

ZSIM: PROGRAM DOCUMENTATION
(Manual for TR-88/1)

G.T. Brauns
S.H. Ardalan
and
M.B. Steer

Center for Communications and Signal Processing
Electrical and Computer Engineering Department
North Carolina State University

CCSP-TR-88/2

January 1988

Abstract

This report documents the progress of **ZSIM**, a nonlinear **Z**-domain **SIM**ulator for delta-sigma modulators. The simulator uses table look-up techniques to achieve high computation speeds and high accuracy is maintained by using device-level simulations to develop the look-up table. The simulator allows for simulation of circuit nonidealities including clock feed-through, nonlinearities, and hysteresis. This report solely includes a **ZSIM0a1** User's Manual and the simulator source code.

ZSIM
NONLINEAR Z DOMAIN SIMULATOR
USER'S MANUAL

version ZSIM0a1

July 8, 1987

Gregory T. Brauns
Michael B. Steer
Sasan H. Ardalan

Center for Communications and Signal Processing
Department of Electrical and Computer Engineering
North Carolina State University
Raleigh, NC 27695-7911

INDEX

1. VERSION NOTES	2
2. INSTALLATION GUIDE	2
3. I/O CONVENTIONS	3
4. PROGRAM ORGANIZATION	3
5. USERS' GUIDE	5
6. GENERATING YOUR OWN TABLES	13
7. TABLE INTERPOLATION	14
8. EXAMPLE	15

1. VERSION NOTES

ZSIM0a1 is the first distribution of ZSIM. This version is tailored to the simulation of Delta Sigma Modulators (DSM). It has a user-friendly input format but lacks a totally stand alone topology specification. Currently up to third order DSM can be simulated using difference equations and first and second order DSM can be simulated using table methods.

2. INSTALLATION GUIDE

ZSIM0a1 is written in ANSI standard FORTRAN 77 and operation has been verified on DEC MICROVAXES running Ultrix 1.2 or MICROVMS operating systems.

ZSIM0a1 is distributed as two sets of files - ZSIMALL.FOR and the individual module files.

ZSIMALL.FOR is just all the routines below plus fdate.f concatenated together.

**** ZSIMALL file ****

zsimall should be compiled and linked.

**** individual module files ****

The files below should be compiled and linked together.

agen	aquant	ascint	calfbw	calsdr
casfil	cassec	compar	decimate	desim
dffil	dtable	envirn	fft	fread
gauss	gendec	gentor	init	intgtr
nlfil	nodept	nodset	noiset	polate
ranf	sdrsub	tablrd	zana	zbdec
zcalsdr	zcmd	zdec	zdump	zequat
zhelp	zinput	zprint	zsdr	zset
zsim				

The program calls a routine fdate using

call fdate(date)

where date is dimensioned as

character*24 date

On UNIX systems this returns the date. On nonunix machines the above routine can be provided or the dummy file fdate.f (included in this distribution) can be included in the above list.

3. I/O CONVENTIONS

All input to ZSIM is converted to uppercase at input. On some systems, e.g. unix based, user's need to be aware that files are always accessed in uppercase by the program.

There is a high level input capability so that input can be taken from the keyboard and/or from a disk file - see user's guide for more information.

comments can be included by preceding an input line by ' * '.

4. PROGRAM ORGANIZATION

The program is organized by sets of modules with a common set of high level input routines contained in the file zinput.f.

The program is block oriented. A high level view of the program is as follows:

Level of program: top next

ZSIM

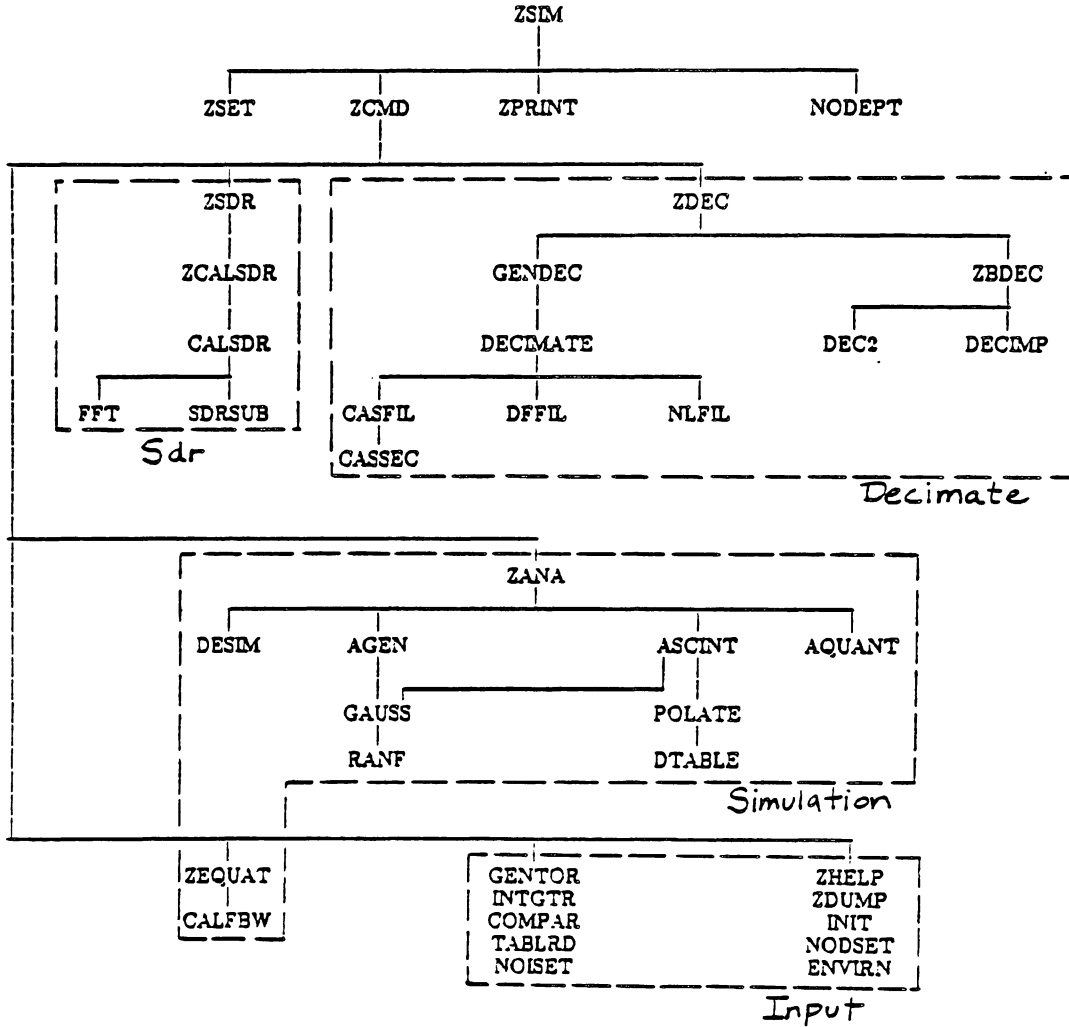
```
INPUT               ! Input parameters
SIMULATION          ! Perform simulation of sampled data system
DECIMATE            ! Perform decimation
SDR                 ! Perform SDR calculations
```

Greater detail is given on the next page.

Description of a circuit must be in terms of nodes however in ZSIM0a1 arbitrary topology has not been completely implemented. ZSIM needs to be informed as to the interconnection of blocks in the sampled data system. At present only first and second order delta sigma modulators can be modeled (indicated by the parameter CIRCUIT) using table modeling techniques and first second and third order delta sigma modulators (again indicated by the parameter CIRCUIT) using difference equation techniques.

The distinction between a table based simulation and a difference equation simulation is indicated by the command to do the simulation (the command SIMULATE performs a table based simulation and DESIMULATE does a difference equation simulation).

ZSIM: A Nonlinear Z-domain SIMulator



Input Routines: ZINPUT, FREAD

5. USER'S GUIDE

This guide is organized by commands. ZSIM is largely self documenting and help on each command is available by typing "COMMAND" HELP.

Commands Available:

COMMAND	KEYWORD	DESCRIPTION
SIMULATE	SIM	Perform z-domain simulation (tables)
DESIMULATE	DES	Perform z-domain simulation (diff eq)
GEN	GEN	Define generator input
SCINT	SCINT	Define integrator input
QUANT	QUANT	Define quantizer input
TABLE	TABLE	Define table description
CIRCUIT	CIRC	Define circuit type
ENVIRONMENT	ENV	Define simulation features
HISTOGRAM	HIST	Specify histogram for the tables
CLEAR	CLEAR	Erase all parameters to default
DUMP	DUMP	Dump contents of stored node values
STOP	STOP	End session
INIT	INIT	Initialize quantities
DECIMATE	DEC	Perform a decimation
SDR	SDR	Signal-to-Distortion calculation
EQUATION	EQ	Specify difference equation parameters
NOISE	NOISE	Define Gaussian white noise at a node

The following input/output commands are available:

read filename	the file filename is opened for input
read	the previously opened file is now used for input
write filename	the file filename is opened for output
eof	the input data file is closed
end	the input data file is temporarily closed
title	write line to output file
prompt	write line to terminal
echo on	the flag prt is set
echo off	the flag prt is cleared
	If prt is set all lines input are echoed.

CIRCUIT

Usage:

CIRCUIT HELP
: This Message

CIRCUIT = n
: Set order of modulator to "n"

#####

CLEAR

Usage:

CLEAR HELP
: This Message

CLEAR
: Set everything to default value

#####

DECIMATE

Usage:

DEC HELP
: This Message

DECIMATE n1 n2 TYPE=ZBDEC
This performs a decimation on the values of
node n1 and outputs the result (usually
the baseband) at n2 TYPE indicates the type
of canned decimation.

DECIMATE n1 n2 TYPE=GENDEC NUMBER=nn WINDOW=sss TAPS=nn
INTDEC=nn BBANDDEC=bb FILE=filename FILTER=sss
This performs a decimation on the values of
node n1 and outputs the result (usually
the baseband) at n2. NUMBER can be to
base 2, e.g.: NUMBER=nn BASE 2
WINDOWS available: UNIFORM TRIANGLE PARABOLIC
TAPS : length of the decimation filter.
INTDEC: FIR decimation factor
BBANDDEC : baseband decimation factor.
FILE specifies file with filter coefficients
FILTERs available: DIRECT_form
NORMALIZED_lattice CASCADE_form

#####

Included in this distribution of ZSIM is the file
VB.CAS which specifies a cascade filter
designed for decimation to voiceband. VB.CAS
has a 3.4 kHz rolloff for an 8kHz sampling
frequency. Coefficients for other filters
must be generated by the user. A program
for doing this is available from CCSP.

DIFFERENCE EQUATION SIMULATION

Usage:

DES HELP
: This Message

DES
: Simulate a circuit described by difference
: equations

#####

DUMP

Usage:

DUMP HELP
: This Message

DUMP file n no
: Dump first "no" values at node "n"
: to "file"

#####

INITIALIZE

Usage:

INIT HELP
: This Message

INIT NODE n x
: INitializes NODE n to x. This is required
: as several blocks use previous node values
: in calculations

#####

NOISE

Usage:

NOISE HELP
: This Message

NOISE n x1 x2
: Add Gaussian white noise to node "n" with
: mean "x1" and standard deviation "x2"

** NOT TESTED OR FULLY IMPLEMENTED **

#####

EQUATION #####
 Usage:

EQ HELP

: This Message

EQ GAIN#=xxx1 DELTA#=xxx2 OFINT#=xxx4 SAT#%=xxx5
 : # = 1, 2, or 3 for respective integrator
 : but may be left out to describe all
 : integrators.
 : This inputs ideal circuit parameters:
 : GAIN# : gain of integrator
 : DELTA# : reference voltage of quantizer
 : OFINT# : integrator offset referred to
 : the output
 : SAT#% : integrator saturation
 : % = '+' for upper limit
 : % = '-' for lower limit

NOTE Finite Gain Bandwith is not implemented.

Default Values are:

gain1	1.00	gain2	1.00	gain3	1.00
delta1	1.00	delta2	1.00	delta3	1.00
ofint1	0.	ofint2	0.	ofint3	0.
sat1	5.00	sat2	5.00	sat3	5.00

Difference equation implemented:

$y\#(\text{new}) = y\#(\text{old}) + \text{gain}\#\{x\# - \text{bit} * \text{delta}\#\} + \text{ofint}\#$
 $y\#(\text{new}) =$ is limited to + or - sat#

where $y\#$ is the output of the # integrator
 $y\#(\text{new})$ is the new output of the integrator
 $y\#(\text{old})$ is the output of the integrator at
 the previous clock cycle
 x is the analog input
 $\text{gain}\#, \text{delta}\#$ and $\text{ofint}\#$ are as defined above
 bit is the output of the
 quantizer (+ or - 1)

#####

```
##### GENERATOR #####  
Usage:
```

```
GEN HELP  
: This Message  
  
GEN no node TYPE = DC AMP=x2  
: generator number "no" at node "node"  
: DC generator of amplitude "x1" and  
  
GEN no node TYPE = RAMP THR[ESHOLD]=x1 AMP=x2  
: generator number "no" at node "node"  
: ramp generator of amplitude "x2" and  
: threshold of "x1"  
  
GEN no node TYPE = SINE FREQ=freq OFFSET=x1 AMP=x2  
: generator number "no" at node "node"  
: sinewave generator of frequency "freq",  
: amplitude "x2", offset "x1", and zero phase  
  
GEN no node TYPE = SINE FREQ=freq AMP=x1 PHASE = x2 DEG  
: generator number "no" at node "node"  
: sinewave generator of frequency "freq",  
: amplitude "x1", and phase of "x2" degrees  
  
GEN no node TYPE = SINE FREQ=freq AMP=x1 DB WRT x3  
PHASE = x4 RAD  
: generator number "no" at node "node"  
: sinewave generator of frequency "freq",  
: amplitude "x1" With Respect To "x3" V and  
: phase of "x4" radians  
  
GEN no node TYPE=STEP INIT=x1 AMP=x2 DELAY=x3  
: step generator with initial value "x1",  
: amplitude "x2" and delay of "x3" seconds
```

```
#####
```

QUANTIZER #####
Usage:

QUANT HELP
: This Message

QUANT n n1 n2 THRES=xxx1 TYPE=sss1 BAND=xxx2 SAME=nnn1
: Quantizer "n" at input node "n1" and output
: node "n2" has THRESHOLD of "xxx1" and
: TYPE = NONHysteresis or HYSTeresis.
: options 1.previously defined quantizer
: number "nnn1" uses the same
: data table
: 2."xxx2" is a deadzone region
: where the quantizer output zero.
:
: NOTE:: option 1 is presently disabled
: since the quantizer is ideal and not
: characterized by a table as of now.
: TYPE must be specified NONHysteresis
: until a hysteresis table is programmed.
: Integrator table must include x3=0
: (see section 5 on generating tables)
: when BAND option is used.

SIMULATION USING TABLES #####
Usage:

SIM HELP
: This Message

SIM
: Simulate a circuit described by tables

#####

SDR #####
Usage:

SDR HELP

: This Message

SDR n1 n2 TYPE=CALSDR NPOINTS=nnn1 NFFT=nnn2 NSKIP=nnn3
: FFTWINDOW=sss1 DECWINDOW=sss2 FB=xxx1 TAPS=nnn4
: This performs a Signal-to-Distortion
: calculation on the values of node n1.
: The calculated spectrum is
: stored as node n2. Options available:
: NPOINTS = total number of input bins
: NFFT = number of bins to use for fft
: NSKIP = number of initial bins to ignore
: (NPOINTS = NFFT + NSKIP)
: FFTWINDOW = UNIFORM HAMMING blackman-HARRIS
: DECWINDOW = IGNORE UNIFORM TRIANGULAR PARABOLIC
: (DECWINDOW should be the same
: used in decimation.)
: The following parameters can be ignored if
: decimation not used (DECWINDOW=IGNORE)
: FB : baseband sampling frequency
: TAPS : number of taps in decimation filter
: must be the same as that used in
: the decimate command.

#####

Warning: The parameters in SDR must agree
with those in the decimate command.

We suggest that FFTWINDOW = UNIFORM be used
as only this has been debugged and tested.
This amounts to not using a window and
is satisfactory provided the input signal
coincides with a bin.

The SDR calculation determines the signal
level SIG, the total distortion level DIST,
the noise level NOIS, and the harmonic level
HARM. DC is excluded from the calculation.

$$\text{DIST} = \text{NOIS} + \text{HARM}$$

Since very few bins (of the frequency spectrum)
are typically available, the noise in the
location of the harmonics is estimated by
averaging the noise on either side of the
harmonics and subtracting it from the total
signal level at the harmonic bins. The
FFTWINDOW used determines the signal spread
and the number of bins over which the signal
levels should be determined.

SCINT - INTEGRATOR

Usage:

SCINT HELP

: This Message

SCINT n TYPE=SWITCH n1 n2 n3 SAME=nnn1

: switched capacitor integrator number "n" with
 : positive input node "n1" and negative input
 : node "n2" and output node "n3".
 : option-> previously defined integrator number
 : "nnn1" uses the same data table

SCINT n TYPE=ANALOG DIM=nnn1 n1 n2 SAME=nnn2

: analog integrator number "n" with table
 : DIMENSION "nnn1" (excluding input) and input
 : node "n1" and output node "n2".
 : option-> previously defined integrator number
 : "nnn2" uses the same data table

WARNING:: Since arbitrary circuit topology has
 not been completely implemented, the ANALOG
 integrator should not be used.

#####

STOP

Usage:

STOP HELP

: This Message

STOP

: End ZSIM session

#####

TABLE

Usage:

TABLE HELP

: This Message

TABLE SCINT n file

: "file" is the input file which contains
 : the TABLE that describes the switched-
 : capacitor integrator number "n".

TABLE QUANT n file

: "file" is the input file which contains
 : the TABLE that describes the quantizer
 : number "n".

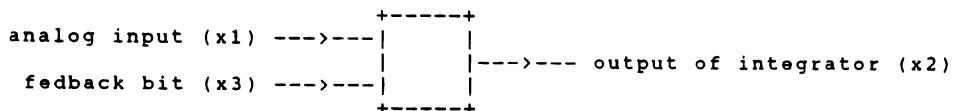
NOTE: quantizer table not yet operational!

#####

6. GENERATING YOUR OWN TABLE

Currently tables can only be used to describe the characteristics of switched capacitor integrators.

Integrator



x1 and x2 can be any real value
x3 must be an integer

This is described by the difference equation

$$x2(n) = x2(n-1) + G * (x1(n) - x3(n-1))$$

where n and n-1 refer to cycle numbers and G is gain.

format of table:

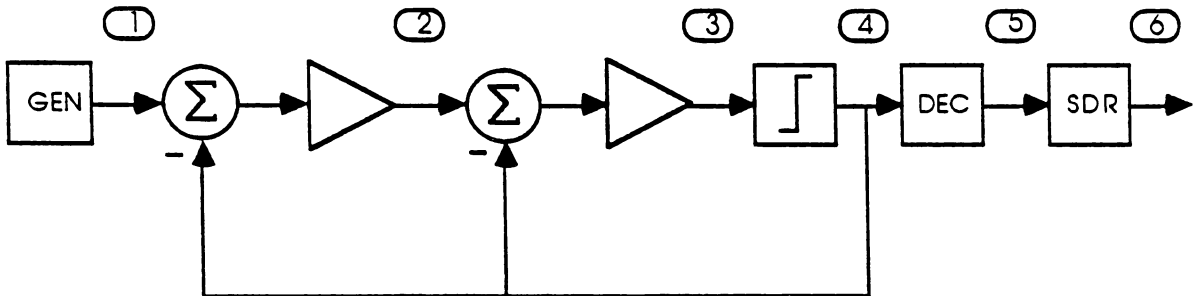
comments:

* comment lines

x1(1) ... x1(i) ... x1(I)	! This is the input to the ! integrator at the current ! cycle. Values of x1 are ! in monotonic order.
x2(1) ... x2(j) ... x2(J)	! This is the output of the ! integrator at the previous ! cycle. Values of x2 are in ! monotonic order.
x3(1) ... x3(k) ... x3(K)	! This is the binary feedback ! bit for the integrator at ! the current cycle. ! Values of x3 are in monotonic ! order and must be integers.
begin	! instruction initiating table ! ! The tables are arranged so ! that x3 varies slowest, ! then x2 and x1 varies the ! fastest each line of ! the table corresponds to ! constant x2 and x3 and ! the numbers on a line are ! for the output x2 at ! the current cycle for ! each value of x1.

8. EXAMPLE

Table based simulation of a second-order Delta-Sigma Modulator clocked at 1.024 MHz with an input signal of 1 kHz. The SDR is calculated for voiceband. Circuit:



Input files: example_circuit, TABLE1, TABLE2, VB.CAS
 Output files: example_output, NODE4, NODE5, NODE6
 These files are printed below together with a keyboard transcript of the session.

KEYBOARD TRANSCRIPT (EXAMPLE)

```
% zsim
  ZSIM   version ZSIM0a1
  Nonlinear Z Domain Simulator
  Brauns, Steer, Ardalán, ECE Dept
  North Carolina State University

* read example_circuit

"example_circuit          "
"example_output          "

Decimation Parameters From File: " VB.CAS
32768 points decimated to 256
Signal-to-Distortion Ratio (dB)    54.3512
Signal-to-Noise Ratio (dB)        55.2068
Signal-to-Harmonic Distortion Ratio (dB) 61.8269
  DUMP file: "NODE4          "
    no. values output = 100 node number = 4
  DUMP file: "NODE5          "
    no. values output = 256 node number = 5
  DUMP file: "NODE6          "
    no. values output = 128 node number = 6
, ? replacing old generator ?

Decimation Parameters From File: " VB.CAS
32768 points decimated to 256
Signal-to-Distortion Ratio (dB)    63.6741
Signal-to-Noise Ratio (dB)        63.8709
Signal-to-Harmonic Distortion Ratio (dB) 77.2103

%
```

INPUT FILE: example_circuit

```
* file: example_circuit
print parameters on
echo on
write example_output
histogram
```

```
* -----
*          EXAMPLE SIMULATION
*          Perform one table method simulation and one difference equation
*          simulation. Tables are derived by difference equation.
* -----
```

```
* set up environment: sampling frequency and the number of clock cycles
environ freq=1.024 MHZ
environ cycles=32768
```

```
* since the topology is not completely automatic, specify that we are
* simulating a second order circuit
circuit=2
```

```
* describe generator: the level is with respect to 2.5 V
* we could have used      AMP = 0.079
gen 1 1 type=sine freq=1000 amp=-30 db wrt 2.5
```

```
* describe the first switched capacitor integrator with tables
scint 1 type=switch 1 4 2
table scint 1 table1
```

```
* describe the second switched capacitor integrator with tables
scint 2 type=switch 2 4 3
table scint 2 table2
```

```
* describe the quantizer as ideal; output is +1 or -1
quant 1 3 4 threshold=0 type=nonhys
```

```
* initialize node values - not completely necessary by this will
* eliminate some warning messages
init node 1 0
init node 2 0
init node 3 0
init node 4 1

* now start the simulation
simulate

* *****
*   VOICEBAND PERFORMANCE   parabolic decimation weighting
*                           sdr calculated using 256 baseband bins
*                           sdr calculated with equalization
*                           sdr ignores first 128 transient bins

* input: -30 dB wrt 2.5 V

decimate 4 5 type=gendec number=32768 window=para taps=128 intdec=32
          bbanddec=4 file=vb.cas filter=cascade

sdr 5 6 type=calsdr npoints=256 nskip=128 nfft=128 fftwin=unif
      decwin=para fb=4000 fs=8000 taps=128 signal=1000

* print out first 100 bits of bitstream
dump node4 4 100

* print out the baseband signal
dump node5 5 256

* print out the spectrum for the second set of 128 baseband points
dump node6 6 128

* -----
* now simulate with difference equations for input
* signal level of -20 dB wrt 2.5 V
gen 1 1 type=sine freq=1000 amp=-20 dB wrt 2.5
eq gain1=0.1 gain2=0.5 sat=1.5 delta1=2.5 delta2=0.25

init node 1 0
init node 2 0
init node 3 0
init node 4 1

desimulate
```

```

decimate 4 5 type=gendec number=32768 window=para taps=128
          intdec=32 bbanddec=4 file=vb.cas filter=cascade

sdr 5 6 type=calsdr npoints=256 nskip=128 nfft=128 fftwin=unif
      decwin=para fb=4000 fs=8000 taps=128 signal=1000

stop

```

INPUT FILE: TABLE1

```

* INTEGRATOR #1
* Integrator gain      : 0.10000
* Switch voltage      : 2.50000
begin
x1= -2.50 -1.75 -1.00 -0.50 -0.10 0.10 0.50 1.00 1.75 2.50
x2= -0.90 -0.70 -0.50 -0.30 -0.10 0.10 0.30 0.50 0.70 0.90
x3= -1      1
-0.900 -0.825 -0.750 -0.700 -0.660 -0.640 -0.600 -0.550 -0.475 -0.400
-0.700 -0.625 -0.550 -0.500 -0.460 -0.440 -0.400 -0.350 -0.275 -0.200
-0.500 -0.425 -0.350 -0.300 -0.260 -0.240 -0.200 -0.150 -0.075 0.000
-0.300 -0.225 -0.150 -0.100 -0.060 -0.040 0.000 0.050 0.125 0.200
-0.100 -0.025 0.050 0.100 0.140 0.160 0.200 0.250 0.325 0.400
0.100 0.175 0.250 0.300 0.340 0.360 0.400 0.450 0.525 0.600
0.300 0.375 0.450 0.500 0.540 0.560 0.600 0.650 0.725 0.800
0.500 0.575 0.650 0.700 0.740 0.760 0.800 0.850 0.900 0.900
0.700 0.775 0.850 0.900 0.900 0.900 0.900 0.900 0.900 0.900
0.900 0.900 0.900 0.900 0.900 0.900 0.900 0.900 0.900 0.900
-0.900 -0.900 -0.900 -0.900 -0.900 -0.900 -0.900 -0.900 -0.900 -0.900
-0.900 -0.900 -0.900 -0.900 -0.900 -0.900 -0.900 -0.850 -0.775 -0.700
-0.900 -0.900 -0.850 -0.800 -0.760 -0.740 -0.700 -0.650 -0.575 -0.500
-0.800 -0.725 -0.650 -0.600 -0.560 -0.540 -0.500 -0.450 -0.375 -0.300
-0.600 -0.525 -0.450 -0.400 -0.360 -0.340 -0.300 -0.250 -0.175 -0.100
-0.400 -0.325 -0.250 -0.200 -0.160 -0.140 -0.100 -0.050 0.025 0.100
-0.200 -0.125 -0.050 0.000 0.040 0.060 0.100 0.150 0.225 0.300
0.      0.075 0.150 0.200 0.240 0.260 0.300 0.350 0.425 0.500
0.200 0.275 0.350 0.400 0.440 0.460 0.500 0.550 0.625 0.700
0.400 0.475 0.550 0.600 0.640 0.660 0.700 0.750 0.825 0.900
done

```

INPUT FILE: TABLE2

```

* INTEGRATOR #2
* Integrator gain      : 0.500000
* Switch voltage      : 0.250000
begin
x1= -0.70 -0.50 -0.30 -0.10  0.10  0.30  0.50  0.70
x2= -1.50 -1.30 -0.80 -0.40 -0.10  0.10  0.40  0.80  1.30  1.50
x3= -1.0  1.0
-1.5000 -1.5000 -1.5000 -1.4250 -1.3250 -1.2250 -1.1250 -1.0250
-1.5000 -1.4250 -1.3250 -1.2250 -1.1250 -1.0250 -0.9250 -0.8250
-1.0250 -0.9250 -0.8250 -0.7250 -0.6250 -0.5250 -0.4250 -0.3250
-0.6250 -0.5250 -0.4250 -0.3250 -0.2250 -0.1250 -0.0250  0.0750
-0.3250 -0.2250 -0.1250 -0.0250  0.0750  0.1750  0.2750  0.3750
-0.1250 -0.0250  0.0750  0.1750  0.2750  0.3750  0.4750  0.5750
 0.1750  0.2750  0.3750  0.4750  0.5750  0.6750  0.7750  0.8750
 0.5750  0.6750  0.7750  0.8750  0.9750  1.0750  1.1750  1.2750
 1.0750  1.1750  1.2750  1.3750  1.4750  1.5000  1.5000  1.5000
 1.2750  1.3750  1.4750  1.5000  1.5000  1.5000  1.5000  1.5000
-1.5000 -1.5000 -1.5000 -1.5000 -1.5000 -1.4750 -1.3750 -1.2750
-1.5000 -1.5000 -1.5000 -1.4750 -1.3750 -1.2750 -1.1750 -1.0750
-1.2750 -1.1750 -1.0750 -0.9750 -0.8750 -0.7750 -0.6750 -0.5750
-0.8750 -0.7750 -0.6750 -0.5750 -0.4750 -0.3750 -0.2750 -0.1750
-0.5750 -0.4750 -0.3750 -0.2750 -0.1750 -0.0750  0.0250  0.1250
-0.3750 -0.2750 -0.1750 -0.0750  0.0250  0.1250  0.2250  0.3250
-0.0750  0.0250  0.1250  0.2250  0.3250  0.4250  0.5250  0.6250
 0.3250  0.4250  0.5250  0.6250  0.7250  0.8250  0.9250  1.0250
 0.8250  0.9250  1.0250  1.1250  1.2250  1.3250  1.4250  1.5000
 1.0250  1.1250  1.2250  1.3250  1.4250  1.5000  1.5000  1.5000
done

```

INPUT FILE: VB.CAS

(This file describes the coefficients for the
cascade filter)
(This file was produced by a program external
to ZSIM)

```

5
1.000000 0.0000000E+00
-1.347701 1.000000
-1.253062 1.000000
-0.9458774 0.9999999
8.8302962E-02 0.9999999
-1.530978 0.9652057
-1.513509 0.8840800
-1.517837 0.7729724
-1.531974 0.6509438
-0.7699454 0.0000000E+00
32000.00
1909.065000000000

```

OUTPUT FILE: example_output

```
"example_output"
" Mon Oct 5 08:55:05 1987"
```

```
ZSIM version ZSIM0a2
Nonlinear Z Domain Simulator
Brauns, Steer, Ardalan, ECE Dept
North Carolina State University
```

```
* histogram
* * -----
* *           EXAMPLE SIMULATION
* *           Perform one table method simulation and one difference equation
* *           simulation. Tables are derived by difference equation.
* * -----
* * set up environment: sampling frequency and the number of clock cycles
* environ freq=1.024 MHZ
* environ cycles=32768
* * since the topology is not completely automatic, specify that we are
* * simulating a second order circuit
* circuit=2
* * describe generator: the level is with respect to 2.5 V
* * we could have used AMP = 0.079
* gen 1 1 type=sine freq=1000 amp=-30 db wrt 2.5
* * describe the first switched capacitor integrator with tables
* scint 1 type=switch 1 4 2
* table scint 1 table1
T * INTEGRATOR #1
T * Integrator gain : 0.10000
T * Switch voltage : 2.50000
T begin
T x1= -2.50 -1.75 -1.00 -0.50 -0.10 0.10 0.50 1.00 1.75 2.50
T x2= -0.90 -0.70 -0.50 -0.30 -0.10 0.10 0.30 0.50 0.70 0.90
T x3= -1 1
T -0.900 -0.825 -0.750 -0.700 -0.660 -0.640 -0.600 -0.550 -0.475 -0.400
T -0.700 -0.625 -0.550 -0.500 -0.460 -0.440 -0.400 -0.350 -0.275 -0.200
T -0.500 -0.425 -0.350 -0.300 -0.260 -0.240 -0.200 -0.150 -0.075 0.000
T -0.300 -0.225 -0.150 -0.100 -0.060 -0.040 0.000 0.050 0.125 0.200
T -0.100 -0.025 0.050 0.100 0.140 0.160 0.200 0.250 0.325 0.400
T 0.100 0.175 0.250 0.300 0.340 0.360 0.400 0.450 0.525 0.600
T 0.300 0.375 0.450 0.500 0.540 0.560 0.600 0.650 0.725 0.800
T 0.500 0.575 0.650 0.700 0.740 0.760 0.800 0.850 0.900 0.900
T 0.700 0.775 0.850 0.900 0.900 0.900 0.900 0.900 0.900 0.900
```

```

T   0.900  0.900  0.900  0.900  0.900  0.900  0.900  0.900  0.900  0.900
T  -0.900 -0.900 -0.900 -0.900 -0.900 -0.900 -0.900 -0.900 -0.900 -0.900
T  -0.900 -0.900 -0.900 -0.900 -0.900 -0.900 -0.900 -0.850 -0.775 -0.700
T  -0.900 -0.900 -0.850 -0.800 -0.760 -0.740 -0.700 -0.650 -0.575 -0.500
T  -0.800 -0.725 -0.650 -0.600 -0.560 -0.540 -0.500 -0.450 -0.375 -0.300
T  -0.600 -0.525 -0.450 -0.400 -0.360 -0.340 -0.300 -0.250 -0.175 -0.100
T  -0.400 -0.325 -0.250 -0.200 -0.160 -0.140 -0.100 -0.050  0.025  0.100
T  -0.200 -0.125 -0.050  0.000  0.040  0.060  0.100  0.150  0.225  0.300
T   0.     0.075  0.150  0.200  0.240  0.260  0.300  0.350  0.425  0.500
T   0.200  0.275  0.350  0.400  0.440  0.460  0.500  0.550  0.625  0.700
T   0.400  0.475  0.550  0.600  0.640  0.660  0.700  0.750  0.825  0.900
T done

```

```

* * describe the second switched capacitor integrator with tables
* scint 2 type=switch 2 4 3
* table scint 2 table2
T   * INTEGRATOR #2
T   * Integrator gain   : 0.500000
T   * Switch voltage   : 0.250000
T begin
T x1= -0.70  -0.50  -0.30  -0.10  0.10  0.30  0.50  0.70
T x2= -1.50  -1.30  -0.80  -0.40  -0.10  0.10  0.40  0.80  1.30  1.50
T x3= -1.0   1.0
T  -1.5000 -1.5000 -1.5000 -1.4250 -1.3250 -1.2250 -1.1250 -1.0250
T  -1.5000 -1.4250 -1.3250 -1.2250 -1.1250 -1.0250 -0.9250 -0.8250
T  -1.0250 -0.9250 -0.8250 -0.7250 -0.6250 -0.5250 -0.4250 -0.3250
T  -0.6250 -0.5250 -0.4250 -0.3250 -0.2250 -0.1250 -0.0250  0.0750
T  -0.3250 -0.2250 -0.1250 -0.0250  0.0750  0.1750  0.2750  0.3750
T  -0.1250 -0.0250  0.0750  0.1750  0.2750  0.3750  0.4750  0.5750
T   0.1750  0.2750  0.3750  0.4750  0.5750  0.6750  0.7750  0.8750
T   0.5750  0.6750  0.7750  0.8750  0.9750  1.0750  1.1750  1.2750
T   1.0750  1.1750  1.2750  1.3750  1.4750  1.5000  1.5000  1.5000
T   1.2750  1.3750  1.4750  1.5000  1.5000  1.5000  1.5000  1.5000
T  -1.5000 -1.5000 -1.5000 -1.5000 -1.5000 -1.4750 -1.3750 -1.2750
T  -1.5000 -1.5000 -1.5000 -1.4750 -1.3750 -1.2750 -1.1750 -1.0750
T  -1.2750 -1.1750 -1.0750 -0.9750 -0.8750 -0.7750 -0.6750 -0.5750
T  -0.8750 -0.7750 -0.6750 -0.5750 -0.4750 -0.3750 -0.2750 -0.1750
T  -0.5750 -0.4750 -0.3750 -0.2750 -0.1750 -0.0750  0.0250  0.1250
T  -0.3750 -0.2750 -0.1750 -0.0750  0.0250  0.1250  0.2250  0.3250
T  -0.0750  0.0250  0.1250  0.2250  0.3250  0.4250  0.5250  0.6250
T   0.3250  0.4250  0.5250  0.6250  0.7250  0.8250  0.9250  1.0250
T   0.8250  0.9250  1.0250  1.1250  1.2250  1.3250  1.4250  1.5000
T   1.0250  1.1250  1.2250  1.3250  1.4250  1.5000  1.5000  1.5000
T done
* * describe the quantizer as ideal; output is +1 or -1
* quant 1 3 4 threshold=0 type=nonhys

```

```

* * initialize node values - not completely necessary by this will
* * eliminate some warning messages
* init node 1 0
* init node 2 0
* init node 3 0
* init node 4 1
* * now start the simulation
* simulate

```

```

*****
INTEGRATOR OUTPUT

```

```

Integrator # 1
  Input max value = 7.90569e-02      Input min value = -7.90569e-02
  Output max value = 0.425817      Output min value = -0.427949

Integrator # 2
  Input max value = 0.425817      Input min value = -0.427949
  Output max value = 0.334090      Output min value = -0.337891

```

```

*****

```

```

HISTOGRAM FOR TABLE 1

```

```

x1 values: 10
x2 values: 10
x3 values: 2

```

```

0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 916 916 0 0 0 0
0 0 0 0 9721 9721 0 0 0 0
0 0 0 0 15164 15164 0 0 0 0
0 0 0 0 6663 6663 0 0 0 0
0 0 0 0 304 304 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 294 294 0 0 0 0
0 0 0 0 7563 7563 0 0 0 0
0 0 0 0 14927 14927 0 0 0 0
0 0 0 0 8821 8821 0 0 0 0
0 0 0 0 1163 1163 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0

```

HISTOGRAM FOR TABLE 2

x1 values: 8
 x2 values: 10
 x3 values: 2

```

0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 297 6158 8703 2858 16 0
0 0 297 7552 14924 8832 1163 0
0 0 0 1394 6221 5974 1147 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 891 6257 6345 979 0 0 0
0 916 9718 15175 6666 293 0 0
0 25 3461 8830 5687 293 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

```

```

* * *****
* * VOICEBAND PERFORMANCE      parabolic decimation weighting
* *                               sdr calculated using 256 baseband bins
* *                               sdr calculated with equalization
* *                               sdr ignores first 128 transient bins
* * input: -30 dB wrt 2.5 V
* * decimate 4 5 type=gendec number=32768 window=para taps=128
* *       intdec=32 bbanddec=4 file=vb.cas filter=cascade
* *       Decimation Parameters From File: " VB.CAS "
* *       32768 points decimated to 256
* * sdr 5 6 type=calsdr npoints=256 nskip=128 nfft=128 fftwin=unif
* *       decwin=para fb=4000 fs=8000 taps=128 signal=1000
* * baseband frequency: 4000.00 Hz.; sampling frequency: 8000.00 Hz.
* * 128 total fft bins
* * 64 bins in baseband; signal on bin 17
* * 3 harmonics in baseband

```

FFT RESULTS

Signal Strength : 1.9999918172406 Total Distortion : 7.3436389151260d-06
Total Noise : 6.0304125187815d-06 Total Harmonics : 1.3132267903341d-06

Signal-to-Distortion (dB): 54.3512
Signal-to-Noise (dB): 55.2068
Signal-to-Harmonic (dB): 61.8269

* * print out first 100 bits of bitstream
* dump node4 4 100
 DUMP file: "NODE4" "
 no. values output = 100 node number = 4
* * print out the baseband signal
* dump node5 5 256
 DUMP file: "NODE5" "
 no. values output = 256 node number = 5
* * print out the spectrum for the second set of 128 baseband points
* dump node6 6 128
 DUMP file: "NODE6" "
 no. values output = 128 node number = 6

* * -----
* * now simulate with difference equations for input
* * signal level of -20 dB wrt 2.5 V
* gen 1 1 type=sine freq=1000 amp=-20 dB wrt 2.5
 ? replacing old generator ?
* eq gain1=0.1 gain2=0.5 sat=1.5 delta1=2.5 delta2=0.25
* init node 1 0
* init node 2 0
* init node 3 0
* init node 4 1
* simulate

INTEGRATOR OUTPUT

Integrator # 1
 Input max value = 0.250000 Input min value = -0.250000
 Output max value = 0.461255 Output min value = -0.464121

Integrator # 2
 Input max value = 0.461255 Input min value = -0.464121
 Output max value = 0.352487 Output min value = -0.354863

```
decimate 4 5 type=gendec number=32768 window=para taps=128
      intdec=32 bbanddec=4 file=vb.cas filter=cascade
Decimation Parameters From File: " VB.CAS "
      32768 points decimated to 256
sdr 5 6 type=calsdr npoints=256 nskip=128 nfft=128 fftwin=unif
      decwin=para fb=4000 fs=8000 taps=128 signal=1000
baseband frequency: 4000.00 Hz.; sampling frequency: 8000.00 Hz.
128 total fft bins
64 bins in baseband; signal on bin 17
3 harmonics in baseband
```

FFT RESULTS

```
Signal Strength : 1.9999991705350      Total Distortion : 8.5826086590927d-07
Total Noise      : 8.2024190071623d-07  Total Harmonics  : 3.8018992284193d-08
```

```
Signal-to-Distortion (dB): 63.6741
Signal-to-Noise      (dB): 63.8709
Signal-to-Harmonic   (dB): 77.2103
```

* stop

```

C.....
C
C      AGEN
C
C      PURPOSE:
C          This subroutine simulates the generator
C
C      PARAMETERS:
C          i      : generator number
C          node   : circuit value at node i
C          cycle  : clock cycle of simulation
C          gen    : information array for generator
C          f      : frequency of generator in Hz <output>
C.....
C      subroutine AGEN(i,node,cycle,gen,f)
C
C          integer i,cycle,nocyc,block,tsadd
C          integer mgen,pgen,circuit
C          common/blkmax/mgen,pgen,mscint,pscint
C          common/parmax/mxblk,mxdim,mxdim3,mxtele,mxnode,mxicyc,mxtype,mxnumx
C          common/funblk/block,period,nocyc,tsadd,pi,freqpt,circuit
C          real gen(mgen,pgen),node(mxnode),out,noise(10,3)
C          common/nose/noise
C          character*20 str
C          logical kf,fin,prt,ofile,rfile
C          common/freerd/nwrite,ntype,aval,ival,str,ich,kf,fin,prt,ofile,rfile
C
C      BRANCH TO GENERATOR TYPE
C
C          goto(100,200,300,400)int(gen(i,1))
C
C      DC GENERATOR
C
C      100    out=gen(i,3)
C            f=0.
C            goto 1000
C
C      SINE GENERATOR
C
C      200    f=gen(i,2)
C            phase=gen(i,6)
C            angle=2.0*pi*f*period*(cycle-1) + phase
C            out=gen(i,5)+gen(i,3)*sin(angle)
C            goto 1000
C
C      RAMP GENERATOR
C
C      300    if((cycle-1)*period.ge.gen(i,5))then
C            out=gen(i,3)
C            else
C            out=gen(i,3)/gen(i,5)*period*(cycle-1)
C            endif
C            f=0.
C            goto 1000
C
C      STEP GENERATOR
C
C      400    if((cycle-1)*period.ge.gen(i,5))then
C            out=gen(i,3)
C            else
C            out=gen(i,2)
C            endif
C            f=0.
C
C      ADD NOISE IF SPECIFIED
C
C      1000   iii=int(gen(i,4))
C            if(int(noise(iii,1)).eq.1)then
C            call gauss(noise(iii,3),noise(iii,2),y)
C            out=out+y
C            endif
C
C      DEFINE OUTPUT OF GENERATOR
C
C

```

```

node(int(gen(i,4))=out
return
end
C.....
C
C   AQUANT
C
C   PURPOSE:
C       This subroutine simulates the quantizer.
C
C   PARAMETERS:
C       i       : quantizer number
C       node    : present node values
C       nodep   : previous node values
C       cycle   : clock cycle of simulation
C       quant   : information array for quantizers
C
C   SUBROUTINES CALLED:
C       gauss
C.....
C       subroutine AQUANT(i,node,nodep,cycle,quant)
C
C       integer i,mquant,pquant,cycle
C       common/blkmax/mgen,pgen,mscint,pscint,mquant,pquant
C       common/parmax/mxblk,mxdim,mxdim3,mxtele,mxnode,mxicyc,mxtype,mxnumx
C       real quant(mquant,pquant),node(mxnode),nodep(mxnode),in,out
C       character*20 str
C       logical kf,fin,prt,ofile,rfile
C       common/freerd/nwrite,ntype,aval,ival,str,ich,kf,fin,prt,ofile,rfile
C
C       in=node(int(quant(i,3)))
C       out=node(int(quant(i,4)))
C
C   C   BRANCH TO QUANTIZER TYPE
C
C       goto(10,50)int(quant(i,1))
C
C   C   QUANTIZER WITH NO HYSTERESIS
C
C   10   if(in.ge.quant(i,2)+quant(i,6))then
C         out=1
C       else if(in.le.quant(i,2)-quant(i,6))then
C         out=-1
C       else
C         out=0
C       endif
C       goto 1000
C
C   C   QUANTIZER WITH HYSTERESIS
C
C   50   write(ntype,*)' ? hysteresis quantizer not yet programmed ?'
C
C   C   STORE OUTPUT AS PRESENT & PREVIOUS VALUE
C
C   1000 node(int(quant(i,4))=out
C         nodep(int(quant(i,4))=out
C         return
C         end
C.....
C
C   ASCINT
C
C   PURPOSE:
C       This subroutine simulates the integrator
C
C   PARAMETERS:
C       i       : integrator number
C       node    : present node values
C       nodep   : previous node values
C       scint   : information array for integrator
C       ptable  : pointer array to integrator table
C       tables  : vector containing integrator table
C
C   SPECIAL VARIABLES:

```

```

C          x1 : present input to integrator
C          x2 : previous value of integrator output
C          x3 : previous value for switched-reference input
C
C          SUBROUTINES CALLED:
C          polate
C
C.....
C          subroutine ASCINT(i,cycle,node,nodep,scint,ptable,tables)
C
C          integer block,tsadd,nocyc,fbindx,mscint,pscint,circuit,out,cycle
C          real x1,x2,x3
C          common/blkmax/mgen,pgen,mscint,pscint,mquant,pquant
C          common/parmax/mxblk,mxdim,mxdim3,mxtele,mxnode,mxicyc,mxtype,mxnumx
C          common/funblk/block,period,nocyc,tsadd,pi,freqpt,circuit
C          integer ptable(mxtype,mxblk,mxdim3)
C          real scint(mscint,pscint),tables(mxtele),noise(10,3)
C          real node(mxnode),nodep(mxnode),maxout(5),minout(5)
C          real maxin(5),minin(5)
C          common/nose/noise
C          character*20 str
C          logical kf,fin,prt,ofile,rfile
C          common/freord/nwrite,ntype,aval,ival,str,ich,kf,fin,prt,ofile,rfile
C          common/intmax/maxout,minout,maxin,minin
C
C          fbindx=int(scint(i,2))
C          x2=0.
C          x3=0.
C
C          DEFINE SIGNALS FOR ANALOG INTEGRATOR
C
C          if(int(scint(i,1)).eq.1)then
C            out=int(scint(i,4))
C            x1=node(int(scint(i,3)))
C            if(ptable(fbindx,i,1).eq.2)then
C              if(cycle-1.eq.0)then
C                maxout(i)=0.0
C                minout(i)=0.0
C                maxin(i)=0.0
C                minin(i)=0.0
C                x2=nodep(out)
C                x3=nodep(int(scint(i,4)))
C              else
C                x2=nodep(out)
C                x3=nodep(int(scint(i,4)))
C              endif
C            endif
C
C          DEFINE SIGNALS FOR SWITCHED REFERENCE INTEGRATOR
C
C          else if(scint(i,1).eq.2)then
C            out=int(scint(i,5))
C            x1=node(int(scint(i,3)))
C            if(cycle-1.eq.0)then
C              maxout(i)=0.
C              minout(i)=0.
C              maxin(i)=0.0
C              minin(i)=0.0
C              x2=nodep(out)
C              x3=nodep(int(scint(i,4)))
C            else
C              x2=nodep(out)
C              x3=nodep(int(scint(i,4)))
C            endif
C          endif
C
C          INTERPOLATE SIGNAL VALUES TO OBTAIN OUTPUT
C
C          call POLATE(ptable(fbindx,i,1),fbindx,i,ptable,tables,node(out),
C          +             x1,x2,x3)
C
C          ADD NOISE IF SPECIFIED
C
C          if(int(noise(out,1)).eq.1)then
C            call gauss(noise(out,3),noise(out,2),y)

```

```

        node(out)=node(out)+y
    endif
C
C STORE PRESENT VALUES AS PAST VALUES FOR NEXT CLOCK CYCLE
C
        nodep(out)=node(out)
C
C DETERMINE MAXIMUM, MINIMUM INTEGRATOR OUTPUT
C
        if(node(out).gt.maxout(i))maxout(i)=node(out)
        if(node(out).lt.minout(i))minout(i)=node(out)
        if(x1.gt.maxin(i))maxin(i)=x1
        if(x1.lt.minin(i))minin(i)=x1
        return
    end
C.....
C
C CALFBW
C
C PURPOSE:
C     To calculate difference equation coefficients that model
C     finite gain-bandwidth of operational amplifiers.
C
C.....
        subroutine calfbw(ft,fs,gamma,a,b,c)
        real ft,fs,gamma,a,b,c,alpha1,alpha2,gamma2,pi
        data pi/3.1415926/

        gamma2=1.0/(1.0+gamma)
        alpha1=pi*ft*gamma2/fs
        alpha2=pi*ft/fs
        a=1.0-exp(-alpha1)
        b= gamma2*exp(-alpha1)*(1.0-exp(-alpha2))
        c= gamma2/exp(alpha1+alpha2)
        return
    end
C.....
C
C CALSDR
C
C     This subroutine calculates the sdr of an input baseband signal
C
C PARAMETERS:
C
C     b      : array containing baseband signal          <input>
C     spec   : array containing calculated spectrum in dB <output>
C     nfft   : Number of points to do fft on           <input>
C     nskip  : number of points to skip
C     sdr    : signal to distortion ratio               <output>
C     snr    : signal to noise ratio                   <output>
C     sthd   : signal to harmonic distortion ratio     <output>
C     nn     : number of data points in b              <input>
C     ntype  : logical unit number for terminal output
C     nwrite : logical unit number for file output
C     iwind  : window to be used in FFT
C             iwind = 0  uniform
C             iwind = 1  hamming
C             iwind = 2  blackman-harris (4 sample)
C     iw     : decimation window type required for equalization
C             iw = 0    do not equalize
C                   irregardless of type of
C                   decimation windowing.
C             iw = 1    uniform window
C             iw = 2    triangular window
C             iw = 3    parabolic window
C     fb     : baseband frequency
C     sampfq : baseband sampling frequency
C     nfast  : number of weights in decimator (number of fast cycles)
C
C VARIABLES:
C
C     xx     : real array of length at least 2*(2**m)
C     x      : complex array of length at least 2**m
C     spread : the number of fft bins on each side of the main signal
C             for a specific window type

```

```

c
c.....
      subroutine calcdr(b,spec,nn,nfft,nskip,sdr,snr,sthdp,prnt,ntype,
+          nwrite,iwind,iw,fb,sampfq,nfast,sigfq)

CGTB_START
      real sigfq,period,fb,sampfq,pi
      integer iw,nwrite,ntype,nfast
      logical prt
CGTB_END
      integer block,nocyc,circuit,spread,tsadd,freqpt
      common/funblk/block,period,nocyc,tsadd,pi,freqpt,circuit
      complex x(32768)
      integer nfft,nskip,nn
      real xx(65536),y(32768),spec(32768),b(32768),equal(32768)
      logical fftyes
      common/fourie/snoise,sig,totno1,totno2,totno3,snr1,snr2,snr3,fftyes
      equivalence (x,xx)

      do 10 i=1,nn
          y(i)=b(i)
10      continue
CMBS_START
      if(nskip.ne.0)then
          do 106 i=nskip+1,nn
              y(i-nskip)=y(i)
c          write(ntype,*)'y(i)',y(i)
106      continue
          endif
CMBS_END
c
c      put signal through window
c
c          tdeng=0.0
c uniform
      if(iwind.eq.0)then
          spread=0
          do 108 j=1,nfft
              xx(2*j-1)=y(j)
              xx(2*j)=0.0
              tdeng=tdeng+xx(2*j-1)**2
108      continue
c hamming
          elseif(iwind.eq.1)then
              spread=1
              do 109 j=1,nfft
                  wd=0.54-0.46*cos(2.0*pi*(j-1)/(nfft-1))
                  xx(2*j-1)=y(j)*wd
                  xx(2*j)=0.0
                  tdeng=tdeng+xx(2*j-1)**2
109      continue
c 4-sample blackman-harris (-74dB)
          elseif(iwind.eq.2)then
              spread=2
              a0=.40217
              a1=.49703
              a2=.09392
              a3=.00183
              do 110 j=1,nfft
                  rrad=2.0*pi*(j-1)/(nfft-1)
                  wd=a0-a1*cos(rrad)+a2*cos(rrad*2)-a3*cos(rrad*3)
                  xx(2*j-1)=y(j)*wd
                  xx(2*j)=0.0
                  tdeng=tdeng+xx(2*j-1)**2
110      continue
          endif
c
c calculate equalization weights
c
CGTB_START
      df=sampfq/nfft
CGTB_END
      fs=1.0/period
      do 22 i=1,nfft
          if(iw .eq. 0)then

```

```

        equal(i)=1.0
    else
        freq=(i-1)*df
        temp=sinxox(freq,nfast,fs)
        temp=1.0/temp
        if(iw .eq. 1 )equal(i)=temp
        if(iw .eq. 2 )equal(i)=temp**2
        if(iw .eq. 3 )equal(i)=temp**3
    endif
22  continue

    call fft(x,nfft)

    call sdrsub(xx,equal,sdr,snr,sth,nfft,spread,iw,fb,sampfq,
+         period,sigfq,prt,nwrite)

    write(ntype,*)'   Signal-to-Distortion Ratio (dB)           ',sdr
    write(ntype,*)'   Signal-to-Noise Ratio (dB)              ',snr
    write(ntype,*)'   Signal-to-Harmonic Distortion Ratio (dB) ',sth
    fftpts=nfft
    do 900 j=1,nfft
        spec(j)=xx(2*j-1)*xx(2*j-1)+xx(2*j)*xx(2*j)
        if (spec(j) .eq. 0.0) then
            spec(j) = -100.0
        else
            spec(j)=10.0*alog10(spec(j))
        endif
900  continue

c
c put snr outputs into common block
c
    snr1=sdr
    snr2=snr
    snr3=sth
    fftyes=.true.
    return
end

c.....
c
c   SINOX
c
c   PURPOSE:
c       To equalize for decimation filtering equalization ( sinc(x)
c   correction. )
c.....
c
c   function sinxox(f,N,fs)
c       real f
c       real fs,pi
c       integer N
c       data pi/3.14159265/
c
c       if (f .eq. 0.0) then
c           sinxox=1.0
c           return
c       endif
c       fN=N
c       x1=f*fN*pi/fs
c       x2=f*pi/fs
c       sinxox=sin(x1)/sin(x2)/fN
c       return
c       end
c.....
c
c   CASFIL
c
c   PURPOSE:
c       Subroutine implements cascade form IIR digital Filter
c.....
c
c cascade form filter coefficients are read from a file
c The inputs from the file are as follows:
c ns: number of sections

```

```

c  zc(1,i) zc(2,i) i=1 to ns the numerator coefficeients
c  pc(1,i) pc(2,i) i=1 to ns the denominator coefficeients
c  In the z domain:
c
c  xx() is the real array of input samples
c  y() is the array of filtered samples calculated by the subroutine
c  npts is the number of points in the arrays
c
c*****
c  Sasan H Ardalan, CCSP August 2, 1986
c
c*****
      subroutine casfil(xx,y,npts,ird)
      dimension xx(npts),y(npts)
      real zc(2,20),pc(2,20)
      real xs(3),ys(3),ycas(20,2)
      integer i,j,npts,ns,ird

      read(ird,*)ns
      do 33 i=1,ns
33      read(ird,*)zc(1,i),zc(2,i)
      do 34 i=1,ns
34      read(ird,*)pc(1,i),pc(2,i)
      read(ird,*)fs
      read(ird,*)wnorm
c      write(6,*)'IIR Filtering Cascade Form, Sampling Rate:',fs
      do 70 j=1,3
          xs(j)=0.0
          ys(j)=0.0
70      continue
      do 71 i=1,ns
          do 71 j=1,2
71      ycas(i,j)=0.0
          do 80 i=1,npts
              do 50 j=1,ns
                  if (j .eq. 1 )then
                      xs(2)=0.0
                      xs(3)=0.0
                      xs(1)=xx(i)
                      i1=i-1
                      if(i1 .gt. 0)  xs(2)=xx(i1)
                      i2=i-2
                      if(i2 .gt. 0)  xs(3)=xx(i2)
                      endif
                      if(j .gt. 1 ) then
                          do 55 jj=1,3
55      xs(jj)=ys(jj)
                          endif
                          do 56 jj=1,2
56      ys(jj+1)=ycas(j,jj)
                          call cassec(xs,ys,zc(1,j),zc(2,j),pc(1,j),pc(2,j))
                          do 57 jj=1,2
57      ycas(j,jj)=ys(jj)
50      continue
          y(i)=ys(1)/wnorm
80      continue
          return
      end

      subroutine cassec(xs,ys,b1,b2,a1,a2)
      dimension xs(3),ys(3)
      ys(1)=xs(1)+b1*xs(2)+b2*xs(3)-a1*ys(2)-a2*ys(3)
      return
      end
c.....
c
c  COMPAR
c
c  PURPOSE:
c      This subroutine stores information about quantizers.
c
c  PARAMETERS:
c      quant : information array for quantizers
c      qntflg : flag array to acknowledge presence of quantizer
c      ptable : pointer array for tables
c

```

```

C
C STORAGE:   for quantizer number i
C           quant(i,1) : type [1:no hysteresis , 2:hysteresis]
C           quant(i,2) : threshold value
C           quant(i,3) : input node
C           quant(i,4) : output node
C           quant(i,5) : quantizer reference code [set to 3]
C           quant(i,6) : one-sided zero-band value [independent of (i,2)]
C
C SUBROUTINES CALLED:
C           fread
C
C.....
C           subroutine COMPAR(quant,qntflg,ptable)
C
C           integer block,nocyc,tsadd,mquant,pquant,circuit,fbindx
C           common/blkmax/mgen,pgen,mscint,pscint,mquant,pquant
C           common/parmax/mxblk,mxdim,mxdim3,mxtele,mxnode,mxicyc,mxtype,mxnumx
C           common/funblk/block,period,nocyc,tsadd,pi,freqpt,circuit
C           logical qntflg(mquant),replac,done
C           integer ptable(mxtype,mxblk,mxdim3)
C           real quant(mquant,pquant),save(10)
C
C           character*20 str
C           logical kf,fin,prt,ofile,rfile
C           common/freerd/nwrite,ntype,aval,ival,
+           str,ich,kf,fin,prt,ofile,rfile
C
C           call fread
C           if(fin)then
C             write(ntype,*)' ? insufficient quantizer data ?'
C             if(prt)write(nwrite,*)' ? insufficient quantizer data ?'
C             return
C           endif
C           if(str(1:4).eq.'HELP')then
C             write(ntype,50)
C             if(prt)write(nwrite,50)
50           format(
+           ' ##### QUANTIZER #####',/,
+           ' Usage:',/,
+           '   QUANT HELP',/,
+           '   : This Message',/,/,
+           '   QUANT n n1 n2 THRES=xxx1 TYPE=sss1 BAND=xxx2 SAME=nnn1',/,/,
+           '   : Quantizer "n" at input node "n1" and output node "n2"',/,/,
+           '   : has THRESHOLD of "xxx1" and TYPE = NONHysteresis or',/,/,
+           '   : HYSTERESIS.',/,/,
+           '   : options 1.previously defined quantizer number "nnn1"',/,/,
+           '   :           uses the same data table',/,/,
+           '   :           2."xxx2" is a deadzone region where the',/,/,
+           '   :           quantizer is undecisive (output=0).',/,/,
+           '   : NOTE:: option 1 is presently disabled since the',/,/,
+           '   :           quantizer is ideal and not characterized by',/,/,
+           '   :           a table as of now.',/,/,
+           '   :           TYPE must be specified NONHysteresis',/,/,
+           '   :           until a hysteresis table is programmed',/,/,
+           ' #####')
C             return
C           endif
C
C DETERMINE WHICH QUANTIZER IS INPUT
C
C           i=ival
C
C DETERMINE IF MEMORY IS AVAILABLE
C
C           if(block.eq.mxblk)then
C             write(ntype,*)' ? At maximum block capacity ?'
C             return
C           else if(i.gt.mquant)then
C             write(ntype,*)' ? At maximum quantizer capacity ?'
C             return
C           endif
C
C DETERMINE IF QUANTIZER i EXISTS
C
C           replac=.false.

```

```

done=.true.
if(.not.qntflg(i))then
  block=block+1
  qntflg(i)=.true.
else
  write(ntype,*)' ? replacing old quantizer ?'
  replac=.true.
endif
C
C SAVE OLD QUANTIZER INFO IN CASE OF INPUT ERROR
C
  do 5 j=1,pquant
    save(j)=quant(i,j)
  5 continue
C
C READ QUANTIZER INPUT
C
  quant(i,5)=3
  fbindx=int(quant(i,5))
  call fread
  if(fin)goto 100
  quant(i,3)=ival
  call fread
  if(fin)goto 100
  quant(i,4)=ival
  call fread
  if(fin)goto 100
  if(str(1:3).ne.'THR'.and.str(1:3).ne.'thr')goto 200
  call fread
  if(fin)goto 100
  quant(i,2)=aval
  call fread
  if(fin)goto 100
  if(str(1:3).ne.'TYP'.and.str(1:3).ne.'typ')goto 200
  call fread
  if(fin)goto 100
  if(str(1:2).eq.'NO'.or.str(1:2).eq.'no')quant(i,1)=1
  if(str(1:3).eq.'HYS'.or.str(1:3).eq.'hys')quant(i,1)=2
  quant(i,6)=0
  call fread
  if(fin)goto 1000
  if(str(1:3).eq.'BAN'.or.str(1:3).eq.'ban')then
    call fread
    if(fin.or.aval.eq.0.0)goto 100
    quant(i,6)=abs(aval)
  endif
C
C DETERMINE IF HYSTERESIS QUANTIZER HAS SAME CHARACTERISTICS AS ANOTHER ONE
C
  if(int(quant(i,1)).eq.2)then
    ptable(fbindx,i,1)=1
    call fread
    if(.not.fin)then
      if(str(1:1).ne.'S'.and.str(1:1).ne.'s')goto 200
      call fread
      if(fin)goto 100
      if(ival.lt.1.or.ival.gt.mquant)goto 200
      if(.not.qntflg(ival))then
        write(ntype,46)ival
46      format(' ? Quantizer ',i3,' table DOES NOT EXIST ?')
        goto 1000
      endif
      do 10 jj=1,mxdim3
        ptable(fbindx,i,jj)=ptable(fbindx,ival,jj)
10      continue
      endif
    endif
    goto 1000
C
C ERROR MESSAGES
C
100 write(ntype,*)' ? insufficient quantizer data ?'
    if(replac)then
      write(ntype,*)' ? old quantizer still present ?'
    endif

```

```

done=.false.
goto 1000

200  write(ntype,*)' ? illegal quantizer input format ?'
     if(replac)then
       write(ntype,*)' ? old quantizer still present ?'
     endif
done=.false.

c
c  RESTORE OLD QUANTIZER INFO IF INPUT ERROR OCCURS
c
1000  if(.not.done)then
      do 15 j=1,pquant
        quant(i,j)=save(j)
      15  continue
      endif
      return
      end
c.....
c
c  DECIMA
c
c  PURPOSE:
c    To decimate and filter pulse stream
c
c  This program decimates and baseband filters
c  the Sigma Delta Modulation pulse stream
c
c  Inputs to the program include number of samples generated,
c  Decimation filter length and weighting
c  Decimation factor , Baseband Filter Selection,
c
c  Subroutines called:
c    casfil,dffil or nlfil: Base Band Filters
c
c    Arrays
c  y: Array for baseband filtering output and y-axis plotting
c  D: Integer array of decimation filter weights
c  TF: Integer array, Tapped Delay Line for FIR decimation Filt.
c
c    Variables
c  fs: Sigma Delta Modulator Fast Sampling frequency
c  p() : sample value SDM pulse Stream, ipt= +-1
c  iaccum: Decimation filter Accumulator
c
c  M: 2**M is the number of SDM samples at fs sampling rate
c  nf: is the length of the decimation filter
c  idec: is the decimation factor
c
c    Integer Control Variables
c  iw:    determines weights,1=uniform,2=triangular,3=parabolic
c.....
c    subroutine decima(p,bb,M,nbb,fs,fb,
c    #          iw,nf,idec,ibbdec,ifch,filtyp)
c
c  P() contains Delta-Sigma Pulse stream as +-1 integer
c  2**M is the number of Delta-Sigma Samples
c  bb() is the baseband result OUTPUT
c  nbb number of baseband samples OUTPUT
c  fs is fast rate sampling rate for p()
c  fb is the baseband sampling rate OUTPUT
c  iw determines the weighting: 1-Uniform 2-Triangular 3-Parabolic
c  nf number of taps for FIR decimation filter
c  idec is the decimation factor
c  ibbdec is the decimation factor after baseband filtering
c  ifch is the logical number of the file containing baseband
c    IIR filter parameters
c  filtyp is the IIR filter type 0-none 1-direct form
c    2- Normalized Lattice 3- Cascade Form
c
c  nbb=2**M/idec/ibbdec
c  fb=fs/idec/ibbdec
c
c    real bb(1)

```

```

integer p(1)
real xx(4096),y(4096)
integer D(4096),TF(4096)
real fs,fb
integer filtyp
integer iw
integer M,nf,idec
integer ifch
data PI/3.1415926/

c
c  idec is decimation factor
c
c      nd=M/idec
c
c  iw:
c  1 Uniform
c  2 Triangular
c  3 Sinc (3)'
c
c  nf:  Number of Fast Cycles log 2
c
c      nfi=nf-1
c      nfd2=int(nf/2)
c      nfd3=int(nf/3.)+1
c
c  setup weights for decimation filters
c
c      wnorm=0.0
c      do 78 i=1,nf
c      if (iw .eq. 1) then
c      D(i)=1
c      else if (iw .eq. 3) then
c      if(i .le. nfd3) then
c      D(i)=int(i*(i+1)/2)
c      D(nf-i+1)=D(i)
c      else if(i .le. nfd2) then
c      D(i)=int(nfd3*(nfd3+1)/2+(i-nfd3)*(2*nfd3-1-i))
c      D(nf-i+1)=D(i)
c      endif
c      else if (iw .eq. 2) then
c      if(i .lt. nfd2) then
c      D(i)=i
c      else
c      D(i)=nf-i
c      end if
c      end if
c
c  initialized Tapped Delay Line to Zero
c
c      TF(i)=0
c      wnorm=wnorm+D(i)
78  continue
c
c  decimate delta sigma bit stream
c
c      nd1=nd-1
c      jj=0
c      do 79 j=1,nd
c      do 75 ii=1,idec
c      jj=jj+1
c
c  shift new SDM sample into delay array TF
c
c      do 40 k=nf,2,-1
40  TF(k)=TF(k-1)
c      TF(1)=p(jj)
c
c  Filter Prior to Decimation
c
c      iaccum=0
c      do 70 k=1,nf
70  iaccum=iaccum+TF(k)*D(k)
75  continue
c
c  Store Decimated Sample in array y()

```

```

c
c      y(j)=iaccum/wnorm
79      continue
c
c At this point y() contains nd decimated SDM samples
c
c
c Base Band Filter decimated SDM samples
c xx() filtered samples of y()
c
c      FILTERING
c      filtyp:
c      0 no filtering
c      1 direct form coefficients
c      2 normal lattice coefficients
c      3 cascade form coefficients
c
CGTB_START
      if(filtyp .eq. 0)then
        do 87 i=1,nd
          bb(i)=y(i)
87          continue
          nbb=nd
          fb=fs/idec/2.0
        else
CGTB_END
          if(filtyp .eq. 1)then
            call dffil(y,xx,nd,ifch)
          else if( filtyp .eq. 2 )then
            call nlfil(y,xx,nd,ifch)
          else if(filtyp .eq. 3) then
            call casfil(y,xx,nd,ifch)
          endif
          close(unit=ifch)
c
c decimate filtered samples to baseband rate
c
c      ibbdec  decimation factor
c
c      k=0
c      do 88 i=1,nd,ibbdec
c      k=k+1
c      bb(k)=xx(i)
88      continue
c      nbb=nd/ibbdec
CGTB_START
c      fb=fs/idec/ibbdec/2.0
c      fb=fs/idec/ibbdec
CGTB_END
      endif
c
c      Number of points: nd
c
c      return
c      end
c.....
c
c      DESIM
c
c      PURPOSE:
c      To perform a simulation using difference equations
c      for one clock cycle.
c.....
c.....
c      subroutine DESIM(cycle,node,nodep,nodes)
c
c      integer cycle,nodes,circuit,block,nocyc,tsadd,freqpt
c      real period,pi
c      common/funblk/block,period,nocyc,tsadd,pi,freqpt,circuit
c      common/parmax/mxblk,mxdim,mxdim3,mxtele,mxnnode,mxicyc,mxtype,mxnumx
c      real node(mxnnode),nodep(mxnnode)
c      real mindel,maxdel,offset
c      real a1,a2,a3,fgbw1,fgbw2,fgbw3
c      real delta1,delta2,delta3,ofint1,ofint2,ofint3
c      real satmn1,satmx1,satmn2,satmx2,satmn3,satmx3

```

```

real aal,aa2,aa3,b1,b2,b3,c1,c2,c3
common/diffeq/a1,a2,a3,fgbw1,fgbw2,fgbw3,
+   delta,delta2,delta3,
+   ofint1,ofint2,ofint3,
+   satmn1,satmx1,satmn2,satmx2,satmn3,satmx3,
+   maxdel,mindel,offset,
+   aal,aa2,aa3,b1,b2,b3,c1,c2,c3
real maxout(5),minout(5),maxin(5),minin(5)
common/intmax/maxout,minout,maxin,minin
if(cycle.eq.1)then
  do 5 ii=1,5
    maxout(ii)=0.0
    minout(ii)=0.0
    maxin(ii)=0.0
    minin(ii)=0.0
5   continue
endif

xin=node(1)
pt=nodep(nodes)
y1=nodep(2)
y2=nodep(3)
y3=nodep(4)
C These statements are valid only if unity-gain bandwidth is infinite
C
e1=xin-pt*delta1
if (circut .eq. 3) then
  y1d=y1
  y1=y1+e1*a1+ofint1
  u1=c1*u1+aal*y1+b1*y1d
  u1=max(satmn1,min(satmx1,u1))
  e2=u1-pt*delta2
  y2d=y2
  y2=y2+e2*a2+ofint2
  u2=c2*u2+aa2*y2+b2*y2d
  u2=max(satmn2,min(satmx2,u2))
  e3=u2-pt*delta3
  y3d=y3
  y3=y3+e3*a3+ofint3
  u3=c3*u3+aa3*y3+b3*y3d
  u3=max(satmn3,min(satmx3,u3))
  e5=u3
C These statements store maximum integrator values
  if(u1.gt.maxout(1))maxout(1)=u1
  if(u1.lt.minout(1))minout(1)=u1
  if(xin.gt.maxin(1))maxin(1)=xin
  if(xin.lt.minin(1))minin(1)=xin
  if(u2.gt.maxout(2))maxout(2)=u2
  if(u2.lt.minout(2))minout(2)=u2
  if(u1.gt.maxin(2))maxin(2)=u1
  if(u1.lt.minin(2))minin(2)=u1
  if(u3.gt.maxout(3))maxout(3)=u3
  if(u3.lt.minout(3))minout(3)=u3
  if(u2.gt.maxin(3))maxin(3)=u2
  if(u2.lt.minin(3))minin(3)=u2
C These statements are valid only if unity-gain bandwidth is infinite
  node(2)=u1
  node(3)=u2
  node(4)=u3
  nodep(2)=u1
  nodep(3)=u2
  nodep(4)=u3
elseif (circut .eq. 2) then
  y1d=y1
  y1=y1+e1*a1+ofint1
  u1=c1*u1+aal*y1+b1*y1d
  u1=max(satmn1,min(satmx1,u1))
  e2=u1-pt*delta2
  y2d=y2
  y2=y2+e2*a2+ofint2
  u2=c2*u2+aa2*y2+b2*y2d
  u2=max(satmn2,min(satmx2,u2))
  e5=u2
C These statements store maximum integrator values
  if(u1.gt.maxout(1))maxout(1)=u1

```

```

        if(u1.lt.minout(1))minout(1)=u1
        if(xin.gt.maxin(1))maxin(1)=xin
        if(xin.lt.minin(1))minin(1)=xin
        if(u2.gt.maxout(2))maxout(2)=u2
        if(u2.lt.minout(2))minout(2)=u2
        if(u1.gt.maxin(2))maxin(2)=u1
        if(u1.lt.minin(2))minin(2)=u1
C These statements are valid only if unity-gain bandwidth is infinite
        node(2)=u1
        node(3)=u2
        nodep(2)=u1
        nodep(3)=u2
    elseif (circut .eq. 1) then
        y1d=y1
        y1=y1+e1*a1+ofint1
        u1=c1*u1+aa1*y1+b1*y1d
        u1=max(satmn1,min(satmx1,u1))
        e5=u1
C These statements store maximum integrator values
        if(u1.gt.maxout(1))maxout(1)=u1
        if(u1.lt.minout(1))minout(1)=u1
        if(xin.gt.maxin(1))maxin(1)=xin
        if(xin.lt.minin(1))minin(1)=xin
C These statements are valid only if unity-gain bandwidth is infinite
        node(2)=u1
        nodep(2)=u1
    endif
c
c Quantize Error e5
c
        if (e5 .ge. 0.0 ) then
            pt = maxdel
        else
            pt = mindel
        endif
        node(nodes)=pt
        nodep(nodes)=pt
        return
    end
C.....
c
c   DFFIL
c
c   PURPOSE:
c       To simulate a direct form IIR digital Filter
c
c   Direct form filter coefficients are read from a file
c   The inputs from the file are as follows:
c   m n : where m is the number of zeroes plus one
c         n is the number of poles
c   b(i) i=1,m+n : the numerator and denominator coefficients
c
c   In the z domain:
c        $Y(z)/X(z)=C(z)/[1+A(z)]$ 
c        $C(z)=c_0+c_1z^{**}(-1)+... c_{m-1}z^{**}(-m+1)$ 
c        $A(z)=a_1z^{**}(-1)+... a_nz^{**}(-n)$ 
c
c   The subroutine calculates the vector inner product  $y(k)=x\_Tb$ 
c    $x\_$  where  $x\_=[x(k) x(k-1) \dots x(k-m+1) -y(k-1) -y(k-2) \dots -y(k-n)]^T$ 
c    $b\_=[c_0 c_1 \dots c_{m-1} a_1 a_2 \dots a_n]^T$ 
c
c   xx() is the real array of input samples
c   y() is the array of filtered samples calculated by the subroutine
c   npts is the number of points in the arrays
c
c   Sasan H Ardalan, CCSP July 26, 1985
c
C.....
        subroutine dffil(xx,y,npts,ird)
        double precision x(40),b(40)
        real xx(npts),y(npts)
        real temp1,temp2
        real sum
        integer i,j,n,nt,npts,np1,m

```

```

integer mp1,ird
real x1,yr
read(ird,*)m,n
nt=n+m
read(ird,*)(b(i),i=1,nt)
read(ird,*)fs
read(ird,*)wnorm
c write(6,*)'IIR Direct Form Filtering, Sampling Rate:',fs
c
c ***
c *** initialize input vector x and coefficient vector b ***
c ***
do 20 i=1,nt
x(i)=0.0
20 continue
sum=0.0
npl=npts+1
c ***
c *** the following loop calculates the filter response ***
c ***
do 30 j=1,npl
x1=xx(j)
yr=-sum
c
c ***
c *** shift the new value of xx(m) into the vector x ***
c ***
temp2=x1
do 24 i=1,m
temp1=x(i)
x(i)=temp2
temp2=temp1
24 continue
c ***
c *** shift -y into vector x ***
c ***
temp2=yr
mp1=m+1
do 25 i=mp1,nt
temp1=x(i)
x(i)=temp2
temp2=temp1
25 continue
c ***
c *** calculate value of y. ***
c *** yest= x(transpose)*b ***
c ***
sum=0.0
do 27 i=1,nt
sum=sum+x(i)*b(i)
27 y(j)=sum/wnorm
30 continue
return
end
c .....
c
c ENVIRN
c
c PURPOSE:
c This subroutine edits the environment for the program simulation
c
c EDIT PARAMETERS:
c PERIOD - sampling period
c FREQ - sampling frequency
c CYCLES - number of cycles to be simulated
c
c SUBROUTINES CALLED:
c fread
c .....
c
c subroutine ENVIRN
c
integer block,nocyc,circuit
common/funblk/block,period,nocyc,tsadd,pi,freqpt,circuit
common/parmax/mxblk,mxdim,mxdim3,mxtele,mxnode,mxicyc,mxtype,mxnumx

```

```

character*20 str
logical kf,fin,prt,ofile,rfile
common/freerd/nwrite,ntype,aval,ival,str,ich,kf,fin,prt,ofile,rfile

    call fread
    if(str(1:4).eq.'HELP')then
    write(ntype,50)
    if(prt)write(nwrite,50)
50    format(
+ ' ##### ENVIRONMENT #####',/,
+ ' Usage:',/,/,
+ '   ENV HELP',/,/,
+ '   : This Message',/,/,
+ '   ENV CYCLES=nnn1  FREQ=xxx1  PERIOD=xxx2',/,/,
+ '   : control external circuit parameters:',/,/,
+ '   : "nnn1": number of simulation cycles',/,/,
+ '   : "xxx1": clock sampling frequency',/,/,
+ '   : "xxx2": clock sampling period',/,/,
+ '   : FREQ and PERIOD should not be used simultaneously',/,/,
+ ' #####')
    return
    endif

c
c CHANGE SAMPLING PERIOD
c
25    if (str(1:3).eq.'PER'.or.str(1:3).eq.'per')then
    call fread
    if(fin)goto 500
    period=aval
    call fread
    if(str(1:2).eq.'MS'.or.str(1:2).eq.'ms')period=period*1.0e-03
    if(str(1:2).eq.'US'.or.str(1:2).eq.'us')period=period*1.0e-06
    if(str(1:2).eq.'NS'.or.str(1:2).eq.'ns')period=period*1.0e-09
    if(str(1:2).eq.'PS'.or.str(1:2).eq.'ps')period=period*1.0e-12

c
c CHANGE SAMPLING FREQUENCY
c
    else if(str(1:3).eq.'FRE'.or.str(1:3).eq.'fre')then
    call fread
    if(fin)goto 500
    period=1.0/aval
    call fread
    if(str(1:2).eq.'PH'.or.str(1:2).eq.'ph')period=period*1.0e-12
    if(str(1:2).eq.'GH'.or.str(1:2).eq.'gh')period=period*1.0e-09
    if(str(1:2).eq.'MH'.or.str(1:2).eq.'mh')period=period*1.0e-06
    if(str(1:2).eq.'KH'.or.str(1:2).eq.'kh')period=period*1.0e-03

c
c CHANGE NUMBER OF CYCLES TO SIMULATE
c
    else if (str(1:5).eq.'CYCLE'.or.str(1:5).eq.'cycle')then
    call fread
    if(fin)goto 500
    if(ival.gt.mxicyc)then
        write(ntype,*)' Too many cycles specified, Maximum =',mxicyc
        if(prt)then
            write(ntype,*)' Too many cycles specified, Maximum =',mxicyc
        else
            nocyc=ival
        endif
        call fread
    endif
    else
        return
    endif
    goto 25

500    write(ntype,*)'? argument expected for "environment" ?'
    return
    end

C.....
c FDATE
c
c Dummy routine for systems that do not supply the date
c
C.....

```

```

      subroutine fdate(date)
      character*24 date
      date=' DATE GOES HERE
      return
      end
C.....
C
C      FFT
C
C      PURPOSE:
C          To compute the fast fourier transform (FFT) of a sequence
C.....
      subroutine fft(x,n)

      complex x(*),u,w,t
      pi=3.141592653

C
C      unscramble bits
C
      d=n*1.0
      tmpm=log10(d)
      tmp=log10(2.)
      m=nint(tmpm/tmp)
      nv2=n/2
      nml=n-1
      j=1
      do 7 i=1,nml
          if(i.ge.j)goto 4
          t=x(j)
          x(j)=x(i)
          x(i)=t
      4      k=nv2
      5      if(k.ge.j)goto 6
          j=j-k
          k=k/2
          goto 5
      6      j=j+k
      7      continue

C
C      fft of "m" stages for 2**m points
C
      do 20 l=1,m
          le=2**l
          le1=le/2
          u=(1.0,0.0)
          w=cmplx(cos(pi/float(le1)),sin(pi/float(le1)))
          do 30 j=1,le1
              do 10 i=j,n,le
                  ip=i+le1
                  t=x(ip)*u
                  x(ip)=x(i)-t
                  x(i)=x(i)+t
      10          continue
                  u=u*w
      30          continue
      20      continue
      return
      end
C.....
C
C      GAUSS
C
C      PURPOSE:
C          Computes a normally distributed random number with a given
C          mean and standard deviation
C
C      PARAMETERS:
C          s : desired standard deviation of the normal distribution
C          am : desired mean of the normal distribution
C          v : gaussian random number generated
C
C      SUBROUTINES CALLED:
C          ranf
C
C      METHOD:

```

```

c          Uses 12 uniform random numbers to compute normal random
c          numbers by central limit theorem. The result is then
c          adjusted to match the given mean and standard deviation.
c          The uniform random numbers computed within the subroutine
c          are found by the power residue method.
c
c.....
c          subroutine GAUSS(s,am,v)
c
c              real s,am,v,a,y
c
c          COMPUTE THE SUM OF 12 RANDOM NUMBERS
c
c              a=0.0
c              do 50 i=1,12
c                  call ranf(y)
c                  a=a+y
c          50      continue
c
c          COMPUTE GAUSSIAN RANDOM NUMBER
c
c              v=(a-6.0)*s+am
c              return
c              end
c.....
c
c          GENDEC
c
c          PURPOSE:
c              Handles commands for decimation and baseband filtering.
c
c          PARAMETERS:
c          xxin  - delta sigma bit stream          <input>
c          xout  - filtered baseband                <output>
c          nin   - number of bits in xin           <input>
c          fs    - sampling frequency              <input>
c
c          ARGUMENTS:
c          nin   - number of bits in xxin         <input>
c          nout  - number of samples in xout     <output>
c.....
c          subroutine gendec(xxin,xout,fs)
c          common/file/fileno
c          integer fileno
c          common/parmax/mxblk,mxdim,mxdim3,mxtele,mxnode,mxicyc,mxtype,mxnumx
c          integer nin,nout
c          character*20 str
c          logical kf,fin,prt,ofile,rfile
c          common/freerd/nwrite,ntype,aval,ival,
c          +          str,ich,kf,fin,prt,ofile,rfile
c          integer xin(65536),filtyp
c          real xxin(65536),xout(65536)
c
c          call fread
c          if(fin)goto 100
c          if (str(1:3).eq.'NUM')then
c          20      call fread
c                  if(fin)goto 100
c                  if(ival.gt.mxicyc)then
c                      write(ntype,*)' Too many cycles specified, Maximum =',mxicyc
c                      if(prt)then
c                          write(ntype,*)' Too many cycles specified, Maximum =',mxicyc
c                      endif
c                  else
c                      nin=ival
c                  endif
c                  call fread
c                  if(str(1:4).eq.'BASE')then
c                      if(ival.ne.2)then
c                          write(ntype,*)'? Incorrect base specified, Command Terminated ?'
c                          return
c                      else
c                          nin = 2 ** nin
c                      endif
c

```

```

        else
            goto 20
        endif
elseif(str(1:4).eq.'WIND')then
    call fread
    if(fin)goto 100
    if(str(1:4).eq.'UNIF')then
        iw=1
    elseif(str(1:4).eq.'TRIA')then
        iw=2
    elseif(str(1:4).eq.'PARA')then
        iw=3
    else
        write(ntypr,*)' ? Decimation window type not recognized?'
        if(ofile)write(nwrite,*)
+         ' ? Decimation window type not recognized?'
        return
    endif
elseif(str(1:6).eq.'INTDEC')then
    call fread
    if(fin)goto 100
    idec=ival
elseif(str(1:8).eq.'BBANDDEC')then
    call fread
    if(fin)goto 100
    ibbdec=ival
elseif(str(1:4).eq.'TAPS')then
    call fread
    if(fin)goto 100
    nfast=ival
elseif(str(1:4).eq.'FILE')then
    call fread
    if(fin)goto 100
    open(unit=fileno,file=str,status='OLD')
    write(ntypr,*)' Decimation Parameters From File: ',str,' '
    write(nwrite,*)' Decimation Parameters From File: ',str,' '
elseif(str(1:4).eq.'FILT')then
    call fread
    if(fin)goto 100
CGTB_START
c     Only FIR decimate, do not IIR filter baseband
    if(str(1:4).eq.'NONE')then
        filtyp=0
CGTB_END
c     Direct Form Filter
    elseif(str(1:4).eq.'DIRE')then
        filtyp=1
c     Normalized Lattice Filter
    elseif(str(1:4).eq.'NORM')then
        filtyp=2
c     Cascade Form Filter
    elseif(str(1:4).eq.'CASC')then
        filtyp=3
    else
        write(ntypr,*)' ? Filter type not recognized?'
        if(ofile)write(nwrite,*)' ? Filter type not recognized?'
        return
    endif
else
    write(ntypr,2)str(1:10)
    format(' ? unrecognizable command ? ',a10)
    goto 100
endif
call fread
if(.not.fin)goto 20
500 do 44 ii=1,nin
    xin(ii)=nint(xxin(ii))
44 continue
call decima(xin,xout,nin,nout,fs,fb,iw,
+         nfast,idec,ibbdec,fileno,filtyp)
write(ntypr,*)' ',nin,' points decimated to ',nout
if(ofile)write(nwrite,*)' ',nin,' points decimated to ',nout
return
c
c ERROR STATEMENTS

```

```

c
100 write(ntype,*)' ? Insufficient data to do Decimation ?'
    if(ofile)
+   write(nwrite,*)' ? Insufficient data to do Decimation ?'
    return
    end
c.....
c
c   GENTOR
c
c   PURPOSE:
c       This subroutine stores information about the input signal.
c
c   PARAMETERS:
c       gen      : information array for generators
c       genflg   : acknowledges presence of generator i
c
c   STORAGE:    for generator i
c
c   CAPITALS INDICATE KEYWORDS AND THEIR MINIMUM LENGTH
c
c       gen(i,1) : TYPE [1:DC , 2:SINE , 3:RAMP , 4:STEP]
c       gen(i,2) : if type=2 , FREQUENCY (in Hertz)
c                 : if type=4 , INITIAL dc value before step
c       gen(i,3) : AMPLITUDE
c       gen(i,4) : output node
c       gen(i,5) : if type=2 , dc OFFSET
c                 : if type=3 , ramp THRESHOLD (in seconds after initial
c                   cycles skipped)
c                 : if type=4 , DELAY of step after initial cycles skipped
c                   (in seconds)
c       gen(i,6) : if type=2 , PHASE of sinewave (stored in radians)
c
c   SUBROUTINES CALLED:
c       fread
c
c   EXAMPLES:
c       GEN 1 [ 1 ] [ TYPE = SINE ] [ FREQ=1000 ] [ AMP=0.03548 ]
c           [ PHASE = 180 DEG ]
c
c       GEN 1 [ 1 ] [ TYPE= SINE ] [ FREQ=1000 ] [ AMP=-5 ] [ DB WRT 1 ]
c           [ PHASE = 180 RAD ]
c           1000 HZ sinewave generator with amplitude of
c           -5 dB With Rescpet To 1 V.
c
c       GEN 1 [ 1 ] [ TYPE=STEP ] [ INIT=0 ] [ AMP=0.3 ] [ DELAY=0.001 ]
c
c.....
c   subroutine GENTOR(gen,genflg)
c
c       integer block,tsadd,i,mgen,pgen,circuit
c       common/blkmax/mgen,pgen,mscint,pscint,mquant,pquant
c       common/parmax/mxblk,mxdim,mxdim3,mxtele,mxnode,mxicyc,mxtype,mxnumx
c       common/funblk/block,period,nocyc,tsadd,pi,freqpt,circuit
c       real gen(mgen,pgen),save(10)
c       logical genflg(mgen),replac,done
c       character*20 str
c       logical kf,fin,prt,ofile,rfile
c       common/freerd/nwrite,ntype,aval,ival,
+       str,ich,kf,fin,prt,ofile,rfile
c       data pi/3.141592654/
c
c   PROCESS REMAINDER OF INPUT LINE
c
c
10   call fread
    if(fin)goto 100
    if(str(1:4).eq.'HELP')then
        write(ntype,50)
        if(prt)write(nwrite,50)
50   format(
+   ' ##### GENERATOR #####',/,
+   ' Usage:',/,
+   '   GEN HELP',/,
+   '       : This Message',/,
+   '   GEN no node TYPE = DC AMP=x2',/,

```

```

+ '      : generator number "no" at node "node"',//,
+ '      : DC generator of amplitude "x1" and',//,
+ '      GEN no node TYPE = RAMP THR[ESHOLD]=x1 AMP=x2',//,
+ '      : generator number "no" at node "node"',//,
+ '      : ramp generator of amplitude "x2" and',//,
+ '      : threshold of "x1"',//,
+ '      GEN no node TYPE = SINE FREQ=freq OFFSET=x1 AMP=x2',//,
+ '      : generator number "no" at node "node"',//,
+ '      : sinewave generator of frequency "freq"',//,
+ '      : amplitude "x2", offset "x1", and zero phase',//,
+ '      TYPE "MORE" FOR MORE ')
      call input(' ',nerr,1)
      write(ntype,70)
      if(prt)write(nwrite,70)
70      format(
+ '      GEN no node TYPE = SINE FREQ=freq AMP=x1 PHASE = x2 DEG',//,
+ '      : generator number "no" at node "node"',//,
+ '      : sinewave generator of frequency "freq"',//,
+ '      : amplitude "x1", and phase of "x2" degrees',//,
+ '      GEN no node TYPE = SINE FREQ=freq AMP=x1 DB WRT x3',//,
+ '      PHASE = x4 RAD',//,
+ '      : generator number "no" at node "node"',//,
+ '      : sinewave generator of frequency "freq"',//,
+ '      : amplitude "x1" With Respect To "x3" V and',//,
+ '      : phase of "x4" radians',//,
+ '      GEN no node TYPE=STEP INIT=x1 AMP=x2 DELAY=x3',//,
+ '      : step generator with initial value "x1", amplitude',//,
+ '      : "x2" and delay of "x3" seconds',//,
+ '      #####')
      return
      endif
C
C PROCESS INPUT ACCORDING TO FIRST FIELD OF INPUT LINE
C
C DETERMINE WHICH GENERATOR IS BEING INPUT
C
      if(fin)goto 100
      if(kf)goto 200
      i=ival
C
C DETERMINE IF MEMORY SPACE IS AVAILABLE
C
      if(block.eq.mxbk)then
        write(ntype,*)' ? At maximum block capacity ?'
        if(ofile)write(nwrite,*)' ? At maximum block capacity ?'
        return
      else if(i.gt.mgen)then
        write(ntype,*)' ? At maximum generator capacity ?'
        if(ofile)write(nwrite,*)' ? At maximum generator capacity ?'
        return
      endif
C
C DETERMINE IF GENERATOR i EXISTS
C
      replac=.false.
      done=.true.
      if(.not.genflg(i))then
        block=block+1
        genflg(i)=.true.
      else
        write(ntype,*)' ? replacing old generator ?'
        if(ofile)write(nwrite,*)' ? replacing old generator ?'
        replac=.true.
      endif
C
C SAVE OLD GENERATOR INFORMATION IN CASE OF INPUT ERROR
C
      do 5 j=1,pgen
        save(j)=gen(i,j)
        if(.not.replac)gen(i,j)=0.
5      continue
C
C READ GENERATOR DATA
C
C get node number

```

c

```
call fread
if(fin)goto 100
if(.not.kf)then
  gen(i,4)=ival
  call fread
  if(fin)return
elseif(.not.replac)then
  write(ntype,*) '? node number required on first specification ?'
  if(ofile) write(nwrite,*)
+   '? node number required on first specification ?'
  goto 200
endif
20  if(str(1:4).eq.'TYPE')then
  call fread
  if(fin)goto 100
  if(str(1:2).eq.'DC')then
    gen(i,1)=1
  elseif(str(1:4).eq.'SINE')then
    gen(i,1)=2
  elseif(str(1:4).eq.'RAMP')then
    gen(i,1)=3
  elseif(str(1:4).eq.'STEP')then
    gen(i,1)=4
  else
    go to 200
  endif
elseif(str(1:3).eq.'AMP')then
  call fread
  if(fin)goto 100
  gen(i,3)=aval
  call fread
  if(fin)return
  if(str(1:2).eq.'DB')then
    if(gen(i,1).ne.2)goto 200
    call fread
    if(fin)goto 100
    if(str(1:3).eq.'WRT')then
      call fread
      if(fin.or.kf)goto 100
      gen(i,3)=10**((gen(i,3)/20.)*aval)
    else
      goto 200
    endif
  else
    goto 20
  endif
elseif(str(1:5).eq.'PHASE')then
  if(gen(i,1).ne.2)goto 200
  call fread
  if(fin)goto 100
  gen(i,6)=aval
  call fread
  if(.not.fin.and.str(1:3).eq.'DEG')then
    gen(i,6)=gen(i,6)*pi/180
  elseif(.not.fin.and.str(1:3).eq.'RAD')then
    continue
  else
    write(ntype,*)
+   '? Phase must be specified as Degrees or Radians'
    if(ofile)write(nwrite,*)
+   '? Phase must be specified as Degrees or Radians'
    goto 200
  endif
elseif(str(1:4).eq.'FREQ')then
  if(gen(i,1).ne.2)goto 200
  call fread
  if(fin)goto 100
  gen(i,2)=aval
elseif(str(1:4).eq.'INIT')then
  if(gen(i,1).ne.4)goto 200
  call fread
  if(fin)goto 100
  gen(i,2)=aval
elseif(str(1:5).eq.'DELAY')then
```

```

        if(gen(i,1).ne.4)goto 200
        call fread
        if(fin)goto 100
        gen(i,5)=aval
    elseif(str(1:3).eq.'THR')then
        if(gen(i,1).ne.3)goto 200
        call fread
        if(fin)goto 100
        gen(i,5)=aval
    elseif(str(1:6).eq.'OFFSET')then
        if(gen(i,1).ne.2)goto 200
        call fread
        if(fin)goto 100
        gen(i,5)=aval
    else
        goto 200
    endif
    call fread
    if(fin)return
    goto 20
c
c  ERROR STATEMENTS
c
100  write(ntype,*)' ? insufficient generator data ?'
    if(replac)then
        write(ntype,*)' ? old generator still present ?'
        if(ofile)write(nwrite,*)' ? old generator still present ?'
    endif
    done=.false.
    goto 1000

200  write(ntype,*)' ? illegal generator input format ?'
    if(ofile)write(nwrite,*)' ? illegal generator input format ?'
    if(replac)then
        write(ntype,*)' ? old generator still present ?'
        if(ofile)write(nwrite,*)' ? old generator still present ?'
    endif
    done=.false.
c
c  RESTORE OLD GENERATOR IF ERROR OCCURRED DURING INPUT
c
1000  if(.not.done)then
        do 15 j=1,pgen
            gen(i,j)=save(j)
15    continue
        endif
        return
        end
c.....
c
c  INIT
c
c  PURPOSE:
c      This subroutine initializes various quantities
c
c  PARAMETERS:
c      nodep  : array of node values
c
c  CAPITALS INDICATE KEYWORDS AND THEIR MINIMUM LENGTH
c
c  SUBROUTINES CALLED:
c      fread
c
c  EXAMPLES:
c      INIT NODE=2 0.05
c          This initializes the value of x at node 2 to
c          0.05.  i.e.  nodep(2)=0.05
c.....
c.....
c      subroutine INIT(nodep)

common/parmax/mxblk,mxdim,mxdim3,mxtele,mxnode,mxicyc,mxtype,mxnumx
real nodep(mxnode)
character*20 str
logical kf,fin,prt,ofile,rfile

```

```

        common/freerd/nwrite, ntype, aval, ival,
+         str, ich, kf, fin, prt, ofile, rfile
C
C  PROCESS REMAINDER OF INPUT LINE
C
10    call fread
      if(fin)goto 100
      if(str(1:4).eq.'HELP')then
        write(ntype,50)
        if(prt)write(nwrite,50)
50    format(
+     ' ***** INITIALIZE *****',/,
+     ' Usage:',/,
+     '   INIT HELP',/,
+     '       : This Message',/,
+     '   INIT NODE n x',/,
+     '       Initializes NODE n to x. This is required as several',/,
+     '       blocks use previous node values in calculations',/,
+     ' *****')
      return
    endif

C
C  DETERMINE WHICH QUANTITY IS BEING INITIALIZED
C
      if(fin.or..not.kf)goto 100
      if(str(1:4).eq.'NODE')then

C
C  get node number
C
      call fread
      if(fin)goto 100
      if(.not.kf)then
        node=ival
      else
        write(ntype,*) '? node number required ?'
        if(ofile) write(nwrite,*)
+         '? node number required ?'
        goto 100
      endif

C
C  get initial value of node
C
      call fread
      if(fin)goto 100
      if(.not.kf)then
        nodep(node)=aval
      else
        write(ntype,*) '? Initial value of node required ?'
        if(ofile) write(nwrite,*)
+         '? Initial value of node required ?'
        goto 100
      endif
      else
        goto 100
      endif
      return

C
C  ERROR STATEMENTS
C
100  write(ntype,*)'? ? initialization error ?'
      return
      end

C.....
C
C  INTGTR
C
C  PURPOSE:
C    This subroutine stores data for an integrator
C
C  PARAMETERS:
C    scint   : information array for integrators
C    intflg  : acknowledges presence of integrator i
C    ptable  : pointer to tables for integrator values
C
C  STORAGE:   for integrator i

```

```

c      scint(i,1) : type [1:analog , 2:switched_reference]
c      scint(i,2) : integrator reference code [set to 1]
c      scint(i,3) : analog input, node 1
c      scint(i,4) : output node , node 2
c      scint(i,5) : switch input, node 3
c
c      SUBROUTINES CALLED:
c      fread
c.....
c      subroutine INTGTR(scint,intflg,ptable)
c
c      integer mscint,pscint,block,tsadd,fbindx,circuit
c      common/blkmax/mgen,pgen,mscint,pscint,mquant,pquant
c      common/parmax/mxblk,mxdim,mxdim3,mxtele,mxnode,mxicyc,mxtype,mxnumx
c      integer ptable(mxtype,mxblk,mxdim3)
c      real scint(mscint,pscint),save(10)
c      logical intflg(mscint),replac,done
c      common/funblk/block,period,nocyc,tsadd,pi,freqpt,circuit
c      character*20 str
c      logical kf,fin,prt,ofile,rfile
c      common/freerd/nwrite,ntype,aval,ival,
+      str,ich,kf,fin,prt,ofile,rfile
c
c      call fread
c      if(fin)then
c        write(nwrite,*)'insufficient data for "scint" ?'
c        if(prt) write(ntype,*)'insufficient data for "scint" ?'
c        return
c      endif
c      if(str(1:4).eq.'HELP')then
c        write(ntype,50)
c        if(prt)write(nwrite,50)
50      format(
+ ' ##### INTEGRATOR #####',/,
+ ' Usage:',/,
+ '   SCINT HELP',/,
+ '   : This Message',/,
+ '   SCINT n TYPE=SWITCH n1 n2 n3 SAME=nnn1',/,
+ '   : switched capacitor integrator number "n"',/,
+ '   : with positive input node "n1"',/,
+ '   : and negative input node "n2"',/,
+ '   : and output node "n3"',/,
+ '   : option-> previously defined integrator number "nnn1"',/,
+ '   : uses the same data table',/,
+ '   SCINT n TYPE=ANALOG DIM=nnn1 n1 n2 SAME=nnn2',/,
+ '   : analog integrator number "n" with table DIMENSION',/,
+ '   : "nnn1" (excluding input) and input node "n1"',/,
+ '   : and output node "n2"',/,
+ '   : option-> previously defined integrator number "nnn2"',/,
+ '   : uses the same data table',/,
+ ' #####')
c      return
c      endif
c
c      DETERMINE WHICH INTEGRATOR IS INPUT
c
c      i=ival
c
c      DETERMINE IF MEMORY SPACE IS AVAILABLE
c
c      if(block.eq.mxblk)then
c        write(ntype,*)' ? At maximum block capacity ?'
c        return
c      else if (i.gt.mscint)then
c        write(ntype,*)' ? At maximum integrator capacity ?'
c        return
c      endif
c
c      DETERMINE IF INTEGRATOR i EXISTS
c
c      replac=.false.
c      done=.true.
c      if(.not.intflg(i))then
c        block=block+1
c        intflg(i)=.true.
c      else

```

```

        write(ntype,*)'? replacing old integrator ?'
        replac=.true.
    endif
c
c  SAVE OLD INTEGRATOR INFORMATION IN CASE OF INPUT ERROR
c
    do 5 j=1,pscint
        save(j)=scint(i,j)
5    continue
c
c  READ INTEGRATOR DATA
c
    scint(i,2)=1
    fbindx=int(scint(i,2))
    call fread
    if(fin)goto 100
    if(str(1:3).ne.'TYP'.and.str(1:3).ne.'typ')goto 200
    call fread
    if(fin)goto 100

    if(str(1:3).eq.'ANA'.or.str(1:3).eq.'ana')then
        scint(i,1)=1
        call fread
        if(fin)goto 100
        if(ival.le.6.and.ival.ge.0)then
            ptable(fbindx,i,1)=ival
        else
            write(ntype,*)'? maximum table dimension is six (6) ?'
        endif

        call fread
        if(fin)goto 100
        scint(i,3)=ival
        call fread
        if(fin)goto 100
        scint(i,4)=ival
        scint(i,5)=0
    else if(str(1:3).eq.'SWI'.or.str(1:3).eq.'swi')then
        scint(i,1)=2
        call fread
        if(fin)goto 100
        scint(i,3)=ival
        call fread
        if(fin)goto 100
        scint(i,4)=ival
        call fread
        if(fin)goto 100
        scint(i,5)=ival
        ptable(fbindx,i,1)=3
    else
        write(ntype,*)'? ? no such integrator type ??'
    endif
c
c  DETERMINE IF INTEGRATOR HAS SAME CHARACTERISTICS AS ANOTHER ONE
c
    call fread
    if(.not.fin)then
        if(str(1:1).ne.'S'.and.str(1:1).ne.'s')goto 200
        call fread
        if(fin)goto 100
        if(ival.lt.1.or.ival.gt.mscint)goto 200
        if(.not.intflg(ival))then
            write(ntype,46)ival
46        format(' ? Integrator ',i3,' table DOES NOT EXIST ?')
            goto 1000
        endif
        do 10 jj=1,mxdim3
            ptable(fbindx,i,jj)=ptable(fbindx,ival,jj)
10        continue
    endif
    goto 1000
c
c  ERROR STATEMENTS
c
100    write(ntype,*)'? insufficient integrator data ?'

```

```

        if(replac)then
            write(ntype,*)' ? old integrator still present ?'
        endif
        done=.false.
        goto 1000

200    write(ntype,*)' ? illegal integrator input format ?'
        if(replac)then
            write(ntype,*)' ? old integrator still present ?'
        endif
        done=.false.

c
c  RESTORE OLD INTEGRATOR IF ERROR OCCURRED DURING INPUT
c
1000   if(.not.done)then
        do 15 j=1,pscint
            scint(i,j)=save(j)
15     continue
        endif
        return
        end

c.....
c
c      NLFIL
c
c      PURPOSE:
c          To simulate a normalized lattice filter
c
c          This subroutine filters an array of real
c          numbers using the normalized lattice filter
c
c          The subroutine requires the channel for a file containing
c          the filter parameters
c.....
        subroutine nlfil(x,y,npts,ird)
        real x(npts),y(npts)
        double precision k(35),c(35),nu(35),xb(35)
        double precision xf,wsc,sum,tmp
        integer n,n1,i,j,m,ird
        read(ird,*)n
        do 10 i=1,n
            read(ird,*)k(i)
            tmp=k(i)*k(i)
            tmp=1.0-tmp
            c(i)=sqrt(tmp)
10     continue
        n1=n+1
        do 20 i=1,n1
20     read(ird,*)nu(i)
            read(ird,*)wsc
            read(ird,*)fs
            read(ird,*)wnorm
        c      write(6,*)'Normalized Lattice Filtering, Sampling Rate:',fs
        c
        c  initialize all delays to zero
        c
        do 30 i=1,n1
30     xb(i)=0.0
        c
        c  begin filtering
        c
        do 40 j=1,npts
            xf=x(j)
            do 42 m=1,n
                i=n-m+2
                xb(i)=xb(i-1)*c(i-1)+k(i-1)*xf
                xf=xf*c(i-1)-k(i-1)*xb(i-1)
42     continue
            xb(1)=xf
            sum=0.0
            do 44 m=1,n1
                sum=sum+xb(m)*nu(m)
44     continue
            y(j)=sum
40     continue

```

```

        return
        end
c.....
c
c      NODEPT
c
c      PURPOSE:
c          This subroutine prints the node values
c
c      PARAMETERS:
c          x      : array containing circuit values
c          ptnode : print that node
c          ndpres : does the node exist
c.....
c      subroutine NODEPT(x,ptnode,ndpres)
c
c          integer cycle,numnod,a,nocyc,freqpt,circuit,mx,psucyc
c          common/parmax/mxblk,mxdim,mxdim3,mxtele,mxnnode,mxicyc,mxtype,mxnumx
c          common/funblk/block,period,nocyc,tsadd,pi,freqpt,circuit
c          real x(mxicyc,mxnnode)
c          logical ptnode(mxnnode),ndpres(mxnnode),found
c          character*20 str
c          logical kf,fin,prt,ofile,rfile
c          common/freerd/nwrite,ntype,aval,ival,str,ich,kf,fin,prt,ofile,rfile
c
c      LOCATE THE LARGEST NODE VALUE
c
c          do 1 i=1,mxnnode
c            if(ndpres(i))mx=i
c          1   continue
c
c      DEFINE COLUMNS FOR OUTPUT
c
c          a=1
c          found=.false.
c          numnod=0
c          do 20 i=a,mxnnode
c            if(ptnode(i).and.ndpres(i))then
c              node1=i
c              a=i+1
c              numnod=1
c              if(i.eq.mx)found=.true.
c              goto 25
c            endif
c          20  continue
c
c          goto 5000
c
c          do 30 i=a,mxnnode
c            if(ptnode(i).and.ndpres(i))then
c              node2=i
c              a=i+1
c              numnod=2
c              if(i.eq.mx)found=.true.
c              goto 35
c            endif
c          30  continue
c
c          do 40 i=a,mxnnode
c            if(ptnode(i).and.ndpres(i))then
c              node3=i
c              a=i+1
c              numnod=3
c              if(i.eq.mx)found=.true.
c              goto 45
c            endif
c          40  continue
c
c          do 50 i=a,mxnnode
c            if(ptnode(i).and.ndpres(i))then
c              node4=i
c              a=i+1
c              numnod=4
c              if(i.eq.mx)found=.true.

```

```

        goto 80
    endif
50    continue
c
c PRINT TABLE HEADING
c
80    write(nwrite,1000)
    if(numnod.eq.1)write(nwrite,1010)node1
    if(numnod.eq.2)write(nwrite,1020)node1,node2
    if(numnod.eq.3)write(nwrite,1030)node1,node2,node3
    if(numnod.eq.4)write(nwrite,1040)node1,node2,node3,node4
c
c PRINT TABLE VALUES
c
    ifir=nocyc-mxicyc+1
    if(ifir.le.1)then
        psucyc=1
    else
        psucyc=ifir
    endif
90    do 100 cycle=1,mxicyc,freqpt
    if(cycle.gt.nocyc)goto 5000
    if(numnod.eq.1.and.found)then
        write(nwrite,1062)psucyc,int(x(cycle,node1))
    else if(numnod.eq.1)then
        write(nwrite,1060)psucyc,x(cycle,node1)
    else if(numnod.eq.2.and.found)then
        write(nwrite,1072)psucyc,x(cycle,node1),
+         int(x(cycle,node2))
    else if(numnod.eq.2)then
        write(nwrite,1070)psucyc,x(cycle,node1),
+         x(cycle,node2)
    else if(numnod.eq.3.and.found)then
        write(nwrite,1082)psucyc,x(cycle,node1),
+         x(cycle,node2),int(x(cycle,node3))
    else if(numnod.eq.3)then
        write(nwrite,1080)psucyc,x(cycle,node1),
+         x(cycle,node2),x(cycle,node3)
    else if(numnod.eq.4.and.found)then
        write(nwrite,1092)psucyc,x(cycle,node1),
+         x(cycle,node2),x(cycle,node3),int(x(cycle,node4))
    else
        write(nwrite,1090)psucyc,x(cycle,node1),
+         x(cycle,node2),x(cycle,node3),x(cycle,node4)
    endif
    psucyc=psucyc+freqpt
100   continue
c
c PRINT ANOTHER TABLE FOR THE NEXT 4 NODES
c
    goto 10

1000  format(// ' NODE VALUES' //)
1010  format(' Cycle',x,1(3x,'Node ',i3,6x)/)
1020  format(' Cycle',x,2(3x,'Node ',i3,6x)/)
1030  format(' Cycle',x,3(3x,'Node ',i3,6x)/)
1040  format(' Cycle',x,4(3x,'Node ',i3,6x)/)
1050  format(' ')
1060  format(i6,x,1(2x,e15.8))
1062  format(i6,x,2x,7x,i2)
1070  format(i6,x,2(2x,e15.8))
1072  format(i6,x,2x,e15.8,2x,7x,i2)
1080  format(i6,x,3(2x,e15.8))
1082  format(i6,x,2(2x,e15.8),2x,7x,i2)
1090  format(i6,x,4(2x,e15.8))
1092  format(i6,x,3(2x,e15.8),2x,7x,i2)
5000  return
    end
c .....
c
c NODSET
c
c PURPOSE:
c     This subroutine determines which circuit nodes will be printed
c

```

```

c     PARAMETERS:
c     ptnode : should the node value be stored
c
c     SUBROUTINES CALLED:
c     fread
c
c.....
c     subroutine NODSET(ptnode)
c
c     common/parmax/mxblk,mxdim,mxdim3,mxtele,mxnode,mxicyc,mxtype,mxnumx
c     logical ptnode(mxnode),offnod
c     logical cpypar,cpynod,over,clear
c     common/commnd/cpypar,cpynod,over,clear
c     integer freqpt,circuit
c     common/funblk/block,period,nocyc,tsadd,pi,freqpt,circuit
c     character*20 str
c     logical kf,fin,prt,ofile,rfile
c     common/freerd/nwrite,ntype,aval,ival,str,ich,kf,fin,prt,ofile,rfile
c
c     offnod=.false.
c     call fread
c     if(fin)then
5       write(ntype,*)'? argument expected for "print node" ?'
c       return
c     endif
c
c     DETERMINE HOW OFTEN TO PRINT THE NODE VALUES
c
c     if(str(1:3).eq.'FRE'.or.str(1:3).eq.'fre')then
c     call fread
c     if(fin)then
c       write(ntype,*)'? argument expected for "print node freq=" ?'
c       return
c     else
c       freqpt=ival
c       if(ival.eq.0)freqpt=10
c       return
c     endif
c     endif
c
c     DETERMINE IF PRINT MODE IS ON OR OFF
c
c     cpynod=.true.
c     if(str(1:3).eq.'OFF'.or.str(1:3).eq.'off')then
c     call fread
c     if(fin)then
c     cpynod=.false.
c     return
c     endif
c     offnod=.true.
c     goto 70
c     endif
c
c     DETERMINE IF THE REQUIRED ACTION INVOLVES EVERY NODE
c
c     if(str(1:1).eq.'A'.or.str(1:1).eq.'a')then
c     do 20 i=1,mxnode
c       ptnode(i)=.true.
20    continue
c     return
c     endif
c
c     READ EACH INDIVIDUAL NODE SPECIFIED
c
c     goto 70
50    call fread
c     if(fin)return
70    if(ival.eq.0)write(ntype,*)'? noninteger node value ?'
c     if(ival.eq.0)goto 50
c     if(offnod)then
c       ptnode(ival)=.false.
c     else
c       ptnode(ival)=.true.
c     endif
c     goto 50

```

```

end
C .....
C
C NOISET
C
C PURPOSE:
C   This subroutine stores node noise information
C
C STORAGE:
C   noise (i,1) : noise existence [ 1/noise, 0/no noise]
C   noise (i,2) : mean value of noise
C   noise (i,3) : standard deviation of noise
C .....
C   subroutine NOISET
C
C   real noise(10,3)
C   common/nose/noise
C   logical kf,fin,prt,ofile,rfile
C   character*20 str
C   common/freerd/nwrite,ntype,aval,ival,str,ich,kf,fin,prt,ofile,rfile
C
C   call fread
C   if(fin)goto 100
C   if(str(1:4).eq.'HELP')then
C     write(ntype,50)
C     if(prt)write(nwrite,50)
50   format(
C + ' ##### NOISE #####',/,
C + ' Usage:',/,/,
C + '   NOISE HELP',/,
C + '   : This Message',/,/,
C + '   NOISE n x1 x2',/,
C + '   : Add Gaussian white noise to node "n" with mean "x1"',/,
C + '   : and standard deviation "x2"',/,/,
C + ' #####')
C   return
C   endif
C
C READ CIRCUIT NODE TO ADD NOISE
C
C   if(fin.or.aval.eq.0.0)goto 100
C   i=ival
C
C READ MEAN VALUE OF NOISE
C
C   call fread
C   if(fin.or.aval.eq.0.0)goto 100
C   noise(i,2)=aval
C
C READ STANDARD DEVIATION
C
C   call fread
C   if(fin.or.aval.eq.0.0)goto 100
C   noise(i,3)=aval
C   noise(i,1)=1.0
C   goto 1000
C
100  write(ntype,*)'? insufficient or illegal input for "noise" ?'
1000  return
      end
C .....
C
C POLATE
C
C PURPOSE:
C   This subroutine interpolates table data.
C
C METHOD:
C   Newton interpolation routine .
C   See p.23 Carnahan, Luther and Wilkes
C
C PARAMETERS:
C   dim      : number of variables used in interpolation (max=3)
C   fbindx   : functional block code

```

```

c      blk      : block i of fbindx type
c      ptable   : pointer to table values
c      tables   : table of functional block values
c      out      : output of interpolation
c      x1       : signal number 1
c      x2       : signal number 2
c      x3       : signal number 3 (discrete)
c
c.....
      subroutine POLATE(dim,fbindx,blk,ptable,tables,out,x1,x2,x3)

      integer x1beg,x2beg,x3beg,dim,knt,fbindx,beg,blk,st1
      integer stal,fin1,sta2,fin2,g1,g2,g3,g4
      common/parmax/mxblk,mxdim,mxdim3,mxtele,mxnode,mxicyc,mxtype,mxnumx
      real pt1,pt2,pt3,pt4,s1,s2,f1,f2,deltaa,deltac
      real x1,x2,x3,fnewt
      real tables(mxtele),out
      real a1(20),b1(20),table(20,20)
      integer ptable(mxtype,mxblk,mxdim3)
      character*20 str
      logical kf,fin,prt,ofile,rfile
      common/freerd/nwrite,ntype,aval,ival,str,ich,kf,fin,prt,ofile,rfile
      common/table/exceed
      integer exceed
CGTB_START
      integer hist(5,50,50,3)
      logical phist
      common/histgm/phist,hist
CGTB_END
      data exceed/0/

      knt=mxnumx
      goto(10,50,50)dim
c*****
c
c   ONE DIMENSIONAL INTERPOLATION
c
c*****
10      beg=ptable(fbindx,blk,2)
      ideg=ptable(fbindx,blk,3)
      x2beg=ptable(fbindx,blk,4)
      if(ideg.ge.x2beg)ideg=x2beg-1
      if(x1.lt.tables(beg).or.x1.gt.tables(beg-1+x2beg))then
        write(ntype,*)'? table values exceeded ?'
        exceed=exceed+1
        if(exceed.gt.100)goto 9900
        goto 1000
      endif
c
c   LINEAR INTERPOLATION
c
      if(ideg.eq.1)then
        if(x1.eq.tables(beg))then
          stal=1
          fin1=2
          else
            do 30 i=1,ptable(fbindx,blk,4)
              if(x1.le.tables(beg+(i-1)))goto 40
              continue
            stal=i-1
            fin1=i
          endif
          st1=ptable(fbindx,blk,4)
          if(tables(beg+fin1-1)-tables(beg+stal-1).eq.0)then
            out=0.
          else
            out=(x1-tables(beg+stal-1))
            out=out/(tables(beg+fin1-1)-tables(beg+stal-1))
          endif
          out=out*(tables(beg+st1+fin1-1)-tables(beg+st1+stal-1))
          out=out+tables(beg+st1+stal-1)
          return
        else
c
c   POLYNOMIAL INTERPOLATION

```

```

c
    if(ptable(fbindx,blk,4).lt.6)then
        mx=ptable(fbindx,blk,4)-1
    else
        mx=5
    endif
    do 20 i=1,ptable(fbindx,blk,4)
        a1(i)=tables(ptable(fbindx,blk,2)+(i-1))
        b1(i)=tables(ptable(fbindx,blk,2)+ptable(fbindx,blk,4)+(i-1))
20    continue
    call DTABLE(a1,b1,table,ptable(fbindx,blk,4),mx,trubl,knt)
    if(trubl.ne.0.0)goto 1000
    out=fnewt(a1,b1,table,ptable(fbindx,blk,4),mx,ideg,x1,trubl,knt)
    if(trubl.ne.0.0)goto 1000
    return
    endif
c*****
c
c   MULTI-DIMENSIONAL INTERPOLATION
c
c*****
50    x1beg=ptable(fbindx,blk,2)
        x2beg=x1beg+ptable(fbindx,blk,4)
        x3beg=x2beg+ptable(fbindx,blk,5)
c
c   CHECK FOR TABLE OVERFLOW ERROR
c
    if(x1.lt.(tables(x1beg)-0.0001).or.x1.gt.(tables(x2beg-1)+0.0001))then
        write(ntype,*)'? table values exceeded ?'
        exceed=exceed+1
        if(exceed.gt.100)goto 9900
        goto 1000
    endif
    if(x2.lt.(tables(x2beg)-0.0001).or.x2.gt.(tables(x3beg-1)+0.0001))then
        write(ntype,*)'? table values exceeded ?'
        exceed=exceed+1
        if(exceed.gt.100)goto 9900
        goto 1000
    endif
    if(x1.lt.tables(x1beg)) x1=tables(x1beg)
    if(x1.gt.tables(x2beg-1)) x1=tables(x2beg-1)
    if(x2.lt.tables(x2beg)) x2=tables(x2beg)
    if(x2.gt.tables(x3beg-1)) x2=tables(x3beg-1)
c
c   FIND BEGINNING OF OUTPUT TABLE
c
    beg=ptable(fbindx,blk,2)
    do 55 i=4,mxdim3
        beg=beg+ptable(fbindx,blk,i)
55    continue

    if(dim.eq.3)then
        do 58 i=1,ptable(fbindx,blk,6)
            if(x3.eq.tables(x3beg+(i-1)))goto 59
58    continue
        write(ntype,*)'?? interpolation error -> non-discrete 3rd dim ??'
        return
59    beg=beg+ptable(fbindx,blk,4)*ptable(fbindx,blk,5)*(i-1)
    endif
c
c   FIND INTERPOLATION INPUT VALUES FOR X1
c
    if(x1.eq.tables(ptable(fbindx,blk,2)))then
        stal=1
        finl=2
    else
        do 90 i=1,ptable(fbindx,blk,4)
            if(x1.le.tables(ptable(fbindx,blk,2)+(i-1)))goto 100
90    continue
100    stal=i-1
        finl=i
    endif
    s1=tables(ptable(fbindx,blk,2)+stal-1)
    f1=tables(ptable(fbindx,blk,2)+finl-1)
c

```

```

C  FIND INTERPOLATION INPUT VALUES FOR X2
C
      if(x2.eq.tables(x2beg))then
        sta2=1
        fin2=2
      else
        do 60 i=1,ptable(fbindx,blk,5)
          if(x2.le.tables(x2beg+(i-1)))goto 70
60      continue
70      sta2=i-1
        fin2=i
      endif
      s2=tables(x2beg+sta2-1)
      f2=tables(x2beg+fin2-1)
C
C  FIND THE FOUR (4) INTERPOLATING POINTS IN THE OUTPUT TABLE
C
      g1=beg+sta1-1+ptable(fbindx,blk,4)*(sta2-1)
      g2=beg+fin1-1+ptable(fbindx,blk,4)*(sta2-1)
      g3=beg+sta1-1+ptable(fbindx,blk,4)*(fin2-1)
      g4=beg+fin1-1+ptable(fbindx,blk,4)*(fin2-1)
      pt1=tables(g1)
      pt2=tables(g2)
      pt3=tables(g3)
      pt4=tables(g4)
C
C  FORMULA FOR INTERPOLATING
C
      if(f1-s1.eq.0)then
        deltaa=0.
      else
        deltaa=(x1-s1)/(f1-s1)
      endif
      if(f2-s2.eq.0)then
        deltac=0.
      else
        deltac=(x2-s2)/(f2-s2)
      endif
      out=pt1+deltaa*(pt2-pt1)+deltac*(pt3-pt1)
      out=out+deltaa*deltac*(pt4+pt1-pt3-pt2)
C
CGTB_START
C  This determines which table points are used.
      ideep=1
      if(ptable(fbindx,blk,6).eq.2)then
        if(nint(x3).eq.1)ideep=2
      else
        if(nint(x3).eq.0)ideep=2
        if(nint(x3).eq.1)ideep=3
      endif

      hist(blk,sta1,sta2,ideep)=1+hist(blk,sta1,sta2,ideep)
      hist(blk,sta1,fin2,ideep)=1+hist(blk,sta1,fin2,ideep)
      hist(blk,fin1,sta2,ideep)=1+hist(blk,fin1,sta2,ideep)
      hist(blk,fin1,fin2,ideep)=1+hist(blk,fin1,fin2,ideep)
CGTB_END

1000  return
9900  write(ntype,*) '? POLATE - Too many function values exceeded ?'
      write(nwrite,*) '? POLATE - Too many function values exceeded ?'
      write(nwrite,*) '? POLATE x1=',x1,' x2=',x2,' x3=',x3
      stop
      end
C.....
C
C      DTABLE
C
C      PURPOSE:
C          This subroutine creates a table of divided differences
C.....
C.....
      subroutine DTABLE(a,b,table,u,m,tru,bl,k)
      dimension a(n),b(n),table(k,k)

```

```

        if(m.lt.n)goto 2
        trubl=1.0
        return

2       nml=n-1
        do 3 i=1,nml
3       table(i,1)=(b(i+1)-b(i))/(a(i+1)-a(i))
        continue
        if(m.le.1)goto 20

        do 10 j=2,m
          do 15 i=j,nml
            isub=i+1-j
            table(i,j)=(table(i,j-1)-table(i-1,j-1))/(a(i+1)-a(isub))
15         continue
10        continue

20       trubl=0.0
        return
        end
C.....
C
C       FNEWT
C
C       PURPOSE:
C         This function performs the actual newton interpolation
C         assuming that the table of divided differences is present
C
C.....
function fnewt(a,b,table,n,m,ideg,xarg,trubl,k)

dimension a(n),b(n),table(k,k)

if(ideg.le.m) goto 2
trubl =2.0
fnewt=0.0
return

2       do 4 i=1,n
          if(i.eq.n.or.xarg.le.a(i)) goto 5
4       continue

5       max=i+ideg/2
C
        if(max.le.ideg) max=ideg+1
        if(max.gt.n) max=n
C
        yest=table(max-1,ideg)
        if(ideg.le.1) goto 13
        idegm1 = ideg - 1
        do 12 i=1,idegm1
          isubl = max-i
          isub2 = ideg-i
          yest=yest*(xarg-a(isubl))+table(isubl-1,isub2)
12        continue
13        isubl=max-ideg
        trubl=0.0
        fnewt=yest*(xarg-a(isubl))+b(isubl)
        return
        end
C.....
C
C       RANF
C
C       PURPOSE:
C         This function generates a random number between zero and one.
C
C       CONDITIONS:
C         If (c < 0) then c represents a new seed and a new sequence begins
C
C         If (c = 0) then the previous number in the sequence is returned
C
C         If (c > 0) then the next number in the sequence is returned
C
C         A default seed can be provided (lstx) such that the sequence is

```

```

c         repeatable if a new seed is not provided.
c
c     PARAMETERS:
c         xout : output random number
c
c.....
c         subroutine RANF(xout)
c
c             real xout
c             integer c
c             common/misc/c
c             character*20 str
c             logical kf,fin,prt,ofile,rfile
c             common/freerd/nwrite,ntype,aval,ival,str,ich,kf,fin,prt,ofile,rfile
c             double precision w,a,lstx
c             common/rdom/lstx
c
c             w=1.4074d14
c             xprime=9973.0
c             a=273673163155.0
c
c     START NEW SEQUENCE
c
c         if(c.lt.0)then
c             c=abs(c)
c             lstx=c*xprime
c             lstx=a*lstx
c             lstx=lstx-(int(lstx/w)*w)
c             xout=lstx/w
c             return
c
c     RETURN LAST X
c
c         else if(c.eq.0)then
c             xout=lstx/w
c             return
c
c     RETURN NEXT NUMBER IN SEQUENCE
c
c         else
c             lstx=xprime*lstx
c             lstx=lstx-(int(lstx/w)*w)
c             xout=lstx/w
c         endif
c         return
c         end
c.....
c
c     SDRSUB
c
c     PURPOSE: To calculate signal-to-total distortion ratios,
c             signal-to-white noise ratios, and signal-to-harmonic
c             distortion ratios for a fourier sequence of numbers
c
c.....
c         subroutine sdrsub(xx,equal,sdr,snr,sth,n,spread,iw,fb,sampfq,
c             +             period,sigfq,prt,nwrite)
c
c     CGTB_START
c         logical prt
c         integer iw
c         real fb,period,sigfq,sampfq
c
c     CGTB_END
c         real xx(n),equal(n),sdr,snr,sth
c         double precision spec(32768),totsp
c         integer n,nwrite,spread
c         logical tot3
c         common/fourie/snoise,sig,totno1,totno2,totno3,snr1,snr2,snr3,fftyes
c
c     DETERMINE BASEBAND BINS
c
c     CGTB_START
c         if(iw.eq.0) sampfq=1.0/period
c         iband=nint(n*fb/sampfq)
c         iict=nint(n*sigfq/sampfq+1)
c         if(iict.eq.1)then

```

```

write(ntype,*)'No sdr calculation completed for D.C. Signal'
if(prt)
+   write(nwrite,*)'No sdr calculation completed for D.C. Signal'
return
endif
write(nwrite,*)'baseband frequency:',fb,'Hz.; sampling',
+   ' frequency:',sampfq,'Hz.'
write(nwrite,*)n,'total fft bins'
write(nwrite,*)iband,'bins in baseband; signal on bin',iict
c
c   get energy and equalize for sin(x)/x correction
c
totssp=0.0
do 900 j=1,iband+1
  spec(j)=xx(2*j-1)*xx(2*j-1)+xx(2*j)*xx(2*j)
  spec(j)=spec(j)*equal(j)*equal(j)
  totssp=totssp+spec(j)
900 continue
CGTB_END
totno1=2*spec(1)
totno2=2*spec(1)
totno3=0.0
sig=0.0
iharm=nint(iband/(iict-1)*1.0-1.0)
write(nwrite,*)iharm,'harmonics in baseband'
icount=1
c totno1: total distortion
c totno2: white noise
c totno3: harmonic noise
c iharm : number of harmonics
c icount: present harmonic
do 449 j=2,iband+1
  if(j.ge.iict-spread.and.j.le.iict+spread)then
    sig=sig+2*spec(j)
  else
    totno1=totno1+2*spec(j)
    tot3=.false.
    itmp=icount*(iict-1)+iict
    if(j.ge.itmp-spread.and.j.le.itmp+spread)then
      tot3=.true.
      if(j.eq.itmp+spread)icount=icount+1
    endif
    if(tot3)then
      totno3=totno3+2*spec(j)
    else
      totno2=totno2+2*spec(j)
    endif
  endif
449 continue
c
c take out white noise from signal
c
ave=(spec(iict-spread-1)+spec(iict+spread+1))/2.0
sig=sig-2*(2*spread+1)*ave
totno1=totno1+2*(2*spread+1)*ave-2*(spec(1)+spec(2))
totno2=totno2+2*(2*spread+1)*ave-2*(spec(1)+spec(2))
c
c separate harmonic noise from white noise
c
do 67 ii=1,iharm
  itmp=ii*(iict-1)+iict
  if(itmp+spread+1.gt.iband+1)then
    ave=(spec(itmp-spread-1)+spec(itmp-spread-2))/2.0
  else
    ave=(spec(itmp-spread-1)+spec(itmp+spread+1))/2.0
  endif
CGTB_START
if(spec(itmp).gt.ave)then
  totno2=totno2+2*(2*spread+1)*ave
  totno3=totno3-2*(2*spread+1)*ave
else
  do 3 i=-1.0*spread,spread
    totno2=totno2+spec(itmp+i)
    totno3=totno3-spec(itmp+i)
  3 continue

```

```

        endif
CGTB_END
67 continue
CGTB_START
    if(totno3.lt.0.0)then
        totno3=0.0
        totno2=totno2-totno3
    endif
CGTB_END
c
c compute snr ratios in dB
c
    if(totno1.lt.1.0e-15)then
        sdr=sig/1.0e-15
    else
        sdr=sig/totno1
    endif
    if(sdr.lt.1.0e-10)then
        sdr=-100.0
    else
        sdr=10.0*log10(sdr)
    endif
    if(totno2.lt.1.0e-15)then
        snr=sig/1.0e-15
    else
        snr=sig/totno2
    endif
    if(snr.lt.1.0e-10)then
        snr=-100.0
    else
        snr=10.0*log10(snr)
    endif
    if(totno3.lt.1.0e-15)then
        sthd=sig/1.0e-15
    else
        sthd=sig/totno3
    endif
    if(sthd.lt.1.0e-10)then
        sthd=-100.0
    else
        sthd=10.0*log10(sthd)
    endif
    write(nwrite,*)
    write(nwrite,312)
312 format(' ',78('-'))
    write(nwrite,*) ' FFT RESULTS'
    write(nwrite,*)
+   write(nwrite,*)' Signal Strength      : ',sig/totsp,
+   ' Total Distortion : ',totno1/totsp,
+   write(nwrite,*)' Total Noise      : ',totno2/totsp,
+   ' Total Harmonics : ',totno3/totsp
    write(nwrite,*)
    write(nwrite,*)' Signal-to-Distortion (dB): ',sdr
    write(nwrite,*)' Signal-to-Noise (dB): ',snr
    write(nwrite,*)' Signal-to-Harmonic (dB): ',sthd
    write(nwrite,312)
    return
end
C.....
C
C TABLRD
C
C PURPOSE:
C This subroutine reads the elements of the input table for a
C functional block.
C
C PARAMETERS:
C scint : information array for integrator
C quant : information array for quantizer
C ptable : pointer to table values
C tables : values for functional blocks
C
C.....
subroutine TABLRD(scint,quant,ptable,tables)

```

```

integer block,tsadd,circuit,pngen,pscint,pquant
common/blkmax/mgen,pngen,mscint,pscint,mquant,pquant
common/parmax/mxblk,mxdim,mxdim3,mxtele,mxnode,mxicyc,mxtype,mxnumx
common/funblk/block,period,nocyc,tsadd,pi,freqpt,circuit
integer z,count,outpt,nod,fbindx,n,nerr
integer ptable(mxtype,mxblk,mxdim3)
real tables(mxtele),scint(mscint,pscint),quant(mquant,pquant)
logical inerr,lsave
character*20 str,ty,infile*25,outfil*25
logical kf,fin,prt,ofile,rfile,ifile,wfile
common/freerd/nwrite,ntype,aval,ival,str,ich,kf,fin,prt,ofile,rfile
common/iostr/nread,nprint,ifile,wfile,ntti,nwrt,ndski,
+ ndsko,infile,outfil
c
c TABLE FOR WHAT BLOCK
c
call fread
if(fin)then
write(nwrite,*)'insufficient data for "table" ?'
if(prt)write(ntype,*)'insufficient data for "table" ?'
endif
if(str(1:4).eq.'HELP')then
write(ntype,50)
if(prt)write(nwrite,50)
50 format(
+ ' ##### TABLE #####',/
+ ' Usage:',//,
+ ' TABLE HELP',//,
+ ' : This Message',//,
+ ' TABLE SCINT n file',//,
+ ' : "file" is the input file which contains the TABLE',//,
+ ' : that describes the switched capacitor integrator',//,
+ ' : number "n".',//,
+ ' TABLE QUANT n file',//,
+ ' : "file" is the input file which contains the TABLE',//,
+ ' : that describes the quantizer number "n".',//,
+ ' NOTE: quantizer table not yet operational!',//,
+ ' #####')
return
endif
ty=str
call fread
n=ival
if (ty(1:3).eq.'SCI'.or.ty(1:3).eq.'sci')then
fbindx=int(scint(n,2))
nod=ptable(fbindx,n,1)
else if(ty(1:3).eq.'QUA'.or.ty(1:3).eq.'qua')then
fbindx=int(quant(n,5))
nod=1
endif
ptable(fbindx,n,2)=tsadd
call fread
ij=3
if(.not.fin)then
ij=1
lsave=rfile
rfile=.true.
isave=nread
nread=10
open(unit=nread,file=str,status='OLD')
endif
c
c CHECK FOR BEGINNING OF TABLE
c
z=0
5 inerr=.false.
call input('T',nerr,ij)
call fread
c check to see if line is comment line
if(str(1:1).eq.'*')goto 5
if(str(1:3).ne.'BEG'.and.str(1:3).ne.'beg')then
write(ntype,1000)' ? "begin" expected ?'
inerr=.true.
endif
if(nod.ne.1)goto 94

```

```

call fread
if(str(1:3).ne.'INT'.and.str(1:3).ne.'int')then
  write(ntype,1000)' ? "interpolation=n" expected ?'
  inerr=.true.
endif
94  z=z+1
    if(z.eq.5)goto 2000
    if(inerr)goto 5
c
c  READ DISCRETE VARIABLES
c
    if(nod.eq.1)then
      call fread
      ptable(fbindx,n,3)=ival
    else
      ptable(fbindx,n,3)=1
    endif
    outpt=1
    do 10 j=1,nod
      count=0
15    call input('T',nerr,ij)
      call fread
      if(str(1:1).eq.'*')goto 15
      do 20 k=1,mxnumx
        call fread
        if(tsadd.gt.mxtele)then
          write(ntype,1020)
          write(nwrite,1020)
          goto 2000
        else if(.not.fin)then
          count=count+1
          tables(tsadd)=aval
          tsadd=tsadd+1
        else
          goto 30
        endif
20      continue
30      ptable(fbindx,n,j+3)=count
        outpt=outpt*count
10    continue
c
c  CHECK FOR AVAILABLE SPACE IN TABLE
c
    k=outpt+tsadd-1
    if(k.gt.mxtele)then
      write(ntype,1020)
      write(nwrite,1020)
      goto 2000
    endif
c
c  READ BLOCK OUTPUT FOR NOD-DIMENSIONAL TABLE
c
    do 100 i=nod+4,9
      ptable(fbindx,n,i)=1
100   continue
c
    do 250 ii=1,ptable(fbindx,n,9)
      do 240 m=1,ptable(fbindx,n,8)
        do 230 l=1,ptable(fbindx,n,7)
          do 220 k=1,ptable(fbindx,n,6)
            do 210 j=1,ptable(fbindx,n,5)
              call input('T',nerr,ij)
              do 200 i=1,ptable(fbindx,n,4)
                call fread
                if(fin)then
                  write(ntype,1000)' ? table too small '
                  write(nwrite,1000)' ? table too small '
                  ij=1
                  goto 2000
                endif
                tables(tsadd)=aval
                tsadd=tsadd+1
200      continue
            if(nod.eq.1)goto 500
210    continue

```

```

                if(nod.eq.2)goto 500
220                continue
                if(nod.eq.3)goto 500
230                continue
                if(nod.eq.4)goto 500
240                continue
                if(nod.eq.5)goto 500
250                continue
C
C   RESTORE UNUSED VARIABLE POINTERS TO ZERO
C
500                do 600 i=nod+4,9
                ptable(fbindx,n,i)=0
600                continue
C
C   CHECK FOR END OF INPUT
C
                call input('T',nerr,ij)
                call fread
                if(str(1:3).ne.'DON'.and.str(1:3).ne.'don')write(ntype,1000)' ? "done"
+   expected ? '
C
C   ERROR STATEMENTS
C
1000               format(' ','? illegal input format in table ?',a40)
1020               format(' ','? table input exceeds internal table storage ?')

2000               if(ij.eq.1)then
                close(nread)
                nread=isave
                rfile=lsave
                endif
                return
                end
C.....
C
C   ZANA
C
C   PURPOSE:
C       This subroutine simulates the circuit.
C
C   PARAMETERS:
C       x          : storage array for circuit node values
C       gen        : storage array for generator
C       scint      : storage array for integrator
C       quant      : storage array for quantizer
C       ptable     : pointer array for table look-up
C       tables     : array containing functional block values
C       ndpres    : acknowledges presence of circuit nodes
C       node       : contains present node values
C       nodep     : contains previous node values
C
C   SPECIAL VARIABLES:
C       stocnt    : initialize to 1; psuedo-cycle value for node storage
C       ix        : initial seed for psuedo-random number (ix < 0 )
C.....
C       subroutine ZANA(x,gen,scint,quant,ptable,tables,ndpres,
+                   node,nodep)

                integer cycle,nocyc,block,tsadd,mxnnode,circuit,nodes,ix
                integer mgen,pgen,mscint,pscint,mquant,pquant
                common/misc/ix
                common/blkmax/mgen,pgen,mscint,pscint,mquant,pquant
                common/parmax/mxblk,mxdim,mxdim3,mxtele,mxnnode,mxicyc,mxtype,mxnumx
                common/funblk/block,period,nocyc,tsadd,pi,freqpt,circuit
                real tables(mxtele),x(mxicyc,mxnnode),gen(mgen,pgen)
                real scint(mscint,pscint),quant(mquant,pquant)
                real node(mxnnode),nodep(mxnnode)
                integer ptable(mxtype,mxblk,mxdim3),stocnt
                common/sto/stocnt
                logical ndpres(mxnnode)
                logical cpypar,cpynod,over,clear
                common/commnd/cpypar,cpynod,over,clear

```

```

character*20 str
logical kf,fin,prt,ofile,rfile
common/frerdr/nwrite,ntype,aval,ival,str,ich,kf,fin,prt,ofile,rfile
integer topol
common/simtyp/topol
call fread
if(str(1:4).eq.'HELP')then
  if(topol.eq.1)then
    write(ntype,50)
    if(prt)write(nwrite,50)
50   format(
+     ' ##### TABLE SIMULATION #####',/,
+     ' Usage:',/,/,
+     '   SIM HELP',/,
+     '       : This Message',/,/,
+     '   SIM ',/,/,
+     '       : Simulate a circuit described by tables',/,/,
+     ' #####' )
  else if(topol.eq.0)then
    write(ntype,55)
    if(prt)write(nwrite,55)
55   format(
+     ' ##### DIFFERENCE EQUATION SIMULATION #####',/,
+     ' Usage:',/,/,
+     '   DES HELP',/,
+     '       : This Message',/,/,
+     '   DES ',/,/,
+     '       : Simulate a circuit described by',/,/,
+     '       : difference equations',/,/,
+     ' #####' )
    endif
    return
  endif
endif

c
c DETERMINE NUMBER OF NODES IN THE CIRCUIT
c
  if(circuit.eq.0)then
    nodes=2
  else if(circuit.eq.1)then
    nodes=3
  else if(circuit.eq.2)then
    nodes=4
  endif

c
c PERFORM THE SIMULATION
c
  stocnt=1
  ix=-7841
2   do 100 cycle=1,nocyc
c
c SIMULATE FIRST ORDER DELTA-SIGMA
c
c
  if(circuit.eq.1)then
    call AGEN(1,node,cycle,gen,f)
    if(topol.eq.0)then
      call DESIM(cycle,node,nodep,nodes)
    elseif(topol.eq.1)then
      call ASCINT(1,cycle,node,nodep,scint,ptable,tables)
      call AQUANT(1,node,nodep,cycle,quant)
    endif
c
c SIMULATE SECOND ORDER DELTA-SIGMA
c
  else if(circuit.eq.2)then
    call AGEN(1,node,cycle,gen,f)
    if(topol.eq.0)then
      call DESIM(cycle,node,nodep,nodes)
    elseif(topol.eq.1)then
      call ASCINT(1,cycle,node,nodep,scint,ptable,tables)
      call ASCINT(2,cycle,node,nodep,scint,ptable,tables)
      call AQUANT(1,node,nodep,cycle,quant)
    endif
c
c THIS IS TO TEST THE NOISE GENERATOR AND/OR TRI-LEVEL COMPARATOR

```

```

c
      else if(circuit.eq.0)then
        call AGEN(1,node,cycle,gen,f)
        call AQUANT(1,node,nodep,cycle,quant)
      endif
c
c STORE NODE VALUES
c
      if(cycle.gt.0)then
        do 99 ii=1,nodes
          x(cycle,ii)=node(ii)
99      continue
        endif
100     continue
c
c ACKNOWLEDGE PRESENCE OF NODES
c
      ndpres(1)=.true.
      ndpres(2)=.true.
      ndpres(3)=.true.
      if(circuit.eq.2)ndpres(4)=.true.
      if(circuit.eq.0)ndpres(3)=.false.
c SET EACH PREVIOUS NODE VALUE TO ZERO AFTER COMPLETION OF SIMULATION
      do 300 i=1,nodes
        nodep(i)=0.0
300     continue
      return
      end
c.....
c
c ZCALSDR
c
c PURPOSE:
c   This subroutine stores information about the input signal.
c
c PARAMETERS:
c   x      : circuit node signals <input>
c
c SUBROUTINES CALLED:
c   fread, calcdr
c.....
      subroutine ZCALSDR(x,nodein,nodout)

      common/parmax/mxblk,mxdim,mxdim3,mxtele,mxnnode,mxicyc,mxtype,mxnumx
      character*20 str
      integer nodein,nodout,block,nocyc,circuit,tsadd,freqpt
      real x(mxicyc,mxnnode),sampfq
      logical kf,fin,prt,ofile,rfile
      common/freerd/nwrite,ntype,aval,ival,
+         str,ich,kf,fin,prt,ofile,rfile
      data pi/3.141592654/
      data nin,nskip,nfft/0,0,0/
c
c READ SDR PARAMETERS
c
10    call fread
      if(fin)then
        if(nfft.ne.0.and.(nin.ge.(nskip+nfft)))then
          call calcdr(x(1,nodein),x(1,nodout),nin,nfft,nskip,sdr,snr,
+         sthd,prt,ntype,nwrite,iwind,iw,fb,sampfq,nfast,sigfq)
        else
          write(ntype,*)' ZANA - not enough samples for SDR calculation'
          write(ntype,*)'      nin = ',nin
          write(ntype,*)'      nskip = ',nskip
          write(ntype,*)'      nfft = ',nfft
          write(nwrite,*)' ZANA - not enough samples for SDR calculation'
        endif
        return
      elseif(str(1:6).eq.'FFTWIN')then
        call fread
        if(fin)goto 100
        if(str(1:4).eq.'UNIF')then
          iwind=0
        elseif(str(1:4).eq.'HAMM')then

```

```

        iwind=1
    elseif(str(1:4).eq.'HARR')then
        iwind=2
    else
        write(ntype,*)' ? Window type not recognized?'
        if(ofile)write(nwrite,*)' ? Window type not recognized?'
        return
    endif
elseif(str(1:6).eq.'DECWIN')then
    call fread
    if(fin)goto 100
    if(str(1:4).eq.'IGNO')then
        iw=0
    elseif(str(1:4).eq.'UNIF')then
        iw=1
    elseif(str(1:4).eq.'TRIA')then
        iw=2
    elseif(str(1:4).eq.'PARA')then
        iw=3
    else
        write(ntype,*)' ? Decimation window type not recognized?'
        if(ofile)write(nwrite,*)
+         ' ? Decimation window type not recognized?'
        return
    endif
endif
CGTB_START
C "SIGNAL" must be defined for a correct sdr calculation.
C "sigfq" is the frequency of the input signal in Hertz.
    elseif(str(1:6).eq.'SIGNAL')then
        call fread
        if(fin)goto 100
        sigfq=aval
c write(ntype,*)'read signal'
C Make a distinction between FS (sampling frequency) and FB (baseband
c frequency). FS is only needed when decimation occurs.
    elseif(str(1:2).eq.'FS')then
        call fread
        if(fin)goto 100
        sampfq=aval
CGTB_END
    elseif(str(1:2).eq.'FB')then
        call fread
        if(fin)goto 100
        fb=aval
    elseif(str(1:4).eq.'TAPS')then
        call fread
        if(fin)goto 100
        nfast=ival
    elseif(str(1:6).eq.'NPOINT')then
        call fread
        if(fin)goto 100
        nin=ival
    elseif(str(1:4).eq.'NFFT')then
        call fread
        if(fin)goto 100
        nfft=ival
    elseif(str(1:5).eq.'NSKIP')then
        call fread
        if(fin)goto 100
        nskip=ival
    else
        write(ntype,*)' ? illegal input format ?'
        if(ofile)write(nwrite,*)' ? illegal input format ?'
        return
    endif
    goto 10
c
c ERROR STATEMENTS
c
100 write(ntype,*)' ? Insufficient data to do SDR calculation ?'
    if(ofile)
+     write(nwrite,*)' ? Insufficient data to do SDR calculation ?'
    return
end
C.....

```

```

c
c
c      ZCMD
c
c
c      PURPOSE:
c          This subroutine controls the command sequence
c
c      COMMANDS:
c          simulate: perform the z-domain simulation
c          gen      : define generator input
c          scint    : define integrator input
c          quant    : define quantizer input
c          table    : define table to describe functional blocks
c          hist     : set table histogram flag
c          print    : set printing features
c          noise    : define gaussian noise
c          circuit  : define circuit type
c          env      : environment; define simulation features
c          clear    : erase all parameters to default
c          pause    : stop to perform printing features
c          stop     : end session
c          dump     : dump contents of stored node values
c          init     : initialize quantities
c          decimate : perform a decimation
c          sdr      : do Signal-to-Distortion ratio calculation
c
c      PARAMETERS:
c          x        : storage array for node values
c          gen      : informational array of generators
c          scint    : informational array of integrators
c          quant    : informational array of quantizers
c          ptable   : pointer to table values
c          tables   : vector of functional block values
c          genflg   : acknowledges presence of generators
c          intflg   : acknowledges presence of integrators
c          qntflg   : acknowledges presence of quantizers
c          ptnode   : should node value be stored
c          ndpres   : acknowledges presence of node
c
c      SUBROUTINES USED:
c          input, fread, gentor, intgtr, compar, init, zdec
c          nodset, zana, envirn, noiset, tablrd, dsmsdr
c
c.....
c      +
c          subroutine ZCMD(x,gen,scint,quant,ptable,tables,genflg,intflg,
c              qntflg,ptnode,ndpres,node,nodep)
c
c          integer mgen,pgen,mscint,pscint,mquant,pquant
c          integer tsadd,block,circuit,nerr
c          common/blkmax/mgen,pgen,mscint,pscint,mquant,pquant
c          common/parmax/mxblk,mxdim,mxdim3,mxtele,mxnode,mxicyc,mxtype,mxnumx
c          common/funblk/block,period,nocyc,tsadd,pi,freqpt,circuit
c          integer ptable(mxtype,mxblk,mxdim3)
c          real gen(mgen,pgen),scint(mscint,pscint),tables(mxtele)
c          real quant(mquant,pquant),x(mxicyc,mxnode)
c          real node(mxnode),nodep(mxnode)
c          logical ptnode(mxnode),ndpres(mxnode),genflg(mgen),intflg(mscint)
c          logical qntflg(mquant)
c          logical cpypar,cpynod,clear,over
c          common/commnd/cpypar,cpynod,over,clear
c          character*20 str
c          logical kf,fin,prt,ofile,rfile
c          common/freerd/nwrite,ntype,aval,ival,str,ich,kf,fin,prt,ofile,rfile
c          integer topol
c          common/simtyp/topol
c
c      CGTB_START
c          integer hist(5,50,50,3)
c          logical phist
c          common/histgm/phist,hist
c
c      CGTB_END
c
c      READ A LINE OF INPUT
c
c      10      call input(' ',nerr,3)
c             call fread
c             if(fin)goto 10

```

```

c
c  PROCESS INPUT ACCORDING TO FIRST FIELD OF INPUT LINE
c
      if(str(1:3).eq.'SIM'.or.str(1:3).eq.'DES')then
        if(str(1:3).eq.'DES')topol=0
        if(str(1:3).eq.'SIM')topol=1
        call ZANA(x,gen,scint,quant,ptable,tables,ndpres,node,
+           nodep,bitts)
        goto 1000
      else if(str(1:4).eq.'DUMP')then
        idump=9
        call ZDUMP(x,mxicyc,mxnode,idump)
      else if(str(1:4).eq.'HELP'.or.str(1:1).eq.'?')then
        call ZHELP
      else if(str(1:4).eq.'INIT')then
        call INIT(nodep)
      else if(str(1:2).eq.'EQ')then
        call ZEQUAT
      else if(str(1:3).eq.'DEC')then
        call ZDEC(x)
      else if(str(1:3).eq.'SDR')then
        call ZSDR(x)
      else if(str(1:3).eq.'GEN')then
        call GENTOR(gen,genflg)
      else if(str(1:3).eq.'SCI')then
        call INTGTR(scint,intflg,ptable)
      else if(str(1:3).eq.'QUA')then
        call COMPAR(quant,qntflg,ptable)
      else if(str(1:5).eq.'TABLE')then
        call TABLRD(scint,quant,ptable,tables)
CGTB_START
      else if(str(1:4).eq.'HIST')then
        phist=.true.
CGTB_END
      else if(str(1:5).eq.'PRINT')then
        call fread
        if(str(1:3).eq.'NOD')then
          call NODSET(ptnode)
        else if(str(1:3).eq.'PAR')then
          cpypar=.true.
          call fread
          if(str(1:3).eq.'OFF')cpypar=.false.
        else
          write(ntype,*)' ? argument expected for "print" ?'
        endif
      else if(str(1:5).eq.'NOISE')then
        call NOISET
      else if(str(1:3).eq.'ENV')then
        call ENVIRN
      else if(str(1:3).eq.'CIR')then
        call fread
        if(str(1:4).eq.'HELP')then
          write(ntype,50)
          if(prt)write(nwrite,50)
50      format(
+ ' ##### CIRCUIT #####',/,
+ ' Usage:',/,
+ '   CIRCUIT HELP',/,
+ '   : This Message',/,
+ '   CIRCUIT = n',/,
+ '   : Set order of modulator to "n"',/,
+ ' #####')
        else
          circuit=ival
        endif
      else if(str(1:5).eq.'CLEAR')then
        call fread
        if(str(1:4).eq.'HELP')then
          write(ntype,52)
          if(prt)write(nwrite,52)
52      format(
+ ' ##### CLEAR #####',/,
+ ' Usage:',/,
+ '   CLEAR HELP',/,
+ '   : This Message',/,

```

```

+ ' CLEAR ',/,
+ ' : Set everything to default value',/,
+ ' #####')
  else
    clear=.true.
    goto 1000
  endif
  else if(str(1:4).eq.'STOP')then
    call fread
    if(str(1:4).eq.'HELP')then
      write(ntype,54)
      if(prt)write(nwrite,54)
54 format(
+ ' ##### STOP #####',/,
+ ' Usage:',/,/,
+ ' STOP HELP',/,
+ ' : This Message',/,/,
+ ' STOP ',/,
+ ' : End ZSIM session',/,/,
+ ' #####')
    else
      over=.true.
      goto 1000
    endif
  else if(str(1:5).eq.'PAUSE')then
    call fread
    if(str(1:4).eq.'HELP')then
      write(ntype,56)
      if(prt)write(nwrite,56)
56 format(
+ ' ##### PAUSE #####',/,
+ ' Usage:',/,/,
+ ' PAUSE HELP',/,
+ ' : This Message',/,/,
+ ' PAUSE ',/,
+ ' : Stop simulation to perform printing features',/,/,
+ ' #####')
    else
      goto 1000
    endif
  elseif(str(1:1).eq.'*')then
    jj=1
  else
    write(ntype,2)str(1:10)
    if(prt)write(nwrite,2)str(1:10)
2 format(' ? unrecognizable command ? ',a10)
  endif
  goto 10
c
1000 return
end
c.....
c
c ZDEC
c
c PURPOSE:
c This subroutine calls the appropriate decimation routine
c
c PARAMETERS:
c x : array of node values
c
c CAPITALS INDICATE KEYWORDS AND THEIR MINIMUM LENGTH
c
c SUBROUTINES CALLED:
c fread, zbdec, gendec
c
c EXAMPLES:
c DECIMATE n1 n2 TYPE=ZBDEC
c This performs a decimation on the values of node n1
c and outputs the result (usually the baseband) at node
c n2. TYPE indicates the type of canned decimation.
c
c DECIMATE n1 n2 TYPE=GENDEC NUMBER=nn WINDOW=sss TAPS=nn
c INTDEC=nn BBANDDEC=bb FILE=filename FILTER=sssss
c This performs a decimation on the values of node n1

```

```

c          and outputs the result (usually the baseband) at node
c          n2. TYPE indicates the type of canned decimation.
c          NUMBER can be to base 2, e.g.: NUMBER=nn BASE 2
c          WINDOW's available: UNIFORM TRIANGLE PARABOLIC
c          TAPS : length of the decimation filter
c          INTDEC: FIR decimation factor
c          BBANDDEC : baseband decimation factor
c          FILE specifies file with equalizer coefficients
c          FILTER's available: NONE DIRECT_form
c                          NORMALized_lattice CAScade_form
c
c          Available types are
c              ZBDEC
c              GENDEC
c          See the routine of that name for extra parameters
c
c.....
c          subroutine zdec(x)
c
c          common/parmax/mxblk,mxdim,mxdim3,mxtele,mxnnode,mxicyc,mxtype,mxnumx
c          real x(mxicyc,mxnnode),period,pi
c          integer circuit
c          integer block,nocyc,tsadd,mgen,pgen,mscint,pscint,mquant,pquant
c          common/funblk/block,period,nocyc,tsadd,pi,freqpt,circuit
c          common/file/fileno
c          integer fileno,nodein,nodout,nout
c          character*20 str
c          logical kf,fin,prt,ofile,rfile
c          common/freerd/nwrite,ntype,aval,ival,
c          +          str,ich,kf,fin,prt,ofile,rfile
c
c          PROCESS REMAINDER OF INPUT LINE
c
c          10      call fread
c          if(fin)goto 100
c          if(str(1:4).eq.'HELP')then
c              write(ntype,50)
c              if(prt)write(nwrite,50)
c          50      format(
c          + ' ##### DECIMATE #####',/,
c          + ' Usage:',/,
c          + '   DEC HELP',/,
c          + '   : This Message',/,
c          + '   DECIMATE n1 n2 TYPE=ZBDEC',/,
c          + '   This performs a decimation on the values of node n1',/,
c          + '   and outputs the result (usually the baseband) at n2',/,
c          + '   TYPE indicates the type of canned decimation.',/)
c          write(ntype,55)
c          if(prt)write(nwrite,55)
c          55      format(
c          + '   DECIMATE n1 n2 TYPE=GENDEC NUMBER=nn WINDOW=sss TAPS=nn',/,
c          + '   INTDEC=nn BBANDDEC=bb FILE=filename FILTER=sss',/,
c          + '   This performs a decimation on the values of node n1',/,
c          + '   and outputs the result (usually the baseband) at n2',/,
c          + '   NUMBER can be to base 2, e.g.: NUMBER=nn BASE 2',/,
c          + '   WINDOWs available: UNIFORM TRIANGLE PARABOLIC',/,
c          + '   TAPS : length of the decimation filter',/,
c          + '   INTDEC: FIR decimation factor',/,
c          + '   BBANDDEC : baseband decimation factor',/,
c          + '   FILE specifies file with filter coefficients',/,
c          + '   FILTERs available: NONE DIRECT_form',/,
c          + '   NORMALized_lattice CAScade_form',/,
c          + ' #####')
c          return
c          endif
c
c          GET NODE NUMBERS
c
c          if(.not.kf)then
c              nodein=ival
c          else
c              write(ntype,*) '? node number required ?'
c              if(ofile) write(nwrite,*)
c          + ' ? node number required ?'
c              goto 100

```

```

endif
call fread
if(fin)goto 100
if(.not.kf)then
  nodout=ival
else
  write(ntype,*) '? node number required ?'
  if(ofile) write(nwrite,*)
+   '? node number required ?'
  goto 100
endif

C
C DETERMINE WHICH TYPE OF DECIMATION TO USE
C
  call fread
  if(fin.or..not.kf)goto 100
  if(str(1:4).eq.'TYPE')then

C
C FIND OUT WHICH TYPE OF DECIMATION IS TO BE USED AND CALL APPROPRIATE
C ROUTINE
C
  call fread
  if(fin.or..not.kf)goto 100
  if(str(1:6).eq.'GENDEC')then
    fs=1./period
    call gendec(x(1,nodein),x(1,nodout),fs)
  elseif(str(1:5).eq.'ZBDEC')then
    call zbdec(x(1,nodein),x(1,nodout),nocyc)
  else
    write(ntype,*)' Decimation type unknown'
    if(ofile) write(nwrite,*)' Decimation type unknown'
    goto 100
  endif
  else
    goto 100
  endif
  return

C
C ERROR STATEMENTS
C
100 write(ntype,*)' ? Decimation Error'
   if(ofile) write(nwrite,*)' ? Decimation Error'
   return
   end

C.....
C
C ZDUMP
C
C PURPOSE:
C   To write stored node values to a file.
C
C PARAMETERS:
C   x - Array containing stored node values
C   mxicyc - first dimension of x
C   mxnode - second dimension of x
C   idump - l.u.n. of output file
C
C USAGE:
C   DUMP file n no
C
C           dump first 'no' values at node 'n' to 'file'
C.....
  subroutine zdump(x,mxicyc,mxnode,idump)
  integer mxicyc,mxnode,idump,ntype
  character*20 file
  real x(mxicyc,mxnode)

C
  character*20 str
  logical kf,fin,prt,ofile,rfile
  common/freerd/nwrite,ntype,aval,ival,str,ich,kf,fin,prt,ofile,rfile

C
  call fread
  if(fin)then
    write(ntype,*)' ZDUMP - input error, no file specified, try again'

```

```

        return
    endif
CGTB_START
    if(str(1:4).eq.'HELP')then
        write(ntype,50)
        if(prt)write(nwrite,50)
50      format(
+      ' ##### DUMP #####',//,
+      ' Usage:',//,
+      '   DUMP HELP',//,
+      '       : This Message',//,
+      '   DUMP file n no',//,
+      '       : Dump first "no" values at node "n" to "file"',//,
+      ' #####')
        return
    endif
CGTB_END
    file=str(1:20)
    write(ntype,3)file
    write(nwrite,3)file
3      format('          DUMP file: ',a20,'')
    open(unit=idump,file=file,status='UNKNOWN')
    call fread
    n=ival
    if(n.gt.mxnode)then
        write(ntype,*) 'Incorrect Node Number specified for DUMP'
        return
    endif
    if(fin)then
c      close(unit=idump)
        write(ntype,*) 'ZDUMP - input error, number not specified, try again'
        return
    endif
    call fread
    if(fin)then
c      close(unit=idump)
        write(ntype,*) 'ZDUMP - input error, no node specified, try again'
        return
    endif
    no=ival
    write(ntype,10)no,n
    if(ofile)write(nwrite,10)no,n
10      format('          no. values output = ',i10,' node number = '
+      ',i10)
    write(idump,20) (x(i,n),i=1,no)
20      format(' ',g15.6)
c      close(unit=idump)
    return
    end
c.....
c
c      ZEQUAT
c
c      PURPOSE:
c          This subroutine inputs the parameters for a simulation of
c          an ideal difference equation circuit.
c
c      PARAMETERS:
c          x      : array of node values
c
c      CAPITALS INDICATE KEYWORDS AND THEIR MINIMUM LENGTH
c
c      SUBROUTINES CALLED:
c          fread
c
c      EXAMPLES:
c          EQ GAIN#=xxx1 DELTA#=xxx2 OFFSET#=xxx3 FGBW#=xxx4
c              OFINT#=xxx5 SAT#%=xxx6
c          This inputs ideal circuit parameters:
c          # defines which integrator (remove # to refer to all
c          integrators with one command)
c          GAIN# : gain of respective integrator
c          DELTA# : reference voltage of quantizer
c          OFFSET# : offset of quantizer
c          FGBW# : finite gain bandwidth of integrator

```

```

C          OFINT#: offset of the integrator referred to the output
C          SAT#% : integrator saturation
C                  (% is optional: replace with
C                  '+' for upper limit and
C                  '-' for lower limit)
C.....
C          subroutine zequat
C          real mindel,maxdel,offset
C          real a1,a2,a3,fgbw1,fgbw2,fgbw3
C          real delta1,delta2,delta3,ofint1,ofint2,ofint3
C          real satmn1,satmx1,satmn2,satmx2,satmn3,satmx3
C          real aa1,aa2,aa3,b1,b2,b3,c1,c2,c3
C          common/diffeq/a1,a2,a3,fgbw1,fgbw2,fgbw3,
+          delta1,delta2,delta3,
+          ofint1,ofint2,ofint3,
+          satmn1,satmx1,satmn2,satmx2,satmn3,satmx3,
+          maxdel,mindel,offset,
+          aa1,aa2,aa3,b1,b2,b3,c1,c2,c3
C          integer block,nocyc,tsadd,circuit,freqpt
C          real period
C          common/funblk/block,period,nocyc,tsadd,pi,freqpt,circuit
C
C          character*20 str
C          logical kf,fin,prt,ofile,rfile
C          common/freerd/nwrite,ntype,aval,ival,
+          str,ich,kf,fin,prt,ofile,rfile
C
C          set values for infinite gain bandwidth product
C
C          data aa1,aa2,aa3/1.0,1.0,1.0/
C          data b1,b2,b3/0.0,0.0,0.0/
C          data c1,c2,c3/0.0,0.0,0.0/
C
C          PROCESS REMAINDER OF INPUT LINE
C
C          call fread
C          if(fin)goto 100
C          if(str(1:4).eq.'HELP')then
C          write(ntype,50)
C          if(prt)write(nwrite,50)
50          format(
+          ' ##### EQUATION #####',/,
+          ' Usage:',/,
+          ' EQ HELP',/,
+          ' : This Message',/,/,
+          ' EQ GAIN#=xxx1 DELTA#=xxx2 FGBW#=xxx3',/,
+          ' : OFINT#=xxx4 SAT#%=xxx5',/,
+          ' : # = 1, 2, or 3 for respective integrator but may be',/,
+          ' : left out to describe all integrators.',/,
+          ' : This inputs ideal circuit parameters:',/,
+          ' : GAIN# : gain of integrator',/,
+          ' : DELTA# : reference voltage of quantizer',/,
+          ' : FGBW# : finite gain bandwidth of integrator',/,
+          ' : OFINT# : integrator offset referred to the output',/,
+          ' : SAT#% : integrator saturation (% is optional flag -',/,
+          ' : : % = '+' for upper limit ',/,
+          ' : : % = '-' for lower limit ',/,/,
+          ' #####')
C          return
C          endif
C
C          READ EQUATION PARAMETERS
C
C          if(fin)then
C          maxdel=1
C          mindel=-1
C          fs=1./period
C          if(fgbw1.ne.0.0) call calfbw(fgbw1,fs,a1,aa1,b1,c1)
C          if(fgbw2.ne.0.0) call calfbw(fgbw2,fs,a2,aa2,b2,c2)
C          if(fgbw3.ne.0.0) call calfbw(fgbw3,fs,a3,aa3,b3,c3)
C          return
C          elseif(str(1:5).eq.'GAIN1')then
C          call fread
C          if(fin)goto 100

```

```

    a1=aval
elseif(str(1:5).eq.'GAIN2')then
    call fread
    if(fin)goto 100
    a2=aval
elseif(str(1:5).eq.'GAIN3')then
    call fread
    if(fin)goto 100
    a3=aval
elseif(str(1:4).eq.'GAIN')then
    call fread
    if(fin)goto 100
    a1=aval
    a2=aval
    a3=aval
elseif(str(1:5).eq.'FGBW1')then
    call fread
    if(fin)goto 100
    fgbw1=abs(aval)
elseif(str(1:5).eq.'FGBW2')then
    call fread
    if(fin)goto 100
    fgbw2=abs(aval)
elseif(str(1:5).eq.'FGBW3')then
    call fread
    if(fin)goto 100
    fgbw3=abs(aval)
elseif(str(1:4).eq.'FGBW')then
    call fread
    if(fin)goto 100
    fgbw1=abs(aval)
    fgbw2=abs(aval)
    fgbw3=abs(aval)
CGTB_START
elseif(str(1:5).eq.'SAT1+')then
    call fread
    if(fin)goto 100
    satmx1=abs(aval)
elseif(str(1:5).eq.'SAT1-')then
    call fread
    if(fin)goto 100
    satmn1=-1.0*abs(aval)
elseif(str(1:5).eq.'SAT2+')then
    call fread
    if(fin)goto 100
    satmx2=abs(aval)
elseif(str(1:5).eq.'SAT2-')then
    call fread
    if(fin)goto 100
    satmn2=-1.0*abs(aval)
elseif(str(1:5).eq.'SAT3+')then
    call fread
    if(fin)goto 100
    satmx3=abs(aval)
elseif(str(1:5).eq.'SAT3-')then
    call fread
    if(fin)goto 100
    satmn3=-1.0*abs(aval)
elseif(str(1:4).eq.'SAT+')then
    call fread
    if(fin)goto 100
    satmx1=abs(aval)
    satmx2=abs(aval)
    satmx3=abs(aval)
elseif(str(1:4).eq.'SAT-')then
    call fread
    if(fin)goto 100
    satmn1=-1.0*abs(aval)
    satmn2=-1.0*abs(aval)
    satmn3=-1.0*abs(aval)
CGTB_END
elseif(str(1:4).eq.'SAT1')then
    call fread
    if(fin)goto 100
    satmx1=abs(aval)

```

```

    satmn1=-1.0*satmx1
elseif(str(1:4).eq.'SAT2')then
    call fread
    if(fin)goto 100
    satmx2=abs(aval)
    satmn2=-1.0*satmx2
elseif(str(1:4).eq.'SAT3')then
    call fread
    if(fin)goto 100
    satmx3=abs(aval)
    satmn3=-1.0*satmx3
elseif(str(1:3).eq.'SAT')then
    call fread
    if(fin)goto 100
    satmx1=abs(aval)
    satmn1=-1.0*satmx1
    satmx2=abs(aval)
    satmn2=-1.0*satmx2
    satmx3=abs(aval)
    satmn3=-1.0*satmx3
elseif(str(1:6).eq.'DELTA1')then
    call fread
    if(fin)goto 100
    delta1=abs(aval)
elseif(str(1:6).eq.'DELTA2')then
    call fread
    if(fin)goto 100
    delta2=abs(aval)
elseif(str(1:6).eq.'DELTA3')then
    call fread
    if(fin)goto 100
    delta3=abs(aval)
elseif(str(1:5).eq.'DELTA')then
    call fread
    if(fin)goto 100
    delta1=abs(aval)
    delta2=abs(aval)
    delta3=abs(aval)
elseif(str(1:6).eq.'OFINT1')then
    call fread
    if(fin)goto 100
    ofint1=abs(aval)
elseif(str(1:6).eq.'OFINT2')then
    call fread
    if(fin)goto 100
    ofint2=abs(aval)
elseif(str(1:6).eq.'OFINT3')then
    call fread
    if(fin)goto 100
    ofint3=abs(aval)
elseif(str(1:5).eq.'OFINT')then
    call fread
    if(fin)goto 100
    ofint1=abs(aval)
    ofint2=abs(aval)
    ofint3=abs(aval)
elseif(str(1:6).eq.'OFFSET')then
    call fread
    if(fin)goto 100
    offset=aval
    write(notype,*)'OFFSET currently hardwired to 0.0'
    if(prt)write(nwrite,*)'OFFSET currently hardwired to 0.0'
else
    write(notype,*)' ? illegal input format ?'
    if(ofile)write(nwrite,*)' ? illegal input format ?'
    return
endif
call fread
goto 10

```

c

c ERROR STATEMENTS

c

```

100 write(notype,*)' ? Insufficient data to do ideal simulation ?'
    if(ofile)
+   write(nwrite,*)' ? Insufficient data to do ideal simulation ?'

```

```

        return
    end
C.....
C
C    ZHELP
C
C    PURPOSE:
C        Produces a HELP Description
C
C.....
    subroutine zhelp

        character*20 str
        logical kf,fin,prt,ofile,rfile
        common/freerd/nwrite,ntype,aval,ival,str,ich,kf,fin,prt,ofile,rfile
        write(ntype,50)
50    format(
+ ' ##### HELP #####',/,
+ ' Commands Available:',/,
+ ' COMMAND      KEYWORD  DESCRIPTION',/,
+ ' SIMULATE     SIM      Perform z-domain simulation (tables)',/,
+ ' DESIMULATE   DES      Perform z-domain simulation (diff.eq.)',/,
+ ' GEN          GEN      Define generator input',/,
+ ' SCINT        SCINT    Define integrator input',/,
+ ' QUANT        QUANT    Define quantizer input',/,
+ ' TABLE       TABLE   Define table description',/,
+ ' NOISE        NOISE    Define gaussian noise',/,
+ ' CIRCUIT      CIRC     Define circuit type',/,
+ ' ENVIRONMENT  ENV      Define simulation features',/,
+ ' CLEAR        CLEAR    Erase all parameters to default',/,
+ ' DUMP         DUMP     Dump contents of stored node values',/,
+ ' TYPE "MORE" FOR MORE')
        call input(' ',nerr,1)
        write(ntype,70)
70    format(
+ ' STOP         STOP     End session',/,
+ ' INIT         INIT     Initialize quantities',/,
+ ' DECIMATE     DEC      Perform a decimation',/,
+ ' SDR          SDR      Signal-to-Distortion calculation',/,
+ ' EQUATION     EQ       Specify difference equation parameters',/,
+ ' Help on individual commands is available by typing',/,
+ '              "COMMAND" HELP',/,
+ ' TYPE "MORE" FOR MORE')
        call input(' ',nerr,1)
        write(ntype,20)
20    format(
+ ' ##### HELP #####',/,
+ ' The following input/output commands are available:',/,
+ ' read filename : the file filename is opened for input',/,
+ ' read          : the previously opened file is now used for',/,
+ '               input',/,
+ ' write filename : the file filename is opened for output',/,
+ ' eof           : the input data file is closed',/,
+ ' end          : the input data file is temporarily closed',/,
+ ' title        : write line to output file',/,
+ ' prompt       : write line to terminal',/,
+ ' echo on      : the flag prt is set',/,
+ ' echo off     : the flag prt is cleared',/,
+ '              If prt is set all lines input are echoed.',/,
+ ' #####')
        return
    end
C.....
C
C    ZINPUT
C
C    This subroutine reads in a line of length nchars into an array line
C    and prints it out again if reading from a file and print is specified,
C    and the line is converted to upper case.
C    line is then operated on by the free-read routine fread.
C
C    iopt = 1
C
C    iopt = 2      ; initialize device unit numbers
C

```

```

c      iopt = 3
c      If a line is
c          read filename ; the file filename is opened for input
c                          and the first line is read
c          read chbuff   ; the character buffer is opened for
c                          and subsequent input is taken from it
c          read          ; the previously opened file is now used for
c                          input
c          write filename ; the file filename is opened for output
c          eof           ; the input data file is closed
c          end           ; the input data file is temporarily closed
c          title         ; write line to output file
c          prompt        ; write line to terminal
c          echo on       ; the flag prt is set
c          echo off      ; the flag prt is cleared
c                          If prt is set all lines input are echoed.
c      In all the above cases the next line is read and returned (unless
c      it is one of the above lines).
c
c      iopt = 4          ;rewind - next time input is called no new line is
c                          read in. The effect is to rescan the line.
c
c      Routines required:
c      iopt      routines
c      1         fread getlin
c      2         -
c      3         fread getlin readop
c      4         fread
c
c*****
c
c      common block chbuff
c
c      This contains a character buffer 'cbuff' which usually contains
c      commands generated by the program and to be read as normal input.
c      'cbuff' is treated as just another file and so must follow the
c      conventions of files (ie eof, end etc.). Each record (line of input)
c      must be separated either by a 'line-feed' or a 'carriage return, line-
c      feed pair'.
c      lcwrit - index of last character written (goes from 1,2,3,..infinity)
c      lcread - index of last character read (goes from 1,2,3,.....infinity)
c      lcwrit-lcread must not be > nochar
c      nochar - the dimension of cbuff
c      bufflg=true routine takes input from cbuff
c      bufflg=false routine takes input as determined by the flag 'rfile'
c
c.....
c      subroutine input(prompt,nerr,iopt)
c      common/nxt/llp,kount,nchars
c      logical bufflg
c      character*1 cbuff,prompt
c      common/chbuff/nochar,lcwrit,lcread,bufflg,cbuff(500)
c      common/three/line(500)
c      common/iostr/nread,nprint,ifile,wfile,
+   ntti,nwrt,ndski,ndsko,infile,outfil
c      logical ifile,wfile
c      character*1 line
c      character*25 infile,outfil
c      character*4 readc,writec,eof,end,title,echo,on,off,prompt
c      logical rewind
c
c      character*20 str
c      logical kf,fin,prt,ofile,rfile
c      common/freerd/nwrite,ntype,aval,ival,str,ich,kf,fin,prt
+   ,ofile,rfile
c
c      data rewind /.false./
c      data nochar/500/
c      data readc,writec,eof,end,title,echo,on,off,prompt/'READ',
+   'WRIT','EOF','END','TITL','ECHO','ON','OFF','PROM'/
c*****
c      if(iopt.eq.1)then
c          rewind=.false.

```

```

        if(rewind)then
            kount=0
            lp=0
            rewind=.false.
        else
            call getlin(prompt,nerr)
        endif
        return
c****
elseif(iopt.eq.2)then
c     initialize unit numbers
        ntti=5
        nread=ntti
        ntype=6
c     input file
        ndski=7
c     output file
        ndsko=8
        nprint=ndsko
        nwrite=ndsko
c     rewind(nwrite)
        nwrt=ndsko
c     flags
        rfile=.false.
        wfile=.false.
        ifile=.false.
        return
c****
elseif(iopt.eq.3)then
30    if(rewind)then
            kount=0
            lp=0
            rewind=.false.
        else
            call getlin(prompt,nerr)
        endif
        if(nerr.ne.0)then
            call endin(1)
            goto 30
        endif
c     check for key words
        call fread
        if(str(1:4).eq.'TITL'.or.str(1:4).eq.'titl')then
            if(.not.prt)then
70         write(nwrite,70)prompt,(line(i),i=1,nchars)
                format(' ',a1,' ',120a1,/,(' ',120a1))
            endif
        elseif(str(1:4).eq.'prom'.or.str(1:4).eq.'PROM')then
75         write(ntype,75)(line(i),i=1,nchars)
                format(' ',500a1)
        elseif(str(1:4).eq.'READ'.or.str(1:4).eq.'read')then
            call readop
        elseif(str(1:5).eq.'write'.or.str(1:5).eq.'WRITE')then
            call writop
        elseif(str(1:3).eq.'eof'.or.str(1:3).eq.'EOF')then
            call endin(1)
        elseif(str(1:3).eq.'end'.or.str(1:3).eq.'end')then
            call endin(2)
        elseif(str(1:4).eq.'echo')then
            call fread
            if(str(1:2).eq.'on'.or.str(1:2).eq.'ON')then
                prt=.true.
            elseif(str(1:3).eq.'off'.or.str(1:3).eq.'OFF')then
                prt=.false.
            else
                write(ntype,*)' Parameter error for ECHO'
            endif
        else
            llp=0
            kount=0
            fin=.false.
            call upcase(line,nchars)
            return
        endif
c****

```

```

        elseif(iopt.eq.4)then
            rewind=.true.
            return
        endif
c****
        goto 30
    end
c.....
c
c     GETLIN
c
c     get a line of input which is subsequently read using fread
c
c     This version reads 500 character lines.
c.....
        subroutine getlin(prompt,nerr)
            character*1 prompt
            character*1 line(500)
            common/nxt/lp,kount,nchars
            common/three/line
            common/iostr/nread,nprint,ifile,wfile,
+            ntti,nwrt,ndski,ndsko,infile,outfil
            character*25 infile,outfil
            logical ifile,wfile
            character*1 blank,char
            logical bufflg
            character*1 cbuff,dollar
            common/chbuff/nochar,lcwrit,lcread,bufflg,cbuff(500)
c
            character*20 str
            logical kf,fin,prt,ofile,rfile
            common/freerd/nwrite,ntype,aval,ival,str,ich,kf,fin,prt
+            ,ofile,rfile
c
            data blank,dollar/' ','$'/
c
            kount=0
            lp=0
            nerr=0
            if(.not.bufflg)goto 500
c
            get line from internal character buffer
c
c
c     nchars=0
c     if(lcread-lcwrit)30,1300,1300
c     lcread=lcread+1
c     call bufdmp(nwrite)
            index=mod(lcread,nochar)+1
            if(cbuff(index).eq.dollar)goto 40
            nchars=nchars+1
            line(nchars)=cbuff(index)
            goto 20
c
c     40     if(nchars.eq.0)goto 10
            goto 1100
c
c
c     500     if(.not.rfile) write(ntype,600)
c     600     format(1x)
c     700     continue
            if(rfile) go to 900
            write(ntype,800) prompt
c
c     800     format(' ',a1,' ', '$')
c
c     read in a line
c
c     900     read(nread,1000,end=1300,err=1500) line
c     1000    format(500a1)
c
c     get length of string
c
c
c     do 1050 n=0,499
            nstore=n
            if(line(500-n).ne.blank)goto 1060
c
c     1050    continue
            nstore=500

```

```

1060      nchars=500-nstore
        if(nchars.eq.0) go to 700
1100      if(prt)write(nwrite,1200) prompt,(line(j),j=1,nchars)
1200      format(' ',a1,' ',80a1/,(' ',80a1))
        return
c
c      end of file
c
1300      write(ntype,1400)
1400      format(' ??input-f-hard eof ??')
        nerr=1
        return
c
c      err
c
1500      write(ntype,1600)
1600      format(' ??input-f-hard err ??')
        nerr=2
        return
        end
c.....
c
c      FREAD
c
c      this is the free read subroutine (originally from equi)
c      str      actual field read. max length=20
c      ich      number of characters in str
c
c.....
        subroutine fread
        character*1 x,iblack,iequ,icomma,num(3),ie
        logical oflow
        character*1 line(500)
        character*1 k0,k9,tab
        common /nxt/ lp,kount,nchars
        common/three/line
c
c      character*20 str
        logical kf,fin,prt,ofile,rfile
        common/freerd/nwrite,ntype,aval,ival,str,ich,kf,fin,prt
+      ,ofile,rfile
c
c      data iblack,icomma,iequ,k0,k9,ie /' ',' ',' ','='','0','9','e'/
        data num /'+','-','.'/
        data tab/' '/
c
        ntype=6
        kf=.false.
        oflow=.false.
        fin=.false.
5      lp=lp+1
        if(lp.gt.nchars)go to 200
        x=line(lp)
c      ignore separators
        if(x.eq.iblack.or.x.eq.tab
+      .or. x.eq.icomma.or.x.eq.iequ)go to 5
        if(x .ge. k0 .and. x .le. k9)go to 20
        do 10 i=1,3
10      if(x .eq. num(i))go to 20
c
c      *** here for alpha arg
c
        kf=.true.
        go to 30
c
c      *** here for numeric arg
c
20      kf=.false.
30      ich=0
        str='
c
c      *** here to get characters
c
c      write(0,35)lp,nchars
c35      format('lp,nchars=',2i10)

```

```

        lpold=lp
        do 50 lp=lpold,nchars
        ich=ich+1
c       write(0,39)lp,ich
c39      format('lp,ich=',2i10)
        x=line(lp)
        if(lp.gt.nchars.or.x.eq.tab
+       .or.x.eq.icomma.or.x.eq.iequ) go to 120
c       blanks do not matter
        if(x.eq.iblank.and.kf) go to 120
        if(x.eq.iblank.and.line(lp-1).ne.ie) go to 120
45      if(ich.gt.20) oflow=.true.
        if(.not.oflow)str(ich:ich)=x
c       write(0,49)str
c49      format('str=',a20)
50      continue
        ich=ich+1
120     if(ich.gt.20) oflow=.true.
c
c *** here on end of str(1:4)
c
        kount=kount+1
        if(oflow)go to 160
c       if(kf) write(0,1115) str(1:4)
c1115    format(' str(1:4) = ',a4)
c       if alphanumeric return
125     if(kf) return
126     str(ich:ich)=icomma
c       write(0,49)str
c       if numeric decode
c       need to implement test for valid number
        read(str,150)aval
150     format(g20.0)
c       write(0,1111) aval
c1111    format(' fread-aval = ',g20.8)
        if(aval.gt.2.147e9 .or. aval.lt.-2.147e9 )go to 220
c       create a chopped integer
        ival=ifix(aval)
        return
c
c *** here if arg too long
c
160     write(ntype,180)str
180     format(1x,a20,' field width overflow ;
+       trailing characters lost')
        go to 125
c
c *** here on end of line
c
200     fin=.true.
210     aval=0.
        daval=0.
220     ival=0
        str(1:20)='
        return
250     write(ntype,255)
255     format(' ?fread-w-input conversion error?')
        fin=.true.
        go to 210
        end
c.....
c
c       UPCASE
c
c       PURPOSE:
c       Convert a string of characters to uppercase
c
c       PARAMETERS
c       line      -string of characters
c       nchars   -number of characters in line
c.....
c       subroutine upcase(line,nchars)
c       character*1 line(nchars),char
c

```

```

c      transliterate to UPPER CASE
c
do 10 n=1,nchars
  char=line(n)
  if(char.eq.'a')then
    line(n)='A'
  elseif(char.eq.'b')then
    line(n)='B'
  elseif(char.eq.'c')then
    line(n)='C'
  elseif(char.eq.'d')then
    line(n)='D'
  elseif(char.eq.'e')then
    line(n)='E'
  elseif(char.eq.'f')then
    line(n)='F'
  elseif(char.eq.'g')then
    line(n)='G'
  elseif(char.eq.'h')then
    line(n)='H'
  elseif(char.eq.'i')then
    line(n)='I'
  elseif(char.eq.'j')then
    line(n)='J'
  elseif(char.eq.'k')then
    line(n)='K'
  elseif(char.eq.'l')then
    line(n)='L'
  elseif(char.eq.'m')then
    line(n)='M'
  elseif(char.eq.'n')then
    line(n)='N'
  elseif(char.eq.'o')then
    line(n)='O'
  elseif(char.eq.'p')then
    line(n)='P'
  elseif(char.eq.'q')then
    line(n)='Q'
  elseif(char.eq.'r')then
    line(n)='R'
  elseif(char.eq.'s')then
    line(n)='S'
  elseif(char.eq.'t')then
    line(n)='T'
  elseif(char.eq.'u')then
    line(n)='U'
  elseif(char.eq.'v')then
    line(n)='V'
  elseif(char.eq.'w')then
    line(n)='W'
  elseif(char.eq.'x')then
    line(n)='X'
  elseif(char.eq.'y')then
    line(n)='Y'
  elseif(char.eq.'z')then
    line(n)='Z'
  endif
10  continue
      return
      end
c.....
c
c      READOP
c
c      This file contains a number of file handling routines
c      open a file for reading
c
c.....
      subroutine readop
      common/nxt/llp,kount,nchars
      logical bufflg
      character*1 cbuff
      character*4 fill13,fill1
      common/chbuff/nochar,lcwrit,lcread,bufflg,cbuff(500)
      common/iostr/nread,nprint,ifile,wfile,

```

```

+   ntti,nwrt,ndski,ndsko,infile,outfil
    logical ifile,wfile
    character*25 infile,outfil
    character*1 lblank
c
    character*20 str
    logical kf,fin,prt,ofile,rfile
+   common/freerd/nwrite,ntype,aval,ival,str,ich,kf,fin,prt
    ,ofile,rfile
c
    data lblank/' '/
    data fill13,fill1/'CHBU','PRIN'/
c
c**** read instruction
c
10   call fread
    if(str(1:4).ne.fill13)goto 20
    bufflg=.true.
    goto 140
20   if(str(1:4).eq.' ' .or.str(1:4).eq.fill1) go to 90
    infile=str//'
80   call fread
    if(ifile) close(unit=ndski)
    ifile=.true.
    ista=0
    open(unit=ndski,file=infile,status='OLD',iostat=ista)
    if(ista.ne.0)goto 120
    rewind(unit=ndski)
c
c
90   if(ifile.and.prt) write(nwrite,100) infile
    if(ifile) write(ntype,100) infile
100  format(/,' ',a25,'')
110  if(.not.ifile) go to 120
    rfile=.true.
    ifile=.true.
    nread=ndski
    goto 140
120  if(prt)write(nwrite,130)
    write(ntype,130)
130  format(' ??no input file open ??')
140  return
    end
c.....
c
c           WRITOP
c
c       Open a file for output
c.....
    subroutine writop
    character*4 fill1,fill6,fill5
    common/nxt/llp,kount,nchars
+   common/iostr/nread,nprint,ifile,wfile,
    ntti,nwrt,ndski,ndsko,infile,outfil
    logical ifile,wfile
    character*25 infile,outfil
    character*1 lblank
    character*24 date
c
    character*20 str
    logical kf,fin,prt,ofile,rfile
+   common/freerd/nwrite,ntype,aval,ival,str,ich,kf,fin,prt
    ,ofile,rfile
c
    data lblank/' '/
    data fill1,fill6,fill5/'prin','lp ','tty '/
c
c**** write instruction
c
10   continue
    call fread
    if(str(1:4).eq.fill1.or.str(1:4).eq.fill6) go to 170
    if(str(1:4).eq.fill5) go to 180
    if(str(1:4).eq.' ' .and.ofile) go to 130
    if(str(1:4).eq.' ') go to 150

```

```

        outfil=str//'      '
c      if(.not.ofile) go to 110
c90    write(ndsko,100)
c100   format(1x)
        open(unit=ndsko,file=outfil,status='UNKNOWN',iostat=ista)
        rewind(unit=ndsko)
        write(ndsko,140) outfil
c *****
c      SYSTEM SPECIFIC LINE
c
c      for a UNIX system
c      call fdate(date)
c
c *****
        write(ndsko,140) date
        write(nwrite,*)' '
        write(nwrite,*)' ZSIM version ZSIM0a2'
        write(nwrite,*)' Nonlinear Z Domain Simulator'
        write(nwrite,*)' Brauns, Steer, Ardalan, ECE Dept'
        write(nwrite,*)' North Carolina State University'
        write(nwrite,*)' '
130    ofile=.true.
        wfile=.true.
        nwrite=ndsko
        nprint=nwrite
        write(ntype,140) outfil
140    format(' ',a25,' ',/)
        go to 190
150    write(ntype,160)
160    format(' ??no output file open ??')
        go to 190
170    wfile=.false.
        nwrite=nwrt
        nprint=nwrt
        go to 190
180    wfile=.false.
        nwrite=ntype
        nprint=nwrt
        go to 190
190    return
        end
c.....
c
c      ENDIN
c
c      iopt=1 close input file
c      if input was from a file take next line from the terminal
c      if input was from the internal character buffer take the next
c      line of input from the previous input medium.
c      iopt=2 close input file temporarily
c
c.....
        subroutine endin(iopt)
        common/nxt/llp,kount,nchars
        logical bufflg
        character*1 cbuff
        common/chbuff/nochar,lcwrit,lcread,bufflg,cbuff(500)
        common/iostr/nread,nprint,ifile,wfile,
+      ntti,nwrt,ndski,ndsko,infile,outfil
        logical ifile,wfile
c
        character*20 str
        logical kf,fin,prt,ofile,rfile
        common/freerd/nwrite,ntype,aval,ival,str,ich,kf,fin,prt
+      ,ofile,rfile
c
        character*25 infile,outfil
        goto (60,10)iopt
c
c**** end instruction
c
10      if(.not.bufflg)goto 40
120     if(ofile) write(nwrite,30)
120     if(.not.rfile)write(ntype,30)
130     format(' end on "chbuff -internal character buffer"')

```

```

        nread=ntti
        if(rfile)nread=ndski
        bufflg=.false.
        go to 100
40      if(ofile.and.prt) write(nwrite,50) infile
        write(ntype,50) infile
50      format(' end on "',a25,'"')
        nread=ntti
        rfile=.false.
        go to 100
c
c**** eof instruction
c
60      if(rfile)goto 70
        if(bufflg)goto 20
        go to 100
70      continue
        close(unit=ndski)
        if(prt)write(nwrite,80) infile
        write(ntype,80) infile
80      format(' eof on "',a25,'"')
90      rfile=.false.
        ifile=.false.
        nread=ntti
100     return
        end
c.....
c
c      ZPRINT
c
c      PURPOSE:
c          To print the parameters for program ZSIM
c
c      PARAMETERS:
c          ptable : pointer to table values
c          tables : table of functional block values
c.....
c      subroutine ZPRINT(ptable,tables,exec)
c
c          integer mgen,pgen,mscint,pscint,mquant,pquant
c          integer block,tsadd,circuit,fftyes,exec(5,2)
c          common/blkmax/mgen,pgen,mscint,pscint,mquant,pquant
c          common/parmax/mxblk,mxdim,mxdim3,mxtele,mxnode,mxicyc,mxnumx
c          common/funblk/block,period,nocyc,tsadd,pi,freqpt,circuit
c          integer ptable(mxtype,mxblk,mxdim3)
c          real tables(mxtele),maxout(5),minout(5),maxin(5),minin(5)
c          character*20 str
c          logical kf,fin,prt,ofile,rfile
c          common/freerd/nwrite,ntype,aval,ival,str,ich,kf,fin,prt,ofile,rfile
c          common/fourie/snoise,sig,totnol,totno2,totno3,snr1,snr2,snr3,fftyes
c          common/intmax/maxout,minout,maxin,minin
CGTB_START
        integer hist(5,50,50,3)
        logical phist
        common/histgm/phist,hist
CGTB_END
c
c      PRINT PROGRAM LIMITS
c
CMBS      write(nwrite,740)
CMBS      write(nwrite,737)
CMBS      write(nwrite,*) '          CIRCUIT PARAMETERS'
CMBS      write(nwrite,738)
CMBS      write(nwrite,9060)' Max. num. of functional blocks',mxblk
CMBS      write(nwrite,9060)' Max. num. of input generators',mgen
CMBS      write(nwrite,9060)' Max. num. of integrators',mscint
CMBS      write(nwrite,9060)' Max. num. of quantizers',mquant
CMBS      write(nwrite,9060)' Max. num. of nodes',mxnode
CMBS      write(nwrite,9060)' Max. num. of table dimensions',mxdim
CMBS      write(nwrite,9060)' Max. size of table vector',mxtele
CMBS      write(nwrite,9060)' Max. num. of values for an input node',mxnumx
CMBS      write(nwrite,9060)' Max. num. of cycles to store',mxicyc
c
c      PRINT INTEGRATOR RESULTS

```

```

C
    write(nwrite,737)
    write(nwrite,*)'          INTEGRATOR OUTPUT'
    write(nwrite,*)
    do 5 jjj=1,circuit
    write(nwrite,*)' Integrator #',jjj
    write(nwrite,*)'   Input max value = ',maxin(jjj),
+   '   Input min value = ',minin(jjj)
    write(nwrite,*)'   Output max value = ',maxout(jjj),
+   '   Output min value = ',minout(jjj)
    write(nwrite,*)
5      continue
    write(nwrite,738)
C 737   format(/79('')/'',77x,')
C 738   format('',77x,')/79('')/)
    737   format(/79(''))
    738   format(79('')/)
CGTB_START
    if(.not.phist)goto 1010
    write(nwrite,*)
    do 120 ii=1,circuit
    write(nwrite,*)
    write(nwrite,*)'HISTOGRAM FOR TABLE',ii
    write(nwrite,*)
    write(nwrite,*)'      x1 values:',ptable(1,ii,4)
    write(nwrite,*)'      x2 values:',ptable(1,ii,5)
    write(nwrite,*)'      x3 values:',ptable(1,ii,6)
    write(nwrite,*)
    do 121 k=1,ptable(1,ii,6)
    do 122 j=1,ptable(1,ii,5)
        write(nwrite,*)(hist(ii,i,j,k), i=1,ptable(1,ii,4))
122    continue
121    continue
    write(nwrite,738)
120    continue
CGTB_END
C 740   format(//)
c9000   format(//a40//)
c9020   format(' ',a35,f16.3)
c9040   format(' ',a40,e16.8)
    9060   format(' ',a40,i7)
1010    return
    end
C.....
C
C      ZSDR
C
C      PURPOSE:
C          This subroutine calls the appropriate routine to calculate
C          Signal-to-Distortion Calculation Routine
C
C      PARAMETERS:
C          x      : array of node values
C
C      CAPITALS INDICATE KEYWORDS AND THEIR MINIMUM LENGTH
C
C      SUBROUTINES CALLED:
C          fread, zcalsdr
C
C      EXAMPLES:
C          SDR n1 n2 TYPE=CALSDR NPOINTS=nnn1 NFFT=nnn2 NSKIP=nnn3
C          FFTWINDOW=sss1 DECWINDOW=sss2 FB=xxx1
C          TAPS=nnn4
C          This performs a Signal-to-Distortion calculation on the
C          values of node n1. The calculated spectrum is stored as
C          node n2. Options available:
C          FFTWINDOW = UNIFORM HAMMING blackman-HARRIS
C          DECWINDOW = IGNORE UNIFORM TRIANGULAR PARABOLIC
C          (DECWINDOW should be the same used in decimation.)
C          FB : baseband sampling frequency
C.....
    subroutine zsdr(x)

    common/parmax/mxblk,mxdim,mxdim3,mxtele,mxnode,mxicyc,mxtype,mxnumx

```

```

        real x(mxicyc,mxnode)
        character*20 str
        logical kf,fin,prt,ofile,rfile
        common/freerd/nwrite,ntype,aval,ival,
+           str,ich,kf,fin,prt,ofile,rfile
c
c  PROCESS REMAINDER OF INPUT LINE
c
10      call fread
        if(fin)goto 100
        if(str(1:4).eq.'HELP')then
        write(ntype,50)
          if(prt)write(nwrite,50)
50      format(
+       ' ##### SDR #####',/,
+       ' Usage:',/,/,
+       '   SDR HELP',/,/,
+       '   : This Message',/,/,
+       '   SDR n1 n2 TYPE=CALSDR NPOINTS=nnn1 NFFT=nnn2 NSKIP=nnn3',/,
+       '   : FFTWINDOW=sss1 DECWINDOW=sss2 FB=xxx1',/,/,
+       '   : TAPS=nnn4',/,/,
+       '   : This performs a Signal-to-Distortion calculation on',/,
+       '   : the values of node n1. The calculated spectrum is',/,
+       '   : stored as node n2. Options available:',/,/,
+       '   : FFTWINDOW = UNIFORM HAMMING blackman-HARRIS',/,
+       '   : DECWINDOW = IGNORE UNIFORM TRIANGULAR PARABOLIC',/,
+       '   : (DECWINDOW should be the same used in decimation.)',/,
+       '   : TAPS is the number of taps used in the decimation',/,
+       '   : FB : baseband sampling frequency ',/,/,
+       ' #####')
        return
      endif
c
c  PROCESS INPUT ACCORDING TO FIRST FIELD OF INPUT LINE
c
c  GET NODE NUMBERS
c
        if(.not.kf)then
          nodein=ival
        else
          write(ntype,*) '? node number required ?'
          if(ofile) write(nwrite,*)
+         '? node number required ?'
          goto 100
        endif
        call fread
        if(fin)goto 100
        if(.not.kf)then
          nodout=ival
        else
          write(ntype,*) '? node number required ?'
          if(ofile) write(nwrite,*)
+         '? node number required ?'
          goto 100
        endif
c
c  DETERMINE WHICH TYPE OF SDR CALCULATION TO USE
c
        call fread
        if(fin.or..not.kf)goto 100
        if(str(1:4).eq.'TYPE')then
c
c  FIND OUT WHICH TYPE OF SDR IS TO BE USED AND CALL APPROPRIATE
c  ROUTINE
c
        call fread
        if(fin.or..not.kf)goto 100
        if(str(1:6).eq.'CALSDR')then
          call zcalsdr(x,nodein,nodout)
        else
          write(ntype,*)' Type unknown'
          if(ofile) write(nwrite,*)' Type unknown'
          goto 100
        endif
      else

```

```

        goto 100
    endif
    return
C
C   ERROR STATEMENTS
C
100   write(ntype,*)' ? SDR Error'
        if(ofile) write(nwrite,*)' ? SDR Error'
        return
    end
C.....
C
C   ZSET
C
C   PURPOSE:
C       Set up all the parameters for program ZSIM.
C
C   PARAMETERS:
C       flag      : indicates action required
C                  [ 1/initialize I/O; 2/edit parameters ]
C       x         : storage array for circuit node values
C       ptable    : pointer to table values
C       tables    : table of functional block values
C       genflg    : acknowledges presence of generators
C       intflg    : acknowledges presence of integrators
C       qntflg    : acknowledges presence of quantizers
C       ptnode    : should node value be printed
C       ndpres    : acknowledges presence of node
C       node      : store present value of nodes
C       nodep     : store previous value of nodes
C.....
C       subroutine ZSET(flag,x,ptable,tables,genflg,intflg,qntflg,
+         ptnode,ndpres,node,nodep)
C
C       integer flag,block,tsadd,mgen,pgen,mscint,pscint,mquant,pquant
C       integer freqpt,circuit,fftyes
C       common/blkmax/mgen,pgen,mscint,pscint,mquant,pquant
C       common/parmax/mxblk,mxdim,mxdim3,mxtele,mxnode,mxicyc,mxtype,mxnumx
C       common/file/fileno
C       integer fileno
C       common/funblk/block,period,nocyc,tsadd,pi,freqpt,circuit
CGTB_START
C       integer hist(5,50,50,3)
C       logical phist
C       common/histgm/phist,hist
CGTB_END
C       real mindel,maxdel,offset
C       real a1,a2,a3,fgbw1,fgbw2,fgbw3
C       real delta1,delta2,delta3,ofint1,ofint2,ofint3
C       real satmn1,satmx1,satmn2,satmx2,satmn3,satmx3
C       real aa1,aa2,aa3,b1,b2,b3,c1,c2,c3
C       common/diffeq/a1,a2,a3,fgbw1,fgbw2,fgbw3,
+         delta1,delta2,delta3,
+         ofint1,ofint2,ofint3,
+         satmn1,satmx1,satmn2,satmx2,satmn3,satmx3,
+         maxdel,mindel,offset,
+         aa1,aa2,aa3,b1,b2,b3,c1,c2,c3
C       integer ptable(mxtype,mxblk,mxdim3)
C       real noise(10,3),pi
C       real tables(mxtele),x(mxicyc,mxnode),node(mxnode),nodep(mxnode)
C       logical genflg(mgen),intflg(mscint),qntflg(mquant)
C       logical ptnode(mxnode),ndpres(mxnode)
C       logical cyppar,cpynod,over,clear
C       common/commnd/cyppar,cpynod,over,clear
C       character*20 str
C       logical kf,fin,prt,ofile,rfile
C       common/freerd/nwrite,ntype,aval,ival,str,ich,kf,fin,prt,ofile,rfile
C       common/fourie/snoise,sig,totno1,totno2,totno3,snr1,snr2,snr3,fftyes
C
C   IN GOTO STATEMENT(FOR FLAG=2) CHANGE TO LABEL 500 WHEN "dsmped" IS
C   WORKING AND DELETE THE COMMENT ON LINE 500.
C
        goto(10,1000)flag
C

```

```

c  INITIALIZE I/O AND CONSTANTS
c
10  call input(' ',nerr,2)
    write(ntype,*)' ZSIM  version ZSIM0a2'
    write(ntype,*)' Nonlinear Z Domain Simulator'
    write(ntype,*)' Brauns, Steer, Ardalan, ECE Dept'
    write(ntype,*)' North Carolina State University'
    write(ntype,*)' '
    fileno=11
c
c  SET DEFAULT VALUES OF ARRAYS
c
    do 30 j=1,mgen
        genflg(j)=.false.
30   continue

    do 40 j=1,mscint
        intflg(j)=.false.
40   continue

    do 50 i=1,mxtele
        tables(i)=0.
50   continue

    do 60 i=1,mxtype
        do 62 j=1,mxblk
            do 64 k=1,mxdim+3
                ptable(i,j,k)=0
64         continue
62         continue
60         continue

    do 70 i=1,mquant
        qntflg(i)=.false.
70   continue

    do 74 i=1,mxnode
        node(i)=0.
        nodep(i)=0.
        noise(i,1)=0.
        do 76 j=1,mxicyc
            x(j,i)=0.0
76         continue
74         continue

    do 80 i=1,mxnode
        ptnode(i)=.false.
        ndpres(i)=.false.
80   continue

CGTB_START
    do 120 ii=1,mxblk
        do 121 i=1,mxnumx
            do 122 j=1,mxnumx
                do 123 k=1,3
                    hist(ii,i,j,k)=0
123                continue
122            continue
121        continue
120    continue

    phist=.false.
CGTB_END
c
c  SET CONSTANTS
c
    pi=3.1415926536
c
c  SET DEFAULT VALUES
c
    freqpt=10
    block=0
    period=1.0e6
    nocyc=65536
    tsadd=1

```

```

        circuit=1
        cpypar=.false.
        cpynod=.false.
        clear=.false.
        over=.false.
c for snr calculations
        fftyes=0
        snoise=0.
        sig=0.
        totno1=0.
        totno2=0.
        totno3=0.
        snr1=0.
        snr2=0.
        snr3=0.
c for difference equation simulations
        mindel=-1.
        maxdel=1.
        offset=0.
        a1=1.
        a2=1.
        a3=1.
        delta1=1.
        delta2=1.
        delta3=1.
        fgbw1=0.0
        fgbw2=0.0
        fgbw3=0.0
        ofint1=0.0
        ofint2=0.0
        ofint3=0.0
        satmn1=-5.
        satmx1=5.
        satmn2=-5.
        satmx2=5.
        satmn3=-5.
        satmx3=5.
        goto 1000
630         format(/)
640         format(80a1)
650         format(' %what? try again.')
```

```

660         format(1x,80a1)
670         format(i10,a1)
680         format(' "ZSET",line =',i7)
1000        return
        end
c .....
c
c        ZSIM
c
c        Main Routine
c
c .....
        real gen(5,6),scint(3,5),quant(3,6),tables(10000),x(66000,6)
        integer exec(5,2),nodeno(5),nodety(5)
        common/file/fileno
        integer fileno
        real period,pi,node(5),nodep(5),noise(10,3)
        integer circuit,ptable(3,5,9)
        integer block,nocyc,tsadd,mgen,pgen,mscint,pscint,mquant,pquant
        logical cpypar,cpynod,clear,over,genflg(2),intflg(3),qntflg(3)
        logical ptnode(5),ndpres(5)
        common/blkmax/mgen,pgen,mscint,pscint,mquant,pquant
        common/parmax/mxblk,mxdim,mxdim3,mxtele,mxnode,mxicyc,mxtype,mxnumx
        common/funblk/block,period,nocyc,tsadd,pi,freqpt,circuit
        common/commnd/cpypar,cpynod,over,clear
        common/nose/noise
        character*20 str
        logical kf,fin,prt,ofile,rfile
        common/freerd/nwrite,ntype,aval,ival,str,ich,kf,fin,prt,ofile,rfile
c
c        SET SIZE OF ARRAYS
c
        mxblk=5
        mgen=5

```

```

    pgen=6
    mscint=3
    pscint=5
    mquant=3
    pquant=6
    mxdim=6
    mxtele=10000
    mxicyc=66000
    mxnode=6
    mxtype=3
    mxnumx=50
    mxdim3=mxdim+3
c
c  INITIALIZE I/O AND DEFAULT VALUES
c
  50      call ZSET(1,x,ptable,tables,genflg,intflg,qntflg,ptnode,ndpres,
    +      node,nodep)
c
c  HANDLE COMMAND SEQUENCE
c
  100     call ZCMD(x,gen,scint,quant,ptable,tables,genflg,intflg,
    +      qntflg,ptnode,ndpres,node,nodep,exec,nodeno,nodety)
    if(clear)goto 50
    if(over) goto 1999
c
c  PRINT ALL CIRCUIT PARAMETERS
c
    if (cpypar)call ZPRINT(ptable,tables,exec)
    if (cpynod)call NODEPT(x,ptnode,ndpres)
    goto 100
1999     end

```