIT(50)
      DIMENSION XFIT(50)
      DIMENSION SNRDS(10),DBINDS(10)
      DIMENSION SNRCOM(50),SNRSIM(50),DBM(50)
      DIMENSION FREQ(4)
      DIMENSION XINPUT2(15),TWO110(15),TWO115(15),TWO120(15),FFREQ(10
      DIMENSION SNRTWO10(10),SNRTWO15(10),SNRTWO20(10)
C
```

```
COPEN MAGNETIC TAPE TO STORE DATA FOR GRAPH PLOTTING
C
      CALL UTPOP
C
C STORE THE CONSTATS VCMIN,VCMAX,HMIN,HMAX,AND THE SAMPLING FREQUENCI
      VCCOM=0.40
      VCMAXM=3.40
      VCMAXM=3.8
      VCM=2.10
      HMAXM=2.4
      HMINM=0.05
      ALPHAM=0.25
      LIMAZ=50
      FREQ(1)=19.2
      FREQ(2)=28.0
      FREQ(3)=38.4
      FREQ(4)=56.0
      FP=FREQ(4)*1000.0
      FP=FREQ(1)*1000.0
      NR=18
      NRM=29
      NRMM=NRM-1
      NRTWO=13
      NRTHRE=12
      KPOINTS=6
C
C DEFINE AND STORE AXIS LENGTHS AND SCALING INFORMATION FOR
C FOR PLOTTING THE REQUIRED GRAPHS
C
      XMIN=10.0
      XMAX=110.0
      YMIN=10.0
      YMAX=30.0
      XINS=8.0
```

```
      YINS=5.0
      XINS=5.0
      YINS=4.0
      CALL TWOBIT(FFREQ,SNRTWO10,SNRTWO15,SNRTWO20,
     1XINPUT2,TWO110,TWO115,TWO120)
C
C CALL THE PLOTTING ROUTINES AND START PLOTTING THE REQUIRED
C GRAPHS
C
      CALL UTP4C(XMIN,XMAX,YMIN,YMAX,XINS,YINS,
     1'FREQUENCY IN KHZ',2,
     2'MAX SNR',1)
      CALL UTP4B(FFREQ,SNRTWO10,KPOINTS,0)
      CALL UTP4B(FFREQ,SNRTWO15,KPOINTS,0)
      CALL UTP4B(FFREQ,SNRTWO20,KPOINTS,0)
      CALL RESULT(INPUT,THREE1,FOUR1,FIVE1,FREQ,DBM,
     2THREE3,THREE4,
     3TRE,FRE,FVE,
     1THREE2,FOUR2,FIVE2)
      KPOINTS=4
      XMAX=60.0
      CALL UTP4C(XMIN,XMAX,YMIN,YMAX,XINS,YINS,
     1'FREQUENCY IN KHZ',2,
     2'MAX SNR',1)
      CALL UTP4B(FFREQ,SNRTWO10,KPOINTS,0)
      CALL UTP4B(FFREQ,SNRTWO15,KPOINTS,0)
      CALL UTP4B(FFREQ,SNRTWO20,KPOINTS,0)
      CALL UTP4B(FREQ,TRE,KPOINTS,0)
      CALL UTP4C(XMIN,XMAX,YMIN,YMAX,XINS,YINS,
     1'FREQUENCY IN KHZ',2,
     2'MAX SNR',1)
      CALL UTP4B(FFREQ,SNRTWO10,KPOINTS,0)
      CALL UTP4B(FREQ,TRE,KPOINTS,0)
      CALL UTP4B(FREQ,FRE,KPOINTS,0)
      CALL UTP4B(FREQ,FVE,KPOINTS,0)
C
```

```
C REDEFINE THE AXIS LENGTHS ETC,
C
      XMIN=-55.0
      XMAX=5.0
      YMIN=0.0
      YMAX=20.0
      XINS=7.5
      YINS=5.0
      XINS=5.0
      YINS=4.0
C
C CONTINUE THE PLOTTING
C
      CALL UTP4C(XMIN,XMAX,YMIN,YMAX,XINS,YINS,
     3'INPUT POWER IN DBS',3,
     4'SNR',1)
      CALL UTP4B(XINPUT2,TWO110,NRTWO,0)
      CALL UTP4B(XINPUT2,TWO115,NRTWO,0)
      CALL UTP4B(XINPUT2,TWO120,NRTWO,0)
      CALL UTP4B(INPUT,THREE1,NRTHRE,0)
      DO18J=1,NRM
      READ(1,19)XX(J),VVCC(J),YTT(J)
   19 FORMAT(3F0.0)
   18 CONTINUE
      DO20J=1,NRM
      WRITE(2,21)XX(J),VVCC(J),YTT(J)
   21 FORMAT(1H ,3F20.3)
   20 CONTINUE
      CFIX=3
      CALL FITTER(LIMAZ,HMAXM,ALPHAM,VCFIT,VCCOM,VCMAXM,HFIT,
     1XX,
     1VVCC,
     1XFIT,VCM,CFIX,HMINM)
      CALL UTP4C(0.0,8.0,0.0,4.0,5.0,4.0,
     1'INPUT VOLTAGE IN VOLTS',3,
     1'SYLLABIC INT OUTPUT',3)
      CALL UTP4B(XFIT,VCFIT,NRMM,0)
      CALL UTP4B(XFIT,VVCC,NRMM,0)
C*************************************************************
```

```
      CALL UTP4C(0.0,8.0,0.0,4.0,5.0,4.0,
     1'INPUT VOLTAGE IN VOLTS',3,
     1'FEEDBACK YT IN VOLTS',3)
      CALL UTP4B(XFIT,HFIT,NRMM,0)
      CALL UTP4B(XFIT,YTT,NRMM,0)
      WRITE(2,4443)(XFIT(MA),VCFIT(MA),VVCC(MA),HFIT(MA),YTT(MA),
     1MA=1,NRMM)
 4443 FORMAT(5X,5F15.3)
      NRMM=NRM-7
      CALL UTP4C(0.0,4.0,0.0,2.0,5.0,4.0,
     1'INPUT VOLTAGE IN VOLTS',3,
     1'SYLLABIC INT OUTPUT',3)
      CALL UTP4B(XFIT,VCFIT,NRMM,0)
      CALL UTP4B(XFIT,VVCC,NRMM,0)
      CALL UTP4C(0.0,4.0,0.0,2.0,5.0,4.0,
     1'INPUT VOLTAGE IN VOLTS',3,
     1'FEEDBACK YT IN VOLTS',3)
      CALL UTP4B(XFIT,HFIT,NRMM,0)
      CALL UTP4B(XFIT,YTT,NRMM,0)
      NRMM=NRM-1
      CALL COMPARE(FP,FACTOR,DBP)
      FACTOR=SQRT(FACTOR)
      NRHO=4
    3 DO7777KADJST=1,2
      NRM=29
      NRP=12
      CALL SNRACL(XFIT,HFIT,YTT,FACTOR,MRHO,NRM,DBM,THREE1,
     1SNRCOM,SNRSIM,
     1KADJST,
     1INPUT,NRP)
  444 CONTINUE
 7777 CONTINUE
C
C CLOSE THE MAGNETIC TAPE
C
      CALL UTPCL
      STOP
      END
```

```
C
      SUBROUTINE TWOBIT(FFREQ,SNRTWO10,SNRTWO15,SNRTWO20,
     1XINPUT2,TWO110,TWO115,TWO120)
      DIMENSION TWO110(15),TWO210(15),TWO310(15),TWO410(15),TWO510(15
      DIMENSION TWO610(15),TWO115(15),TWO215(15),TWO315(15),TWO415(15
      DIMENSION TWO515(15),TWO615(15),TWO120(15),TWO220(15),TWO320(15
      DIMENSION TWO420(15),TWO520(15),TWO620(15),XINPUT2(15)
      DIMENSION FFREQ(10),SNRTWO10(10),SNRTWO15(10),SNRTWO20(10)
      KPOINTS=6
      NRTWO=13
      DO25I=1,6
      READ(1,24)FFREQ(I),SNRTWO10(I),SNRTWO15(I),SNRTWO20(I)
      WRITE(2,26)FFREQ(I),SNRTWO10(I),SNRTWO15(I),SNRTWO20(I)
   25 CONTINUE
      DO18I=1,NRTWO
      READ(1,19)XINPUT2(I),TWO120(I),TWO220(I),TWO320(I),TWO420(I),
     1TWO520(I),TWO620(I)
      XINPUT2(I)=-1.0*XINPUT2(I)
      WRITE(2,20)XINPUT2(I),TWO120(I),TWO220(I),TWO320(I),TWO420(I),
     2TWO520(I),TWO620(I)
   18 CONTINUE
      GAINDB=22.3
C
C ATTENUATE THE IPUT SIGNAL BY 22.3 DB TO COMPENSATE FOR
C THE GAIN OF THE INPUT AMPLIFIER.
C
      GAINSHFT=2.3
      CORECT=GAINDB-GAINSHFT
      DO22I=1,NRTWO
      READ(1,24)TWO115(I),TWO215(I),TWO315(I),TWO415(I),TWO515(I),
     5TWO615(I)
      WRITE(2,20)XINPUT2(I),TWO115(I),TWO215(I),TWO315(I),TWO415(I),
     4TWO515(I),TWO615(I)
   22 CONTINUE
      DO21I=1,NRTWO
      READ(1,24)TWO110(I),TWO210(I),TWO310(I),TWO410(I),TWO510(I),
```

```
     6TWO610(I)
      WRITE(2,20)XINPUT2(I),TWO110(I),TWO210(I),TWO310(I),TWO410(I),
     7TWO510(I),TWO610(I)
   21 CONTINUE
   19 FORMAT(7F0.0)
   20 FORMAT(1H ,5X,7F13.2)
   24 FORMAT(6F0.0)
   26 FORMAT(5X,6F15.2)
      RETURN
      END
C
```

```
C
      SUBROUTINE RESULT(INPUT,THREE1,FOUR1,FIVE1,FREQ,DBM,
     2THREE3,THREE4,
     3TRE,FRE,FVE,
     1THREE2,FOUR2,FIVE2)
      REAL INPUT
      DIMENSION FREQ(4)
      DIMENSION THREE1(15)
      DIMENSION THREE2(15)
      DIMENSION THREE3(15)
      DIMENSION THREE4(15)
      DIMENSION FOUR1(15)
      DIMENSION FOUR2(15)
      DIMENSION FOUR3(15)
      DIMENSION FOUR4(15)
      DIMENSION FIVE1(15)
      DIMENSION FIVE2(15)
      DIMENSION FIVE3(15)
      DIMENSION FIVE4(15)
      DIMENSION INPUT(15)
      DIMENSION FMT(12)
      DIMENSION TRE(4)
      DIMENSION FRE(4)
      DIMENSION FVE(4)
      DIMENSION SRLENG(4),XNUM1(4),XNUM2(4),XNUM3(4),XNUM4(4)
      READ(1,2)FMT
    2 FORMAT(12A6)
      READ(1,FMT)(INPUT(I),I=1,12)
      READ(1,FMT)(THREE1(I),I=1,12)
      READ(1,FMT)(FOUR1(I),I=1,12)
      READ(1,FMT)(FIVE1(I),I=1,12)
      READ(1,FMT)(THREE2(I),I=1,12)
      READ(1,FMT)(FOUR2(I),I=1,12)
      READ(1,FMT)(FIVE2(I),I=1,12)
      READ(1,FMT)(THREE3(I),I=1,12)
      READ(1,FMT)(FOUR3(I),I=1,12)
      READ(1,FMT)(FIVE3(I),I=1,12)
      READ(1,FMT)(THREE4(I),I=1,12)
```

```
      READ(1,FMT)(FOUR4(I),I=1,12)
      READ(1,FMT)(FIVE4(I),I=1,12)
      DO7I=1,12
      INPUT(I)=-1.0*INPUT(I)
    7 CONTINUE
      READ(1,3)FMT
    3 FORMAT(12A6)
      TRE(1)=THREE1(7)
      TRE(2)=THREE2(6)
      TRE(3)=THREE3(6)
      TRE(4)=THREE4(5)
      FRE(1)=FOUR1(7)
      FRE(2)=FOUR2(6)
      FRE(3)=FOUR3(5)
      FRE(4)=FOUR4(5)
      FVE(1)=FIVE1(6)
      FVE(2)=FIVE2(6)
      FVE(3)=FIVE3(5)
      FVE(4)=FIVE4(5)
      XNUM1(1)=TRE(1)
      XNUM1(2)=FRE(1)
      XNUM1(3)=FVE(1)
      XNUM1(4)=13.7
      XNUM2(1)=TRE(2)
      XNUM2(2)=FRE(2)
      XNUM2(3)=FVE(2)
      XNUM2(4)=16.6
      XNUM3(1)=TRE(3)
      XNUM3(2)=FRE(3)
      XNUM3(3)=FVE(3)
      XNUM3(4)=19.7
      XNUM4(1)=TRE(4)
      XNUM4(2)=FRE(4)
      XNUM4(3)=FVE(4)
      XNUM4(4)=21.5
      DO99I=1,4
   99 SRLENG(I)=I
```

```
      WRITE(2,4)
    4 FORMAT(/////)
      WRITE(2,FMT)(INPUT(I),THREE2(I),FOUR2(I),FIVE2(I),I=1,12)
      WRITE(2,5)
    5 FORMAT(/////)
      WRITE(2,FMT)(INPUT(I),THREE3(I),FOUR3(I),FIVE3(I),I=1,12)
      WRITE(2,6)
    6 FORMAT(/////)
      WRITE(2,FMT)(INPUT(I),THREE4(I),FOUR4(I),FIVE4(I),I=1,12)
      WRITE(2,6)
      WRITE(2,FMT)(FREQ(I),TRE(I),FRE(I),FVE(I),I=1,4)
      WRITE(2,6)
      WRITE(2,FMT)(SRLENG(I),XNUM1(I),XNUM2(I),XNUM3(I),I=1,4)
      WRITE(2,6)
      WRITE(2,FMT)(SRLENG(I),XNUM2(I),XNUM3(I),XNUM4(I),I=1,4)
      RETURN
      END
```

```
C
C
      SUBROUTINE FITTER(LIMAZ,HMAXM,ALPHAM,VCFIT,VCCOM,VCMAXM,HFIT,
     1XX,
     1VVCC,
     1XFIT,VCM,CFIX,HMINM)
      INTEGER CFIX
      DIMENSION XX(LIMAZ)
      DIMENSION XFIT(50)
      DIMENSION HFIT(LIMAZ)
      DIMENSION VCFIT(50)
      NRM=29
      NRMM=NRM-1
      DO1I=1,NRM
      X=XX(I)
      GO TO(5,6,7,8),CFIX
    5 CFIT=1.0
      GO TO 9
    6 CFIT=0.86
      GO TO 9
    7 CFIT=0.83
      GO TO 9
    8 CFIT=0.66
    9 XINDEX=X*ALPHAM*CFIT
      XINDEX=X*0.22
      XINDEX=0.21*X
      RECIP=1.0/EXP(XINDEX)
      XFIT(I)=X
      VCFIT(I)=VCMAXM*(1.0-RECIP)+VCCOM
      GRAD=(HMAXM-HMINM)/(VCM-VCCOM)
      HFIT(I)=GRAD*(VCFIT(I)-VCCOM)+HMINM
      IF(HFIT(I).GT.2.4)HFIT(I)=2.4
    1 CONTINUE
      WRITE(2,4)(XFIT(K),VCFIT(K),HFIT(K),K=1,NRMM)
    4 FORMAT(1H ,10X,F10.3,10X,F10.3,10X,F10.3)
      RETURN
      END
```

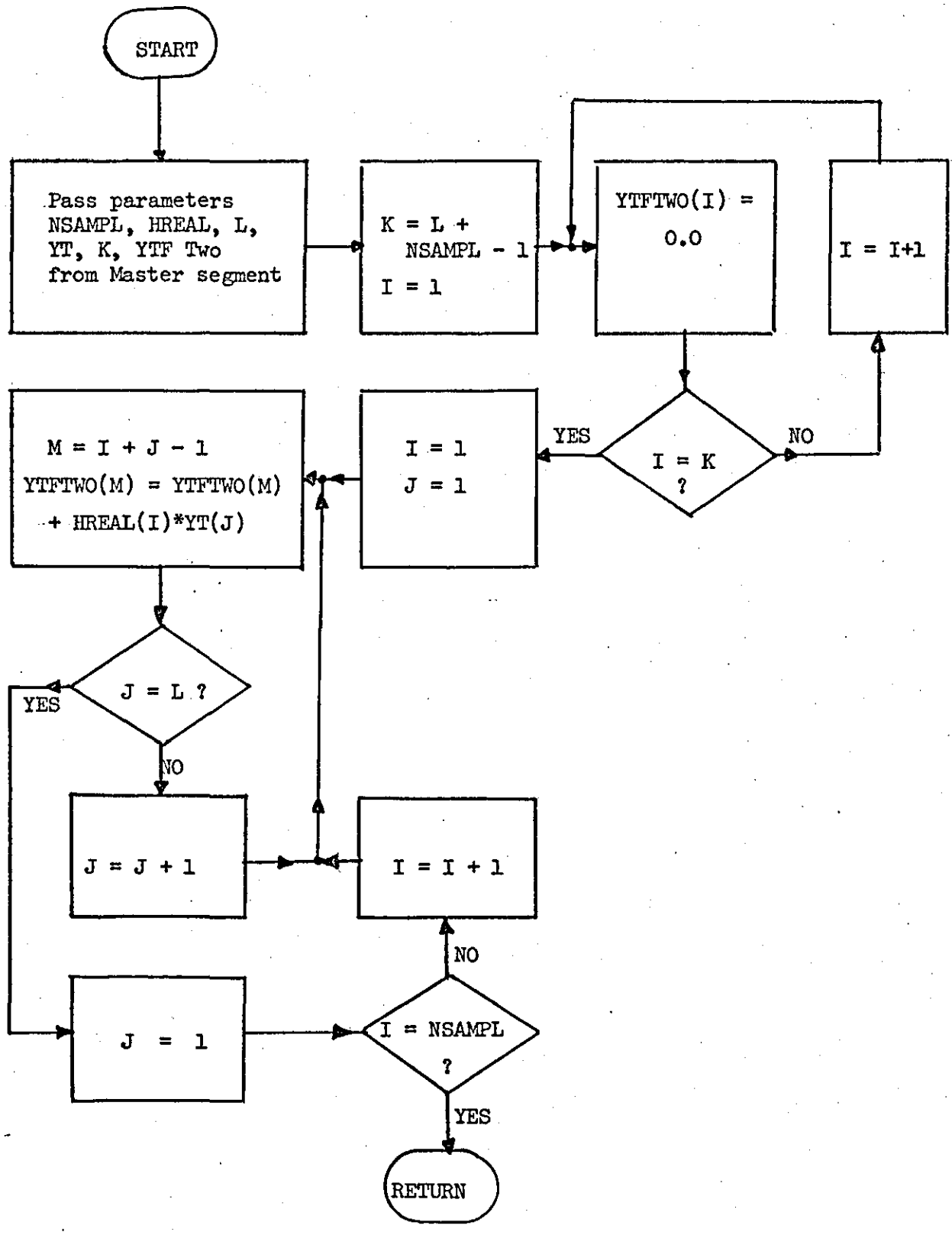


Figure E2 Flow chart of "SUBROUTINE COMPARE"

```
C
C
      SUBROUTINE COMPARE(FP,FACTOR,DBP)
      PI=3.141259
      TONE=.0002
      CONST=0.6
      BWIDTH=2400.0
      FONE=1/2.0*PI*TONE
      DENOM1=8.0*PI*PI*CONST
      DENOM2=3.0*FONE*FONE*BWIDTH+BWIDTH*BWIDTH*BWIDTH
      DENOM=DENOM1*DENOM2
      FACTOR=3.0*FP*FP*FP/DENOM
      WRITE(2,1)FACTOR
    1 FORMAT(1H ,10X,E14.7)
      DBP=10.0*ALOG10(FACTOR)
      WRITE(2,2)DBP
    2 FORMAT(1H ,10X,E14.7)
      RETURN
      END
```

```
C
C
C
C
      SUBROUTINE SNRACL(XFIT,HFIT,YTT,FACTOR,MRHO,NRM,DBM,THREE1,
     1SNRCOM,SNRSIM,
     1KADJST,
     1INPUT,NRP)
      REAL INPUT
      DIMENSION SNRDS(10),DBINDS(10)
      DIMENSION INPUT(NRP)
      DIMENSION SNRCOM(NRM),SNRSIM(NRM)
      DIMENSION XFIT(NRM),HFIT(NRM),YTT(NRM),DBM(NRM),THREE1(12)
      IF(KADJST.EQ.2.AND.MRHO.GE.3)GO TO 35
      NRMM=NRM-1
C
C ATTENUATE THE IPUT SIGNAL BY 22.3 DB TO COMPENSATE FOR
C THE GAIN OF THE INPUT AMPLIFIER.
C
      DBGAIN=22.3
      DBSHFT=2.3
      CORRDB=DBGAIN-DBSHFT
      DBCORR=CORRDB
      IF(MRHO.GT.1.OR.KADJST.GT.1)GO TO 7
      CALL UTP4C(-55.0,5.0,0.0,20.0,8.0,5.0,
     1'INPUT POWER IN DBMS',3,
     1'STNR IN DBS',2)
    7 IF(MRHO.EQ.4)GO TO 29
      IF(MRHO.EQ.3)GO TO 18
      DO17I=1,NRM
      GO TO(19,20),MRHO
      GO TO 21
   19 HF=HFIT(I)
   20 HF=YTT(I)
      SNRCSM=20.0*ALOG10(XFIT(I)*FACTOR/HF)
   21 IF(XFIT(I).GT.HF)GO TO 1
      GO TO 23
    1 SNRCSM=20.0*ALOG10(HF*FACTOR/XFIT(I))
```

```
 23 GO TO(24,25),MRHO
 24 SNRCOM(I)=SNRCSM
    GO TO 26
 25 SNRSIM(I)=SNRCSM
 26 DBM(I)=20.0*ALOG10(XFIT(I))
    DBM(I)=DBM(I)-DBCORR
    GO TO 555
 17 CONTINUE
 18 KILO1=8
    DO2J=1,KILO1
    READ(1,3)DBINDS(J),SNRDS(J)
  2 CONTINUE
  3 FORMAT(2F0.0)
    WRITE(2,4)(DBINDS(K),SNRDS(K),K=1,KILO1)
  4 FORMAT(1H ,10X,2F20.3)
555 GO TO(27,28,39),MRHO
 27 CALL UTP4B(DBM,SNRCOM,NRM,0)
    GO TO 30
 28 CALL UTP4B(DBM,SNRSIM,NRM,0)
    GO TO 30
 39 CALL UTP4B(DBINDS,SNRDS,KILO1,0)
 29 CALL UTP4B(INPUT,THREE1,NRP,0)
 40 WRITE(2,41)(INPUT(I),THREE1(I),I=1,NRP)
 41 FORMAT(1H ,10X,2F20.3)
    GO TO 35
 30 WRITE(2,31)(DBM(I),SNRCOM(I),SNRSIM(I),I=1,NRM)
 31 FORMAT(1H1,10X,E14.7,10X,E14.7,10X,E14.7)
    GO TO(35,5),KADJST
  5 LIB=1
    GO TO 35
 35 RETURN
    END
```

```
C
C
C
DOCUMENT DATA
C
C
C  THE DOCUMENT DATA CONTAIN THE RESULTS OF THE MEASUREMENTS
C   MADE ON THE HARDWARE AND THE SIMULATED MODELS OF THE SCALE
C SYSTEM AND ITS ASSOCIATED  FILTERS ETC.,
C  THE MEANING OF ANY PARTICULAR BLOCK OR SINGLE WORD OF DATA
C  CAN BE FOUND BY SIMPLY FOLLOWING THE READ STATEMENTS IN THE
C  MASTER AND SUBROUTINE SEGMENTS.
C
C
19.2       13.0       13.4       13.0
32.0       17.5       18.0       17.3
38.4       19.0       19.5       19.5
56.0       23.5       23.5       23.0
76.8       26.5       26.5       26.0
100.0      28.5       28.5       28.5
50.0   5.7   8.5   10.0   13.0   16.0   19.5
45.0   9.0   13.0   14.5   17.0   21.0   24.0
40.0   10.5   16.0   18.5   22.0   25.0   28.5
37.0   12.0   16.5   19.0   23.0   26.0   28.5
36.0   12.5     117.0     119.2     23.0   25.00   5.5
 3.0   113.0   17.3   19.5   22  24.0   24.5
30.0   13.0   13.0   16.0   17.0   15.5   16.0
25.0   12.0   12.0   13.0   12.0   12.0
25.0   12.0   12.0   13.0   12.0   12.0
20.0   10.0   10.0   10.0   10.0   10.0   9.5
15.0   9.0   9.0   9.0   9.5   8.5   8.5
10.0   8.4   8.0   8.0   8.0   8.0   8.0
5.0   7.5   7.5   7.5   7.5   7.5   7.5
3.0   6.0   6.0   6.0   6.0   6.0   6.0
6.5   9.0   10.0   12.5   16.0   19.0
9.5   14.0   14.5   17.0   21.0   23.5
11.0   16.5   18.5   21.5   25.0   28.0
13.5   18.2   18.5   23.5   26.5   28.5
13.5   18.2   18.5   23.5   26.5   28.5
12.5   18.0   19.5   22.5   19.0   25.0
```

```
13.4   16.0   16.5   17.0   16.5   16.0
12.0   12.5   12.8   12.5   12.0   12.0
10.5   10.0   10.0   10.0   9.5    10.0
9.0    9.0    9.0    8.5    8.5    8.5
8.5    8.5    8.0    8.0    8.0    8.0
7.5    7.5    7.5    7.5    7.5    7.5
5.5    6.0    5.5    6.0    6.0    6.0
5.5      8.0      9.5      12.5      16.0      19.5
9.0      13.0     14.0     17.0      21.0      24.0
10.5     16.0     18.0     22.0      25.0      28.0
12.4   18.0   19.5   23.0   25.5   26.5                    13.5
12.5     16.5     18.5     18.2      23.5      26.5
12.0     17.0     19.0     23.0      24.0      24.5
13.0     16.0     17.0     18.0      16.5      16.0
12.0     12.5     13.0     13.0      12.0      12.0
10.0     10.5     10.5     10.0      10.0      9.5
9.0      10.0     9.       9.0       8.5       8.5
8.0      8.0      8.0      8.0       8.0       8.0
7.5      7.5      7.5      7.5       7.5       7.5
7.0      6.5      6.5      6.5       7.0       7.0
(6F10.2)
50.00      45.00      40.00      35.00      30.00      25.00
20.00      15.00      10.00      5.000      1.000      0.000
6.50       9.00       10.7       11.3       13.0       14.2
14.50      14.00      11.30      8.400      6.900      6.600
7.500      8.500      9.000      10.30      12.40      14.00
14.20      13.60      11.30      8.400      6.800      6.600
6.500      7.900      8.800      10.70      12.70      14.00
13.80      13.00      11.30      8.400      6.800      6.600
7.800      11.20      14.00      15.40      16.50      18.20
17.00      15.00      12.00      8.700      8.500      7.000
10.80      13.60      14.60      15.00      17.00      17.60
16.00      14.50      12.00      8.700      8.500      7.000
11.40      13.40      13.80      15.00      16.90      17.00
15.60      14.00      11.60      8.700      8.600      7.000
10.20      14.00      17.40      19.60      20.80      21.00
18.20      15.60      12.00      9.600      8.700      7.000
```

13.30	17.00	19.00	19.40	20.70	19.70
17.00	14.80	12.00	9.000	8.700	7.000
13.70	14.40	18.40	19.00	20.40	18.70
16.20	14.30	11.80	9.000	8.700	7.000
12.80	17.70	16.40	24.50	25.50	23.30
19.00	15.60	12.40	9.000	8.700	7.300
15.00	19.40	18.20	24.50	24.50	21.00
17.40	14.80	12.00	9.000	8.700	7.300
16.30	20.30	18.50	24.00	23.00	19.80
16.80	14.50	12.00	9.000	8.700	7.300

(1H ,10X,F10.2,10X,F10.2,10X,F10.2,10X,F10.2)

0.0175	0.4	0.05
0.04	0.44	0.082
0.09	0.48	0.17
0.15	0.53	0.23
0.18	0.60	0.32
0.25	0.65	0.39
0.29	0.65	0.399
0.31	0.68	0.44
0.41	0.72	0.49
0.45	0.74	0.53
0.47	0.77	0.56
0.58	0.86	0.68
0.67	0.87	0.70
0.86	0.99	0.87
1.05	1.04	0.94
1.25	1.22	1.20
1.53	1.32	1.34
1.75	1.36	1.39
1.94	1.44	1.51
2.21	1.65	1.80
2.60	1.60	1.72
2.85	1.74	1.92
3.37	1.76	1.96
4.03	1.89	2.13
4.86	2.03	2.33
5.12	2.14	
5.44	2.11	2.40
6.00	2.18	2.40
11.0	3.5	2.4

```
-35.0        3.25
-30.0        6.5
-25.0        11.75
-21.0        14.5
-20.0        14.20
-15.0        10.0
-10.0        6.00
-5.0         2.50
        0.0160      0.0400      0.0880      0.1440      0.1760
        0.2480      0.2880      0.3040      0.4080      0.4480
        0.4640      0.5760      0.6640      0.8560      1.048
        1.248       1.528       1.744       1.936       2.208
        2.600       2.848       3.368       4.024
        0.0330      0.0480      0.0960      0.1520      0.1840
        0.2560      0.2960      0.3120      0.4160      0.4560
        0.4720      0.5840      0.6720      0.8640      1.048
        1.198       1.408       1.570       1.714       1.918
        2.158       2.282       2.542       2.870
        0.0331      0.0331      0.0960      0.1520      0.2080
        0.2640      0.3200      0.3200      0.4320      0.4320
        0.4880      0.6000      0.6560      0.8800      1.042
        1.210       1.420       1.588       1.714       1.924
        2.162       2.274       2.554       2.862
        FINISH
****
```