TECHNICAL DETAILS OF FOCAS
(Fiber Optic Cable Simulator)

DEC - TR 471

Ann B. Ewalt

March 11, 1987

DEC-TR
471

**** RESTRICTED DISTRIBUTION ****

TECHNICAL DETAILS OF FOCAS
(Fiber Optic Cable Simulator)

DEC - TR 471

Ann B. Ewalt

March 11, 1987

Keywords: FOCAS, fiber optics, optical cable, optical fiber, fiber optic communications

ABSTRACT

This document is intended to provide the technical information necessary to support and/or modify the simulation tool FOCAS. FOCAS is used to simulate the physical behavior of fiber optic communications links. This document assumes that the reader is familiar with the "FOCAS User's Guide", DEC TR-470.

Distibuted Systems Technical Office
Digital Equipment Corporation
110 Spitbrook Road
Nashua, New Hampshire 03602

CONTENTS

I. INTRODUCTION 1

II. BASIC NETWORK ANALYSIS ALGORITHMS . . . 2

III. OPTICAL FIBER MODEL & ENHANCEMENTS . . . 5

IV. NETWORK SUBSECTION MODELS

A. GEN - trapezoidal generator & overlapping l's correction 10

B. SINE - sine wave generator 13

C. TEE - TEE network with series L, R and shunt C 14

D. FIBER - fiber optic cable driven by source with Gaussian spectral distribution . . . 15

E. HPF - high pass filter 17

F. EAMP - voltage dependent voltage generator . . 18

G. IAMP - voltage dependent current generator . . 19

H. LINE - rectangular stripline 20

I. FILTER - response F(w) = {cos wTo}**N 22

J. SHUNT - shunt impedance 23

K. SERIES - series impedance 24

L. DELTA - standard PI circuit with series L and shunt C 25

M. BPF - bandpass filter 26

N. SRC - interface to user defined source data file 28

V. FOCAS PROGRAMS

A. Data Flow 31
B. Calling Structures 36
C. Parameter Map for Array RLC 44
D. Listings 47

VI. ADDING A NEW NETWORK TYPE 110

VII. ENHANCEMENTS

A. Complex Filters & Resonant Circuits . . . 112
B. Optical Cable Splices . . . 113
C. Reduction of Timing Uncertainty . . . 113
D. Fiber Models . . . 114
E. Graphics Support on MicroVax . . . 114

VIII. APPENDICES . . . 115

Appendix I. LED/CABLE Reference Data for Figures 1 & 2 . 117
Appendix II. Fiber Response Function with Constant D . 119
Appendix III. Fiber Response Function with D = SO*(L-Lc) . 120
Appendix IV. Integrals & Mathematical Identities . . . 123
Appendix V. CIRCUIT_GRAPH.FOR Listing . . . 125

IX. ACKNOWLEDGEMENTS 129

X. BIBLIOGRAPHY 130

XI. FIGURES 131

Figure 1. Cable 1 Rise & Fall Time Data

Figure 2. Cable 2 Rise & Fall Time Data

Figure 3. Source Gaussian Spectrum

Figure 4. Overshoot with Leading & Trailing 1's

Figure 5. Corrected Waveform with Leading & Trailing 1's

Figure 6. Overlapping 1's Correction: PER+Do < D1+P1

Figure 7. Overlapping 1's Correction
(D1+P1<PER+Do<D1+P1+F) .and. (D1+P1+F>PER+Do+R)

INTRODUCTION

FOCAS is a SIMULATION tool for analyzing the physical behavior of point-to-point fiber optic communications links. All circuit analysis is performed in the frequency domain, then transformed into the time domain. Hence, all input waveforms are assumed to be periodic in time.

FOCAS contains a collection of network subsections (circuits) from which the simulation models are constructed. These models represent the behavior of a circuit or signal generator in the frequency domain. Three voltage sources are included: GEN, an asymmetric periodic trapeziodal wave source; SINE, a periodic sine wave source; and SRC, a digitized waveform with voltage specified at equally spaced time intervals. These source models all return the component of the source's frequency spectrum at the desired angular frequency. All of the other circuit type models (TEE, FIBER, HPF, EAMP, IAMP, LINE, FILTER, SHUNT, SERIES, DELTA, BPF) return the general circuit parameters (A, B, C, D) for the particular circuit type at the specified angular frequency. A simulation circuit is composed of a series of these cascaded two-terminal-pair networks.

Once FOCAS is installed on a system, the user creates a simulation model by simply creating the model in a the file, e.g. "MODELXYZ.". At the system prompt, RUN FOCAS invokes the simulator. FOCAS prompts to the user for the simulation model name as the only input required.

Output voltages are based on terminal pair differentials. FOCAS provides output in both graphical and tabular form. The graphics files may be displayed directly on REGIS terminals. Output data files provide an interface to graphing routines for various other display devices.

FOCAS creates up to seven output files as specified by control statements:

MODELXYZ.TG = time domain graph in REGIS (always created)
MODELXYZ.TD = time domain data file (always created)

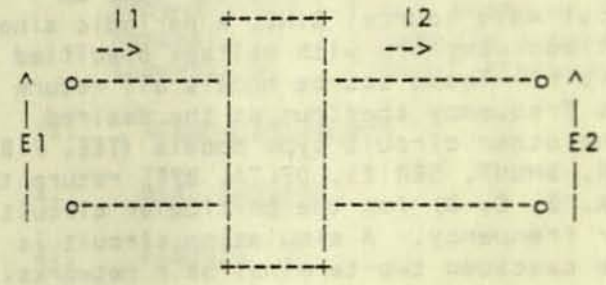
MODELXYZ.FG = frequency domain graph in REGIS (created when control line .FDG specified)
MODELXYZ.FD = frequency domain data file (created when control line .FDG specified)

MODELXYZ.PD = phase vs. frequency data file (created when control line .PDG specified)
MODELXYZ.PG = phase vs. frequency graph in REGIS (created when control line .PDG specified)

MODELXYZ.DAT = times at which output voltage waveform crosses specified voltage levels (created when there is crossover or extremum data

II. BASIC NETWORK ANALYSIS ALGORITHM

FOCAS performs all network calculations in the frequency domain, then translates the results back into the time domain by use of the fast Fourier transform (subroutine FFT842). Complex networks are broken down into simple sections for which the chain matrices can be calculated (Ref. Weinberg, pp.23-26). The parameters: A, B, C, and D in the chain matrix completely describe the response of the network subsection as shown below.



The chain matrix is defined by:

$$(1) \qquad \begin{vmatrix} A & B \\ C & D \end{vmatrix} \cdot \begin{vmatrix} E2 \\ I2 \end{vmatrix} = \begin{vmatrix} E1 \\ I1 \end{vmatrix}$$

where the general circuit parameters are given by:

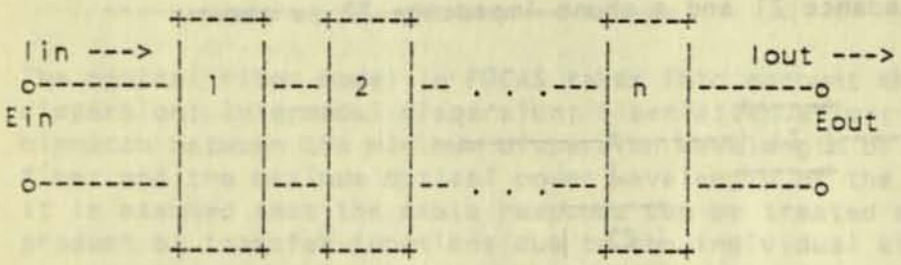
(2) $A = E1/E2$ when $I2 = 0$ (open output)

(3) $B = E1/I2$ when $E2 = 0$ (output shorted)

(4) $C = I1/E2$ when $I2 = 0$ (open output)

(5) $D = I1/I2$ when $E2 = 0$ (output shorted).

Chain matrices have the desirable property that the chain matrices of cascaded two-terminal-pair subsections can be multiplied together to determine the behavior of the composite network. The illustration on the top of the following page shows how this property is applied.



$$(6) \qquad \begin{vmatrix} A_{tot} & B_{tot} \\ C_{tot} & D_{tot} \end{vmatrix} \cdot \begin{vmatrix} E_{out} \\ I_{out} \end{vmatrix} = \begin{vmatrix} E_{in} \\ I_{in} \end{vmatrix}$$

where:

$$(7) \qquad \begin{vmatrix} A_{tot} & B_{tot} \\ C_{tot} & D_{tot} \end{vmatrix} = \begin{vmatrix} A_1 & B_1 \\ C_1 & D_1 \end{vmatrix} \begin{vmatrix} A_2 & B_2 \\ C_2 & D_2 \end{vmatrix} \cdots \begin{vmatrix} A_n & B_n \\ C_n & D_n \end{vmatrix}$$

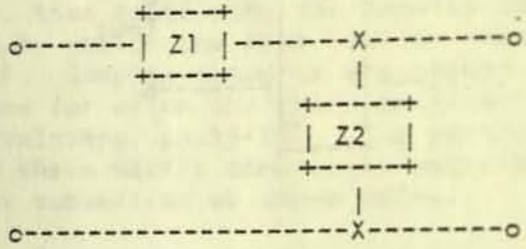
A more useful form of these equations relates the output voltage and current to the input voltage and current:

$$(8) \qquad \begin{vmatrix} D_{tot} & B_{tot} \\ C_{tot} & A_{tot} \end{vmatrix} \cdot \begin{vmatrix} E_{in} \\ -I_{in} \end{vmatrix} = \begin{vmatrix} E_{out} \\ -I_{out} \end{vmatrix} .$$

It should be noted that the term "chain matrix" refers to the form given in (1):

$$\begin{vmatrix} A & B \\ C & D \end{vmatrix} .$$

As an illustration, consider a network comprised of a series impedance Z1 and a shunt impedance Z2 as shown below:



This configuration is described by the product of two two chain matrices: M1 * M2 where M1 represents the series impedance and M2 the shunt impedance.

$$(9) \quad M1 = \begin{vert} 1 & Z1 \\ 0 & 1 \end{vert} \quad \text{and} \quad M2 = \begin{vert} 1 & 0 \\ 1/Z2 & 1 \end{vert}$$

If Z1 represents a Capacitor, C, and Z2 a resistor, R, the network is a simple high pass filter. The individual chain matrices are:

$$M1 = \begin{vert} 1 & 1/sC \\ 0 & 1 \end{vert} = \begin{vert} 1 & -j/wc \\ 0 & 1 \end{vert} \; ; \qquad s = jw$$

and

$$M2 = \begin{vert} 1 & 0 \\ 1/R & 1 \end{vert} .$$

The chain matrix for the network, Mtot, is:

$$Mtot = \begin{vert} (1+Z1/Z2) & Z1 \\ 1/Z2 & 1 \end{vert} = \begin{vert} (1 + 1/sRC) & 1/sC \\ 1/R & 1 \end{vert} .$$

This is precisely the result shown for the HPF (Section IV) with the inductance, L, equal to zero.

III. OPTICAL FIBER MODELS & ENHANCEMENTS

The optical fiber model in FOCAS takes into account chromatic dispersion; intermodal dispersion; fiber attenuation; and the mismatch between the minimum dispersion wavelength of the fiber and the maximum optical power wavelength of the source. It is assumed that the cable response can be treated as the product of transfer functions due to the individual effects:

(10) $G_{tot}(w,L) = P_{LED}(w,L) * Gc(w,L) * G_{IM}(w) * Ga(w)$

(11) Gtot(w,L) = cable response at angular frequency w and wavelength L

(12) $P_{LED}(w,L)$ = optical source output power at angular frequency w and wavelength L

(13) Gc(w,L) = fiber chromatic dispersion transfer function at angular frequency w and wavelength L

(14) $G_{IM}(w)$ = fiber intermodal dispersion transfer function at angular frequency w

(15) Ga(w) = fiber attenuation transfer function at angular frequency w

The fiber optic cable is modelled by a single network subsection with characteristics of both the optical fiber and the optical source. Mathematical details are given in Appendices II & III. Each of the individual physical effects is discussed separately below.

A. SOURCE SPECTRUM

The normalized optical transmitter power spectrum is approximated by a Gaussian wavelength distribution.

(16) $P_{LED}(w,L) = \frac{1}{Z*SQRT(PI)} \exp\{ - (L-Lc)**2/Z**2\}$ where

(17) Z = F/[2*sqrt(ln 2)] = standard deviation of the source spectrum

F = width of the spectral source at half power in nanometers

B. CHROMATIC DISPERSION

The FIBER model approximates the dispersion coefficient of the fiber with a constant based on the Sellmeier equation. The dispersion coefficient, D in ns/(nm-km), is given by:

$$(18)\quad D = 1.176 * \sqrt{Do^{**2} + \frac{(0.85*So*FWHM)^{**2}}{8}}$$

where one has:

Lo = minimum dispersion wavelength of the fiber

Lc = peak power wavelength of the source

So = slope of the dispersion curve in ns/(nm**2-km)

FWHM = source half max. spectral width

$$(19)\quad Do = \frac{So}{4} * \left| \frac{Lc - Lo^{**4}}{\cdot\, Lc^{**3}} \right|$$

The expression in equation (19) is the classical representation of the Sellmeir equation. Equation (18) was suggested by Siecor's Dieter Schickentanz and his treatment of chromatic dispersion, then modified by empirical data. The value of D given by equation (18) can be thought of as an average dispersion value weighted by the source's spectral distribution.

An experiment was run in which seven different LED's were used to excite two different optical fiber cable plants, and the 10-90% rise and fall times of the output waveforms compared with predictions. Appendix I contains the specifications of the cables and LED's used in this experiment. Figures 1 and 2 show the results.

The SIECOR points in the Figures correspond to the values predicted using Schickentanz's dispersion and bandwidth summing equations. These relationships assume that all fiber frequency responses follow Gaussian distributions and can be summed accordingly:

$$(20)\quad \frac{1}{BWtot^{**2}} = \frac{1}{BW1^{**2}} + \ldots + \frac{1}{BWn^{**2}}$$

SIMER points use the same cable model as FOCAS, with the exception that the value of D is not multiplied by the factor of 1.176 as in equation (18).

It is evident from the Figures that the FOCAS model provided the best prediction of the output rise and fall times of the models available. However, the disprecancies with the measured values are of some concern. Subsequent review of the data showed some lack of repeatability in the measurements due to changes in the fiber pigtails through which measurements were made. Plans have been made to repeat these tests and perform others.

In addition, the simulation runs all assumed simple rounded trapezoidal edge shapes without ringing for the LED's, while observation showed that, particularly for those LED's with large differences between predicted and measured results, the transisitons actually had a good deal of ringing. There are presently experiments being run to determine the importance of exactly modelling source turnon and turnoff characteristics.

It is anticipated that the chromatic dispersion model in FOCAS will undergo further refinement as more data become available. Appendix III contains the derivation of an alternate optical fiber model which includes first order dependence of dispersion on wavelength:

(21) $D(L) = D(L0) + D'(L)*(L-L0) + D''(L)*[(L-L0)**2]/2 + \ldots$

which reduces to:

(22) $D(L) = S0*(L-L0)$

because dispersion varies linearly with wavelength near L0, and L0 is defined as the wavelength at which D(L0) is zero and D'(L0) is equal to S0 (Ref. 2).

Modifications which simply require that a different expression be used to compute D can be made directly in SUBROUTINE PRECALC. The dispersion calculation is labelled in the calculations for the Type 7 Circuit. More complex models, showing the explicit wavelength dependence of D can be added as new network types. Appendix IV contains basic integrals and identities to assist in future modifications to the fiber model(s).

The effect of chromatic dispersion is to delay wavelength component L by an amount, T, proportional to fiber length, X. This delay results in a phase shift as shown in equation (24).

(23) $T = D*(L-Lc)*(X**Qc)$ = delay introduced to spectral component at wavelength L with respect to Lc

Qc = chromatic length scale factor

(24) $Gc(w) = \exp\{ jw*T\}$ = phase shift introduced to spectral component at wavelength L with respect to a component at the central wavelength, Lc

$= \exp\{ jw*D*(L-Lc)*(X**Qc)\}$

C. FIBER ATTENUATION

The fiber attenuation is assumed to be exponential with effective fiber length, X**Qa

(25) Ga(w) = exp{-K*A} = fiber transfer function due to attenuation

(26) A = [ro + rw*(L-Lc)]*(X**Qa) = fiber attenuation in dB

(27) K = (ln 10)/10 = 0.2306 = dB conversion factor

ro = DC optical power attenuation of the fiber in dB/km

rw = wavelength dependent optical power attenuation of the fiber in dB/(nm-km)

Qa = attenuation length scale factor, assumed equal to Qc (chromatic length scale factor) in present FIBER model, but not in alternate model, Appendix III

D. INTERMODAL DISPERSION

The intermodal dispersion of the fiber is described by a Gaussian distribution of effective optical path lengths, (Xeff = X**Qm).

(28) $G_{IM}(w) = \exp\{-(d^{**}2)\}$ where

(29) d = (0.1325 * w * Xeff)/FI

(30) Bm = FI/Xeff = modal bandwidth of the fiber in Gigahertz

FI = cable bandwidth due to intermodal dispersion in GHz-km

Xeff = X**Qm = effective modal cable length

X = fiber physical length in kilometers

Qm = intermodal length scale factor

E. FIBER RESPONSE

Combining all of the physical effects, one obtains the fiber response function, Pe(w):

$$(31) \qquad Pe(w) = \int_{-\infty}^{\infty} Gtot(w,L)\, dL \qquad \text{with } Gtot(w,L) \text{ defined in (10).}$$

Detailed evaluation of integral (31) is given in Appendix II for the case in which equation (18) is used for dispersion; Appendix III for the case in which equation (22) is used for dispersion.

The general circuit parameters for a two terminal pair fiber model are then given by:

$$(32) \qquad A = D = 1 / Pe(w)$$

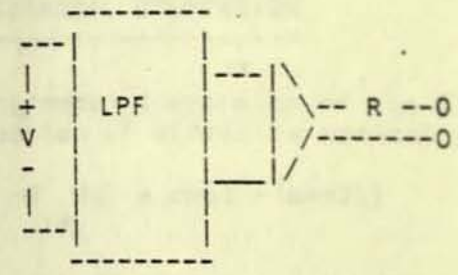
$$(33) \qquad B = C = 0.$$

NETWORK SUBSECTION MODELS

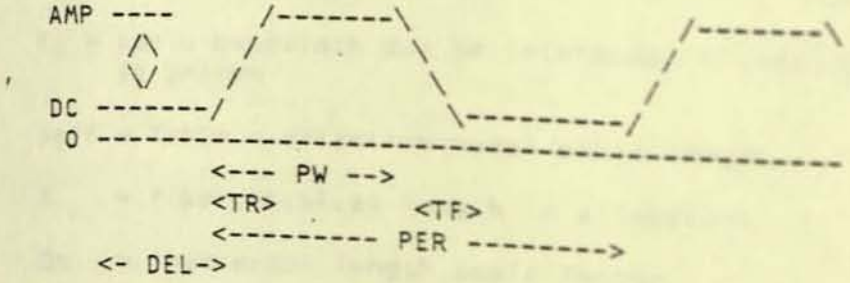
The network subsection types are as follows:

GEN	- A trapezoidal waveform voltage generator
SINE	- A sinewave generator
EAMP	- Voltage dependent voltage generator
IAMP	- Voltage dependent current generator
TEE	- The standard T circuit with series L and shunt C
DELTA	- The standard PI circuit with series L and shunt C
LINE	- A transmission line, skin effect for stripline
FIBER	- Fiber optic cable dispersion model
HPF	- High Pass Filter
XFORMER	- Transformer
FILTER	- Cosine frequency domain filter
SHUNT	- Polynomial expression for a shunt impedance
SERIES	- Polynomial expression for a series impedance
BPF	- Bandpass filter
SRC	- User defined input waveform

A. GEN Network



The GEN is a periodic trapezoidal waveform generator followed by low pass filter and a unity voltage gain buffer with an output impedance equal to R.



Parameters:

PW = "ideal" pulse width, i.e. the time between the decisions to turn on and turn off

TR = 10-90 % risetime (leading edge) in nanoseconds.

TF = 10-90 % falltime (trailing edge) in nanoseconds. (default=TR)

PER = pulse repetition period in nanoseconds

AMP = pulse amplitude in volts, defaults to one.

DEL = pulse delay in nanoseconds, measured from the leading edge. Defaults to zero.

BW = filter bandwidth in gigahertz, defaults to 50 divided by the risetime.

DC = DC voltage offset in volts, defaults to zero

R = generator output resistance in ohms, defaults to zero.

If several GEN sections are cascaded together to form a data stream, FOCAS automatically sets some conditions to simplify user input. FOCAS compares the risetimes of the first two GEN sections. If they are not equal, and the second is nonzero (i.e. not to be set equal to the first), a dual slope leading edge is assumed. If they are equal, or if the second risetime is not specified (i.e. initially zero), a single sloped leading edge is assumed. For the single slope case, TR, TF, and AMP are all set to the values specified for the first GEN. For the dual slope case, all odd numbered GEN's are assigned the TR, TF, and AMP values of the first GEN; all even numbered GEN's receive the values specified by the user for the second GEN.

In all cases, only the first GEN need specify PER and BW. Whenever any of the input parameters is missing from a subsequent GEN specification, FOCAS automatically determines whether to use the single or dual slope model.

Similarly, FOCAS automatically checks if the first and last bits in a periodic data stream are one's. If there are potentially overlapping one's, i.e., the optical transmitter does not have time to completely turn off before receiving another signal to turn on, FOCAS will combine the first and last GEN 's while preserving the data integrity, without driving the transmitter output beyond its maximum level.

Without this "overlapping 1's" checking, nonphysical overshoot behavior as shown in Figure 4 would occur. Subroutine PRECALC performs the overlapping 1's checking and correction to produce the waveform shown in Figure 5 by comparing the last GEN's output pulse with the second repetition of the pulse generated by the first GEN in the string.

The correction algorithm works by checking for the conditions which result in overshoot:

(34) $PER + Do < D1 + P1$ (See Figure 6)

or

(35) $(D1+P1 < PER+Do < D1+P1+F)$ .and. $(D1+P1+F > PER+Do+R)$ (See Figure 7)

where: D1 = pulse delay of the last (NGEN-th) GEN of the input simulation model

P1 = ideal pulse width of the last GEN of th input model

Do = pulse delay of the first GEN of the input model

Po = ideal pulse width of the first GEN of the input model

PER = period of the data stream

R = 0-100% rise time of the source (same for all GEN's)

F = 0-100% fall time of the source (same for all GEN's)

When either of these conditions is encountered, subroutine PRECALC modifies the input simulation model by replacing the last GEN with a GEN with corrected pulse width, PL, and then dropping the first GEN. The corrected pulse width is calculated to combine the input bits represented by the original model's first and last GEN's.

(36) $PL = Po + CORR$

(37) $CORR = |(PER+Do-D1)|$

Invoking the overlapping 1's correction results in a model with (NGEN-1) generators, where NGEN is the number of GEN's in the original simulation model. Figure 6 illustrates the original and corrected GEN pulses corresponding to condition (34). Figure 7 shows the original and corredted pulses corresponding to condition (35).

Checking only occurs between the first and last GEN sections. It is assumed that the user has exercised care in defining all internal GEN sections to eliminate the potential for signals that overshoot the maximum HIGH level.

B. SINE Network

SINE is sinewave generator which in the frequency domain is a delta function at the frequency fo. It has an output impedance R. In the time domain:

$$F(t) = AMP * SIN (2 PI F t + PHI) + DC$$

Parameters:

AMP = Peak sinewave amplitude, defaults to one.

F = sinewave frequency in GHz, must be an exact multiple of the simulation reciprocal period.

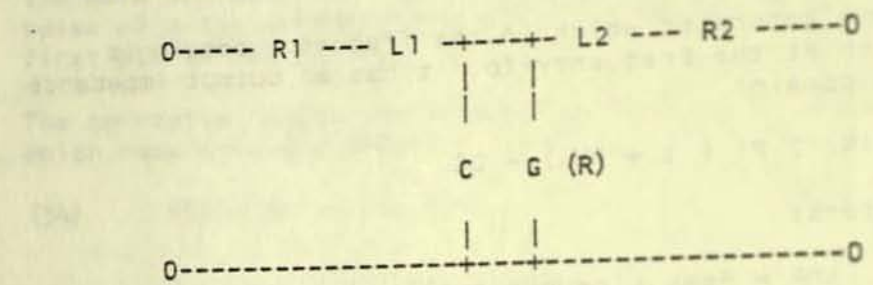
DEL = delay in nanoseconds, defaults to zero. (PHI = -W * DEL)

DC = dc voltage offset in volts, defaults to zero.

R = generator output resistance in ohms, defaults to zero.

There are no general circuit parameters for SINE. As a source generator, it is represented by its frequency sprectrum.

C. TEE Network



Parameters:

R1, R2 = series resistances in ohms, default to zero. Will have a skin effect component if F1 or F2 are specified, respectively.

L1, L2 = series inductances in nanohenries, default to zero. Will have an internal inductance component if F1 or F2 are specified respectively.

C = shunt capacitance in nanofarads, defaults to zero.

G (R) = shunt conductance in mhos, defaults to zero. Either G or R (in ohms) may be specified. If both are specified by mistake, the R value supercedes the G value.

F1, F2 = The skin effect break frequencies in Gigahertz: Defined as the frequency at which the skin effect resistance is equal to the dc resistance. Skin effect resistance and internal inductance are assumed to be square-root frequency dependent. Default to infinity (no skin effect).

The skin effect model is:

$$Rac = \text{conductor resistance} = R1 \cdot SQRT\left(1. + (F/F1)^{2} \right)^{1/4}$$

for conductor 1 (R1,L1) where F is the frequency.

Xint = internal conductor inductive reactance = Rac - R1

for conductor 1. Appropriate substitutions are made for conductor 2 (R2,L2).

D. FIBER Network

FIBER models the chromatic and intermodal dispersion, and attenuation losses in an optical cable. The model assumes intermodel dispersion based on a gaussian distribution of path lengths and chromatic dispersion based on a source with a gaussian spectral density and a linear material dispersion coeficient for the cable. The fiber cable delay is not included. The frequency response, P(w), of the cable is given by:

$$P(w) = e^{-a + b - c - d - j\,2\,b\,c}$$

$$a = K\,Ao\,X^{Qc} \quad = \text{Static attenuation term}$$

$$b = K\,K'\,Aw\,FWHM\,X^{Qc} \quad = \text{Wavelength dependent attenuation}$$

$$c = w\,K'\,FWHM\,D\,X^{Qc} \quad = \text{Wavelength dependent time delay}$$

$$d = 0.1325\,w\,X^{Qm}/FI \quad = \text{Intermodal dispersion}$$

$$K = \ln(10)/10$$

$$K' = 1./(4\ SQRT(\ln 2))$$

Parameters:

Ao = DC optical power attenuation of cable in db/km
Defaults to zero. (Same as r_o in models, section III)

Aw = wavelength dependent optical power attenuation in db/(nanometer-kilometer). Defaults to zero. Note that Aw is positive when attenuation increases with wavelength. (Same as r_ω in models, Section III)

FWHM = optical source half power full bandwidth in nanometers. Defaults to zero.

LC = optical source center wavelength in nanometers; used in calculating D when LC specified with LO and SO

LO = cable minimum dispersion wavelength in nm.; used in calculating D when LC, LO and SO specified

SO = slope of cable dispersion curve at LO in nanoseconds per (nanometer**2)-kilometer; used in calculating D when SO, LC, and LO specified

FIBER NETWORK (cont.)

Fl = cable bandwidth in GHz-Km due to intermodal dispersion. Defaults to infinity.

X = cable length in Km. Defaults to zero.

Qm = Intermodal length scale factor (default=1)

Qc = Chromatic length scale factor (default=1)

D = fiber material dispersion coefficient in nanoseconds per nanometer-kilometer; If D is zero or not specified, it is calculated from input parameters LC, LO, SO using the equations below:

$$D = 1.176 * \left\{ \sqrt{ Do^{**2} + \frac{(0.85 \times SO \times FWHM)^{**2}}{8} } \right\}$$

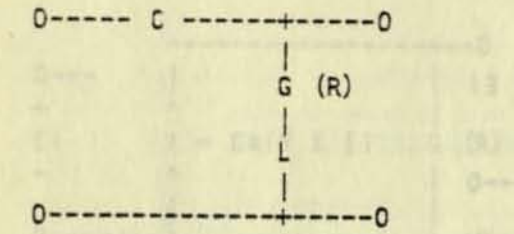
$$Do = \frac{SO}{4} \left(LC - \frac{LO^{**4}}{LC^{**3}} \right) \quad \text{Sellmeir's Equation}$$

The chain matrix parameters for the FIBER model are given by:

A = 1/P(w) B = 0 C = 0 D = 1/P(w)

E. HPF Network

A high pass filter or ac coupler.



Parameters:

C = capacitance in nanofarad, must be present.

G = shunt conductance in mhos, either G or R must be specified.

R = shunt resistance in ohms, if present it overrides an input G value and G = 1/R.

L = shunt inductance in nanohenries, defaults to zero.

The general circuit parameters for the HPF are given by:

$$A = Ar + j * Ai$$

$$Ar = 1 - \frac{L}{C\,[R^{**}2 + (wL)^{**}2]}$$

$$Ai = \frac{-R}{wC\,[R^{**}2 + (wL)^{**}2]}$$

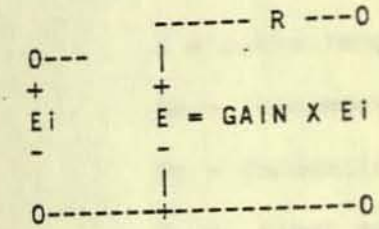
$$B = -j/(wC)$$

$$C = 1/(R + wL)$$

$$D = 1$$

F. EAMP Network

EAMP is a voltage dependent voltage generator with a specified bandwidth and output resistance.



Parameters:

GAIN = voltage gain, defaults to one

BW = filter bandwidth in gigahertz, defaults to 50 divided by the minimum risetime.

R = generator output resistance in ohms, defaults to zero.

The filter does not change the infinite input impedance of IAMP and has the low pass filter frequency domain response:

$$F(w) = \frac{1}{1 + j\ w/(2\ PI\ BW)}$$

The general circuit parameters are given by the following relations:

$$W0 = PI2*BW \qquad (PI2 = 6.28\ .\ .\ .)$$

$$d = \frac{1}{GAIN} * [\ 1 + j * w/W0\ ]$$

$$A = d$$

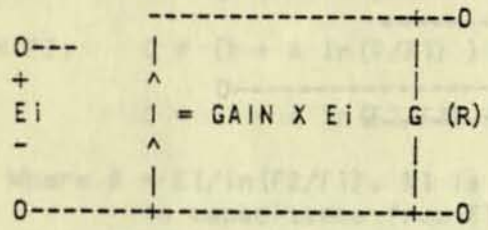
$$B = R*d = \frac{R}{GAIN} * [\ 1 + j * w/W0\ ]$$

$$C = (d/R)*10.D\text{-}9 = \frac{10**(-9)}{R*GAIN} * [\ 1 + j * w/W0\ ]$$

$$D = d*10**9 = \frac{10**(-9)}{GAIN} * [\ 1 + j * w/W0\ ]$$

G. IAMP Network

IAMP is a voltage dependent current generator with a specified bandwidth and output conductance.



Parameters:

GAIN = current gain in mhos, defaults to one

BW = filter bandwidth in gigahertz, defaults to 50 divided by the minimum risetime.

G = generator output conductance in mhos, defaults to zero.

R = optional output resistance in ohms, if present G = 1/R; overriding an input G value.

The filter does not change the infinite input impedance of IAMP and has the low pass filter frequency domain response:

$$F(w) = \frac{1}{1 + j\ w/(2\ PI\ BW)}$$

The general circuit parameters are given by the relationships:

WO = PI2 * BW (PI2=6.28...)

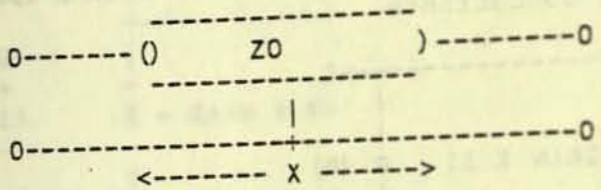
$$d = \frac{1}{GAIN} * [\ 1 + j * w/WO\ ]$$

A = G*d B = d

C = G*d*(10**-6) D = d*(10**-6)

H. LINE Network

Transmission line model for a rectangular stripline of length X and characteristic impedance Z0.



Parameters:

Z0 = characteristic impedance in ohms

TD = propagation delay in nanoseconds per unit X

X = line length

R = series line resistance in ohms per unit X. Defaults to zero.

TC = stripline conductor thickness in cm, used to calculate skin effect. Defaults to zero; no skin effect.

WC = stripline conductor width in cm, this optional parameter will calculate R in ohms/cm. If present, will over-ride an input value for R. Skin effect assumes WC>>TC and a symmetric stripline.

Q = relative conductor resistivity compared to copper. Defaults to one.

D = dielectric loss tangent, assumed to be independent of frequency, i.e., the real part of the shunt impedance has the same frequency dependence as the imaginary part (capacitive reactance).

F1, F2, F3, K1, K2 = generate a variation in line capacitance as a function of frequency. Default to zero.

Conductor losses will not be calculated if both R and WC are zero or absent.

The frequency variable capacitance is A piece-wise linear model with log frequency as follows:

The value of line C is computed from the input values for ZO and TD, and the capacitance per unit length is:

F < F1, C

F1 < F < F2, C * (1 + A ln(F/F1))

F2 < F C * (1 + A ln(F2/F1) + B ln(F/F2))

Where A = K1/ln(F2/F1), K1 is the fractional change in capacitance from F1 to F2

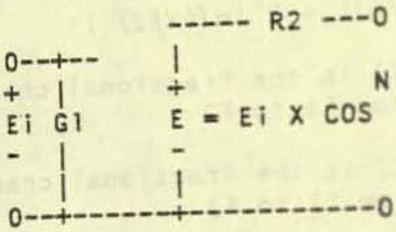
B = K2/ln(F3/F2), K2 is the fractional change in capacitance from F2 to F3

1. FILTER Network

Computes the response: $F(w) = \cos^{N}(w\ To)$ if Fo not specified

and

$F(w) = \cos^{N}(w/4*Fo)$ if Fo is specified



Parameters:

G1 = Shunt input conductance, mhos. Defaults to zero.

R1 = Shunt input resisance, ohms. If specified, replaces G1 where G1 = 1/R1.

R2 = Series output resistance, ohms. Defaults to zero.

To = Function period, nanoseconds. Must be present.

N = Power for raised cosine, defaults to 1.

Fo = Location of first zero in response in GHz. If present, overrides To, where:

$To = 1 / (4\ Fo)$.

The chain matrix parameters are given by:

$A = 1/F(w)$ $\quad B = R2 / F(w)$ $\quad C = G1 / F(w)$

$D = 0$

J. SHUNT Network

Provides a shunt impedance expressed as a polynomial:

$$Z = K \frac{A0 + A1*S + A2*S^2 + A3*S^3 + A4*S^4}{B0 + B1*S + B2*S^2 + B3*S^3 + B4*S^4}$$

Where: S = j w; the complex radian frequency

Units must be consistent, and a scaling in the frequency domain will scale in the output time domain.

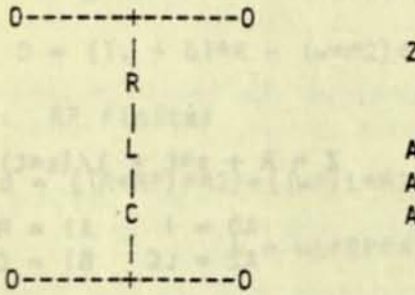
To avoid dividing by zero, the real part of the numerator is not allowed (by the program) to be less than 10^{-16}. In addition, if none of the Bi coefficients are specified, B0 is automatically set equal to one.

Input parameters are:

A0, A1, A2, A3, A4, B0, B1, B2, B3, and B4 as the coefficients described above. Each defaults to zero (with the exception of B0 as described above).

K = scaling parameter, defaults to 1

EXAMPLE:



Z = R + s*L + 1/(s*C)

A0 = 1
A1 = RC
A2 = LC

The chain matrix parameters are:

A = 1 B = 0

C = 1/Z D = 1

K. SERIES Network

Provides a series impedance expressed as a polynomial:

$$Z = K \frac{A0 + A1*S + A2*S^2 + A3*S^3 + A4*S^4}{B0 + B1*S + B2*S^2 + B3*S^3 + B4*S^4}$$

Where: S = j w; the complex radian frequency

Units must be consistent, and a scaling in the frequency domain will scale in the output time domain.

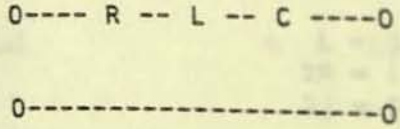
To avoid dividing by zero, the real part of the numerator is not allowed (by the program) to be less than 10^{-16}. In addition, if none of the Bi coefficients are specified, B0 is automatically set equal to one.

Input parameters are: A0, A1, A2, A3, A4, B0, B1, B2, B3, B4, and K.

A0, A1, A2, A3, A4, B0, B1, B2, B3, and B4 as the coefficients described above. Each defaults to zero (with the exception of B0 as described above).

K = scaling parameter, defaults to 1

EXAMPLE:



$$Z = R + s*L + 1/(s*C)$$

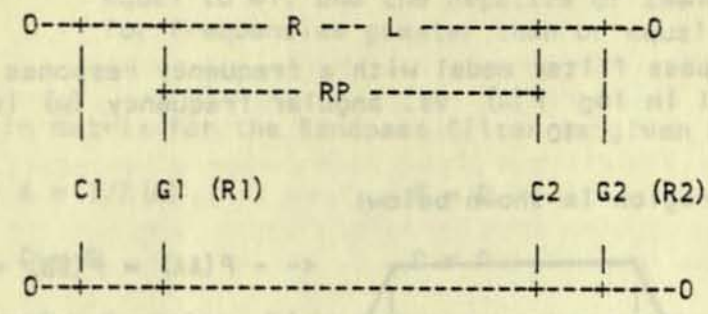
A0 = 1 A1 = RC
A2 = LC B1 = C

The chain matrix parameters are:

A = 1 B = Z

C = 0 D = 1

L. DELTA Network



Parameters:

R = series resistance in ohms, defaults to zero.

L = series inductance in nanohenries, defaults to zero.

RP = parallel series resistance in ohms, defaults to infinity.

C1, C2 = shunt capacitances in nanofarads, default to zero.

G1 (R1), G2, (R2) = shunt conductances in mhos, default to zero. Either G or R (in ohms) may be specified. If both are specified by mistake, the R value supersedes the G value.

The general circuit parameters are given by the following relationships.

Case I: RP defaults to infinity:

```
A = {1.+ G2*R - (w**2)*C2*L} + j*{wL*G2+wR*C2}

B = R + j*wL

C = [G1+G2 + R*{G1*G2-C1*C2*(w**2)}-XL*{wG1*C2+wG2*C1}] +
    j*{wC1+wC2+R*(wG1*C2+wG2*C1)+wL*(G1*G2-C1C2*(w**2)}

D = {1. + G1*R - (w**2)*C1*L} + j*{wL*G1 + wR*C1}
```

Case II: RP finite:

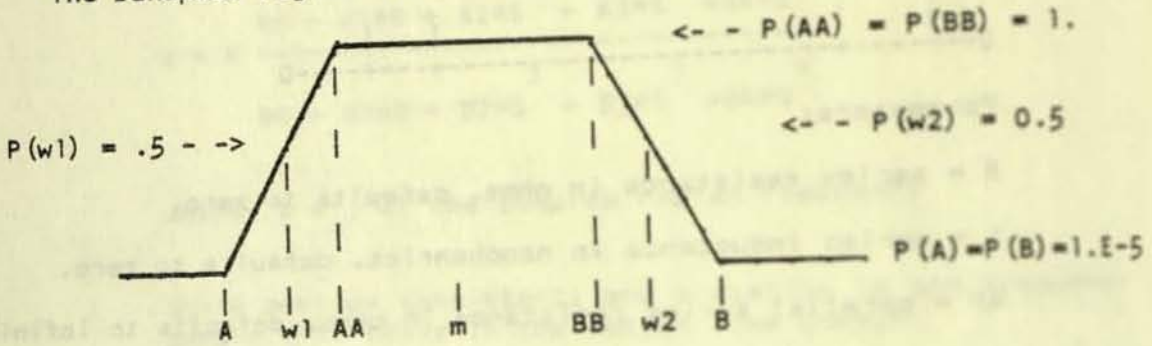
Let d = {(R+RP)**2}+{(w*)L**2}; a = RP*(R*(R+RP)+wL**2)/d; and b = wL*RP**2/d, then:

```
A = (1.+G2*a-w*b*C2) +j*(G2*b + w*a*C2)

B = a + j*b

C = {G1 + G2 + a*(G1*G2-C1C2*(w**2) - b*(wG1*C2+wG2*C1)} +
    j*[wC1+wC2+a*(wG1*C2+wG2*C1) + b*{G1*G2-C1C2*(w**2)}]

D = {1.+G1*a-w*b*C1} + j*{G1*b + w*a*C1}
```

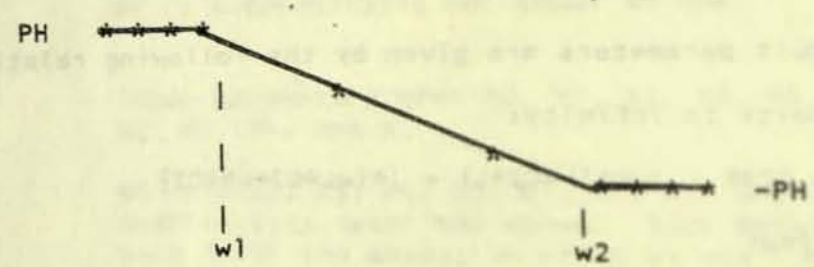
M. BPF Network

BPF is a bandpass filter model with a frequency response which is trapezoidal in $\log_{10} P(w)$ vs. angular frequency (w) in octaves.

The bandpass region is shown below:



The phase shift as a function of frequency is a constant, PH, for w<w1, and constant equal to -PH for w>w2. Between w1 and w2 the phase shift is linear with frequency, and zero at the midpoint, m, between w1 and w2:



Input parameters are:

F1 = frequency in Gigahertz at the low frequency 3 dB point of the trapezoid (corresponding to w1)

F2 = frequency in Gigahertz at the high frequency 3 dB point of the trapezoid (corresponding to w2)

S1 = slope of the low frequency edge of the trapezoid in dB/octave (default = 6 dB/octave)

S2 = slope of the high frequency edge of the trapezoid in dB/octave (default = 6 dB/octave)

K = gain (default = 1.)

PH = constant phase shift for frequencies less than or equal to w1, and the negative of the phase shift for frequencies greater than or equal to w2

The chain matrix for the Bandpass filter is given by:

$$A = 1/F(w) \qquad B = 0$$

$$C = 0 \qquad D = 0$$

The transfer function, F(w), for the bandpass filter is

$$F(w) = P(w) * \exp\{ j*PHI \}$$

where P(w) is given by:

$$P(w) = \begin{cases} 0 & \text{for } w < A \text{ or } w > B \\ 1 & \text{for } AA < w < BB \\ 0.5 * \{10 ** [\frac{s1}{10} * (\frac{w}{w1} - 1)]\} & \text{for } A < w < AA \\ 10 ** \{- \frac{s2}{10} * (\frac{w}{BB} - 1) \} & \text{for } BB < w < B \end{cases}$$

and the phase shift, PHI, is given by:

$$PHI = \begin{cases} PH & w < w1 \\ Ps * \{w-m\} & w1 < w < w2 \\ -PH & w < w2 \end{cases}$$

The values of A and B are found by setting P(A)=10**-4 and P(B)=10**-4.

$$AA = w1 * (1 + 3/s1)$$

$$BB = w2/(1 + 3/s2)$$

$$A = w1 * \{1 - 46.99/s1\}$$

$$B = BB * \{(50/s2) + 1\}$$

N. SRC Network

SRC is not a true network section in the sense that it models circuit behavior. Rather, it provides the interface between FOCAS, and the user defined file, SOURCE.DTA which contains a digitized input data stream of interest. The purpose of providing this interface is to allow the user to investigate complex edge behavior (turnon and turnoff ringing) of real optical sources.

SOURCE.DTA may contain either the time or frequency domain representation of the input waveform. The number of points, NT, in SDOURCE.DTA must be a power of 2, less than or equal to 8192. For time domain data, the input format is FORTRAN F20.3. For frequency domain data, it is FORTRAN 2E16.2. (See User's Guide, section II-A-3 for examples.)

In addition to requiring that a data file named SOURCE.DTA be in the directory, the explicit input parameters are:

NT = number of points in SOURCE.DTA; must be a power of two, less than or equal to 8192

DT = time interval between consecutive data points (ns) or angular frequency interval between consecutive points in Gigaradians/sec

PER = period of waveform in ns

TYPE = 1 if SOURCE.DTA contains frequency domain representat
2 if SOURCE.DTA contains time domain representation

R = source output resistance in ohms, defaults to zero

V. FOCAS PROGRAMS

FOCAS consists of two FORTRAN routines: FOCAS.FOR and FFT842.
A 20,000-page page file quota is required to link FOCAS.

A. DATA FLOW

FOCAS_ANALYZE is the skeleton of FOCAS. It calls all of the subroutines to read and interpret the simulation model, perform the network analysis, and write the output files.

a) SIMULATION MODEL DESCRIPTION DATA STRUCTURES:

Data about the network subsections' circuit parameter values is passed in the COMMON array RLC, which is created in subroutine INPUT. In addition to setting up circuit parameter matrix RLC, INPUT calls subroutine CONTROL which interprets the control statements in the simulation model; generates output pointers NEYE, NFF, and NPP; and defines structures to control the voltage level crossing calculations as described below.

INPUT writes the simulation model's filename to AFILE, and generates names for the output files based on AFILE. For a model in file MODELXYZ., these other files are:

```
BFILE = MODELXYZ.TG = time domain graph output file (REGIS)
DFILE = MODELXYZ.TD = ASCII time domain output data file
FGLE  = MODELXYZ.FG = power spectrum graph output file (REGIS)
FDLE  = MODELXYZ.FD = ASCII power spectrum data file
PGLE  = MODELXYZ.PG = phase angle graph output file (REGIS)
PDLE  = MODELXYZ.PD = ASCII phase angle data file
CRLE  = MODELXYZ.DAT = ASCII file with voltage crossing times
                       and extrema times and values; timing jitter
```

INPUT also creates the vector KIND, which is a pointer to the circuit type for each of the network subsections. KIND(N) is used to define the network type for the N-th subsection:

```
KIND(N) =    1  <--->  GEN
             2  <--->  TEE, etc.  (see Parameter Map for RLC for
                                   further detail - next section)
```

The period of the waveform, PER, is also passed in COMMON, and is taken to be the maximum of the periods of all the generators in the model.

The generator output impedance is taken to be the sum, RGEN, of the internal resistances of all the generators in the simulation model, and is passed in COMMON.

b) CONTROL DATA STRUCTURES

The following structures have their values defined in subroutine CONTROL, and are passed in COMMON:

NOUT = vector of pointers to locations of .OUT statements; NOUT(K) is the number of the circuit element after which the K-th .OUT statement appears; NOUT is initially set in SUBROUTINE CONTROL, and is later corrected in SUBROUTINE PRECALC if overlapping 1's case was encountered between 1st and last input GEN

MOUT = number of .OUT statements in the simulation model; MOUT must be less than or equal to 5

KD = vector of pointers to those .OUT statements which specify that extrema also be calculated; KD(J)=1 if the J-th .OUT also requires extrema, else KD(J)=0

IFF = pointer indicating that the number of frequency terms, NF, is to be limited to 8192 to speed up calculations

NCROSS(5,3) is a matrix of pointers to those .OUT statements which request that crossing times for V1, V2, or V3 be computed. NCROSS(K,J) is equal to one if Vj (j=1,2,3) is specified in the K-th .OUT statement in the simulation model.

VOLT(5,3) contains the values of the crossing levels; VOLT(K,J) indicates that the K-th .OUT statement (which is located after network subsection NOUT(K)) specifies that FOCAS is to calculate the times at which Vj = VOLT(K,J).

Additional control parameters that are defined in subroutine CONTROL and passed as parameters in the subroutine calling sequence are:

NEYE = 1 if an EYE pattern is required
NFF = 1 if power spectrum calculation required
NPP = 1 if phase shift vs. frequency calculation required
DC = vector of DC waveform offsets for the various .OUT statements
VOFF = vertical axis offset; set in the .END or .EYE statement to shift location of zero volts in the time domain graph
TOFF = vector of horizontal shifts for each of the .OUT statements
NC = number of circuit subsections in the model
TDIS = time interval to be graphed along the horizontal axis in time domain graphs
FS = full scale vertical axis length for time domain graph

Definition of the model's circuit and control descriptions is complete when subroutine INPUT is exitted. Next, subroutine PRECALC does some initial setup calculations and data massaging of circuit parameters. Intermediate calculations which need be performed only once because they are not frequency dependent are performed in PRECALC. An example of such a computation is the multiplication of resistance times capacitance, R*C, which is needed in each frequency calculation, but does not change, so can be done only once for all, then passed through RLC. PRECALC also checks and corrects for overlapping leading and trailing ones in the input data stream.
PRECALC also returns:

NGEN = number of generator in the model, after overlapping 1's have been eliminated

NF = number of frequency terms to be calculated

c) NETWORK CALCULATIONS AND DATA FLOW

After all the initial setup, FOCAS is ready to perform the detailed network analysis. This is all performed in the frequency domain, then inverse Fourier Transformed back into the time domain.

The frequency domain analysis is performed in subroutine TRANSFORM. At each frequency, TRANSFORM will calculate the network response at each of the MOUT .OUT statements. The treatment of the internal generators and the circuit subsections is somewhat different. The user specifies the generator's time domain behavior, but FOCAS automatically converts this waveform into its frequency spectrum. This is possible because the specific generator models in FOCAS all have simple, standard fourier series representations.

For additional generator models, which must exhibit more detailed source behavoir, one defines a function, SOURCE, which would Fourier Transform the input waveform using subroutine FFT842, then return the frequency component when called by subroutine TRANSFORM.

A circuit's behavior is determined by is chain matrix, which is a simple algebraic function of the circuit parameters. The A, B, C, D parameters of the chain matrix are calculated in the subroutines or functions called by the subsection type name, e.g. GEN, TEE, BPF. They are then stored in COMMON array ABCD.

ABCD = array of A, B, C, D parameters for each two terminal-pair network

```
ABCD(N,1,1) = A for N-th network
ABCD(N,1,2) = B for N-th network
ABCD(N,2,1) = C for N-th network
ABCD(N,2,2) = D for N-th network
```

TRANSFORM works through the entire network, one angular frequency at a time, starting at the DC term (w=0), and stepping by an amount equal to (2*PI/PER) up to the maximum angular frequency: (NF-1)*(2*PI/PER).

For each frequency, TRANSFORM will calculate the response of each subsection, then multiply the subsection responses together to find the entire network's response at the particular frequency. The frquency domain response thus calculated is returned in the COMMON array EW(5,8192). The first index of EW points to which of the MOUT .OUT waveforms is described. The second index indicates the frequency term. Thus,

EW(K,IX) = complex frequency response of the network after the NOUT(K)-th circuit subsection, (i.e. after the K-th .OUT statement)

If an overflow condition is detected during the complex multiplications in TRANSFORM, IFLAG is set, and the frequency calculations will be terminated before all NF terms have been computed. An error message is then printed to alert the user.

Once the complex frequency domain response of the network is known, the power spectrum, phase shift, and time domain computations are performed directly in FOCAS_ANALYZE. These waveforms are all calculated at one network location at a time. At the K-th .OUT statement (after subsection NOUT(K)), the real and imaginary parts of the frequency response are found:

```
RX(IW) = real part of EW(K,IW)
RY(IW) = imaginary part of EW(K,IW).
```

The power spectrum is then simply the complex magnitude of EW and is passed to GRAPHICS in the COMMON array FOUT.

```
EOUT(K,IW) = SQRT((RX(IW)**2)+(RY(IW)**2))
```

The phase angle is the arctan(RY(IW)/RX(IW)) and is also passed through COMMON array EOUT.

To find the time domain response of the network after the K-th .OUT, subroutine FFT842 must inverse fourier transform the frequency repsonse after the NOUT(K)-th subsection. The input waveform and its transform are passed to and from FFT842 in arrays RX and RY in its calling sequence.

The time domain response is the real part of the inverse fourier transform of the frequency response. It is passed to support routines CROSS and GRAPHICS in COMMON array EOUT(5,8192).

Subroutines CROSS and GRAPHICS operate on the waveforms stored in array EOUT. Thus, when graphing the power spectrum rather than time domain response on the network, FOUT must be copied into EOUT. Similarly, when graphing the phase angle output, GOUT must be copied into EOUT.

2. CALLING STRUCTURES

FOCAS_ANALYZE

FOCAS_ANALYZE is the skeleton of FOCAS. It calls all of the subroutines to read and interpret the simulation model, perform the network analysis, and write the output files.

VARIABLES: CHAR AFILE, ANF, CHAR BELL, CHAR BFILE, CHAR CRLE, CHAR DFILE, ERR, CHAR ESC, FDIS, CHAR FDLE, CHAR FGLE, FS, I, ID, IERR, IFF, IK, IRR, K, MOUT, NC, NEYE, NF, NF1, NFF, NGEN, NPP, CHAR PDLE, PER, CHAR PGLE, PI2, RGEN, TDIS, VOFF, CHAR W132, CHAR W80, ZE

ARRAYS: ABCD, DC, EDGE, EOUT, EW, KD, KIND, NCROSS, NOUT, RL RX, RY, TOFF, VOLT

FUNCTIONS AND SUBROUTINES REFERENCED: CROSS, FFT842, GRAPHICS, INPUT, MTH$ATAN2, MTH$SQRT, PRECALC, TRANSFORM

SUBROUTINE BPF(N,W)

Returns the A, B, C, D parameters for a Bandpass Filter modelled as a trapeziod in Log P(w) vs w. (TYPE 15 CIRCUIT)

VARIABLES: A, AA, B, BB, F, FF, K, N, PHI, PI2, S1, S2, W, W1, W2, ZERO

ARRAYS: ABCD, RLC

FUNCTIONS AND SUBROUTINES REFERENCED: MTHCOS, MTHSIN

FUNCTION BRACKET(T,X)

* KEEPS THE TRUNCATED VALUE OF T BETWEEN O AND X

VARIABLES: T, X

SUBROUTINE CONTROL(M,NC,ITEM,TDIS,TOFF,DC,FS,VOFF,IERR,NEND,NEYE, NFF,NPP)

* PERFORMS CONTROL STATEMENT PROCESSING

VARIABLES: FS, IERR, IFF, J, K, M, MOUT, NC, NCONT, CHAR ND, N NEYE, NFF, NJ, CHAR NOW, NPP, NVAR, RGEN, CHAR SPA, CHAR TAB, TDIS, VOFF

ARRAYS: DC, CHAR ITEM, KD, KIND, CHAR NAME, NCROSS, CHAR ND NOUT, TOFF, VOLT

FUNCTIONS AND SUBROUTINES REFERENCED: DIGIT

SUBROUTINE CROSS(AFILE,CRLE,NF,TDIS,TOFF,NEYE)

* COMPUTES THE CROSSOVER TIMING, MAXIMA & MINIMA, TIMING JITTER, AND PER CENT NOISE MARGIN OF WAVEFORMS PASSED IN COMMON ARRAY EOUT. WRITES RESULTS TO FILE CRLE

VARIABLES: CHAR AFILE, AJ, AN1, AN2, ANF, AX1, AX2, CHAR CRLE, DELT, DELTT, I, IC1, IC2, IN, INN, INO, INO2, INX, IVH, IVL, IVLO, IVM, IXN, IXX, J, J1, J2, JH, JL, JLO, JP, JT, K, KDER, M, MAX, MAXO, MOUT, N, N1, N2, NDER, NEYE, NF, NM, PER, RGEN, TDIS, TJ1, TJ2, TO, V1, V2, VO

ARRAYS: DERMAX, DERMIN, EOUT, EW, IVN, IVX, KD, KIND, MAXMIN, NCROSS, NMAT, NN, NOUT, NX, RLC, RN, TD, TMAT, TOFF, VMAT, VOLT, XTD

FUNCTIONS AND SUBROUTINES REFERENCED: FORSCLOSE, FORSOPEN

SUBROUTINE DELTA(N,W)

* SOLVES FOR THE A,B,C,D PARAMETERS FOR THE PI CIRCUIT FOR THE Nth NETWORK ELEMENT AT FREQUENCY W (TYPE 3 CIRCUIT)

VARIABLES: A, B, BC1, BC2, D, G1, G2, N, PI2, R, RP, W, XL, ZERO

ARRAYS: ABCD, RLC

FUNCTION DIGIT(M,ITEM,IERR)

* CONVERTS A CHARACTER STRING TO A NUMBER

VARIABLES: A, I, IERR, J, K, L, M, CHAR N, PLUS

ARRAYS: CHAR INTEG, CHAR ITEM, TEMP

SUBROUTINE EAMP(N,W)

* RETURNS THE A, B, C, D PARAMETERS FOR A VOLTAGE DEPENDENT VOLTAGE SOURCE WITH EXTERNAL RESISTANCE AND A DEFINED BANDWIDTH (TYPE 6 CIRCUIT)

VARIABLES: D, N, PI2, W, WO, ZERO

ARRAYS: ABCD, RLC

SUBROUTINE FFT842(IN, N, X, Y)

* THIS PROGRAM REPLACES THE VECTOR Z=X+iY BY ITS FINITE DISCRE
COMPLEX FOURIER TRANSFORM IF IN=0 (FREQUENCY DOMAIN TO TIME
DOMAIN). THE INVERSE TRANSFORM IS CALCULATED FOR IN=1. IT
PERFORMS AS MANY BASE 8 ITERATIONS AS POSSIBLE AND THEN FINIS
WITH A BASE 4 ITERATION OR A BASE 2 ITERATION IF NEEDED.

VARIABLES: FI, FN, I, IJ, IN, IPASS, J, J1, J10, J11, J12, J13,
J14, J2, J3, J4, J5, J6, J7, J8, J9, J1, L1, L10, L11
L12, L13, L14, L15, L2, L3, L4, L5, L6, L7, L8, L9,
LENGT, M, N, N2POW, N8POW, NT, NTHPO, NXTLT, P7, PI2, R

ARRAYS: L, X, Y

FUNCTIONS AND SUBROUTINES REFERENCED: ATAN, SQRT, R2TX, R4TX, R8TX

SUBROUTINE FIBER(N,W)

* OPTICAL FIBER MODEL - Returns A, B, C, D parameters
(TYPE 7 CIRCUIT)

VARIABLES: AO, AW, H, N, P, PHI, PI2, W, WTC, WTM, ZERO

ARRAYS: ABCD, RLC

FUNCTIONS AND SUBROUTINES REFERENCED: MTH$DCOS, MTH$DEXP, MTH$D

SUBROUTINE FILTER(N,W)

* DELAY LINE FILTER MODEL (TYPE 12 CIRCUIT)

VARIABLES: M, N, P, PI2, Q, W, X, ZERO

ARRAYS: ABCD, RLC

FUNCTIONS AND SUBROUTINES REFERENCED: MTH$DCOS

FUNCTION GEN(N,W)

* RETURNS THE VOLTAGE GENERATOR FREQUENCY COMPONENT
FOR THE Nth NETWORK ELEMENT AT FREQUENCY W (TYPE 1 CIRCU

VARIABLES: AF, AMP, AP, AR, BETA, C, DF, DN, DNC, DO, N, PER,
PHI, PI2, PW, TFM, TRM, W, WO, ZERO

ARRAYS: ABCD, RLC

FUNCTIONS AND SUBROUTINES REFERENCED: MTH$DCOS, MTH$DSIN

```
SUBROUTINE GRAPHICS(AFILE,BFILE,DFILE,NF,TDIS,TOFF,DC,FS,VOFF,
                    1 NEYE,ID)

      * WRITES TO THE OUTPUT FILES

      VARIABLES:  CHAR AFILE, AFS, AMP, ANF, AO, AVOFF, CHAR BFILE,
                  DELT, CHAR DFILE, DT, EO, EOFF, ESC, FS, FWO, I,
                  ID, IFF, IN, J, K, MM, MOUT, NEYE, NF, NP, NT, NT1,
                  OUT, P2, PER, RGEN, T, TC, TDIS, TG, TG2, TGO,
                  TINC, TO, TOUT, TS, VOFF, Y

      ARRAYS:     DC, EOUT, EW, KD, KIND, NOUT, ORD, OUTPUT, TOFF

      FUNCTIONS AND SUBROUTINES REFERENCED:  BRACKET, FOR$CLOSE, FOR$OPEN

FUNCTION GSIN(N,W)

      * RETURNS THE SINEWAVE GENERATOR FREQUENCY COMPONENT
        FOR THE Nth NETWORK ELEMENT AT FREQUENCY W (TYPE 8 CIRCUIT)

      VARIABLES:  DN, N, PER, PHI, PI2, W, WO, ZERO

      ARRAYS:     ABCD, RLC

      FUNCTIONS AND SUBROUTINES REFERENCED: MTH$DCOS, MTH$DSIN


SUBROUTINE HPF(N,W)

      * SOLVES FOR THE A,B,C,D PARAMETERS FOR THE HPF CIRCUIT
        FOR THE Nth NETWORK ELEMENT AT FREQUENCY W (TYPE 10 CIRCUIT)

      VARIABLES:  AI, AR, B, D, GC, GL, GLW, N, PI2, W, ZERO

      ARRAYS:     ABCD, RLC

SUBROUTINE IAMP(N,W)

      * A VOLTAGE DEPENDENT CURRENT SOURCE WITH EXTERNAL RESISTANCE
        AND A DEFINED BANDWIDTH (TYPE 9 CIRCUIT)

      VARIABLES:  D, N, PI2, W, WO, ZERO

      ARRAYS:     ABCD, RLC
```

```
SUBROUTINE INFILTER(NVAR,K,ITEM)

      * Special Handling for FILTER Network, places the implied
        after the Sine, Cosine, or Gaussian statements

    VARIABLES:  I, K, NVAR

    ARRAYS:     CHAR ITEM


SUBROUTINE INPUT(AFIL,BFIL,DFIL,FGL,FDL,PGL,PDL,NC,TDIS,TOFF,DC,
                 CRL,FS,VOFF,NEYE,ID,nff,npp)

    VARIABLES:  CHAR AFIL, CHAR BFIL, CHAR CRL, CROSS_SIZE, CHAR DF
                CHAR FDL, CHAR FGL, FREQ_DATA_SIZE, FREQ_GRAPH_SIZE
                FS, I, CHAR ICDAT, ID, IEIERR, IFF, IT, J, K, LT, M
                MOUT, NC, CHAR ND, NEND, NEYE, NFF, NKIND, CHAR NOM
                NPP, NVAR, CHAR PDL, PER, CHAR PGL, PHASE_DATA_SIZE
                PHASE_GRAPH_SIZE, PI2, RGEN, SIZE, CHAR SPA, STATUS
                CHAR TAB, TDIS, TIME_DATA_SIZE, TIME_GRAPH_SIZE, VO

    ARRAYS:  ABCD, DC, CHAR ITEM, KD, KIND, CHAR NAME, CHAR NDATA,
             NOUT, RLC, TOFF

    FUNCTIONS AND SUBROUTINES REFERENCED:  CONTROL, DIGIT,
            FOR$CLOSE, FOR$OPEN, INFILTER, LIB$SIGNAL, STR$TRIM


SUBROUTINE LINE(N,W)

      * SOLVES FOR THE A,B,C,D PARAMETERS FOR A TRANSMISSION LIN
        FOR THE Nth NETWORK ELEMENT AT FREQUENCY W (TYPE 4 CIRCU

    VARIABLES:  A, B, BC, C, CON, D, GAM, GAMJ, N, PI2, Q, QO,
                R, RT, W, W1, W2, X, XL, Y, Z, ZERO, ZO

    ARRAYS:     ABCD, RLC

    FUNCTIONS AND SUBROUTINES REFERENCED:  MTH$ALOG, MTH$CDCOS,
                MTH$CDSIN, MTH$CDSQRT, MTH$COS, MTH$COSH, MTH$SIN,
                MTH$SINH, MTH$SQRT


SUBROUTINE MABCD(A,M,IFLAG)

      * SOLVES THE MATRIX MULTIPLICATION A(I,J) = ABCD(M,I,J)*A

    VARIABLES:  C, I, IFLAG, J, K, M, N, PI2, ZERO

    ARRAYS:     A, ABCD, B, RLC

    FUNCTIONS AND SUBROUTINES REFERENCED:  MTH$CDABS
```

```
SUBROUTINE MOVE(A,M)

      *  MOVES ABCD(M,I,J) TO A(I,J)

    VARIABLES:  I, J, M, N, PI2, ZERO

    ARRAYS:     A, ABCD, B, RLC


SUBROUTINE PRECALC(NC,NF,TDIS,TOFF,NGEN)

   *  CALCULATION OF CERTAIN CIRCUIT PARAMETERS TO REDUCE
      RECURRENT CALCULATIONS, INITIALIZATIONS, & OVERLAPPING 1
      CORRECTION

    VARIABLES:  A, ALC, ANF, B, BT, BW, C, CORR, DEL, DEL1, DN, DN1,
                DP, DT, ERR, F1, F2, I, IBG, ICK, ICT, IDF, IFF, IGEN,
                II, J, K, MOUT, N1, NC, NDGEN, NF, NGEN, NSRC, NT, P,
                P1, P2, PER, PI2, PN, PN1, R1  R2, RGEN, T, TDF50,
                TDIS, TDR50, TDR51, TDR52, TF, TR, TR1, TR2, TRM, TRO,
                TYPE, WO, X, XC, XM, ZERO

    ARRAYS:     ABCD, DC1,EDGE,KD, KIND, NOUT, RLC, TOFF, XX, YY

    FUNCTIONS AND SUBROUTINES REFERENCED:  FFT842, FOR$CLOSE, FOR$OPEN,
                MTH$ALOG, MTH$ALOG10, MTH$DSQRT, MTH$SQRT, RISE


FUNCTION RISE(WGEN,TR,A)

      *  COMPUTES THE DELAY FOR THE INPUT WAVEFORM
         WGEN IS THE GENERATOR FILTER 3DB FREQUENCY (RAD/SEC),
         TR IS THE 20 % TO 80 % GENERATOR RISETIME
         A IS THE FRACTIONAL AMPLITUDE

    VARIABLES:  A, C, DF, F, I, TR, WGEN, X, Y

    FUNCTIONS AND SUBROUTINES REFERENCED:  MTH$EXP

SUBROUTINE R2TX(NTHPO, CRO, CR1, CIO, CI1)

      RADIX 2 ITERATION SUBROUTINE OF FFT842

   VARIABLES:  FI1, K, NTHPO, R1

   ARRAYS:     CIO, CI1, CRO, CR1
```

```
SUBROUTINE R4TX(NTHPO, CRO, CR1, CR2, CR3, CI0, CI1, CI2, CI3)

      * RADIX 4 ITERATION SUBROUTINE OF FFT842

      VARIABLES:  FI1, FI2, FI3, FI4, K, NTHPO, R1, R2, R3, R4

      ARRAYS:     CI0, CI1, CI2, CI3, CRO, CR1, CR2, CR3

SUBROUTINE R8TX(NXTLT, NTHPO, LENGT, CRO, CR1, CR2, CR3, CR4, CR5, CR6,
                CR7, CI0, CI1, CI2, CI3, CI4, CI5, CI6, CI7)

      * RADIX 8 ITERATION SUBROUTINE OF FFT842

      VARIABLES:  AI0, AI1, AI2, AI3, AI4, AI5, AI6, AI7, ARO, AR1, A
                  AR3, AR4, AR5, AR6, AR7, ARG, BI0, BI1, BI2, BI3, B
                  BI5, BI6, BI7, BRO, BR1, BR2, BR3, BR4, BR5, BR6, B
                  C1, C2, C3, C4, C5, C6, C7, J, K, LENGT, NTHPO, NXT
                  P7, PI2, S1, S2, S3, S4, S5, S6, S7, SCALE, TI, TR

      ARRAYS:     CI0, CI1, CI2, CI3, CI4, CI5, CI6, CI7, CRO, CR1, C
                  CR3, CR4, CR5, CR6, CR7

      FUNCTIONS AND SUBROUTINES REFERENCED:  COS, SIN


SUBROUTINE  SERIES(N,W)

      *  POLYNOMIAL MODEL FOR A SERIES IMPEDANCE
         UP TO FOUR ZEROS AND FOUR POLES  (TYPE 14 CIRCUIT)

      VARIABLES:  N, PI, PI2, PR, W, W2, W3, W4, ZERO, ZI, ZR

      ARRAYS:     ABCD, RLC


SUBROUTINE SHUNT(N,W)

      *  POLYNOMIAL MODEL FOR A SHUNT IMPEDANCE
         UP TO FOUR ZEROS AND FOUR POLES (TYPE 13 CIRCUIT)

      VARIABLES:  N, PI, PI2, PR, W, W2, W3, W4, ZERO, ZI, ZR

      ARRAYS:     ABCD, RLC


FUNCTION SOURCE(N,W)

      *  RETURNS THE SOURCE GENERATOR FREQUENCY COMPONENT
         FOR THE Nth NETWORK ELEMENT AT FREQUENCY W
         TYPE 16 CIRCUIT - SRC, experimental data

      VARIABLES:  AX, IFOLD, IX, N, PER, PI2, W, ZERO

      ARRAYS:     ABCD, EDGE, RLC
```

SUBROUTINE STUB(N,W)

* SOLVES FOR THE A,B,C,D PARAMETERS FOR A TRANSMISSION LINE STUB FOR THE Nth ELEMENT AT FREQUENCY W (TYPE 5 CIRCUIT)

VARIABLES: A, BC, C, CON, D, GAM, GAMJ, N, PI2, Q, QO, R, RT, W, X, XL, Y, YL, Z, ZERO, ZO

ARRAYS: ABCD, RLC

FUNCTIONS AND SUBROUTINES REFERENCED: MTH$ALOG, MTH$CDCOS, MTH$CDSIN, MTH$CDSQRT, MTHCOS, MTHCOSH, MTHSIN, MTHSINH, MTH$SQRT

SUBROUTINE TEE(N,W)

* SOLVES FOR THE A,B,C,D PARAMETERS FOR THE TEE CIRCUIT FOR THE Nth NETWORK ELEMENT AT FREQUENCY W (TYPE 2 CIRCUIT)

VARIABLES: BC, G, N, PI2, R1, R2, RAC, RT, W, W1, W2, XL1, XL2, ZERO

ARRAYS: ABCD, RLC

SUBROUTINE TRANSFORM(NC,NF,NGEN)

* COMPUTES THE FREQUENCY DOMAIN RESPONSE OF THE NETWORK

VARIABLES: EG, I, IFF, IFLAG, K, KOUT, M, MM, MOUT, N1, N2, NC, NF, NGEN, NS, PER, PI2, RGEN, W, WO, ZERO

ARRAYS: A, ABCD, B, EOUT, EW, KD, CHAR NODENAME, NOUT, RLC

FUNCTIONS AND SUBROUTINES REFERENCED: BPF, DELTA, EAMP, FIBER, FILTER, GEN, GSIN, HPF, IAMP, LINE, MABCD, MOVE, SERIES, SHUNT, SOURCE, STUB, TEE, XFORMER

SUBROUTINE XFORMER(N,W)

* TRANSFORMER MODEL (TYPE 11 CIRCUIT)

VARIABLES: M, N, PI2, Q, W, ZERO

ARRAYS: ABCD, RLC

B. PARAMETER MAP FOR THE ARRAY RLC

The array RLC in FOCAS passes circuit parameters between the various subroutines. Element RLC(n,p) is the p-th circuit parameter of n-th subsection of the simulation model. RLC is used in conjunction with array KIND(14) to completely specify a subsection. The numerical value of KIND(n) identifies the subsection circuit type. For example if KIND(53) = 7, the fifty-third element in the simulation model is an optical fiber. RLC(53,8) is the length (X) of the fiber as shown below.

KIND = Type p	1 GEN	2 TEE	3 DELTA	4 LINE	5 STUB	6 EAMP	7 FIBER	8 SIN	9 IAM
1	PW	R1	G1	ZO	ZO	GAIN	AO	F	GAI
2	TR	R2	G2	TD	TD	R	AW	AMP	R
3	PER	G	C1	X	X	F	FWHM	DC	G
4	AMP	L1	C2	R	R		FI	DEL	F
5	DEL	L2	L	TC	TC		D	R	
6	BW	C	R	D	D		QM		
7	DC	F1	R1	K1			QC		
8	R	F2	R2	K2			X		
9	TF	R	RP	WC	GL		LC		
10				Q	RL		LO		
11				F1	CL		SO		
12				F2	WC				
13				F3					
14									
15									
16	WO	W1		CONST	CONST	WO	A	WO	
17		W2					TC		
18							A'		
19	TFM			L	L		TM		
20	TRM			C	C				

1-13 INPUT PARAMETERS
14-20 PARAMETERS COMPUTED IN SUBROUTINE PRECALC

PARAMETER MAP FOR THE ARRAY RLC (cont.)

IND	10	11	12	13	14	15	16
ype	HPF	XFORMER	FILTER	SHUNT	SERIES	BPF	SRC
0							
1	C	L1	TO	A0	A0	F1	EF
2	L	L2	N	A1	A1	F2	PW
3	G	M	R2	A2	A2	S1	PER
4	R	K	G1	A3	A3	S2	AMP
5		N	R1	A4	A4	K	DEL
6			F0	B0	B0	PH	NT
7				B1	B1		DT
8				B2	B2		TYPE
9			SINE	B3	B3		R
10			COS	B4	B4		
11			GAUS	K	K		
12							
13							
14						PS	
15							
16		L1/M				AA	
17		L2/M				BB	
18		M-L1L2/M				A	
19	G/C					B	
20	G*L		TYPE			M	

1-13 INPUT PARAMETERS
14-20 PARAMETERS COMPUTED IN SUBROUTINE PRECALC

TYPE = 1, SINE (not inplemented)
2, COSINE
3, GAUSSIAN (not inplemented)

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
                                                                   C
                                                                   C
                    COPYRIGHT (c) 1987 BY                          C
        DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.               C
                                                                   C
This software is furnished under a license and may be used and  copied  C
only  in  accordance  with  the  terms  of  such  license and with the  C
inclusion of the above copyright notice.  This software or  any  other  C
copies  thereof may not be provided or otherwise made available to any  C
other person.  No title to and ownership of  the  software  is  hereby  C
transferred.                                                      C
                                                                   C
The information in this software is subject to change  without  notice  C
and  should  not  be  construed  as  a commitment by Digital Equipment  C
Corporation.                                                      C
                                                                   C
DIGITAL assumes no responsibility for the use or  reliability  of  its  C
software on equipment which is not supplied by DIGITAL.           C
                                                                   C
+                                                                  C
                                                                   C
FACILITY:    FOCAS                                                 C
                                                                   C
ABSTRACT:    FOCAS is a simulation tool for the analysis of electronic  C
             networks with fiber optic cable links.               C
                                                                   C
ENVIRONMENT: VAX/VMS V4.4 User Mode                                C
                                                                   C
AUTHORS:       Ann Ewalt, Mike Taylor, Don Marshall                C
                                                                   C
CREATION DATE:   3-FEBRUARY-1987                                   C
                                                                   C
MODIFIED BY: Ann Ewalt 2/16/87, 3/9/87, 3/11/87                    C
------------------------------------------------------------------ C


CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
                                                              C
   MAIN PROGRAM FOCAS.FOR   Fiber Optic CAble Simulator        C
                                                              C
   Frequency Domain Analysis of Cascaded Networks             C
   Maximum number of networks allowed = 500    Rev 3/1187      C
                                                              C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
C
C     FOCAS_ANALYZE is the skeleton of FOCAS.  It calls all
C     of the subroutines to read and interpret the simulation
C     model, perform the network analysis, and write the
C     output files.
C
C     1.  MODEL DESCRIPTION DATA STRUCTURES:
C         ----------------------------------
C
C     Data about the network subsections' circuit parameter values
C     is passed in the COMMON array RLC, which is created in subroutine
C     INPUT.  In addition to setting up circuit parameter matrix RLC,
C     INPUT calls subroutine CONTROL which interprets the control
C     statements in the simulation model; generates output pointers
C     NEYE, NFF, and NPP; and defines structures to control the
C     voltage level crossing calculations as described below.
C
C     INPUT writes the simulation model's filename to AFILE,
C     and generates names for the output files based on AFILE.
C     For a model in file MODELXYZ., these other files are:
C
C     BFILE = MODELXYZ.TG = time domain graph output file (REGIS)
C     DFILE = MODELXYZ.TD = ASCII time domain output data file
C     FGLE  = MODELXYZ.FG = power spectrum graph output file (REGIS)
C     FDLE  = MODELXYZ.FD = ASCII power spectrum data file
C     PGLE  = MODELXYZ.PG = phase angle graph output file (REGIS)
C     PDLE  = MODELXYZ.PD = ASCII phase angle data file
C     CRLE  = MODELXYZ.DAT = ASCII file with voltage crossing times
C                            and extrema times and values; timing jitter
C
C     INPUT also creates the vector KIND, which is a pointer to the
C     circuit type for each of the network subsections.  KIND(N) is
C     used to define the network type for the N-th subsection:
C
C     KIND(N) =      1  <--->  GEN
C                    2  <--->  TEE
C                    3  <--->  DELTA
C                    4  <--->  LINE
C                    5  <--->  STUB
C                    6  <--->  EAMP
C                    7  <--->  IAMP
C                    8  <--->  FIBER
C                    9  <--->  SINE
C                   10  <--->  HPF
C                   11  <--->  XFORMER
C                   12  <--->  FILTER
C                   13  <--->  SHUNT
C                   14  <--->  SERIES
C                   15  <--->  BPF
C                   16  <--->  SRC
C
C     The period of the waveform, PER, is also passed in COMMON,
C     and is taken to be the maximum of the periods of all the
C     generators in the model.  The generator output impedance is
C     the sum, RGEN, of the internal resistances of all the generators
C     in the simulation model, and is passed in COMMON.
C!
```

2. CONTROL DATA STRUCTURES

The following structures have their values defined in subroutine CONTROL, and are passed in COMMON:

NOUT = vector of pointers to locations of .OUT statements; NOUT(K) is the number of the circuit element after which the K-th .OUT statement appears; NOUT is initially set in SUBROUTINE CONTROL, and is later corrected in SUBROUTINE PRECALC if overlapping 1's case was encountered between 1st and last input GEN

MOUT = number of .OUT statements in the simulation model; MOUT must be less than or equal to 5

KD = vector of pointers to those .OUT statements which specify that extrema also be calculated; KD(J)=1 if the J-th .OUT also requires extrema, else KD(J)=0

IFF = pointer indicating that the number of frequency terms, NF, is to be limited to 8192; used to speed up calculations

NCROSS(5,3) is a matrix of pointers to those .OUT statements which request that crossing times for V1, V2, or V3 be computed. NCROSS(K,J) is equal to one if Vj (j=1,2,3) is specified in the K-th .OUT statement in the simulation model.

VOLT(5,3) contains the values of the crossing levels; VOLT(K,J) indicates that the K-th .OUT statement (which is located after network subsection NOUT(K)) specifies that FOCAS is to calculate the times at which Vj = VOLT(K,J).

Additional control parameters that are defined in subroutine CONTROL and passed as parameters in the subroutine calling sequence are:

NEYE = 1 if an EYE pattern is required
NFF = 1 if power spectrum calculation required
NPP = 1 if phase shift vs. frequency calculation required
DC = vector of DC waveform offsets for the various .OUT statements
VOFF = vertical axis offset; set in the .END or .EYE statement to shift location of zero volts in the time domain graph
TOFF = vector of horizontal shifts for each of the .OUT statements
NC = number of circuit subsections in the model
TDIS = time interval to be graphed along the horizontal axis in time domain graphs
FS = full scale vertical axis length for time domain graph

```
C
C
C       Definition of the model's circuit and control descriptions is
C       complete when subroutine INPUT is exitted.  Next, subroutine
C       PRECALC does some initial setup calculations and data massaging
C       of circuit parameters.  Intermediate calculations which need be
C       performed only once because they are not frequency dependent are
C       performed in PRECALC.  An example of such a computation is the
C       multiplication of resistance times capacitance, R*C, which is
C       needed in each frequency calculation, but does not change, so can
C       be done only once for all, then passed through RLC.  PRECALC
C       also checks and corrects for overlapping leading and trailing
C       ones in the input data stream.
C       PRECALC also returns:
C                       NGEN  = number of generator in the model,
C                               after overlapping 1's have been
C                               eliminated
C                       NF    = number of frequency terms to be calculated
C
C
C       3.  NETWORK CALCULATIONS AND DATA FLOW
C           ----------------------------------
C
C       After all the initial setup, FOCAS is ready to perform
C       the detailed network analysis.  This is all performed
C       in the frequency domain, then inverse Fourier Transformed
C       back into the time domain.
C
C       The frequency domain analysis is performed in subroutine
C       TRANSFORM.  At each frequency, TRANSFORM will calculate the
C       network response at each of the MOUT  .OUT statements.
C       The treatment of the internal generators and the circuit
C       subsections is somewhat different.  The user specifies
C       the generator's time domain behavior, but FOCAS automatically
C       converts this waveform into its frequency spectrum.
C       This is possible because the specific generator models in
C       FOCAS all have simple, standard fourier series representations.
C
C       For additional generator models, which must exhibit more
C       detailed source behavoir, one defines a function, SOURCE,
C       which would Fourier Transform the input waveform using
C       subroutine FFT842, then return the frequency component
C       when called by subroutine TRANSFORM.
C
C       A circuit's behavior is determined by is chain matrix,
C       which is a simple algebraic function of the circuit
C       parameters.  The A, B, C, D parameters of the chain
C       matrix are calculated in the subroutines or functions
C       called by the subsection type name, e.g. GEN, TEE, BPF.
C       They are then stored in COMMON array ABCD.
C
C       ABCD  = array of A, B, C, D parameters for each two terminal-
C               pair network;
C
C                 ABCD(N,1,1) = A for N-th network
C                 ABCD(N,1,2) = B for N-th network
C                 ABCD(N,2,1) = C for N-th network
C                 ABCD(N,2,2) = D for N-th network
C!
```

TRANSFORM works through the entire network, one angular frequency at a time, starting at the DC term (w=0), and stepping by an amount equal to (2*PI/PER) up to the maximum angular frequency: (NF-1)*(2*PI/PER).

For each frequency, TRANSFORM will calculate the response of each subsection, then multiply the subsection responses together to find the entire network's response at the particular frequency. The frequency domain response thus calculated is returned in the COMMON array EW(5,32768). The first index of EW points to which of the MOUT .OUT waveforms is described. The second index indicates the frequency term. Thus,

```
EW(K,IX) = complex frequency response of the network
           after the NOUT(K)-th circuit subsection,
           (i.e. after the K-th  .OUT statement)
```

If an overflow condition is detected during the complex multiplications in TRANSFORM, IFLAG is set, and the frequency calculations will be terminated before all NF terms have been computed. An error message is then printed to alert the user. To find the time domain response of the network after the K-th .OUT, subroutine FFT842 must inverse inverse fourier transform the frequency response, EW(K;), after the NOUT(K)-th subsection. The input waveform and its transform are passed to and from FFT842 as their real and imaginary parts in arrays RX and RY in the calling sequence.

The time domain response is the real part of the inverse fourier transform of the frequency response. It is passed to support routines CROSS and GRAPHICS in COMMON array EOUT(5,32768).

Once the complex frequency domain response of the network is known, the power spectrum, phase shift, and time domain computations are performed directly in FOCAS_ANALYZE. These waveforms are all calculated at one network location at a time. At the K-th .OUT statement (after subsection NOUT(K)), the real and imaginary parts of the frequency response are found:

```
RX(IW) = real part of EW(K,IW)
RY(IW) = imaginary part of EW(K,IW).
```

The power spectrum is then simply the complex magnitude of EW. The power spectrum is passed to GRAPHICS in COMMON array EOUT.

```
EOUT(K,IW) = SQRT((RX(IW)**2)+(RY(IW)**2))
```

The phase angle is the arctan(RY(IW)/RX(IW)) and is also passed to GRAPHICS in COMMON array EOUT.

Subroutines CROSS and GRAPHICS operate on the waveforms stored in array EOUT.

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc!
```

```
      CHARACTER*1 ESC,BELL
      CHARACTER*4 W80,W132
      CHARACTER*32 AFILE,BFILE,DFILE,FGLE,FDLE,PGLE,PDLE,CRLE
      DIMENSION RX(32768),RY(32768),TOFF(5),DC(5)
      DOUBLE PRECISION PER,PI2
      COMPLEX*16 ABCD(500,2,2),EW(5,32768),ZERO
      complex*16 edge(8192)
      COMMON/NET/RLC(500,20),ABCD,PI2,ZERO
      COMMON/MISC/KIND(500),RGEN,NOUT(5),MOUT,KD(5),iff
      COMMON/WAVES/EW,EOUT(5,32768)
      COMMON/TIME/PER
      common/expt1/edge
      COMMON/MATCH/VOLT(5,3),NCROSS(5,3)
      DATA PI2/6.28318530717958/, PER/0.0/, RGEN/0.0/, VOFF/0.0/
      DATA W80/'[?31'/, W132/'[?3h'/
      ESC=CHAR(27)
      BELL=CHAR(7)
      ZERO=DCMPLX(0.0,0.0)
      iff=0              !Fourier transform speed up by limiting NF
      TYPE *,'            Fiber Optic CAble Simulator FOCAS'
      TYPE *,'                 Revised  11 MARCH 1987'
      TYPE *

      CALL INPUT(AFILE,BFILE,DFILE,FGLE,FDLE,PGLE,PDLE,NC,TDIS,TOFF,DC,
     1 CRLE,FS,VOFF,NEYE,ID,nff,npp)

      ik=kind(1)
      if((ik.eq.1).or.(ik.eq.8).or.(ik.eq.16))go to 686
      type *,' '
      type *,' ***   A SOURCE IS REQUIRED IN THE MODEL  *** '
      TYPE *,BELL

686   IF(NEYE.EQ.1)TYPE *,'  FREQ. DOMAIN RESPONSE WILL NOT BE
     1  CALCULATED  --  EYE PATTERN REQUESTED'
      IF(NEYE.EQ.1)TYPE *,' '

      IF(NEYE.EQ.1)ID=0

      CALL PRECALC(NC,NF,TDIS,TOFF,NGEN)
      nf1=nf                              ! Used to indicate abnormal end
      CALL TRANSFORM(NC,NF,NGEN)
      type *,' '
      TYPE *,'Number of frequency terms computed =',NF
      anf=nf
      err=per/anf
      ierr=1000*err
      irr=(err/2.)*1000
      if(ik.eq.16)irr=irr+500*rlc(1,7)          !Cannot be more accurate
      if(ik.eq.16)ierr=2*irr                    !than input data
      type *,' '
      type *,' Time co-ord uncertainty    = ',irr,' ps'
      if(neye.eq.1)type *,' Timing jitter uncertainty = ',ierr,' ps'
      type *,' '
      if(nf1.eq.nf)go to 667      ! O.K.
!
```

```
TYPE *,' '
type *,' *** WARNING - reduced accuracy ***'
TYPE *,' '
IF(NF.GE.128)GO TO 667
TYPE *,' NOT ENOUGH TERMS TO USE RESULTS '
TYPE *,BELL
STOP

DO K=1,MOUT
DO I=1,NF              !Find time domain response
RX(I)=REAL(EW(K,I))    !Setup for FFT842
RY(I)=-DIMAG(EW(K,I)) !Setup for FFT842
ENDDO

CALL FFT842(0,NF,RX,RY) !Inverse Fourier Transform

DO I=1,NF        !Time domain response
EOUT(K,I)=RX(I)
ENDDO
ENDDO
CALL CROSS(AFILE,CRLE,NF,TDIS,TOFF,NEYE)

ID=ID-1                  !Get ready for time domain graph
CALL GRAPHICS(AFILE,BFILE,DFILE,NF,TDIS,TOFF,DC,FS,VOFF,NEYE,ID)
ID=ID+1                  !Reset ID to chk for freq domain outputs
IF(ID.EQ.0.0)GO TO 11

if(nff.eq.0.and.npp.eq.0)go to 11  !Done if no power spectra or
                                   !phase angle outputs required

do i=1,5         !Setup offsets to zero for freq. domain outputs
toff(i)=0.
enddo

do k=1,mout      !Setup for power spectrum outputs
do i=1,nf
rx(i)=real(ew(k,i))
ry(i)=dimag(ew(k,i))
eout(k,i)=(rx(i)**2)+(ry(i)**2)
eout(k,i)=sqrt(eout(k,i))
enddo
enddo

FS=amax1(eout(1,1),eout(1,2))  !Rqrd for freq. domain scaling

do k=1,mout      !Find maximum of power spectra for normalization
do i=1,nf        !on .fg graphs
FS=AMAX1(FS,EOUT(K,I))
enddo
enddo

do k=1,mout      !Renormalize power spectrum to its maximum value
do i=1,nf
eout(k,i)=eout(k,i)/fs
enddo
enddo
                                   !
```

```
      do k=1,mout  !Don't displace freq. or phase response outputs
      dc(k)=0.
      enddo
      fs=1.

      ANF=NF
      FDIS=ANF/(6.*PER)          !Max freq to be displayed in spectrum
      if(nf.gt.8192)fdis=(8192./(6.*per))   !Limit max. freq displayed
      if(nff.eq.0)go to 12
      CALL GRAPHICS(AFILE,FGLE,FDLE,NF,fdis,TOFF,DC,FS,0.0,NEYE,ID)

      if(npp.eq.0)go to 11
12    do k=1,mout                !Setup for phase vs. freq graph
      do i=1,nf
      eout(k,i)=atan2(ry(i),rx(i))
      enddo
      enddo
      fs=pi2

      CALL GRAPHICS(AFILE,PGLE,PDLE,NF,fdis,TOFF,DC,fs,pi2/2.,NEYE,ID)

11    TYPE *,BELL
      STOP
      END     .1
```

```
      SUBROUTINE  BPF(N,W)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      Returns the A, B, C, D parameters for a Bandpass                 C
      Filter modelled as a trapeziod in Log P(w) vs w.                 C
                      (TYPE 15 CIRCUIT)                                C
                           2/2/87                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      COMPLEX*16 ABCD(500,2,2),ZERO,FF
      DOUBLE PRECISION PI2,F
      COMMON/NET/RLC(500,20),ABCD,PI2,ZERO
      phi=0.
      wl=rlc(n,1)
      W2=RLC(N,2)
      S1=RLC(N,3)              !Slope in dB/octave
      S2=RLC(N,4)              !Slope in dB/octave
      k=rlc(n,5)               !Gain
      AA=rlc(n,16)
      BB=rlc(n,17)
      a=rlc(n,18)
      b=rlc(n,19)

      IF(W.LE.A.OR.W.GE.B)F=0.0
      IF(AA.LE.W.AND.BB.GE.W)F=10.0

      IF(A.LT.W.AND.AA.GT.W)go to 20
      IF(BB.LT.W.AND.B.GT.W)go to 30
      go to 10
      f=(w/wl)-1
      f=sl*f/10.
      f=10**f
      f=5*f
      go to 10

      f=(w/bb)-1
      f=s2*f/10.
      f=1-f
      f=10**f

      IF(F.LT.1.E-16)F=1.E-16
      f=f/10.                            !Normalize F btwn 0 and 1
      f=k*f                   !Multiply by gain

      if(w.lt.wl)phi=rlc(n,6)
      if(w.gt.w2)phi=-rlc(n,6)
      if((w.ge.wl).and.(w.le.w2))go to 40
      go to 50
      phi=w-rlc(n,20)         !w-midpt btwn wl and w2
      phi=rlc(n,14)*phi
      FF=F*DCMPLX(COS(PHI),SIN(PHI))
      FF=1./FF
```

```
      ABCD(N,1,1)=FF
      ABCD(N,1,2)=ZERO
      ABCD(N,2,1)=ZERO
      ABCD(N,2,2)=ZERO

      RETURN
      END



      FUNCTION BRACKET(T,X)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
C  KEEPS THE TRUNCATED VALUE OF T BETWEEN 0 AND X                     C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      IF(T.GT.0.0)GO TO 1
      BRACKET=T+X
      RETURN
1     IF(JINT(T).GT.X)GO TO 2
      BRACKET=T
      RETURN
2     BRACKET=T-X
      RETURN
      END

!
```

```
!  CONTROL

      SUBROUTINE CONTROL(M,NC,ITEM,TDIS,TOFF,DC,FS,VOFF,IERR,NEND,NEYE,
     1   nff,npp)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
C          CONTROL STATEMENT PROCESSING                               C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      CHARACTER*1 ITEM(130),SPA,TAB
      CHARACTER*2 NDATA(6,10),ND
      CHARACTER*3 NAME(10),NOW
      DIMENSION TOFF(4),DC(4)
      COMMON/MISC/KIND(500),RGEN,NOUT(5),MOUT,KD(5),iff
      COMMON/MATCH/VOLT(5,3),NCROSS(5,3)

      DATA NAME/
     1   'OUT','END','EYE','TIT','FDG','PDG','TIT','TIT','TIT','TIT'/

      DATA NDATA/
UT      1    2    3    4    5    6
     1 'V1','V2','V3','TO','DC','DV',
ND      1    2    3    4    5
     2 'TX','DC','FS','FS','FS','FS',
'E      1    2    3    4    5    6
     3 'TX','DC','FS','FS','FS','FS',
T       1    2    3    4    5    6
     4 '  ','  ','  ','  ','  ','  ',
G       1    2    3    4    5    6
     5 'FF','  ','  ','  ','  ','  ',
G       1    2    3    4    5    6
     6 '  ','  ','  ','  ','  ','  ',
ASS     1    2    3    4    5    6
     7 '  ','  ','  ','  ','  ','  ',
ASS     1    2    3    4    5    6
     8 '  ','  ','  ','  ','  ','  ',
ASS     1    2    3    4    5    6
     9 '  ','  ','  ','  ','  ','  ',
ASS     1    2    3    4    5    6
     1 '  ','  ','  ','  ','  ','  '/

      DATA NCROSS/15*0/, NCONT/6/, SPA/' '/, TAB/'    '/
      NOW(1:1)=ITEM(M+1)
      NOW(2:2)=ITEM(M+2)
      NOW(3:3)=ITEM(M+3)
      M=M+4
      DO NJ=1,NCONT              !NCONT= number of types of control statements
      IF(NOW.EQ.NAME(NJ))GO TO (100,200,200,400,500,600),NJ
      ENDDO
      TYPE *,'****** IMPROPER CONTROL CHARACTERS ******'
      IERR=1
      RETURN
      IF(MOUT.LT.5)GO TO 110           !.OUT
      TYPE *,'****** TOO MANY .OUTs SPECIFIED - LAST ONE IGNORED ******'
      RETURN                                    !
```

```
                                                    !  CONTROL


110     MOUT=MOUT+1
        NOUT(MOUT)=NC
        GO TO 205                       !END or EYE statement
200     NEND=1
205     DO K=M,130
        IF(ITEM(K).EQ.'!')RETURN
        IF(ITEM(K).EQ.SPA)GO TO 210
        ENDDO
210     M=K

        DO 220 J=1,6
        DO 230 K=M,130
        IF(ITEM(K).EQ.'!')RETURN
        IF(ITEM(K).EQ.SPA)GO TO 230
        ND(1:1)=ITEM(K)
        ND(2:2)=ITEM(K+1)
        DO NVAR=1,6
        IF(ND.EQ.NDATA(NVAR,NJ))GO TO 240
        ENDDO
230     CONTINUE
        IF(K.GE.130)RETURN
        TYPE *,'****** UNDEFINED VARIABLE NAME ******'
        IERR=1
        RETURN

240     M=K+1
        IF(NJ.EQ.1.AND.NVAR.EQ.6)GO TO 260
        DO K=M,130
        IF(ITEM(K).EQ.SPA.AND.ITEM(K+1).NE.SPA)GO TO 250
        ENDDO
        TYPE *,'****** INCORRECT DATA FORMAT ******'
        IERR=1
        RETURN

250     M=K+1                           !Convert the numeric value
        GO TO(260,245,245,400),NJ
245     GO TO(251,252,253),NVAR                 !END or EYE statement
251     TDIS=DIGIT(M,ITEM,IERR)                 !TDIS = display time
        GO TO 255
252     IF(NVAR.EQ.2)VOFF=DIGIT(M,ITEM,IERR)    !VOFF = Vertical offset
        GO TO 255
253     FS=DIGIT(M,ITEM,IERR)                   !FS = Vertical full scale
255     IF(NJ.NE.3)GO TO 290
        NEYE=1
        IF(MOUT.EQ.1)GO TO 290
        TYPE *,'****** ONLY ONE OUTPUT ALLOWED FOR EYE ******'
        IERR=1
        RETURN
                                                    !
```

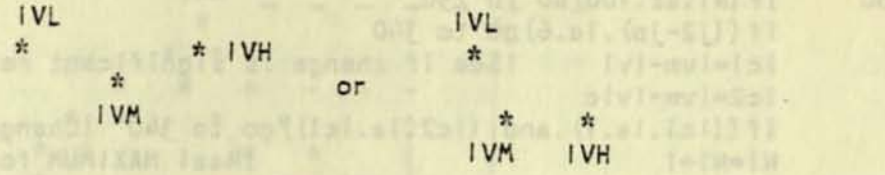
```
                                                    !  CONTROL

 GO TO(280,280,280,270,275,300),NVAR      !OUT statement
 TOFF(MOUT)=DIGIT(M,ITEM,IERR)
 GO TO 290
 DC(MOUT)=DIGIT(M,ITEM,IERR)
 GO TO 290
 NCROSS(MOUT,NVAR)=1                !V1,V2,V3
 VOLT(MOUT,NVAR)=DIGIT(M,ITEM,IERR)
 IF(IERR.LT.2)GO TO 220             !Bad input
 TYPE *,'****** VARIABLE VALUE ERROR ******'
 RETURN
 KD(MOUT)=1
 CONTINUE
 RETURN             !Title Line
 nff=1              !Frequency domain outputs

 DO K=M,130         !Read rest of line
 IF(ITEM(K).EQ.'!')RETURN
 IF(ITEM(K).EQ.SPA)GO TO 211
 ENDDO
 M=K

 DO 221 J=1,6
 DO 231 K=M,130
 IF(ITEM(K).EQ.'!')RETURN
 IF(ITEM(K).EQ.SPA)GO TO 231
 ND(1:1)=ITEM(K)
 ND(2:2)=ITEM(K+1)
 DO NVAR=1,6
 IF(ND.EQ.NDATA(NVAR,NJ))GO TO 241
 ENDDO
 CONTINUE
 IF(K.GE.130)RETURN
 TYPE *,'****** UNDEFINED VARIABLE NAME ******'
 IERR=1
 RETURN

 M=K+1
 IF(NJ.EQ.5.AND.NVAR.EQ.1)IFF=1
 DO K=M,130
 IF(ITEM(K).EQ.SPA.AND.ITEM(K+1).NE.SPA)GO TO 250
 ENDDO
 TYPE *,'****** INCORRECT DATA FORMAT ******'
 IERR=1
 continue
 return
 npp=1              !Phase shift outputs
 return

 END
```

```
      SUBROUTINE CROSS(AFILE,CRLE,NF,TDIS,TOFF,NEYE)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                               C
C     COMPUTES THE CROSSOVER TIMING, MAXIMA & MINIMA.                            C
C     TIMING JITTER, AND PER CENT NOISE MARGIN                                  C
C                                                                               C
C     CROSS computes the crossover & extrema timings; and the                   C
C          extrema values of the waveforms passed in array                      C
C          EOUT(5,32768) through COMMON.  CROSS writes this                     C
C          data into output file CRLE.                                          C
C                                                                               C
C      TMAT   =  array of cross over timings in nanoseconds                     C
C      DERMAX =  array with times of maxima in nanoseconds                      C
C      DERMIN =  array with times of minima in nanoseconds                      C
C      IVX    =  array with maxima values in millivolts                         C
C      IVN    =  array with minima values in millivolts                         C
C      KD     =  vector of pointers to .OUT statements requiring                C
C                extrema calculations                                           C
C      RN     =  vector of percent noise margin for .OUT                        C
C                locations with DV specified, KD(K).ne.0                        C
C      MAXMIN =  vector of pointers to .OUT statement locations                 C
C      XTD = TOFF = vector of time offsets defined by user                      C
C                                                                               C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

      CHARACTER*32 AFILE,CRLE
      COMPLEX*16 EW(5,32768)
      DOUBLE PRECISION PER
      DIMENSION TMAT(5,100),NMAT(5),VMAT(5),TOFF(5),TD(20),
     1 DERMAX(5,100),DERMIN(5,100),MAXMIN(5),XTD(5)
      dimension ivx(5,100),ivn(5,100),rn(5),nn(5),nx(5)
      COMMON/NET/RLC(500,20),ABCD,PI2,ZERO
      COMMON/MISC/KIND(500),RGEN,NOUT(5),MOUT,KD(5)
      COMMON/WAVES/EW,EOUT(5,32768)
      COMMON/TIME/PER
      COMMON/MATCH/VOLT(5,3),NCROSS(5,3)

      DATA KDER/0/,NDER/0/,IN/0/,MAX/0/,INO/0/,MAXO/0/
      DATA IVX/500*0/,IVN/500*0/
C     KDER              Derivative output indicator
C     NDER              Max number of maxima or minima
      ANF=NF            !Number of time increments computed
      DELT=PER/ANF      !Time delay between samples
      DELTT=DELT
      if(kind(1).eq.16)deltt=delt+rlc(1,7)
C     IN                Number of Outputs counter, not more than 10
C     MAX               Crossover counter for all outputs
C                       N is the number of crossovers detected for each
C                            output, not more than 100   !
```

```
DO 50 K=1,MOUT
TO=-TOFF(K)           !User input time delay
DO 100 I=1,3
IF(NCROSS(K,I).EQ.0)GO TO 100  !Check if crossing calc. rqrd
VO=VOLT(K,I)                   !Crossing level voltage
N=0
IN=IN+1
NMAT(IN)=NOUT(K)               !Location of IN-th .OUT
VMAT(IN)=VO
TD(IN)=TOFF(K)
J1=-TO/DELT
IF(J1.LT.1)J1=J1+NF
IVL=1000*(EOUT(K,J1)-VO)       !Round off to millivolts
jlo=j1-1
if(jlo.eq.0)jlo=nf
ivlo=1000*(eout(k,jlo)-vo)
JL=J1+1
JH=J1+NF

DO 150 J=JL,JH                 !Find crossing locations by sign
J2=J                           !change btwn (IVH-IVO) and (IVL-IVO)
IF(J2.GT.NF)J2=J-NF
iVH=1000*(EOUT(K,J2)-VO)

IF(IVL)10,120,20
IF(IVH)121,200,200
IF(IVH)200,200,121
IF(N.EQ.100)GO TO 240
N=N+1
AJ=J2-2
TMAT(IN,N)=AJ*DELT+TO+DELT*IVL/(IVL-IVH)
go to 122
if(ivlo.ne.0)go to 120         !Check if at corner of constant
if(n.eq.100)go to 240          !region equal to VO
n=n+1
aj=j2-3                        !Time index of ivlo
tmat(in,n)=aj*delt+to          !Time co-ord of ivlo
ck to make sure this isn't the previous crossing value instead
a corner that would have been missed otherwise:

if(n.eq.1)go to 122            !First point so it can't be a repeat
if(tmat(in,n).ne.tmat(in,n-1))go to 122   !Not a repeat
tmat(in,n)=0.                  !Reset incorrect crossing value
n=n-1                          !Reset crossing counter

IF(NEYE.EQ.0)GO TO 120
DO M=1,100
IF(TMAT(IN,N).LE.TDIS)GO TO 120
TMAT(IN,N)=TMAT(IN,N)-TDIS
ENDDO

IVLO=IVL
IVL=IVH
CONTINUE
```

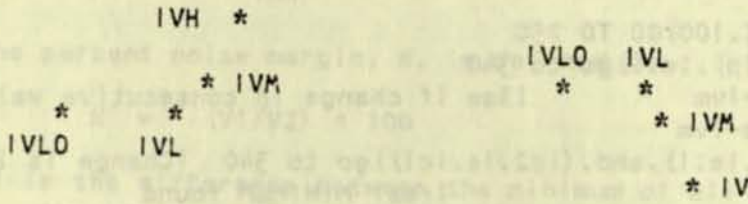
```
240     IF (N.GT.MAX) MAX=N
        IF (IN.EQ.10) GO TO 300
100     CONTINUE
300     IF (KD(K) .EQ.0) GO TO 50       !Extrema calculations setup
            N1=0
            N2=0
            INO=INO+1
            J1=-TO/DELT
            IF (J1.LT.1) J1=J1+NF
            JT=J1+1
            IF (JT.GT.NF) JT=JT-NF
            IVM=EOUT (K,JT) *1000       !Round off slight variations
            IVL=EOUT (K,J1) *1000       !Round off
            jlo=j1-1
            if (jlo.eq.0) jlo=nf
            ivlo=eout (k,jlo) *1000
        jp=0          !Index of previous extremum

            JL=J1+2
            jh=j1+nf

        DO 350 J=JL,JH                      !Find maxima & minima
        J2=J
        IF (J2.GT.NF) J2=J-NF
        IVH=EOUT (K,J2) *1000               !Round off slight variations

C               Extrema are found by sorting through IVL, IVM,
C               and IVH, where IVL corresponds to the voltage
C               value with low index number of the triplet (L,M,H)
C               IVM has the middle index value, and IVH the highest
C               of the three index values.  When IVM is greater
C               than IVL and IVH a maximum is found;  when IVM is
C               less than IVL and IVH a minimum has been found.
C
C               The values of IVL, IVM, and IVH are rounded off to
C               an integral number of millivolts to eliminate slight
C               variations in values due to numerical roundoffs.
C               Similarly, when a local extremum is found, the
C               differences between IVL and IVM, and between IVL and
C               IVLO (the point preceeding IVL) arechecked to make sure
C               that they are significant (i.e. greater than 1 mv).
C
C               Statement 360 corresponds to treatment of a simple
C               maximum of the forms:
C
C                         IVM
C                          *                              IVM IMH
C                   IVL                                    *   *
C                    *                    or
C                                                  IVL *
C                              * IVH
C
!
```

Similarly, statement 370 corresponds to treatment of a simple minimum of the forms:



For cases in which a corner (i.e. change in value after a constant region) is encountered, and additional the low index point from the previous pass through the loop is required to identify the corner. This is saved as IVLO.

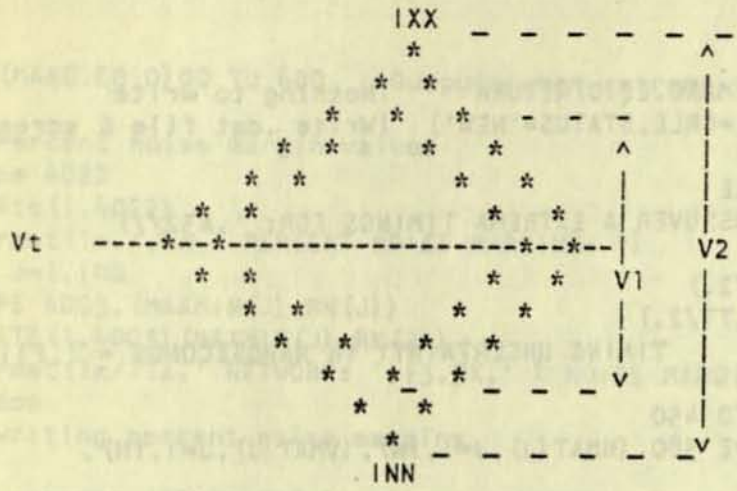
Statements 339 and 341 correspond to corner minima and maxima respectively:




```
IF(IVL-IVM)310,340,320
IF(IVH-IVM)360,360,339
IF(IVH-IVM)341,370,370

if(ivlo.ne.ivl)go to 340   !Check to make sure have a corner min.
if((j2-jp).le.5)go to 340  !SLow changes or close extrema rejected
if(n2.ge.100)go to 250
n2=n2+1
nn(ino)=n2                 !Count number of min for this .OUT
aj=j2-2
dermin(ino,n2)=aj*delt+to        !Min. at IVL
maxmin(ino)=nout(k)
ivn(ino,n2)=ivl            !Minimum in millivolts

jp=j2-2
go to 340

if(ivlo.ne.ivl)go to 340  !Check for corner maximum
if((j2-jp).le.5)go to 340 !Reject slow changes<->close extrema
if(n1.ge.100)go to 250
n1=n1+1
nx(ino)=n1                !Count number of max for this .OUT
aj=j2-2
dermax(ino,n1)=aj*delt+to        !Corner max. at IVL
maxmin(ino)=nout(k)
ivx(ino,n1)=ivl           !Maximum in millivolts

jp=j2-2
go to 340
```

```
                                                        !  CROSS

360     IF (N1.GE.100) GO TO 250
        if ((j2-jp).le.6) go to 340
        ic1=ivm-ivl      !See if change is significant relative
        ic2=ivm-ivlo
        if ((ic1.le.1).and.(ic2.le.ic1)) go to 340  !Change insignificant
                                                    !Real MAXIMUM found
        N1=N1+1
        nx(ino)=n1                  !Count number of maxima for this .OUT
        AJ=J2-2
        DERMAX(INO,N1)=AJ*DELT+TO+DELT*(IVL-IVH)/(IVH+IVL-2*IVM)/2.
        MAXMIN(INO)=NOUT(K)
        ivx(ino,n1)=IVM             !Maximum in millivolts
        XTD(INO)=TOFF(K)
        jp=j2-1
        GO TO 340

370     IF (N2.GE.100) GO TO 250
        if ((j2-jp).le.6) go to 340
        ic1=ivl-ivm           !See if change in consecutive values signific
        ic2=ivlo-ivm
        if ((ic1.le.1).and.(ic2.le.ic1)) go to 340  !Change is insignificant
        N2=N2+1                      !real MINIMUM found
        nn(ino)=n2                  !Count number minima for this .OUT
        AJ=J2-2
        DERMIN(INO,N2)=AJ*DELT+TO+DELT*(IVL-IVH)/(IVH+IVL-2*IVM)/2.
        MAXMIN(INO)=NOUT(K)
        ivn(ino,n2)=ivm             !Minimum in millivolts

        jp=j2-1

340     ivlo=ivl
        IVL=IVM
        IVM=IVH
350     CONTINUE

250     IF (N.GT.MAX) MAX=N
        IF (N1.GT.MAXO) MAXO=N1
        IF (N2.GT.MAXO) MAXO=N2
        IF (INO.GE.5) GO TO 270
C!
```

Noise margin calculations, based on EYE pattern:



The percent noise margin, N, is defined as:

N = (V1/V2) * 100

V1 is the difference between the minimum of all the maxima, INX, and the maximum of all the minima, IXN: V1 = INX - IXN.
V2 is the difference between the maximum of all the maxima, IXX, and the minimum of all the minima, INN: V2 = IXX - INN.

Although it is easier to interpret N with an EYE pattern, the calculation can be run wherever there has been an extremum calculation.

```
  do j=1,ino       !Find noise margin wherever DV specified
     ixx=ivx(j,1) !Initialize ixx, ixx, ixn, inx
     inn=ivn(j,1)
     ixn=ivn(j,1)
     inx=ivx(j,1)
  do i=2,nx(j)     !Find max & min of maxima values
     ixx=jmax0(ixx,ivx(j,i))
     inx=jmin0(inx,ivx(j,i))
     enddo
  do i=2,nn(j)     !Find max & min of minima values
     ixn=jmax0(ixn,ivn(j,i))
     inn=jmin0(inn,ivn(j,i))
     enddo
nd V1 and V2 for the .OUT statement corresponding
 this value of j
     v1=inx-ixn            !V1
     v2=ixx-inn            !V2
     rn(j)=100.*(v1/v2)    !Per cent noise margin
     enddo
nd of noise margin calculations
```

```
                                                    !  CROSS
50      CONTINUE

270     IF(MAX.EQ.0.AND.MAX0.EQ.0)RETURN        !Nothing to write
        OPEN(UNIT=1,FILE=CRLE,STATUS='NEW')     !Write .dat file & screen
        TYPE 880,AFILE
        WRITE(1,880)AFILE
880     FORMAT(1X/'  CROSSOVER & EXTREMA TIMINGS FOR: ',A32//)

        TYPE 890,(DELTT/2.)
        WRITE(1,890)(DELTT/2.)
890     FORMAT(1X,'          TIMING UNCERTAINTY IN NANOSECONDS = ',F11.3/)

        IF(MAX.EQ.0)GO TO 450
        if(neye.ne.1)TYPE 380,(NMAT(J),J=1,IN),(VMAT(J),J=1,IN),
        1 (TD(J),J=1,IN)
380     FORMAT(1X/' CROSSOVER TIMING IN NANOSECONDS. '//
        1 ' NETWORK',<IN>(7X,I3,2X)/
        2 ' VOLTAGE ',<IN>(F11.3,1X)/
        3 ' DELAY   ',<IN>(F11.3,1X)/)
        WRITE(1,380)(NMAT(J),J=1,IN),(VMAT(J),J=1,IN),
        1 (TD(J),J=1,IN)
        DO I=1,MAX
        if(neye.ne.1)TYPE 400,(TMAT(J,I),J=1,IN)
        WRITE(1,400)(TMAT(J,I),J=1,IN)
400     FORMAT(9X,<IN>(F11.3,1X))
        ENDDO

C   Timing jitter calculation - executed only when EYE pattern requested

        IF(NEYE.NE.1)GO TO 450
        NM=JMOD(MAX,2)
        IC1=MAX/2
        AX1=TMAT(1,2)
        AN1=AX1
        AX2=TMAT(1,1)
        AN2=AX2
        DO I=1,IC1
        IC2=1+2*I
        AX1=AMAX1(TMAT(1,2*I),AX1)
        AN1=AMIN1(TMAT(1,2*I),AN1)
        IF((NM.EQ.0).AND.(I.EQ.IC1))GO TO 4002
        AX2=AMAX1(TMAT(1,IC2),AX2)
        AN2=AMIN1(TMAT(1,IC2),AN2)
4002    ENDDO
        TJ1=AX1-AN1
        TJ2=AX2-AN2
        TYPE 4001,(TJ1,TJ2)
        WRITE (1,4001)(TJ1,TJ2)
4001    FORMAT(1X,/,9X,'TIMING JITTER = ',F8.4,4X,F8.4,' NS')
!
```

```
                                                    !  CROSS


50      IF(MAXO.EQ.0)GO TO 600    !Outputs when extrema found

   Write Percent noise margin values
        type 4022
        write(1,4022)
022     format(1x,/,6x,' PERCENT NOISE MARGINS: ')
        DO J=1,INO
        TYPE 4003,(MAXMIN(J),RN(J))
        WRITE(1,4003) (MAXMIN(J),RN(J))
003     format(1x/11x,' NETWORK: ',I3,5X,' % NOISE MARGIN = ',F9.4)
        enddo
    Done writing percent noise margins

    Write extrema values and locations

        IN02=2*INO
        if(neye.ne.1)TYPE 500,(MAXMIN(J),J=1,INO),(XTD(J),J=1,INO)
00      FORMAT(1X/' MAXIMA AND MINIMA TIMING IN NANOSECONDS'//
        1 ' NETWORK        ',<INO>(7X,I3,14X)/
        2 ' DELAY          ',<INO>(F11.3,13X)/
        3 '           ',<INO>('      MAXIMA       MINIMA ')/)
        WRITE(1,500) (MAXMIN(J),J=1,INO),(XTD(J),J=1,INO)
        DO I=1,MAXO
        if(neye.ne.1)TYPE 550,(DERMAX(J,I),DERMIN(J,I),J=1,INO)
        WRITE(1,550) (DERMAX(J,I),DERMIN(J,I),J=1,INO)
50      FORMAT(9X,<IN02>(F11.3,1X))
        ENDDO

        if(neye.ne.1)type 700
        WRITE(1,700)
00      FORMAT(1X/'  EXTREMA VALUES IN MILLIVOLTS  '/)
        do j=1,ino
        if(neye.ne.1)TYPE 750,(MAXMIN(J))
        WRITE(1,750) (MAXMIN(J))
50      FORMAT(1X/'  NETWORK: ',I3)
        do i=1,maxo
        if(neye.ne.1)TYPE 760,I,IVX(J,I),IVN(J,I)
        WRITE(1,760) I,IVX(J,I),IVN(J,I)
0       FORMAT(9X,I4,' MAX = ',I6,5X,'MIN = ',I6)
        enddo
        enddo
        type *,' '

0       CLOSE(UNIT=1)
        RETURN
        END
```

! DELTA

```
      SUBROUTINE DELTA(N,W)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C     SOLVES FOR THE A,B,C,D PARAMETERS FOR THE PI CIRCUIT
C     FOR THE Nth NETWORK ELEMENT AT FREQUENCY W (TYPE 3 CIRCUIT)
C                       12/11/85
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      COMPLEX*16 ABCD(500,2,2),ZERO
      DOUBLE PRECISION PI2
      COMMON/NET/RLC(500,20),ABCD,PI2,ZERO

      G1=RLC(N,1)                 ! G1
      G2=RLC(N,2)                 ! G2
      BC1=W*RLC(N,3)              ! w*C1
      BC2=W*RLC(N,4)              ! w*C2
      XL=W*RLC(N,5)               ! w*L
      R=RLC(N,6)
      IF(RLC(N,9).EQ.0.0)THEN            ! RP default to infinity
      ABCD(N,1,1)=DCMPLX( (1.+G2*R-BC2*XL),(XL*G2+R*BC2) )      !A
      ABCD(N,1,2)=DCMPLX(R,XL)                                   !B
      ABCD(N,2,1)=DCMPLX((G1+G2+R*(G1*G2-BC1*BC2)-XL*(G1*BC2+G2*BC1)),
     1 (BC1+BC2+R*(G1*BC2+G2*BC1)+XL*(G1*G2-BC1*BC2)) )          !C
      ABCD(N,2,2)=DCMPLX( (1.+G1*R-BC1*XL) ,(XL*G1+R*BC1) )      !D
      ELSE
      RP=RLC(N,9)                        ! Finite RP case
      D=(R+RP)**2+XL**2
      A=RP*(R*(R+RP)+XL**2)/D
      B=XL*RP**2/D
      ABCD(N,1,1)=DCMPLX( (1.+G2*A-BC2*B),(G2*B+BC2*A) )        !A
      ABCD(N,1,2)=DCMPLX(A,B)                                    !B
      ABCD(N,2,1)=DCMPLX((G1+G2+A*(G1*G2-BC1*BC2)-B*(G1*BC2+G2*BC1)),
     1 (BC1+BC2+A*(G1*BC2+G2*BC1)+B*(G1*G2-BC1*BC2)) )           !C
      ABCD(N,2,2)=DCMPLX( (1.+G1*A-BC1*B) ,(G1*B+BC1*A) )        !D
      ENDIF
      RETURN
      END
```

!

```
                                                    !  DIGIT
      FUNCTION DIGIT(M,ITEM,IERR)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
                                                                      C
 CONVERTS A CHARACTER STRING TO A NUMBER                              C
                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      CHARACTER*1 ITEM(130),INTEG(15),N
      DIMENSION TEMP(130)
      DATA INTEG/'1','2','3','4','5','6','7','8','9','0','.',' ',
     1 'K','M','U'/
      DIGIT=0.
      K=0                                  !Decimal Point Location
      PLUS=1.0                             !Sign and Exponent
      IF(ITEM(M).NE.'-')GO TO 10
      PLUS=-1.0
      M=M+1
      IF(ITEM(M).EQ.'+')M=M+1              !Ignore +

      DO 50 I=M,130
      N=ITEM(I)
      DO J=1,15                            !Match Character
      IF(N.EQ.INTEG(J))GO TO 30
      ENDDO
      IERR=1                               !Error, no match
      RETURN
      TEMP(I)=0.
      IF(J.EQ.11.AND.K.NE.0)GO TO 20
      IF(J.EQ.11)K=I          !A period
      IF(J.EQ.12)GO TO 60     !A blank
      IF(J.LT.10)TEMP(I)=J
      IF(J.EQ.13)PLUS=PLUS*1.E3            !K, M, and U exponents
      IF(J.EQ.14)PLUS=PLUS*1.E-3
      IF(J.EQ.15)PLUS=PLUS*1.E-6
      CONTINUE

      IF(K.EQ.0)K=I
      IF(ABS(PLUS).EQ.1.0)THEN
        IF(K.EQ.0)K=I
        L=I-1
      ELSE
        IF(K.EQ.0)K=I-1
        L=I-2
      ENDIF

      DO 70 J=M,L
      IF(J.EQ.K)GO TO 70
      A=10.**(K-J)
      IF(J.LT.K)A=0.1*A
      DIGIT=DIGIT+TEMP(J)*A
      CONTINUE
      M=I+1
      DIGIT=PLUS*DIGIT
      RETURN
      END           !
```

```
                                                          !    EAMP
                                                          !    FIBER



      SUBROUTINE EAMP(N,W)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C     RETURNS THE A, B, C, D PARAMETERS FOR A                          C
C     A VOLTAGE DEPENDENT VOLTAGE SOURCE WITH EXTERNAL RESISTANCE      C
C     AND A DEFINED BANDWIDTH (TYPE 6 CIRCUIT)                         C
C                  1/20/86                                             C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      COMPLEX*16 ABCD(500,2,2),ZERO,D
      DOUBLE PRECISION PI2
      COMMON/NET/RLC(500,20),ABCD,PI2,ZERO

      WO=RLC(N,16)
      D=DCMPLX(1.0,W/WO)/RLC(N,1)
      ABCD(N,1,1)=D
      ABCD(N,1,2)=RLC(N,2)*D
      ABCD(N,2,1)=1.D-9*D/RLC(N,2)
      ABCD(N,2,2)=1.D-9*D

      RETURN
      END



      SUBROUTINE FIBER(N,W)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C     OPTICAL FIBER MODEL - Returns A, B, C, D parameters              C
C     (TYPE 7 CIRCUIT)                                                 C
C     2/14/86                                                          C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      COMPLEX*16 ABCD(500,2,2),ZERO,P
      DOUBLE PRECISION PI2,AO,AW,WTC,WTM,H,PHI
      COMMON/NET/RLC(500,20),ABCD,PI2,ZERO

      AO=RLC(N,16)               !Static Attenuation
      WTC=W*RLC(N,17)            !Chromatic dispersion term
      AW=RLC(N,18)               !Chromatic Loss term
      WTM=W*RLC(N,19)            !Intermodal bandwidth term
      H=AO+WTC**2+WTM**2
      IF(H.GT.20.)H=20.
      P=DEXP(H)
      IF(AW.EQ.0.0)GO TO 100     !Simplify if no wavelength dependent loss
        PHI=2.*AW*WTC
        P=P*DCMPLX(DCOS(PHI),DSIN(PHI))
100   ABCD(N,1,1)=P
      ABCD(N,1,2)=ZERO
      ABCD(N,2,1)=ZERO
      ABCD(N,2,2)=P

      RETURN
      END

!
```

! FILTER

```
    SUBROUTINE  FILTER(N,W)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
    DELAY LINE FILTER MODEL                                              C
    (TYPE 12 CIRCUIT)                                                    C
    4/11/86                                                              C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
    COMPLEX*16 ABCD(500,2,2),ZERO,Q
    DOUBLE PRECISION PI2,P,X
    COMMON/NET/RLC(500,20),ABCD,PI2,ZERO

    X=W*RLC(N,1)
    M=RLC(N,2)
    P=(DCOS(X))**M
    IF(DABS(P).LT.1.D-20)P=1.D-20
    Q=1./P
    ABCD(N,1,1)=Q
    ABCD(N,1,2)=Q*RLC(N,3)
    ABCD(N,2,1)=Q*RLC(N,4)
    ABCD(N,2,2)=ZERO

    RETURN
    END
```

```
      FUNCTION GEN(N,W)
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
C      RETURNS THE VOLTAGE GENERATOR FREQUENCY COMPONENT
C      FOR THE Nth NETWORK ELEMENT AT FREQUENCY W (TYPE 1 CIRCUIT)
C                     12/11/85
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      COMPLEX*16 ABCD(500,2,2),ZERO,DNC,GEN,DF
      DOUBLE PRECISION DN,DO,AP,AR,AF,PHI,PER,PI2,BETA,C
      COMMON/NET/RLC(500,20),ABCD,PI2,ZERO
      COMMON/TIME/PER

      PW=RLC(N,1)                                    !Pulse width
      AMP=RLC(N,4)                                   !Amplitude
      WO=RLC(N,16)                                   !Gen Bandwidth rad/s
      TRM=RLC(N,20)                                  !0% to 100% risetime
      AR=0.5*TRM*W
      AP=0.5*W*PW
      C=2.*AMP
      DN=C*PW/PER
      IF(W.EQ.0.0) DN=0.5*DN+RLC(N,7)               !DC Component
      IF(AP.GT.1.E-5) DN=DN*DSIN(AP)/AP
      IF(AR.GT.1.E-5) DN=DN*DSIN(AR)/AR

C  GENERATOR DELAY
      PHI=-W*RLC(N,5)
      DNC=DN*DCMPLX(DCOS(PHI),DSIN(PHI))            !Delay

C ASSYMETRICAL TRAPEZOID
      TFM=RLC(N,19)
      AF=0.5*TFM*W
      IF(TFM.EQ.TRM.OR.AF.EQ.0.0.OR.AR.EQ.0.0)GO TO 100
      BETA=PHI-AP+PI2/4.0
      DO=AF
      IF(AR.NE.0.0) DO=DO*DSIN(AR)/AR
      DF=C*(DSIN(AF)-DO)*DCMPLX(DCOS(BETA),DSIN(BETA))/(AF*W*PER)
      DNC=DNC+DF

C  GENERATOR LOW PASS FILTER
100   GEN=DNC/DCMPLX(1.0,(W/WO))
      RETURN
      END
```

```
                                                    !  GRAPHICS


      SUBROUTINE GRAPHICS(AFILE,BFILE,DFILE,NF,TDIS,TOFF,DC,FS,VOFF,
     1 NEYE,ID)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C     WRITES TO THE OUTPUT FILES 12/29/86                              C
C                                                                      C
C     AFILE  =  name of simulation model file; used in labelling       C
C     BFILE  =  name of graphics output file                           C
C     DFILE  =  name of data output file                               C
C     NF     =  number of terms to be graphed                          C
C     TDIS   =  horizontal axis scale                                  C
C     TOFF   =  horizontal axis offset for time domain                 C
C     DC     =  vector of vertical offsets for various outputs,        C
C               maximum of 5 allowed                                   C
C     FS     =  vertical axis scale                                    C
C     VOFF   =  vertical axis offset                                   C
C     NEYE   =  1 for EYE pattern; otherwise 0                         C
C     ID     =  1 for frequency domain output; otherwise 0             C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      CHARACTER*32 AFILE,BFILE,DFILE
      COMPLEX*16 EW(5,32768)
      DOUBLE PRECISION PER,P2
      INTEGER ESC      ! This is the graphics escape character
      DIMENSION TOFF(5),DC(5),ORD(11),OUTPUT(350,11)
      COMMON/MISC/KIND(500),RGEN,NOUT(5),MOUT,KD(5),iff
      COMMON/TIME/PER
      COMMON/WAVES/EW,EOUT(5,32768)
      DATA ESC/027/    ! This is its octal equivalent
      P2=6.28318530717985
      FWO=1./PER
      AFS=FS
      AVOFF=VOFF                    !Vertical offset
      IF(FS.EQ.0.0)AFS=2.0          !Vertical scale factor for time
      AMP=1./AFS
      AO=450.*AMP*AVOFF+20.         !Vertical offset

      TINC=TDIS/350.                !Horizontal display time increment
      TG=35.*TINC                   !Horizontal axis labeling increment
      TGO=TG                        !Horizontal axis initial offset
      TS=TG                         !Waveform time offset

      IF(NEYE.EQ.0)GO TO 10
      TINC=2.*TINC                  !Eye pattern values
      TG=35.*TINC
      TGO=175.*TINC
      TS=87.5*TINC
      ANF=NF                        !Number of time samples
                                                    !
```

```
        DELT=PER/ANF                    !Time delay per waveform sample
        IF(ID.EQ.1)DELT=FWO             !Freq delay per waveform output
        Y=TINC/DELT                     !Ratio of display sample to waveform sam
        TO=1.-TS/DELT-Y                 !Display offset in number of waveform sam
        DO I=1,100
        IF(ABS(TO).LT.ANF)GO TO 20
        TO=TO+ANF
        ENDDO

20      OPEN(UNIT=21,FILE=BFILE,STATUS='NEW',CARRIAGECONTROL='LIST')
        WRITE(21,90)ESC                 !CLEAR SCREEN
90      FORMAT(1X,A1,'[2J')
        WRITE(21,100)ESC                !ENTER GRAPHICS
100     FORMAT(1X,A1,'Pps(a[0,479][767,0])')

C       ************************************************************************
C       *****DRAW GRID AND TICK MARKS*****************************************
C       ************************************************************************

        WRITE(21,110)  !MAKE GRID LIGHTER THAN THE WAVEFORM PLOT
110     FORMAT(1X,'w(i2)')
        J=-20
        K=-25
        DO 130 I=1,11  !DRAW GRID LINES
         J=J+70
         K=K+45
         WRITE(21,120)J,J,K,K
120     FORMAT(1X,'p['13',20]v['13',470]p[50,'13']v[750,'13']')
130     CONTINUE
        J=50
        K=20
        DO 140 I=1,50  !DRAW TICK MARKS
         J=J+14
         K=K+9
         WRITE(21,150)J,J,K,K
150     FORMAT(1X,'p['13',30]v['13',20]p[50,'13']v[60,'13']')
140     CONTINUE

C       ************************************************************************
C       *****PLOT WAVEFORMS HERE**********************************************
C       ************************************************************************
C
        WRITE(21,190)
190     FORMAT(1X,'w(i3)')              !Return to normal intensity

        DO 300 K=1,MOUT
        EOFF=DC(K)
        TC=0
        T=TO+TOFF(K)/DELT               !EYE pattern sample counter
310     NP=0                            !Total time offset in number of wavefor
                                        !Loop to here for EYE pattern
        WRITE(21,210)
210     FORMAT(1X,'p[48,0]')
        TOUT=-TINC-TG
```

```
                                                    !  GRAPHICS

      DO 400 I=1,350
      IF(K.EQ.1.)OUTPUT(I,1)=I*TINC+TOUT
      T=Y+T                          !Increment sample number
      T=BRACKET(T,ANF)
      NT=JINT(T)                              !Integer
      DT=T-NT
      NT1=NT+1
      IF(NT1.GT.NF)NT1=NT1-NF !Recycle if NT2 out of range (too large)
      TC=TC+Y                                 !Increment EYE pattern sample counter
      EO=(1.-DT)*EOUT(K,NT)+DT*EOUT(K,NT1)    !Interpolate
      OUTPUT(I,K+1)=EO+EOFF
      OUT=AMP*(EO+EOFF)*450.+AO
      IF(I.GT.1)GO TO 410
      WRITE(21,200)OUT              !First point is a point, not a line
0     FORMAT(1X,'p[+2,'G']V[]')
      GO TO 400
0     IF(OUT.GE.0.0)WRITE(21,420)OUT
0     FORMAT(1X,'V[+2,'G']')
      IF(OUT.LT.0.0)WRITE(21,430)
0     FORMAT(1X,'P[+2,0]')     !Write a blank for negative display locations
0     CONTINUE
      WRITE(21,500)
0     FORMAT(1X,'[] (e)')
      IF(NEYE.EQ.0)GO TO 300
      IF(TC.LT.ANF)GO TO 310  !Finish for EYE pattern.
0     CONTINUE

*****************************************************************************
*****LABEL AXES HERE*********************************************************
*****************************************************************************

      DO I=1,11
      ORD(I)=I-1
      ORD(I)=0.1*AFS*ORD(I)-VOFF
      ENDDO
      WRITE(21,621)ORD
      FORMAT(1X,'p[1,28]t(s1,i0)"'F7.2'"'/
     1 1X,'p[1,73]t(s1,i0)"'F7.2'"'/
     2 1X,'p[1,118]t(s1,i0)"'F7.2'"'/
     3 1X,'p[1,163]t(s1,i0)"'F7.2'"'/
     4 1X,'p[1,208]t(s1,i0)"'F7.2'"'/
     5 1X,'p[1,253]t(s1,i0)"'F7.2'"'/
     6 1X,'p[1,298]t(s1,i0)"'F7.2'"'/
     7 1X,'p[1,343]t(s1,i0)"'F7.2'"'/
     8 1X,'p[1,388]t(s1,i0)"'F7.2'"'/
     9 1X,'p[1,433]t(s1,i0)"'F7.2'"'/
     1 1X,'p[1,478]t(s1,i0)"'F7.2'"')
      LABEL ABSCISSA AXIS                    !
```

```
!  GRAPHICS


      DO I=1,6
       MM=25+137*(I-1)
       IF(I.EQ.6)MM=714
       TG2=2.*TG*(I-1)-TG0
      if(id.eq.1)tg2=1000.*tg2          !Convert freq to MHz
       WRITE(21,700)MM,TG2
700    FORMAT(1X,'p['13',15]t(s1,i0)"'F7.2'"')
      ENDDO
       IF(ID.NE.1)WRITE(21,710)AFILE,ESC                    ! PLOT TITLE
710    FORMAT(1X,'p[230,479]t(s0,i0)"AMPLITUDE VS TIME IN NS.",
      1 " INPUT FILE:","'A32'"','s(a[0,0][767,479])',A1,'\')   !EXIT GRA

      IF(ID.EQ.1)WRITE(21,711)AFILE,ESC
711   FORMAT(1X,'p[230,479]t(s0,i0)"AMPLITUDE VS FREQ IN MHZ.",
      1 " INPUT FILE:","'A32'"','s(a[0,0][767,479])',A1,'\')   !EXIT GRA


      CLOSE(UNIT=21)
!     IF(ID.EQ.0)RETURN          !if ID=1, Create an output file DFILE
      IN=MOUT+1                  !with the graphed data
      OPEN(UNIT=21,FILE=DFILE,STATUS='NEW')
      DO I=1,350
      WRITE(21,1000)(OUTPUT(I,J),J=1,IN)
1000  FORMAT(1X,F20.3,<MOUT>(1X,F20.3))
      ENDDO
      CLOSE(UNIT=21)
      RETURN
      END

!
```

```
                                                    !   GSIN

      FUNCTION GSIN(N,W)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      RETURNS THE SINEWAVE GENERATOR FREQUENCY COMPONENT                   C
      FOR THE Nth NETWORK ELEMENT AT FREQUENCY W (TYPE 8 CIRCUIT)          C
                        12/16/86                                           C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      COMPLEX*16 ABCD(500,2,2),ZERO,GSIN
      DOUBLE PRECISION DN,PHI,PI2
      COMMON/NET/RLC(500,20),ABCD,PI2,ZERO
      COMMON/TIME/PER

      WO=RLC(N,16)                  !Sinewave frequency, 2*pi*F in GRad/s
      DN=0.0
      IF(W.EQ.0.0) DN=RLC(N,3)              !DC Component
      IF(W.EQ.WO) DN=RLC(N,2)               !Amplitude

 GENERATOR DELAY
      PHI=-W*RLC(N,4)                                   !Delay
      GSIN=DN*DCMPLX(DCOS(PHI),DSIN(PHI))

      RETURN
      END
```

```
!     HPF
!     IAMP
```

```
      SUBROUTINE HPF(N,W)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C     SOLVES FOR THE A,B,C,D PARAMETERS FOR THE HPF CIRCUIT          C
C     FOR THE Nth NETWORK ELEMENT AT FREQUENCY W  (TYPE 10 CIRCUIT)  C
C                        11/3/86 new B coeff                         C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      COMPLEX*16 ABCD(500,2,2),ZERO
      DOUBLE PRECISION PI2,GL,GC,D,AR,AI,B
      COMMON/NET/RLC(500,20),ABCD,PI2,ZERO

      GL=RLC(N,20)       !L/R
      GC=RLC(N,19)       !1/RC
      GLW=W*GL
      D=1./(1.+GLW**2)
      AR=1.-D*GL*GC
      IF(W.GE.1.E-5)THEN
        AI=-D*GC/W
        B=-1./(W*RLC(N,1))
      ELSE
        AI=-D*GC*1.E+5
        B=-1.E+5/RLC(N,1)
      ENDIF
      ABCD(N,1,1)=DCMPLX(AR,AI)                    !A
      ABCD(N,1,2)=B*DCMPLX(0.0,1.0)                !B
      ABCD(N,2,1)=RLC(N,3)/DCMPLX(1.,GLW)          !C
      ABCD(N,2,2)=DCMPLX(1.0,0.0)                  !D
      RETURN
      END


      SUBROUTINE IAMP(N,W)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C     A VOLTAGE DEPENDENT CURRENT SOURCE WITH EXTERNAL RESISTANCE    C
C     AND A DEFINED BANDWIDTH (TYPE 9 CIRCUIT)                       C
C                        1/20/86                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      COMPLEX*16 ABCD(500,2,2),ZERO,D
      DOUBLE PRECISION PI2
      COMMON/NET/RLC(500,20),ABCD,PI2,ZERO

      WO=RLC(N,16)
      D=DCMPLX(1.0,W/WO)/RLC(N,1)
      ABCD(N,1,1)=RLC(N,3)*D
      ABCD(N,1,2)=D
      ABCD(N,2,1)=1.D-6*RLC(N,3)*D
      ABCD(N,2,2)=1.D-6*D

      RETURN
      END

!
```

! INFILTER

```
      SUBROUTINE INFILTER(NVAR,K,ITEM)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
C Special Handling for FILTER Network, places the implied "1" afterC
C the Sine, Cosine, or Gaussian statements                            C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      CHARACTER*1 ITEM(130)

      IF(NVAR.LT.9)RETURN
      DO I=K,130
      IF(ITEM(I).EQ.' ')GO TO 10
      ENDDO
10    ITEM(I-1)='1'
      ITEM(I-2)=' '
      K=I-3

      RETURN
      END
```

! INPUT

```
      SUBROUTINE INPUT(AFIL,BFIL,DFIL,FGL,FDL,PGL,PDL,NC,TDIS,TOFF,DC,
     1 CRL,FS,VOFF,NEYE,ID,nff,npp)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C      DATA ENTRY SUBROUTINE 1/28/87                                   C
C                                                                      C
C      INPUT reads in the simulation model, interprets network         C
C      subsections and writes the subsection circuit parameters        C
C      into array RLC, and calls subroutine CONTROL to interpret       C
C      simulation model control statements.  Input also generates      C
C      the names for the output files, based on the name of the        C
C      simulation model.  INPUT also passes the control pointers:      C
C      NEYE, NFF, and NPP to the main routine.                         C
C                                                                      C
C      AFIL  = file containing simulation model                        C
C      BFIL  = time domain graph output file                           C
C      DFIL  = time domain data output file                            C
C      FGL   = frequency domain graph output file                      C
C      FDL   = frequency domain data output file                       C
C      PGL   = freq. vs. phase graphics output file                    C
C      PDL   = freq. vs phase data output file                         C
C      NC    = number of network sections in simulation model          C
C      TDIS  = horizontal display scale for time domain                C
C      TOFF  = vector of horizontal shifts for time displays           C
C      DC    = vector of vertical shifts for displays                  C
C      CRL   = file containing locations of voltage crossings          C
C      FS    = vertical axis scale for time domain output              C
C      VOFF  = vertical offset for time domain graphs                  C
C      NEYE  = 1 if EYE pattern output requested, else=0               C
C      ID    = 1 if frequency domain outputs requested, else=0         C
C      NFF   = 1 if frequency domain outputs requested                 C
C      NPP   = 1 if phase shift outputs requested                      C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      CHARACTER*1 ITEM(130),SPA,TAB,ICDAT
      CHARACTER*2 NDATA(13,20),ND
      CHARACTER*3 NAME(20),NOW
      CHARACTER*32 AFIL,BFIL,DFIL,FGL,FDL,PGL,PDL,CRL
      COMPLEX*16 ABCD(500,2,2)
      DOUBLE PRECISION PER,PI2
      DIMENSION TOFF(5),DC(5)
      COMMON/NET/RLC(500,20),ABCD,PI2
      COMMON/MISC/KIND(500),RGEN,NOUT(5),MOUT,KD(5),iff

      integer Status, size
      integer Str$trim
      integer time_graph_size, time_data_size
      integer freq_graph_size, freq_data_size
      integer phase_graph_size, phase_data_size, cross_size
!
```

```
                                                        !  INPUT
      DATA NAME/
C          1     2     3     4     5     6     7     8     9     10
     1 'GEN','TEE','DEL','LIN','STU','EAM','FIB','SIN','IAM','HPF',
C          11    12    13    14    15    16    17    18    19    20
     2 'XFO','FIL','SHU','SER','BPF','SRC','   ','   ','   ','   '/
      DATA NDATA/
C GEN      1     2     3     4     5     6     7     8     9     10    11    12    13
     1 'PW','TR','PE','AM','DE','BW','DC','R ','TF','MO','MO','MO','MO',
C TEE      1     2     3     4     5     6     7     8     9     10    11    12    13
     2 'R1','R2','G ','L1','L2','C ','F1','F2','R ','R ','R ','R ','R ',
C DELTA    1     2     3     4     5     6     7     8     9     10    11    12    13
     3 'G1','G2','C1','C2','L ','R ','R1','R2','RP','RP','RP','RP','RP',
C LINE     1     2     3     4     5     6     7     8     9     10    11    12    13
     4 'ZO','TD','X ','R ','TC','D ','K1','K2','WC','Q ','F1','F2','F3',
C STUB     1     2     3     4     5     6     7     8     9     10    11    12    13
     5 'ZO','TD','X ','R ','TC','D ','K1','K2','GL','RL','CL','WC','WC',
C EAMP     1     2     3     4     5     6     7     8     9     10    11    12    13
     6 'GA','R ','BW','BW','BW','BW','BW','BW','BW','BW','BW','BW','BW',
C FIBER    1     2     3     4     5     6     7     8     9     10    11    12    13
     7 'AO','AW','FW','FI','D ','QM','QC','X ','LC','LO','SO','SO','SO',
C SIN      1     2     3     4     5     6     7     8     9     10    11    12    13
     8 'F ','AM','DC','DE','R ','R ','R ','R ','R ','R ','R ','R ','R ',
C IAMP     1     2     3     4     5     6     7     8     9     10    11    12    13
     9 'GA','R ','G ','BW','BW','BW','BW','BW','BW','BW','BW','BW','BW',
C HPF      1     2     3     4     5     6     7     8     9     10    11    12    13
     1 'C ','L ','G ','R ','R ','R ','R ','R ','R ','R ','R ','R ','R ',
C XFORMER  1     2     3     4     5     6     7     8     9     10    11    12    13
     2 'L1','L2','M ','K ','N ','N ','N ','N ','N ','N ','N ','N ','N ',
C FILTER   1     2     3     4     5     6     7     8     9     10    11    12    13
     3 'TO','N ','R2','G1','R1','FO','FO','FO','SI','CO','GA','GA','GA',
C SHUNT    1     2     3     4     5     6     7     8     9     10    11    12    13
     4 'AO','A1','A2','A3','A4','BO','B1','B2','B3','B4','K ','K ','K ',
C SERIES   1     2     3     4     5     6     7     8     9     10    11    12    13
     5 'AO','A1','A2','A3','A4','BO','B1','B2','B3','B4','K ','K ','K ',
C BPF      1     2     3     4     5     6     7     8     9     10    11    12    13
     6 'F1','F2','S1','S2','K ','PH','PH ','PH','PH','PH','PH','PH','PH',
C SRC      1     2     3     4     5     6     7     8     9     10    11    12    13
     7 'EF','PW','PE','AM','DE','NT','DT','TY','R ','R ','R ','R ','R ',
C UNASS    1     2     3     4     5     6     7     8     9     10    11    12    13
     8 '   ','   ','   ','   ','   ','   ','   ','   ','   ','   ','   ','   ','   ',
C UNASS    1     2     3     4     5     6     7     8     9     10    11    12    13
     9 '   ','   ','   ','   ','   ','   ','   ','   ','   ','   ','   ','   ','   ',
C UNASS    1     2     3     4     5     6     7     8     9     10    11    12    13
     1 '   ','   ','   ','   ','   ','   ','   ','   ','   ','   ','   ','   ','   ',
C UNASS    1     2     3     4     5     6     7     8     9     10    11    12    13
     2 '   ','   ','   ','   ','   ','   ','   ','   ','   ','   ','   ','   ','   '/

      DATA SPA/' '/,TAB/'      '/,IERR/0/,MOUT/0/,KD/5*0/
      NEND=0          !End of the model pointer
      NEYE=0          !EYE pattern pointer
      nff=0           !Frequency domain output pointer
      npp=0           !Phase angle output pointer
      NC=0            !Number of circuit subsections counter
      NKIND=16        !Number different circuit subsection types
      ID=1            !Pointer for freq. domain outputs in GRAPHICS
```

! INPUT

```
      TYPE 20
20    FORMAT(' Enter the input file name: ',$)
      ACCEPT 40,AFIL
40    FORMAT(A32)

C     Code to generate file names for output files
            ! trim out extra spaces
      status = str$trim( AFIL , AFIL , size )
      If (.NOT. Status) Call lib$signal( %val ( Status ))
            ! make filenames by adding file type
      BFIL = AFIL (1:size) // 'tg'  ! filetype
      DFIL = AFIL (1:size) // 'td'
      FDL  = AFIL (1:size) // 'fd'
      FGL = AFIL (1:size) // 'fg'
      PGL = AFIL (1:size) // 'pg'
      PDL = afil (1:size) // 'pd'
      CRL = afil (1:size) // 'dat'

      ! get string size
      status = str$trim( BFIL,
     2                   BFIL, time_graph_size )
      status = str$trim( DFIL,
     2                   DFIL, time_data_size )
      status = str$trim( FGL,
     2                   FGL, freq_graph_size )
      status = str$trim( FDL,
     2                   FDL, freq_data_size )
      status = str$trim( PGL,
     2                   PGL, phase_graph_size)
      status = str$trim( PDL,
     2                   PDL, phase_data_size )
      status = str$trim( CRL,
     2                   CRL, cross_size )

      TYPE *,'*************************************************************'
      OPEN(UNIT=1,FILE=AFIL,STATUS='OLD')

      DO 500 I=1,1000 !Will read 1000 statements in simulation model
      DO J=1,130                          !Initialize input array
      ITEM(J)=SPA
      ENDDO
      READ(1,120) (ITEM(K),K=1,130)       !Read in an input line
120   FORMAT(130A1)
      TYPE 125, (ITEM(J),J=1,80)          !Type it back
125   FORMAT(1X,80A1)
      DO J=1,130
      IT=ICHAR(ITEM(J))
      IF(IT.GE.97.AND.IT.LE.122) IT=IT-32       !Capitalize everything
      ITEM(J)=CHAR(IT)
      IF(ITEM(J).EQ.','.OR.ITEM(J).EQ.'='.OR.ITEM(J).EQ.TAB)
     1 ITEM(J)=SPA                    !Replace , = or tab with space character
      ENDDO
!
```

```
                                                    !  INPUT

      DO 140 M=1,130                          !Look for special first characters
      IF(ITEM(M).EQ.SPA)GO TO 140             !Space
      IF(ITEM(M).EQ.'!')GO TO 500             !Comment
      IF(ITEM(M).EQ.'.')GO TO 350             !Control Line
      IF(ITEM(M).EQ.'+')GO TO 190             !Continuation Line
      GO TO 160                               !Must be a Circuit
140   CONTINUE
      GO TO 500                               !Null line

160   NOW(1:1)=ITEM(M)                        !Place the next three char in NOW
      NOW(2:2)=ITEM(M+1)
      NOW(3:3)=ITEM(M+2)
      DO LT=1,NKIND
      IF(NOW.EQ.NAME(LT))GO TO 170            !Match with circuit NAME
      ENDDO
      TYPE *,'****** UNDEFINED CIRCUIT NAME ******'
      IERR=1
      GO TO 500
170   NC=NC+1
      IF(NC.LE.500)GO TO 175  !Only 500 Networks allowed
      TYPE *,'****** TOO MANY NETWORKS SPECIFIED ******'
      STOP
175   KIND(NC)=LT                             !Circuit number = LT
      M=M+2
190   M=M+1
      DO K=M,130
         IF(ITEM(K).EQ.'!')GO TO 500  !Next line if Comment
         IF(ITEM(K).EQ.SPA)GO TO 185  !Look for space character
      ENDDO
185   M=K

      DO 300 J=1,13
      DO 310 K=M,130
      IF(ITEM(K).EQ.'!')GO TO 500             !Next line if Comment
      IF(ITEM(K).EQ.SPA)GO TO 310             !Passover space characters
      ND(1:1)=ITEM(K)
      ND(2:2)=ITEM(K+1)
      DO NVAR=1,13
      IF(ND.EQ.NDATA(NVAR,LT))GO TO 195       !Match data variable names
      ENDDO

      TYPE *,'****** UNDEFINED VARIABLE NAME ******'
      IERR=1
      GO TO 500
310   CONTINUE
      IF(K.GE.130)GO TO 500
195   IF(LT.EQ.12)CALL INFILTER(NVAR,K,ITEM) !Special FILTER Handling
      M=K+1
      DO K=M,130
      IF(ITEM(K).EQ.SPA.AND.ITEM(K+1).NE.SPA)GO TO 200
```

```
                                                        !  INPUT

      ENDDO
      TYPE *,'****** INCORRECT DATA FORMAT ******'
      IERR=1
      GO TO 500

200   M=K+1
      RLC(NC,NVAR)=DIGIT(M,ITEM,IERR)  !Convert the numeric value
      IF(IERR.LT.2)GO TO 300           !Bad input
      TYPE *,'****** VARIABLE VALUE ERROR ******'
      GO TO 500

300   CONTINUE
350   CALL CONTROL(M,NC,ITEM,TDIS,TOFF,DC,FS,VOFF,IERR,NEND,NEYE,nff,npp
      IF(NEND.GT.0)GO TO 600

500   CONTINUE         !Comment or blank line, no processing, next line

600   CLOSE(UNIT=1)    !.END, end of input file
      IF(MOUT.GT.0)GO TO 1000
      TYPE *,'****** NO OUTPUT WAS SPECIFIED ******'
1000  TYPE *,'*************************************************************
      TYPE *,' '
      TYPE *,' THE FOLLOWING OUTPUT FILES WILL BE CREATED: '
      type *,' '
      type *,BFIL(1:time_graph_size ), ' = time domain REGIS file'
      type *,DFIL(1:time_data_size ), ' = time domain data file'
      IF(NFF.EQ.1)type *,FGL(1:freq_graph_size ),' = freq. domain REGIS'
      IF(NFF.EQ.1)type *,FDL(1:freq_data_size ), ' = freq. domain data'
      IF(NPP.EQ.1)type *,PGL(1:phase_graph_size),'= phase vs. freq. REGIS
      IF(NPP.EQ.1)type *,PDL(1:phase_data_size),'= phase vs. freq. data fi
      type *,' '
      TYPE *,'*************************************************************

      IF(IERR.GT.0)STOP
      RETURN
      END

!
```

```
      SUBROUTINE LINE(N,W)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C     SOLVES FOR THE A,B,C,D PARAMETERS FOR A TRANSMISSION LINE           C
C     FOR THE Nth NETWORK ELEMENT AT FREQUENCY W (TYPE 4 CIRCUIT)          C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      COMPLEX*16 ABCD(500,2,2),ZERO,Z,Y,ZO,GAM,GAMJ,A,B,C,Q,CON
      DOUBLE PRECISION PI2,X
      COMMON/NET/RLC(500,20),ABCD,PI2,ZERO
      CON=DCMPLX(0.0,-1.0)
      BC=W*RLC(N,20)
      W1=RLC(N,11)
      IF(W1.EQ.0.0)GO TO 5
      IF(W.LE.W1)GO TO 5
      W2=RLC(N,12)
      IF(W2.EQ.0.0.OR.W.LE.W2)THEN
        BC=BC*(1.+RLC(N,17)*ALOG(W/W1))
        ELSE
        BC=BC*(1.+RLC(N,7)+RLC(N,18)*ALOG(W/W2))
      ENDIF
5     XL=W*RLC(N,19)
      RT=RLC(N,4)
      QO=RLC(N,16)
      IF(RT.EQ.0.0.OR.QO.EQ.0.0)GO TO 20        !Skin effect calculation
      QO=QO*SQRT(W)
      IF(QO.LT.1.E-5)GO TO 20
      R=0.5*QO*RT
      IF(QO.GT.15)GO TO 10
      R=R/(COSH(QO)-COS(QO))
      RT=R*(SINH(QO)+SIN(QO))
      XL=XL+R*(SINH(QO)-SIN(QO))
      GO TO 20
10    RT=R
      XL=XL+R
20    X=RLC(N,3)
      D=RLC(N,6)
      Z=DCMPLX(RT,XL)
      Y=BC*DCMPLX(D,1.)
      GAM=X*CDSQRT(Z*Y)
      GAMJ=CON*GAM
      A=CDCOS(GAMJ)
      IF(W.GT.1.E-5)GO TO 50
      Q=1.0+GAM**2/6.0
      B=Z*X*Q
      C=Y*X*Q
      GO TO 100
50    Q=CDSIN(GAMJ)/CON
      ZO=CDSQRT(Z/Y)
      B=ZO*Q
      C=Q/ZO
100   ABCD(N,1,1)=A
      ABCD(N,1,2)=B
      ABCD(N,2,1)=C
      ABCD(N,2,2)=A
      RETURN
      END               !
```

```
!    MABCD
!    MOVE


      SUBROUTINE MABCD(A,M,IFLAG)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C     SOLVES THE MATRIX MULTIPLICATION A(I,J) = ABCD(M,I,J)*A(I,J)     C
C                        2/16/87                                       C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      COMPLEX*16 ABCD(500,2,2),ZERO,A(2,2),B(2,2)
      DOUBLE PRECISION PI2
      COMMON/NET/RLC(500,20),ABCD,PI2,ZERO
      N=2
      DO 100 I=1,N
      DO 100 J=1,N
      B(I,J)=ZERO
      DO 100 K=1,N
      B(I,J)=B(I,J)+A(I,K)*ABCD(M,K,J)
      C=CDABS(B(I,J))
      IF(C.GT.1.E34) IFLAG=1
100   CONTINUE
      DO 200 I=1,N
      DO 200 J=1,N
200   A(I,J)=B(I,J)
      RETURN
      END



      SUBROUTINE MOVE(A,M)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C                    MOVES ABCD(M,I,J) TO A(I,J)                       C
C                          12/31/85                                    C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      COMPLEX*16 ABCD(500,2,2),ZERO,A(2,2),B(2,2)
      DOUBLE PRECISION PI2
      COMMON/NET/RLC(500,20),ABCD,PI2,ZERO
      N=2
      DO 100 I=1,N
      DO 100 J=1,N
100   A(I,J)=ABCD(M,I,J)
      RETURN
      END

!
```

```
                                                        !  PRECALC

      SUBROUTINE PRECALC(NC,NF,TDIS,TOFF,NGEN)
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                        c
c     CALCULATION OF CERTAIN CIRCUIT PARAMETERS TO REDUCE                c
c     RECURRENT CALCULATIONS, INITIALIZATIONS, & OVERLAPPING 1           c
c     CORRECTION                                                         c
c                                                                        c
c                                                                        c
c     NC   = number of network sections (circuits) in AFILE              c
c     NF   = number of frequency terms computed in Fourier Transform     c
c     NGEN = total number of generators in model in AFILE                c
c     TDIS = amount of time to be graphed                                c
c     EDGE = array of frequency terms for a SRC model                    c
c     XX   = array of time dependent voltages for SRC input;             c
c            also for real part of Fourier transform of time waveform    c
c     YY   = array for imaginary part of Fourier Transform of SRC        c
c            time domain waveform                                        c
c                                                                        c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      COMPLEX*16 ABCD(500,2,2),ZERO
      complex*16 edge(8192)
      DOUBLE PRECISION PER,PI2
      DIMENSION TOFF(5),dcl(500)
      dimension xx(8192),yy(8192)
      COMMON/NET/RLC(500,20),ABCD,PI2,ZERO
      COMMON/MISC/KIND(500),RGEN,NOUT(5),MOUT,KD(5),iff
      COMMON/TIME/PER
      common/exptl/edge
      DATA XX/8192*0/,YY/8192*0/
      DEL=1000000.
      NGEN=0
      ndgen=0            !Count the number of GEN networks
      nsrc=0             !Count number of SRC networks
      TRM=1.E10
      IDF=0
! Setup for default for dual slope case
      trl=rlc(1,2)
      tr2=rlc(2,2)
      if((tr2.ne.trl).and.(tr2.ne.0.0)) idf=1   !Dual slope case

      DO 50 I=1,NC
      GO TO(120, 200, 300, 400, 500, 600, 700, 800, 900,1000,
     1    1100,1200,1300,1400,1500,1600,1700,1800,1900,2000),KIND(I)
C GEN-GENERATOR INPUT PROCESSING
! Upon exitting PRECALC the GEN entires in RLC are:
!   10-90% rise and fall times in elements 2 and 9 respectively
!   50%   pulse width in element 1
20    IF(RLC(I,1).NE.0.0)GO TO 125
      TYPE *,' ****** GENERATOR PULSE WIDTH NOT SPECIFIED ******'
      STOP
25    NGEN=NGEN+1
      ndgen=ndgen+1
      dcl(i)=rlc(i,7)
      ick=jmod(i,2)          !Check for even or odd GEN
      IF(I.EQ.1)GO TO 130                      ! Check for 1st GEN
```

```
                                                        !  PRECALC

      IF(RLC(I,3).EQ.0.0)RLC(I,3)=RLC(I-1,3)     ! Set PER = previous PER
      IF(RLC(I,6).EQ.0.0)RLC(I,6)=RLC(I-1,6)     ! Set BW = previous BW
      IF(idf.eq.0)go to 21    ! Single slope case
      if((ick.eq.1).and.(rlc(i,2).eq.0.0))rlc(i,2)=rlc(1,2) ! tr def
      if((ick.eq.0).and.(rlc(i,2).eq.0.0))rlc(i,2)=rlc(2,2)
      if((ick.eq.1).and.(rlc(i,9).eq.0.0))rlc(i,9)=rlc(1,9) ! tf def
      if((ick.eq.0).and.(rlc(i,9).eq.0.0))rlc(i,9)=rlc(2,9)
      if((ick.eq.1).and.(rlc(i,4).eq.0.0))rlc(i,4)=rlc(1,4) !amp def
      if((ick.eq.0).and.(rlc(i,4).eq.0.0))rlc(i,4)=rlc(2,4)
      go to 130

21    IF(RLC(I,2).EQ.0.0)RLC(I,2)=RLC(I-1,2)     ! Set tr= previous tr
      IF(RLC(I,9).EQ.0.0)RLC(I,9)=RLC(I-1,9)     ! Set tf = previous tf
      IF(RLC(I,4).EQ.0.0)RLC(I,4)=RLC(I-1,4)     ! SET AMP= prev. AMP

130   P=RLC(I,3)                                          !period
      IF(P.GT.PER)PER=P
      TR=RLC(I,2)                                         !10-90% risetime
      TR=.75*TR                                           !20-80% risetime
      IF(TR.EQ.0.0)TR=1.E-4*PER                           !TR default
      BW=RLC(I,6)                                         !Bandwidth
      WO=PI2*BW                                           !Rad/sec
      IF(WO.EQ.0.0.OR.WO.GT.(50./TR))WO=50./TR
      RLC(I,16)=WO              !Place WO in location 16
      TRO=ALOG(4.)/WO           !20-80 Delay for a step function
      IF(TRO.GT.TR)GO TO 110
      TRO=TR                    !Compute the 0-100 % Ramp risetime T
      T=0.99*TR
      DO J=1,100
       TRO=RISE(WO,T,0.8)-RISE(WO,T,0.2)
       IF(ABS(TRO-TR).LT.1.E-4)GO TO 140
       T=T+1.2*(TR-TRO)
      ENDDO

110   T=0.0
      TYPE 115,I,T
115   FORMAT(' Line ',I2,' Risetime cannot be achieved with',
     1 ' this bandwidth')
      STOP
140   T=T/0.6
      RLC(I,20)=T                          !Place T in location 20
      IF(TRM.GT.T)TRM=T
      TDR50=RISE(WO,(0.6*T),0.5)-T/2.
      IF(NGEN.EQ.1)TDR51=TDR50             !Setup for overlapping 1's
      IF(NGEN.EQ.2)TDR52=TDR50             !Setup for overlapping 1's
      TF=RLC(I,9)                          ! 10-90% falltime
      TF=.75*TF                            ! 20-80% falltime
      IF(TF.EQ.0.0)TF=TR
      IF(TF.EQ.TR)GO TO 170
145   TRO=ALOG(4.)/WO
      IF(TRO.GT.TF)GO TO 150    !20-80 Delay for a step function
      TRO=TF
      T=0.99*TF                 !Compute the 0-100 % Ramp risetime T
      IF(I.EQ.1)DEL1=RLC(1,5)
!                                          !used in overlapping 1's section
```

```
                                                    !  PRECALC

      DO J=1,100
       TRO=RISE(WO,T,0.8)-RISE(WO,T,0.2)
       IF(ABS(TRO-TF).LT.1.E-4)GO TO 160
       T=T+1.2*(TF-TRO)
      ENDDO

150   T=0.0
      TYPE 115,I,T
160   T=T/0.6
      TDF50=RISE(WO,(0.6*T),0.5)-T/2.
      !  convert "ideal" pulse width (instantaneous rise and fall
      !  times) to real 50% crossing width assumed for the rest of GEN
      !  setup and calculations
      RLC(I,1)=RLC(I,1)+(5.*(TF-TR)/6.)
      RLC(I,1)=RLC(I,1)+TDR50-TDF50    !Modify pulse width
170   RLC(I,19)=T                      !Place T in location 19
      IF(RLC(I,4).EQ.0.0)RLC(I,4)=1.0          !Amplitude defaults to one
      RLC(I,5)=RLC(I,5)+RLC(I,1)/2.-TDR50
      RGEN=RGEN+RLC(I,8)                       !Generator resistance sum
      GO TO 50

C Type 2 Circuit (TEE)
200   RLC(I,16)=PI2*RLC(I,7)
      RLC(I,17)=PI2*RLC(I,8)
      IF(RLC(I,9).NE.0.0)RLC(I,3)=1./RLC(I,9)  !G = 1./R
      GO TO 50
C Type 3 circuit (DELTA)
300   IF(RLC(I,7).NE.0.0)RLC(I,1)=1./RLC(I,7)  !G1 = 1./R1
      IF(RLC(I,8).NE.0.0)RLC(I,2)=1./RLC(I,8)  !G2 = 1./R2
      GO TO 50

C Type 4 Circuit (LINE)
400   IF(RLC(I,1).NE.0.0.AND.RLC(I,2).NE.0.0.AND.RLC(I,3).NE.0.0)
      1 GO TO 410
405   TYPE *,' ****** LINE IMPROPERLY SPECIFIED ******'
      STOP
410   IF(RLC(I,10).EQ.0.0)RLC(I,10)=1.0        !Q defaults to one
      RLC(I,11)=PI2*RLC(I,11)          !F1 -> 2 PI F1
      RLC(I,12)=PI2*RLC(I,12)          !F2 -> 2 PI F2
      RLC(I,13)=PI2*RLC(I,13)          !F3 -> 2 PI F3
      IF(RLC(I,12).EQ.0.0.AND.RLC(I,11).NE.0.0)GO TO 405
      IF(RLC(I,11).NE.0.0)RLC(I,17)=RLC(I,7)/ALOG(RLC(I,12)/RLC(I,11))
      IF(RLC(I,13).NE.0.0.AND.RLC(I,12).NE.0.0)
      1 RLC(I,18)=RLC(I,8)/ALOG(RLC(I,13)/RLC(I,12))
      RLC(I,19)=RLC(I,1)*RLC(I,2)      !Place Inductance in location 19
      RLC(I,20)=RLC(I,2)/RLC(I,1)      !Place Capacitance in location 20
      IF(RLC(I,9).EQ.0.0)GO TO 420     !WC = 0
      IF(RLC(I,5).GT.0.0)GO TO 425     !TC > 0, WC > 0
      TYPE *,' ****** WC CANNOT BE SPECIFIED IF TC IS NOT SPECIFIED ******'
      STOP
425   RLC(I,4)=1.7241E-6*RLC(I,10)/(RLC(I,9)*RLC(I,5))        !Compute Rdc
420   RLC(I,16)=RLC(I,5)*SQRT(PI2/(1.7241E-6*RLC(I,10)))!Compute const
      GO TO 50
                    !
```

```
C  Type 5 Circuit (STUB)
500     IF(RLC(I,1).NE.0.0.AND.RLC(I,2).NE.0.0.AND.RLC(I,3).NE.0.0)
        1 GO TO 510
        TYPE *,' ****** STUB IMPROPERLY SPECIFIED ******'
        STOP
510     RLC(I,17)=RLC(I,7)/PI2              !K1 > K1/2 PI
        RLC(I,18)=RLC(I,8)/PI2              !K2 > K2/2 PI
        RLC(I,19)=RLC(I,1)*RLC(I,2)         !Place Inductance in location 19
        RLC(I,20)=RLC(I,2)/RLC(I,1)         !Place Capacitance in location 20
        IF(RLC(I,10).NE.0.0)RLC(I,9)=1./RLC(I,10)         !GL = 1./RL
        IF(RLC(I,12).EQ.0.0)GO TO 520       !WC = 0
        IF(RLC(I,5).GT.0.0)GO TO 525        !TC > 0, WC > 0
        TYPE *,' ****** WC CANNOT BE SPECIFIED IF TC IS NOT SPECIFIED *****
        STOP
525     RLC(I,4)=1.7241E-6/(RLC(I,12)*RLC(I,5))  !Compute Rdc
520     RLC(I,16)=RLC(I,5)*SQRT(PI2/1.7241E-6)!Compute const
        GO TO 50
C  Type 6 Circuit(EAMP)
600     IF(RLC(I,1).EQ.0.0)RLC(I,1)=1.0                 ! GAIN defaults to 1
        IF(RLC(I,2).LT.1.E-6)RLC(I,2)=1.E-6             ! R defaults to 1E-6
        BW=RLC(I,3)
        WO=PI2*BW
        IF(WO.EQ.0.0)WO=50./TRM                !F defaults to MAX
        RLC(I,16)=WO                           !Place WO in location 16
        GO TO 50
C  Type 7 Circuit(FIBER)
700     IF(RLC(I,6).EQ.0.0)RLC(I,6)=1.0                 !QM defaults to one
        IF(RLC(I,7).EQ.0.0)RLC(I,7)=1.0                 !QC defaults to one
C    Code for Siecor Dispersion Model
        IF(RLC(I,5).NE.0.0)GO TO 720                    !D not zero
        ALC=RLC(I,9)
        IF(ALC.NE.0.0)GO TO 710
        TYPE *,'****** NO LC SPECIFIED ******'          !LC not zero
        STOP
710     DP=0.25*RLC(I,11)*(ALC-RLC(I,10)**4/ALC**3)
        rlc(i,5)=.85*rlc(i,3)*rlc(i,11)
        rlc(i,5)=rlc(i,5)**2                            !0.85*FWHM*SO
        rlc(i,5)=rlc(i,5)/8.
        rlc(i,5)=(dp**2)+rlc(i,5)
        RLC(I,5)=SQRT(rlc(i,5))
        RLC(I,5)=1.176*RLC(I,5)
720     XM=RLC(I,8)**RLC(I,6)
        XC=RLC(I,8)**RLC(I,7)                           !Intermodal length correcti
        A=ALOG(10.)/10.                                 !Chromatic length correctio
        B=4.*SQRT(ALOG(2.))
        C=A/B*RLC(I,2)*RLC(I,3)*XC
        RLC(I,18)=C
        RLC(I,16)=A*ABS(RLC(I,1))*XC-C**2               !Wavelength dependent atte
        RLC(I,17)=RLC(I,3)*RLC(I,5)*XC/B                !Total attenuation
        IF(RLC(I,4).LT.1.E-4)RLC(I,19)=0.0              !TC, chromatic dispersio
        IF(RLC(I,4).GE.1.E-4)RLC(I,19)=B*XM/RLC(I,4)/(4.*PI2) !TM,material di
        GO TO 50                                        !FI defaults to infinity
!
```

```
C Type 8 Circuit(GSIN)
800     IF(RLC(I,1).NE.0.0)GO TO 810
        TYPE *,'****** GENERATOR FREQUENCY NOT SPECIFIED ******'
        STOP

810     RLC(I,16)=PI2*RLC(I,1)                  !F > WO
        IF(RLC(I,2).EQ.0.0)RLC(I,2)=1.0 !AMP defaults to one
        NGEN=NGEN+1                             !Increase number of generators
        RGEN=RGEN+RLC(I,5)                      !Generator resistance sum
        P=1./RLC(I,1)
        IF(P.GT.PER)PER=P
        IF(TRM.GT.P)TRM=P
        RLC(I,4)=RLC(I,4)+1./(4.*RLC(I,1))      !Add quadrature phase shift
        GO TO 50

C Type 9 Circuit(IAMP)
900     IF(RLC(I,1).EQ.0.0)RLC(I,1)=1.0         ! GAIN defaults to 1
        IF(RLC(I,2).NE.0.0)RLC(I,3)=1./RLC(I,2) !If R not zero, G=1/R
        IF(RLC(I,3).LT.1.E-6)RLC(I,3)=1.E-6     !G defaults 1.E-6
        BW=RLC(I,4)
        WO=PI2*BW
        IF(WO.EQ.0.0)WO=50./TRM                 !F > MAX
        RLC(I,16)=WO                            !Place WO in location 16
        GO TO 50

C Type 10 Circuit(HPF)
1000    IF(RLC(I,1).GT.0.0)GO TO 1010
        TYPE *,'****** C MISSING FROM HPF NETWORK ******'
        STOP
1010    IF(RLC(I,4).NE.0.0)RLC(I,3)=1./RLC(I,4)
        IF(RLC(I,3).GT.0.0)GO TO 1020
        TYPE *,'****** SHUNT CONDUCTANCE MISSING FROM HPF NETWORK ******'
        STOP
1020    RLC(I,20)=RLC(I,2)*RLC(I,3)             !G * L
        RLC(I,19)=RLC(I,3)/RLC(I,1)             !G / C
        GO TO 50

C Type 11 Circuit(XFORMER)
1100    IF(RLC(I,1).EQ.0.0.AND.RLC(I,5).EQ.0.0)GO TO 1110 !L1 = 0 & N = 0
        IF(RLC(I,2).EQ.0.0.AND.RLC(I,5).EQ.0.0)GO TO 1110 !L2 = 0 & N = 0
        IF(RLC(I,1).EQ.0.0.AND.RLC(I,2).EQ.0.0)GO TO 1110 !L1 = 0 & L2 = 0
        IF(RLC(I,1).EQ.0.0)RLC(I,1)=RLC(I,2)/RLC(I,5)**2  !L1 = 1 / N**2
        IF(RLC(I,2).EQ.0.0)RLC(I,2)=RLC(I,1)*RLC(I,5)**2  !L2 = L1 N**2
        IF(RLC(I,4).GT.0.0)RLC(I,3)=RLC(I,4)*SQRT(RLC(I,1)*RLC(I,2))
        IF(RLC(I,3).EQ.0.0)RLC(I,3)=SQRT(RLC(I,1)*RLC(I,2)) !K > 1
      RLC(I,16)=RLC(I,1)/RLC(I,3)               !L1/M
      RLC(I,17)=RLC(I,2)/RLC(I,3)               !L2/M
      RLC(I,18)=RLC(I,1)*RLC(I,2)/RLC(I,3)-RLC(I,3)    !L1 L2 / M - M
      GO TO 50

1110  TYPE *,'****** INDUCTOR MISSING FROM TRANSFORMER ******'
      STOP
!
```

```
                                                        !  PRECALC

C  Type 12 Circuit(FILTER)
1200    IF(RLC(I,5).NE.0.0)RLC(I,4)=1.0/RLC(I,5)  !G1 > 1 / R1
        IF(RLC(I,2).EQ.0.0)RLC(I,2)=1.0
        IF(RLC(I,6).NE.0.0)RLC(I,1)=0.25/RLC(I,6)  !First Zero
        IF(RLC(I,1).EQ.0.0)GO TO 1210
        GO TO 50

1210    TYPE *,'****** To MUST BE SPECIFIED IN FILTER ******'
        STOP
C  Type 13 Circuit(SHUNT)
1300    IF(ABS(RLC(I,11)).EQ.0.0)RLC(I,11)=1.0  !K defaults to One
        bt=abs(rlc(i,6))+abs(rlc(i,7))+abs(rlc(i,8))+abs(rlc(i,9))
        bt=bt+abs(rlc(i,10))
        if(bt.eq.0)rlc(i,6)=1.     !set B0=1 if no B's spec'd
        GO TO 50
C  Type 14 Circuit(SERIES)
1400    IF(ABS(RLC(I,11)).EQ.0.0)RLC(I,11)=1.0  !K defaults to One
        bt=abs(rlc(i,6))+abs(rlc(i,7))+abs(rlc(i,8))+abs(rlc(i,9))
        bt=bt+abs(rlc(i,10))
        if(bt.eq.0)rlc(i,6)=1.     !set B0=1 if no B's spec'd
        GO TO 50
C  Type 15 Circuit(BPF)
1500    rlc(i,1)=pi2*rlc(i,1)              !w1
        rlc(i,2)=pi2*rlc(i,2)              !w2
        if(rlc(i,3).eq.0)rlc(i,3)=6.       !S1 default = 6 dB/octave
        if(rlc(i,4).eq.0)rlc(i,4)=6.       !S2 default = 6 dB/octave
        rlc(i,3)=abs(rlc(i,3))
        rlc(i,4)=abs(rlc(i,4))
        if(rlc(i,5).eq.0)rlc(i,5)=1.       !K default = 1.  (gain)
        rlc(i,16)=rlc(i,1)*(1.+3./rlc(i,3))      !AA
        rlc(i,17)=rlc(i,2)/((1.+3./rlc(i,4)))    !BB
        rlc(i,18)=1-(46.99/rlc(i,3))
        rlc(i,18)=rlc(i,1)*rlc(i,18)             !a = low freq cutoff
        rlc(i,19)=1+(50/rlc(i,4))
        rlc(i,19)=rlc(i,17)*rlc(i,19)            !b = hi freq. cutoff
        RLC(I,20)=0.5*(RLC(I,1)+RLC(I,2))        !m=midpt betwn w1 and w2
        rlc(i,14)=rlc(i,1)-rlc(i,2)
        rlc(i,14)=(2*rlc(i,6))/rlc(i,14)         !ps=slope of phase line
        GO TO 50
C  Type 16 Circuit (SOURCE)
1600    ngen=ngen+1
        nsrc=nsrc+1
        if(nsrc.le.1)go to 1601
        type *,' ONLY 1 SRC ALLOWED PER MODEL '
        stop
1601    rgen=rgen+rlc(i,9)
        P=rlc(i,3)
        if(p.gt.per)per=p
        OPEN(UNIT=1,FILE='SOURCE.DTA',STATUS='OLD')
        TYPE=RLC(I,8)
        IF((TYPE.EQ.1).OR.(TYPE.EQ.2))GO TO 5002
        TYPE *,' '

!
```

```
      TYPE *,' DATA FILE TYPE INCORRECT '
      STOP
5002  nt=RLC(1,6)      !Number of points, must be .le. 8192
      if(nt.le.8192)go to 6002
      type *,' TOO MANY POINTS IN SOURCE.DTA '
      stop
6002  do ii=1,11
      if(nt.lt.2**ii)go to 5071
      enddo
5071  nt=2**(ii-1)     !Reset number of points to be a power of 2
      if(rlc(1,6).eq.nt)go to 5051
      type *,' '
      type *,' NOT ALL POINTS IN SOURCE.DTA WILL BE READ; reduced '
      TYPE *,' to a power of 2 as required '
      TYPE *,' '
      dt=rlc(1,7)              !Time increment between points
      per=(nt-1)*dt            !Correct period for truncated waveform

5051  IF(TYPE.EQ.1)GO TO 5003   !SOURCE.DTA already has frequency data

!     Read time domain data

      DO ii=1,NT
      READ(1,5662)XX(ii)
5662  FORMAT(F20.3)
      ENDDO
      CLOSE(UNIT=1)
! Calculate frequency spectrum of input waveform and place it
! in complex COMMON array EDGE
      CALL FFT842(1,NT,XX,YY)
      DO ii=1,NT
      EDGE(ii)=CMPLX(XX(ii),YY(ii))
      edge(ii)=dconjg(edge(ii))
      ENDDO
      GO TO 50
5003  dt=per/nt
      rlc(1,7)=amax1(dt,rlc(1,7))
      do ii=1,nt
      read (1,5772)edge(ii)
5772  format(2e16.2)  !May be changed for different data format
      enddo
      CLOSE(UNIT=1)
      go to 50

C Type 17 Circuit
1700  CONTINUE
C Type 18 Circuit
1800  CONTINUE
C Type 19 Circuit
1900  CONTINUE
C Type 20 Circuit
2000  CONTINUE
50    CONTINUE
!
```

```
!
!       Correction for overlapping 1's so that source level does not
!       exceed normalized "ON" level amplitude
!
        IF(KIND(1).NE.1)GO TO 555    !Skip overlapping 1's correction
                                     !if 1st circuit not a GEN

        N1=NGEN
        IGEN=1          !One or two trapezoids per edge
        IF(RLC(1,2).NE.RLC(2,2))N1=NGEN-1        !Two trapezoid case
        F1=5*RLC(1,9)/4.                         !0-100% falltime
        R1=5*RLC(1,2)/4.                         !0-100% risetime
        F2=5*RLC(2,9)/4.
        R2=5*RLC(2,2)/4.
        PN1=RLC(N1,1)+0.5*(R1-F1)   !ideal width (next to) last GEN
        PN=RLC(NGEN,1)+.5*(R2-F2)   !ideal width for last GEN
        P1=RLC(1,1)+0.5*(R1-F1)          !ideal pulse width for 1st
        P2=RLC(2,2)+0.5*(R2-F2)          !ideal width for 2nd GEN

        IF(N1.NE.NGEN)GO TO 9556
        TDR52=TDR51
9556    IF(N1.EQ.NGEN)GO TO 9559
        IGEN=2
9559    DN1=RLC(N1,5)-RLC(N1,1)/2.+TDR51
        DN=RLC(NGEN,5)-RLC(NGEN,1)/2.+TDR52
        CORR=ABS(RLC(N1,3)+DEL1-DN1)          !CORR=|PER+DEL1-DN1|
        IF((RLC(1,3)+DEL1).LE.(DN1+PN1))GO TO 9555

        IF((RLC(1,3)+DEL1).GE.(DN1+PN1+F1))GO TO 555
        IF((DN1+PN1+F1).LE.(RLC(1,3)+DEL1+R1))GO TO 555

9555    RLC(N1,1)=RLC(1,1)+CORR          !correct width (next to) last GEN
        RLC(NGEN,1)=RLC(IGEN,1)+CORR  !correct width last GEN
        RLC(N1,5)=DN1+RLC(N1,1)/2.-TDR51
        RLC(NGEN,5)=DN+RLC(NGEN,1)/2.-TDR52
        IBG=IGEN+1

        DO I=IBG,NC                  ! drop extra GEN and reset pointers
        DO J=1,20
        RLC(I-IGEN,J)=RLC(I,J)
        KIND(I-IGEN)=KIND(I)
        ENDDO
        ENDDO
        NGEN=NGEN-IGEN
        NC=NC-IGEN

        do i=2,(ndgen-2)  !replace DC offsets of GEN networks after correcti
        rlc(i-1,7)=dcl(i)
        enddo
        rlc(ndgen-1,7)=dcl(1)

        DO I=1,MOUT
        NOUT(I)=NOUT(I)-IGEN
        ENDDO

                                                              !
```

```
                                                    !  PRECALC
555     ABCD(NGEN,1,1)=DCMPLX(1.0,0.0)
        ABCD(NGEN,1,2)=DCMPLX(RGEN,0.0)
        ABCD(NGEN,2,1)=ZERO
        ABCD(NGEN,2,2)=DCMPLX(1.0,0.0)


C DISPLAY PROCESSING
        IF(TDIS.EQ.0.0)TDIS=PER !Default value for horizontal display scale

        DO K=1,MOUT      !Make sure delay is in range
        DO I=1,100
        IF(TOFF(K).LT.(I*PER))GO TO 9999
        ENDDO
9999    TOFF(K)=TOFF(K)-(I-1)*PER
        ENDDO

CALCULATING THE NUMBER OF FREQUENCY TERMS TO BE COMPUTED

        if(kind(1).ne.16)go to 5114  !SRC determines number of terms
        nf=nt
        go to 5115

5114    IF(32768.*TRM.LT.PER)TRM=PER/32768.
        X=PER/TRM
        K=ALOG10(10.*X)/ALOG10(2.)
        IF(K.LT.8)K=8
        IF(K.GT.13)K=15
        NF=2**K
        ict=0
5225    anf=nf
        err=per/anf      !Check time increment magnitude=error in timing jitter
        if(err.le.0.125)go to 5115
        if(nf.lt.32768)nf=2*nf
        ict=ict+1
        if(ict.ge.2)go to 5115
        go to 5225
5115    if((iff.eq.1).and.(nf.gt.8192))nf=8192
        RETURN
        END
```

```
      FUNCTION RISE(WGEN,TR,A)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                          C
C  COMPUTES THE DELAY FOR THE INPUT WAVEFORM                               C
C  WGEN IS THE GENERATOR FILTER 3DB FREQUENCY (RAD/SEC),                   C
C  TR IS THE 20 % TO 80 % GENERATOR RISETIME                               C
C  A IS THE FRACTIONAL AMPLITUDE                                           C
C                                                                          C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      C=WGEN*TR/0.6
      X=A*C
      DO I=1,100
      Y=EXP(-X)
      IF(X.GT.C) THEN                              !Delay > TR
       F=C*(1.-A)+Y*(1.-EXP(C))
       DF=Y*(EXP(C)-1.)
      ELSE                                         !Delay <= TR
       F=X-1.0-A*C+Y
       DF=1.-Y
      END IF
      IF(ABS(F).LT.1.E-6)GO TO 200
      X=X-F/DF
      ENDDO
      TYPE *,'NON-CONVERGENT'
200   RISE=X/WGEN
      RETURN
      END

      SUBROUTINE  SERIES(N,W)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C            POLYNOMIAL MODEL FOR A SERIES IMPEDANCE                       C
C               UP TO FOUR ZEROS AND FOUR POLES                            C
C                    (TYPE 14 CIRCUIT)                                     C
C                       12/17/86                                           C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      COMPLEX*16 ABCD(500,2,2),ZERO
      DOUBLE PRECISION PI2,PR,PI,ZR,ZI
      COMMON/NET/RLC(500,20),ABCD,PI2,ZERO
      W2=W**2
      W3=W**3
      W4=W**4
      ZR=RLC(N,1)-W2*RLC(N,3)+W4*RLC(N,5)
      ZI=W*RLC(N,2)-W3*RLC(N,4)
      PR=RLC(N,6)-W2*RLC(N,8)+W4*RLC(N,10)
      PI=W*RLC(N,7)-W3*RLC(N,9)
      IF(DABS(PR).LT.1.E-16) PR=1.E-16
      ABCD(N,1,1)=1.0
      ABCD(N,1,2)=DCMPLX(ZR,ZI)/(RLC(N,11)*DCMPLX(PR,PI))
      ABCD(N,2,1)=ZERO
      ABCD(N,2,2)=1.0
      RETURN
      END
```

!

! SHUNT
! SOURCE

```
      SUBROUTINE SHUNT(N,W)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C          POLYNOMIAL MODEL FOR A SHUNT IMPEDANCE
C             UP TO FOUR ZEROS AND FOUR POLES                       C
C                  (TYPE 13 CIRCUIT)                                C
C                      5/21/86                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      COMPLEX*16 ABCD(500,2,2),ZERO
      DOUBLE PRECISION PI2,PR,PI,ZR,ZI
      COMMON/NET/RLC(500,20),ABCD,PI2,ZERO

      W2=W**2
      W3=W**3
      W4=W**4
      ZR=RLC(N,1)-W2*RLC(N,3)+W4*RLC(N,5)
      ZI=W*RLC(N,2)-W3*RLC(N,4)
      PR=RLC(N,6)-W2*RLC(N,8)+W4*RLC(N,10)
      PI=W*RLC(N,7)-W3*RLC(N,9)
      IF(DABS(ZR).LT.1.E-16) ZR=1.E-16
      ABCD(N,1,1)=1.0
      ABCD(N,1,2)=ZERO
      ABCD(N,2,1)=DCMPLX(PR,PI)/(RLC(N,11)*DCMPLX(ZR,ZI))
      ABCD(N,2,2)=1.0
      RETURN
      END


      FUNCTION SOURCE(N,W)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C     RETURNS THE SOURCE GENERATOR FREQUENCY COMPONENT      C
C     FOR THE Nth NETWORK ELEMENT AT FREQUENCY W            C
C     TYPE 16 CIRCUIT - SRC, experimental data              C
C     3/11/87                                               C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      COMPLEX*16 ABCD(500,2,2),ZERO,source
      COMPLEX*16 EDGE(8192)
      COMMON/NET/RLC(500,20),ABCD,PI2,ZERO
      COMMON/TIME/PER
      COMMON/EXPTL/EDGE
!
! return source(n,w) from table lookup and fold spectrum abt NF/2
!
      AX=((W*PER)/PI2)+1          !index of frequency term
      ix=ax
      SOURCE=EDGE(IX)
      ifold=rlc(n,6)/2            ! fold spectrum abt NF/2
      if(ix.ne.1.and.ix.le.ifold) source=2*source
      if(ix.gt.ifold) source=0.
      return
      end
```

```
      SUBROUTINE STUB(N,W)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C     SOLVES FOR THE A,B,C,D PARAMETERS FOR A TRANSMISSION LINE STUB  C
C     FOR THE Nth NETWORK ELEMENT AT FREQUENCY W (TYPE 5 CIRCUIT)     C
C                        12/30/85                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      COMPLEX*16 ABCD(500,2,2),ZERO,Z,Y,ZO,GAM,GAMJ,A,C,Q,CON,YL
      DOUBLE PRECISION PI2,X
      COMMON/NET/RLC(500,20),ABCD,PI2,ZERO

      CON=DCMPLX(0.0,-1.0)
      BC=W*RLC(N,20)*(1.0+ALOG(1.0+W*(RLC(N,17)+W*RLC(N,18))))
      RT=RLC(N,4)
      XL=W*RLC(N,19)

      QO=RLC(N,16)
      IF(RT.EQ.0.0.OR.QO.EQ.0.0) GO TO 20          !Skin efffect calculation
      QO=QO*SQRT(W)
      IF(QO.LT.1.E-5) GO TO 20
      R=0.5*QO*RT
      IF(QO.GT.15) GO TO 10
      R=R/(COSH(QO)-COS(QO))
      RT=R*(SINH(QO)+SIN(QO))
      XL=XL+R*(SINH(QO)-SIN(QO))
      GO TO 20
10    RT=R
      XL=XL+R

20    X=RLC(N,3)
      D=RLC(N,6)

      YL=CMPLX(RLC(N,9),W*RLC(N,11))
      Z=DCMPLX(RT,XL)
      Y=BC*DCMPLX(D,1.)
      GAM=X*CDSQRT(Z*Y)
      GAMJ=CON*GAM
      A=CDCOS(GAMJ)

      IF(W.GT.1.E-5) GO TO 50
      Q=1.0+GAM**2/6.0
      C=(YL*A+Y*X*Q)/(A+YL*Z*X*Q)
      GO TO 100

50    Q=CDSIN(GAMJ)/CON
      ZO=CDSQRT(Z/Y)
      C=(YL*A+Q/ZO)/(A+ZO*YL*Q)
100   ABCD(N,1,1)=DCMPLX(1.0,0.0)
      ABCD(N,1,2)=ZERO
      ABCD(N,2,1)=C
      ABCD(N,2,2)=DCMPLX(1.0,0.0)

      RETURN
      END
!
```

! TEE

```
      SUBROUTINE TEE(N,W)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C     SOLVES FOR THE A,B,C,D PARAMETERS FOR THE TEE CIRCUIT              C
C     FOR THE Nth NETWORK ELEMENT AT FREQUENCY W (TYPE 2 CIRCUIT)        C
C                    12/11/85                                            C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      COMPLEX*16 ABCD(500,2,2),ZERO
      DOUBLE PRECISION PI2
      COMMON/NET/RLC(500,20),ABCD,PI2,ZERO

      R1=RLC(N,1)
      R2=RLC(N,2)
      G=RLC(N,3)
      XL1=W*RLC(N,4)
      XL2=W*RLC(N,5)
      BC=W*RLC(N,6)
      W1=RLC(N,16)
      W2=RLC(N,17)
      IF(W1.EQ.0.0)GO TO 10
      RT=R1*(1.+(W/W1)**2)**.25
      RAC=RT-R1
      R1=RT
      XL1=XL1+RAC
10    IF(W2.EQ.0.0)GO TO 20
      RT=R2*(1.+(W/W1)**2)**.25
      RAC=RT-R2
      R2=RT
      XL2=XL2+RAC
20    ABCD(N,1,1)=DCMPLX( (1.+R1*G-XL1*BC), (XL1*G+R1*BC) )!A
      ABCD(N,1,2)=DCMPLX((R1+R2+G*R1*R2-G*XL1*XL2-BC*(R1*XL2+R2*XL1)),
     1 (XL1+XL2+G*(R1*XL2+R2*XL1)+BC*R1*R2-XL1*XL2*BC) )      !B
                                                              !C
      ABCD(N,2,1)=DCMPLX(G,BC)
      ABCD(N,2,2)=DCMPLX( (1.+R2*G-XL2*BC), (XL2*G+R2*BC) )   !D
      RETURN
      END
```

```
      SUBROUTINE TRANSFORM(NC,NF,NGEN)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C     COMPUTES THE FREQUENCY DOMAIN RESPONSE OF THE NETWORK             C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      DOUBLE PRECISION PER,PI2
      COMPLEX*16 ABCD(500,2,2),ZERO,A(2,2),B(2,2),EW(5,32768),
     1 EG,GEN,GSIN,SOURCE
      CHARACTER*6 NODENAME(54)
      COMMON/NET/RLC(500,20),ABCD,PI2,ZERO
      COMMON/MISC/KIND(500),RGEN,NOUT(5),MOUT,KD(5),iff
      COMMON/WAVES/EW,EOUT(5,32768)
      COMMON/TIME/PER
      DATA IFLAG/0/
      WO=PI2/PER
      TYPE *,' THERE ARE ',NF,' FREQ. TERMS TO COMPUTE'   !busy msg
      type *,' '
      DO 100 M=1,NF
      MM=MOD(M,300)    !busy msg processing
      IF((NF.GT.300).AND.(MM.EQ.0))TYPE 990,M
990   FORMAT(' FOCAS IS ALIVE & WELL . . . freq term: ',I6)
      W=WO*(M-1)
      EG=ZERO
      DO 200 K=1,NC
      GO TO (220,240,260,280,300,320,330,340,350,360,
     1      380,400,420,440,460,480,500,520,540,560),KIND(K)
220   EG=EG+GEN(K,W)
      GO TO 200                             !Type 1 Circuit
240   CALL TEE(K,W)
      GO TO 200                             !Type 2 Circuit
260   CALL DELTA(K,W)
      GO TO 200                             !Type 3 Circuit
280   CALL LINE(K,W)
      GO TO 200                             !Type 4 Circuit
300   CALL STUB(K,W)
      GO TO 200                             !Type 5 Circuit
320   CALL EAMP(K,W)
      GO TO 200                             !Type 6 Circuit
330   CALL FIBER(K,W)
      GO TO 200                             !Type 7 Circuit
340   EG=EG+GSIN(K,W)
      GO TO 200                             !Type 8 Circuit
350   CALL IAMP(K,W)
      GO TO 200                             !Type 9 Circuit
360   CALL HPF(K,W)
      GO TO 200                             !Type 10 Circuit
380   CALL XFORMER(K,W)
      GO TO 200                             !Type 11 Circuit
400   CALL FILTER(K,W)
      GO TO 200                             !Type 12 Circuit
```

```
                                                    !  TRANSFORM
420     CALL SHUNT(K,W)                             !Type 13 Circuit
        GO TO 200
440     CALL SERIES(K,W)                            !Type 14 Circuit
        GO TO 200
460     CALL BPF(K,W)                               !Type 15 Circuit
        GO TO 200
480     eg=eg+SOURCE(K,W)                           !Type 16 Circuit
        GO TO 200
500     CONTINUE                                    !Type 17 Circuit
520     CONTINUE                                    !Type 18 Circuit
540     CONTINUE                                    !Type 19 Circuit
560     CONTINUE                                    !Type 20 Circuit
200     CONTINUE
        N1=NGEN
        CALL MOVE(A,N1)
        DO 5000 K=1,MOUT
        KOUT=NOUT(K)
        IF(KOUT.EQ.N1)GO TO 5200
        NS=N1+1
        DO 5100 I=NS,KOUT
5100    CALL MABCD(A,I,IFLAG)
5200    IF(KOUT.EQ.NC)GO TO 5300
        CALL MOVE(B,(KOUT+1))
        N2=KOUT+2
        IF(N2.GT.NC)GO TO 5400
        DO 5500 I=N2,NC
5500    CALL MABCD(B,I,IFLAG)
5400    EW(K,M)=EG/(A(1,1)+A(1,2)*B(2,1)/B(1,1))
        GO TO 5000
5300    EW(K,M)=EG/A(1,1)
5000    N1=KOUT
        IF(IFLAG.EQ.1)GO TO 6000
100     CONTINUE

        RETURN
6000    DO I=1,14
        IF(M.LT.2**I)GO TO 6100
        ENDDO
6100    NF=2**(I-1)
        RETURN
        END
```

```
      SUBROUTINE XFORMER(N,W)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C              TRANSFORMER MODEL                                        C
C              (TYPE 11 CIRCUIT)                                        C
C                  4/11/86                                              C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      COMPLEX*16 ABCD(500,2,2),ZERO
      DOUBLE PRECISION PI2
      REAL M
      COMMON/NET/RLC(500,20),ABCD,PI2,ZERO

      M=RLC(N,3)
      Q=W*M
      IF(Q.LT.1.E-20)Q=1.E-20
      ABCD(N,1,1)=RLC(N,16)                       !L1/M
      ABCD(N,1,2)=DCMPLX(0.0,W*RLC(N,18))         !W (L1 L2/M - M)
      ABCD(N,2,1)=DCMPLX(0.0,(-1.0/Q))
      ABCD(N,2,2)=RLC(N,17)                       !L2/M

      RETURN
      END
```

```
C-----------------------------------------------------------------------
C SUBROUTINE:  FFT842
C FAST FOURIER TRANSFORM FOR N=2**M
C COMPLEX INPUT
C-----------------------------------------------------------------------

      SUBROUTINE FFT842(IN, N, X, Y)

C THIS PROGRAM REPLACES THE VECTOR Z=X+iY BY ITS  FINITE DISCRETE,
C COMPLEX FOURIER TRANSFORM IF IN=0 (FREQUENCY DOMAIN TO TIME DOMAIN).
C THE INVERSE TRANSFORM IS CALCULATED FOR IN=1.  IT PERFORMS AS MANY BASE
C 8 ITERATIONS AS POSSIBLE AND THEN FINISHES WITH A BASE 4 ITERATION
C OR A BASE 2 ITERATION IF NEEDED.

C THE SUBROUTINE IS CALLED AS SUBROUTINE FFT842 (IN,N,X,Y).
C THE INTEGER N (A POWER OF 2), THE N REAL LOCATION ARRAY X, AND
C THE N REAL LOCATION ARRAY Y MUST BE SUPPLIED TO THE SUBROUTINE.

      DIMENSION X(2), Y(2), L(15)
      COMMON /CON2/ PI2, P7
      EQUIVALENCE (L15,L(1)), (L14,L(2)), (L13,L(3)), (L12,L(4)),
     *    (L11,L(5)), (L10,L(6)), (L9,L(7)), (L8,L(8)), (L7,L(9)),
     *    (L6,L(10)), (L5,L(11)), (L4,L(12)), (L3,L(13)), (L2,L(14)),
     *    (L1,L(15))

      PI2 = 8.*ATAN(1.)
      P7 = 1./SQRT(2.)
      DO 10 I=1,15
        M = I
        NT = 2**I
        IF (N.EQ.NT) GO TO 20
10    CONTINUE
      TYPE *,' N IS NOT A POWER OF TWO FOR FFT842'
      STOP
20    N2POW = M
      NTHPO = N
      FN = NTHPO
      IF (IN.EQ.1) GO TO 40
      DO 30 I=1,NTHPO
        Y(I) = -Y(I)
30    CONTINUE
40    N8POW = N2POW/3
      IF (N8POW.EQ.0) GO TO 60

C RADIX 8 PASSES, IF ANY.

      DO 50 IPASS=1,N8POW
        NXTLT = 2**(N2POW-3*IPASS)
        LENGT = 8*NXTLT
        CALL R8TX(NXTLT, NTHPO, LENGT, X(1), X(NXTLT+1), X(2*NXTLT+1),
     *      X(3*NXTLT+1), X(4*NXTLT+1), X(5*NXTLT+1), X(6*NXTLT+1),
     *      X(7*NXTLT+1), Y(1), Y(NXTLT+1), Y(2*NXTLT+1), Y(3*NXTLT+1),
     *      Y(4*NXTLT+1), Y(5*NXTLT+1), Y(6*NXTLT+1), Y(7*NXTLT+1))
                                                                !
50    CONTINUE
```

```
C IS THERE A FOUR FACTOR LEFT

  60  IF (N2POW-3*N8POW-1) 90, 70, 80

!     GO THROUGH THE BASE 2 ITERATION

  70  CALL R2TX(NTHPO, X(1), X(2), Y(1), Y(2))
      GO TO 90
C
C GO THROUGH THE BASE 4 ITERATION
C
  80  CALL R4TX(NTHPO, X(1), X(2), X(3), X(4), Y(1), Y(2), Y(3), Y(4))
C
  90  DO 110 J=1,15
        L(J) = 1
        IF (J-N2POW) 100, 100, 110
 100    L(J) = 2**(N2POW+1-J)
 110  CONTINUE
      IJ = 1
      DO 130 J1=1,L1
      DO 130 J2=J1,L2,L1
      DO 130 J3=J2,L3,L2
      DO 130 J4=J3,L4,L3
      DO 130 J5=J4,L5,L4
      DO 130 J6=J5,L6,L5
      DO 130 J7=J6,L7,L6
      DO 130 J8=J7,L8,L7
      DO 130 J9=J8,L9,L8
      DO 130 J10=J9,L10,L9
      DO 130 J11=J10,L11,L10
      DO 130 J12=J11,L12,L11
      DO 130 J13=J12,L13,L12
      DO 130 J14=J13,L14,L13
      DO 130 JI=J14,L15,L14
        IF (IJ-JI) 120, 130, 130
 120    R = X(IJ)
        X(IJ) = X(JI)
        X(JI) = R
        FI = Y(IJ)
        Y(IJ) = Y(JI)
        Y(JI) = FI
 130    IJ = IJ + 1
      IF (IN.EQ.1) GO TO 150
      DO 140 I=1,NTHPO
        Y(I) = -Y(I)
 140  CONTINUE
      GO TO 170
 150  DO 160 I=1,NTHPO
        X(I) = X(I)/FN
        Y(I) = Y(I)/FN
 160  CONTINUE
 170  RETURN
      END
```

```
!  R2TX
!  R4TX
```

```
C
C-------------------------------------------------------------------------
C SUBROUTINE:  R2TX
C RADIX 2 ITERATION SUBROUTINE
C-------------------------------------------------------------------------
C
      SUBROUTINE R2TX(NTHPO, CRO, CR1, CI0, CI1)
      DIMENSION CRO(2), CR1(2), CI0(2), CI1(2)
      DO 10 K=1,NTHPO,2
        R1 = CRO(K) + CR1(K)
        CR1(K) = CRO(K) - CR1(K)
        CRO(K) = R1
        FI1 = CI0(K) + CI1(K)
        CI1(K) = CI0(K) - CI1(K)
        CI0(K) = FI1
   10 CONTINUE
      RETURN
      END
C
C-------------------------------------------------------------------------
C SUBRDUTINE:  R4TX
C RADIX 4 ITERATION SUBROUTINE
C-------------------------------------------------------------------------
C
      SUBROUTINE R4TX(NTHPO, CRO, CR1, CR2, CR3, CI0, CI1, CI2, CI3)
      DIMENSION CRO(2), CR1(2), CR2(2), CR3(2), CI0(2), CI1(2), CI2(2),
     *    CI3(2)
      DO 10 K=1,NTHPO,4
        R1 = CRO(K) + CR2(K)
        R2 = CRO(K) - CR2(K)
        R3 = CR1(K) + CR3(K)
        R4 = CR1(K) - CR3(K)
        FI1 = CI0(K) + CI2(K)
        FI2 = CI0(K) - CI2(K)
        FI3 = CI1(K) + CI3(K)
        FI4 = CI1(K) - CI3(K)
        CRO(K) = R1 + R3
        CI0(K) = FI1 + FI3
        CR1(K) = R1 - R3
        CI1(K) = FI1 - FI3
        CR2(K) = R2 - FI4
        CI2(K) = FI2 + R4
        CR3(K) = R2 + FI4
        CI3(K) = FI2 - R4
   10 CONTINUE
      RETURN
      END
```

!

```
C
C-----------------------------------------------------------------------
C SUBROUTINE:  R8TX
C RADIX 8 ITERATION SUBROUTINE
C-----------------------------------------------------------------------
C
      SUBROUTINE R8TX(NXTLT, NTHPO, LENGT, CR0, CR1, CR2, CR3, CR4,
     *    CR5, CR6, CR7, CI0, CI1, CI2, CI3, CI4, CI5, CI6, CI7)
      DIMENSION CR0(2), CR1(2), CR2(2), CR3(2), CR4(2), CR5(2), CR6(2),
     *    CR7(2), CI1(2), CI2(2), CI3(2), CI4(2), CI5(2), CI6(2),
     *    CI7(2), CI0(2)
      COMMON /CON2/ PI2, P7
      SCALE = PI2/FLOAT(LENGT)
      DO 30 J=1,NXTLT
        ARG = FLOAT(J-1)*SCALE
        C1 = COS(ARG)
        S1 = SIN(ARG)
        C2 = C1**2 - S1**2
        S2 = C1*S1 + C1*S1
        C3 = C1*C2 - S1*S2
        S3 = C2*S1 + S2*C1
        C4 = C2**2 - S2**2
        S4 = C2*S2 + C2*S2
        C5 = C2*C3 - S2*S3
        S5 = C3*S2 + S3*C2
        C6 = C3**2 - S3**2
        S6 = C3*S3 + C3*S3
        C7 = C3*C4 - S3*S4
        S7 = C4*S3 + S4*C3
        DO 20 K=J,NTHPO,LENGT
          AR0 = CR0(K) + CR4(K)
          AR1 = CR1(K) + CR5(K)
          AR2 = CR2(K) + CR6(K)
          AR3 = CR3(K) + CR7(K)
          AR4 = CR0(K) - CR4(K)
          AR5 = CR1(K) - CR5(K)
          AR6 = CR2(K) - CR6(K)
          AR7 = CR3(K) - CR7(K)
          AI0 = CI0(K) + CI4(K)
          AI1 = CI1(K) + CI5(K)
          AI2 = CI2(K) + CI6(K)
          AI3 = CI3(K) + CI7(K)
          AI4 = CI0(K) - CI4(K)
          AI5 = CI1(K) - CI5(K)
          AI6 = CI2(K) - CI6(K)
          AI7 = CI3(K) - CI7(K)
          BR0 = AR0 + AR2
          BR1 = AR1 + AR3
          BR2 = AR0 - AR2
          BR3 = AR1 - AR3
          BR4 = AR4 - AI6
          BR5 = AR5 - AI7
          BR6 = AR4 + AI6
          BR7 = AR5 + AI7
```

```
      BI0 = AI0 + AI2
      BI1 = AI1 + AI3
      BI2 = AI0 - AI2
      BI3 = AI1 - AI3
      BI4 = AI4 + AR6
      BI5 = AI5 + AR7
      BI6 = AI4 - AR6
      BI7 = AI5 - AR7
      CR0(K) = BR0 + BR1
      CI0(K) = BI0 + BI1
      IF (J.LE.1) GO TO 10
      CR1(K) = C4*(BR0-BR1) - S4*(BI0-BI1)
      CI1(K) = C4*(BI0-BI1) + S4*(BR0-BR1)
      CR2(K) = C2*(BR2-BI3) - S2*(BI2+BR3)
      CI2(K) = C2*(BI2+BR3) + S2*(BR2-BI3)
      CR3(K) = C6*(BR2+BI3) - S6*(BI2-BR3)
      CI3(K) = C6*(BI2-BR3) + S6*(BR2+BI3)
      TR = P7*(BR5-BI5)
      TI = P7*(BR5+BI5)
      CR4(K) = C1*(BR4+TR) - S1*(BI4+TI)
      CI4(K) = C1*(BI4+TI) + S1*(BR4+TR)
      CR5(K) = C5*(BR4-TR) - S5*(BI4-TI)
      CI5(K) = C5*(BI4-TI) + S5*(BR4-TR)
      TR = -P7*(BR7+BI7)
      TI = P7*(BR7-BI7)
      CR6(K) = C3*(BR6+TR) - S3*(BI6+TI)
      CI6(K) = C3*(BI6+TI) + S3*(BR6+TR)
      CR7(K) = C7*(BR6-TR) - S7*(BI6-TI)
      CI7(K) = C7*(BI6-TI) + S7*(BR6-TR)
      GO TO 20
10    CR1(K) = BR0 - BR1
      CI1(K) = BI0 - BI1
      CR2(K) = BR2 - BI3
      CI2(K) = BI2 + BR3
      CR3(K) = BR2 + BI3
      CI3(K) = BI2 - BR3
      TR = P7*(BR5-BI5)
      TI = P7*(BR5+BI5)
      CR4(K) = BR4 + TR
      CI4(K) = BI4 + TI
      CR5(K) = BR4 - TR
      CI5(K) = BI4 - TI
      TR = -P7*(BR7+BI7)
      TI = P7*(BR7-BI7)
      CR6(K) = BR6 + TR
      CI6(K) = BI6 + TI
      CR7(K) = BR6 - TR
      CI7(K) = BI6 - TI
20    CONTINUE
30  CONTINUE
    RETURN
    END
```

VI. ADDING A NEW NETWORK TYPE

In order to add a new network type to FOCAS, one must first calculate the chain matrix for the new network section as described in section II of this document. The definitions of A, B, C, and D thus derived are coded into a new subroutine, SUBROUTINE NEWNAME, the purpose of which is to compute the chain matrix for this new network subsection type. The subroutine will compute the ABCD parameters for the network at the frequency W based on the subsection's circuit parameters defined in array RLC.

In general, SUBROUTINE NEWNAME must have as a minimum:

```
      SUBROUTINE XXXX(N,W)
      COMPLEX*16 ABCD(500,2,2),ZERO
      DOUBLE PRECISION PI2                ! = 2.*PI
      COMMON/NET/RLC(500,20),ABCD,PI2,ZERO

                |        Programmer Defined
      VAR=RLC(N, ) -----      "
                |             "
                |             "
                |             "
      ABCD(N,1,1)= -------    "
      ABCD(N,1,2)= -------    "
      ABCD(N,2,1)= -------    "
      ABCD(N,2,2)= -------    "
      RETURN
      END
```

The variable N indicates which of the input model's subsections is being analyzed, and the location of its parameters in the RLC array. The variable W is the radian frequency in Gigaradians/second.

The chain matrix for the Nth network section is represented in FOCAS as ABCD(N,2,2), with the following definitions.

```
ABCD(N,1,1) = A
ABCD(N,1,2) = B
ABCD(N,2,1) = C
ABCD(N,2,2) = D .
```

Once this subroutine has been coded, several changes are made to the existing FOCAS code.

A name for the two-terminal-pair network with the first three characters being unique must be defined, e.g. NEW. Next, the input parameters must each be defined by two character names. Present array space will hold up to 20 network parameters. The following changes are now to be made:

1. SUBROUTINE INPUT

a. Add the new Network name (three characters) to the NAME array.

b. Add the two character parameter names to the NDATA array. More than two characters are ignored so the first two must be unique. The parameter names must be on the line number corresponding to the location of Network name in the NAME array. If less than 12 parameters are specified, fill to the left with the last parameter name. This maps the parameter values into the RLC array.

c. Increase NKIND by one.

2. SUBROUTINE PRECALC

a. Perform any preliminary calculations which are desired to manipulate the input data, set default values, or reduce computational overhead during later frequency dependent calculations.

3. SUBROUTINE TRANSFORM

a. Add a call statement to this subroutine for the Network named above. The value of KIND is the same as the location of the three character name in the NAME array in the INPUT subroutine.

4. Recompile all programs and subroutines and relink, including SUBROUTINE NEWNAME. The new version of FOCAS, supporting the new network subsection type is now ready to run.

VI. ENHANCEMENTS

A. Complex Filters & Resonant Circuits

In the present formulation of FOCAS, one may encounter an arithmetic overflow condition:

%SYSTEM-F-FLTOVF_F, arithmetic fault, floating overflow at PC=00306265, PSL=03C00000

when simulating networks in which more than two subsections have poles and/or zeroes at the same frequency. An example of such a simulation model is given below:

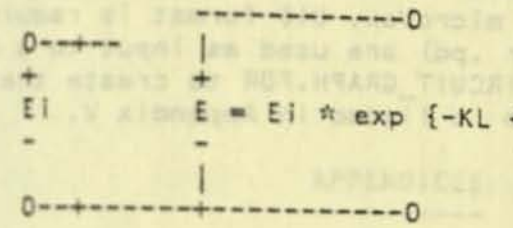
```
GEN TR=.05 PW=.0625 PER=1000 AMP=2 r=1000
series a0=1 a2=25 b1=.16
shunt a1=.012 b0=1 b2=25
series a0=1 a2=25355.98 b1=.0000163
shunt a0=1000 b0=1
.out
.fdg ff
.END TX=6 DC=.5 FS=3.
```

The overflow occurs in subroutine MABCD line 20, where the complex magnitude of each element of the chain matrix is calculated. There are two options to be investigated for overcoming this limitation.

By converting all floating point calculations to the G_FLOATING format the maximum magnitude is increased from 1.7*10**38 to 9*10**307, but the accuracy of double precision is lost. Alternatively one could change the divide by zero checking in all the subsection subroutines (BPF, SERIES, SHUNT, etc) so that any denominator less than 10**-10 (rather than the present 10**-16) is set equal to 10**-10 (as opposed to 10**-16). This change would mean that the response around resonances would be somewhat flattened, but overall accuracy is not altered as much. If it becomes necessary to simualte complex filter sections, both options should be investigated carefully before a choice is made. FOCAS is not intended to support detailed filter design. There are specialized software packages for that purpose.

B. Optical Cable Splices

A splice in the optical fiber can be modeled simply as a section which introduces a loss, L db, and a phase shift, PHI radians. This type of model will neglect the effects of reflections at the near end of the fiber, but will provide a useful approximation to behavior at the far end (beyond the splice). The two-terminal-pair network for such a model would appear as:



$$K = \text{db conversion factor} = \frac{\ln 10}{10} = 0.23026$$

The chain matrix elements for this splice model are:

A = exp {KL - j*PHI} B = 0

C = 0 D = 0

C. Reduced Timing Uncertainty

The uncertainty in voltage crossing times and timing jitter calculations can be reduced in cases in which fewer than 32,768 frequency terms (NF) are calculated by inccreasing the value of NF in subroutine PRECALC to 32,768. However, this change will require substantial changes to the numerical algorithms used in subroutine CROSS to find zeroes and extrema of the output waveforms.

The most serious changes will be required in the extremum computations which presently use empirically determined values to discriminate between real extrema and slight variations in the voltage levels due to the compounding of numerical rounding. It was not felt that such a major overhaul is required at present.

D. Fiber Models

More general fiber models are presented in section III and Appendices II and III of this report.

E. Graphics Support on MicroVax

The standard FOCAS output graphics files with the .tg, .fg, or .pg extensions are in REGIS format. In order to view the output graphically on a microVax, UIS format is required. The data files (.td, .fd, or .pd) are used as input to a separate FORTRAN graphing routine, CIRCUIT_GRAPH.FOR to create the UIS format graphs. This program is listed in Appendix V.

APPENDICES

APPENDIX I. - LED/CABLE REFERENCE DATA FOR FIGURES 1 & 2

LED SPEC'S:

LED #:	1	2	3	4	5	6	7	
FWHM	137	125	130	127	126	114	159	nm
LC	1284	1306	1280	1298	1291	1296	1340	nm

CABLE SPEC'S:

	CABLE 1	CABLE 2	
Length =	2.7	3.38	km
Intermodal Bandwidth	0.82	1.127	GHz-km
Attenuation	1.11	0.74	db/km
LO	1357	1349	nm
SO	9.27*10-5	9.90*10-5	ns/(nm**2)-km

APPENDIX II. FIBER RESPONSE WITH CONSTANT D

Equation (18) is used to define a constant D, to approximate the value of the dispersion coefficient over the source spectrum. The fiber response of equation (31) then becomes:

$$(II\text{-}1)\quad Pe(w) = \frac{e^{-d^2} e^{-Kr_0 X^{Qa}}}{z\sqrt{\pi}} \int_{-\infty}^{\infty} e^{-(L-Lc)^2/z^2 + jwX^{Qc}D(L-Lc) - Kr_w(L-Lc)X^{Qa}}\, dL$$

where $z = FWHM/(2\sqrt{\ln 2})$. (II-1) is simplified by making the substitutions:

$$(II\text{-}2)\quad \alpha = K*r_0*(X**Qa)$$

$$(II\text{-}3)\quad \beta = K*r_w*(X**Qa) - jwD*(X**Qc)$$

$$(II\text{-}4)\quad \lambda = L-Lc$$

$$(II\text{-}5)\quad Pe(w) = \frac{e^{-d^2} e^{-\alpha}}{z\sqrt{\pi}} \int_{-\infty}^{\infty} e^{-\lambda^2/z^2} e^{-\beta\lambda}\, d\lambda$$

Defining $a=1/(z**2)$ and $b=\beta/2$, (II-5) takes the form of the integral (IV-9).

$$(II\text{-}6)\quad Pe(w) = \frac{e^{-d^2} e^{-\alpha}}{z\sqrt{\pi}} \int_{-\infty}^{\infty} e^{-(a\lambda^2 + 2b\lambda)}\, d\lambda$$

$$(II\text{-}7)\quad Pe(w) = \frac{e^{-d^2} e^{-\alpha}}{z\sqrt{\pi}} \sqrt{\pi z^2}\, e^{z^2\beta^2/4}$$

$$(II\text{-}8)\quad Pe(w) = e^{-d^2 - \alpha + z^2\beta^2/4}$$

$$(II\text{-}9)\quad d = 0.1325*w*(X**Qm)/F1$$

$$(II\text{-}10)\quad \alpha = K*r_0*(X**Qa)$$

$$(II\text{-}11)\quad \frac{z^2\beta^2}{4} = \frac{(F**2)}{16(\ln 2)} * \{ K*r_w*(X**Qa) - jwD*(X**Qc) \}^2$$

The FIBER model implemented in FOCAS sets Qa equal to Qc in equation (II-11) as a further restriction.

APPENDIX III. FIBER RESPONSE FUNCTION FOR D=SO*(L-Lc)

Substituting equation (22) into equation (31), the fiber response function becomes:

$$(III\text{-}1)\quad Pe(w) = \frac{e^{-d^2}\, e^{-Kr_0X^{Qa}}}{Z\sqrt{\pi}} \int_{-\infty}^{\infty} e^{-(L-Lc)^2/Z^2 + j\omega X^{Qc} S_0 (L-L_0)(L-Lc) - Kr_\omega (L-Lc)^{\delta}} d$$

where Z = FWHM/(2$\sqrt{\ln 2}$). (III-1) is simplified by substituting:

$$(III\text{-}2)\quad Zo = K*r_0*(X**Qa)$$

$$(III\text{-}3)\quad \lambda = L-Lc; \qquad \lambda = L_0; \qquad \lambda_c = L_c$$

$$(III\text{-}4)\quad Z1 = K*r_\omega*(X**Qa)$$

$$(III\text{-}5)\quad Pe(w) = \frac{e^{-d^2}\, e^{-Z_0}}{Z\sqrt{\pi}} \int_{-\infty}^{\infty} e^{-\lambda^2/Z^2 - Z_1\lambda + j\omega X^{Qc} S_0 \lambda(\lambda+\lambda_c-\lambda_0)} d\lambda$$

Letting I1 be the integral in (III-5), Pe(w) becomes

$$(III\text{-}6)\quad Pe(w) = \frac{\exp\{-(d**2)-Zo\}}{Z*SQRT(PI)} * I1 \quad .$$

The expression for I1 can be simplified by the substitutions:

$$(III\text{-}7)\quad \alpha = jw*(X**Qc)*So$$

$$(III\text{-}8)\quad \delta = \lambda_c - \lambda_0$$

$$(III\text{-}9)\quad I1 = \int_{-\infty}^{\infty} e^{-\lambda^2/Z^2 - Z_1\lambda + \alpha\lambda(\lambda+\delta)} d\lambda = \int_{-\infty}^{\infty} e^{-\lambda^2\left(\frac{1}{Z^2}-\alpha\right) - \lambda(Z_1-\alpha\delta)} d$$

Again substitutions can make the integral more manageable:

$$(III\text{-}10)\quad \beta = [1/(Z**2)] - \alpha = \frac{4(\ln 2)}{(F**2)} - jw*(X**Qc)*So$$

$$(III\text{-}11)\quad \gamma = \frac{Z1-\alpha\delta}{2} = \frac{Kr_\omega*(X**Qa) - jw*(X**Qc)*So*(\lambda_c-\lambda_0)}{2}$$

I1 now has the form (IV-9):

$$(III\text{-}12)\quad I1 = \int_{-\infty}^{\infty} e^{-(k\lambda^2+2l\lambda)}\, d\lambda = \sqrt{\frac{\pi}{k}}\, e^{l^2/k}$$

Making the substitutions:

(III-13) Z2 = So*(X**Qc)*δ

(III-14) Z3 = So*(X**Qc)*(F**2)

(III-15) To = 4*(F**2)*(Z1**2)*(ln 2)

(III-16) T1 = 2*(F**2) * {Z1*Z2*Z3 - 2*(Z1**2)*(ln 2)}

(III-17) T2 = (F**2) * {(Z1**2)*Z3 - 8*Z1*Z2*(ln 2)}

(III-18) T3 = (F**2)*(Z2**2)*Z3

(III-19) T4 = 64*[(ln 2)**2]

(III-20) T5 = 4*(Z3**2)

(III-21) T6 = T2-T3

Equation (III-6) now becomes:

$$(III\text{-}22)\quad Pe(w) = \frac{F*\exp\{-(d**2)-Zo\}}{Z} * \sqrt{\frac{1}{4\ln 2 - jwZ_3}} * \exp\left\{\frac{To+T1*(w**2)+jw*T6}{T4+T5*(w**2)}\right\}$$

with d = 0.01325*(X**Qm)*w/Fl.

Equation (III-22), along with the definitions of the intermediate terms can be used to define a somewhat more generalized fiber cable model than that used in FIBER.

APPENDIX IV. INTEGRALS & MATHEMATICAL IDENTITIES
(Reference 1)

(IV-1) $$\int_{-\infty}^{\infty} e^{-t^2/2}\, dt = \sqrt{2\pi}$$

(IV-2) $$\int_{0}^{z} e^{-t^2/2}\, dt = \frac{\sqrt{\pi}}{2} \operatorname{erf} z$$

(IV-3) $$\int_{0}^{\infty} e^{-(at^2+2bt)}\, dt = \frac{1}{2}\sqrt{\frac{\pi}{a}}\, e^{b^2/a} \operatorname{erfc} b/\sqrt{a}$$

(IV-4) $$\frac{2}{\sqrt{\pi}} \int_{z}^{\infty} e^{-t^2}\, dt \equiv \operatorname{erfc} z = 1 - \operatorname{erf} z$$

(IV-5) $$\operatorname{erf}(-z) = -\operatorname{erf}(z)$$

(IV-6) $$\frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} e^{-(t-m)^2/2\sigma^2}\, dt = \frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{x-m}{\sigma\sqrt{2}}\right)\right]$$

(IV-7) $$\operatorname{erfc}(-z) = 1 + \operatorname{erf}(z)$$

(IV-8) $$\operatorname{erfc}(z) + \operatorname{erfc}(-z) = 2$$

(IV-9) $$\int_{-\infty}^{\infty} e^{-(at^2+2bt)}\, dt = \sqrt{\frac{\pi}{a}}\, e^{b^2/a}$$

APPENDIX V. CIRCUIT_GRAPH.FOR LISTING

```
      program circuit_graph
      ! plot circuit output
      !  8-OCT-1986  Mike Taylor
      !  Modified 3-MARCH-1987  Ann Ewalt
      ! based on an examples in the
      ! MicroVMS Workstation Graphics Programming Guide
      ! Generates a graph of the two input

      implicit integer(a-z)
      external circuit_graph_cmd  ! cld file
      external lib$get_foreign    !
      external cli$dcl_parse, cli$present, cli$get_value

      include 'sys$library:uisentry'
      include 'sys$library:uisusrdef'

      character*4 string
      real array1(350), array2(350)
      real xx(350),yy(350)
      real largest_x, smallest_x, largest_y, smallest_y
      real world_x1, world_x2, world_y1, world_y2
      real scale_x, scale_y
      real x, x2, height, y

      CHARACTER*32 DFILE
      CHARACTER*255 command_line
      CHARACTER*255 verb
      CHARACTER*255 title, x_title, y_title

      ! get the name of the input file from the user.

      status = cli$get_value( '$line', command_line, command_line_size)
      if ( .NOT. status ) call lib$signal( %val( status ))
      ! type *, command_line(1:command_line_size)

      status = cli$get_value( '$verb', verb, verb_size)
      if ( .NOT. status ) call lib$signal( %val( status ))
      ! type *, verb (1:verb_size)

      status = cli$dcl_parse( command_line , circuit_graph_cmd  )
      if ( .NOT. status ) call lib$signal( %val( status ))

      if (cli$present( 'file_name' ) ) then
        call cli$get_value( 'file_name' , DFILE )
      else
89      TYPE 90
        ACCEPT 95,DFILE
      endif
```

```
90      FORMAT(' Enter the graph's data file name: ',$)
95      FORMAT(A32)
99    continue

      If (cli$present( 'title' ) ) then
         call cli$get_value( 'title' , title , title_size )
      else
           title = ' '
           title_size = 1
      endif

      If (cli$present( 'x_title' ) ) then
         call cli$get_value( 'x_title' , x_title , x_title_size )
      else
          x_title =  ' Time '
          x_title_size = 6
      endif

      If (cli$present( 'y_title' ) ) then
         call cli$get_value( 'y_title' , y_title , y_title_size )
      else
           y_title = ' Voltage '
           y_title_size = 9
      endif

      ! the first  col in file is time in nano-seconds or frequency
      ! in MHz  (plot on X)
      ! the second col in file is volts or phase shift in radians,
      ! or normalized power (plot on y)
      ! find the largest and smallest of array

      largest_x = 0.0
      smallest_x = 0.0
      largest_y = 0.0
      smallest_y = 0.0
        MOUT = 1  ! number of graphs
        points = 350

        OPEN( UNIT=21, FILE=DFILE, STATUS='OLD',READONLY)
        DO I=1,Points
           read (21,1000) array1(I), array2(I) ! ann had a 2D array
           IF ( largest_x  .LT. array1(I) ) largest_x = array1(I)
           IF ( smallest_x .GT. array1(I) ) smallest_x = array1(I)
           IF ( largest_y  .LT. array2(I) ) largest_y = array2(I)
           IF ( smallest_y .GT. array2(I) ) smallest_y = array2(I)
        ENDDO
1000    FORMAT(1X,F20.3,<MOUT>(1X,F20.3))    ! <mout>?
        CLOSE(UNIT=21)
```

```
      ! type *, largest_x, smallest_x, largest_y, smallest_y

      ! create the display and window
      vd_id = uisScreate_display( -15.0,-15.0,410.0,350.0,240.0,200.0 )

      ! the first four numbers are the world coordinates
      ! the window size in centimeters

      wd_id = uisScreate_window(vd_id,'SysSworkstation',
     2' Focas Output ')

      ! make the axis 5 times wider than normal
      Call uisSset_line_width( vd_id, 0,16,5.0 )
      Call uisSPlot( vd_id, 16,0,0, 0,330.0 )  ! draw Y axis
      Call uisSPlot( vd_id, 16,0,0, 400.0,0 )  ! draw X axis

      ! the legend of the graph
      Call uisStext( vd_id, 0, title ,
     2 190.0,345.0 )

      ! set the scaling factors
      ! 330 is the coordinat distance
      scale_x = (ABS(largest_x) + ABS(smallest_x)) / 400.0
      scale_y = (ABS(largest_y) + ABS(smallest_y)) / 330.0
      ! type *, scale_x, scale_y

      ! information along the Y axis
      ! Y is always between -1 and 4 (for now!)
      do 20 I = 33,330,33
         y = (float(I) * scale_y ) + smallest_y
         if ( y .GE. 1.0 ) then
              encode (4,11,string) y
         else
              encode (4,10,string) y
         endif
         y = float(I)
10    format (F3.1)
11    format (F3.1)
         Call uisSPlot( vd_id, 16, 0,y,-3.0,y )  ! tick mark

20       Call uisStext( vd_id, 0, string, -14.0, Y ) ! value
      Call uisStext( vd_id, 0, Y_title , -7.0, 340.0 )
            ! information along the X axis
      do 40 I = 40, 400 , 40
         y = (float(I) * scale_x) + smallest_x
         n = y
         encode (4,30,string) n
         y = float(I)
30    format (F10.2)
         Call uisSPlot( vd_id, 16, y,0,y,-3.0 )  ! tick mark
         y = y - 8 ! center the value under the tick
40       Call uisStext( vd_id, 0, string, y, -7.0 ) ! value
```

```
      Call uis$text( vd_id, 0, x_title , 200.0, -12.0 )

      ! convert the input points to plotting coordinates

      do 50 i = 1,Points

         xx(i) = (array1(i) + ABS(smallest_x)) /scale_x
         yy(i) = (array2(i) + ABS(smallest_y)) /scale_y

50    continue

      ! set the line width back to normal
      Call uis$set_line_width( vd_id, 0,16,1.0 )

      do 100 i = 2,Points    ! do the plot

         Call uis$Plot( vd_id, 16, xx(i-1), yy(i-1), xx(i), yy(i) )

100   continue

      pause  ! back to DCL, use continue to retrun to the program
             ! if you run this from inside a TPU subprocess
             ! the pause doesn't work.
      end
```

ACKNOWLEDGEMENTS

FOCAS is based on an early version of SIMER, a network analysis tool developed by Don Marshall in the HPS technology group. Without Don's patience and the running start SIMER provided, FOCAS would not have been completed as quickly or in as useable a form.

Mike Taylor of the SARA Technical Office helped as a general software consultant throughout the project. He made several improvements to the user interface that are impossible to get when restricted to FORTRAN. In addition, he coded the UIS graphics routine for microVax support and served as a sounding board and sanity check for many enhancements and corrections to the code. Mike spent many late nights working on code improvements.

Seymon Ginzburg and Don Knudson (DS A/D) provided on-going sanity checks for the modelling and results and made frequent suggestions for improvements. Bruce Schofield, Hans Rolla, and Bill Gover (P&DS) provided experimental support to validate these checks, and, when necessary, to redirect the efforts.

BIBLIOGRAPHY

1. Abramowitz, Milton & Stegun, Irene, "Handbook of Mathematical Functions", Dover Publications, Inc., New York, N. Y., 1972

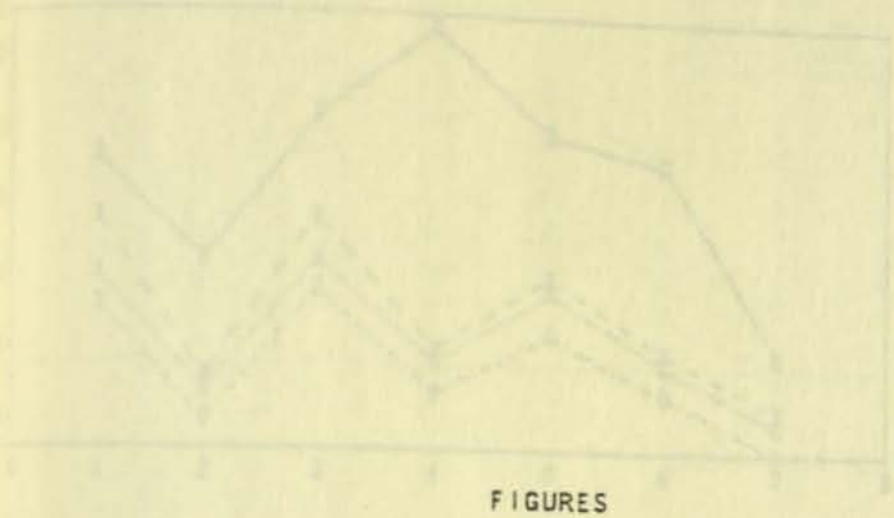
2. Button, Leslie; Love, Walter; & Hawk, Robert, "Chromatic Dispersion in Optical Fibers: Comparison of Measurements with Models", Corning Glass Works, Corning, N. Y.

3. Keiser, Gerd, "Optical Fiber Communications", McGraw-Hill Book Company, 1983

4. Marshall, Don, "SIMER Application Notes", REV 6, 3/20/86

5. Schickentanz, Dieter, notes from private communication with Seymon Ginzburg, Jerry Hutchison, Don Knudson, & Bruce Schofield

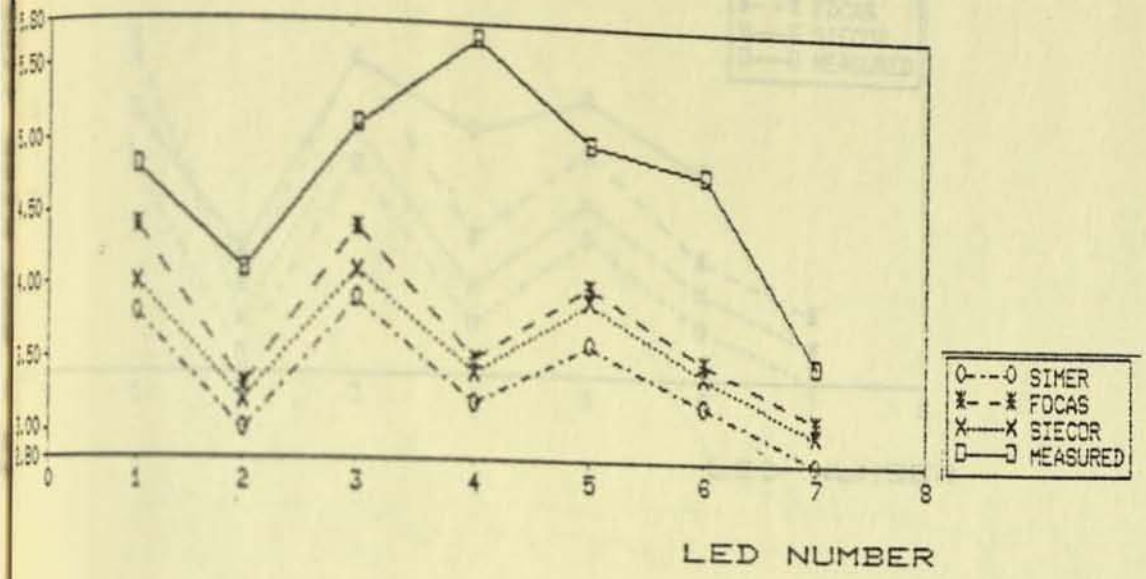
6. Weinberg, Louis, "Network Analysis and Synthesis", Robert E. Kreiger Publishing Co., Inc., Huntington, N.Y., 1975



FIGURES

RISE TIME VS. MODEL

Cable 1



FALL TIME VS. MODEL

Cable 1

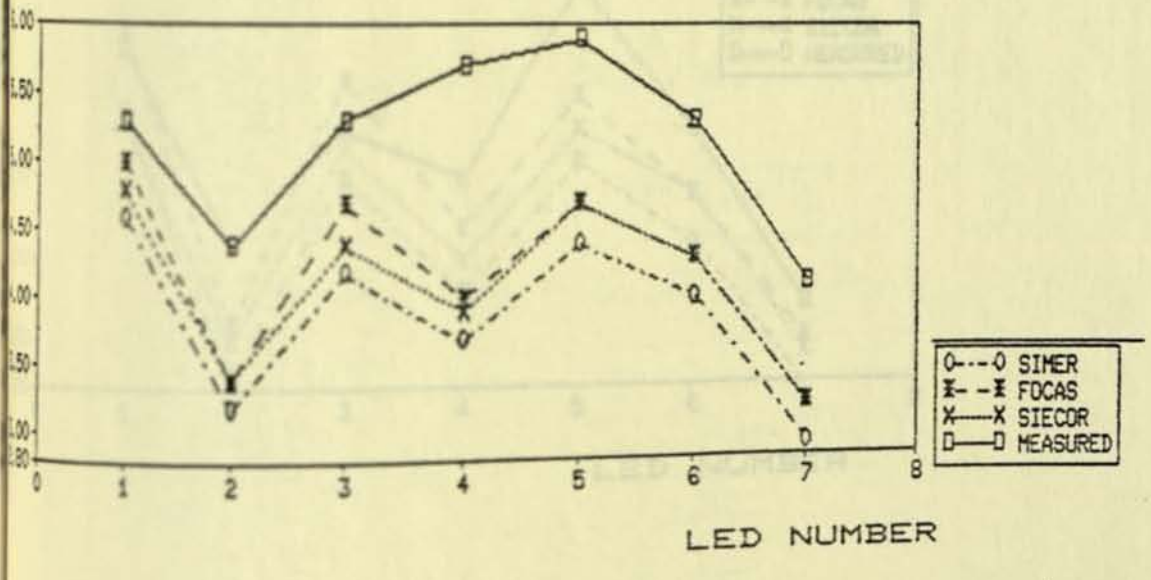
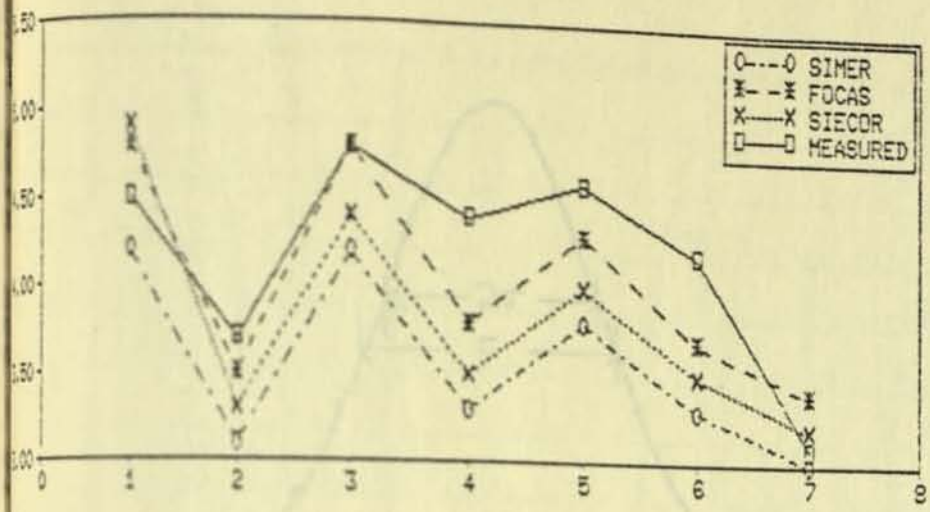


FIGURE 1

RISE TIME VS. MODEL

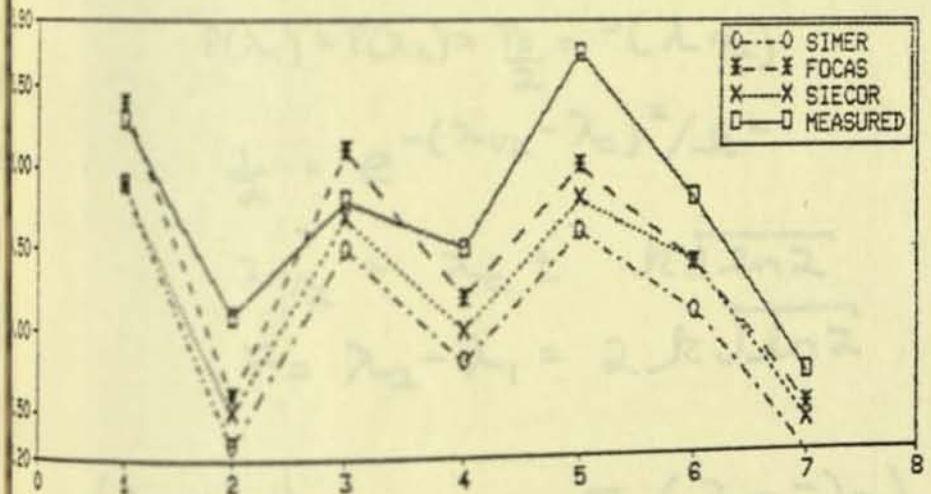
Cable 2



LED NUMBER

FALL TIME VS. MODEL

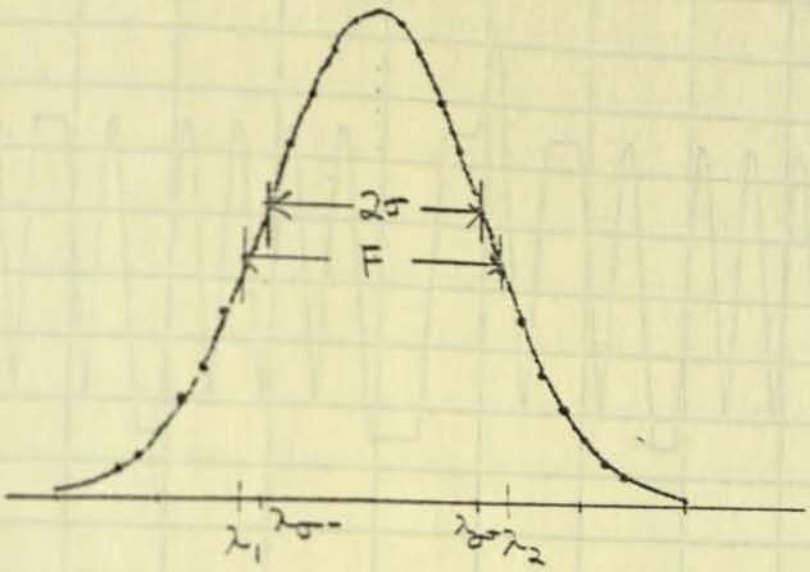
Cable 2



LED NUMBER

FIGURE 2

FIGURE 3. SOURCE GAUSSIAN SPECTRUM



$$P_{LED}(\lambda) = P_0 \, e^{-(\lambda-\lambda_c)^2/k^2} = P_0 \, e^{-(\lambda-\lambda_c)^2/2\sigma^2}$$

λ_c

$$P(\lambda_1) = P(\lambda_2) = \frac{P_0}{2} = P(\lambda_{1/2})$$

$$\frac{1}{2} = e^{-(\lambda_{1/2}-\lambda_c)^2/k^2}$$

$$\lambda_{1/2} = \lambda_c \pm k\sqrt{\ln 2}$$

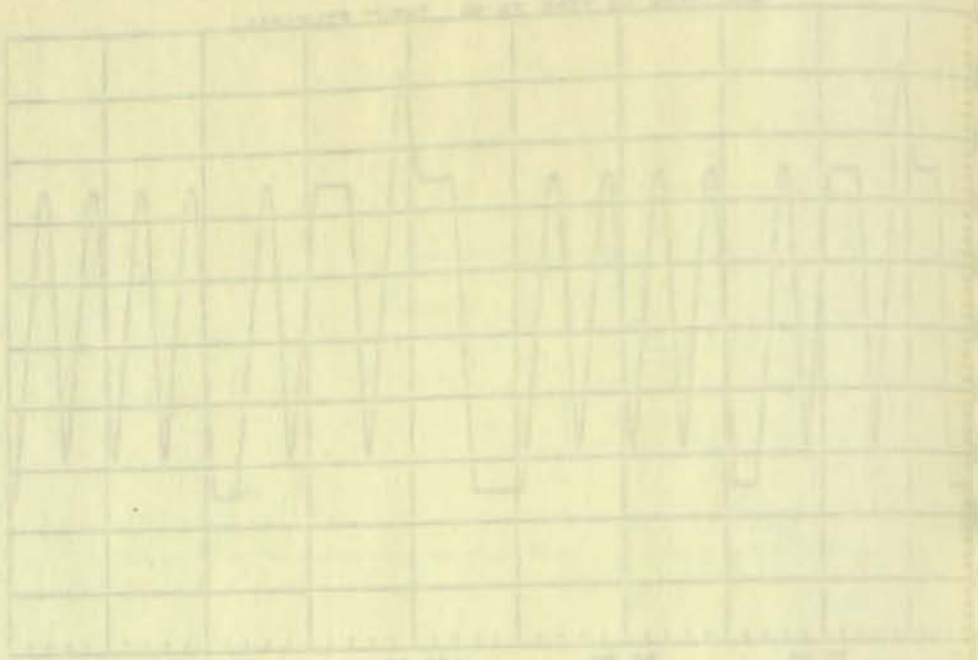
$$F = \lambda_2 - \lambda_1 = 2k\sqrt{\ln 2}$$

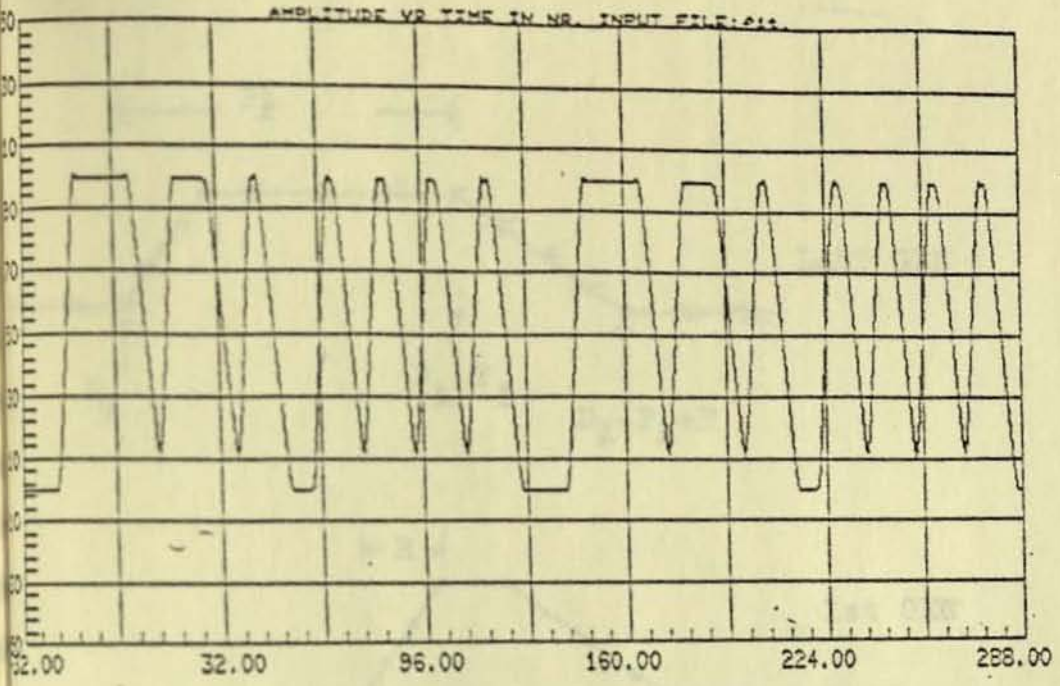
$$\frac{(\lambda_\sigma - \lambda_c)}{\sigma} = \pm 1 = \frac{\sqrt{2}(\lambda_\sigma - \lambda_c)}{k}$$

$$\lambda_\sigma = \lambda_c \pm k\sqrt{2}/2$$

$$2\sigma = \lambda_{\sigma+} - \lambda_{\sigma-} = k\sqrt{2}$$

$$2\sigma = \frac{F}{2\sqrt{\ln 2}}\sqrt{2} = \frac{F}{\sqrt{2\ln 2}} = 0.849F$$





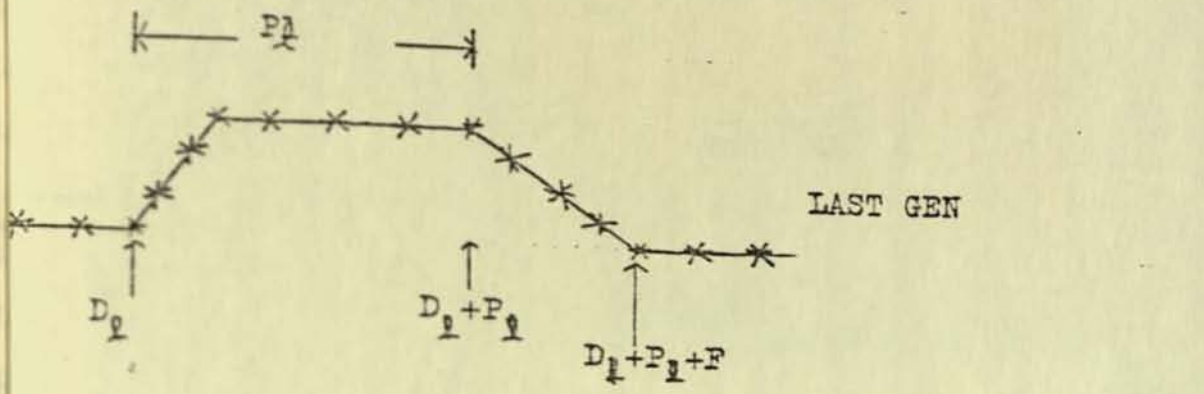
ULATION OF 4 5b PATTERN

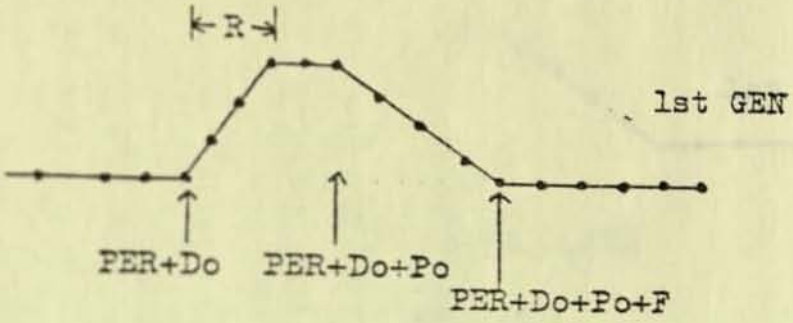
First and last pulse generator overlap to give overshoot
for the electronic case where voltage sources add, but
only result in a single wider pulse for the optical case.

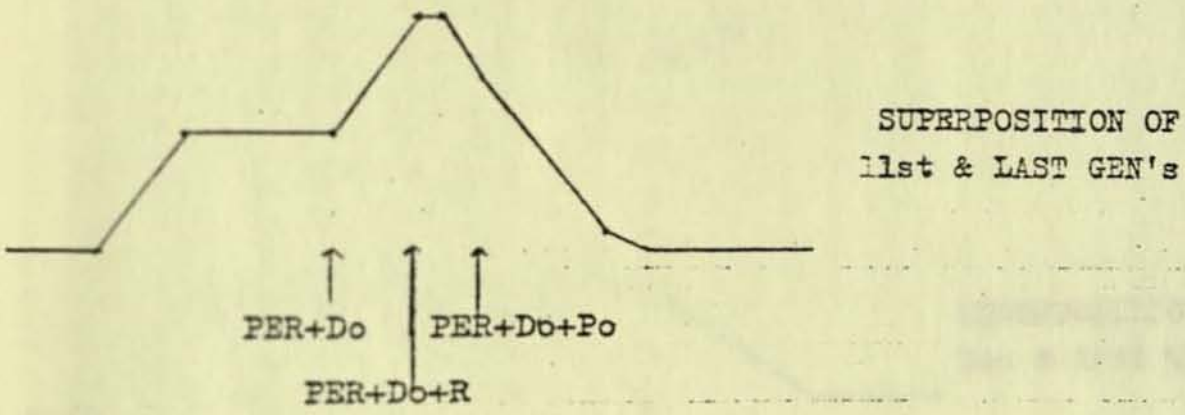
N PW=8 PER=160 TR=4.4 TF=7
N PW=16 DEL=16
N PW=8 DEL=40
N PW=8 DEL=64
N PW=8 DEL=80
N PW=8 DEL=96
N PW=8 DEL=112
N PW=16 DEL=144

TX=320 DC=.5 FS=2

FIGURE 5. CORRECTED 16110 10010 10011







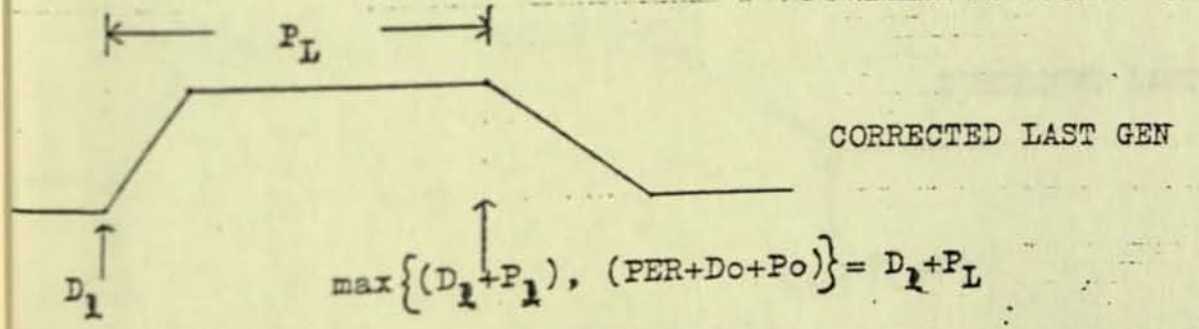
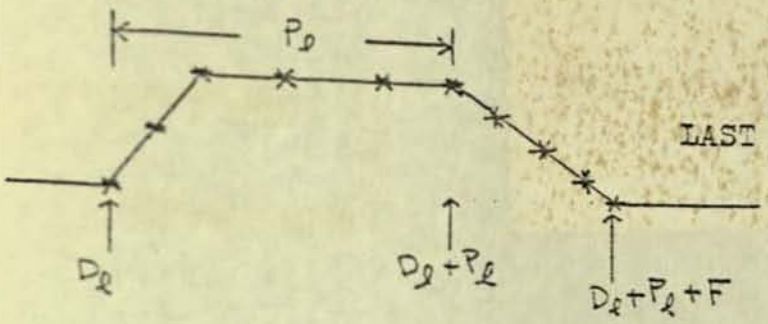
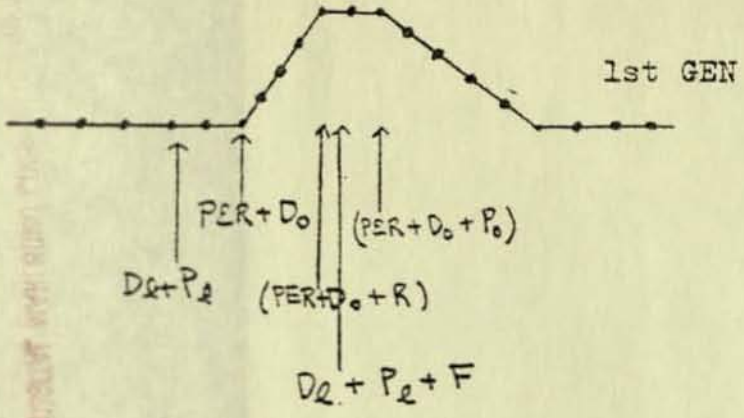
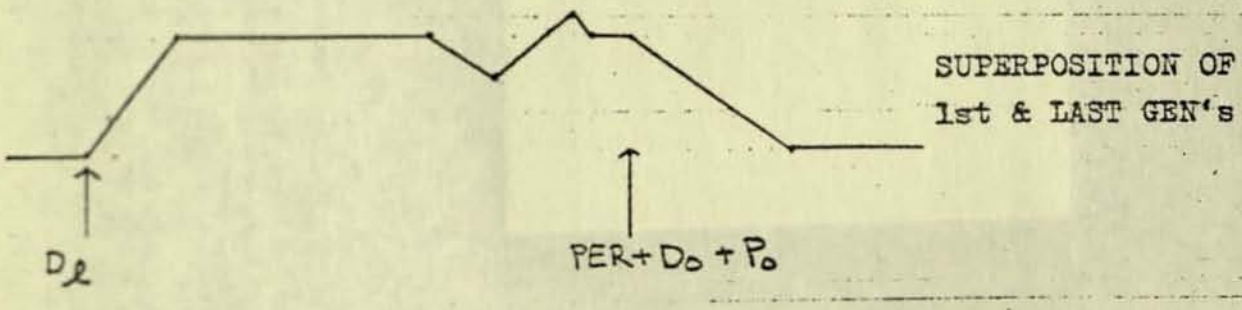


FIGURE 6. OVERLAPPING 1's CORRECTION PER+Do < D_1+P_1







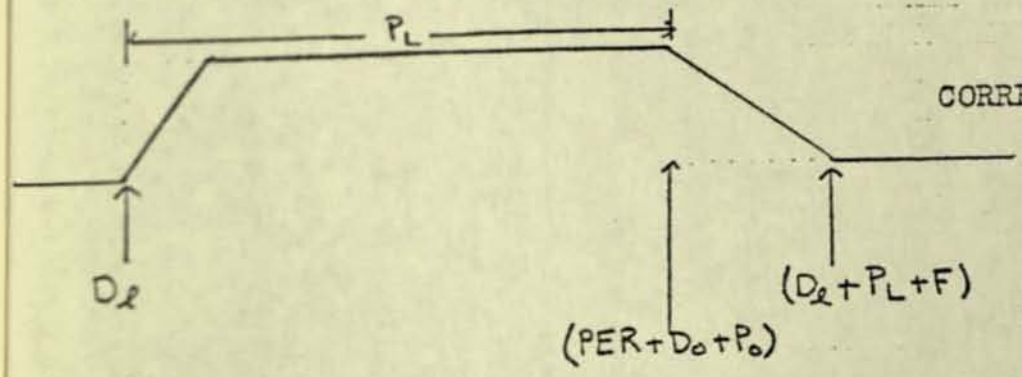


FIGURE 7. OVERLAPPING 1's $(D_\ell + P_\ell < PER + D_o < D_\ell + P_\ell + F) \cap (D_\ell + P_\ell + F > PER + D_o + R)$