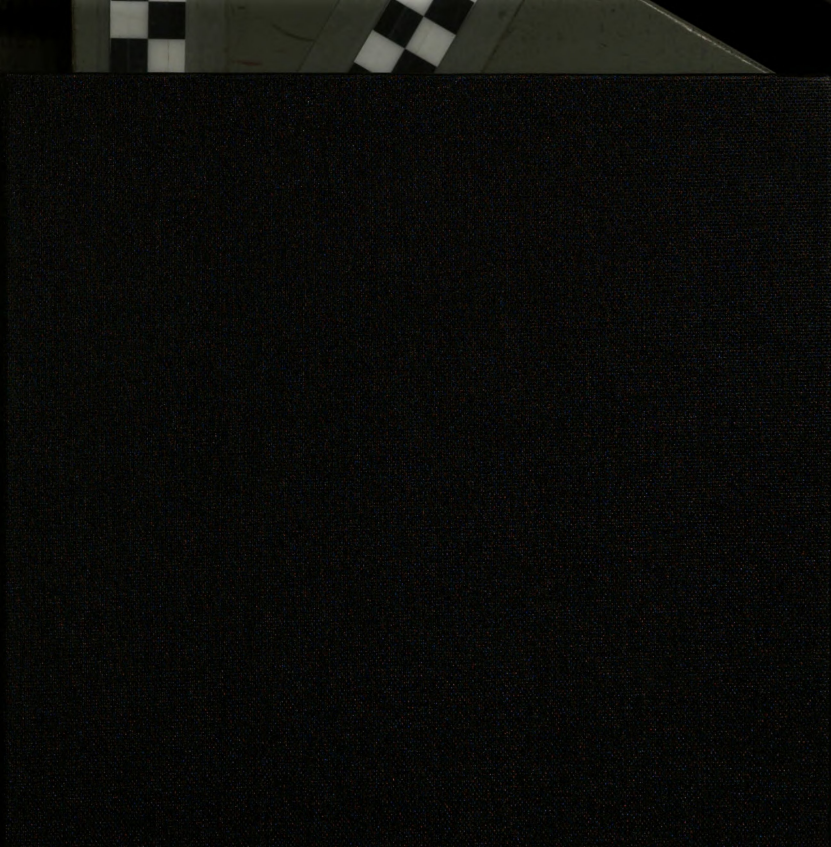




THESIS

2

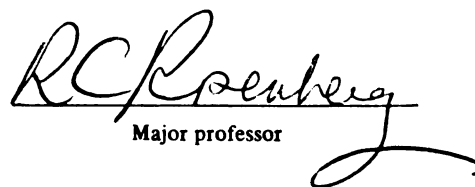


3 1293 01024 0145

This is to certify that the
thesis entitled
IMPROVING VISUALIZATION TECHNIQUES
USING BOND GRAPH METHODS
presented by
JENNIFER LYNN ORDWAY
has been accepted towards fulfillment
of the requirements for
MASTERS degree in MECHANICAL ENGINEERING

Date

April 7, 1995


Major professor



PLACE IN RETURN BOX to remove this checkout from your record.
TO AVOID FINES return on or before date due.

DATE DUE	DATE DUE	DATE DUE
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____

MSU Is An Affirmative Action/Equal Opportunity Institution

c:\circ\datedue.pm3-p.1

**IMPROVING VISUALIZATION TECHNIQUES
USING BOND GRAPH METHODS**

By

Jennifer Lynn Ordway

A THESIS

**Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of**

MASTER OF SCIENCE

in

Mechanical Engineering

Department of Mechanical Engineering

1995

ABSTRACT

IMPROVING VISUALIZATION TECHNIQUES USING BOND GRAPH METHODS

By

Jennifer Lynn Ordway

Interpretation of simulation results becomes more difficult as the complexity of a system increases. Current methods of viewing results usually involve plots of variables versus time. When investigating the behavior of a system as a whole, time plots can become confusing. In these cases, scientific visualization of the system behavior becomes a matter of great importance.

This thesis presents a significantly improved visualization method that allows engineers to view results and connect them to a graphical model. The design is based on the bond graph as a modeling representation of physical systems and the PC-DOS version of ENPORT as a simulation tool. In order to aid in the comprehension of the system's dynamic behavior, power variable results generated in ENPORT are displayed on the corresponding bond graph as the bond color or width.

Two systems, a simple mechanical oscillator and a flatbed trailer suspension system were modeled and simulated to demonstrate the benefits of a dynamic display when attempting to understand physical behaviors.

Dedication

In remembrance of my Grandpa,

Jack D. Ordway,

whom I love with all my heart.

ACKNOWLEDGMENTS

I would like to express my sincere thanks and gratitude to my major professor, Dr. Ronald Rosenberg, for his support and guidance throughout the development of this work.

In addition, I would like to thank the members of my thesis committee: Drs. Alejandro Diaz and Clark Radcliffe. Their encouragement and input were very valuable to me.

My gratitude also extends to the members of the ENPORT development group who have been great sources of knowledge and a pleasure to work with.

Special thanks go to all of my family and friends, especially my parents; Crystal and Gerald Walmsley, and James and Judith Ordway. Their love, support, and guidance are the most precious gifts they could give me.

Above all, thanks to God, who helps me enjoy happy times and pulls me through rough ones.

TABLE OF CONTENTS

List of Tables	vii
List of Figures	viii
Nomenclature	x
Chapter 1 - Introduction	1
1.1 Problem Definition	1
1.2 Research Objectives	3
1.3 Description of Thesis Organization.	4
Chapter 2 - Visualization Examples Using the Dynamic Display Module	6
2.1 Mechanical Oscillator Modeling Example	6
2.2 Automobile Modeling Example.	12
Chapter 3 - Implementation of the Dynamic Display Module	23
3.1 File Importing.	23
3.2 Dynamic Display Module Features Implemented	25
Chapter 4 - Conclusions	36
4.1 Summary of Contributions	36
4.2 Areas for Further Research	37
Appendix A - User's Manual	40

Appendix B - Program Documentation and Code.	50
Appendix C - Example Model Files.	156
References	161

LIST OF TABLES

Table 1. Mechanical Oscillator Model Equations	7
Table 2. Car Suspension Model Equations	15

LIST OF FIGURES

Figure 1. Schematic of a Mechanical Oscillator	6
Figure 2. Bond Graph of Mechanical Oscillator	7
Figure 3. Efforts of Bonds for Mechanical Oscillator	8
Figure 4. Flows of Bonds for Mechanical Oscillator	9
Figure 5. Dynamic Display of Mechanical Oscillator at Initial Time	10
Figure 6. Dynamic Display of Mechanical Oscillator Power	11
Figure 7. Schematic of Flatbed Trailer Suspension System	13
Figure 8. Bond Graph Representation of Flatbed Trailer Suspension	14
Figure 9. Road and Corresponding Vertical Velocity Profile	16
Figure 10. Forces on the Front and Rear Tires	17
Figure 11. Power Variable Response for Flatbed Trailer (Time=0.10 seconds) .	18
Figure 12. Power Variable Response for Flatbed Trailer (Time=0.18 seconds) .	18
Figure 13. Power Variable Response for Flatbed Trailer (Time=0.21 seconds) .	19
Figure 14. Power Variable Response for Flatbed Trailer (Time=0.25 seconds) .	19
Figure 15. Power Variable Response for Flatbed Trailer (Time=0.35 seconds) .	20
Figure 16. Power Variable Response for Flatbed Trailer (Time=0.45 seconds) .	20
Figure 17. Power Variable Response for Flatbed Trailer (Time=0.55 seconds) .	21
Figure 18. Width Legends Available in the Dynamic Display	29

Figure 19. Color Legends Available in the Dynamic Display	30
Figure 20. Time Legend for the Dynamic Display	30
Figure 21. Dynamic Display Module Graphical Display	31
Figure 22. Examples of Bonds of Varying Widths	33
Figure 23. Examples of Bonds with Interior Points	33
Figure A-1. Main Menu for Dynamic Display	40
Figure A-2. Model File Options Menu	41
Figure A-3. Results File Options Menu	42
Figure A-4. Dynamic Display Configuration Window	44
Figure A-5. Initial Screen of the Dynamic Display	47

NOMENCLATURE

C - Capacitance

E - Effort

F - Flow

F - Force

I - Inductance

P - Momentum

Q - Displacement

t - Time

V - Velocity

X - Horizontal Displacement

Y - Vertical Displacement

Chapter 1

Introduction

1.1 Problem Definition

1.1.1 Importance of Simulations and Visualization

In the past, engineers have analyzed and optimized designs through iterative processes or trial and error. Often, engineers will overdesign in order to reduce the time it takes to test different designs. In some situations, these methods are practical and cost effective. However, in areas where tolerances are low and design specifications are stringent, simulations allow the engineer to test designs theoretically to determine effectiveness without spending money to build the actual design.

Simulations are essential to engineering work in order to reduce costs and increase efficiency. In cases where there are opposing design criteria, such as reducing weight while maintaining strength, it would be costly and time consuming to build many models and test them for strength and weight. Simulations provide an engineer with a way to see what designs will most likely be best without the expense and time required to do trial and error testing.

The simulation results will only be valid if the model is valid. Therefore, modeling is the most important aspect of any simulation. Many computer programs have been designed that will perform simulations by numerical integration. These programs allow engineers to focus their energy into creating valid models instead of creating ways to simulate these systems.

Once a simulation has been performed, the engineer has an abundance of information to analyze. In an attempt to understand the physical behavior of systems, various visualization techniques can be used to interpret the results. The most common visualization technique used is viewing of plots of variables versus time. These prove most useful when the behavior of a particular variable or subset of variables is under investigation. However, in large systems, the number of variables of interest may be too large to display with any meaning on a time plot. Some other method of visualization would be helpful.

Due to the complexity of designs in such fields as automotive and aerospace, most engineering designs consist of components in several power domains. For example, automobile engines generally have mechanical, rotational, electrical, and hydraulic components. Bond graphs are very well suited to systems that include more than one power domain. Another advantage of using bond graphs for modeling is that the system equations can be obtained directly from the graphical model. Computer software is available to create solutions to bond graph

models given system parameters. One such program is ENPORT. This program allows the engineer to create a two-dimensional bond graph model, numerically solve the system equations, and view the results.

One method of visualization using bond graph models is to view the magnitude of a variable, such as effort or flow, being displayed as the corresponding bond's width and/or color. This method was successfully implemented in ENPORT in the UNIX environment (Ermer 1994) for both two-dimensional and three-dimensional bond graphs.

1.2 Research Objectives

The goal of this research was to implement a system that would enhance the ENPORT PC-DOS environment by providing an additional tool for visualizing the behavior of a physical system. Once the user has created a bond graph model and enters the system parameters, ENPORT generates the system equations, then solves them by numerical integration over a desired time interval. Currently, ENPORT allows the user to output data in a table format or with plots of variables versus time. The new Dynamic Display module will allow for a third kind of output that enables visualization of the dynamic behavior of the system by tying together the bond graph model and the results obtained from simulations. A major

goal of this research is to allow an engineer to visualize how the power, effort, or flow moves about the system during a given time.

To meet this objective, the current ENPORT Model Builder module was altered to provide a power-related animation directly tied to the system bond graph structure. The Dynamic Display module imports a model file and the corresponding results file from the Model Builder and Solver modules of the PC-DOS version of ENPORT. The animation consists of varying the color and width of the bonds in a two dimensional bond graph model of a system based on the user defined power variable of effort, flow, or power. This research was preceded by line-width variation in ENPORT to display time-averaged power variables on the system graph (Rosenberg and Zhou 1988) and development of a dynamic display of time-varying quantities on the system graph in the UNIX version of ENPORT (Ermer 1994).

1.3 Description of Thesis Organization

This thesis consists of an introduction to the problem, examples of the results achieved, presentation of the implemented design, and conclusions concerning this implementation. The current chapter presents the problem and the motivation for the research as well as the order of presentation of the other sections of this paper. Chapter 2 illustrates the concept of dynamic display and its

application to the solution and visualization of practical problems. Chapter 3 includes a complete discussion of the Dynamic Display module and the features implemented in response to the problem statement. Finally, Chapter 4 presents a summary of what was accomplished and areas of further research. Appendices are included which contain more detailed information about the examples and the computer program that was used to implement this research. Appendix A contains a user's manual for operation of the Dynamic Display module in the PC-DOS version of ENPORT. Appendix B contains the code listings for the software used to create the Dynamic Display module. The ENPORT model files for the examples are available in Appendix C. A list of references used to carry out this research is also included in this document.

Chapter 2

Visualization Examples Using the Dynamic Display Module

2.1 Mechanical Oscillator Modeling Example

In engineering systems, a common component of models of physical systems is a simple mechanical oscillator. To get a basic understanding of how a dynamic display can be useful, the mechanical oscillator as shown in Figure 1 was studied. The system consists of a mass attached to ground with a spring and damper in parallel. This example, while simple, will provide easily visualized dynamic system behavior.

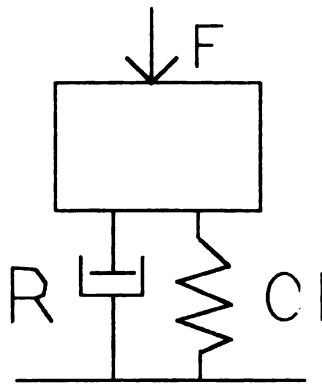


Figure 1. Schematic of a Mechanical Oscillator

A bond graph of the system was constructed in the Enport Model Builder module as shown in Figure 2. The physical components and their corresponding nodes are listed in Table 1. The model file for this system, “Mechosc.enp”, is located in Appendix C and includes the model description and parameters as well as the equations generated by the Model Builder module.

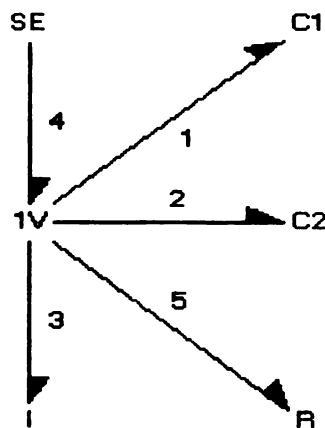


Figure 2. Bond Graph of Mechanical Oscillator

Table 1. Mechanical Oscillator Model Equations

Physical Component	Node Name	Node Type	Node Equation
Velocity of mass	1V	1 Junction	$e1 + e2 + e3 + e5 = e4$
Applied Force	SE	Source of Effort	$e4 = 19.62$
Spring	C1	Capacitive Element	$e1 = 5 \cdot q1$
Dashpot	R	Resistive Element	$e5 = 3 \cdot f5$
Inertia of the Mass	I	Inertial Element	$f3 = p3/2$
Backlash	C2	Capacitive Element	$e2 = \Phi_b \cdot q2$; backlash

The system equations were solved in the Solver module and results were obtained and stored in the results file, "Mechosc.bdt". The system was numerically integrated by the Runge-Kutta integrator from 0 to 10 seconds with an accuracy of 0.1 seconds. The initial conditions were set to zero and all the variables were saved.

Time plots of the results can be seen in Figures 3 and 4, which show the efforts and flows of the bonds over the time record. From these figures, it can be seen that the system has damped oscillations. The two capacitive bonds have efforts that are in phase, but are out of phase with the effort of the inertial and resistive bonds.

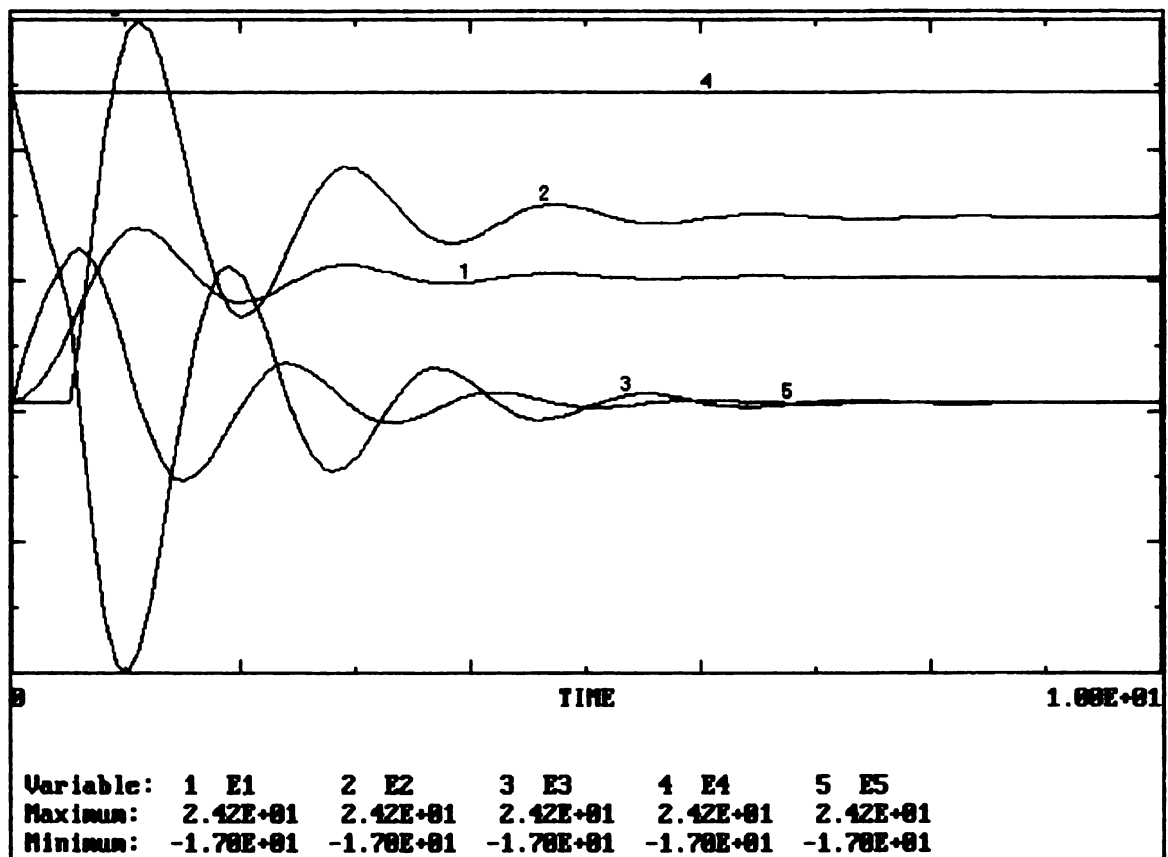


Figure 3. Efforts of Bonds for Mechanical Oscillator

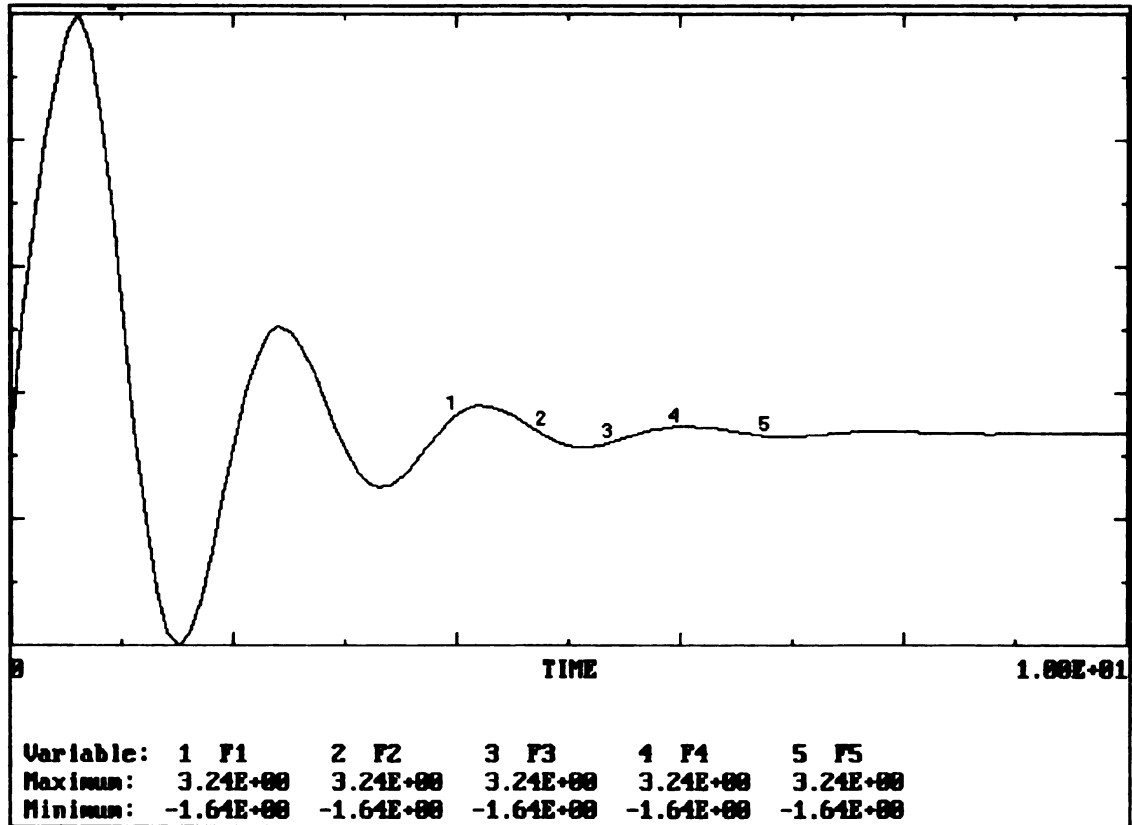


Figure 4. Flows of Bonds for Mechanical Oscillator

A dynamic display was created from the model file and the results file. The dynamic display at the initial time is shown in Figure 5. The width and color legends show that the variable being displayed on the bonds is power. The maximum and minimum values of the power during the time record are also shown in these legends. The time legend shows the current time and the time step size between results data.

Several frames of the dynamic display at varying times are shown in Figure 6 . From these figures, visualization of the transfer of system power between components is possible.

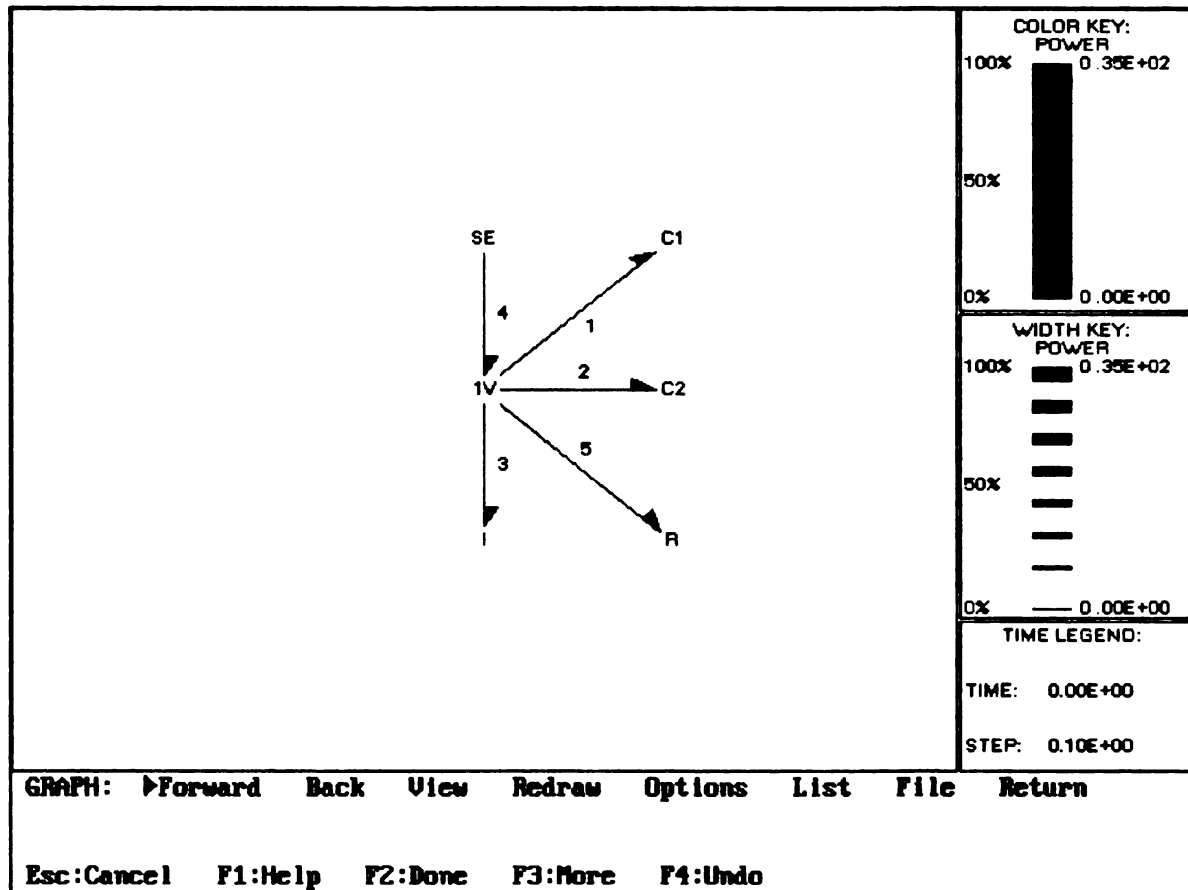


Figure 5. Dynamic Display of Mechanical Oscillator at Initial Time

Initially, the system power begins transferring from the applied force to the inertial element. The system power can be seen being dissipated by the damper after about 0.2 seconds. Power begins transferring to the conductive elements as it flows out of the inertial element indicating that the springs are being compressed after about 0.7 seconds. At approximately 1.1 seconds, the system appears to have no power. The conductive elements begin stretching back at about 1.3 seconds. Eventually, all the oscillations will be damped out.

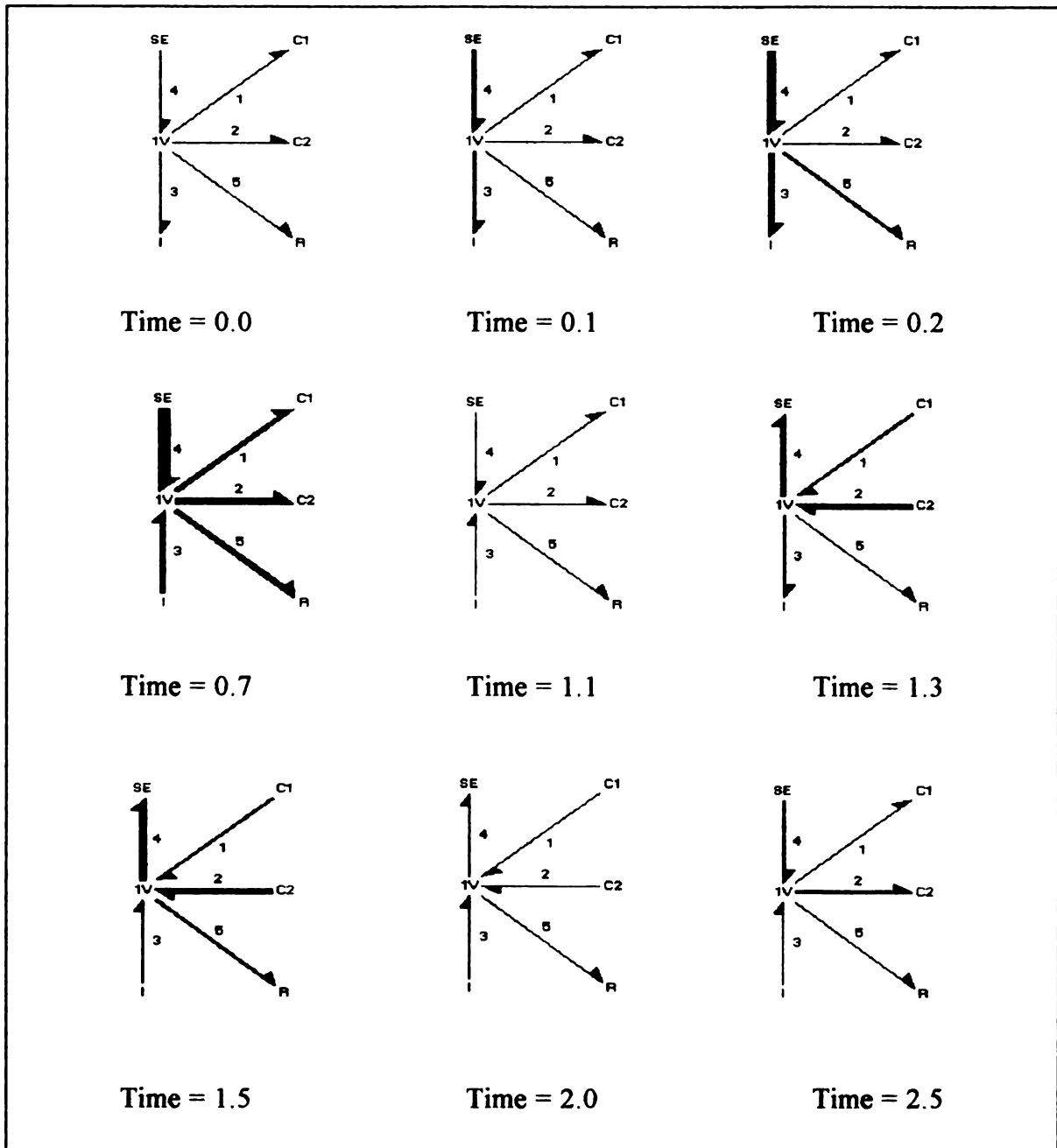


Figure 6. Dynamic Display of Mechanical Oscillator Power

From this example, the usefulness of the dynamic display becomes clear. The dynamics of the system can be easily viewed and visualized from this type of display. Time plots are somewhat abstract since the only thing that connects them to the

system is their labels. The dynamic display, by nature, clearly illustrated the transfer of power throughout the system.

2.2 Flatbed Trailer Suspension System Model

In the automotive industry, design of adequate suspension systems that will damp road vibrations is essential. A simulation of a simple flatbed trailer suspension system encountering a hill was performed in order to demonstrate the use of the Dynamic Display module on systems larger than the mechanical oscillator.

Figure 7 shows a sketch of a flatbed trailer suspension system model. The flatbed trailer body is modeled as a 4 meter long slender uniform body and the suspension systems are located 0.5 meters from each end. This two-dimensional model assumes that the motion of the flatbed trailer is limited to small angles of rotation and that the tires never lose contact with the ground.

A bond graph of the model was constructed using the Model Builder module and can be seen in Figure 8. For this model, the mass was assumed to be evenly distributed across the trailer. The ENPORT model file, "Flatbed.enp", for this example is located in Appendix C along with the equilibrium initial conditions used.

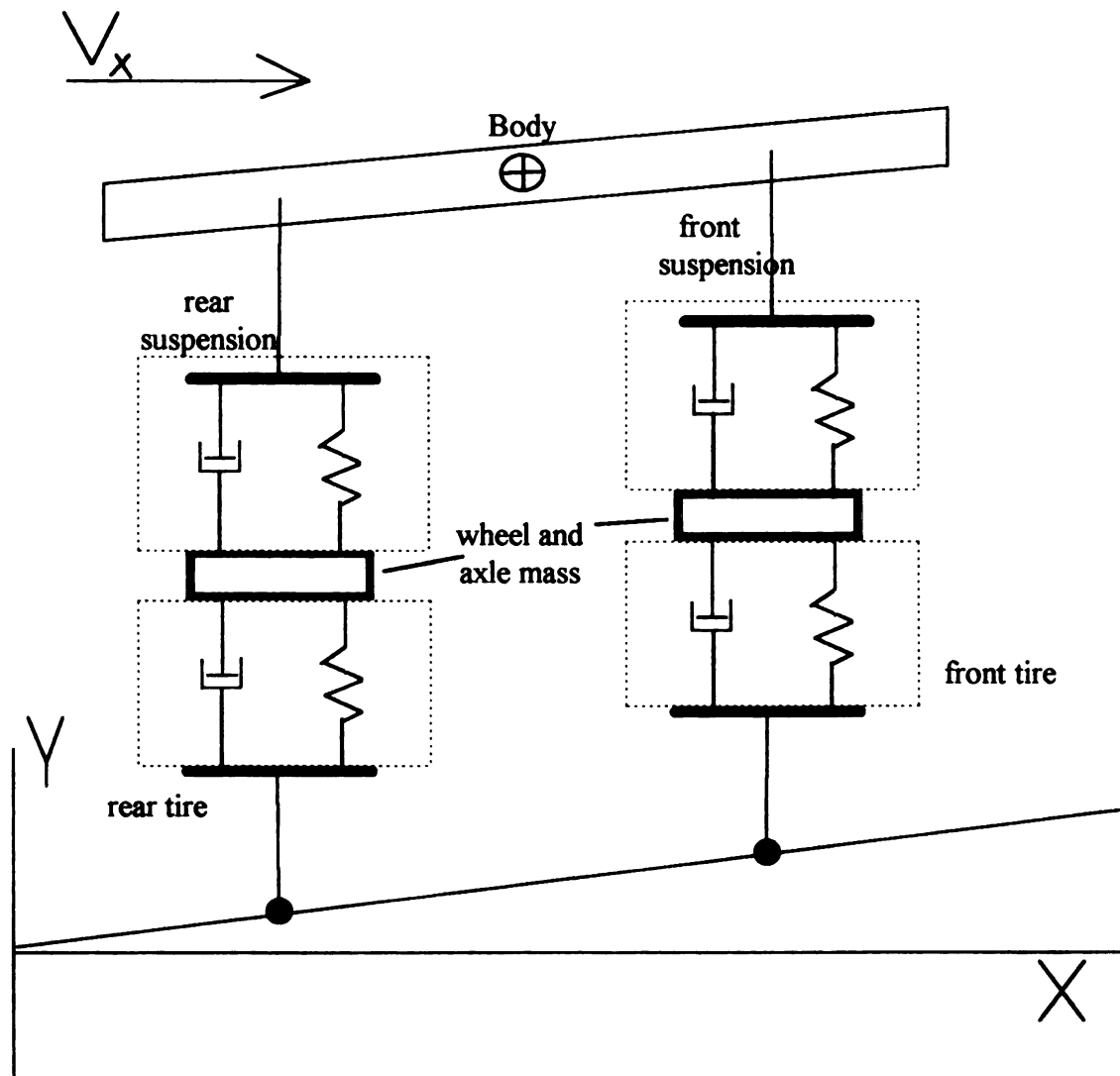


Figure 7. Schematic of Flatbed Trailer Suspension System

Figure 8. Bond Graph Representation of Flatbed Trailer Suspension

The bond graph model was subjected to a road profile consisting of a small hill. The hill was modeled as a ramp function that was 4 meters wide and 0.2 meters high. The vertical velocity due to the ramp was modeled by a pulse source of flow connected to the front and rear tires. A diagram of the road profile and the corresponding source of flow can be seen in Figure 9. The velocity was input to the front tire at the time the ramp was encountered. The same velocity was input to the rear tire after a time delay which is dependent on the horizontal velocity of the flatbed trailer. The delay was calculated for a constant horizontal velocity of 100 km/hr, a typical highway speed. The specific values for the delay and the duration of the pulse are available in Table 2 as well as a summary of the equations used for this simulation.

Table 2. Flatbed Trailer Body and Suspension Model Equations

Node Name	Node Type	Node Equation	Parameter
IM	Inertial Element	$F2 = P2M / MVEH$	$MVEH = 600 \text{ kg}$
SEW	Source of Effort	$E1M = WVEH$	$WVEH = -5886 \text{ N}$
IROT	Inertial Element	$FR1 = PR1/JVEH$	$JVEH = 800 \text{ kg m}^2$
TFR	Transformer	$F13R = DR * F14R$ $E14R = DR * E13R$	$DR = 1.5$
TFF	Transformer	$F12F = DF * F13F$ $E13F = DF * E12F$	$DF = 1.5$
CSF	Capacitive Element	$E11F = KSUS * Q11F$	$KSUS = 10000 \text{ N/m}$
RSF	Resistive Element	$E10F = DSUS * F10F$	$DSUS = 1700 \text{ Ns/m}$
CSR	Capacitive Element	$E11R = KSUS * Q11R$	$KSUS = 10000 \text{ N/m}$
RSR	Resistive Element	$E10R = DSUS * F10R$	$DSUS = 1700 \text{ Ns/m}$
IWHR	Inertial Element	$F6R = P6R / MWH$	$MWH = 20 \text{ kg}$
IWHF	Inertial Element	$F6F = P6F / MWH$	$MWH = 20 \text{ kg}$
SEWR	Source of Effort	$E7R = WWT$	$WWT = -196.2 \text{ N}$
SEWF	Source of Effort	$E7F = WWT$	$WWT = -196.2 \text{ N}$
CTR	Capacitive Element	$E4R = KTIR * Q4R$	$KTIR = 180000 \text{ N/m}$
CTF	Capacitive Element	$E4F = KTIR * Q4F$	$KTIR = 180000 \text{ N/m}$
RTR	Resistive Element	$E3F = DTIR * F3F$	$DTIR = 6000 \text{ Ns/m}$
RTF	Resistive Element	$E3R = DTIR * F3R$	$DTIR = 6000 \text{ Ns/m}$
SFR	Source of Flow	$F1R = \text{SLOPE};$ $F1R = 0; t < \text{DEL3}$ $F1R = 0; \text{DEL4} < t$	$\text{SLOPE} = 0.05$ $\text{DEL3} = 0.208435 \text{ sec.}$ $\text{DEL4} = 0.34458 \text{ sec.}$
SFF	Source of Flow	$F1F = \text{SLOPE};$ $F1F = 0; t < \text{DEL1}$ $F1F = 0; \text{DEL2} < t$	$\text{DEL1} = 0.1000 \text{ sec.}$ $\text{DEL2} = 0.24458 \text{ sec.}$ $\text{SLOPE} = 0.05$

The sorted equations were loaded from the model file, “Flatbed.enp, into the Solver module. A results file, “Flatbedxx.bdt”, was created that included results for all the variables for times from 0.08 seconds to 1.0 seconds. To show that the assumption that the tire never loses contact with the ground is correct for this simulation, a time plot of the force in the tires was made as shown in Figure 10. This

shows that the forces in the tires are positive for the time period of the simulation, therefore, the tires never lose contact with the ground.

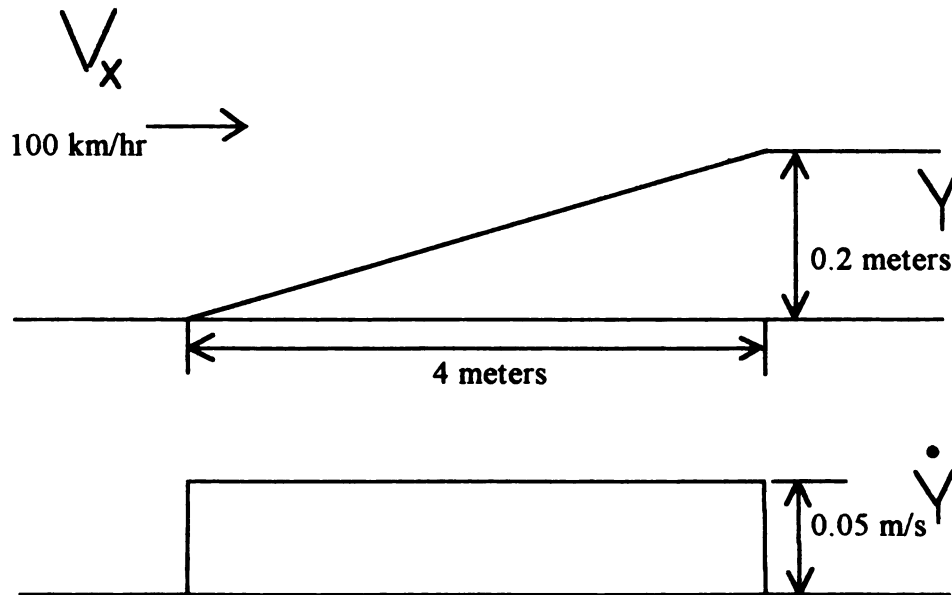


Figure 9. Road and Corresponding Vertical Velocity Profile

To create a dynamic display of power of the system, the model and the corresponding results file were loaded into the Dynamic Display module. Figure 11 shows the power of the system when the front tire initially comes in contact with the hill. The power can be seen moving through the suspension system and up into the body of the trailer in Figure 12. In Figures 13 and 14, the color was varied as well as the width. Both tires are in contact with the hill in Figure 13. The front tire has just reached the top of the hill in Figure 14 and a large power transfer from the spring of

the front tire corresponding to the release of some of the power stored in the spring. Figure 15 shows the system right after the rear tire has reached the top of the hill. The power in the rear of the trailer is concentrated in the tire while the power in the front of the trailer is concentrated in the suspension system. As the system begins to stabilize, as shown in Figure 16, the power is shown in the front and rear suspension systems as well as in the body of the vehicle. Figure 17 shows that as the system nears steady state, the rotational power diminishes.

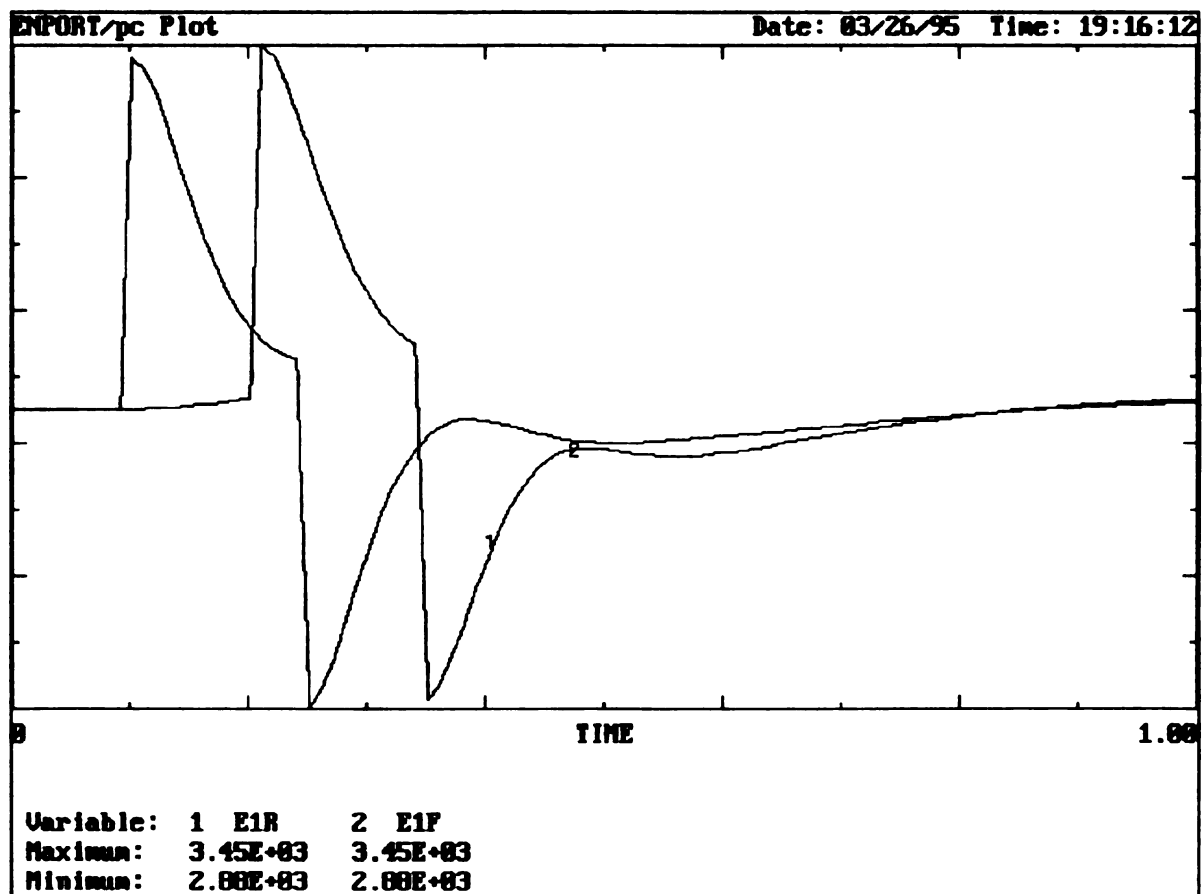


Figure 10. Forces on the Front and Rear Tires

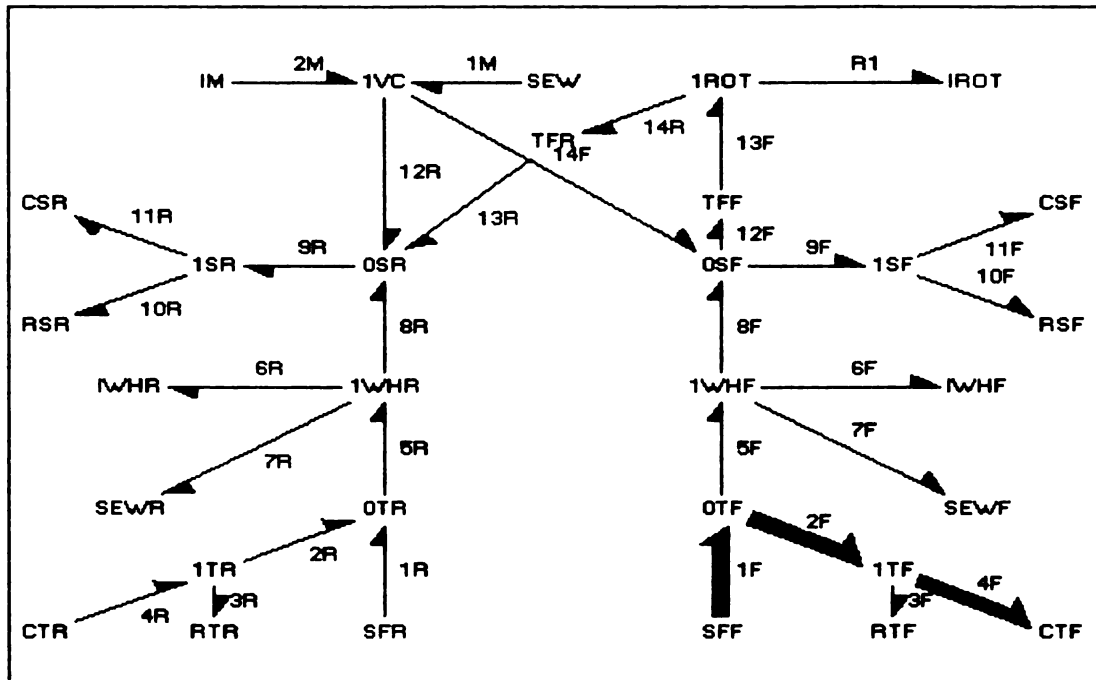


Figure 11. Power Variable Response for Flatbed Trailer (Time = 0.10 seconds)

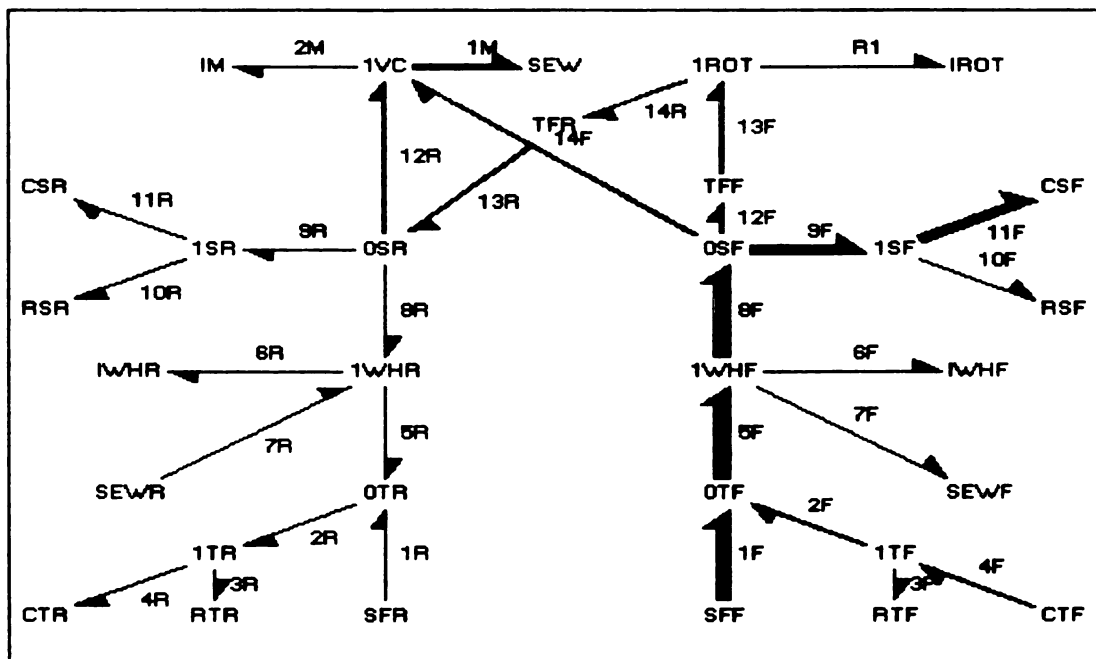


Figure 12. Power Variable Response for Flatbed Trailer (Time = 0.18 seconds)

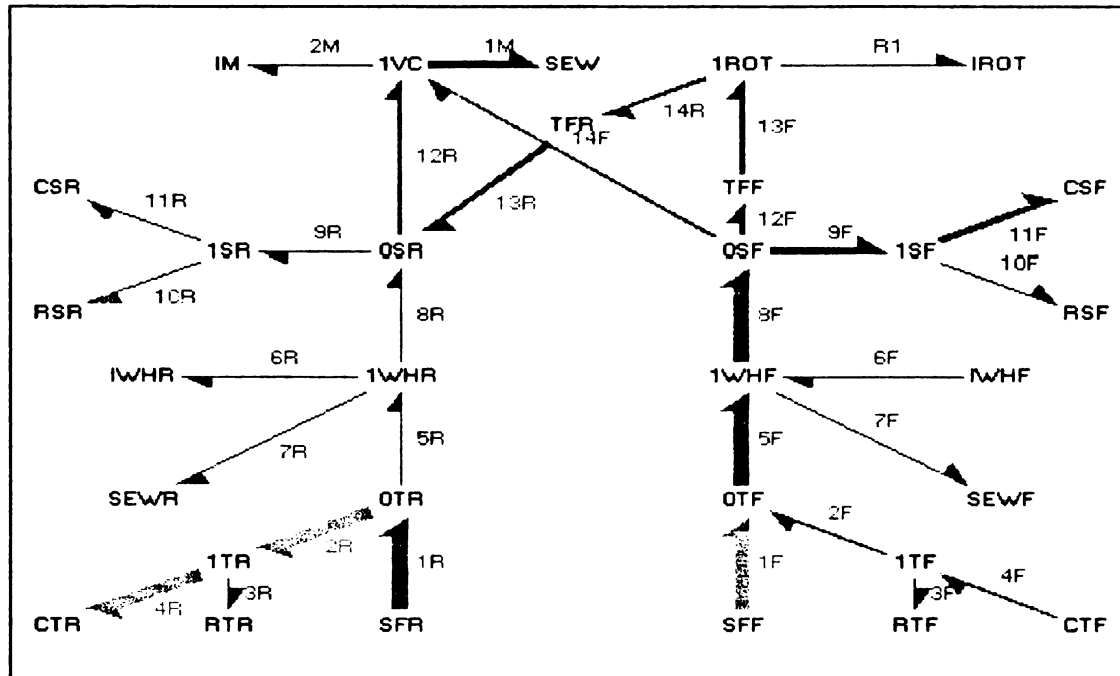


Figure 13. Power Variable Response for Flatbed Trailer (Time = 0.21 seconds)

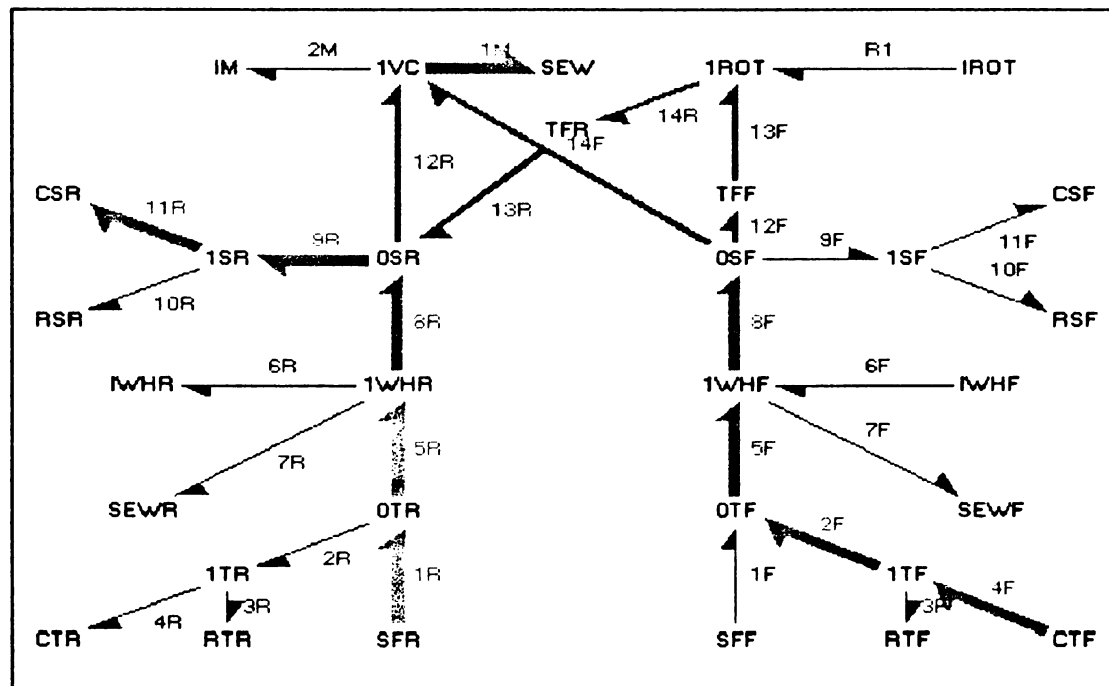


Figure 14. Power Variable Response for Flatbed Trailer (Time = 0.25 seconds)

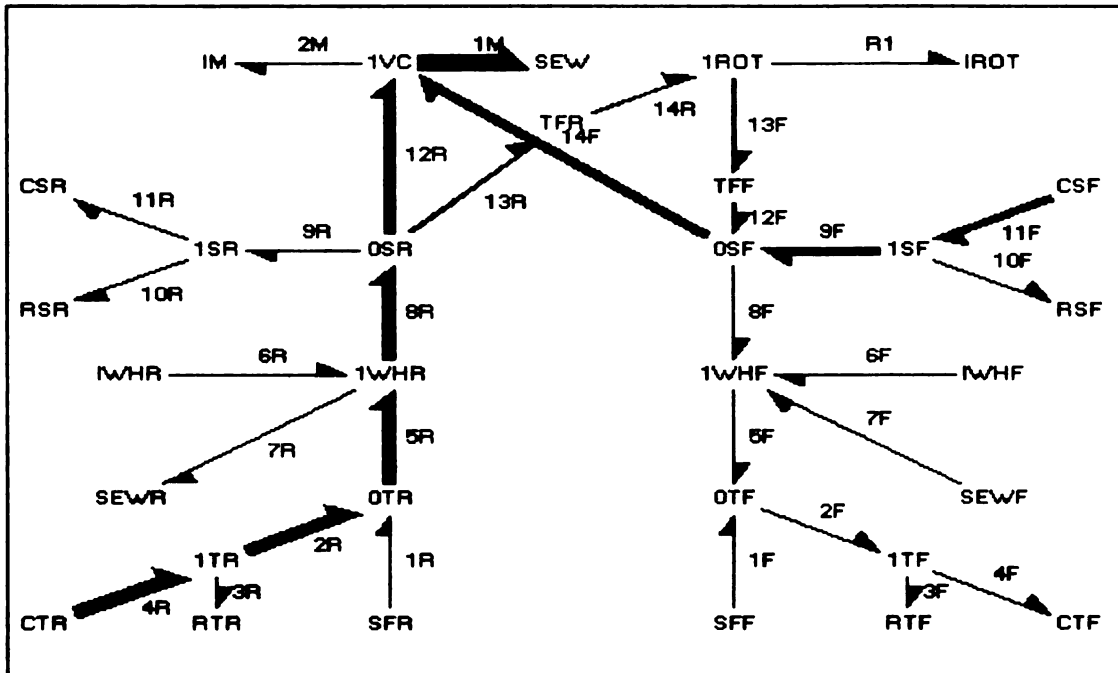


Figure 15. Power Variable Response for Flatbed Trailer (Time = 0.35 seconds)

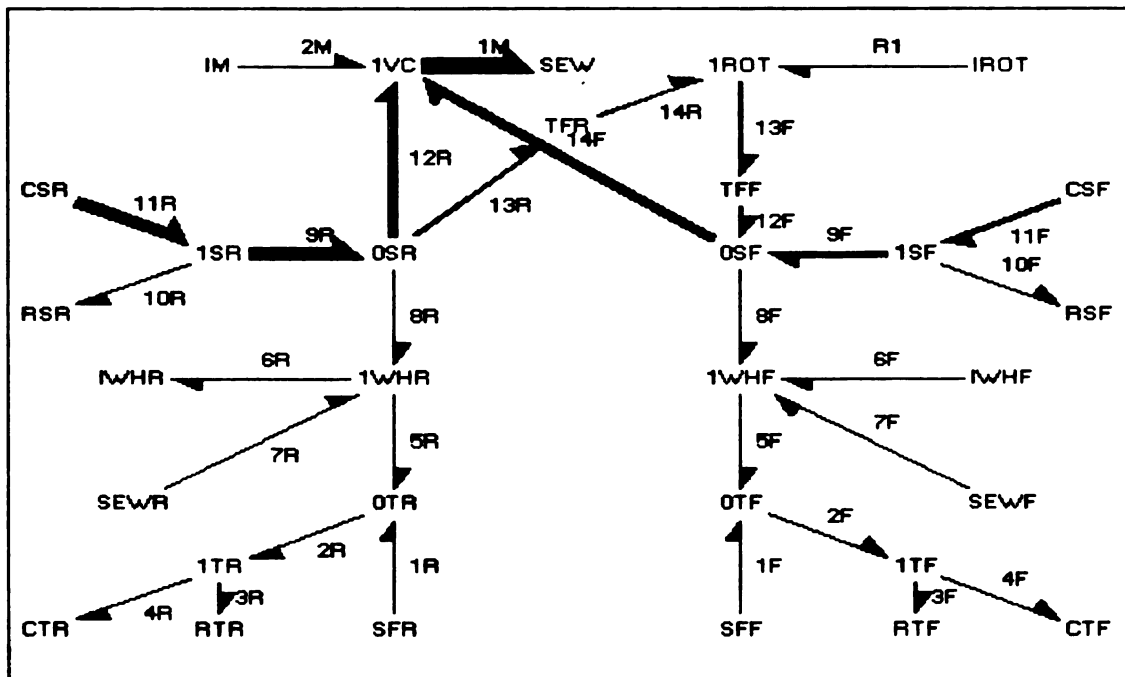


Figure 16. Power Variable Response for Flatbed Trailer (Time = 0.45 seconds)

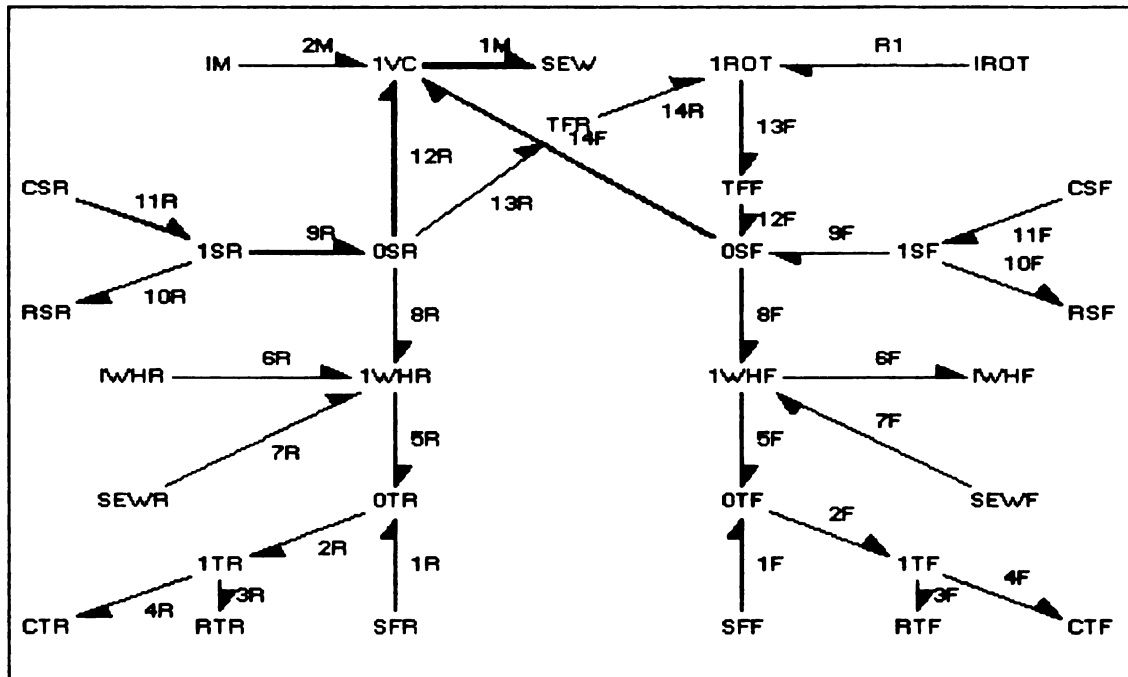


Figure 17. Power Variable Response for Flatbed Trailer (Time = 0.55 seconds)

From a dynamic display, changes in the sign of data are clearly evident from either the direction of the bond changing or a change in the sign of the label on the bond. In this example, it is very clear when an element is adding or removing power from the system. This can also be very useful for model verification. For example, from a dynamic display of the effort in the flatbed trailer model, one could show that the force in the tires was never negative.

The dynamic display provides valuable qualitative information about the system behavior. Information and insight gained from viewing the dynamic display can be utilized best if used in conjunction with time plots. Once a problem area or an area of interesting behaviors is determined from the dynamic display, time plots and tables

of the data can be used for quantitative analysis of the system. The dynamic display can be used as a method of sorting through the abundance of data generated when numerical simulations of large systems are performed to determine parts of the system that should be analyzed in more detail.

Chapter 3

Implementation of the Dynamic Display Module

This chapter contains a description of the implemented design and discussion of the reasons for the design choices. Program code referenced in this chapter can be found in Appendix B.

3.1 File Importing

The Dynamic Display module requires knowledge of the bond graph model structure and the response data obtained from the computer generated solution of the system equations to produce the dynamic display for visualization. This information is obtained from the Model Builder and Solver modules of Enport. Once the user has created a bond graph model in the Model Builder module and input the system parameters, the system equations are generated. This information is then used by the Solver module to numerically integrate the equations over a desired time interval and obtain the solution.

3.1.1 Importing Model Files

Model files are created in the Model Builder module of Enport. The Model Builder module is used to create a bond graph and define the system equations. Once a model has been made in the graphical editor, the system equations obtained from the model can be viewed and edited. In order to solve the equations, they must be sorted by the Model Builder and saved with the bond graph to a model file with the suffix “.enp”. The file “ddfil1.for” contains the code that handles model file loading and viewing. The Dynamic Display module needs the information to recreate the bond graph model only. Although the system equations are not loaded, they may be viewed from the Dynamic Display module. This option enables the user to view exactly what equations were solved.

3.1.2 Importing Results Files

Results files are binary and have the suffix “.bdt”. They contain the results generated by the Solver module. In the Solver module, the user must specify solution control information such as initial conditions, the type of integrator desired, the time interval, and the time storage interval. Once output variables have been chosen, the Solver numerically integrates the sorted equations obtained from the model file and generates the results file.

To create a dynamic display, the user needs to save only effort and flow variables in the Solver module since the dynamic display is limited to effort, flow, or power. If power is desired for display, the user will need to save the effort and flow of every bond that they will want to display since the Dynamic Display module multiplies the effort and the flow to get the power.

The program code in the file “ppinfl.for” allows for user selection of the results file and the variables to be displayed. Variables desired for display are selected from a list of the variables saved in the results file. Bonds that do not have the correct type of result data available to them will appear gray and have a width of 1 pixel in the dynamic display. The data from the results file will be loaded as needed by the subroutine “getres” in the file “ddisp1.for”. This subroutine accesses the results directly by record number based on the time of the results being displayed.

3.2 Dynamic Display Module Features Implemented

A configuration must be chosen for running the Dynamic Display module that sets the variables representing bond width and color, the way negative data will be handled, and time advancement. Other viewing options are available when the graph is displayed. Configurations may be set by loading a configuration file or by menu selections. The file “ddcfgw.for” contains the subroutines that implement the file

operations and the configuration menu window located under the “Options” menu in the main menu.

3.2.1 Variable Choices and Bond Coding

At most, the user may choose to display one of the following types of variables for a given run: effort, flow, or power. This choice was made because when one type of variable is represented by width and another type by color, the display becomes confusing and it is hard to visualize how the system is behaving. This situation would be further complicated by the handling of negative values. Currently, the color and width only represent the absolute value of the variable. Negative data can be indicated by either switching the direction of the bond or by a negative sign appearing by the bond name. If different variables were represented by color and width, a choice would have to be made on how to handle negative values for width and color separately. While this can be done, it generates a display that can be more confusing than useful.

The width and color of the bonds may be chosen to be constant instead of varying with time. Maintaining a constant width will be useful in cases where the system is very big and larger bond widths distort the graph or cover bond labels. Constant color was provided so that black and white hard copies of the display could be made without having some of the bonds appear gray from grayscaling of the

colors. The color can also show the power type. This may be desirable in cases where more than one power domain is present in a system.

When the bond width or color is determined by the effort, flow, or power; the maximum and minimum of all the variables selected for display are calculated by the subroutine "getres" in the file "ddisp1.for". The maximum and minimum are found by comparing the absolute value of all the results that will be used in the display. The largest value will be the maximum and the smallest value will be the minimum. The minimum value will usually be zero. The resulting range of the variables is divided by the number of different widths or colors possible to determine the increments between widths or colors.

Since different maximum widths may be desired for different size systems, the user can choose the desired maximum width to be between two and eight pixels when width is variable. The maximum width will represent the largest magnitude values of the variable while a width of one pixel will represent the values closest to zero. The subroutine "findwd" in the file "ddisp2.for" determines the bond width based on the ratio of the current results to the maximum value of the variable type during the time interval. When the width is constant, the width may be set between one and eight pixels.

The color range for variable color includes eight colors with the highest values represented by the lightest color and the smallest values represented by the darkest color. Bond color is determined in the subroutine "findcl" in the file "ddisp2.for" based on the ratio of the current results of the variable to the maximum value of the variable type in the system over the time interval. The colors used to represent the power type are consistent with those in the Model Builder module. When the color is chosen to be constant, the user may pick from a list of 12 colors with which to display the bonds.

A legend containing information on the width may be shown on the computer screen during the display. Eight different width legends are shown in Figure 18. Due to the fact that the type of variable being shown is also displayed in the legend, there are 22 width legends possible. For variable width, the legends display what type of variable is being displayed, the scale, and the maximum and minimum of the variable type. When the width is constant, a message indicating this is displayed in the legend. The width legend is drawn by the subroutine "wdlgnd" in the file "ddisp1.for".

There are five possible color legends; three of which are displayed in Figure 19. When the color is determined by power type, the color legend displays the colors and the corresponding power types. A message is displayed if color is held constant. The color legend is drawn by the subroutine "crlgnd" in the file "ddisp1.for".

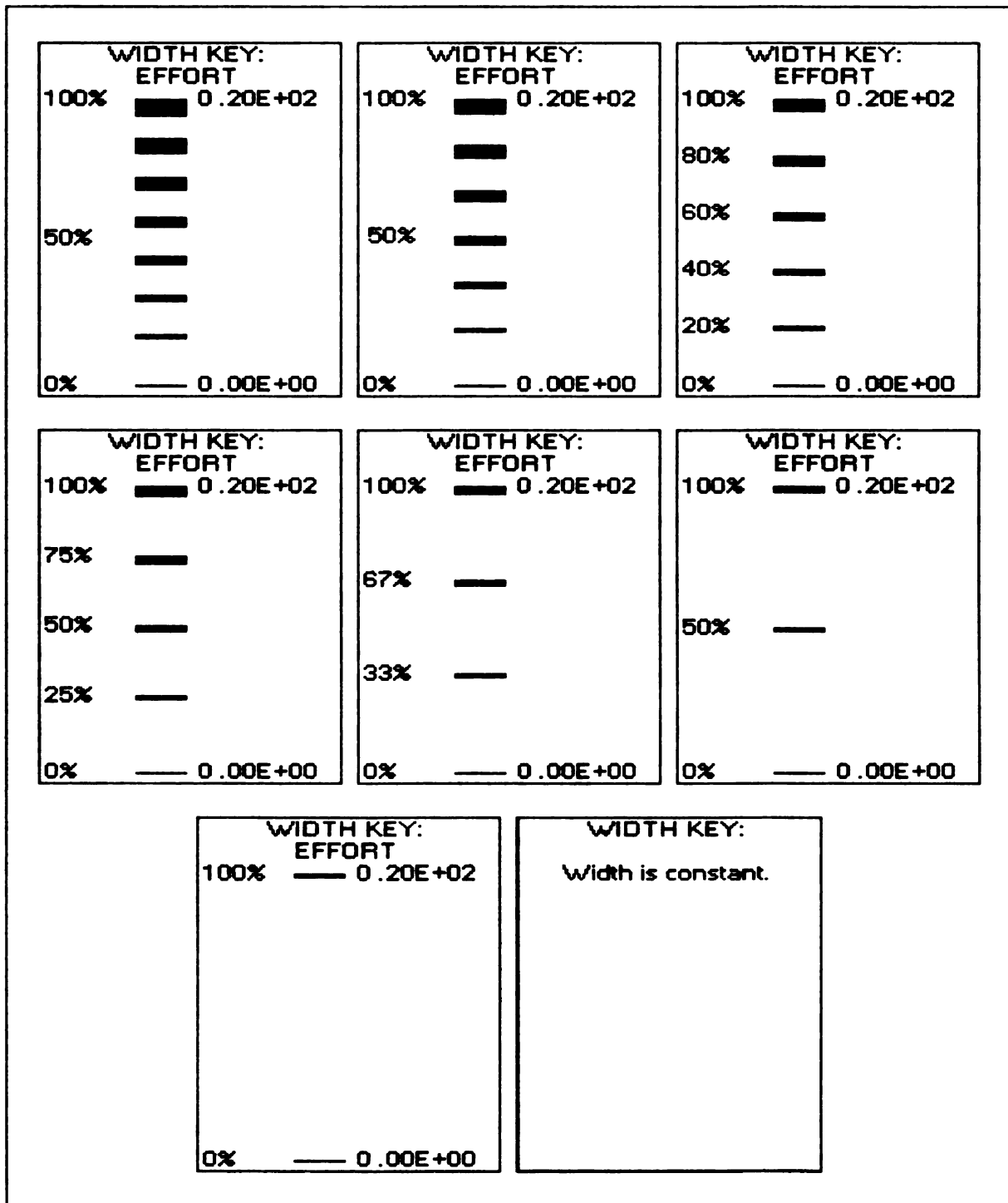


Figure 18. Width Legends Available in the Dynamic Display

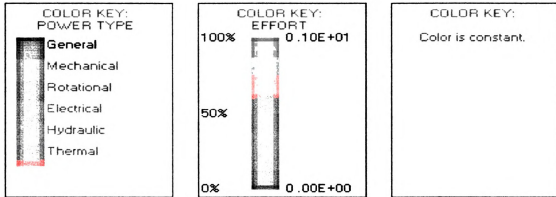


Figure 19. Color Legends Available in the Dynamic Display

The time legend displays the time and the time step between results. An example of the time legend is shown in Figure 20. Every time the graph is redrawn, the legend is updated with the new time. The time legend is created in the subroutine "tmlgnd" in the file "ddisp1.for".

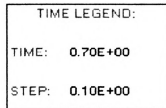


Figure 20. Time Legend for the Dynamic Display

The user may choose to not show the legends, also. In this case, the logical variable "LEGEND" in the "sysdrw" subroutine will be ".FALSE."; and the time will be displayed below the graph instead of in the legend. The width and color legends are not shown at all. The graph will then be displayed on a larger screen which is the

same size as the editing screen in the model builder. This option is especially desirable if the user uses the program frequently and is familiar with the legends.

3.2.2 Frame Advancement Options

When the dynamic display is begun, the bond graph model initially appears as it does in the Model Builder, scaled to fit the screen; with bond color determined by power type and a bond width of one pixel. Once the dynamic display has been started, the graph will change to show the chosen variables. An example of the Dynamic Display screen can be seen in Figure 21. This screen also allows the user to select viewing options for the display which may include displaying causality, removing the legends, and zooming in on a part of the graph.

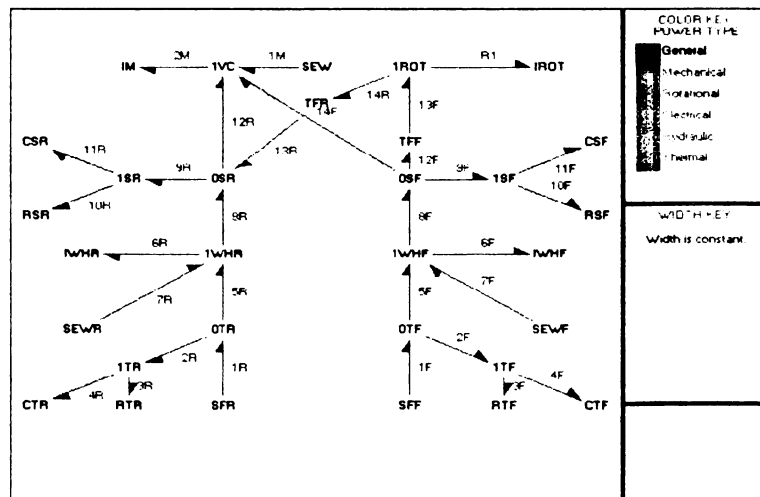


Figure 21. Dynamic Display Module Graphical Display

When the user wants to be able to proceed through the time record at leisure, manual frame advancement may be chosen. This allows the user to move forward or backward through the time record of the results step by step. At each time step, the results are read in using the subroutine "getres" and the system is redrawn by the subroutine "sysdrw", which calls many other subroutines. While moving through the time record, the user has many options available in menu form, such as zooming in or out or displaying the causality of the bonds. These options will be discussed further.

To facilitate animation of the results, automatic frame advancement is available. If this is chosen, the user may then enter the desired time between frames. This time can be anywhere between 0.00 and 9.99 seconds. In this way the user could choose the time advancement to occur with spacing equivalent to the time step between results and thus see the results in real time. If the results cannot be displayed as fast as desired, the results will be displayed as fast as possible. The subroutines "settim" and "gettim" are used in the subroutine "ddcntr" to keep track of the time. When the final time is reached, the last result will be displayed along with a menu of options similar to that for manual frame advancement.

3.2.3 Drawing the System

Since bond width may vary, depending on the user choices, a new bond was designed. To make bonds of varying widths, a polygon function is used. The resulting

bonds have a closed arrow head, as can be seen in Figure 22. The subroutine “drarrow” in the file “ddisp2.for” determines the indices of the arrow based on the desired width and endpoints of the bond. For cases where there are no interior points, the polygons are also drawn by this subroutine.

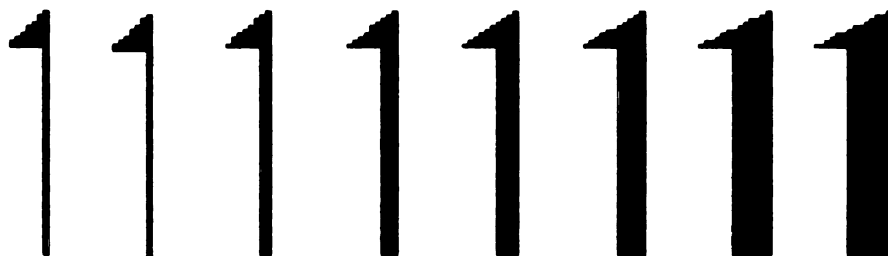


Figure 22. Examples of Bonds of Varying Widths

When there are interior points in a bond, that is, the bond is not straight, the line segments are sent to the “drarrow” subroutine and the points on each segment are determined. These points are saved in an array for all the line segments and then sent to the subroutine “inter”, which determines the intersections of the line segments and draws the resulting polygon. Examples of the resulting bonds are shown in Figure 23.



Figure 23. Examples of Bonds with Interior Points

3.2.4 Options Available During Display

For large systems, it is often desirable to look at small parts of the system. When the system bond graph is initially displayed, it is scaled to fit the window. The user can then change this view by zooming in on a desired portion of the graph. The view may also be changed by moving to the right, the left, up, or down. A zoom out option is also available as well as an option to frame the graph by scaling the system graph to fit the window. The subroutine “gfzoom” was used to implement these viewing features.

If for some reason the graph does not look as it should, a redrawing option is available that calls the subroutine “redraw”. This subroutine redraws the system graph and the legends.

In some cases it is desirable to add some things to the graph, such as causality. There are options available to the user to choose the grid visibility, the frame visibility, the causality visibility, whether to show the node color, whether to show the legends, and whether to show the connector labels. The subroutine “gfopts” in the file “ddisp4.for” sets logical variables from the user choices, which are then used by “ddcntr” to vary the display. When visible, the grid will appear as white dots on the graph. When the frame is not visible, the frames around the legends will remain, but the frame around the graph will not. The causality is shown with a causal stroke in the

same color as the bond. When the node color is shown, the node color represents the power type of the node. When the legends are shown, they will appear on the right side of the screen. The option of not showing the connector labels was provided for convenience when the system is large. Removing the connector labels can help unclutter a graph if the labels are covering up bonds, etc.

Information about the nodes and connectors is available to the user during the Dynamic Display through use of the subroutine “gflist”. A list of the nodes and connectors can be shown as well as the node and connector attributes. The node-connector list displays each node and its corresponding connectors. The node attributes list displays information about the node including: its type, its power type, its location on the graph, and its visibility. The connector attributes list shows the starting and ending node of each connector, the type and power type of each connector, and the visibility of each connector.

Chapter 4

Conclusions

4.1 Summary of Contributions

The research work completed has significantly increased the usefulness of the PC version of Enport by tying visualization of the results to the bond graph model structure. The principal research objective of implementing a system in the PC-DOS environment that will display the dynamic response directly on the system bond graph model has been achieved.

The Dynamic Display module provides the link between the system bond graph structure and the dynamic response of the system. This link allows the engineer to take advantage of the knowledge that bond graphs supply about system behavior through the visualization provided by the dynamic display. Since the whole system response can be viewed at once, model verification is easily done, as well as gaining insight about the model behavior. Complex system behaviors can be easily shown through use of this display.

The primary functions implemented in the Dynamic Display module were: 1) loading of bond graph models into the Dynamic Display module; 2) loading of results

from results files into the Dynamic Display module; 3) multiplication of effort and flow to obtain the power when desired; 4) seven different scaling options for width; 5) constant color and width options; 6) a color by power type option; 7) bond width and color variation for a bond graph display of variables as if they were flowing on the graph; 8) time controls necessary to control the animation manually or automatically; 9) two options for handling negative values; 10) viewing controls that provide control of the size of the bond graph and what is shown on the bond graph. Two engineering system examples, a mechanical oscillator and an automobile suspension system, were developed to demonstrate the application of the Dynamic Display module.

4.2 Areas for Further Research

Several areas have been identified as possibilities of future research. Most of these pertain to improvement of the display allowing more user control and a higher quality display. The following areas warrant further research.

1. In some cases it may be desirable to have different scaling options for different power types. Some power domains, such as hydraulics, have small powers relative to the other domains. This may cause the power to be shown as close to zero on the hydraulic bonds for the entire time record. It is suggested that until this feature is implemented, choosing the bonds of only one power type at a time for display will eliminate the problem.

2. It would be desirable to only redraw the bonds that change for each time step. This is especially desirable for large systems where the redrawing of the system is most noticeable. In the Dynamic Display module, the entire visible graph is redrawn at each time step. Reducing the number of items to redraw at each time step would decrease the minimum time step possible, since the minimum time between frames is limited by the time it takes to draw the system.

3. Logarithmic scaling would be useful for systems whose results' magnitudes are logarithmically spaced. In these cases, a linear scale tends to make most of the results that are smaller than the maximum value appear close to zero.

4. Signal variables are not displayed in this module. However, there is code available in the subroutine "drarow" in the file "ddisp2.for" to generate signals of differing width and color, just as the code to generate bonds does. The only difference between the signal and bond drawing techniques is that a signal can never switch directions and therefore negative signal data is always indicated by a negative sign in front of the signal name.

5. In the future, it would be desirable to allow the user to edit the bond graph in the Dynamic Display module. Utilities could be created to allow the user to move or delete the legends and the connector labels. Another feature that would be useful would be to allow the user to move the nodes without removing the connectors, or to

add interior points to a connector. These features would be especially useful for large systems where the dynamic display may be very cluttered.

6. Allowing the user to select subsets of the time record for display would be useful as well as allowing the user to choose the time step between displayed results. Without forcing the solution to be calculated in the Dynamic Display module, these features could be implemented if the user was limited to pick an initial and final time that was in the time record saved and a time step that was an integer multiple of the time step between saved results.

7. It would be very desirable to allow the user to click on a connector with the mouse and display the exact value of the variable being displayed at that time. This would greatly improve the quantitative information given by the dynamic display. Other options such as some type of indication of when a connector reached its maximum value would be desirable also.

8. Allowing the user to select the number and order of colors in the color legend would be useful since there is not an obvious order of colors that makes sense. Graphic tools could be implemented that would allow the user to select from a wide range of colors, including shades of the same color.

APPENDICES

Appendix A

User's Guide

The following section can be used as reference for running the dynamic display environment. Information on the functions available is located under the appropriate headings. Most selections may be made with either the mouse or with keyboard strokes.

Starting the Dynamic Display Module. The Dynamic Display Module can be loaded by typing "DD" and pressing "Enter" at the DOS prompt in the "Enptdev" directory. Pressing any key will cause the Enport logo screen to disappear and the main screen to appear with the line menu at the top of the screen as shown in Figure A-1. The menu at the top of this screen provides the functions of loading model and results files, setting the controls for the display, and opening the graphical display screen. The last line of the screen shows the option keys and their functions. These keys work almost everywhere in the Dynamic Display Module.



Figure A-1. Main Menu for Dynamic Display

Importing of Enport Files

In order for a dynamic display to be made, a model and the corresponding results file is needed. The model file is imported from the Model Builder Module and the results file from the Solver Module of Enport.

Model Files Operations. Selecting "Model File" from the top line of the main screen will bring up a window containing model file options as shown in Figure A-2. The "Load a file" option opens a window containing the names of the files currently available. To view the file without loading it, select the "Show file contents" option and the list of the available files will appear. If the desired file is not on the list, select "Add a file" to add the filename to the list. There will be a prompt for the path and the filename, which must have the suffix ".enp". The "Delete file" option allows the user to remove a filename from the list. To exit the model file options menu, select "Return" or press the function key "F2".

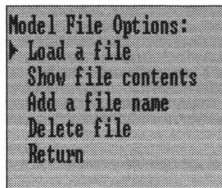


Figure A-2. Model File Options Menu

Results File Options. Selecting "Results file" from the top line of the main screen will bring up a window containing results file options as shown in Figure A-3. The "Load variables" option opens a window listing the results files currently available. When a file is selected from the list, a window with the variables saved in the file appears. All desired results for display purposes should be selected from this list. The "Scan file" option allows a results file to be chosen from the current list for scanning purposes. The information that will be shown includes: information about the file and what model it pertains to, solution control information, and a list of the saved variables. When one of these files is selected, the results file will be displayed on the screen. If the desired file is not on the list, select "Add a file" to add the filename to the list. A prompt will appear for the path and the filename of the file. The filename must have the suffix '.bdt' to indicate it is a results file. The "Delete file" option removes a filename from the file list. To exit the "Results File Options" menu, select "Return".

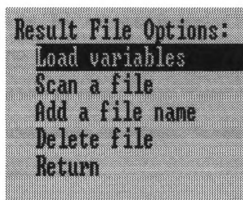


Figure A-3. Results File Options Menu

Setting the Display Configuration

Display configurations may be created or they may be loaded from a file. If no configuration is chosen, the default configuration will be loaded at display time.

Loading a Configuration File. Selecting "Display Options" from the main screen will bring up a window displaying the configuration file options for the display. The configuration may be loaded from a file or selected by several menu choices. The "Load file" option opens a window listing the configuration files currently available. Selecting a file causes the file to be loaded.

Making a New Configuration. Select "Show configuration" from the "Display Options" menu to display the current configuration as shown in Figure A-4. If no file has been loaded, the default configuration will be shown. Move through the configuration menu using the "Tab" key to go forward or both the "Shift" and "Tab" keys to go backwards. The mouse can also be used to select a field. To leave the configuration window and save the changes, select "Done". Select "Cancel" to exit the configuration window without saving the changes. To reset the configuration to the original configuration, select "Reset".

BOND WIDTH	BOND COLOR	MISCELLANEOUS
WIDTH BY: ►EFFORT ↑	COLOR BY: POWER TYPE ↑	NEGATIVE DATA: DIRECTION ↓
WIDTH <IF CONST>: 1 ↓	COLOR <IF CONST>: BRIGHT WHITE ↑	FRAME ADVANCEMENT: MANUAL ↓
MAXIMUM WIDTH: 8 ↑		TIMING:0.50seconds
CANCEL	DONE	RESET

Figure A-4. Display Configuration Window

Width Options. The "Width by:" option in the "Show configuration" window allows for specification of whether the width of the bonds will represent effort, flow, power, or be constant. The selected option is displayed. To see another selection, move through the options using the up and down arrows on the keyboard. To choose a menu item, press "Enter" or click on it with the mouse. If constant width is chosen, the desired width can be set to integer values between 1 and 8 pixels with the "Width (if const):" option. If the width is not constant, the maximum width can be set to and integer between 2 and 8 pixels using the "Maximum Width:" option.

Color Options. Selecting the "Color by:" option from the "Show configuration" window allows for the specification of whether the color will represent power type, effort, flow, power, or be constant. If the width was not constant, the

user will be limited to choose color to either represent the same variable as width, the power type, or remain constant. When the color is chosen to be constant, the color may be selected with the "Color (if const):" option.

Negative Data Handling. To choose how negative results should be represented, select "Direction" or "Negative Sign" from the "Negative Data:" option in the "Show configuration" window. If "Direction" is chosen and the data is negative, the direction of the bond will change. If "Negative Sign" is chosen, negative data will be indicated by a negative sign in front of the bond label.

Timing Options. The "Frame Advancement:" option from the "Show Configuration" window allows for the choice of "Automatic" or "Manual" frame advancement. When "Automatic" frame advancement is chosen, the time between frames should be entered from the keyboard at the "Timing:" prompt. The value is entered in seconds and hundredths of a second. Values between 0 and 9.99 seconds are allowed.

Saving a Configuration to a File. Selecting "Save to file" from the "Display Options" menu allows the user to save the current configuration to a file. The user will be prompted for a path and filename. The filename must be 8 characters or less with the suffix '.cfg' to indicate that it is a configuration file. Selection of the "Save as

Default" option will cause the current configuration to be saved to a file called 'Default.cfg'. This will overwrite the existing default file.

Adding and Deleting Configuration Files. Select "Add a file" to add a filename to the current configuration file list. There will be prompts for the path and the filename of the file. The filename must have the suffix '.cfg' to indicate it is a configuration file. When "Delete file" is selected, the current list of configuration files will appear. To delete a file, select the file to be deleted and press return.

Viewing the Dynamic Display

When "Graph" is selected from the menu line at the top of the screen, the bond graph model will appear on the screen as shown in Figure A-5. Power type will be shown by the color of the bonds and the bond width will be 1 pixel.

The Dynamic Display will be started by selecting "Start" or "Forward" from the line menu below the graph. The following legends are shown on the screen with the dynamic display: a color legend, a width legend, and a time legend. The color legend displays what the color represents and shows the maximum and minimum values of the data for cases where the color represents effort, flow, or power. The width legend shows what width represents and the maximum and minimum values of the data for

cases where the width is not constant. The time legend shows the current time and the time step size once the dynamic display has begun.

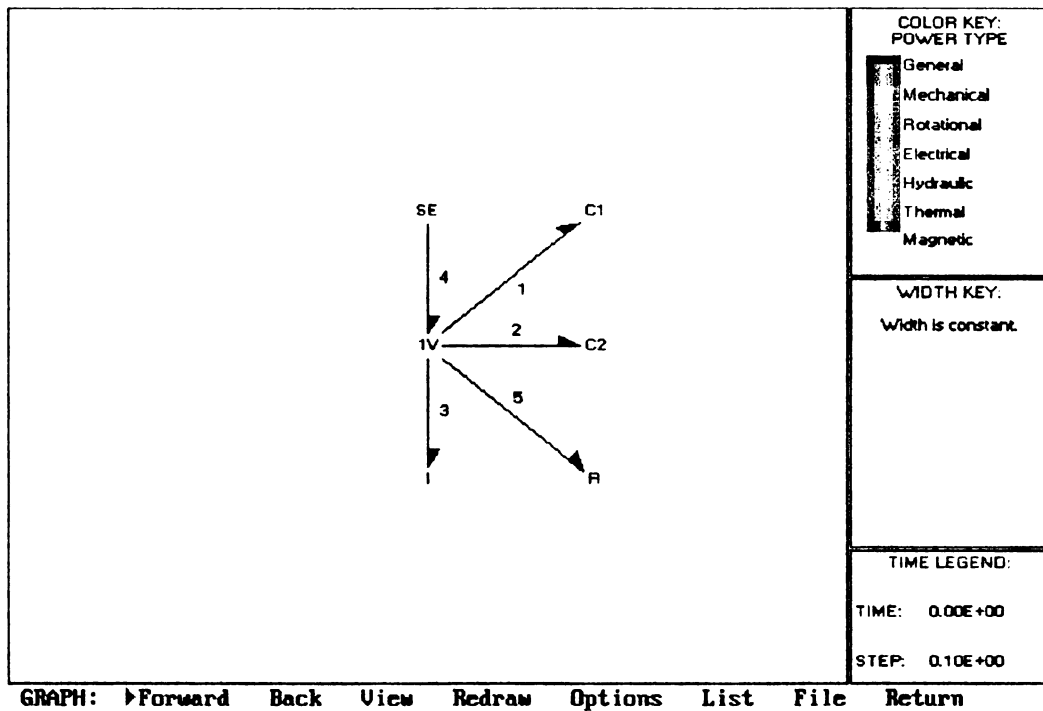


Figure A-5. Initial Screen of the Dynamic Display Module

Automatic Frame Advancement. If automatic frame advancement was chosen, the menu at the bottom of the screen will appear and include the following options: "Start", "View", "Redraw", "Options", "Lists", "File", and "Return". Selecting "Start" will cause the dynamic display to begin. The results will be displayed at the user specified rate. However, if the user specified rate cannot be achieved, the program will display the results as fast as possible. At any point during the automatic advancement, the display may be canceled by pressing "Escape". When the final time

is reached, the manual frame advancement menu will appear. The user may then select "Return" to leave the display or choose among the menu items.

Manual Frame Advancement. If manual frame advancement was chosen, the list of options will be the same as for automatic frame advancement except that "Forward" and "Backward" will appear instead of "Start". "Forward" is selected to advance manually through the results by the specified time step. "Backward" allows the user to go back steps in increments equal to the time step. At any point during the display, the user may select any of the other menu items.

View Options. When "View" is chosen from the line menu displayed below the graph, a new menu will appear. The user can then choose to zoom in on a specific area, zoom out to see more of the graph, or adjust the view up, down, right, or left.

Display Options. Selecting "Options" from the menu causes a list of options to appear. These options include: "Grid visible", "Frame visible", "Causality visible", "Show node color", "Show legends", and "Show conn label". They may be chosen by entering an "x" in front of "Y" for yes or "N" for no.

List Features. Several lists are available for view when "List" is chosen. These lists include: "Node-connector list", "Node attributes", and "Connector attributes". When one of these lists is chosen, a window will appear that contains the list.

Leaving the Dynamic Display Module

The Dynamic Display module must be exited from the main screen. Selecting "Leave" from the top line menu or pressing the "F2" function key will cause exiting from the module.

Appendix B

Program Code

The software developed for this application is stored in several files in the directory “c:\enptdev\jlo\dddev\src\dd”. It is necessary to compile this code along with the ENPORT library routines. What follows is a listing of the FORTRAN language code used to implement the Dynamic Display module.


```

30  CONTINUE
C
DO 40 J=1,MNBD
  BGNAM(J)=' '
  IBDSEL(J)=0
  IBDSEL(J*MNBD)=0
  IELSD(J,1)=0
  IELSD(J,2)=0
  BDCX(J)=0
40  CONTINUE
  INTPT(1)=1
C
C----- Set the node and connector default types
  NOTPDF=.FALSE.
  NODEFT='M'
  CNTPDF=.FALSE.
  CNGEFT='B'
  NDFSDF=.TRUE.
C
C  CMS=.TRUE.
C
C----- Problem equation definition (FUNCEK)
  PARNUM=0
  INEQ(1)=1
  NEWS=0
  IFVOT(1)=1
  IPVOT(1)=1
  IEQPTR(1)=1
C
C----- Set certain graph display parameters (BGFEXK)
C----- This set of data defines the graph window in pixels
  RS = GFVC.NUMTEXROWS
  XS = GFVC.NUMPIXELS
  YS = GFVC.NUMPIXELS
  XNPXL = 0
  XNPXL = XS - (6*(XS/RS) + 2)
  XFPXL = XS
  YNPXL = 0
  YNPXL = YS - (4*(YS/RS) + 2) ! Leave 4 rows at the bottom
  YRPXL = YS
  CXPXL = XS - 1
  CXPXL = XS - 1
  CXPXL = XS
  CYNPXL = 0
  CYRPXL = YNPXL+2/5
  CYRPXL = CYRPXL - 1
C----- This set of data defines the width legend window in pixels
  WNPXL = XS - (6*(XS/RS) + 2)
  WXPXL = XS - 1
  WXPXL = XS
  WNPXL = 0
  WRPXL = WNPXL+2/5
  WRPXL = WRPXL - 1
C----- This set of data defines the time legend window in pixels
  TNPXL = XS - (6*(XS/RS) + 2)
  TXPXL = XS - 1
  TXPXL = XS
  TNPXL = 0
  TRPXL = TNPXL+2/5
  TRPXL = TRPXL - 1
C----- This set of data defines the graph window in window units
  ACTND(1)=.TRUE.

```

[illegible]


```

DATA MESS1 '/' Error opening: '/'
DATA MESS2 '/' *** Error reading/writing file '/'
DATA MESS3 '/' *** The directory is empty. '/'

C
C-----DOVFIL-----
IPAG= 0
OPAG = SETACTIVEPAGE(IPAG) ! SETVISUALPAGE in MNUSEL
C----- Find directory file and read into LISTF
C
C----- (directory was checked at startup)
C
C----- DRYNAM= MODPTH(1:LMOD) //'ENPFIL.DRY'
C----- OPEN(FRUNIT,FILE=DRYNAM,STATUS='O.D')
C----- READ(PRUNIT,(3X,I4,'') NF
C----- DO 20 I= 1,NF
C----- READ(PRUNIT,'(A12') LISTF(I)
C----- REW'ND(PRUNIT)
C----- CLOSE(PRUNIT)
C----- CHGRDY= .FALSE.

C
C----- Get next option selection
C----- IOPT= -3
C----- IOPT= -3
C----- CONTINUE
C----- IOPTH= IOPT
C----- CALL MNUSEL(LIST,MUIM,IOPT)
C
C----- IF (IOPT.EQ.0) THEN
C----- DPAG= 3
C----- FILNAM= 'DOFIL.HLP'
C----- CALL SHOHLP:DPAG,FILNAM)
C----- IOPTH= IOPT
C----- ELSEIF (IOPT.EQ.1) THEN
C----- IF (NF.EQ.0) THEN
C----- MESSAGE= MESS4
C----- CALL LNMMSGD(24)
C----- GOTO 100
C----- ENDIF
C----- HDR= ' File names:'
C----- FILNAM= ''
C----- CALL LSTLST(IPAG,MLTML,HDR,LISTF,NF,'SEL',EVTP,FILNAM)
C----- IF (EVTP.EQ.'CGET') THEN
C----- Clear display area for file read status report
C----- CALL TEXTBOX(3,41,15,20,BLUE)
C----- Clear any prior model stuff but don't redraw screen
C----- SCRFIL= .FALSE.
C----- CALL DOINIT(SCRFIL)
C----- Read model from file
C----- TRUNAM= MODPTH(1:LMOD) //FILNAM
C----- OPEN(PRUNIT,FILE=TRUNAM,STATUS='OLD')
C----- CALL EPREAD(FILNAM)
C----- REWIND(PRUNIT)
C----- CLOSE(PRUNIT)
C----- ENDIF
C----- ELSEIF (IOPT.EQ.2) THEN
C----- IF (NF.EQ.0) THEN
C----- MESSAGE= MESS4
C----- CALL LNMMSGD(24)
C----- GOTO 100
C----- ENDIF
C----- HDR= ' File names:'
C----- FILNAM= ''
C----- CALL LSTLST(IPAG,MLTML,HDR,LISTF,NF,'SEL',EVTP,FILNAM)
C----- IF (EVTP.EQ.'CGET') THEN

```

```

C
INCLUDE 'COMWBK.CBK'
INCLUDE 'VIEWBK.CBK'

CHARACTER VUEENM*8, VUEENM*32, VUETTL*72, VUEDSC*(..0)*72
CHARACTER EVT*VP*4, EVT*P*4, STRING*40
INTEGER*2 VUENLD, N, IR, IC, IPAG, CPAG, DLIM(4), STATUS
LOGICAL RLUD(4)

C
EXTERNAL TREHOT, EVT*TP, SPKEYL, GETSTR, EDLSEG, UPCASE
EXTERNAL TEXTBOX, WRITESTRING
INTRINSIC CHAR, MIN

C
DATA DLIM /12,2,21,73/, RLUD /4*.FALSE./

C
C**DOVHDR.....
CPAG = GETACT*VEPAGE(1)
IF (CPAG.NE.IPAG) THEN
  CPAG = SETACTIVEPAGE(IPAG)
  STATUS = SETVISUALPAGE(IPAG)
ENDIF
STATUS = SETBKCOLOR(BLUE)
CALL CLEARSCREEN($GCLEARSCREEN)
CALL SPKEYL

C----- Put up the labels
STRING= 'ENFOR Model Header Block '
CALL WRITESTRING(STRING,BLUE,WHITE,2,2,27)
STRING= ' Model Name: '
CALL WRITESTRING(STRING,BLACK,CYAN,4,2,14)
STRING= ' File Name: '
CALL WRITESTRING(STRING,BLACK,CYAN,6,2,14)
STRING= ' Model title: '
CALL WRITESTRING(STRING,BLACK,CYAN,8,2,14)
STRING= ' Model Description: '
CALL WRITESTRING(STRING,BLACK,CYAN,11,2,20)

C
C----- Get the current header data
CALL TREHOT('GET',VUEENM,VUEENM,VUETTL,VUEDSC,VUENLD)

C----- Put up the data
STRING= VUEENM
CALL WRITESTRING(STRING,BLACK,WHITE,4,16,6)
STRING= VUEENM
CALL WRITESTRING(STRING,BLACK,WHITE,6,16,32)
STRING= VUETTL
CALL WRITESTRING(STRING,BLACK,WHITE,9,2,72)
DO 20 N=1,10
  STRING= VUEDSC(N)
20 CALL WRITESTRING(STRING,BLACK,WHITE,11*N,2,72)

C
C----- Set up cursor in Mode_name field as default
IR= 4
IC= 16

C
C----- Get data from next field
100 CONTINUE
IF (IR.EQ.4) THEN
  CALL GETSTR(IR,16,8,VUEENM,EVT?)
  CALL UPCASE(VUEENM)
ELSEIF (IR.EQ.6) THEN
  CALL GETSTR(IR,16,32,VUEENM,EVT?)
  CALL UPCASE(VUEENM)
ELSEIF (IR.EQ.9) THEN

```


[illegible]


```
C
C----- Notes:   The calling fuction sets the active page, the display
                  page is set just before the user is prompted for input.
C
C      INCLUDE 'GRAPH.FD'
C      INCLUDE 'PMOUSE.CBK'
C      INCLUDE 'VIEWRK.CBK'
C      INCLUDE 'COMBK.CBK'
C
C      CHARACTER LSTSEG(*)(*), EVTP*4, EVTYP*4, STRING*80, LINSTR*80
C      CHARACTER DATA*80
C      INTEGER DLIM(4), IR, IC, I:, I2, LI, FL, CS, CE, STATUS
C      INTEGER FLIM(2,4), NFELD,NFLD, GPOFT
C      LOGICAL RLUD(4)
C      RECORD /RCOORD/ TMTPOS
C
C      EXTERNAL EVTYP, LINSTR
C      EXTERNAL GETKEYVALUE, WRITESTRING, MEFLD, SHOWMOUSE,HIDEMOUSE
C      INTRINSIC MAX, MIN
C
C      C---EDLSEG*****
C
C      Display the data list
C      CS= DLIM(2)
C      CE= DLIM(4)
C      FL= CE -CS +1
C      DO 5 IR= DLIM(1),DLIM(3)
C          STRING= LSTSEG(IR-DLIM(1)+1)
C          CALL WRITESTRING(STRING,BLACK,WHITE,IR,CS,FL)
C
C      Position the pointer
C      IR= DLIM(1)
C      IC= CS
C      STATUS = DISPLAYCURSOR($G$CURSORON)
C
C      Defined the field array
C      NFLD = 3
C      FLIM(1,1) = DLIM(1) - 1
C      FLIM(1,2) = DLIM(2)
C      FLIM(1,3) = DLIM(1) - 1
C      FLIM(1,4) = DLIM(4)
C      FLIM(2,1) = DLIM(1)
C      FLIM(2,2) = DLIM(2)
C      FLIM(2,3) = DLIM(3)
C      FLIM(2,4) = DLIM(4)
C      FLIM(3,1) = DLIM(3) + 1
C      FLIM(3,2) = DLIM(2)
C      FLIM(3,3) = DLIM(3) + 1
C      FLIM(3,4) = DLIM(4)
C
C----- Set mouse variables
C      GKOPT = KEYOPT
C      IF (MOUSE) THEN
C          LSTPUT = .FALSE.
C          OLDFLD = 2
C          GKOPT = 5
C          CALL SHOWMOUSE()
C      ENDOIF
C
C----- Get some input
C      CALL SETTEXTPOSITION(IR,IC,TXTPOS)
C      CALL GETKEYVALUE(GKOPT,I:,I2)
```



```

C----- Write the Ptr_Dn unless last segment
PCH= 31
IF (L2.EQ.L1) PCH= 32
CALL WRITESTRING(CHAR(PCH),RED,CYAN,MLIM(3),JPU,1)

C----- Display the pointer if select mode
IF (ACT.EQ.'SEL') THEN
CALL WRITESTRING(CHAR(16),RED,CYAN,IP,JPU,1)
ENDIF

C----- Display mouse after drawing menu
IF (MOUSE) CALL SHOWMOUSE()

C----- Define the menu field array
NPLD = L2 - L1 + 2
DO 10 L=1, NPLD
    FLIM(L,1) = MLIM(L) + 2 + (L-1)
    FLIM(L,2) = MLIM(L)
    FLIM(L,3) = MLIM(L) + 2 + (L-1)
    FLIM(L,4) = MLIM(L)
10 CONTINUE
    ! Down arrow is last field
    ! Up arrow is 1st field

C----- Get some input
150 CALL GETKEYVALUE(GKOPT,11,12)
    EVTP= EVTYP(11,12)
CALL MEFLD(2,3,FLIM,NPLD,MEFLD,EVTP)

C----- Crack the user input
IF (EVTP.EQ.'MOVE') THEN
    IF (ACT.EQ.'DSP') THEN
        IF (L2.EQ.71) THEN
            L1= 1
            ! Home
        ELSEIF ( (L2.EQ.73).OR.(L2.EQ.72) ) THEN
            ! PgUp or UpArrow
            IF (L1.GT.1) THEN
                L1= L1 - LSEGL
                LFTBUT = .FALSE.
            END IF
        ELSEIF ( (L2.EQ.81).OR.(L2.EQ.80) ) THEN
            ! PgDn or DnArrow
            IF (L2.IT.L2) THEN
                L1= L1 + LSEGL
                LFTBUT = .FALSE.
            END IF
        ELSEIF (L2.EQ.79) THEN
            L1= L1LAST
        ENDIF
        IF (MOUSE) CALL HIDE MOUSE()
        ELSEIF (ACT.EQ.'SEL') THEN
            IF (MOUSE) CALL HIDE MOUSE()
            CALL WRITECHARS(' ',BLACK,CYAN,IP,JPU,1) ! Erase Ptr
            IF (L2.EQ.71) THEN
                L1= 1
                IP= IPU+1
            ELSEIF (L2.EQ.79) THEN
                L1= L1LAST
                IP= IPLAST
            ELSEIF (L2.EQ.73 ) THEN
                IP= IPD+1
            IF (L1.EQ.1) THEN
                IP= IPU+1
            END IF
        END IF
    END IF
END IF

C----- Put up the background and header
CALL TEXTBOX(MLIM(1)+1,MLIM(2)+1,MLIM(3)+1,MLIM(1),WID,BLACK)
CALL TEXTBOX(MLIM(1),MLIM(2),MLIM(3)+1,MLIM(1),WID,CYAN)
STRING= HDR
CALL WRITESTRING(STRING,BLACK,CYAN,MLIM(1)+1,MLIM(2),WID)
OVRAG = SERV:SUALPAGE(GETACTIVEPAGE())

C----- Display a segment of file starting at L1
100 CONTINUE
C----- Write the Ptr Up unless first segment
PCH= 30
IF (L1.EQ.1) PCH= 32
CALL WRITESTRING(CHAR(PCH),RED,CYAN,IPU,JPU,1)

C----- Write the list data
L2= MIN(L1+1,LSEGL)
IR= IPD
IC= MLIM(2)+3
DO 110 L= L1,L2
    IR= IR+1
    STRING= LIST(L)
    CALL WRITESTRING(STRING,BLACK,CYAN,IR,IC,WID-3)
110 DO 120 L= L2+1,L1+LSEGL
    IR= IR+1
    CALL WRITECHARS(' ',BLACK,CYAN,IR,MLIM(2),WID)
120
C

```



```

DPT6(I) = NEG5(I)
END DO
NSELT6 = NEG5A
L6 = LIST6(DPT6(I))
IF ((DPT4(I).EQ.1.OR.DPT4(I).EQ.5).AND.DPT1(I).EQ.1) THEN
  CALL CFGSEL(-1,FLD6,LBL6,LIST6,LP6,DPT6,NSELT6,ROW,COL,EVTP,SEC)
  GRAND7 = .TRUE.
ELSE
  CALL CFGSEL(0,FLD6,LBL6,LIST6,LP6,DPT6,NSELT6,ROW,COL,EVTP,SEC)
  GRAND7 = .FALSE.
END IF

C----- Set up for "select n" routine: n=1, display len=1 (Section 7)
C
  LBL7 = 'FRAME ADVANCEMENT:'
  FLD7(1) = FLD6(3) + 2
  FLD7(2) = FLD6(2)
  FLD7(3) = FLD7(1) + 1
  FLD7(4) = FLD6(4)
  FLD7(5) = 2
  FLD7(6) = 1
  DO I = 1,2
    DPT7(I) = TIMV(I)
  END DO
  NSELT7 = TIMVA
  LP7 = LIST7(DPT7(1))
  IF ((DPT4(I).EQ.1.OR.DPT4(I).EQ.5).AND.DPT1(I).EQ.1) THEN
    CALL CFGSEL(-1,FLD7,LBL7,LIST7,LP7,DPT7,NSELT7,ROW,COL,EVTP,SEC)
    GRAND7 = .TRUE.
  ELSE
    CALL CFGSEL(0,FLD7,LBL7,LIST7,LP7,DPT7,NSELT7,ROW,COL,EVTP,SEC)
    GRAND7 = .FALSE.
  END IF

C----- Set up for "get string" routine (Section 8)
C
  LBL8 = 'TIMING:'
  FLD8(1) = FLD7(3) + 2
  FLD8(2) = FLD7(2)
  FLD8(3) = FLD7(4) - 11
  FLD8(4) = FLD7(4) - 11
  CDTA8T = SEC8
  IF ((DPT7(1).EQ.1).OR.GRAND7) THEN
    CALL CFGSEL(-1,FLD8,LBL8,CDTA8T,ROW,COL,EVTP,SEC)
    GRAND9 = .TRUE.
  ELSE
    CALL CFGSEL(0,FLD8,LBL8,CDTA8T,ROW,COL,EVTP,SEC)
    GRAND9 = .FALSE.
  END IF

C----- Set up for "get string" routine (Section 9)
C
  LBL9 = '
  FLD9(1) = FLD8(1)
  FLD9(2) = FLD8(3) + 1
  FLD9(3) = FLD8(3) + 3
  FLD9(4) = FLD8(3) + 2
  CDTA9T = HUND9
  IF ((DPT7(1).EQ.1).OR.GRAND9) THEN
    CALL CFGSEL(-1,FLD9,LBL9,CDTA9T,ROW,COL,EVTP,SEC)
    CALL WRITESTRING('seconds',08,WINBC,FLD8(1),FLD8(3)+4,7)
    GRAND9 = .TRUE.
  ELSE
    CALL CFGSEL(0,FLD9,LBL9,CDTA9T,ROW,COL,EVTP,SEC)
    GRAND9 = .FALSE.
  END IF

  CALL WRITESTRING('seconds',9,WINBC,FLD8(1),FLD8(3)+4,7)
  GRAND9 = .FALSE.
END IF

```

```

      ROW = 0
      COL = 0
      END IF
C
      ELSE IF ( (FLD3(1).LE.ROW).AND.(ROW.LE.FLD3(3)).AND.
        (FLD3(2).LE.COL).AND.(COL.LE.FLD3(4))) ) THEN
        (SEC.EQ.3) ) THEN
      + IF (DPT1(1).NE.1) THEN
        ! COLOR IS CONSTANT !
        CALL CPGSEL(3,FLD3,LBL3,LIST3,LP3,DPT3,NSEL3,ROW,COL,EVTP
          ,SEC)
      + PSEC= 3
      + ELSE
        ! Skip Field !
      + IF ( (FLD3(1).LE.ROW).AND.(ROW.LE.FLD3(3)).AND.
        (FLD3(2).LE.COL).AND.(COL.LE.FLD3(4))) ) THEN
        ! If MOUSE, Ignore
      + SEC = PSEC
      + ELSE IF (EVTP.EQ.'TABF') THEN
        SEC = SEC + 1
      + ! If TAB, Skip
      + ELSE IF (EVTP.EQ.'TABB') THEN
        SEC = SEC - 1
      + ! If SHIFT-TAB Skip
      + END IF
      + ROW = 0
      + COL = 0
      + STAT = .TRUE.
      + ! NEGATIVE HANDLING FOR DYNAMIC DISPLAY !
      + CALL CPGSEL(6,FLD6,LBL6,LIST6,LP6,DPT6,NSEL6,ROW,COL,EVTP
        ,SEC)
      + STAT = .FALSE.
      + PSEC= 6
      + END IF
C
      ELSE IF ( (FLD7(1).LE.ROW).AND.(ROW.LE.FLD7(3)).AND.
        (FLD7(2).LE.COL).AND.(COL.LE.FLD7(4))) ) OR.
        (SEC.EQ.7) ) THEN
      + IF (STAT) THEN
        ! Skip Field when NO DYNAMIC DISPLAY !
      + IF (FLD7(1).LE.ROW).AND.(ROW.LE.FLD7(3)).AND.
        (FLD7(2).LE.COL).AND.(COL.LE.FLD7(4))) THEN
        SEC = PSEC
      + ! If MOUSE, ignore
      + ELSE IF (EVTP.EQ.'TABF') THEN
        SEC = SEC + 1
      + ! If TAB, Skip
      + ELSE IF (EVTP.EQ.'TABB') THEN
        SEC = SEC - 1
      + ! If SHIFT-TAB Skip
      + END IF
      + ROW = 0
      + COL = 0
      + ! FRAME ADVANCEMENT FOR DYNAMIC DISPLAY !
      + CALL CPGSEL(7,FLD7,LBL7,LIST7,LP7,DPT7,NSEL7,ROW,COL,EVTP
        ,SEC)
      + PSEC= 7
      + STAT = .FALSE.
      + END IF
C
      ELSE IF ( ( (ROW.EQ.FLD8(1)).AND.
        (FLD8(2).LE.COL).AND.(COL.LE.FLD8(3))) ) OR.
        (SEC.EQ.8) ) THEN
      + IF (STAT.OR.STAT7) THEN
        ! Skip Field if MANUAL or NO DD !
      + IF ( (ROW.EQ.FLD8(1)).AND.(FLD8(2).LE.COL).AND.
        (COL.LE.FLD8(3))) THEN
        SEC = PSEC
      + ! If MOUSE, ignore
      + ELSE IF (EVTP.EQ.'TABF') THEN
        SEC = SEC + 1
      + ! If TAB, Skip
      + ELSE IF (EVTP.EQ.'TABB') THEN
        SEC = SEC - 1
      + ! If SHIFT-TAB Skip
      + END IF
      + ROW = 0
      + COL = 0
      + ! Get SECONDS for Automatic !
      + CALL CPGSEL(8,FLD8,LBL8,CDTAB8,ROW,COL,EVTP,SEC)
      + PSEC= 8
      + END IF
C
      ELSE IF ( ( (ROW.EQ.FLD9(1)).AND.
        (FLD9(2).LE.COL).AND.(COL.LE.FLD9(3))) ) OR.
        (SEC.EQ.9) ) THEN
      +

```



```

C*****FILE*****
C
    IVR = 0
    IAPAG = 0
    NOPTS = 5
    DPAG = 3
    HLPFIL = '....HLP'
    OAPAG = SETACTIVEPAGE(IAPAG) ! VISUALPAGE set in MDUSEL
C----- Find directory file and read into LISTF
C    (directory was checked at startup)
    DRYNAM = ' '
    DRYNAM = SEOPTH(1:LSEQ) //'BCTFIL.DRY'
    OPEN(PRUNIT, FILE=DRYNAM, STATUS='OLD')
C
    READ(PRUNIT, '(3X,I4)') NF
    DO 20 I = 1, NF
        LISTF(L) = ' '
        READ(PRUNIT, '(A34)') LISTF(L) ! A24
    20 CONTINUE
    REWIND(PRUNIT)
    CLOSE (PRUNIT)
C----- Get next option selection
    IOPTH = -1
    OK = .TRUE.
    OKC = .TRUE.
    CHGDRY = .FALSE.
    I = 1
C
    DO WHILE (OK) ! File management
        OK1 = .TRUE.
        OK2 = .TRUE.
        IOPTH = IOPTH
C----- update .dry file
        IF (CHGDRY) THEN
            OPEN(PRUNIT, FILE=DRYNAM, STATUS='OLD', IOSTAT=IOCK)
            IF (IOCK.EQ.0) THEN
                WRITE(PRUNIT, '(NF: ', I4)') NF
                DO 220 I = 1, NF
                    WRITE(PRUNIT, '(A34) ', ERR=922) LISTF(I)
                220 GOTO 230
                WRITE(STRING, '(I4)') I
                MESSAGE = MESSAGE2 // ' at record ' // STRING(1:4)
                CALL LNMSGD(24)
                REWIND(PRUNIT)
            230 ELSE
                MESSAGE = MESSAGE1 // DRYNAM
                CALL LNMSGD(24)
            ENDIF
            CLOSE(PRUNIT)
            ENDIF
            CHGDRY = .FALSE.
C
        FLAG = 0
        CALL MOUSEL(LIST, NOPTS, 3, 5, IOPTH, DPAG, HLPFIL, FLAG)
C----- update file list LISTF
        IF (OK0) THEN
            DO 30 I = 1, NF
                LENG = NCHARS(LISTF(I))
                LISTF(I) = LISTF(I)(1:LENG)
            30 I = LENG, 1, -1

```

```

        IF (LISTF(L)(1:1).EQ.'\\') THEN
            LPT(L) = I ! path length
            LISTF(L) = ' '
            LISTF(L) = LISTF(L)(1:1) ! file path
            J = I + 1
            LISTF(L) = ' '
            LISTF(L) = LISTF(L)(J:LENG) ! file name
            EXIT
        30 CONTINUE
        ENDIF
        OK0 = .FALSE.
        OK0 = .FALSE.
        IF (IOPTH.EQ.0) THEN
            DPAG = 3
            FILNAM = ' '
            FILNAM = 'SLFIL.HLP'
            CALL SHOHF(DAPAG, FILNAM)
            IOPTH = IOPTH
C
        ELSEIF (IOPTH.EQ.1) THEN
            CALL PPVEDT
C----- Scan file
        ELSEIF (IOPTH.EQ.2) THEN
            FILNAM = ' '
            DO WHILE (OK1)
C
                IF (NF.EQ.0) THEN
                    MESSAGE = ' '
                    MESSAGE = '*** The file list is empty. Add file first.'
                    CALL LNMSGD(24)
                    EXIT
                ENDIF
C
                FLAG = 0
                CALL LSTLSS(IAPAG, 3, 35, HDR, LSTFIL, NF, 'SEL',
                    EVTP, FILNAM, FLAG)
                IF (EVTP.EQ.'CRET') THEN
C----- match the selected file
                    IDX = 1
                    DO WHILE (FILNAM.NE. LSTFIL(IDX))
                        IDX = IDX + 1
                    END DO
                    IDX is the selected index of the file
                    TRUNAM = ' '
                    TRUNAM = LISTF(IDX)
                    OPEN(PRUNIT, FILE=TRUNAM, FORM='BINARY', RECL=4,
                        ACCESS='DIRECT', IOSTAT=IOCK, STATUS='OLD')
                    IF (IOCK.NE.0) THEN
                        MESSAGE = ' '
                        MESSAGE = MESSAGE1 // FILNAM
                        CALL LNMSGD(24)
                        FLAG = 1
                        FILNAM = ' '
                        CALL LSTLSS(IAPAG, 3, 35, HDR, LSTFIL, NF, 'SEL',
                            EVTP, FILNAM, FLAG)
C
                    REWIND(PRUNIT)
                    CLOSE (PRUNIT)
                    EXIT
                ENDIF
C

```



```

C
C CHARACTER LSTFIL(50)*12, LISTF(50)*34, FILNAM*12, EVTP*4
C CHARACTER HDR*12, STRING*80
C CHARACTER TRUNAM*64
C INTEGER*2 IOCK
C INTCGER*2 NF, I, I1, IDX, FLAG, RI, CI
C LOGICAL YES, LVES
C
C EXTERNAL LNMGSD, LSTLSS, TEXTBOX, WRITESTRING, WRITESCHAS, YES
C
C DATA HDR / ' File names:' /
C C**RMVFI*****
C C--- See if there is a file to delete
C OKAY=.FALSE.
C IF (NF.EQ.0) THEN
C   MESSAGE = '
C   MESSAGE = ' *** There are no files to delete.'
C   CALL LNMGSD(24)
C   RETURN
C   ENDIF
C   IDX = NF
C
C C--- Get the name of file to delete
C FILNAM = '
C FLAG = 0
C CALL LSTLSS(0,RI,CI,HDR,LSTFIL,NF,'SEL',EVTP,FILNAM, FLAG)
C IF (EVTP.EQ.'CRET') THEN
C C----- User made a selection
C
C CALL TEXTBOX (RI+16, CI-2, 4, 24, BLACK)
C CALL TEXTBOX (2,RI+15, CI-4, 4, 24, RHWHITE, CYAN)
C STRING = '
C STRING = ' Confirm: delete file?'
C CALL WRITESTRING(STRING,RED,CYAN,RI+16,CI-3,22)
C STRING = '
C STRING = '
C STRING = ' Select:
C CALL WRITESTRING(STRING,RED,CYAN,RI+7,CI-3,12)
C LVES= YES(RI+17,CI+9,.FALSE.)
C CALL TEXTBOX (RI+15, CI-4, 5, 26, BLUE)
C
C IF (LVES) THEN
C C----- Update LISTF
C I1= C
C DO 20 I= 1,NF-1
C   IF (LSTFIL(I).EQ.FILNAM) THEN
C     IDX=I
C     I1= 1
C C----- Delete file from disk
C TRUNAM = '
C TRUNAM = LISTF(IDX)
C OPEN(PRUNIT,FILE=TRUNAM,STATUS='OLD',IOSTAT=IOCK)
C IF (IOCK.EQ.0) CLOSE(PRUNIT,STATUS='DELETE')
C   ENDIF
C   LISTF(I)= LISTF(I+1)
C   LSTFIL(I)= LSTFIL(I+1)
C 20 CONTINUE
C NF= NF -1
C OKAY=.TRUE.
C   ENDIF
C   ENDIF

```

```

C
C C--- Clean up screen dialog area
C FLAG= 1
C CALL LSTLSS(0,RI,CI,HDR,LSTFIL,NF,'SEL',EVTP,FILNAM, FLAG)
C RETURN
C
C END
C CEND:RMVFI*****
C CEOF:PP:NFI*****

```



```

INTRINSIC CHAR, INT, MAX, MIN
C C---LSTLSS-----
C
C----- Save entry page so it can be restored on exit
OAPAG = GETACTIVEPAGE()
IF (OAPAG.NE.DPAG) THEN
  OAPAG = SETACTIVEPAGE(DPAG)
  STATUS = SETCOLOR(BLUE)
  CALL CLEARSCREEN(SGCLEARSCREEN)
END:IF
LSTWID= 0
DO 10 I= 1,LI
10  LSTWID= MAX(LSTWID,NCHARS(LIST(I)))
   LSTWID= MAX(LSTWID,NCHARS(HDR))
C
C----- Clear the display if requested
IF (FLAG.EQ.1) THEN
  CALL TEXTBOX(RI,CI,SEGLN+5,LSTWID+5,BLUE)
  RETURN
END IF
C
C----- Initialize for segment display
L:LAST= INT((LI-1)/SEGLN)*SEGLN +1      : First line of last segment
AROW= R+2
IF(LAST=L:LAST+AROW-1)
  AROW= (LSTWID+4)/2 + CI -1
  AROW= R+SEGLN+3
C
C----- Set up initial pointer/segment display
DO 30 L= 1,LI
30  L= L+1
   L:= INT((L-1)/SEGLN)*SEGLN +1
   H:DX= L
   H:AW= L - LI +AROW +1
   GOTO 31
END:IF
CONTINUE
L:= 1
H:DX= 1
H:AW= AROW +1
31  CONTINUE
C
C----- Set variables before entering get loop
GKOPT = KEYOPT
IF (MOUSE) THEN
  GKOPT = 5
  LEFTUT = .FALSE.
  OLDFLD = AROW - HROW + 1
END IF
C
C----- Put up the background and header
CALL TEXTBOX(RI+1,CI+1,SEGLN+4,LSTWID+4,BLACK)
CALL TEXTBOX(RI,CI,SEGLN+4,LSTWID+4,CYAN)
CALL TEXTBOX(2,RI,CI,SEGLN+4,LSTWID+4,H:WHITE,CYAN)
STRING= HDR
CALL WRITESTRING(STRING,H:WHITE,CYAN,RI+1,CI+2,LSTWID)
OVPAG = SERVISUALPAGE(GETACTIVEPAGE())
C
C----- Display a segment of file starting at LI
100 CONTINUE
C----- Write the Ptr Up unless first segment

```



```

C----- PURPOSE: List contents of a list.
C Allow user to select multiple entries.
C
C----- Inputs: IPAG, page for display
C MLIM, defines window for display of file contents
C HDR, one-line heading
C LIST, list of names (dimension fixed)
C L1, length of active list
C MAXSEL, maximum number of entries that may be selected
C DATPTR, current pointers to LIST (changed in MULLIST)
C NUMSEL, current length of DATPTR (changed in MULLIST)
C
C----- Outputs: EVTP, = CANS, DONE
C DATPTR, new ptrs to LIST
C NUMSEL, new length of DATPTR
C
C INCLUDE 'GRAPH.FD'
C INCLUDE 'COMMBK.CBK'
C INCLUDE 'FMUSE.CBK'
C INCLUDE 'VIDENV.CBK'
C INCLUDE 'VIEWBK.CBK'
C
C CHARACTER HDR*(*), LIST(*)*(*), EVTP*4
C CHARACTER EVTP*4, STRING*80
C INTEGER*2 IPAG, MLIM(4), L1, MAXSEL, DATPTR(*), NUMSEL
C INTEGER FLIM(2), 4), NFID, MFID, GKOPT
C INTEGER*2 L1, L2, L, L1, L2, LSEG
C INTEGER*2 CPAG, IP, JP, IR, IC, WID, IPU, JPU, PCH, IPIAST, LILAST
C INTEGER*2 INDEX, CURIDX, STATUS, TXCOLOR, BKCOLOR
C LOGICAL SELECT, FIX, YES
C
C EXTERNAL LNMUSD, WRITESTR-NG, WRITECHARS, GETKEYVALUE, TXTEOR
C EXTERNAL TEXTBOX, EVTYP, MEFID, SHOWMUSE, HIDEHOUSE, YES
C
C *****MULLIST*****
C
C----- Save entry page so it can be restored on exit
C CPAG = SETACTIVEPAGE(IPAG)
C IF (CPAG.NE.IPAG) THEN
C STATUS = SETBKCOLOR(BLUE)
C STATUS = SETVISUALPAGE(IPAG)
C ENDIF
C
C----- Display format: shaded box from row MLIM(1) to MLIM(3),
C col MLIM(2) to MLIM(4).
C
C Line 1: blank (row MLIM(1))
C 2: header line (HDR)
C 3: Ptr Up is centered (row MLIM(1)+2)
C 4: start of list segment (col MLIM(2)+3)
C n: end of list segment
C n+1: Ptr_Dn is centered (row MLIM(3))
C
C----- Initialize for segment display
C (IPU,JPU for Ptr Up, IP,JP for selector ptr.)
C LSEG= MLIM(3)-MLIM(1)-3
C LILAST= ((L1-1)/LSEG)*LSEG+1 ! First line of last segment
C WID= MLIM(4)-MLIM(2)+1
C IPU= MLIM(1)+2
C IPIAST= L1-LILAST+IPU+1
C JPU= (MLIM(4)-MLIM(2))/2 +MLIM(2)

```

```

C JP= MLIM(2)+1
C
C SECTION REMOVED...
C
C L1= 1
C IP= IPU+1
C
C----- Put up the background and header
C CALL TEXTBOX(MLIM(1)+1,MLIM(2)+1,MLIM(3)+1-MLIM(1),WID,BLACK)
C CALL TEXTBOX(MLIM(1),MLIM(2),MLIM(3)+1-MLIM(1),WID,CYAN)
C
C CALL TEXTBOX(R1+1,C1+1,NOPTS-3,LSTWID+4,BLACK) ! backgrd
C CALL TEXTBOX(R1,C1,NOPTS+3,LSTWID+4,CYAN) ! backgrd
C CALL TXTEOR(L1,R1,C1,NOPTS+3,LSTWID-4,hiwhite,CYAN) ! border
C
C STRING= HDR
C CALL WRITESTR-NG(STRING,BLACK,CYAN,MLIM(1)+1,MLIM(2),WID)
C
C----- Set variables before entering loop
C GKOPT = KEYOPT
C IF (MOUSE) THEN
C GKOPT = 5 ! Don't wait
C LFTBUT = .FALSE. ! mouse flags
C OLDFID = L1+1
C END IF
C
C----- Display a segment of file starting at L1
C 100 CONTINUE
C----- Write the Ptr Up unless first segment
C PCH= 30
C IF (L1.EQ.1) PCH= 32
C CALL WRITESTR-NG(CHAR(PCH),RED,CYAN,IPU,JPU,1)
C----- Write the list data
C L2= MIN(L1-1,LSEG,L1)
C IR= IPU
C IC= MLIM(2)+3
C DO 110 L= L1,L2
C IR= IR+1
C STRING= LIST(L)
C Check to see if the string is currently selected
C TXCOLOR = BLACK
C BKCOLOR = CYAN
C DO 105 INDEX = 1, NUMSEL
C IF (DATPTR(INDEX).EQ. L) THEN
C TXCOLOR = HIWHITE
C GOTO 106
C 105 CONTINUE
C 106 CONTINUE
C DO 120 L= L2+1,L1-1-LSEG
C IR= IR+1
C 120 CALL WRITECHARS(' ',BLACK,CYAN,IR,MLIM(2),WID)
C----- Write the Ptr_Dn unless last segment
C PCH= 31
C IF (L2.EQ.L1) PCH= 32
C CALL WRITESTR-NG(CHAR(PCH),RED,CYAN,MLIM(3),JPU,1)
C----- Display the pointer
C CALL WRITESTR-NG(CHAR(16),RED,CYAN,IP,JPU,1)
C----- Display the mouse after drawing menu
C IF (MOUSE) CALL SHOWMOUSE

```

```

C----- Define the menu field array
NFLD = L2 - L1 + 2
DO I0 L=1, NFLD
  FLIM(L,1) = MLIM(1) + 2 + (L-1)
  FLIM(L,2) = MLIM(2)
  FLIM(L,3) = MLIM(1) + 2 + (L-1)
  FLIM(L,4) = MLIM(4)
CONTINUE
10  NFLD = NFLD + 1
    FLIM(NFLD,1) = MLIM(3)
    FLIM(NFLD,2) = MLIM(2)
    FLIM(NFLD,3) = MLIM(3)
    FLIM(NFLD,4) = MLIM(4)
C
C----- Get some input
150 CALL GETKEYVALUE(GKOPT,11,12)
    EVTP= EVTP(11,12)
    CALL NFLD(2,3,FLIM,NFLD,NFLD,EVTP)
C----- Crack the user input
IF (EVTP.EQ.'MOVE') THEN
  IF (MOUSE) CALL HIDEHOUSE()
  CALL WRITECHARS(' ',CYAN,CYAN,IP,JP,1) ! Erase Ptr
  IF (L2.EQ.71) THEN
    L1= 1
    IP= IPU+1
  ELSEIF (L2.EQ.79) THEN
    L1= L1LAST
    IP= IPLAST
  ELSEIF (L2.EQ.73) THEN
    IF (L1.EQ.1) THEN
      IP= IPU+1
    ELSE
      L1= L1 - LSEG
      IP= MLIM(3) - 1
    ENDIF
  ELSEIF (L2.EQ.81) THEN
    IF (L2.LT.L1) THEN
      L1= L1 + LSEG
      IP= IPU + 1
    ELSE
      IP= IPLAST
    ENDIF
  ELSEIF (L2.EQ.72) THEN
    IF (IP.EQ.(IPU+1)) THEN
      IF (L1.GT.1) THEN
        L1= L1 - LSEG
        IP= MLIM(3) - 1
      ENDIF
    ELSE
      IP= IP-1
    ENDIF
  ELSEIF (L2.EQ.80) THEN
    IF (L2.LT.L1) THEN
      IF (IP.LT.MLIM(3)-1) THEN
        IP= IP+1
      ELSE
        L1= L1 + LSEG
        IP= IPU+1
      ENDIF
    ELSE
      IF (IP.LT.IPLAST) IP= IP+1
    ENDIF
  ENDIF
  ! Up arrow is 1st field
  ! Down arrow is last field
ENDIF
ELSEIF ( (L2.EQ.0).AND.MOUSE ) THEN
  IF ( (NFLD.NE.1).AND.(NFLD.NE.NFLD) ) IP = IPU + NFLD - 1
ENDIF
ELSEIF (EVTP.EQ.'CRET') THEN
  C----- If mouse 'CRET' is on 1st or last field, then page
  IF (NFLD.EQ.1) THEN HIDEHOUSE()
  IF (MOUSE) CALL HIDEHOUSE()
  CALL WRITECHARS(' ',BLACK,CYAN,IP,JP,1) ! Erase Ptr
  IF (L1.EQ.1) THEN
    IP= IPU+1
  ELSE
    L1= L1 - LSEG
    IP= MLIM(3) - 1
    LEFTBUT = .FALSE.
  ENDIF
  GOTO 100
ELSEIF (NFLD.EQ.NFLD) THEN
  IF (MOUSE) CALL HIDEHOUSE()
  CALL WRITECHARS(' ',BLACK,CYAN,IP,JP,1) ! Erase Ptr
  IF (L2.LT.L1) THEN
    L1= L1 + LSEG
    IP= IPU + 1
  ELSE
    IP= IPLAST
  ENDIF
  GOTO 100
ENDIF
C
C----- Process the CRET if not a mouse page
C See if entry is currently selected
CURIDX = L1+IP-IPU
SELECT = .FALSE.
DO 200 INDEX = 1, NUMSEL
  IF ( DATPTR(INDEX) .EQ. CURIDX ) THEN
    SELECT = .TRUE.
  ENDIF
200 CONTINUE
C Check current select status...
IF ( SELECT ) THEN
  C Remove item from index list and decrement NUMSEL
  FIX = .FALSE.
  DO 250 INDEX = 1, NUMSEL
    IF ( DATPTR(INDEX) .EQ. CURIDX ) THEN
      FIX = .TRUE.
    ENDIF
  ENDIF
  IF ( FIX ) THEN
    IF ( INDEX .EQ. NUMSEL ) THEN
      DATPTR(INDEX) = 0
    ELSE
      DATPTR(INDEX) = DATPTR(INDEX+1)
    ENDIF
  ENDIF
250 CONTINUE
  NUMSEL = NUMSEL - 1
  ELSE
C----- Item is not currently selected, attempt to add to list
  IF ( NUMSEL .LT. MAXSEL ) THEN
    NUMSEL = NUMSEL + 1
  ENDIF
ENDIF

```



```

LEGEND = .TRUE.
COUNT = 0
N=0
TIM=0
IR = GFXVC.NUM*EXTROWS - 3
GFACTV= .FALSE.
IOPT1= 1
IOPT2 = 1
J=0
GKOPT = 5
STAT = .FALSE.
C ---- Set up new screen and draw current system graph
DO 5 I=1,200
  VECG(I)=0.0
  VECM(I)=0.0
5 CONTINUE
C
C CALL GETRES(N)
C .Get data limits
C IF (VRMIN.LT.0.0.AND.VRCMAX.GT.0) VRMIN=0.0
C IF (VRMIN.LT.0.0.AND.VRCMAX.GT.0) VRMIN=0.0
C
C ---- Draw first with color by power type and a width of 1
HOLDW=WIDTH
HOLDV=VARWID
HOLDVC=VARCOL
HOLDVT=VARTIM
WIDTH=1
VARWID=0
VARCOL=4
VARTV=0
ARG= 'GFX'
CALL SETMOD(ARG)
IPAG= 0
! Set graphics mode for monitor

C ---- Draw the frame first to avoid blinking
CALL SETVIEWPORT(XMNPXL, YMNPXL, CXXPXL, YMXPXL)
IF (FPMVIS) THEN
  STATUS = SETCOLOR(HIGH*TE)
  STATUS = RECTANGLE(SGBORDER, XMNPXL, YMNPXL, CXXPXL, YMXPXL)
ENDIF
CALL SETVIEWPORT(XMNPXL, YMNPXL, CXXPXL, YMXPXL)
CALL CRLGND(J)
CALL WLGND(J)
CALL TMLGND(J, LEGEND, IR)
  WIDTH = HOLDW
  VARWID = HOLDV
  VARCOL = HOLDVC
  VARTV = HOLDVT
  ARG= 'GFX'
  CALL SETMOD(ARG)
  IPAG= 0
  ! Set graphics mode for monitor
  CALL SETVIEWPORT(XMNPXL, YMNPXL, CXXPXL, YMXPXL)
  IF (LEGEND) THEN
    CALL CRLGND(J)
    CALL WLGND(J)
  END IF
  CALL GETRES(N)
  END IF
CONTINUE
3001
10 ARG= 'GFX'
  CALL SETMOD(ARG)
  ! Set graphics mode for monitor

IPAG= 0
CALL SETVIEWPORT(XMNPXL, YMNPXL, CXXPXL, YMXPXL)
IF (J.NE.0) CALL TMLGND(J, LEGEND, IR)
CALL SYSDRV(IPAG, LEGEND, LABEL)
COUNT = COUNT + 1
J=J+1
2001 CONTINUE
C
C IF (VARTIM.EQ.0) THEN
C
C ---- Get user selection
CALL SPKEYL
CONTINUE
  IOPTH= IOPT
  IOPTH2= IOPT2
  IF (HOLDVT.EQ.1) CALL MMULIN(IPAG,LIST2,LL,IR,IOPT2)
  IF (HOLDVT.EQ.0) CALL MMULIN(IPAG,LIST,LL,IR,IOPT)
C
C ---- Take action on the request
  IF ((IOPT.EQ.0.AND.HOLDVT.EQ.0).OR.
    + ((IOPT2.EQ.0.AND.HOLDVT.EQ.1)) THEN ! Help
    ARG= 'TX'
    CALL SETMOD(ARG)
    FILNAM= 'MSGRED.HLP'
    CALL SHOLF(IPAG,FILNAM)
    IOPTH= IOPTH
    IOPTH2 = IOPTH2
    ELSEIF ((IOPT.EQ.1.AND.HOLDVT.EQ.0).OR.
    + ((IOPT2.EQ.1.AND.HOLDVT.EQ.1)) THEN ! Forward or start
    MESSAGE= 'You have reached the final time.'
    CALL WRITESTRING(MESSAG,MAGENTA,BLACK,IR+25,33)
    GOTO 100
  ELSE
    IF (HOLDVT.EQ.1) CALL SETTIM(0,0,0,0)
    IF (J.EQ.1) GOTO 11
    CALL WRITESTRING(' ',MAGENTA,BLACK,IR+2,28,27)
    CALL GETRES(N)
  END IF
  ELSEIF (IOPT.EQ.2.AND.HOLDVT.EQ.0) THEN ! Back
    COUNT = COUNT - 2
    N = N - 2
    IF (N.LT.0) THEN
      N = 0
      MESSAGE = 'You are at the initial time.'
      CALL WRITESTRING(MESSAG,MAGENTA,BLACK,IR+2,28,27)
    ELSE
      CALL WRITESTRING(' ',MAGENTA,BLACK,IR+2,25,33)
    END IF
    IF (COUNT.LT.0) COUNT = 0
    CALL GETRES(N)
    J = J - 2
  IF (J.LT.1) J = 1
  ELSEIF ((IOPT.EQ.3.AND.HOLDVT.EQ.0).OR.
    + ((IOPT2.EQ.2.AND.HOLDVT.EQ.1)) THEN ! View
    CALL GFZOOM(LEGEND, LABEL)
    GOTO 100
  ELSEIF ((IOPT.EQ.4.AND.HOLDVT.EQ.0).OR.
    + ((IOPT2.EQ.3.AND.HOLDVT.EQ.1)) THEN ! Redraw
    CALL SETMOD('GFX')
    IPAG=0
    CALL SETVIEWPORT(XMNPXL, YMNPXL, CXXPXL, YMXPXL)

```

```

IF (LEGEND) THEN
  CALL CRGEND(0)
  CALL WLGEND(0)
END IF
I = -2
J = J-1
CALL TMLGND(1, LEGEND, IR)
CALL SYSDW(IPAG, LEGEND, LABEL)
Draw the frame last to get a nice solid border
IF (LEGEND) THEN
  CALL SETVIEWPORT(XMNPXL, YMNPXL, CXXPXL, YMAXPXL)
  IF (FRMVIS) THEN
    STATUS=SETCOLOR(HIWHITE)
    STATUS=RECTANGLE(SGBORDER, XMNPXL, YMNPXL, XMXPXL, YMAXPXL)
  ENDIF
ELSE
  CALL SETVIEWPORT(XMNPXL, YMNPXL, CXXPXL, YMAXPXL)
  IF (FRMVIS) THEN
    STATUS=SETCOLOR(HIWHITE)
    STATUS=RECTANGLE(SGBORDER, XMNPXL, YMNPXL, CXXPXL, YMAXPXL)
  ENDIF
ENDIF
END IF
GOTO 100
ELSEIF ((IOPT.EQ.5.AND.HOLDVT.EQ.0).OR.
+ (IOPT2.EQ.4.AND.HOLDVT.EQ.1)) THEN . Options
ARG= 'TXT'
CALL SETMOD(ARG)
CALL GFOPTS(IPAG, LEGEND, LABEL)
J=J-1
CALL SETMOD('GFX')
IPAG=C
CALL SETVIEWPORT(XMNPXL, YMNPXL, CXXPXL, YMAXPXL)
IF (LEGEND) THEN
  CALL CRGEND(0)
  CALL WLGEND(0)
END IF
CALL TMLGND(-2, LEGEND, IR)
CALL SYSDW(IPAG, LEGEND, LABEL)
Draw the frame last to get a nice solid border
IF (LEGEND) THEN
  CALL SETVIEWPORT(XMNPXL, YMNPXL, CXXPXL, YMAXPXL)
  IF (FRMVIS) THEN
    STATUS=SETCOLOR(HIWHITE)
    STATUS=RECTANGLE(SGBORDER, XMNPXL, YMNPXL, XMXPXL, YMAXPXL)
  ENDIF
ELSE
  CALL SETVIEWPORT(XMNPXL, YMNPXL, CXXPXL, YMAXPXL)
  IF (FRMVIS) THEN
    STATUS=SETCOLOR(HIWHITE)
    STATUS=RECTANGLE(SGBORDER, XMNPXL, YMNPXL, CXXPXL, YMAXPXL)
  ENDIF
ENDIF
GOTO 100
ELSEIF ((IOPT.EQ.6.AND.HOLDVT.EQ.0).OR.
+ (IOPT2.EQ.5.AND.HOLDVT.EQ.1)) THEN ! List
ARG= 'TXT'
CALL SETMOD(ARG)
CALL GFLIST(IPAG)
J=J-1
CALL SETMOD('GFX')
IPAG=0
CALL SETVIEWPORT(XMNPXL, YMNPXL, CXXPXL, YMAXPXL)
IF (LEGEND) THEN
  CALL CRGEND(0)
  CALL WLGEND(0)
END IF
CALL TMLGND(-2, LEGEND, IR)
CALL SYSDW(IPAG, LEGEND, LABEL)
Draw the frame last to get a nice solid border
IF (LEGEND) THEN
  CALL SETVIEWPORT(XMNPXL, YMNPXL, CXXPXL, YMAXPXL)
  IF (FRMVIS) THEN
    STATUS=SETCOLOR(HIWHITE)
    STATUS=RECTANGLE(SGBORDER, XMNPXL, YMNPXL, XMXPXL, YMAXPXL)
  ENDIF
ELSE
  CALL SETVIEWPORT(XMNPXL, YMNPXL, CXXPXL, YMAXPXL)
  IF (FRMVIS) THEN
    STATUS=SETCOLOR(HIWHITE)
    STATUS=RECTANGLE(SGBORDER, XMNPXL, YMNPXL, CXXPXL, YMAXPXL)
  ENDIF
ENDIF
GOTO 100
ELSEIF ((IOPT.EQ.7.AND.HOLDVT.EQ.0).OR.
+ (IOPT2.EQ.6.AND.HOLDVT.EQ.1)) THEN ! File
CALL SGZFL
MESSAGE = 'This option is not yet available'
CALL WRITESTRING(MESSAG,MAGENTA,BLACK,IR+2,25,33)
READ(*,*)
MESSAGE = ' '
CALL WRITESTRING(MESSAG,MAGENTA,BLACK,IR+2,25,33)
GOTO 100
ELSEIF ((IOPT.EQ.8.AND.HOLDVT.EQ.0).OR.
+ (IOPT2.EQ.7.AND.HOLDVT.EQ.1)) THEN ! Return
GOTO 800
ENDIF
ELSEIF (VARTIM.EQ.1) THEN
IF (J.EQ.2) THEN
  CALL TEXTBOX(,1,1,80,BLACK)
  CALL WRITESTRING('ESC: Cancel',GREEN,BLACK,IR+4,35,11)
END IF
END IF
IF (COUNT.EQ.NS) THEN
  VARTIM = 0
  HOLDVT = 0
  STAT = .TRUE.
END IF
CALL GETTIM(IHR, IMIN, ISEC, I100th)
IF (ISEC.GT.SEC) THEN
  ELSE IF (I100th.EQ.SEC) THEN
  IF (I100th.IT.HUND) GOTO 4001
  ELSE IF (ISEC.IT.SEC) THEN
  GOTO 4001
END IF
CALL SETTIM(0,0,0,0)
CALL GETRES(N)
CALL SETREVALUE(GKOPT,I1,I2)
EVTP = EVTYP(I1,I2)
IF (EVTP.EQ.'CANC') GOTO 800
IF (J.EQ.1) GOTO 11
END IF
C

```


20

30

[illegible][illegible]

```

      ELSE
      C----- Bond is 'steep' rel to node name box
      XB(N)= XB(N) + (DXP/DYP)*DYN
      YB(N)= YB(N) +DYN
      ENDIF
      C-----
      150 CONTINUE
      C -
      C - To ensure that the endpoints are at exact pixel locations
      CALL GETVIEWCOORD W(XB(1),YB(1),XY)
      CALL GETWINDOWCOORD(XY,XCOORD,XY,YCOORD,WXY)
      XB(1) = WXY.WX
      YB(1) = WXY.WY
      CALL GETVIEWCOORD W(XB(2),YB(2),XY)
      CALL GETWINDOWCOORD(XY,XCOORD,XY,YCOORD,WXY)
      XB(2) = WXY.WX
      YB(2) = WXY.WY
      C
      CALL GETWINDOWCOORD(0,0,WXY)
      REFV=WXY.WX
      REFY=WXY.WY
      CALL GETWINDOWCOORD(1,1,WXY)
      D:SEY = ABS(WXY.WX-REFV)
      D:SPY = ABS(WXY.WY-REFY)
      C
      C----- Prepare for arrowheads and causal stroke
      DSINE= DISAR*DYP/DIS
      DCOSE= DISAP*DXP/DIS
      INC(1)=DISFX/DISAR
      INC(2)=DISPY/DISAR
      C
      CALL DRAROW(DSINB,DCOSB,INC,XB,YB,BTPCH,NOD,ARWPTS,KPTS)
      C
      C----- Draw the connector label
      CALL DRCLBL(XB,YB,BLAB)
      C
      C----- Establish the causal information for this bond
      IF (BTPCH.NE.'B') RETURN
      IF (.NOT.CLCFLG) RETURN
      CALL CONNZI(BLAB,IBOND)
      IF (ENDCAL(1BOND).EQ.0) RETURN
      IF (NOD(1).GT.0) THEN
      IF (.NOT.GFXFIL) THEN
      STATUS = SETCOLOR(ICLR)
      ENDIF
      C----- See if tail segment needs causal stroke
      IF (SUBNOD(NOD(1),BNDCAU(1BOND))) THEN
      XB(1)= XB(1) -DSINB*0.5
      YB(1)= YB(1) +DCOSB*0.5
      XB(2)= XB(1) +DSINB
      YB(2)= YB(1) -DCOSB
      IF (GFXFIL) THEN
      CALL GFXLIN('DXF',XB(1),YB(1),XB(2),YB(2),ICLR)
      ELSE
      CALL MOVETO W(XB(1),YB(1),WXY)
      STATUS = LINETO W(XB(2),YB(2))
      ENDIF
      RETURN
      ENDIF
      IF (NOD(2).GT.0) THEN
      C----- See if head segment needs causal stroke

```



```

WXY(2).WX = XB(2) - 0.866*DCOSB + DSINB*INC(1)
WXY(2).WY = YB(2) - 0.866*DSINB - DCOSB*INC(2)
WXY(3).WX = XB(2) - 0.866*DCOSB + 0.5*DSINB + DSINB*INC(1)
WXY(3).WY = YB(2) - 0.866*DCOSB + 0.5*DSINB - DCOSB*INC(2)
WXY(4).WX = XB(2)
WXY(4).WY = YB(2)
WXY(5).WX = XB(2) - 0.866*DCOSB - 0.5*DSINB - DSINB*INC(1)
WXY(5).WY = YB(2) - 0.866*DSINB + 0.5*DCOSB + 3*DCOSB*INC(2)
WXY(6).WX = XB(2) - 0.866*DCOSB - DSINB*INC(1)
WXY(6).WY = YB(2) - 0.866*DSINB + DCOSB*INC(2)
WXY(7).WX = XB(1) - DSINB*INC(1)
WXY(7).WY = YB(1) + DCOSB*INC(2)
ELSE IF (WIDTH.EQ.4) THEN
  WXY(1).WX = XB(1) + DSINB*INC(1)
  WXY(1).WY = YB(1) - 3*DCOSB*INC(2)
  WXY(2).WX = XB(2) - 0.866*DCOSB + 3*DSINB*INC(1)
  WXY(2).WY = YB(2) - 0.866*DSINB - 3*DCOSB*INC(2)
  WXY(3).WX = XB(2) - 0.866*DCOSB + 0.5*DSINB + 3*DSINB*INC(1)
  WXY(3).WY = YB(2) - 0.866*DSINB + 0.5*DCOSB - 3*DCOSB*INC(2)
  WXY(4).WX = XB(2)
  WXY(4).WY = YB(2)
  WXY(5).WX = XB(2) - 0.866*DCOSB - 0.5*DSINB - 4*DSINB*INC(1)
  WXY(5).WY = YB(2) - 0.866*DSINB + 0.5*DCOSB + 4*DCOSB*INC(2)
  WXY(6).WX = XB(2) - 0.866*DCOSB - DSINB*INC(1)
  WXY(6).WY = YB(2) - 0.866*DSINB + 4*DCOSB*INC(2)
  WXY(7).WX = XB(1) - 4*DSINB*INC(1)
  WXY(7).WY = YB(1) + 4*DCOSB*INC(2)
END IF
ENDIF

C ---- For cases where there are interior points
C IF (MOD(2).EQ.0) THEN
  DO 201 J = 1,7
    ARMPTS(KPTS,J,1) = WXY(J).WX
    ARMPTS(KPTS,J,2) = WXY(J).WY
    CONTINUE
  201 ELSE IF (MOD(1).EQ.0) THEN
    DO 401 J = 1,7
      ARMPTS(KPTS,J,1) = WXY(J).WX
      ARMPTS(KPTS,J,2) = WXY(J).WY
    401 CONTINUE
  C ---- Defining lines for arrow.
  X1 = WXY(1).WX
  X2 = WXY(2).WX
  Y1 = WXY(1).WY
  Y2 = WXY(2).WY
  X3 = WXY(7).WX
  X4 = WXY(6).WX
  Y3 = WXY(7).WY
  Y4 = WXY(6).WY
  X5 = WXY(7).WX
  X6 = WXY(1).WX
  Y5 = WXY(7).WY
  Y6 = WXY(1).WY
  C ---- Draws connector when there are no interior points.
  STATUS = SETCOLOR(ICLR)
  STATUS = POLYGON.W( SGFILLINTERIOR, WXY, 7)
  IF (WIDTH.NE.1.AND.(DSINB.EQ.0.OR.DCOSB.EQ.0)) THEN
    STATUS = SETCOLOR(ICLR)
    CALL MOVEPTO.W(X1,Y1,WXY)
    STATUS = LINETO.W(X2,Y2)
    CALL MOVEPTO.W(X3,Y3,WXY)

```



```

C and change MNSTN accordingly in data.
C *NOTE*
C The order that bonds appear in the junction node-bond group affect
C the order of assignment on a strongly determined junction. The
C last listed bond is first assigned. Careful ordering in the line
C code may reduce the temporary storage requirements (the M stacks).
C
C Careful use of level zero can alleviate problems of overflowing
C NTEMP stack since NTEMP is not used at levels zero and one.
C IERRF= 200 input garbage
C 210 no such node attached to bond
C 220 must be E or F
C 240 source conflict
C 241 derivative causality
C 250 overflow on the M stacks
C 260 overdetermined function
C 270 overdetermined junction (ISJUNK)
C 290 overflow on stack N
C 300 causal loop
C 310 causal loop
C 320 causal loop
C 330 causal loop
C 340 causal loop
C 350 causal loop
C 360 causal loop
C 370 causal loop
C 380 causal loop
C 390 causal loop
C 400 causal loop
C 410 causal loop
C 420 causal loop
C 430 causal loop
C 440 causal loop
C 450 causal loop
C 460 causal loop
C 470 causal loop
C 480 causal loop
C 490 causal loop
C 500 causal loop
C 510 causal loop
C 520 causal loop
C 530 causal loop
C 540 causal loop
C 550 causal loop
C 560 causal loop
C 570 causal loop
C 580 causal loop
C 590 causal loop
C 600 causal loop
C 610 causal loop
C 620 causal loop
C 630 causal loop
C 640 causal loop
C 650 causal loop
C 660 causal loop
C 670 causal loop
C 680 causal loop
C 690 causal loop
C 700 causal loop
C 710 causal loop
C 720 causal loop
C 730 causal loop
C 740 causal loop
C 750 causal loop
C 760 causal loop
C 770 causal loop
C 780 causal loop
C 790 causal loop
C 800 causal loop
C 810 causal loop
C 820 causal loop
C 830 causal loop
C 840 causal loop
C 850 causal loop
C 860 causal loop
C 870 causal loop
C 880 causal loop
C 890 causal loop
C 900 causal loop
C 910 causal loop
C 920 causal loop
C 930 causal loop
C 940 causal loop
C 950 causal loop
C 960 causal loop
C 970 causal loop
C 980 causal loop
C 990 causal loop
C 1000 causal loop
C 1010 causal loop
C 1020 causal loop
C 1030 causal loop
C 1040 causal loop
C 1050 causal loop
C 1060 causal loop
C 1070 causal loop
C 1080 causal loop
C 1090 causal loop
C 1100 causal loop
C 1110 causal loop
C 1120 causal loop
C 1130 causal loop
C 1140 causal loop
C 1150 causal loop
C 1160 causal loop
C 1170 causal loop
C 1180 causal loop
C 1190 causal loop
C 1200 causal loop
C 1210 causal loop
C 1220 causal loop
C 1230 causal loop
C 1240 causal loop
C 1250 causal loop
C 1260 causal loop
C 1270 causal loop
C 1280 causal loop
C 1290 causal loop
C 1300 causal loop
C 1310 causal loop
C 1320 causal loop
C 1330 causal loop
C 1340 causal loop
C 1350 causal loop
C 1360 causal loop
C 1370 causal loop
C 1380 causal loop
C 1390 causal loop
C 1400 causal loop
C 1410 causal loop
C 1420 causal loop
C 1430 causal loop
C 1440 causal loop
C 1450 causal loop
C 1460 causal loop
C 1470 causal loop
C 1480 causal loop
C 1490 causal loop
C 1500 causal loop
C 1510 causal loop
C 1520 causal loop
C 1530 causal loop
C 1540 causal loop
C 1550 causal loop
C 1560 causal loop
C 1570 causal loop
C 1580 causal loop
C 1590 causal loop
C 1600 causal loop
C 1610 causal loop
C 1620 causal loop
C 1630 causal loop
C 1640 causal loop
C 1650 causal loop
C 1660 causal loop
C 1670 causal loop
C 1680 causal loop
C 1690 causal loop
C 1700 causal loop
C 1710 causal loop
C 1720 causal loop
C 1730 causal loop
C 1740 causal loop
C 1750 causal loop
C 1760 causal loop
C 1770 causal loop
C 1780 causal loop
C 1790 causal loop
C 1800 causal loop
C 1810 causal loop
C 1820 causal loop
C 1830 causal loop
C 1840 causal loop
C 1850 causal loop
C 1860 causal loop
C 1870 causal loop
C 1880 causal loop
C 1890 causal loop
C 1900 causal loop
C 1910 causal loop
C 1920 causal loop
C 1930 causal loop
C 1940 causal loop
C 1950 causal loop
C 1960 causal loop
C 1970 causal loop
C 1980 causal loop
C 1990 causal loop
C 2000 causal loop
C 2010 causal loop
C 2020 causal loop
C 2030 causal loop
C 2040 causal loop
C 2050 causal loop
C 2060 causal loop
C 2070 causal loop
C 2080 causal loop
C 2090 causal loop
C 2100 causal loop
C 2110 causal loop
C 2120 causal loop
C 2130 causal loop
C 2140 causal loop
C 2150 causal loop
C 2160 causal loop
C 2170 causal loop
C 2180 causal loop
C 2190 causal loop
C 2200 causal loop
C 2210 causal loop
C 2220 causal loop
C 2230 causal loop
C 2240 causal loop
C 2250 causal loop
C 2260 causal loop
C 2270 causal loop
C 2280 causal loop
C 2290 causal loop
C 2300 causal loop
C 2310 causal loop
C 2320 causal loop
C 2330 causal loop
C 2340 causal loop
C 2350 causal loop
C 2360 causal loop
C 2370 causal loop
C 2380 causal loop
C 2390 causal loop
C 2400 causal loop
C 2410 causal loop
C 2420 causal loop
C 2430 causal loop
C 2440 causal loop
C 2450 causal loop
C 2460 causal loop
C 2470 causal loop
C 2480 causal loop
C 2490 causal loop
C 2500 causal loop
C 2510 causal loop
C 2520 causal loop
C 2530 causal loop
C 2540 causal loop
C 2550 causal loop
C 2560 causal loop
C 2570 causal loop
C 2580 causal loop
C 2590 causal loop
C 2600 causal loop
C 2610 causal loop
C 2620 causal loop
C 2630 causal loop
C 2640 causal loop
C 2650 causal loop
C 2660 causal loop
C 2670 causal loop
C 2680 causal loop
C 2690 causal loop
C 2700 causal loop
C 2710 causal loop
C 2720 causal loop
C 2730 causal loop
C 2740 causal loop
C 2750 causal loop
C 2760 causal loop
C 2770 causal loop
C 2780 causal loop
C 2790 causal loop
C 2800 causal loop
C 2810 causal loop
C 2820 causal loop
C 2830 causal loop
C 2840 causal loop
C 2850 causal loop
C 2860 causal loop
C 2870 causal loop
C 2880 causal loop
C 2890 causal loop
C 2900 causal loop
C 2910 causal loop
C 2920 causal loop
C 2930 causal loop
C 2940 causal loop
C 2950 causal loop
C 2960 causal loop
C 2970 causal loop
C 2980 causal loop
C 2990 causal loop
C 3000 causal loop
C 3010 causal loop
C 3020 causal loop
C 3030 causal loop
C 3040 causal loop
C 3050 causal loop
C 3060 causal loop
C 3070 causal loop
C 3080 causal loop
C 3090 causal loop
C 3100 causal loop
C 3110 causal loop
C 3120 causal loop
C 3130 causal loop
C 3140 causal loop
C 3150 causal loop
C 3160 causal loop
C 3170 causal loop
C 3180 causal loop
C 3190 causal loop
C 3200 causal loop
C 3210 causal loop
C 3220 causal loop
C 3230 causal loop
C 3240 causal loop
C 3250 causal loop
C 3260 causal loop
C 3270 causal loop
C 3280 causal loop
C 3290 causal loop
C 3300 causal loop
C 3310 causal loop
C 3320 causal loop
C 3330 causal loop
C 3340 causal loop
C 3350 causal loop
C 3360 causal loop
C 3370 causal loop
C 3380 causal loop
C 3390 causal loop
C 3400 causal loop
C 3410 causal loop
C 3420 causal loop
C 3430 causal loop
C 3440 causal loop
C 3450 causal loop
C 3460 causal loop
C 3470 causal loop
C 3480 causal loop
C 3490 causal loop
C 3500 causal loop
C 3510 causal loop
C 3520 causal loop
C 3530 causal loop
C 3540 causal loop
C 3550 causal loop
C 3560 causal loop
C 3570 causal loop
C 3580 causal loop
C 3590 causal loop
C 3600 causal loop
C 3610 causal loop
C 3620 causal loop
C 3630 causal loop
C 3640 causal loop
C 3650 causal loop
C 3660 causal loop
C 3670 causal loop
C 3680 causal loop
C 3690 causal loop
C 3700 causal loop
C 3710 causal loop
C 3720 causal loop
C 3730 causal loop
C 3740 causal loop
C 3750 causal loop
C 3760 causal loop
C 3770 causal loop
C 3780 causal loop
C 3790 causal loop
C 3800 causal loop
C 3810 causal loop
C 3820 causal loop
C 3830 causal loop
C 3840 causal loop
C 3850 causal loop
C 3860 causal loop
C 3870 causal loop
C 3880 causal loop
C 3890 causal loop
C 3900 causal loop
C 3910 causal loop
C 3920 causal loop
C 3930 causal loop
C 3940 causal loop
C 3950 causal loop
C 3960 causal loop
C 3970 causal loop
C 3980 causal loop
C 3990 causal loop
C 4000 causal loop
C 4010 causal loop
C 4020 causal loop
C 4030 causal loop
C 4040 causal loop
C 4050 causal loop
C 4060 causal loop
C 4070 causal loop
C 4080 causal loop
C 4090 causal loop
C 4100 causal loop
C 4110 causal loop
C 4120 causal loop
C 4130 causal loop
C 4140 causal loop
C 4150 causal loop
C 4160 causal loop
C 4170 causal loop
C 4180 causal loop
C 4190 causal loop
C 4200 causal loop
C 4210 causal loop
C 4220 causal loop
C 4230 causal loop
C 4240 causal loop
C 4250 causal loop
C 4260 causal loop
C 4270 causal loop
C 4280 causal loop
C 4290 causal loop
C 4300 causal loop
C 4310 causal loop
C 4320 causal loop
C 4330 causal loop
C 4340 causal loop
C 4350 causal loop
C 4360 causal loop
C 4370 causal loop
C 4380 causal loop
C 4390 causal loop
C 4400 causal loop
C 4410 causal loop
C 4420 causal loop
C 4430 causal loop
C 4440 causal loop
C 4450 causal loop
C 4460 causal loop
C 4470 causal loop
C 4480 causal loop
C 4490 causal loop
C 4500 causal loop
C 4510 causal loop
C 4520 causal loop
C 4530 causal loop
C 4540 causal loop
C 4550 causal loop
C 4560 causal loop
C 4570 causal loop
C 4580 causal loop
C 4590 causal loop
C 4600 causal loop
C 4610 causal loop
C 4620 causal loop
C 4630 causal loop
C 4640 causal loop
C 4650 causal loop
C 4660 causal loop
C 4670 causal loop
C 4680 causal loop
C 4690 causal loop
C 
```

```

N2MX(N)= M
IELST(M)= INDEX('CIREF01TG',NBXTP(N)(2:2))
ENDIF
1 CONTINUE
C----- Set up the bond information
C (B is the bond index, C is the connector index)
B= 0
DO 2 C= 1,INBDS
  IF (BDTP(C)(1:1).EQ.'B') THEN
    B= B+1
    B2CX(B)= C
    IBMX(B,1)= N2MX(IELSBD(C,1))
    IBMX(B,2)= N2MX(IELSBE(C,2))
  ENDIF
2 CONTINUE
C----- Now set up the structural information
NPTR(1)= 1
DO 5 M= 1,NEL
  KB= NPTR(M)-1
  LBC= 0
  DO 4 B= 1,NBD
    IF (IBMX(B,1).EQ.M) THEN
      LBC= LBC+1
      NB:MX(KB+LBC)= B
    ELSEIF (IBMX(B,2).EQ.M) THEN
      LBC= LBC+1
      NB:MX(KB+LBC)= B
    ENDIF
  ENDIF
5 CONTINUE
NPTR(M-1)= NPTR(M) +LBC
6 CONTINUE
C----- Initialize
NBSLCK= 0
DO 10 J= 1,MAXNRF
  IRQUE(J)= 0
  COMPL= .FALSE.
  NBD2=NBD*2
  JENT= 0
  INODE= 0
  KBRND= 0
  NR= 0
  NRB= 0
  DO 15 J= 1,NBD2
    ICMX(J)= 1
    DO 20 J= 1,INBDS
      BNDCAU(J)= 0
    ENDIF
  ENDIF
15 CONTINUE
C----- Level one, process Se and Sf elements.
LEVEL= 1
DO 190 J= 1,NEL
  ITP=IELST(J)
  IF (ITYP.EQ.4) THEN
    IT= 6
  ELSEIF (ITYP.EQ.5) THEN
    IT= 3
  ELSE
    IT= 0
  ENDIF
  GOTO 190
ENDIF
NNOD=J

```

```

K=NPTR(J)
NBRND=K
CALL EXTEND(NNOD,NBRND,IT)
IF (IERRF.NE.0) GOTO 900
190 CONTINUE
IF (SPEAKF) THEN
  MESSAGE= ' Source (Se,Sf) causality has been assigned.'
  CALL LMSGD(24)
ENDIF
CALL CSDONE(COMPL)
IF (COMPL) GOTO 800
C----- Level two (C, I)
LEVEL= 2
DO 280 J= 1,NEL
  IF (IELST(J).EQ.1) THEN
    IT= 6
  ELSEIF (IELST(J).EQ.2) THEN
    IT= 3
  ELSE
    GOTO 280
  ENDIF
  NP1=NPTR(J)
  NP2=NPTR(J+1)-1
  DO 270 K=NP1,NP2
    IF (ICMX(K).EQ.1) THEN
      CALL TWPASS(G,K,IT)
      IF (IERRF.NE.0) GOTO 900
    ENDIF
  ENDIF
270 CONTINUE
280 CONTINUE
IF (SPEAKF) THEN
  MESSAGE= ' Storage (C,I) causality has been assigned.'
  CALL LMSGD(24)
ENDIF
CALL CSDONE(COMPL)
IF (COMPL) GOTO 800
C----- Level three ( R )
LEVEL= 3
IERRF= 0
C----- Identify the R-fields
CALL IDNTR
IF (.NOT.OKAY) THEN
  CALL LMSGD(24)
  COMPL= .FALSE.
  GOTO 900
ENDIF
IF (AUTO) THEN
  C----- Assign causality automatically
  CALL RCSASS
  IF (.NOT.OKAY) THEN
    CALL LMSGD(24)
  ENDIF
  CALL CSDONE(COMPL)
  IF (.NOT.COMPL) THEN
    MESSAGE= ' Try User-assigned causality.'
    CALL LMSGD(24)
  ENDIF
  GOTO 800
ELSE
  C----- Assign causality with user help

```



```

C----- PROGRAMMING: T. ZHOU
C
C----- IDENTIFIES THE R-FIELDS AFTER THE LEVEL TWO OF CAUSALITY
C ASSIGNMENT.
C
C INCLUDE 'SIZEBK.CBK'
C INCLUDE 'CAUSEK.CBK'
C INCLUDE 'COMBK.CBK'
C INCLUDE 'GREDBK.CBK'
C INCLUDE 'REFDBK.CBK'
C INCLUDE 'SYGEBK.CBK'
C INCLUDE 'V:EWBK.CBK'
C
C INTEGER JREFLG(50), NB:FN(10), NB:FB(10), NP1, NP2
C INTEGER NBC, I, M, J, KNOD, IBOND, IWAY, NI, NUBED, N
C INTEGER NBOND, NA, NB, NT1, NT2, KNOD1, K, JR, JCI
CIDENTR.....
C
C NP=0
C NBC=0
C DO 5 I=1,MAXERF
C JREFLG(I)=0
C JREFLG(I)=0
C
C Find incident acasual junctions
C M=0
C DO 10 I=1,NEL
C IF(IELST(I).GE.6) THEN
C NP1=NPTR(I)
C NP2=NPTR(I+1)-1
C DO 25 J=NP1,NP2
C IF(ICX(J).EQ.1) THEN
C M=M+1
C IF (M.GT.MAXERF) THEN
C MESSAGE= ' ... Too many nodes in implicit R fields.'
C IERRF= M
C RETURN
C ENDIF
C JREFLG(M)=I
C GOTO 10
C ENDIF
C CONTINUE
C ENDIF
C
C 10 CONTINUE
C
C IF (M.EQ.0) RETURN
C
C----- Starts search the r-fields
C NRSUK=0
C KNDO=0
C 500 NRSUK=NPRLCK+1
C JQUE(NRBLCK)=KNOD+1
C IBOND=0
C IWAY=0
C N1=JREFLG(1)
C
C IF (N1.EQ.0) RETURN
C 1000 NP1=NPTR(N1)
C NP2=NPTR(N1+1)-1
C DO 50 I=1,KNOD
C 50 IF(JREFLG(I).EQ.N1) GOTO 55
C
C KNOD=KNOD+1
C JREFLG(KNOD)=N1
C NUBED=0
C
C DO 60 N=NP1,NP2
C IF(17MX(N).EQ.1) THEN
C NBOND=NB:IMX(N)
C IF (NBOND.EQ.IBOND) GOTO 60
C NA=IBMX(NBOND,1)
C NB=IBMX(NBOND,2)
C IF (NA.EQ.N1) NA=NB
C NT1=NPTR(NA)
C NT2=NPTR(NA+1)-1
C----- Records the graph branches
C IF (NT1.EQ.NT2) THEN
C KNOD=KNOD+1
C JREFLG(KNOD)=NA
C ELSE
C IWAY=IWAY+1
C NB:FB(IWAY)=NBOND
C NB:FN(IWAY)=N1
C NUBED=NUBED+1
C ENDIF
C ENDIF
C CONTINUE
C
C 60 IF(IWAY.EQ.0) GOTO 1001
C IF(NUBED.EQ.1) THEN
C IBOND=NB:FB(IWAY)
C N1=NB:FN(IWAY)
C NB:FB(J-1)=NB:FB(J)
C NB:FN(J-1)=NB:FN(J)
C IWAY=IWAY-1
C ELSE
C IF (IWAY.NE.0) THEN
C IBOND=NB:FB(1)
C N1=NB:FN(1)
C DO 70 J=2,IWAY
C NB:FB(J-1)=NB:FB(J)
C NB:FN(J-1)=NB:FN(J)
C IWAY=IWAY-1
C GOTO 75
C ELSE
C GOTO 1001
C ENDIF
C ENDIF
C
C 75 NA=IBMX(IBOND,1)
C NB=IBMX(IBOND,2)
C IF (NA.EQ.N1) NA=NB
C N1=NA
C GOTO 1000
C
C----- Attracts the R-nodes from array JREFLG
C 1001 IQUE(NRBLCK)=NR+1
C KNOD1=JQUE(NRBLCK)
C DO 220 K=KNOD1,KNOD
C I=JREFLG(K)
C JR=IELST(I)
C IF (JR.EQ.0) GOTO 230
C IF (JR.EQ.3) THEN
C NR=NR+1
C INDXR(NR)=I
C GOTO 220

```

```
C          C      BASED ON THE INFORMATION OBTAINED BY SUBROUTINE IDENTIFR
C
C--- AUTHOR= T. ZHOU, MAY,1986
C
C--- OUTPUT= OKAY, logical status flag
C
INCLUDE 'SIZEBK.CBK'
INCLUDE 'CAUSEK.CBK'
INCLUDE 'COMMEK.CBK'
INCLUDE 'GRIEKK.CBK'
INCLUDE 'FLDSK.CBK'
INCLUDE 'SYBGBK.CBK'
C
CHARACTER NODNAM*(8), BNDNAM*B, CBIT*I
INTEGER NRBLK(1), LRF, I, N, MODI, NI, NO, NK, NBOND
INTEGER NGREE, IT, NDKI, NBP, J
LOGICAL CSUSER
C
EXTERNAL TMPASS, EXTEND
C
*****RCSSAS*****
C
C---- Initialize
NR=0
IREQUE(I)=1
CSUSER=.FALSE.
C
C----- If NCSCMD>0 then use that data
IF (NCSCMD.GT.0) THEN
    CSUSER=.TRUE.
    DO 50 I=1,NCSCMD,3
        NODNAM=CASCMD(I)
        DO 30 J=1,NEL
            IF (NODNAM.EQ.ENAM(MENX(J))) THEN
                NXDI=J
                GOTO 31
            ENDIF
        CONTINUE
        MESSAGE=' *** Node '//NODNAM//' identify failed.'
        GOTO 900
    C
    31   BNDNAM=CASCMD(I+1)
        DO 40 J=1,NBD
            IF (BNDNAM.EQ.BDNAM(BCX(J))) THEN
                DO 35 NBP=NPTR(NDKI),NPTR(NDKI+1)-1
                    IF (NBIMX(NBP).EQ.J) GOTO 41
                ENDF
            CONTINUE
            MESSAGE=' *** Bond '//BNDNAM//' identify failed.'
            GOTO 900
        C
        41   CBIT=CASCMD(I+2)
            IT=6
            IF (CBIT.EQ.'F') IT=3
            CALL EXTEND(NDKI,NBP,IT)
            CONTINUE
        C
        50   ENDIF
    C
C--- For each R-field take note of whether causality done
DO 300 NRBLK(1),NRBLCK
    LI=JRQUE(NRBLK1)
    LRF=JRQUE(NRBLK1+1) -LI
C
C----- PURPOSE: ASSIGN CAUSALITY TO THE COUPLED R-FIELDS AUTOMATICALLY,
```



```

DATA LIST(3) /'Show node attributes'/
DATA LIST(4) /'Show connector attributes'/
DATA LIST(5) /'Return'/

C
C DATA MLIM / 3,3,9,31 /
C
C *****
C
C----- Assume fresh page
OAPAG = SETACTIVEPAGE(IPAG)
STATUS = SETBKCOLOR(BLUE) ! SETVISUALPAGE in MUSEL
CALL CLEARSCREEN(SGCLEARSCREEN)
CALL SPKEYL
IF (INELS.EQ.0) THEN
  MESSAGE= '... There is no system graph defined.'
  CALL INMSGD(24)
  RETURN
ENDIF

C
C----- Set the data highlight color and special key line
C
C IOPT= -1
C CONTINUE
C CALL MUSEL(LIST,MLIM,IOPT)

C IF (IOPT.EQ.0) THEN
  IOPT= 1
  GOTO 10
ELSE IF (IOPT.EQ.4) THEN
  STATUS = SETACTIVEPAGE(OAPAG)
  STATUS = SETVISUALPAGE(OAPAG)
  RETURN
ENDIF

C IF ( (IOPT.GE.1).AND.(IOPT.LE.3) ) THEN
  IPAG= IPAG+1
  STATUS = SETACTIVEPAGE(IPAG)
  STATUS = SETVISUALPAGE(IPAG)
  STATUS = SETBKCOLOR(BLUE)
  CALL CLEARSCREEN(SGCLEARSCREEN)
  CALL SPKEYL
ENDIF

C----- Set mouse variables
GROPT = KEYOPT
IF (MOUSE) THEN
  LFTBUT = .FALSE.
  GROPT = 5
ENDIF

C IF (IOPT.EQ.1) THEN
  STRING= 'Node Connectors'
  CALL WRITESTRING(STRING,BLACK,CYAN,3,3,73)
  BLK= 0
  I= 0
  IRE= MIN(23,INELS-20*BLK+3)
  STRING= ' '
  DO 90 IR= 4,IRE
    TXCOLOR = BLACK
    BKCOLOR = WHITE
    IF (MOD(IR-3,5).EQ.0) BKCOLOR = CYAN
    I= I+1
    STRING(1:8)= ELNAM(IR)
    IF (NBXTP(1)(1:1).EQ.'A') THEN
      TMPSTR= 'MCRO'
    ELSE IF (NBXTP(1)(1:1).EQ.'P') THEN
      TMPSTR= 'PORT'
    ELSE
      J= 1
      CALL NODCLD(NBXTP(1)(2:2),TMPSTR,J)
    ENDIF
    STRING(11:16)= TMPSTR
    J= 1
  C
  C CALL PWRCLD(NBXTP(1)(3:3),REGIME,DATA,J)
  C
  C STRING(17:28)= REGIME
  C WRITE(TMPSTR,'(F8.0)') NYLOC(1)
  C STRING(28:35)= TMPSTR
  C WRITE(TMPSTR,'(F8.0)') NYLOC(1)
  C STRING(38:45)= TMPSTR
  C STRING(47:47)= NBXTP(1)(4:4)
  C ACT= 'F'
  C IF (ACTND(1)) ACT= 'T'

```

```

CALL WRITESTRING(STRING,TXCOLOR,BKCOLOR,IR,3,10)
J1=IBDPTR(1)
J2=IBDPTR(1+1)-1
IF ((J2-J1+1).GT.7) J2= J1+6
WRITE(STRING,1020) (BDNAM(1BDSEL(J)),J=J1,J2)
FORMAT('I48,1X')
CALL WRITESTRING(STRING,TXCOLOR,BKCOLOR,IR,13,63)

```

1020

```

CONTINUE
IF (MOUSE) CALL SHOWMOUSE()
CALL GETKEYVALUE(GROPT,11,12) ! Pause for user input
EVT= EVTYPE(11,12)
CALL MESKL(3,SPID,EVT)
IF (EVT.EQ.'NULL') GOTO 100
IF (MOUSE) CALL HIDE MOUSE()
BLK= BLK+1
IF (INELS.GT.20*BLK) THEN
  IF (EVT.NE.'ZONE'.AND.EVT.NE.'CANC') THEN
    CALL TEXTBOX(4,1,20,80,BLUE)
    GOTO 20
  ENDIF
ENDIF
ELSE IF (IOPT.EQ.2) THEN
  STRING= ' '
  STRING(1:4)= 'Node'
  STRING(11:14)= 'Type'
  STRING(17:26)= 'Power Type'
  STRING(30:34)= 'X Loc'
  STRING(40:44)= 'Y Loc'
  STRING(46:48)= 'Vis'
  STRING(51:53)= 'Act'
  STRING(55:59)= 'Macro'
  CALL WRITESTRING(STRING,BLACK,CYAN,3,3,63)
  BLK= 0
  I= 0
  IRE= MIN(23,INELS-20*BLK+3)
  STRING= ' '
  DO 190 IR= 4,IRE
    TXCOLOR = BLACK
    BKCOLOR = WHITE
    IF (MOD(IR-3,5).EQ.0) BKCOLOR = CYAN
    I= I+1
    STRING(1:8)= ELNAM(IR)
    IF (NBXTP(1)(1:1).EQ.'A') THEN
      TMPSTR= 'MCRO'
    ELSE IF (NBXTP(1)(1:1).EQ.'P') THEN
      TMPSTR= 'PORT'
    ELSE
      J= 1
      CALL NODCLD(NBXTP(1)(2:2),TMPSTR,J)
    ENDIF
    STRING(11:16)= TMPSTR
    J= 1
  C
  C CALL PWRCLD(NBXTP(1)(3:3),REGIME,DATA,J)
  C
  C STRING(17:28)= REGIME
  C WRITE(TMPSTR,'(F8.0)') NYLOC(1)
  C STRING(28:35)= TMPSTR
  C WRITE(TMPSTR,'(F8.0)') NYLOC(1)
  C STRING(38:45)= TMPSTR
  C STRING(47:47)= NBXTP(1)(4:4)
  C ACT= 'F'
  C IF (ACTND(1)) ACT= 'T'

```

90

100

120

20


```

C
C**GFZMIN*****
C
C----- Set up initial gfx cursor
IR = GFXVC.NUM*EXTROWS - 2
C
C----- Pick corner 1
CONTINUE
10 GRD2= 0.5*GRDIST
X1= VPXMN +GRD2
Y1= VPYMN +GRD2
CALL GFXCUR(X1,Y1,P*)
STRING= ' Pick corner 1 of window and hit Enter:'
CALL WRITESTRING(STRING,HIMITE,CLBBK,IR,1,80)
C----- Get data event
CALL GFREV(EVTP)
IF (EVTP.EQ.'DATA') THEN
  X1= XG
  Y1= YG
  ELSEIF (EVTP.EQ.'DONE'.OR. EVTP.EQ.'CANC') THEN
    CALL WRITECHARS(' ',BLACK,BLACK,-R,1,80)
    CALL SYSDRW
    GOTO 800
  ELSEIF (EVTP.EQ.'ERAS'.OR. EVTP.EQ.'UNDO') THEN
    CALL WRITECHARS(' ',BLACK,BLACK,IR,1,80)
    CALL SYSDRW
    GOTO 10
  ELSE
    GOTO 10
  ENDIF
ENDIF
C
C----- Pick corner 2
CONTINUE
20 STRING= ' Pick corner 2 of window and hit Enter:'
CALL WRITESTRING(STRING,HIMITE,CLBBK,IR,1,80)
C----- Get data event
CALL GFREV(EVTP)
IF (EVTP.EQ.'DATA') THEN
  IF (XG.EQ.X1) GOTO 20
  IF (YG.EQ.Y1) GOTO 20
  X2= XG
  Y2= YG
  ELSEIF (EVTP.EQ.'DONE'.OR. EVTP.EQ.'CANC') THEN
    CALL WRITECHARS(' ',BLACK,BLACK,IR,1,80)
    CALL SYSDRW
    GOTO 800
  ELSEIF (EVTP.EQ.'ERAS'.OR. EVTP.EQ.'UNDO') THEN
    CALL WRITECHARS(' ',BLACK,BLACK,-R,1,80)
    CALL SYSDRW
    GOTO 10
  ELSE
    GOTO 10
  ENDIF
ENDIF
C
C----- Get the window definition (X1,Y1 at lower left)
IF (X1.GT.X2) THEN
  TMP= X1
  X1= X2
  X2= TMP
ENDIF
IF (Y1.GT.Y2) THEN
  TMP= Y1

```

```

Y1= Y2
Y2= TMP
ENDIF
C
C----- Shift window corners off the grid to midpoints
X1= X1 -GRD2
X2= X2 +GRD2
Y1= Y1 -GRD2
Y2= Y2 +GRD2
C
C----- Draw window frame and pause
ICLR= 4
STATUS = SETCOLOR(ICLR)
STATUS = RECTANGLE_W($BORDER,X1,Y1,X2,Y2)
ICLR= 15
CALL GETKEYVALUE(KEYOPT,11,12)
EVTP= EVTYP(11,12)
IF (EVTP.EQ.'CANC') THEN
  CALL WRITECHARS(' ',BLACK,BLACK,IR,1,80)
  CALL SYSDRW
  GOTO 800
ELSEIF (EVTP.EQ.'ERAS'.OR. EVTP.EQ.'UNDO') THEN
  CALL WRITECHARS(' ',BLACK,BLACK,IR,1,80)
  CALL SYSDRW
  GOTO 20
ELSEIF (EVTP.EQ.'HELP') THEN
  CALL WRITECHARS(' ',BLACK,BLACK,IR,1,80)
  CALL SYSDRW
  GOTO 10
ELSE
  EVTP= 'DONE'
ENDIF
C
C----- Now we find the center and spread of the window
XC= (X1+X2)/2.
YC= (Y1+Y2)/2.
XR= X2 -X1 +GRDIST
IF (NINT(XR/GRDIST).LT.5) XR= 5.0*GRDIST
YR= Y2 -Y1 +GRDIST
D= GRDIST*NINT(XR*ASPECT/GRDIST)
IF (YR.GT.D) THEN
  C----- Window is tall and skinny, we have to widen it.
  XR= YR/ASPECT
  XR= XR +GRDIST -MOD(XR,GRDIST)
  ELSE
    C----- Window is short and fat, may have to extend Y range for aspect
    YR= MAX((Y2-Y1+GRDIST),D)
  ENDIF
C----- Set the window frames
VPXMN= XC -XR*0.5
VPYMN= VPXMN -GRD2
VPXMN= VPXMN +XR
VPYMN= YC -YR*0.5
IF (ABS(MOD(VPYMN,GRDIST)).LT.GRDIST*0.01)
  VPYMN= VPYMN -GRD2
VPYMN= VPYMN +YR
C
C----- Return section
800 CONTINUE
RETURN
END

```

[illegible]


```

C CHARACTER LBL*(*), DATA*(*), EVTP*4
C CHARACTER EVTYP*4, STRING*80, LINSTR*80
C INTEGER SNUM, FLD(4), ROW, COL, SEC
C INTEGER LN, CL, CS, CE, DFL, LFL, IC, NCHARS, I1, I2, I3
C INTEGER FLIM(21,4), NFLD, MFLD, GKOPT, STATUS
C RECORD /RCCOORD/ TXTPOS
C
C EXTERNAL EVTYP, NCHARS, GETKEYVALUE, WRITESTRING
C EXTERNAL LINSTR, MFLD, SHOWMOUSE, HIDE MOUSE
C INTRINSIC MAX, MIN
C
C***CFGET*****
C
C LN = FLD(1)
C C1 = FLD(2)
C C2 = FLD(3)
C C3 = FLD(4)
C DFL= CE - CS + 1
C LFL= NCHARS(LBL)
C
C----- Define the field array (one element here) and other ME data
C FLIM(1,1) = LN
C FLIM(1,2) = CS
C FLIM(1,3) = LN
C FLIM(1,4) = CE
C
C NFLD = 1
C MFLD = 1
C
C----- Write the label and default data
C IF (SNUM.EQ.-1) THEN : Draw GREY if not selectable
C CALL WRITESTRING(LBL,08,WINBC,LN,FLD(2),LFL)
C ELSE
C CALL WRITESTRING(LBL,FTXTC,WINBC,LN,FLD(2),JFL)
C END IF
C
C STRING= DATA
C IF (SNUM.EQ.-1) THEN : Draw GREY if not selectable
C CALL WRITESTRING(STRING,08,WINBC,LN,CS,DFL)
C ELSE
C CALL WRITESTRING(STRING,DATTC,WINBC,LN,CS,DFL)
C END IF
C
C----- Return if first call
C IF (SNUM.EQ.0.OR.SNUM.EQ.-1) RETURN
C
C----- Highlight the field label
C CALL WRITESTRING(LBL,AFFGC,AFFGC,LN,FLD(2),LFL)
C
C----- Position the pointer
C IC= CS
C
C----- Set mouse control variables
C GKOPT = KEYOPT
C IF (MOUSE) THEN
C GKOPT = 5 ! no wait
C LFTBOT = .FALSE.
C OLDFLD = 0
C CALL SHOWMOUSE()
C ENDF
C
C----- Turn on the cursor
C STATUS = DISPLAYCURSOR(SCCURSORON)
C----- Get some input
C IF (ROW.NE.0) THEN ! Arrived in routine with mouse click

```

```

C
C      / .option #1 of seg LIST(L1)      up arw
C      / .option #2 of seg LIST(L1+1)
C      / .option last List(L1+DLL-1) dn arw
C      Length of box is FLD(4)+4
C
C      INCLUDE 'GRAPH.FD'
C      INCLUDE 'GUTL.FD'
C      INCLUDE 'COMBK.CBK'
C      INCLUDE 'FMOUSE.CBK'
C      INCLUDE 'VDENV.CBK'
C      INCLUDE 'VIEWBK.CBK'
C
C      CHARACTER LBL*(*), LIST(*)*(*), LP*(*), EVTP*4
C      CHARACTER EVTYP*4
C      INTEGER SNUM, FLD(6), DP(*), NSEL, ROW, COL, SEC
C      INTEGER LL, DLL, MAXSEL, FLIM(21,4), NEFD, XFLD, GKOPT
C      INTEGER I, I1, I2, L, L1, L2
C      INTEGER R1, C1, IR, IC, AUROW, ADROW, ACOL, PUC, PDC
C      INTEGER INDEX, TXICLR, BKCLR, SCRC
C      INTEGER IPLAST, L1LAST, LSTWID, LBLWID, PIDX, PROW
C      INTEGER NCHARS
C      LOGICAL SELECT, FIX, REDRAW
C
C      EXTERNAL INMSGD, WRITESTRING, WRITECHARS, GETKEYVALUE
C      EXTERNAL TEXTBOX, EVTYP, NEFD, NCHARS, SHOWMOUSE, HIDE MOUSE
C      INTRINSIC MAX
C
C      C***CFGSEL*****
C
C      C---- Initialize variables
C      R1 = FLD(1) + 1
C      C1 = FLD(2) + 1
C      DLL = FLD(3) - FLD(1)
C      L1 = FLD(5)
C      MAXSEL= FLD(6)
C      PUC = 24
C      PDC = 25
C      IF (DLL.EQ.1) THEN
C        PUC = 18
C        PDC = 18
C      END IF
C      LSTWID = 0
C      DO 10 I = 1, L1
C        LSTWID = MAX(LSTWID, NCHARS(LIST(I)))
C      CONTINUE
C      LBLWID = NCHARS(LBL)
C      SCRC = SCRLI
C      REDRAW = .FALSE.
C      CONTINUE
C
C      C---- Initialize for segment display
C      L1LAST= INT((L1-1)/DLL)*DLL + 1
C      IPLAST= R1 + L1 - L1LAST
C      AUROW = R1
C      ACOL = C1 + LSTWID + 3
C      ADROW = R1 + DLL - 1
C
C      C---- Put up the background and label
C      CALL TEXTBOX(R1,C1,DLL,LSTWID+4,SELBC)
C      IF (SNUM.EQ.-1) THEN
C        ! Draw GREY if not selectable
C        CALL WRITESTRING(LBL,08,WINBC,FLD(1),FLD(2),0)
C      ELSE
        CALL WRITESTRING(LBL,FTXTC,WINBC,FLD(1),FLD(2),0)
      END IF
C
C      C---- Set up initial pointer/segment display
C      DO 20 L= 1,L1
C        ! Initial pointer
C        IF (LIST(L).EQ.LP) THEN
C          L1= INT((L-1)/DLL)*DLL + 1
C          PIDX= L
C          PROW= R1 + MOC(L-1,DLL)
C          GOTO 21
C        ENDIF
C      CONTINUE
C      L1= 1
C      PIDX= 1
C      PROW= R1
C      CONTINUE
C
C      C---- Set mouse variables before entering loop
C      GKOPT = KEYOPT
C      IF (MOUSE) THEN
C        ! Don't wait
C        ! mouse flags
C        LFTBUT = .FALSE.
C        OLDFLD = 0
C      END IF
C
C      C---- Check to see if arrived in routine with the mouse
C      IF (ROW.NE.0) LFTBUT=.TRUE.
C
C      C---- Display a segment of file starting at L1
C      CONTINUE
C      L2= MIN(L1+DLL-1,L1)
C
C      C---- Draw the scroll bars
C      IF (SNUM.EQ.-1) THEN
C        IR= AUROW
C        ! Ptr_Up unless first segment
C        IF (L1.EQ.1) THEN
C          CALL WRITECHARS(CHAR(194),08,SELBC,IR,ACOL,1)
C        ELSE
C          CALL WRITECHARS(CHAR(194),08,SELBC,IR,ACOL,1)
C        ENDIF
C      ELSE
C        IR= AUROW
C        ! Ptr_Up unless first segment
C        IF (L1.EQ.1) THEN
C          CALL WRITECHARS(CHAR(194),08,SELBC,IR,ACOL,1)
C        ELSE
C          CALL WRITECHARS(CHAR(194),08,SELBC,IR,ACOL,1)
C        ENDIF
C      ENDIF
C
C      DO 40 I = 1,DLL-2
C        IF (SNUM.EQ.-1) THEN
C          CALL WRITECHARS(CHAR(179),08,SELBC,AUROW+I,ACOL,1)
C        ELSE
C          CALL WRITECHARS(CHAR(179),08,SELBC,AUROW+I,ACOL,1)
C        ENDIF
C      CONTINUE
C
C      IR= ADROW
C      ! Ptr_Dn unless last segment
C      IF (SNUM.EQ.-1) THEN
C        IF (L2.EQ.L1) THEN
C          CALL WRITECHARS(CHAR(193),08,SELBC,IR,ACOL,1)
C        IF (DLL.EQ.1) CALL WRITECHARS(CHAR(24),08,03,IR,ACOL,1)
C      ELSE

```

```

EVTB= 'MOVE'
GOTO 20
ENDIF

CALL SETTEXTPOSITION(LN,IC,XTPOS)
CALL GETKEYVALUE(GKOPT,II,12)
EVTB= EVTTP(II,12)
CALL MEFLD(5,6,FLIM,NFLD,MFLD,EVTB)

IF ( (EVTB.EQ.'CRET') OR. (EVTB.EQ.'TABF') OR.
+ (EVTB.EQ.'TABB') OR. (EVTB.EQ.'DONE') OR.
+ (EVTB.EQ.'OFF') ) THEN
STRING= LINSTR(LN,CS,CE)
DATA= STRING(1:DFL)
SEC = SNUM + 1
IF (EVTB.EQ.'TABB') SEC = SNUM - 1
ROW= 0
COL = 0
IF (EVTB.EQ.'OFF') THEN
SEC = -1
IF (EVTB.EQ.'TABF' ) THEN
GOTO 800
END IF
IF (EVTB.EQ.'CRET') EVTP = 'TABF'
GOTO 800
ELSEIF (EVTB.EQ.'DATA') THEN
IF (MOUSE) CALL HIDE MOUSE(1)
CALL WRITESTRING(CHAR(11),BLACK,WINBC,LN,IC,1)
IF (MOUSE) CALL SHOW MOUSE(1)
IC= MIN(IC+1,CE)
ELSEIF (EVTB.EQ.'EDIT') THEN
IF (MOUSE) CALL HIDE MOUSE(1)
IF (11.EQ.8) THEN
IF (IC.GT.CS) THEN
LI= IC -CS +1
STRING= LINSTR(LN,CS,CE)
DATA= STRING
DATA(LI+1:DFL-1)= STRING(LI:DFL)
DATA(DFL:DFL)= ' '
STRING= DATA
CALL WRITESTRING(STRING,DATTC,DATBC,LN,CS,DFL)
IC= IC -1
ENDIF
ELSE
STRING= LINSTR(LN,CS,CE)
LI= IC -CS
IF (LI.GT.0) DATA(LI:LI)= STRING(LI:LI)
IF (12.EQ.82) THEN
DATA(LI+1:LI+1)= ' '
DATA(LI+1:DFL)= STRING(LI+1:DFL-1)
IF (LI+2.LE.DFL) DATA(LI+2:DFL)= STRING(LI+1:DFL-1)
ELSEIF (12.EQ.83) THEN
! Delete
IF (LI+2.LE.DFL) DATA(LI+1:DFL-1)= STRING(LI+2:DFL)
DATA(DFL:DFL)= ' '
ENDIF
ENDIF
ELSE
STRING= DATA
CALL WRITESTRING(STRING,DATTC,DATBC,LN,CS,DFL)
ENDIF
IF (MOUSE) CALL SHOW MOUSE(1)
ELSEIF (EVTB.EQ.'CANC') THEN
GOTO 800

```

```

ELSE
  CALL WRITECHARS(CHAR(PDC),08,03,IR,ACOL,1)
  IF ( (DLL.EQ.1) .AND. (L1.EQ.1) )
    + CALL WRITECHARS(CHAR(25),08,03,IR,ACOL,1)
  ENDIF
ELSE
  IF (L2.EQ.L1) THEN
    CALL WRITECHARS(CHAR(193),SCRC,03,IR,ACOL,1)
    IF (DLL.EQ.1) CALL WRITECHARS(CHAR(24),SCRL,03,IR,ACOL,1)
  ELSE
    CALL WRITECHARS(CHAR(PDC),SCRC,03,IR,ACOL,1)
    IF ( (DLL.EQ.1) .AND. (L1.EQ.1) )
      + CALL WRITECHARS(CHAR(25),SCRC,03,IR,ACOL,1)
    ENDIF
  ENDIF
ENDIF
IF (L2.EQ.L1) THEN
  CALL WRITECHARS(CHAR(193),SCRC,03,IR,ACOL,1)
  IF (DLL.EQ.1) CALL WRITECHARS(CHAR(24),SCRL,03,IR,ACOL,1)
ELSE
  CALL WRITECHARS(CHAR(PDC),SCRC,03,IR,ACOL,1)
  IF ( (DLL.EQ.1) .AND. (L1.EQ.1) )
    + CALL WRITECHARS(CHAR(25),SCRC,03,IR,ACOL,1)
  ENDIF
ENDIF
END IF

C----- Write the list data
IC= R1 - 1
IC= C1 + 2
DO 110 L= L1,L2
  IR= IR + 1
  TXTCR = LSTDC
  BKCLR = SELBC
DO 105 INDEX = 1, NSEL
  IF ( (INDEX).EQ.1) THEN
    TXTCR = SELDC
    GOTO 106
  ENDIF
CONTINUE
105 CONTINUE
106 CONTINUE
IF ( (SNUM.EQ.-1) ) THEN
  CALL WRITESTRING(LIST(C),08,BKCLR,IR,IC,LSTW:D)
ELSE
  CALL WRITESTRING(LIST(C),TXTCR,BKCLR,IR,IC,LSTWID)
END IF

C
110 CONTINUE
DO 120 L= L2+1,L1+DLL
  IR= IR-1
  CALL WRITECHARS(' ',SELBC,SELBC,IR,C1+2,LSTW:D)
CONTINUE
120 CONTINUE
C----- Write the current selected option if right conditions
IF ( (SNUM.EQ.-1) ) THEN
  IF ( (MAXSEL.EQ.1) .AND. (DLL.NE.1) ) CALL WRITESTRING
    + (LIST(DF(C)),08,WINBC,FLD(1),FLD(2)+LBLWID+1,LSTWID)
  ELSE
    IF ( (MAXSEL.EQ.1) .AND. (DLL.NE.1) ) CALL WRITESTRING
    + (LIST(DF(1)),SELBC,WINBC,FLD(1),FLD(2)+LBLWID+1,LSTWID)
  END IF
CONTINUE
C----- Return if initial call or redrawn because not selectable
IF ( (SNUM.EQ.0.OR.SNUM.EQ.-1) ) RETURN
C
C----- Display the pointer/ Highlight label
CALL WRITECHARS(PNTAC,PNTCL,SELBC,PROM,C1+1,1)
CALL WRITESTRING(LBL,AFFGC,AFFGC,FLD(1),FLD(2),0)
C
C----- Display the mouse after drawing menu
IF (MOUSE) CALL SHOWMOUSE
C
C----- Define the menu field array

```

```

NFLD = L2 - L1 + 2
FLIM(L1,1) = AUROW
FLIM(L1,2) = ACOL
FLIM(L1,3) = AUROW
FLIM(L1,4) = ACOL
DO 50 L=2, NFLD
  FLIM(L,1) = R1 + L - 2
  FLIM(L,2) = C1
  FLIM(L,3) = R1 + L - 2
  FLIM(L,4) = C1 + LSTWID + 2
CONTINUE
50 CONTINUE
NFLD = NFLD + 1
FLIM(NFLD,1) = ADROW
FLIM(NFLD,2) = ACOL
FLIM(NFLD,3) = ADROW
FLIM(NFLD,4) = ACOL

C----- Get some input
150 CALL GETKEYVALUE(GKOPT,I1,I2)
EVTP= EVTYP(I1,I2)
CALL MEFL2(2,6,FLIM,NFLD,NFLD,SVTP)

C----- Crack the user input
IF (EVTP.EQ.'MOVE') THEN
  IF (MOUSE) CALL HIDEHOUSE()
  CALL WRITECHARS(' ',SELBC,SELBC,PROM,C1+1,1) : Erase Ptr
  IF (L2.EQ.71) THEN
    L1= 1
    PIDX= L1
    PROM= R1
  ELSEIF (L2.EQ.79) THEN
    : End
  IF (L1.EQ.L1LAST) RSDRAW = .TRUE.
  L1= L1LAST
  PIDX= L1
  PROM= IPLAST
ELSEIF (L2.EQ.73) THEN
  : PgUp
  IF (L1.EQ.1) THEN
    PIDX= L1
    PROM= R1
  ELSE
    REDRAW = .TRUE.
    L1= L1 - DLL
    PIDX= L1 + DLL - 1
    PROM= ADROW
  ENDIF
ELSEIF (L2.EQ.81) THEN
  : PgDn
  IF (L2.LT.L1) THEN
    REDRAW = .TRUE.
    L1= L1 - DLL
    PIDX= L1
    PROM= AUROW
  ELSE
    PIDX= L1
    PROM= IPLAST
  ENDIF
ELSEIF (L2.EQ.72) THEN
  : UpArrow
  IF (PROM.EQ.AUROW) THEN
    IF (L1.GT.1) THEN
      REDRAW = .TRUE.
      L1= L1 - DLL
      PIDX = PIDX - 1
      PROM = ADROW
    ENDIF
  ENDIF

```

```

C
C---- Process the CRET if not a mouse page
IF (MOUSE) CALL HIDEMOUSE()
CALL WRITECHARS(' ',SELBC,SELBC,PROW,C1+1,1)
IF (ROW.NE.0) THEN
  ROW = 0
  PIDX = L1 + MFLD - 2
  PROW = AUROW + MFLD - 2
  END IF
C
C IF (MAXSEL.GT.1) THEN      ! Handle multiple selections
C
C---- See if entry is currently selected
SELECT = .FALSE.
DO 200 INDEX = 1, NSEL
  IF ( DP(INDEX).EQ.PIDX ) SELECT = .TRUE.
200  CONTINUE
C
C---- Check current select status
IF ( SELECT ) THEN .Remove item from index list, decrement NSEL
  FIX = .FALSE.
  DO 250 INDEX = 1, NSEL
    IF (DP(INDEX).EQ.PIDX) FIX = .TRUE.
    IF ( FIX ) THEN
      IF ( INDEX .EQ. NSEL ) THEN
        DP(INDEX) = 0
      ELSE
        DP(INDEX) = DP(INDEX+1)
      ENDIF
    ENDIF
  CONTINUE
250
C---- Item is not currently selected, attempt to add to list
IF ( NSEL .LT. MAXSEL ) THEN
  NSEL = NSEL + 1
  DP(NSEL) = PIDX
ELSE
  MESSAGE = 'Maximum number of items already selected.'
  CALL LMMSGD(24)
ENDIF
ENDIF
GOTO 100
ELSE
  ! Handle single selections
  LP= LIST(PIDX)
  DP(1)=PIDX
  CALL WRITESTRING(LIST(DP(1)),LSTDC,BKCLR,PROW,C1+2,0)
ENDIF
GOTO 100
ELSEIF (EVTP.EQ.'OFF') THEN      ! Mouse event off all fields
  SEC = -1
  ROW = MOUSER
  COL = MOUSEC
  IF ( (FID(1).LE.ROW).AND.(ROW.LE.FLD(3)).AND.
    + (FID(2).LE.COL).AND.(COL.LE.FLD(4)) ) GOTO 150
  GOTO 800
ELSEIF ( (EVTP.EQ.'TABF').OR.(EVTP.EQ.'TABB') ) THEN
  SEC = SNUM + 1
  IF (EVTP.EQ.'TABB') SEC = SNUM - 1
  ROW = 0
  COL = 0
  GOTO 800
ENDIF
ENDIF

```

[illegible]

[illegible]


```

C----- INPUTS: LN, line number
C CS, CE, start col, end col for field
C DATA, default data string to start
C----- OUTPUTS: EVTP, CANC, HELP, or DATA
C DATA, If DATA event, then value of string
C
C INCLUDE 'FGRAPH.FD'
C INCLUDE 'FMOUSE.CBK'
C INCLUDE 'VIEWBK.CBK'
C INCLUDE 'COMMCK.CBK'
C INCLUDE 'VIDENV.CBK'
C
C CHARACTER EVTP*4, DATA*(*), EVTYP*4, STRING*80
C INTEGER*2 IPAG, LN, CS, CE, FL, I1, I2, ICS, NCHARS
C INTEGER*2 TXCOLOR, BKCOLOR, FLIM(21,4), NFELD, MFLD, GKOPT
C RECORD /RCOORD/ TXPOS
C
C EXTERNAL EVTYP, GETKEYVALUE, NCHARS, SHOWMOUSE, HIDEHOUSE, MESKL
C INTRINSIC MAX, MIN
C
C**INSTR*****
C
C IF (CURMOD.EQ.TXMOD) THEN
  TXCOLOR = BLACK
  BKCOLOR = WHITE
ELSE
  TXCOLOR = HIWHITE
  BKCOLOR = BLACK ! Dummy
ENDIF
C----- Define mouse variables
GKOPT = KEYOPT
IF (MOUSE) THEN
  GKOPT = 5 ! No wait
  LFBUT = .FALSE.
ENDIF
FLN(1,1) = LN
FLN(1,2) = CS
FLN(1,3) = LN
FLN(1,4) = CE
NFELD = 1
C----- Get length of field (NOTE: LEN(DATA) = CE-CS+1 assumed.)
IPAG = 0
FL = CE - CS + 1
C----- Write the default data and set initial pointer position
5 CONTINUE
STRING = DATA
ICS = 1 + NCHARS(DATA) ! Position cursor in STRING
C----- Get some input
10 CONTINUE
CALL WRITESTRING(STRING, TXCOLOR, BKCOLOR, LN, CS, FL+1) ! Show STRING
CALL SETTEXTPOSITION(LN, CS-1+ICS, TXPOS) ! Show "cursor"
IF (ICS.LE.FL) THEN
  CALL WRITESTRING(CHAR(95), WHITE, BLACK, LN, CS-1+ICS, 1)
ELSE
  CALL WRITESTRING(CHAR(17), WHITE, BLACK, LN, CS-1+ICS, 1)
ENDIF
IF (MOUSE) CALL SHOWMOUSE()
CALL GETKEYVALUE(GKOPT, I1, I2)
EVT = EVTYP(I1, I2)

CALL MFLD(2,3, FLIM, NFELD, MFLD, EVTP)
IF (EVTP.EQ.'NULL') GOTO 15
IF (EVTP.EQ.'CANC'.OR. EVTP.EQ.'HELP') THEN
  IF (MOUSE) CALL HIDEHOUSE()
  RETURN
ELSEIF (EVTP.EQ.'UNDO') THEN
  IF (MOUSE) CALL HIDEHOUSE()
  GOTO 5
ELSEIF (EVTP.EQ.'CRET'.OR. EVTP.EQ.'DONE') THEN
  EVTP = 'DATA'
  DATA = STRING(1:FL)
  IF (MOUSE) CALL HIDEHOUSE()
  RETURN
ELSEIF (EVTP.EQ.'DATA') THEN
  IF (I1.GE.32 .AND. I1.LE.126) THEN
    IF (ICS.LE.FL) THEN
      STRING(ICS:ICS) = CHAR(I1)
      ICS = ICS+1
    ENDIF
  ELSEIF (EVTP.EQ.'MOVE') THEN
    IF (I2.EQ.75) THEN
      IF (ICS.GT.1) THEN
        ICS = ICS - 1
        STRING(ICS:ICS) = ' '
      ELSEIF (I2.EQ.71) THEN
        STRING(1:FL) = ' '
        ICS = 1
      ELSEIF (EVTP.EQ.'EDIT') THEN
        IF (I1.EQ.8) THEN
          IF (ICS.GT.1) THEN
            ICS = ICS - 1
            STRING(ICS:ICS) = ' '
          ENDIF
        ENDIF
      ENDIF
    ENDIF
  ELSEIF (EVTP.EQ.'DELETE') THEN
    IF (I1.EQ.8) THEN
      IF (ICS.GT.1) THEN
        ICS = ICS - 1
        STRING(ICS:ICS) = ' '
      ENDIF
    ENDIF
  ENDIF
  IF (MOUSE) CALL HIDEHOUSE()
  GOTO 10
END
CEND:INSTR*****
C
C MESSAGE: ' SETMOD'
C SETMOD ***** Last Change: 25-Jul-94 mkh
C
C SUBROUTINE SETMOD(ARG)
C----- PURPOSE: Set the screen to alpha or graphics mode.
C C----- INPUT: ARG, = TXT or MON
C C----- OUTPUT: OKAY, MESSAGE
C
C INCLUDE 'FGRAPH.FD'
C INCLUDE 'BGFBK.CBK'
C INCLUDE 'COMMCK.CBK'
C INCLUDE 'VIEWBK.CBK'
C INCLUDE 'VIDENV.CBK'
C

```



```
C
C-----CPFILE:ENIN1-----
C
C--- PURPOSE: Initialize program end model.
C
C--- CONTENTS:
C      ENIN1 l-time initialization of config data
C      ENLOGO put MB, SL, or DD logo on screen
C      C     leave dialog for all modules
C
C--- Last Change: August 27, 1994 rcr
CFOFH:ENIN1-----
C
C      INCLUDE 'FGRAPH.FI'
C      INCLUDE 'PMOUSE.FI'
C
C      MESSAGE:' ENIN1'
CENIN1>>>Last Change: 29-Aug-94 rcr
C
C      SUBROUTINE ENIN1
C
C--- PURPOSE: l-time initialization of config data
C
C--- Inputs: none
C
C--- Outputs: Common data initialized
C
C      INCLUDE 'COMMON.CBK'
C      INCLUDE 'PMOUSE.CBK'
C      INCLUDE 'PATHK.CBK'
C      INCLUDE 'VIEWK.CBK'
C
C      INTEGER NCHARS, L
C
C      EXTERNAL NCHARS
C
C**ENIN1*****
C
C---- Set the data
C      PROGID='IS.JAlpha'
C      VERTYP='PROF'
C      IF (VERTYP.EQ.'STUDENT') THEN
C          MAXUDS=0
C      ELSEIF (VERTYP.EQ.'PROF ') THEN
C          MAXUDS=99
C      ELSEIF (VERTYP.EQ.'PROFFLUS') THEN
C          MAXUDS=99
C      ENDIF
C
C---- Define color mnemonics for color INDEX values
C      B-A-C-K = 0
C      BLUE   = 1
C      GREEN  = 2
C      CYAN   = 3
C      RED    = 4
C      MAGENTA = 5
C      BROWN  = 6
C      WHITE  = 7
C      GRAY   = 8
C      LTBLUE = 9
C      LTGREEN = 10
C      LTCYAN = 11
C      LTRED  = 12
```


[illegible]

Appendix C

ENPORT Model Files

The following pages contain the model files that were used to model and simulate each of the examples that were described in Chapter 2. These model files contain information about the models regarding the position of the nodes and their connectors, the constitutive equations for each node, and the values of the parameters associated with the equations.

HEADING

FILE MODEL 19:27:20 09/28/94 ENPORT/PC 5.3Alpha
MECHOSC.ENF Model_name: MECHOSC

TITLE
Impact-type oscillator, nonlinear for large excursions.

SYSTEM GRAPH DESCRIPTION

NODE	TYPE	XLOC	YLOC	ACT	MACRO
C1	MCGV	400.	300.	T	
C2	MCGV	400.	200.	T	
IV	MIGV	300.	200.	T	
SE	MEGV	300.	300.	T	
I	MIGV	300.	100.	T	
R	MRGV	400.	100.	T	

CONNECTOR TYPE FROM TO VERTICES

1	BG V	IV	C1
2	BG V	IV	C2
3	BG V	IV	I
4	BG V	SE	IV
5	BG V	IV	R

NODE EQUATIONS

Named_parameters:	0
Number of outputs:	5
Node: C1	
Equation:	Y = GAIN (X, P)
Y_list	X_list
E.1	Q.1
Parameters	2.00000E+00
Node: C2	
Equation:	Y = BACKLASH(X, P)
Y_list	X_list
E.2	Q.2
Parameters	1.00000E+00
	2.00000E+01
Node: SE	
Equation:	Y = CON (X, P)
Y_list	X_list
E.4	
Parameters	0.00000E+00
Node: I	
Equation:	Y = ATT (X, P)
Y_list	X_list
F.3	P.3
Parameters	2.00000E+00
Node: R	
Equation:	Y = GAIN (X, P)
Y_list	X_list
E.5	F.5
Parameters	1 1 1
	0.00000E+00

SORTED SYSTEM EQUATIONS

13 14	48 48 48 7 16 6 41 5 16 4
4 4 4	
1 2 3 4 5 6 7 8 9 10 11 12 13 14	
1 2 3 4 5 6 10 11 12 13 14 15 16	
1 1 1 1 2 3 5 9 10 11 11 11 11	
Q1 Q2 P3 E4 E1	
E2 E3 F3 E5 F1	
F2 F4 F5	
8 8 7 1 2 5 6 4 9 3 8 8 8 8	
0.00000E+00 2.00000E+00 1.00000E+00 2.00000E+01 -1.00000E+00	
-1.00000E+00 1.00000E+00 -1.00000E+00 2.00000E+00 0.00000E+00	
3 0 1 5 0	
0 4 1 1 14	
XI XI XI U A A A A A Y Y Y	
Y TIME	
1 2 3	
8 8 7	
4 4	
1 1	
13	
1 2 3 4 5 6 7 8 9 10 11 12 13 14	
1 2 3 4 8 9 6 5 7 10 11 12 13	

INITIAL CONDITIONS

Q1	=	1.0000E-02
Q2	=	-1.0000E-02
P3	=	0.0000E+00

ALGEBRAIC VARIABLES

Loops: 0 Nbr vbls: 0

OUTPUT VARIABLES

Q1

TIME CONTROLS

Initial time = 0.0000E+00
Final time = 1.0000E+00
Number saved = 1001

END-OF-FILE

HEADING

FILE MODEL 18:48:20 03/26/95 ENPORT/PC 5.3A1.pha
 FLATBED.EMP Model_name: FLATBED

TITLE

Flatbed trailer suspension system with weight evenly distributed.

SYSTEM GRAPH DESCRIPTION

NODE	TYPE	XLOC	YLOC	ACT	MACRO
IM	MIGV	-600.	200.	T	
1VC	MIGV	-400.	200.	T	
SEW	MEGV	-200.	200.	T	
1ROT	MIGV	0.	200.	T	
1ROT	MIGV	300.	200.	T	
TFR	MTGV	-200.	100.	T	
TEF	MTGV	0.	0.	T	
OSF	M0GV	0.	-100.	T	
1SF	M1GV	200.	-100.	T	
CSF	MC0V	400.	0.	T	
RSF	MRGV	400.	-200.	T	
CSR	MC0V	-800.	0.	T	
RSR	MRGV	-800.	-200.	T	
1WHR	M1GV	-400.	-300.	T	
1WHR	M1GV	-700.	-300.	T	
1WHR	M1GV	300.	-300.	T	
SEWR	MEGV	-700.	-500.	T	
SEWF	MEGV	300.	-500.	T	
OTR	M0GV	-400.	-500.	T	
OTF	M0GV	0.	-500.	T	
1TR	M1GV	-600.	-600.	T	
1TF	M1GV	200.	-600.	T	
CTR	MC0V	-800.	-700.	T	
CTF	MRGV	400.	-700.	T	
RTF	MRGV	200.	-700.	T	
RTR	MRGV	-600.	-700.	T	
SFR	MF0V	-400.	-700.	T	
SFF	MF0V	0.	-700.	T	
OSR	M0GV	-400.	-100.	T	
1SR	M1GV	-600.	-100.	T	

CONNECTOR TYPE FROM TO VERTICES

1F	BM V	SFF	OTF
2F	BM V	OTF	1TF
3F	BM V	1TF	RTF
4F	BM V	1TF	CTF
5F	BM V	OTF	1WHR
6F	BM V	1WHR	1WHR
7F	BM V	SEWF	1WHR
8F	BM V	1WHR	OSF
9F	BM V	OSF	1SF
10F	BM V	1SF	RSF
1R	BM V	SFR	OTR
2R	BM V	OTR	1TR
3R	BM V	1TR	RTR
4R	BM V	1TR	CTR
5R	BM V	OTR	1WHR

6R	BM V	1WHR	1WHR
7R	BM V	SEWR	1WHR
8R	BM V	1WHR	OSR
9R	BM V	OSR	1SR
10R	BM V	1SR	RSR
11R	BM V	1SR	CSR
12R	BM V	OSR	1VC
13R	BM V	TFR	OSR
11F	BM V	1SF	CSF
12F	BM V	OSF	TEF
13F	BR V	TEF	1ROT
14R	BR V	1ROT	TFR
14F	BM V	OSF	1VC
1M	BM V	SEW	1VC
2M	BM V	1VC	IM
R1	BM V	1ROT	1ROT

NODE EQUATIONS

Named parameters: 16

1	JVEH	8.00000E+02
2	MVEH	6.00000E+02
3	XSUS	1.00000E+04
4	DSUS	1.70000E+03
5	KTR	1.80000E+05
6	DTIR	6.00000E+03
7	WWT	-1.96200E+02
8	MWH	2.00000E+22
9	WVEH	-5.88600E+03
10	DR	1.50000E+00
11	DF	1.50000E+00
12	DEL1	1.00000E-01
13	DEL2	2.44580E-01
14	DEL3	2.08435E-01
15	DEL4	3.44580E-01
16	SLOPE	5.00000E-02

Number of outputs: 21

Node: IM
 Equation: $Y = ATT(X, P)$
 Y_list X_list 1 1 1
 F.2M P.2M Parameters MVEH

Node: SEW
 Equation: $Y = CON(X, P)$
 Y_list X_list 1 0 1
 E.1M Parameters MVEH

Node: 1ROT
 Equation: $Y = ATT(X, P)$
 Y_list X_list 1 1 1
 F.R1 P.R1 Parameters JVEH

Node: TFR
 Equation: $Y = GAIN(X, P)$
 Y_list X_list 1 1 1
 F.13R F.14R Parameters DR
 Equation: $Y = GAIN(X, P)$
 Y_list X_list 1 1 1 Parameters


```

15 16 22 40 28 29 32 33 71 71 1 41 23 9 21
19 2 17 41 17 23 31 20 14 25 24 3 22 4 40
34 10 41 35 11 23 5 6 12 30 13 31 39 36 38
37 7 8 33 32 30 14 18 27 26 14 14 14 17 17
23 23 23 22 30 30 30 31 31 31 34 34 35 35
32 32 33 33 41 41 40 40
WVEH WMT WMT DEL3 SLOPE
DEL4 DEL1 SLOPE DEL2 MVEH
1.00000E+00 1.00000E+00 1.00000E+00 1.00000E+00-1.00000E+00
JVEH DR DR DF DF
1.00000E+00-1.00000E+00 1.00000E+00 1.00000E+00 1.00000E+00
KSUS DSUS KSUS DSUS 1.00000E+00
1.00000E+00-1.00000E+00 1.00000E+00 1.00000E+00-1.00000E+00
MMH MMH 1.00000E+00-1.00000E+00 1.00000E+00 1.00000E+00
-1.00000E+00 1.00000E+00 1.00000E+00 1.00000E+00 1.00000E+00
KTIR KTIR DTIR DTIR 1.00000E+00
-1.00000E+00 1.00000E+00 1.00000E+00 1.00000E+00
8 0 5 28 0
0 29 5 1 71
XI XI XI XI XI XI XI XI XI XI XI XI XI XI XI XI
U A A A A A A A A A A A A A A A A A A
A A A A A A A Y Y Y Y Y Y Y Y Y Y
Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y
1 2 3 4 5 6 7 8
15 16 22 40 28 29 32 33
9 10 11 12 13
9 10 11 12 13
1
1
70
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
31 32 33 34 35 36 37 38 39 40 41 42 43 44 45
46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
61 62 63 64 65 66 67 68 69 70 71 12 13 37 36
1 2 3 4 5 6 7 8 9 10 11 12 13 37 36
31 33 38 35 30 32 39 34 26 24 17 20 18 14 40
27 41 28 19 22 25 23 29 21 16 15 42 43 44 45
46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
61 62 63 64 65 66 67 68 69 70

```

END-FILE

REFERENCES

REFERENCES

- Ermer, G. *Improving Engineering System Design Through Scientific Visualization Methods*. Ph. D. Thesis. Michigan State University: East Lansing, MI, 1994.
- Rosenberg, R. C. and D. C. Karnopp. *Introduction to Physical System Dynamics*. New York, NY: McGraw-Hill, 1983.
- Rosenberg, R.C. and T. Zhou. "Power-Based Model Insight." Presented at the Winter Annual Meeting of ASME, Chicago, IL, Nov. 27 - Dec. 2, 1988.
- Zhou, T. *Improving Design Assessment and Simulation of Large-Scale Dynamic Systems*. Ph. D. Thesis. Michigan State University: East Lansing, MI, 1988.