This is to certify that the

thesis entitled

COMPUTER-AIDED DESIGN OF
PLANAR MECHANICAL SYSTEMS

presented by

John Douglas Reid

has been accepted towards fulfillment
of the requirements for

Master's degree in Mechanical Engineering

R. C. Rosenberg
Major professor

Date 10 November 1983

# COMPUTER-AIDED DESIGN OF PLANAR MECHANICAL SYSTEMS

By

John Douglas Reid

A THESIS

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

MASTER OF SCIENCE

Department of Mechanical Engineering

1983

ma
ap
the
com
sys
ele
the
a P
by
sim
dynam
comp

ABSTRACT

COMPUTER-AIDED DESIGN OF PLANAR MECHANICAL SYSTEMS

By

John D. Reid

The designing of mechanical systems has been a problem addressed by many great engineers. In the past few decades the problem has been approached by a method known as computer-aided design (CAD). This thesis presents an interactive, bond graph based approach to the computer-aided design of planar mechanisms, from schematic diagram to system animation. The mechanisms are composed of typical mechanical elements (e.g. springs, masses and dampers). The graphical input of the mechanical system is performed on an Evans and Sutherland PS300 with a PRIME 750 as the host computer. The system is modeled mathematically by a nonlinear Lagrangian bond graph. The model is formulated for simulation of its dynamic response by a bond graph processor. The dynamic response could then be used to animate the system, in order to complete the design process.

## ACKNOWLEDGMENTS

I would like to thank my major professor, Dr. Ronald Rosenberg. His knowledge, guidance and friendship has made this research possible.

A special thanks goes to the DeVlieg Foundation for providing the fellowship that enabled me to attend Michigan State University as a graduate student.

Finally, for her endless love and understanding I would like to thank my best friend and wife, Monica.

## TABLE OF CONTENTS

Page

Chapter

## LIST OF TABLES

## LIST OF FIGURES

Page

addr
Lagr
deco
comp
appr
sche

stud
type
body
mech
gene

stud
MAP
Plan
the

# Chapter 1

# INTRODUCTION

The mathematical modeling of mechanical systems has been a problem addressed by many great scientists, including Newton, Hamilton and Lagrange. With the advances in computer technology in the past few decades the problem has been approached by a method known as computer-aided design (CAD). This thesis presents an interactive approach to the computer-aided design of planar mechanical systems, from schematic diagram to system animation.

The crux of the problem consists of modeling the system and studying the dynamic response of the system. One distinction among types of problems depends upon the inertia characteristics: (1) Rigid body inertias and (2) Mass point inertias. Spatially, the range of mechanical design problems extends from simple 1-dimensional systems to generally complex 3-dimensional systems.

Systems containing both rigid bodies and mass points have been studied from two viewpoints, planar (2-D) and 3-dimensional systems. MAP [1], DRAM [2] and VECNET [3] are computer programs that simulate planar mechanisms, while IMP [4] and ADAMS [5] are programs that take the 3-D approach.

Mass point and rigid body systems may be studied by the use of bond graphs [6], [7], [8]. Bond graphs have generally been classified for processing into two types, linear and nonlinear. There exist today programs to handle both types of bond graphs. ENPORT [9] is a mature program for simulating linear systems, while recently POLSYAS [10], GEM [11], and CAMP [12] have been developed to handle nonlinear bond graphs.

Computer-aided design of mechanical systems consists of combining computer-aided drafting and mechanical problem simulation software into a complete mechanical design station. The design process starts with a schematic representation and the required geometry and parameters of the system to be designed. From there it is a five step process to design the mechanism in a CAD environment: namely,

1. Draw the design on the computer using an interactive program.
2. Model the system internally in some pre-determined way.
3. Calculate the dynamic response of the system.
4. Animate the graphic display according to the results from 3.
5. Change desired values and repeat the process until the design objectives are satisfied.

In this work the first step was performed on an Evans and Sutherland PS300 computer using a PRIME 750 as the host computer. The work has been carried out in the A. H. Case Center for Computer-Aided Design, a facility of the College of Engineering.

The system is modeled by a nonlinear bond graph. Specifically, the Lagrangian bond graph approach [13], [14] is applied to formulate the desired model. One of the advantages of using bond graphs is that they can handle problems involving several energy types, e.g. mechanical, electrical, hydraulic and so on. Bond graphs also allow one to study the dynamic structure of a problem by using causality, which can be a very useful aid to orginizing equations in nonlinear mechanisms.

It is intended that the dynamic response will be calculated from the bond graph models by a nonlinear bond graph processor. Or the models could be expressed in a form suitable for processing by another simulation program.

The work described in this thesis covers the graphic design portion and the internal modeling of the system such that it will be compatible with a nonlinear bond graph processor. Additional work will be required to use the dynamic response to animate the system, in order to complete the design process.

Chapter 2 describes the interactive program that is used for the graphical input of the mechanical system. Then in Chapter 3 we discuss the bond graph formulation program used to model the mechanical system. Chapter 4 illustrates the use and capabilities of the graphic design program (DESIGN) and the bond graph formulation program (BILDBG). Finally, a summary and some conclusions are discussed in Chapter 5.

# Chapter 2

# GRAPHIC DESIGN DEVELOPMENT

The objective of this section is to describe in detail an interactive program that can draw a desired mechanism on the computer using a set of standard mechanical elements. For an illustration of its use see Chapter 4, which can be read before this Chapter.

The program is menu driven such that satisfactory drawings can be created easily, saved, restored and modified. Items are added to the drawing one at a time. Each item can be translated, rotated and scaled to achieve its desired orientation. A label is added to each item for ease of identification. Each item, once added to the design, can be re-oriented, deleted or replaced by another item.

## 2.1 Program Design

The graphics program consists of three menus: 1. The Master Menu, 2. The Modify Menu and 3. The Drawing Menu. The three menus appear on the right hand side of the screen as shown in Figure 1. The menu that is in use at any particular time is high-lighted by a surrounding box. The function of each menu will now be examined.

C CREATE NEW FILE
R RESTORE OLD FILE
M MODIFY FILE
S SAVE FILE
E EXIT PROGRAM

A ADD NEW ITEM
C CHANGE ITEM
L LOCK IN ITEM
D DELETE ITEM
RP REPLACE ITEM
R RETURN

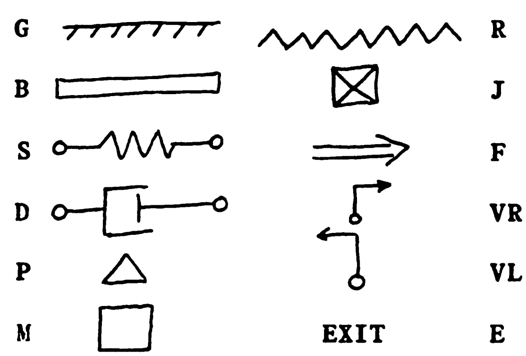


Figure 1. The Graphic Design Menus

### 2.1.1 The Master Menu

The Master Menu controls the main flow of the graphic design program. There are five options available within this menu (see Figure 2). The options perform the following tasks:

a. Create a new file.
b. Restore an old file.
c. Modify the current file in memory. This option causes the program control to go to the Modify Menu.
d. Save the current file.
e. Exit the program.

Note: Unless specified otherwise, the control of the program stays in the current menu after performing the desired option.

2.1

ava

2.1.

desi

whic

the

modi

whic

C    CREATE NEW FILE
R    RESTORE OLD FILE
M    MODIFY FILE
S    SAVE FILE
E    EXIT PROGRAM

Figure 2. The Master Menu

### 2.1.2 The Modify Menu

The modify menu controls the six modification options that are available for a given design (see Figure 3). These options are:

a. Add a new item to the design. This option causes the program control to go to the Drawing Menu.
b. Change the orientation of an item.
c. Lockin the current item to its current orientation.
d. Delete an item from the design.
e. Replace an item by another type of item.
f. Return to the Master Menu.

A    ADD NEW ITEM
C    CHANGE ITEM
L    LOCK IN ITEM
D    DELETE ITEM
RP   REPLACE ITEM
R    RETURN

Figure 3. The Modify Menu

### 2.1.3 The Drawing Menu

The drawing menu consists of the 11 items that are used to make a design (see Figure 4). Upon entering the Drawing Menu the user is asked which item he/she desires to add to the design. Acting upon this input, the chosen item is added and control of the program goes back to the modification menu (Modify). If no item is to be added there is an exit option that returns the program to the modification menu.

bar

sid

(Va.

2.2

the

desi

The m

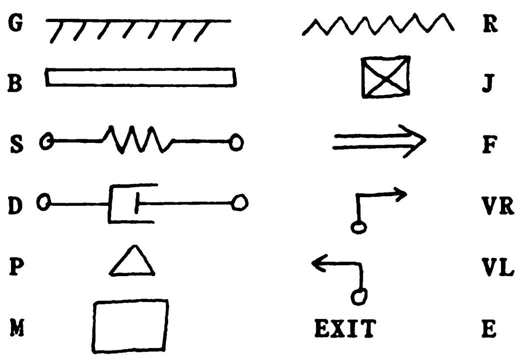


Figure 4. The Drawing Menu

The elements on the left side of Figure 4 consists of a ground (G), a bar (B), a spring (S), a damper (D), a pin (P), and a mass (M). The right side consists of a friction (R), a joint (J), a force (F), and a right (VR) and left (VL) velocity.

## 2.2 Data Base Design

The data base design incorporates the variable naming conventions and the storage requirements that handle the graphical representation of the design. Each drawing is composed of numerous items. Each item has:

1. An individual name.
2. A standard element type associated with it (e.g. spring, mass).
3. An orientation definition consisting of:

| | |
|---|---|
| x-translation | x-scaling |
| y-translation | y-scaling |
| z-rotation | overall scaling |

4. A label and its corresponding x and y position.

The naming conventions and storage requirements will be discussed next.

T

The n

of ele

previo

to add

alread

and th

MASS4

where

to the

Each

with

of Ba

the

SPRIN

eleme

This

an e

The individual name given to each item is stored in the array ITMLST. The name of each item is composed of the first few characters of the type of element it is, followed by the number of that type of element previously added to the drawing plus one. For example: suppose you want to add a mass to the drawing and there have been three masses added already. Then, the name of the new item is made by concatenating 'MASS' and the number of masses previously added plus one (i.e. NUMM+1). Thus, MASS4 is the name of the new item and will be stored in ITMLST( ITEMN ), where ITEMN is the number given to each item relative to when it is added to the drawing. NUMM stands for the NUMber of Masses added to the design. Each element has its own numbering system starting with NUM and ending with the first character of that type of element (e.g. NUMB is the NUMber of Bars added to the drawing).

The element type of each item is stored in ITMTYP. (e.g. Suppose the fourth item added is a spring. Then, ITEMN=4 and ITMTYP( ITEMN )= SPRING.)

The orientation of an item is given by:

```
x-translation = TRANSX ( ITEMN )
y-translation = TRANSY ( ITEMN )
z-rotation    = ROTZ   ( ITEMN )
x-scaling     = SCLX   ( ITEMN )
y-scaling     = SCLY   ( ITEMN )
overall scale = SCL    ( ITEMN )
```

The label of an item begins with the first letter of the type of element it is, followed by a user specified number (between 1 and 20). This allows the user freedom in the labelling convention and also provides an easy way to identify each item. No two labels can be identical. The label name is stored in ITMLBL. The label position is stored in LBLPOS. The x-position is stored in LBLPOS( ITEMN,1 ), while the y-position is stored in LBLPOS( ITEMN,2 ).

The origin, which is supplied on all drawings, is stored as item number 39. This allows the user to re-orient the origin to his/her specifications.

## 2.3 Program Implementation

Because the design program is menu-driven the programming code consists of an initialization section and the three menu handling sections discussed in 2.1. Figure 5 is a general flow chart of the design program. On the left side are the user inputs required by the program. On the right side is the computer output of the program. User input and computer output depend upon the user menu option selection.

The main subroutine calling tree structure is depicted in Figure 6.

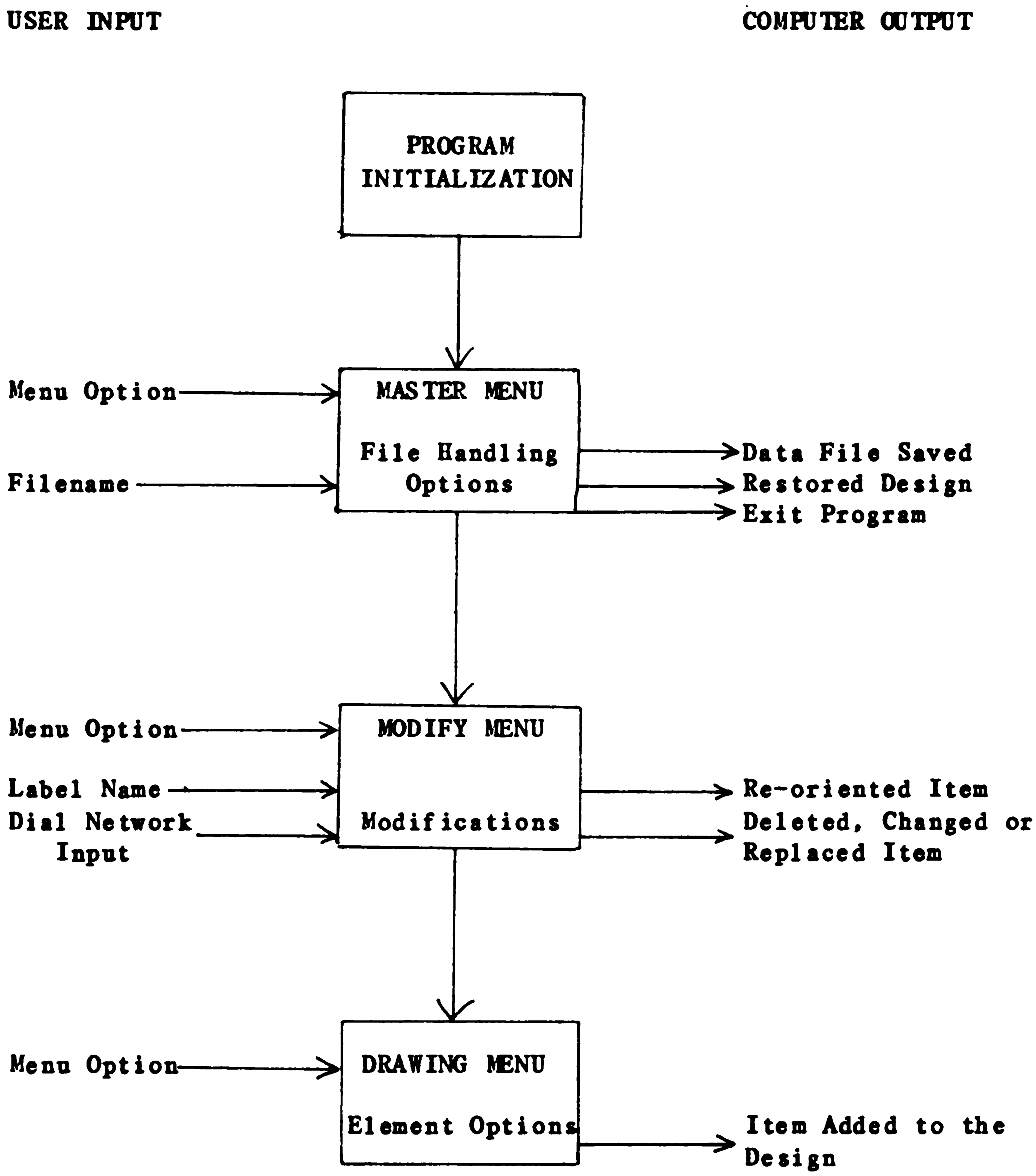


Figure 5. Graphic Design Flowchart

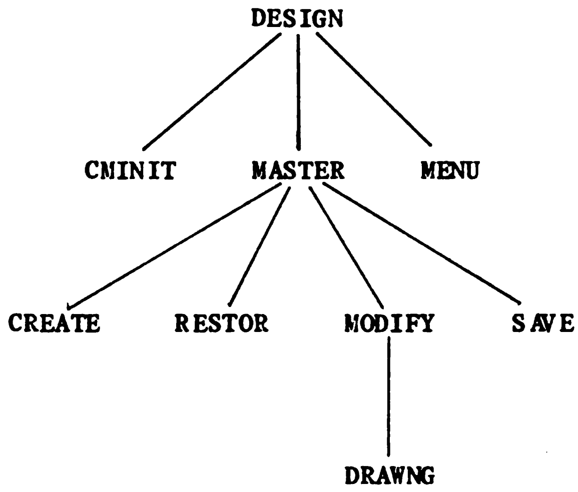


Figure 6. Graphic Design Calling Tree Structure

These main subroutines perform the following functions:

DESIGN - Acts like a driver program.
CMINIT - Initializes the variables.
MENU - Sets up the required Evans and Sutherland menus, objects and networks.
MASTER - Controls the Master Menu.
CREATE - Creates a new file.
RESTOR - Restores an old file.
MODIFY - Controls the Modify Menu.
SAVE - Saves the current file.
DRAWNG - Controls the Drawing Menu.

There are 15 utility subroutines that are called from a number of locations. These subroutines are named:

| | | |
|---|---|---|
| LABELS | ADITEM | GETDAT |
| POSLBL | REINIT | GETITM |
| UNLOCK | DISFIL | INCREM |
| LOCKIN | CONECT | INPUT |
| HLIGHT | DISCON | RESET |

The source code and descriptions of all the graphic design subroutines can be found in Appendix A.

# Chapter 3

# BOND GRAPH DEVELOPMENT

Once the graphic design is complete, the next step is to construct a bond graph. The objective of this chapter is to describe in detail how this task is accomplished. The program developed for this task will be referred to as BILDBG. An illustration of its use is given in Chapter 4, which can be read before this Chapter.

## 3.1 Procedure Design

The Lagrangian bond graph concept is used to develop the bond graph. The general Lagrangian bond graph is shown in Figure 7. The bond graph is developed by starting with the generalized coordinate velocities and building out towards the Compliance-field, Inertia-field, Resistive-field and Source-field through multi-port transformers (MTF's). The procedure used by BILDBG employs this concept with the exception that instead of generalized coordinate velocities, BILDBG uses what are to be known as motion point velocities.

Motion points are the elements that describe the motion of the system. The motion point elements are: masses, bars, joints, pins (fulcrums) and grounds. The allowable motions for each motion point describe the system motion point velocities. The allowable motions for the motion points are:

Masses - x, y (translational) and theta (angular) motion.
Bars - x, y (translational) and theta (angular) motion.
Joints - x and y (translational) motion.
Pins - No motion.
Grounds - No motion.

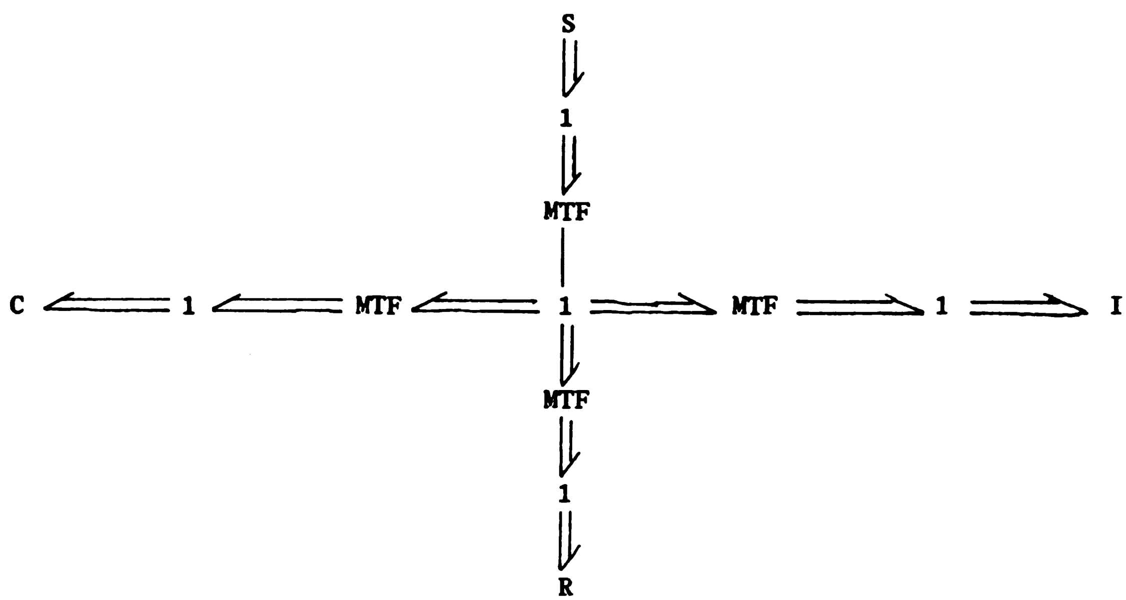


Figure 7. Lagrangian Bond Graph

Note. Nodes are sets of elements. Bonds are vector bonds.

In order for the motion points to be a complete description of the system motion, a design requirement must be imposed upon the drawing:

All elements that are not motion points must
be connected to motion points.

For example, a spring and damper in series must be connected through a joint. This will not effect the system solutions.

W

create

this c

is th

coord

corre

forc

comp

The

The

A

c

When motion points are connected to other motion points, BILDBG creates an internal rigid connector between the two motion points. If this occurs some of the motion points become dependent. This dependency is the difference between using motion point velocities and generalized coordinate velocities. However, the system results are still equivalent.

All elements that are not motion points are referred to as connectors. The connector elements are: springs, dampers, frictions, forces, velocities and rigid connectors. These elements define the compliance, resistive and source fields in the following way:

| | | |
|---|---|---|
| Compliance-field | : | Springs and rigid connectors. |
| Resistive-field | : | Dampers and frictions. |
| Source-field | : | Forces and velocities. |

Therefore, each connector has an element type that it is associated with. These types are referred to as connector types. The connector types are:

| | | |
|---|---|---|
| Springs | : | C-elements. |
| Dampers | : | R-elements. |
| Frictions | : | R-elements. |
| Forces | : | SE-elements. |
| Velocities | : | SF-elements. |

Although the rigid connector has its own unique definition, the rigid connector will be treated like a stiff spring.

The Inertia-field is made by appending I-elements to the appropriate mass motion point velocities.

In order to construct the bond graph the topology of the system must first be known.

### 3.1.1 The Topology of the System

There are two methods of obtaining the topology of the system:

1. Inspect the geometric description of each item and check to see if it is within a specified distance from another.
2. Ask the user for the connection network (i.e. topology).

Due to the complexity of implementing the first method, method two is used in this work.

By asking the user which items are connected to each motion point the connection network is obtained in a systematic way. If the user states that a motion point is connected to another motion point a rigid connector is created internally between the two motion points at that time.

### 3.1.2 The Bond Graph Model

BILDBG develops the bond graph in three steps, they are:

1. Create a 1-Junction for each motion point velocity.
2. Append I-elements to the mass motion 1-Junctions and create the bonds between.
3. For each connector, create a node corresponding to the connector type. Then build the bond graph from the connector to the motion point velocities that the connector is attached to.

For step 1, the number of motion point velocities is equal to the sum of the number of allowable motions for each motion point. There are three allowable motions for each mass and bar (x,y and theta). There are two allowable motions for each joint (x and y). Therefore, the number of motion point velocities (NMPVEL) is given by the expression

$$NMPVEL = 3*NUMM + 3*NUMB + 2*NUMJ \qquad (3.1)$$

Step 2 consists of appending I-elements to each mass motion point velocity. Therefore, there are three times the number of masses I-elements added to the bond graph. A bond is then created between each I-element and its corresponding mass motion point velocity.

The third step starts by creating a node corresponding to each connector. In general, each connector has two end points (EP1 and EP2), and each end point (EP) has an associated x and y motion. See Figure 8.

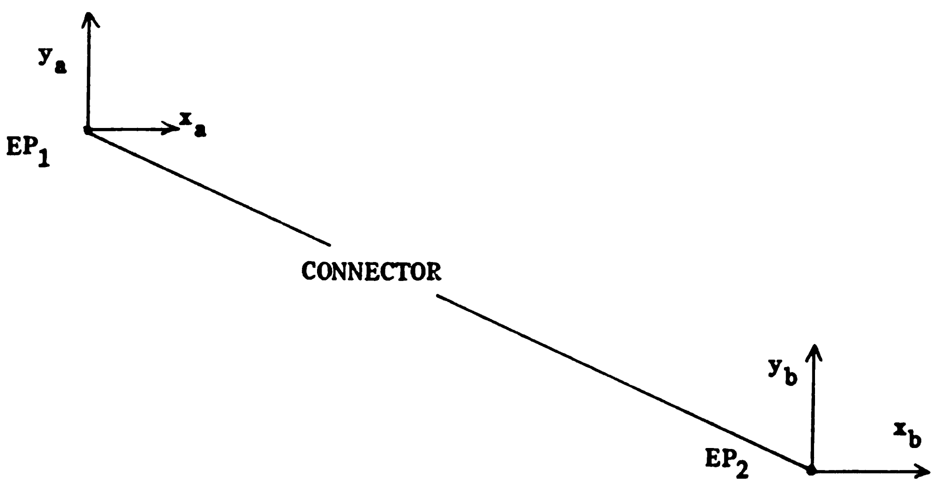


Figure 8. A General Connector

The relative displacement of a connector (L) is given by

$$L = \left[ (x_a - x_b)^2 + (y_a - y_b)^2 \right]^{1/2} \qquad (3.2)$$

where a and b correspond to end points 1 and 2 respectively. The relative velocity is found by differentiating equation (3.2). This is found to be

$$L = r_1 * \dot{x}_a + r_2 * \dot{y}_a + r_3 * \dot{x}_a + r_4 * \dot{y}_b \tag{3.3}$$

where

$$r_1 = A*(x_a - x_b) \tag{3.3a}$$

$$r_2 = A*(y_a - y_b) \tag{3.3b}$$

$$r_3 = -A*(x_a - x_b) \tag{3.3c}$$

$$r_4 = -A*(y_a - y_b) \tag{3.3d}$$

$$A = \left[ (x_a - x_b)^2 + (y_a - y_b)^2 \right]^{-1/2} \tag{3.3e}$$

Note. A super-dot indicates a time-derivative.

A general bond graph can now be constructed between a connector and its associated end points (see Figure 9). Special cases of connector types will be discussed in section 3.1.3 of this chapter. These special cases will allow the bond graph to be simplified in certain systems.

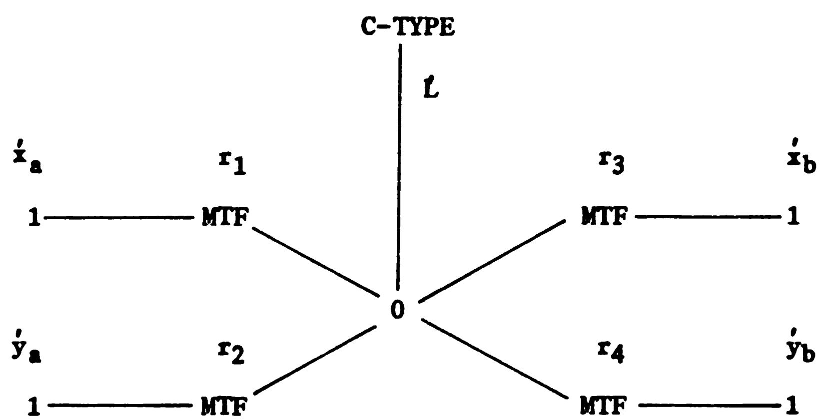


Figure 9. Bond Graph Between a Connector and its End Points

Each connector end point is connected to a motion point. To complete the bond graph, the relationship between the connector end point velocities and the motion point velocities that the connector is connected to, must be determined. This relationship depends upon the type of motion point in question.

Figure 10 shows a connector attached to a mass. In general, the mass motion point has x, y and theta motion. The mass motion point is considered the center of the mass. The connector end point has only x and y motion. The relation between the two is given by

$$x_{EP} = x_{MP} + a*\cos(\theta_{MP}) - b*\sin(\theta_{MP}) \qquad (3.4)$$

$$y_{EP} = y_{MP} + a*\sin(\theta_{MP}) + b*\cos(\theta_{MP}) \qquad (3.5)$$

where a and b are the x and y distances between the mass motion point and the connector end point respectively.

Differentiating equations (3.4) and (3.5) gives us the desired velocity relationships: namely,

$$\dot{x}_{EP} = \dot{x}_{MP} + r_1*\dot{\theta} \qquad (3.6)$$

$$\dot{y}_{EP} = \dot{y}_{MP} + r_2*\dot{\theta} \qquad (3.7)$$

where

$$r_1 = -\left[\ a*\sin(\theta_{MP}) + b*\cos(\theta_{MP})\ \right] \qquad (3.6a)$$

$$r_2 = a*\cos(\theta_{MP}) - b*\sin(\theta_{MP}) \qquad (3.7a)$$

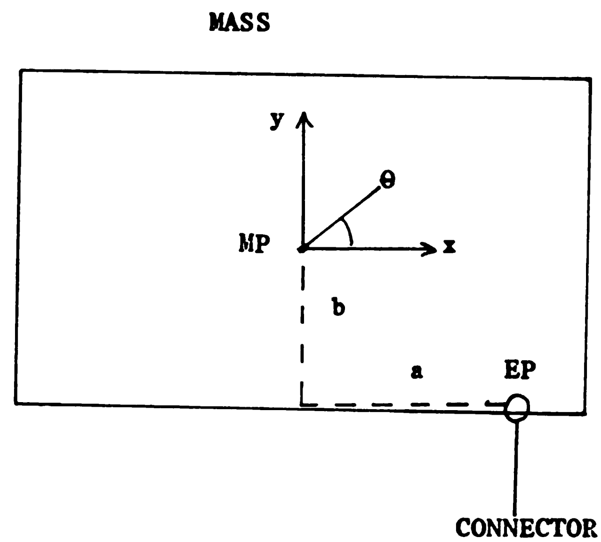


Figure 10. A Connector Attached to a Mass

The bond graph is now defined between a connector end point and a mass motion point (see Figure 11).

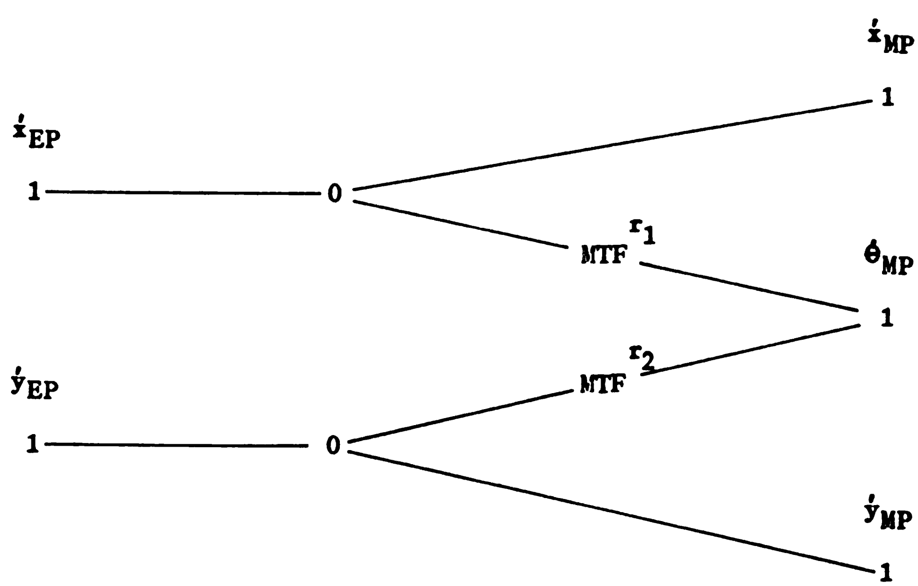


Figure 11. Bond Graph Between a Connector EP and a Mass MP

The bar (massless rod) is similar in motion to the mass element. The difference between the two elements is that the bar has no moment of inertia. Figure 12 shows the required geometry in order to define the relationship between a connector end point and a bar motion point. The bar motion point is defined as the point on the bar that is considered its rotation point.

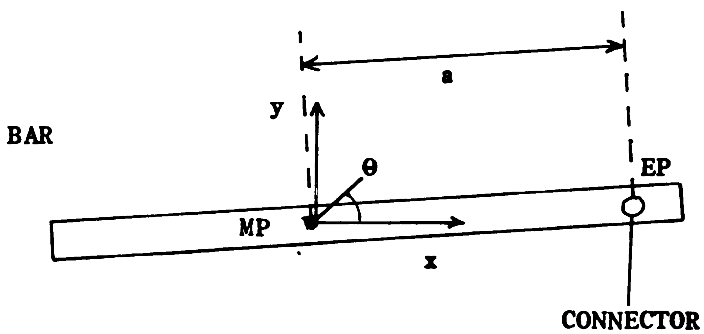


Figure 12. A Connector Attached to a Bar

The corresponding equations between a bar and a connector end point are

$$x_{EP} = x_{MP} + a*\cos(\theta_{MP}) \tag{3.8}$$

$$y_{EP} = y_{MP} + a*\sin(\theta_{MP}) \tag{3.9}$$

Differentiating gives

$$\dot{x}_{EP} = \dot{x}_{MP} + r_1*\dot{\theta} \tag{3.10}$$

$$\dot{y}_{EP} = \dot{y}_{MP} + r_2*\dot{\theta} \tag{3.11}$$

where

$$r_1 = -a*\sin(\theta_{MP}) \tag{3.10a}$$

$$r_2 = a*\cos(\theta_{MP}) \tag{3.11a}$$

The bond graph between a connector end point and a bar motion point is equivalent to the bond graph used for a mass motion point with equations (3.9a) and (3.10a) used in the multi-port transformer equations. (See Figure 11).

When a connector end point is attached to a joint motion point, the end point is considered to move the same as the joint. Figure 13 depicts the required bond graph for this case. However, in general, the bond graph of Figure 13 is equivalent to the bond graph of Figure 11 with the transformer moduli ($r_1$ and $r_2$) equal to zero.

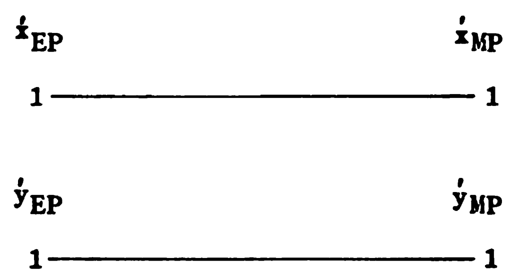


Figure 13. Bond Graph Between a Connector EP and a Joint MP

A connector end point attached to a pin or ground requires no nodes or bonds attached to that particular end point. However, in general, the bond graph is the same as shown in Figure 11. The fact that there is no motion at that particular end point will become apparent because the motion points for pins and grounds have no displacements and zero velocities.

The general bond graph for each connector has now been defined (see Figure 14). It is easy to see from Figure 14 that even the simplest systems would be modelled by a large number of nodes and bonds. The next section of this chapter will discuss some of the reduction techniques available to simplify this general bond graph.

### 3.1.3 The Simplified Bond Graph

In a given system each motion point has an associated motion type. The motion type defines the allowable motions for a given motion point. There are eight different motion types, see Table 1.

Table 1. Motion Types

| MOTION TYPE | ALLOWABLE MOTION | REQUIRED BOND GRAPH |
|---|---|---|
| 0 | No Motion | Figure 15 |
| 1 | X only | Figure 16 |
| 2 | Y only | Figure 17 |
| 3 | X and Y | Figure 18 |
| 4 | Theta only | Figure 19 |
| 5 | X and Theta | Figure 20 |
| 6 | Y and Theta | Figure 21 |
| 7 | X, Y and Theta | Figure 11 |

Specifically, the ground and pin motion points have motion type 0. Joints can have motion types 0, 1, 2 or 3. Bars and masses can have any of the eight types.

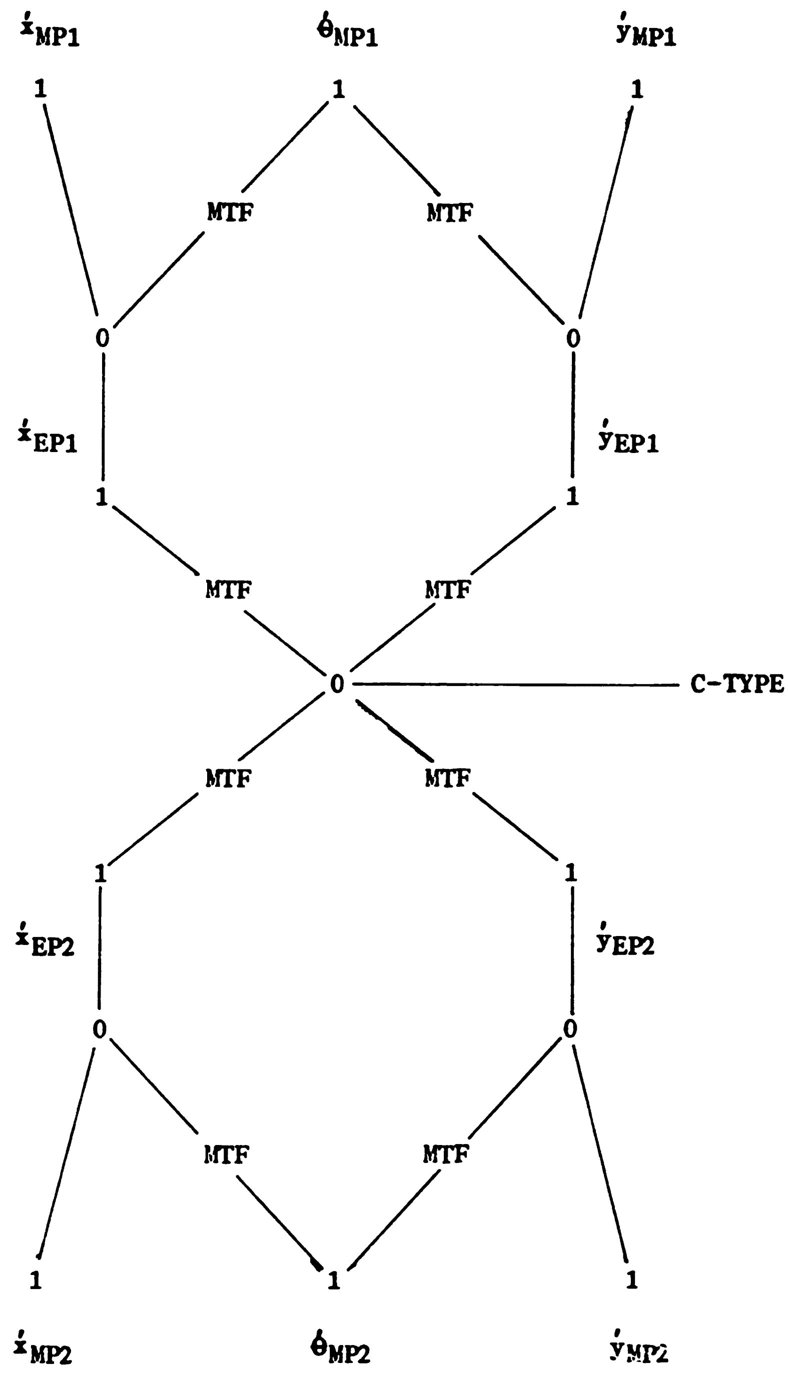


Figure 14. The Bond Graph for a Connector

poi

mot

dev

mot

gra

7.

pr

The bond graph between a motion point and an attached connector end point is constructed in BILDBG according to the motion type for that motion point. Figures 15 through 21 show the bond graphs that are developed for motion types 0 through 6 respectively. The bond graph for motion type 7 is the same as Figure 11. It is noted that the bond graphs for motion types 0 through 6 are simplifications of motion type 7.

The motion types for each motion point are obtained by asking the program operator for the information.


Figure 15. Bond Graph for Motion Type 0

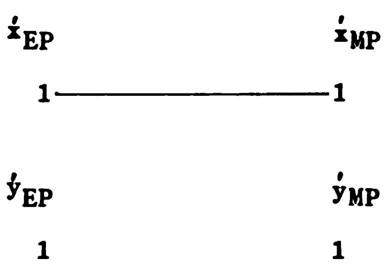


Figure 16. Bond Graph for Motion Type 1

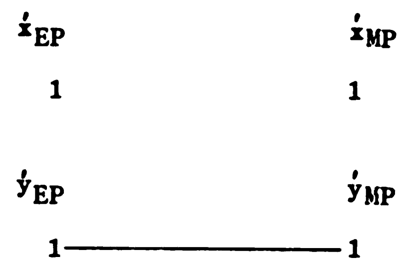


Figure 17. Bond Graph for Motion Type 2

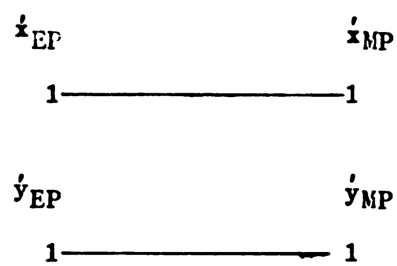


Figure 18. Bond Graph for Motion Type 3

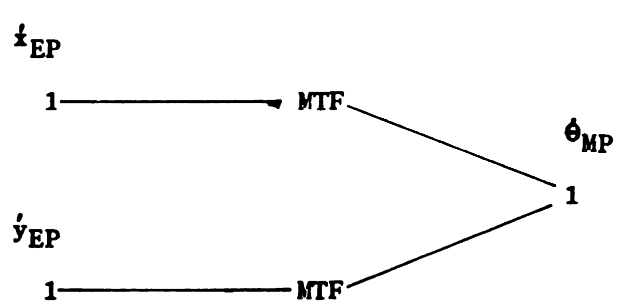


Figure 19. Bond Graph for Motion Type 4

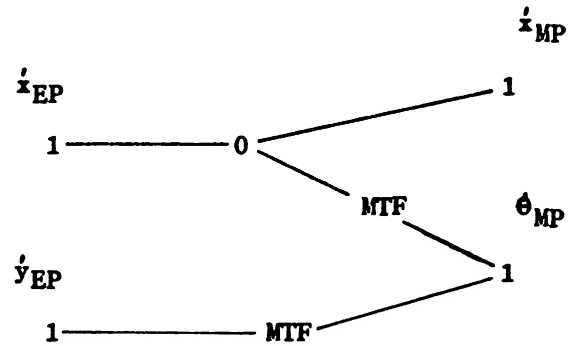


Figure 20. Bond Graph for Motion Type 5

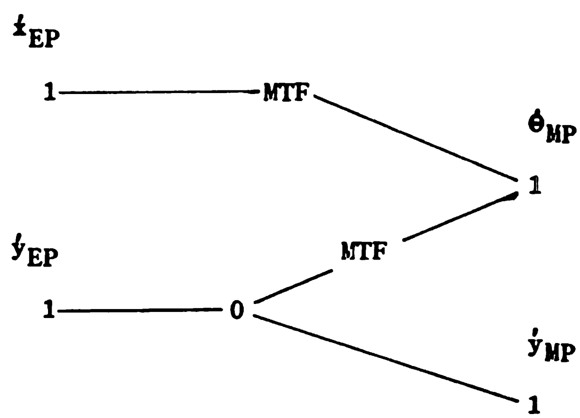


Figure 21. Bond Graph for Motion Type 6

It is apparent from Figures 15 through 21 that certain end point 1-Junctions are not required, (i.e. the end points that do not extend to the motion point 1-Junctions). This further simplification is performed by studying each connector.

In general, each connector is attached to two motion points. Each of these two motion points has an associated motion type. Because of this, there are twelve different types of bond graphs from the connector node to the connector end point 1-Junctions. These different types are referred to as end point types. Table 2 describes each of these end point types. Note that only five bond graphs are required to demonstrate the end point types, but these five bond graphs actually represent twelve different bond graphs when implemented in BILDBG.

Table 2. End Point Types

| END POINT TYPE | MOTION POINT 1 MOTION TYPE | MOTION POINT 2 MOTION TYPE | REQUIRED BOND GRAPH |
|---|---|---|---|
| 1 | 1 or 2 | 0 | Figure 22 |
| OR 1 | 0 | 1 or 2 | |
| 2 | 3, 4, 5, 6 or 7 | 0 | Figure 23 |
| 3 | 0 | 3, 4, 5, 6 or 7 | Figure 23 |
| 4 | 1 | 1 | Figure 24 |
| 5 | 1 | 2 | Figure 24 |
| 6 | 2 | 1 | Figure 24 |
| 7 | 1 | 2 | Figure 24 |
| 8 | 3, 4, 5, 6 or 7 | 1 | Figure 25 |
| 9 | 3, 4, 5, 6 or 7 | 2 | Figure 25 |
| 10 | 1 | 3, 4, 5, 6 or 7 | Figure 25 |
| 11 | 2 | 3, 4, 5, 6 or 7 | Figure 25 |
| 12 | 3, 4, 5, 6 or 7 | 3, 4, 5, 6 or 7 | Figure 9 |

Thus, the bond graph between a connector and the motion points that it is connected to is generated in the following way:

1. Create a node according to the connector type.
2. Determine the end point type of the connector.
3. Develop a portion of the bond graph according to Table 2.
4. Determine the motion type of the motion points attached.
5. Finish the bond graph for this connector according to Table 1.

Examples of this process will be given in Chapter 4.

Certain connectors are shown as having only one end point. Inputs of force and velocity. However, these connectors can be treated as if they did have two end points with the second end point having motion type 0.

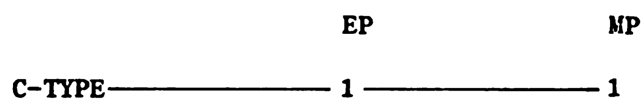


Figure 22. Bond Graph for End Point Type 1

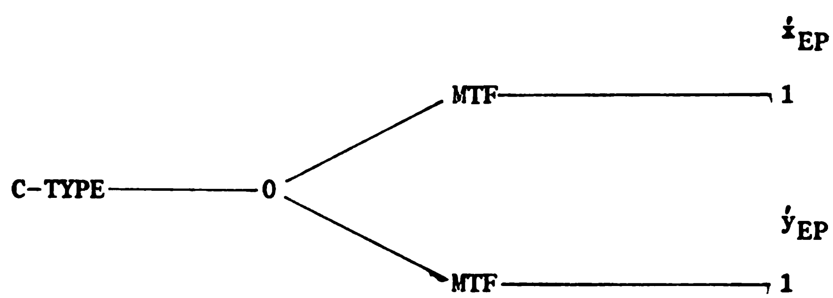


Figure 23. Bond Graph for End Point Types 2 and 3

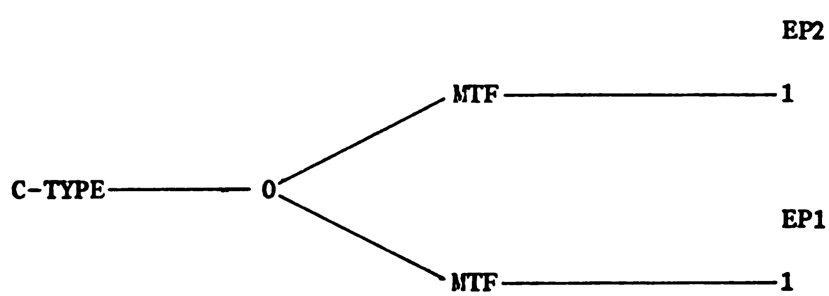


Figure 24. Bond Graph for End Point Types 4, 5, 6 and 7

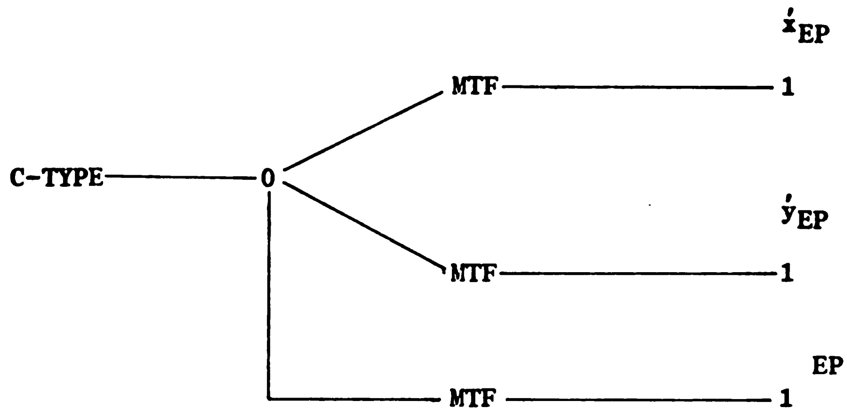


Figure 25. Bond Graph for End Point Types 8, 9, 10 and 11

## 3.2 Data Base Design

The data base consists of four sections: namely,

1. The item lists.
2. The topology of the system.
3. The geometry of the system.
4. The bond graph.

The item lists separate the motion points and connectors and also store the type of each. The motion points are stored in MPNAM in the following order: masses, bars, joints, grounds and pins. The number of motion points is NMP. The motion type associated with each motion point is stored in MOTTYP. The number of allowable motions for each motion point is stored in NMOT( MOTTYP ).

The connectors are stored in CNNAM in the following order: springs, dampers, frictions, forces, velocities and rigid connectors. The number of connectors is NCN. The end point type associated with each connector is stored in IEPTYP.

The topology of the design is stored in two ways. The first way is a list of motion points on a given connector (MPONCN). The motion point attached to end point 1 of connector NCN is stored in MPONCN( NCN,1 ), while the motion point attached to end point 2 is stored in MPONCN( NCN,2 ).

The second method of storing the topology is by a list of connectors incident on a given motion point (CNONMP). CPTR is the pointer that identifies which connectors in CNONMP are attached to a given motion point. For example, the connectors attached to motion point NMP are found in CNONMP( CPTR( NMP ) ) through CNONMP( CPTR( NMP+1 ) - 1 ).

The required geometry of the system is stored in GEOMAS and GEOBAR. GEOMAS stores the local geometry between a mass center and an attached connector. For example, let connector NCN be attached to the mass NMASS. Then the x-distance from the mass center to the connector end point is stored in GEOMAS( NMASS,NCN,1 ), while the y-distance is stored in GEOMAS( NMASS,NCN,2 ). GEOBAR stores the local geometry between a bar (point of rotation) and an attached connector. For example, let connector NCN be attached to the bar NBAR. Then the distance between the two is stored in GEOBAR( NBAR,NCN ).

The bond graph is stored in two parts. First, the actual bond graph, consisting of nodes and bonds, is stored in the same manner as in ENPORT: namely,

IELLST - List of nodes by type number.
NBIMX - Bonds incident on a given node.

IBMX - Nodes adjacent to a given bond.
NPTR - Pointer to start of bonds in NBIMX for node I.
NEL - Number of nodes.
NBD - Number of bonds.

Second, the transformers and the relationship between the bond graph and the graphic design must be maintained.

The relationship between the bond graph and the graphic design consists of maintaining the position and velocity of the motion points, connectors and connector end points. The position and velocity for the motion points are stored in XMP and VMP respectively. Each motion point is considered to have three positions and velocities, namely, x, y and theta. Thus, the position and velocity of motion point NMP is found in:

```
x-position  -  XMP( NMP*3-2 )
y-position  -  XMP( NMP*3-1 )
θ-position  -  XMP( NMP*3 )
x-velocity  -  VMP( NMP*3-2 )
y-velocity  -  VMP( NMP*3-1 )
θ-velocity  -  VMP( NMP*3 )
```

IXMPPT points to the location in XMP and VMP for a given node.

The position and velocity of a connector end point are stored in XEP and VEP respectively. Each connector has two end points and each end point has an x and y description. Therefore, each connector requires four locations in XEP and VEP for their end point values. For example, the position and velocity of connector NCN end points are found in:

```
x-position of EP1  -  XEP( NCN*4-3 )
y-position of EP1  -  XEP( NCN*4-2 )
x-position of EP2  -  XEP( NCN*4-1 )
y-position of EP2  -  XEP( NCN*4 )
x-velocity of EP1  -  VEP( NCN*4-3 )
```

y-velocity of EP1 - VEP( NCN*4-1 )
x-velocity of EP2 - VEP( NCN*4-2 )
y-velocity of EP2 - VEP( NCN*4 )

IXEPPT points to the associated node number of a given end point. For example, the node corresponding to the x-displacement of end point 2 of connector NCN is IXEPPT( NCN*4-1 ).

The connectors require a relative displacement and velocity. These values are stored in XCN and VCN. IXCNPT points to the node corresponding to a given connector. The values for a connector are calculated by Function 5. Function 5 is described later in this section.

The transformers are stored in sequential order in MTF. For example, MTF( NTF ) is the node number of transformer NTF. The required information pertaining to a given transformer is stored in MTFDAT. Figure 26 depicts the general layout of MTFDAT. The value of a transformer is stored in MTFVAL.

| MTF | | | | | | |
|---|---|---|---|---|---|---|
| 1 | FUNCTION NUMBER | CONSTANTS | ARGUMENTS | MODULATED FUNCTION NUMBER | NUMBER OF POINTERS | POINTERS |
| 2 | | | | | | |
| 3 | | | | | | |
| . | | | | | | |
| . | | | | | | |
| . | | | | | | |
| NTF | | | | | | |

Figure 26. Layout of Transformer Data (MTFDAT)

Five function types are required to handle the transformers. Function types 1 through 4 handle the transformers between a connector end point and a motion point theta 1-Junction. These functions correspond to the following transformers:

Function 1 - Transformer between a connector $\dot{x}$ 1-Junction and a mass motion point $\dot{\theta}$ 1-Junction.
Function 2 - Transformer between a connector $\dot{y}$ 1-Junction and a mass motion point $\dot{\theta}$ 1-Junction.
Function 3 - Transformer between a connector $\dot{x}$ 1-Junction and a bar motion point $\dot{\theta}$ 1-Junction.
Function 4 - Transformer between a connector $\dot{y}$ 1-Junction and a bar motion point $\dot{\theta}$ 1-Junction.

These four functions require a function number, 1 or 2 constants and 1 argument. The constant(s) is the local geometry between the motion point and the attached connector end point. The argument is a pointer to the location of the value of theta of the motion point involved. This value is stored in XMP( argument ), as mentioned previously.

Function type 5 handles the transformers between a connector and its end point 1-Junctions. There are no constants required. There are four arguments required. These arguments are the pointers to the x and y positions of each end point in XEP. The modulated function number is the number corresponding to the MTF in question of Figure 9. For example, let the modulated function number be 2. Then, from Figure 9 we know the MTF in question is the MTF between the 0-Junction and the $\dot{y}_a$ 1-Junction. This number is used to identify which expression in Function 5 that gives the value for this transformer. Note, Function 5 evaluates multiple expressions. Some of these expressions are the desired values for the other transformers associated with the connector that is associated with this transformer. Next, in MTFDAT, is the number of pointers that follow. These pointers are used to identify the other transformers in MTF that have been satisfied due to this transformer calling Function 5.

The transformer functions and descriptions can be found in Appendix B.

### 3.3 Program Implementation

The flow of BILDBG is depicted in Figure 27. The calling tree structure of BILDBG is shown in Figure 28. The main subroutines of BILDBG perform the following functions:

| | | |
|---|---|---|
| BILDBG | - | The driver of the program. |
| RESTOR | - | Restores the graphical data of the system. |
| REORDR | - | Re-orders the items into motion points and connectors. |
| INITBG | - | Initializes the required variables. |
| CONECT | - | Determines the topology of the system. |
| GETMOT | - | Gets the allowable motion for each motion point. |
| GETGEM | - | Gets the required local geometry between the mass and bar motion points and there attached connector end points. |
| BG | - | Builds the bond graph. |
| RESULT | - | Writes all pertinent information to a file. |

Note: BILDBG calls the above subroutines in the same order as they are listed.

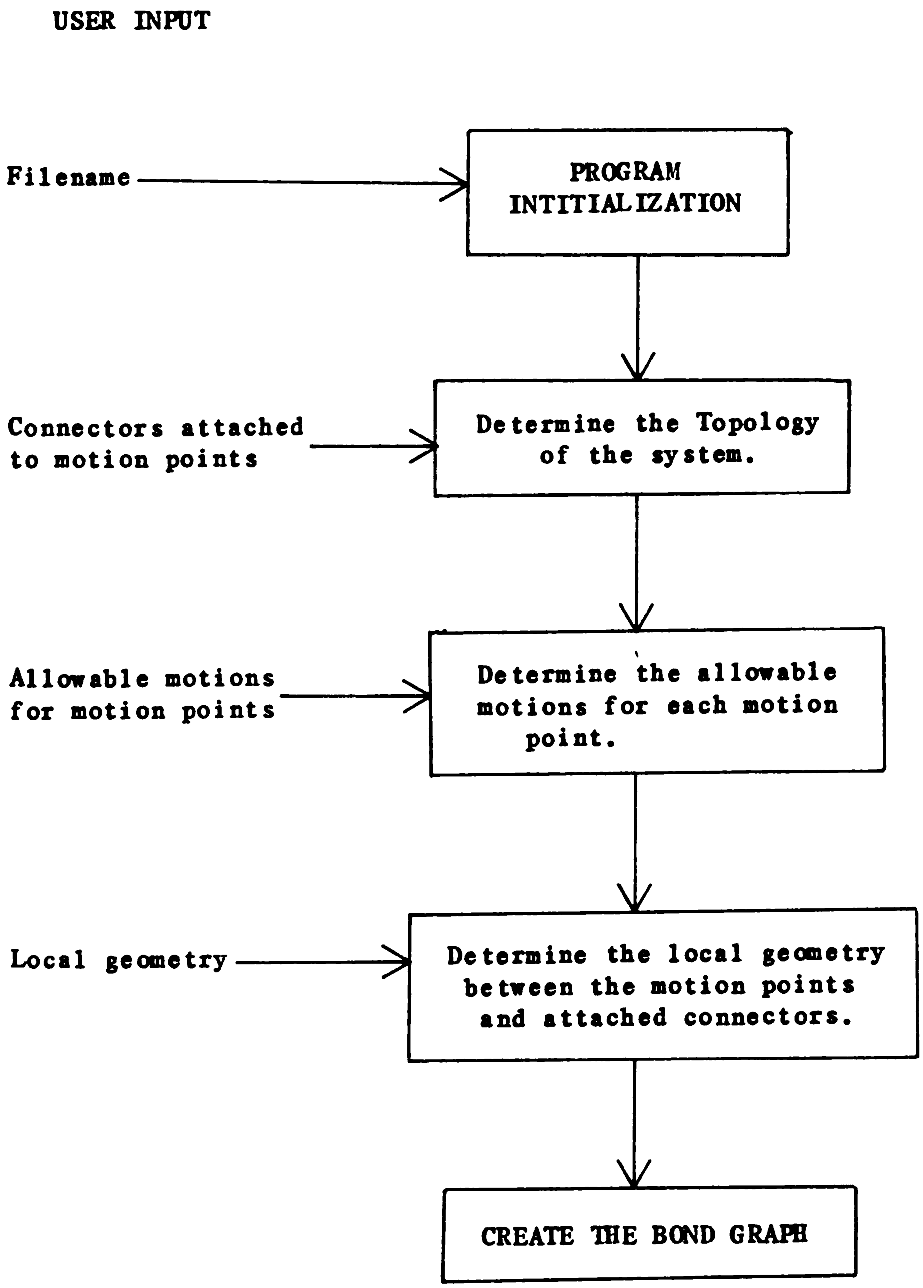


Figure 27. BILDBG Flowchart

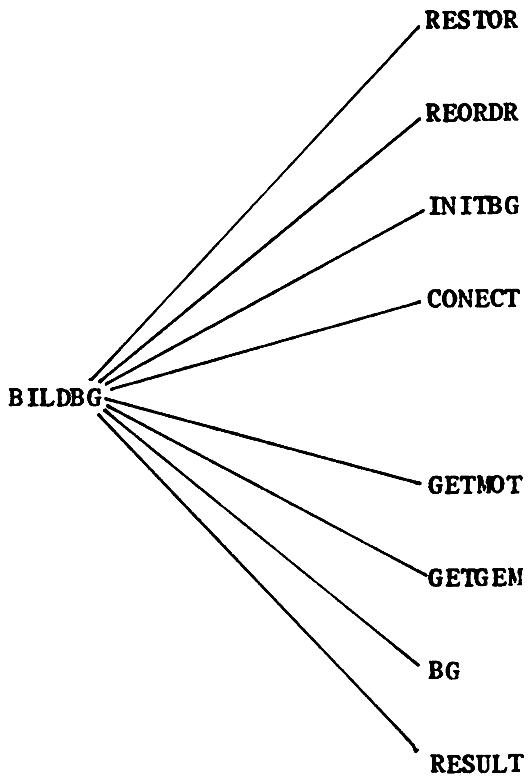


Figure 28. Calling Tree Structure of BILDBG

Subroutine BG is the subroutine that actually builds the bond graph. The flow of BG is depicted in Figure 29. Figure 30 depicts BG's calling tree structure. The main subroutines of BG perform the following functions:

| | | |
|---|---|---|
| BG | - | The driver for developing the actual bond graph. |
| GETEPS | - | Gets the end point type of a specified connector. |
| CNTFEP | - | Stores the transformer data between a connector and its end points. |
| CNMP3 | - | Creates the bond graph from a connector 0-Junction to the specified motion point 1-Junction for motion types 1, 2 or 3. |
| CNMP4 | - | Creates the bond graph from a connector 0-Junction to the specified motion point 1-Junctions for motion type 4. |
| CNMP5 | - | Creates the bond graph from a connector 0-Junction to the specified motion point 1-Junctions for motion type 5. |
| CNMP6 | - | Creates the bond graph from a connector 0-Junction to the specified motion point 1-Junctions for motion type 6. |
| CNMP7 | - | Creates the bond graph from a connector 0-Junction to the specified motion point 1-Junctions for motion type 7. |
| BG0TF1 | - | Builds the bond graph from a connector 0-Junction to the specified end point 1-Junction through a transformer. |
| BGEPTT | - | Builds the bond graph from an end point 1-Junction to a motion point theta 1-Junction through a transformer. |
| BGEPMP | - | Builds the bond graph from an end point 1-Junction to a motion point x or y, and theta 1-Junctions. |
| BNDNOD | - | Creates the bonds incident on a given node array. |

The source code and descriptions of all the bond graph subroutines can be found in Appendix C.

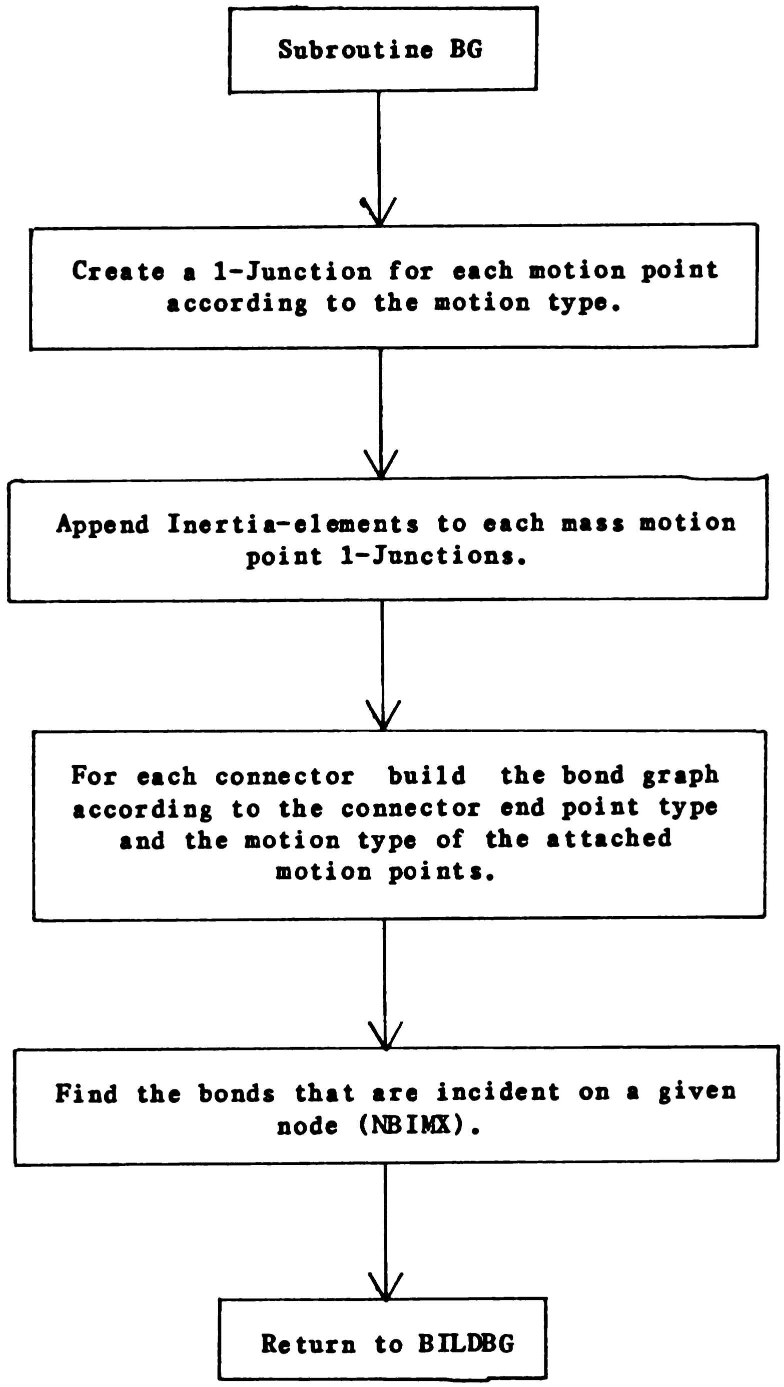


Figure 29. BG Flowchart

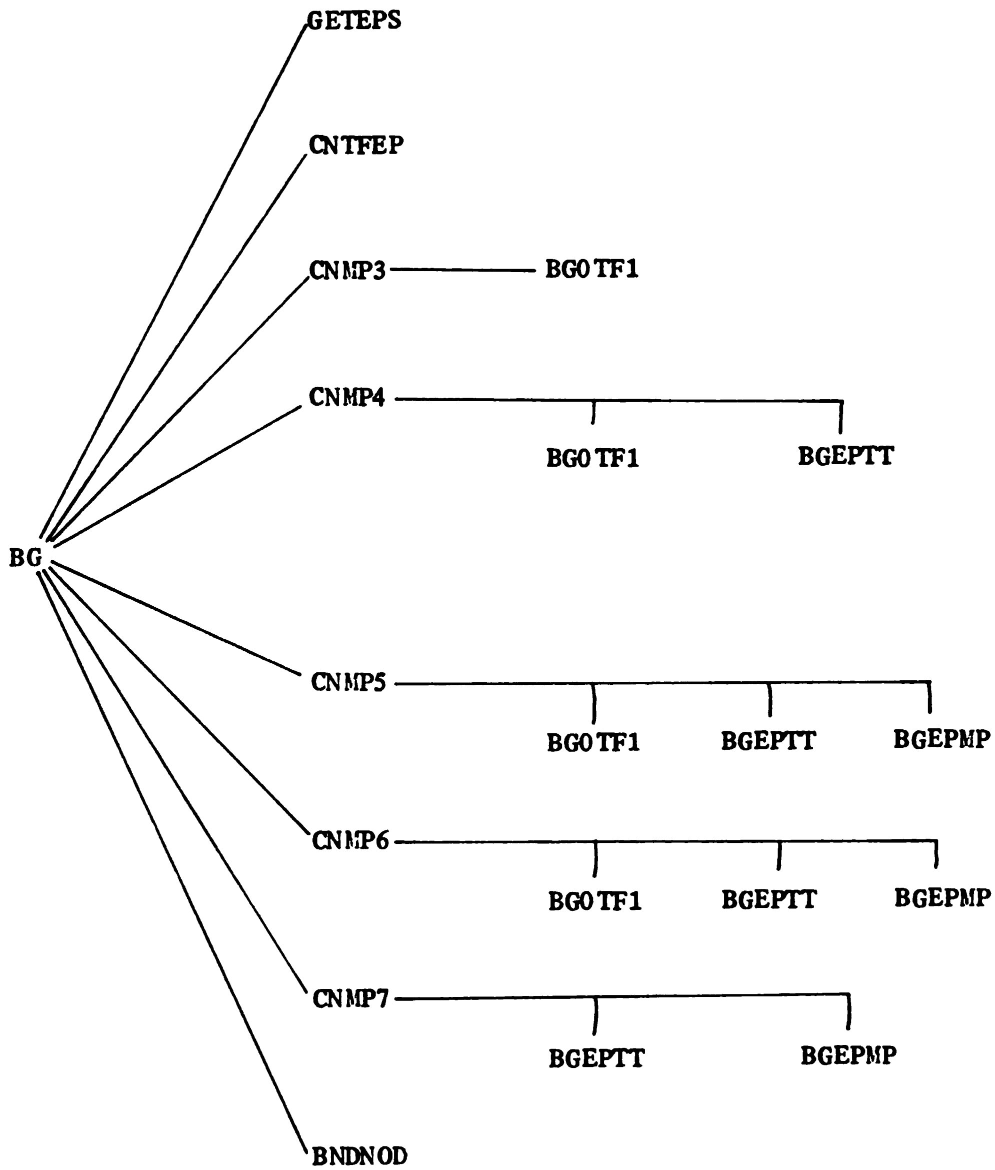


Figure 30. Calling Tree Structure of BG

# Chapter 4

# EXAMPLES

This chapter illustrates the use and capabilities of the two programs described in Chapters 2 and 3, namely, the graphic design program (DESIGN) and the bond graph formulation program (BILDBG). To run these programs one must be logged into an Evans and Sutherland PS300 terminal. The procedure to accomplish this task is located at the PS300 terminal.

## 4.1 Spring-mass-damper System

The spring-mass-damper system of Figure 31 was drawn on the PS300 using the program DESIGN. The following is an example run on how this system was created. Please note that a description of each command is given in parentheses.

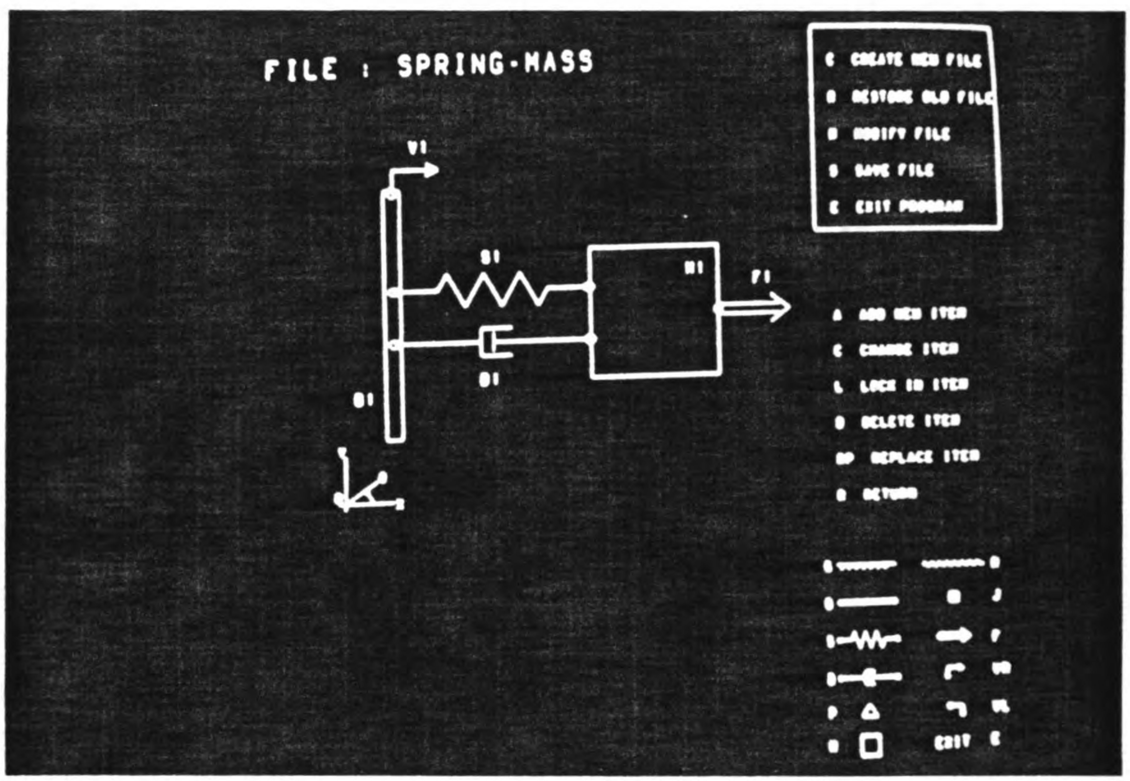


**Figure 31. Spring-mass-damper System**

OK, SEG DESIGN

```
*****************************************************
*****************************************************
**                                                 **
**                                                 **
**           EVANS AND SUTHERLAND PS300            **
**                                                 **
**             GRAPHIC DESIGN PROGRAM              **
**                                                 **
**                                                 **
**                  DEVELOPED BY                   **
**                                                 **
**                   JOHN REID                     **
**                                                 **
**                   FALL 1983                     **
**                                                 **
**                                                 **
*****************************************************
*****************************************************
```

INPUT DESIRED OPTION : C
(Create a new file.)

INPUT NAME OF NEW FILE : SPRING-MASS

INPUT DESIRED OPTION : M
(Modify the current file.)

INPUT DESIRED OPTION : A
(Add an item to the design.)

INPUT DESIRED OPTION : M
(A mass is chosen.)

(Position and scale the mass to the desired orientation
by using the dial network.)

INPUT DESIRED OPTION : L
(Lock in the mass to its current position.)

INPUT INDEX NUMBER FOR LABEL (1-20) : 1

USING THE DIALS, POSITION THE CROSS-HAIRS IN THE
DESIRED LOCATION FOR THE LABEL.

TYPE OK TO CONTINUE : OK
(The label M1 has now been placed at the specified position.)

(Continue adding items one at a time in the above manner
until the design is complete.)

```
INPUT DESIRED OPTION : C
(Change the position of an item.)

INPUT LABEL NAME : O
(The origin is chosen.)

(Position and scale the origin with the dial network.)

INPUT DESIRED OPTION : L
(Lock in the origin to its current position.)

INPUT DESIRED OPTION : R
(Return to the Master Menu.)

INPUT DESIRED OPTION : S
(Save the current design.)

FILE SPRING-MASS HAS BEEN SAVED.

INPUT DESIRED OPTION : E
(Exit program.)

*****  YOU ARE NOW LEAVING DESIGN.  *****
```

Once the system description is completed by using DESIGN a bond graph can be developed by running BILDBG. A sample run of BILDBG for our example follows.

```
OK, SEG BILDBG

    ***************************************************
    ***************************************************
    **                                               **
    **                                               **
    **          EVANS AND SUTHERLAND PS300           **
    **                                               **
    **              BOND GRAPH PROGRAM               **
    **                                               **
    **                                               **
    **                 DEVELOPED BY                  **
    **                                               **
    **                  JOHN REID                    **
    **                                               **
    **                  FALL 1983                    **
    **                                               **
    **                                               **
    ***************************************************
    ***************************************************
```

```
INPUT THE NAME OF THE GRAPHIC FILE TO BE PROCESSED : SPRING-MASS


WHICH ITEMS ARE CONNECTED TO M1 ? : S1,D1,F1

AS I UNDERSTAND IT, THE FOLLOWING ITEMS ARE CONNECTED TO M1 .

    S1      D1      F1

IS THIS CORRECT? (Y OR N) : Y


WHICH ITEMS ARE CONNECTED TO B1 ? : S1,D1,V1

AS I UNDERSTAND IT, THE FOLLOWING ITEMS ARE CONNECTED TO B1 .

    S1      D1      V1

IS THIS CORRECT? (Y OR N) : Y


WHICH MOTIONS ARE ALLOWABLE FOR M1  -- X, Y AND/OR THETA : X


WHICH MOTIONS ARE ALLOWABLE FOR B1  -- X, Y AND/OR THETA : X


LET THE CENTER OF M1  BE THE POINT 0,0.

FROM THIS REFERENCE POINT WHAT IS THE X,Y DISTANCE BETWEEN
M1  AND THE END POINT OF S1  : -1,1


LET THE CENTER OF M1  BE THE POINT 0,0.

FROM THIS REFERENCE POINT WHAT IS THE X,Y DISTANCE BETWEEN
M1  AND THE END POINT OF D1  : -1,-1


LET THE CENTER OF M1  BE THE POINT 0,0.

FROM THIS REFERENCE POINT WHAT IS THE X,Y DISTANCE BETWEEN
M1  AND THE END POINT OF F1  : 1,0


ASSUME B1  IS PARALLEL TO THE X-AXIS.
LET THE POINT WHERE B1  ROTATES ABOUT BE 0,0.

FROM THIS REFERENCE POINT, WHAT IS THE DISTANCE TO THE ENDPOINT OF S1 .
( INCLUDE + OR - SIGN ) : 1
```

```
ASSUME B1  IS PARALLEL TO THE X-AXIS.
LET THE POINT WHERE B1  ROTATES ABOUT BE 0,0.

FROM THIS REFERENCE POINT, WHAT IS THE DISTANCE TO THE ENDPOINT OF D1 .
( INCLUDE + OR - SIGN ) : -1


ASSUME B1  IS PARALLEL TO THE X-AXIS.
LET THE POINT WHERE B1  ROTATES ABOUT BE 0,0.

FROM THIS REFERENCE POINT, WHAT IS THE DISTANCE TO THE ENDPOINT OF V1 .
( INCLUDE + OR - SIGN ) : 2

INSERT NAME OF FILE FOR STORAGE : SPRING.RUN


*****  YOU ARE NOW LEAVING BILDBG.  *****
```

Figure 32 shows a simplified Lagrangian bond graph for the spring-mass-damper system created by BILDBG. The actual bond graph and the data base results from BILDBG can be found in Appendix D for this system.

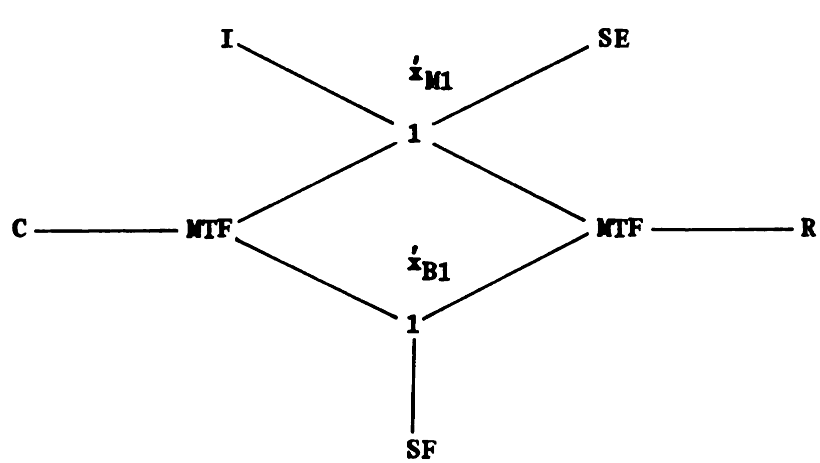


Figure 32. Bond Graph for Spring-mass-damper System

Note. MTF nodes are sets of elements.

## 4.2 Mass-spring Pendulum

A nonlinear mass-spring pendulum is shown in Figure 33. This example demonstrates the internal procedure of how BILDBG creates the nonlinear bond graph of this system. Interactively, the following information is obtained by BILDBG from DESIGN, as illustrated in example 1 of this chapter:

Motion Points:

| | |
|---|---|
| MPNAM(1) = M1 | MOTTYP(1) = 3 |
| MPNAM(2) = G1 | MOTTYP(2) = 0 |
| MPNAM(3) = P1 | MOTTYP(3) = 0 |

Connectors:

| | | |
|---|---|---|
| CNNAM(1) = S1 | CNTYP(1) = 1 | IEPTYP(1) = 2 |
| CNNAM(2) = F1 | CNTYP(2) = 4 | IEPTYP(2) = 2 |
| CNNAM(3) = X1 | CNTYP(3) = 1 | IEPTYP(3) = 0 |

Topology:

| Motion Point | Attached Connectors |
|---|---|
| M1 | S1, F1 |
| G1 | X1 |
| P1 | S1, X1 |

| Connector | Attached to Motion Point(s) |
|---|---|
| S1 | M1, P1 |
| F1 | M1 |
| X1 | P1, G1 |

Local Geometry:

On M1: S1 is located at the center (0,0).
F1 is located at the center (0,0).

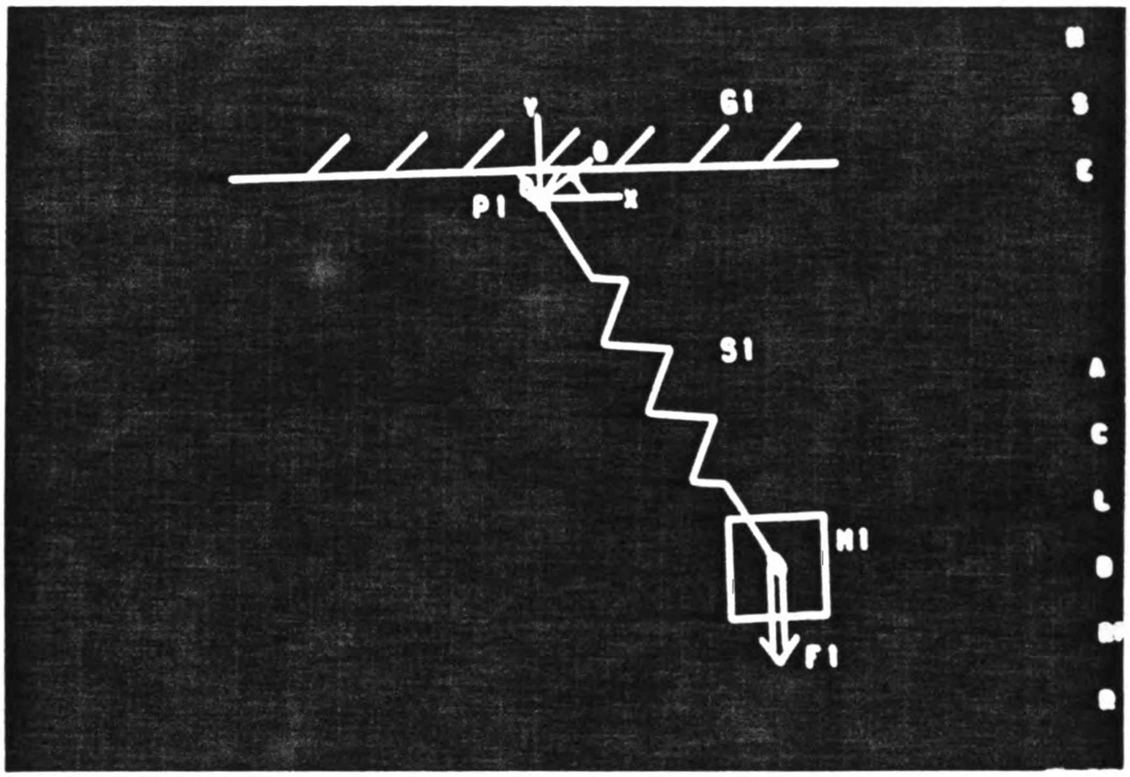


Figure 33. Mass-spring Pendulum

The first step in building the bond graph is to create 1-Junctions for each allowable motion for each motion point. For motion point 1 (mass - M1), the number of allowable motions is given by NMOT( MOTTYP(1) ), which is equal to two (i.e. MOTTYP(1) = 3, this means x and y motion only, which is two allowable motions). Thus, two 1-Junctions are created for motion point 1 as in Figure 34. Motion points 2 and 3, ground - G1 and pin - P1 respectively, have motion types of zero. Thus, no 1-Junctions are required for their motions.

$\dot{x}_{M1}$

1

$\dot{y}_{M1}$

1

Figure 34. Bond Graph for the Motion Points

The next step is to append Inertia-elements to all mass motion point 1-Junctions as shown in Figure 35.

Next, each connector is studied individually. The first connector is a spring (S1). The connector type for S1 is one (i.e. CNTYP(1) = 1), this corresponds to a C-element. Thus, the connector type, 0-Junction and the bond between are now added to the bond graph. See Figure 36.


Figure 35. Inertia-elements Added to the Bond Graph

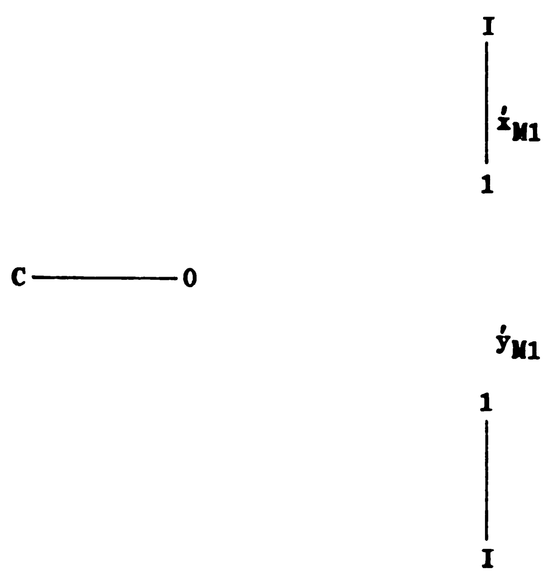


Figure 36. Connector S1 Added to the Bond Graph

The end point type of S1 is noted to be two (i.e. IEPTYP(1) = 2). Referring to Table 2, this end point type corresponds to Figure 23. Next, from the topology, we note that the end points of S1 are attached to M1 and P1. P1 has motion type 0, thus, no bonds or nodes are required for the end point attached to motion point P1. M1 has motion type 3, this corresponds to Figure 18 (see Table 1). Combining this information about the end points of S1, the bond graph is now represented in Figure 37.

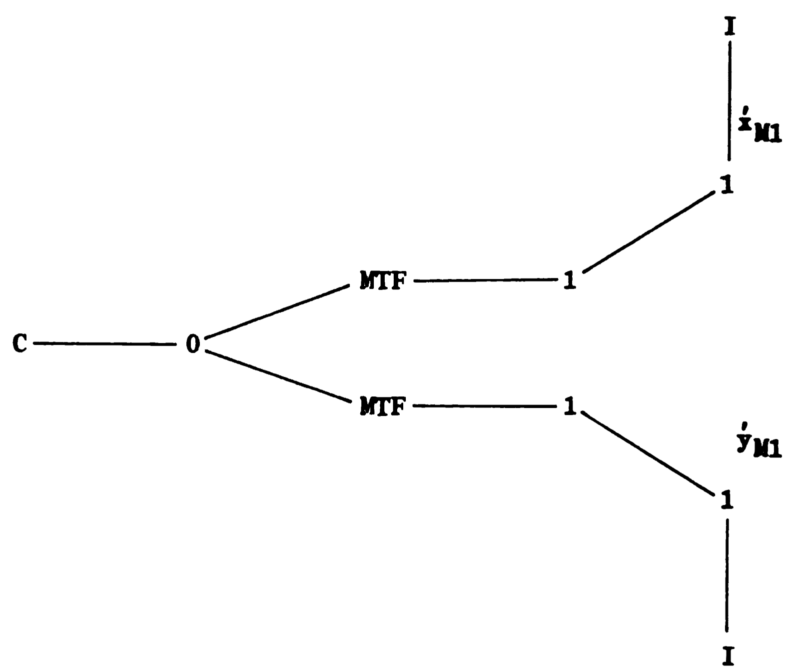


Figure 37. Nodes and Bonds Between S1 and M1 Added

The second connector is a force (F1). F1 has connector type 4, which corresponds to a SE-element. The remaining portion of the bond graph for F1 is developed in the same way as S1 was, described above. The resultant bond graph is shown in Figure 38.

Finally, the remaining connector, rigid connector X1, has an end point type of 0. This means no nodes or bonds are required for connector X1. Thus, the bond graph developed by BILDBG for the mass-spring pendulum of Figure 33 is depicted in Figure 38. The data base results for this system can be found in Appendix E.

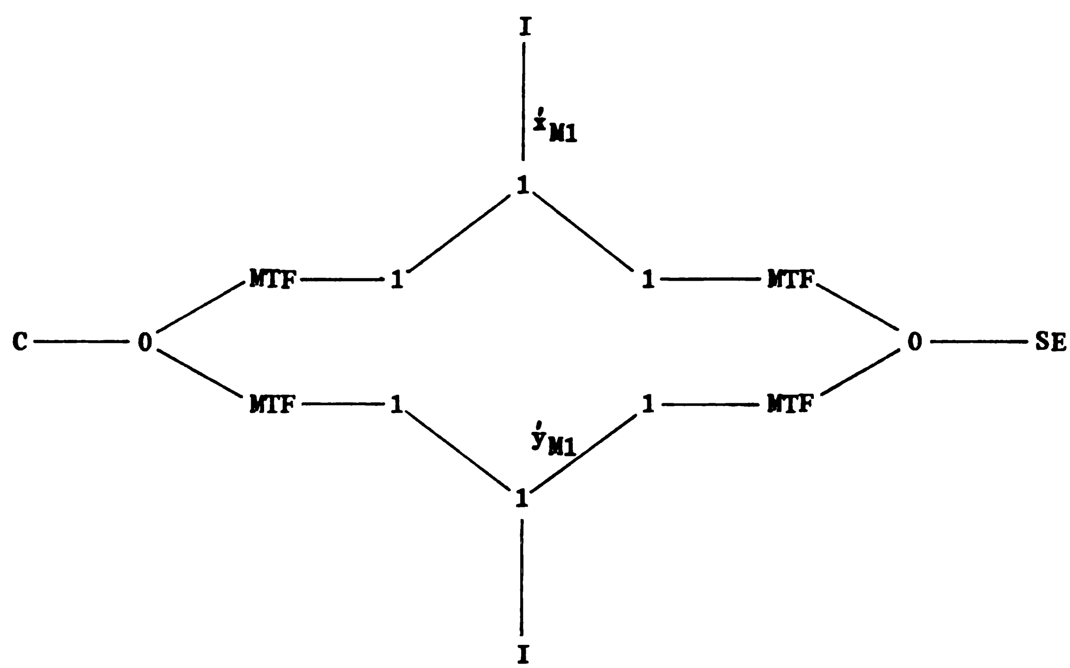


Figure 38. The Completed Bond Graph From BILDBG

4.3 Vehicle Suspension

Consider the planar vehicle suspension of Figure 39. The figure depicts a flatbed trailer traveling along a surface at a constant velocity. The trailer has a mass and a moment of inertia about its center. A load is placed upon the trailer bed.

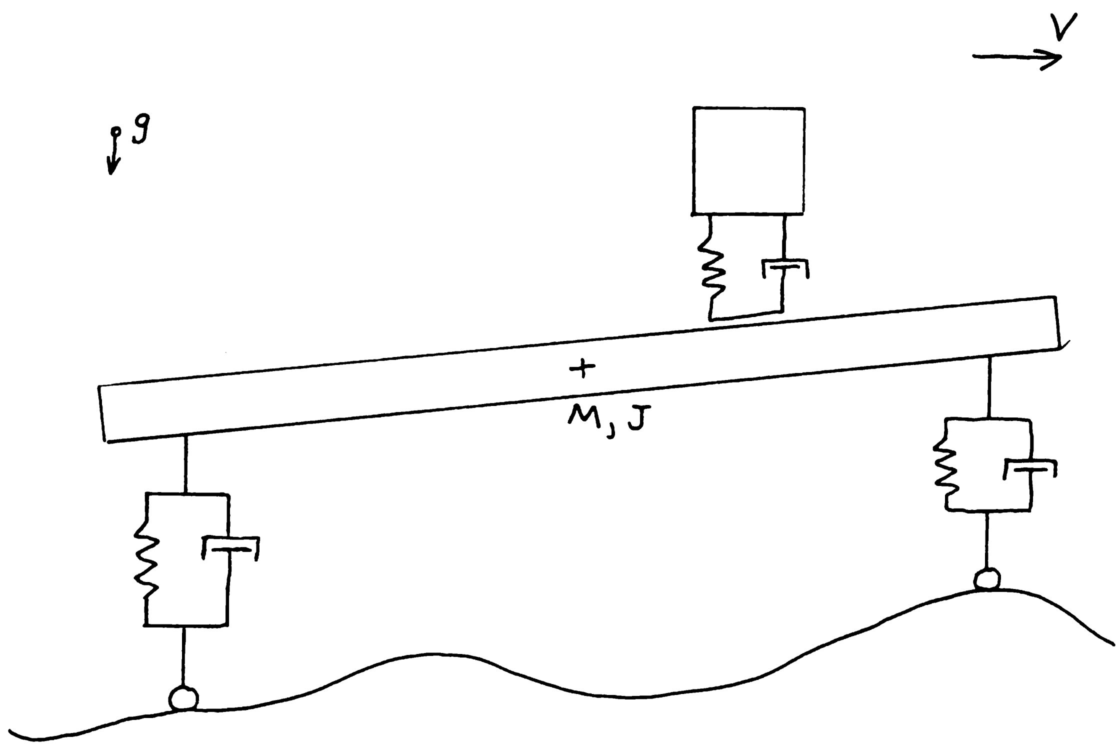


Figure 39. Planar Vehicle Suspension

The flatbed trailer system is drawn on the PS300 as shown in Figure 40 using DESIGN. BILDBG is used to create a nonlinear bond graph for this system. A Lagrangian bond graph for this system is shown in Figure 41. The data base results from BILDBG can be found in Appendix F.

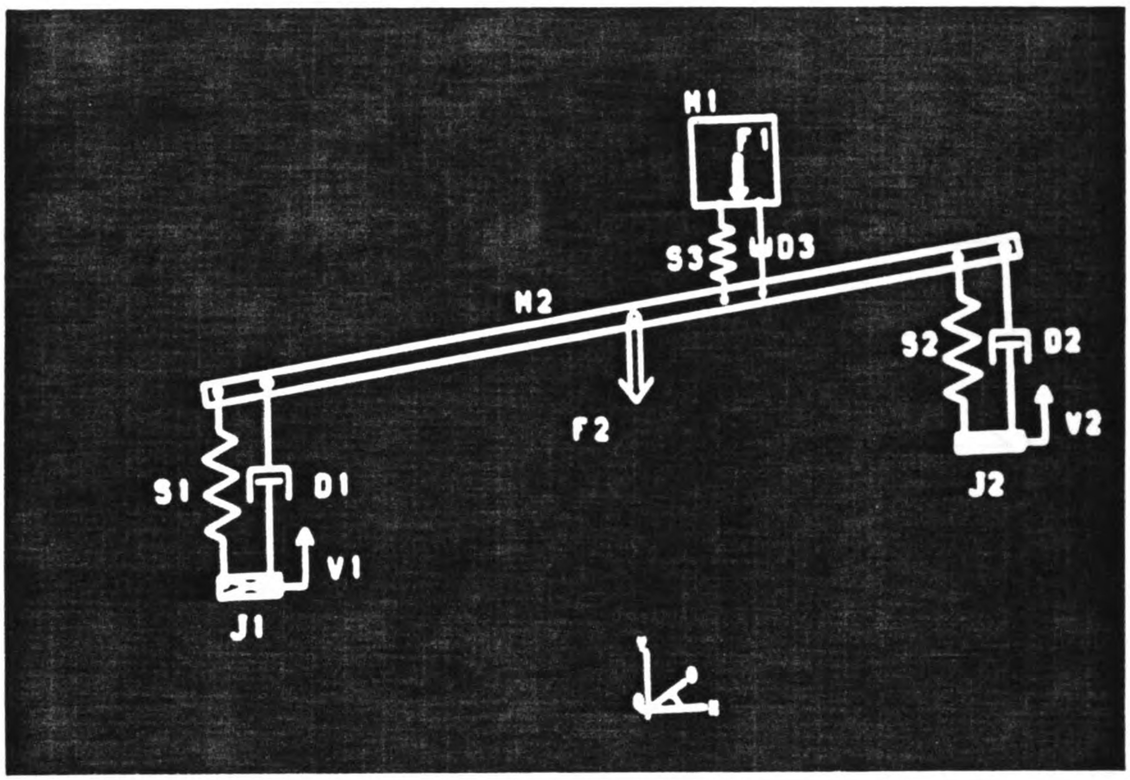


Figure 40. The Vehicle Suspension Drawn By DESIGN

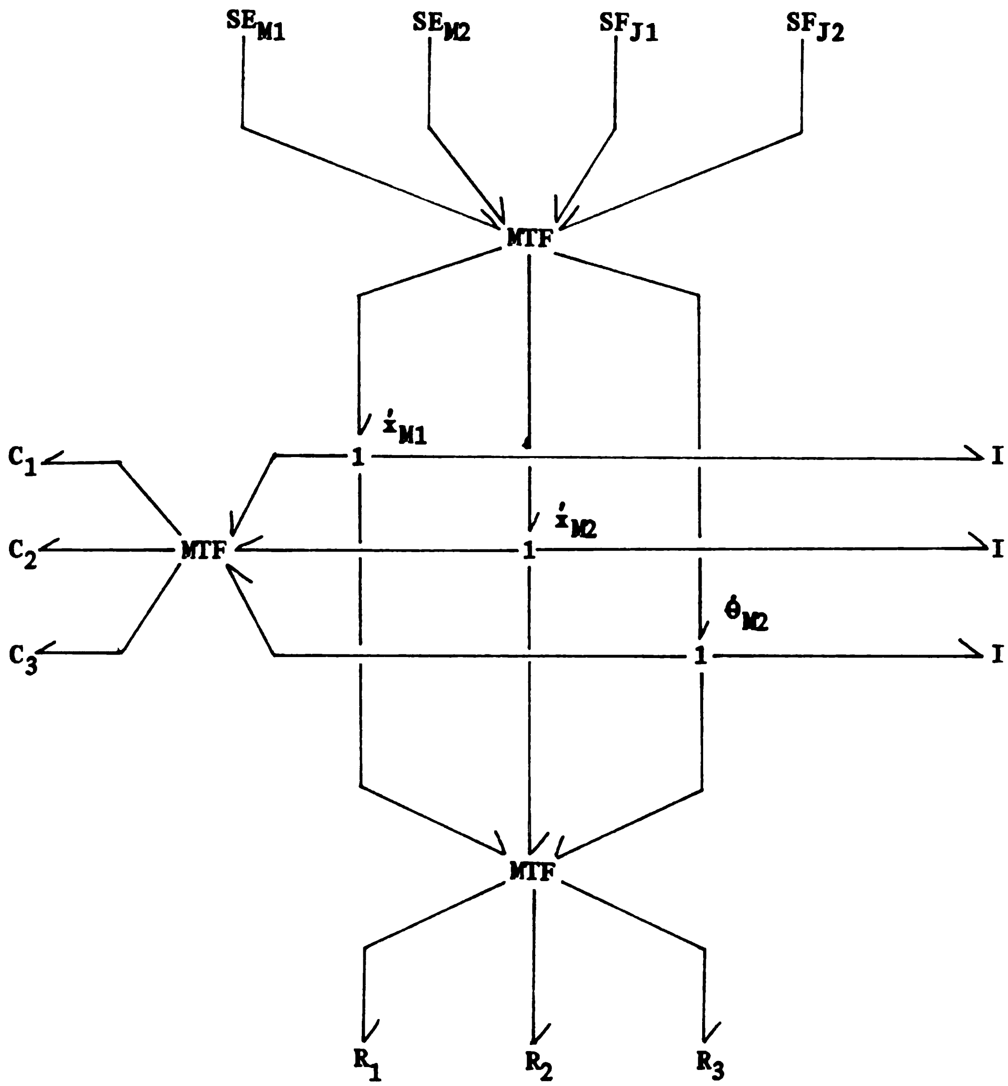


Figure 41. Lagrangian Bond Graph for the Vehicle Suspension

# Chapter 5

# SUMMARY AND CONCLUSIONS

A five step process to designing a mechanical system in a computer-aided design environment was described in this thesis: namely,

1. Draw the design on the computer using an interactive program.
2. Model the system internally in some pre-determined way.
3. Calculate the dynamic response of the system.
4. Animate the graphic display according to the dynamic response.
5. Change desired values and repeat the process until the design objectives are satisfied.

A first pass at accomplishing the first two steps was made. A graphic design program was described that interactively draws a mechanical system on an Evans and Sutherland PS300 computer. A modeling program was also described that mathematically models the completed graphic design by a nonlinear bond graph.

Future work will be required to calculate the dynamic response by a bond graph processor. The dynamic response could then be used to animate the system, in order to complete the design process.

# LIST OF REFERENCES

## LIST OF REFERENCES


1. Fallahi, B., 'A New Approach To Planar Analysis of Mechanisms,' Thesis for Doctor of Philosophy, Purdue University, August 1982.

2. Chace, M. A. and Angell, J. C., 'User's Guide to DRAM,' Department of Mechanical Engineering, University of Michigan, Ann Arbor, 1973.

3. Andrews, G. C. and Kesavan, H. K., 'The Vector-Network Model: A New Approach to Vector Dynamics,' Mechanisms and Machine Theory, Volume 10, 1975, pp 57-75.

4. Sheth, P. N. and Uicker, Jr., J. J., 'IMP (Integrated Mechanisms Program), A Computer Aided Design Analysis System for Mechanisms and Linkages,' Journal of Engineering Industry, Trans. ASME, Volume 94, Series B, No. 2, May 1972, pp. 454-464

5. Orlandea, N., Chace, M. A. and Calahan, D. A., 'A Sparsity-Oriented Approach to the Dynamic Analysis and Design of Mechanical System - Part 1,' ASME paper no. 76-DET-19, 1976.

6. Karnopp, D. C. and Rosenberg, R. C., SYSTEM DYNAMICS: A UNIFIED APPROACH, Wiley, New York, 1975.

7. Rosenberg, R. C., 'Multiport Models in Mechanics,' Journal of DSMS, Trans. ASME, Series G, No. 3, Sept. 1972, pp. 206-212.

8. Karnopp, D. C., 'Alternative Bond Graph Causal Patterns and Equation Formulations for Dynamic Systems,' Journal of DSMC, Trans. ASME, Volume 105, June 1983, pp. 58-63.

9. Rosenberg, R. C., 'ENPORT-5 User's Manual,' A. H. Case Center, College of Engineering, Michigan State University, 1981.

10. Grabowiecki, K., Zalewski, Z. and Zgorzelski, M., 'General Machine Dynamics Simulation System With Finite Elements Capabilities,' Bumar Union, Warsaw, Poland.

11. Stepniewski, W. and Grabowiecki, K., 'A Bond Graph Approach to the Automated Generation of Equations of Motion of Lumped Parameter Systems,' North-Holland Publishing Company, 1978.

12. Granda, J. J., 'Computer Aided Modeling Program (CAMP),' Dissertation for Doctor of Philosophy, University of California Davis, 1983.

13. Rosenberg, R. C. and Zhang, S. C., 'Modelling of Nonlinear Mechanics Problems Using Lagrangian Bondgraph,' Michigan State University, 1982.

14. Karnopp, D. C., 'Lagrange's Equations for Complex Bond Graph Systems,' Journal of DSMC, Trans. ASME, Series G, No. 1, Dec. 1977, pp. 300-306.

# APPENDICES

# APPENDIX A

## SOURCE CODE FOR DESIGN

# APPENDIX A

## SOURCE CODE FOR DESIGN

Subroutines in this appendix appear in the following sequence:

| | | |
|---|---|---|
| 1. DESIGN | 9. DRAWNG | 17. CONECT |
| 2. CMINIT | 10. INPUT | 18. DISCON |
| 3. MENU | 11. HLIGHT | 19. REINIT |
| 4. MASTER | 12. LOCKIN | 20. DISFIL |
| 5. CREATE | 13. UNLOCK | 21. GETITM |
| 6. RESTOR | 14. GETDAT | 22. ADITEM |
| 7. SAVE | 15. RESET | 23. LABELS |
| 8. MODIFY | 16. INCREM | 24. POSLBL |

Common files in this appendix appear in the following sequence:

| | | |
|---|---|---|
| 1. COMFIL | 2. COMBG | 3. SYBGBK |

PS300 files in this appendix appear in the following sequence:

| | | |
|---|---|---|
| 1. MENUS | 3. OBJECTS | 5. OUTPUT.NETWORK |
| 2. VIEW.OBJECT | 4. DIAL.NETWORK | |

```
      PROGRAM DESIGN
C                                                                   C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                   C
CDCCCCC                 PROGRAM   DESCRIPTION                   CCCCC
C                                                                   C
C--- THIS PROGRAM IS USED TO DRAW A 2-DIMENSIONAL MECHANICAL        C
C    MECHANISM USING THE STANDARD ELEMENTS.  (eg. SPRINGS,          C
C    DAMPERS.)                                                      C
C                                                                   C
C--- TO BE RUN ON THE EVANS AND SUTHERLAND PS300.                   C
C                                                                   C
C--- PROGRAMMER: JOHN D. REID          DATE: FALL 1983              C
C                                                                   C
C--- SUBROUTINES CALLED:   CMINIT   MENU   MASTER                   C
C                                                                   C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                   C
CVCCCCCCCC             VARIABLE IDENTIFICATION                CCCCCCCC
C                                                                   C
 INSERT COMFIL
C                                                                   C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                   C
CICCCCCCCC             INITIALIZATION BLOCK                 BLOCK OCCC
C                                                                   C
C--- INITIALIZE DATA IN COMMON BLOCK
C
      CALL CMINIT
C
C--- INITIALIZE GRAPHICS
C
      CALL PSGRAF
      CALL PSINIT('ALL')
C                                                                   C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                   C
CPCCCCCCCC             PROCESS BLOCK                        BLOCK OCCC
C                                                                   C
C--- SUPPRESS WARNINGS.
C
      PRINT *,'SEND FALSE TO <1>WARNING;'
      CALL PSTERM
C
C--- SET UP PS300 VIEWS, STRUCTURES AND NETWORKS.
C
      CALL MENU
C
C--- TALK TO USER.
C
      WRITE(*,900)
      WRITE(*,910)
      WRITE(*,920)
C
C--- CALL MASTER SUBROUTINE THAT RUNS THE SHOW
```

```
C
      CALL MASTER
C
C--- EXIT PROGRAM.
C
      WRITE(*,930)
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CFCCCCCCCC            FORMAT STATEMENTS                      BLOCK 9000
C                                                                     C
900   FORMAT(///,2(/,6X,50('*')),2(/,6X,'**',46X,'**'))
910   FORMAT(6X,'**',10X,'EVANS AND SUTHERLAND PS300',10X,'**',/,
     + 6X,'**',46X,'**',/,6X,'**',12X,'GRAPHIC DESIGN PROGRAM',
     + 12X,'**',2(/,6X,'**',46X,'**'))
920   FORMAT(6X,'**',17X,'DEVELOPED BY',17X,'**',/,
     +   6X,'**',46X,'**',/,
     +   6X,'**',18X,'JOHN REID',19X,'**',/,
     +   6X,'**',46X,'**',/,
     +   6X,'**',18X,'FALL 1983',19X,'**',
     +   2(/,6X,'**',46X,'**'),
     +   2(/,6X,50('*')))
930   FORMAT(////,'*****  YOU ARE NOW LEAVING',
     +      ' DESIGN.  *****',////)
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
      CALL EXIT
      END
      SUBROUTINE CMINIT
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CDCCCCC               SUBROUTINE DESCRIPTION                      CCCCC
C                                                                     C
C--- THIS SUBROUTINE INITIALIZES THE COMMON BLOCK COMFIL.             C
C                                                                     C
C--- PROGRAMMER: JOHN D. REID           DATE: FALL 1983               C
C                                                                     C
C--- CALLED FROM:      DESIGN     REINIT                              C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CVCCCCCCCC            VARIABLE IDENTIFICATION                  CCCCCCCC
C                                                                     C
 INSERT COMFIL
C                                                                     C
C  FOR A DESCRIPTION OF THE VARIABLES IN THIS PROGRAM SEE             C
C  THE COMMON FILE COMFIL.                                            C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CPCCCCCCCC            PROCESS BLOCK                          BLOCK 0CCC
C                                                                     C
      NUMG=0
```

```
      NUMB=0
      NUMS=0
      NUMD=0
      NUMM=0
      NUMJ=0
      NUMF=0
      NUMP=0
      NUMV=0
      NUML=0
      NUMR=0
      NITEM=0
      NDEL=0
      DO 100 I=1,40
        TRANSX(I)=0
        TRANSY(I)=0
        ROTZ(I)=0
        SCLX(I)=1
        SCLY(I)=1
        SCL(I)=1
        LBLPOS(I,1)=0
        LBLPOS(I,2)=0
100   CONTINUE
      LOCKED=.TRUE.
      FILE=.FALSE.
      OLDFIL=.FALSE.
      NWITEM=.FALSE.
C
C--- ORIGIN DATA
C
      ITMLST(39)='ORGIN'
      SCL(39)=.75
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
      RETURN
      END
      SUBROUTINE MENU
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
CDCCCCC              SUBROUTINE DESCRIPTION                       CCCCC
C                                                                      C
C--- THIS SUBROUTINE SETS UP THE REQUIRED EVANS AND SUTHERLAND         C
C    VIEWS, STRUCTURES AND NETWORKS.                                   C
C                                                                      C
C--- PROGRAMMER: JOHN D. REID          DATE: FALL 1983                 C
C                                                                      C
C--- CALLED FROM DESIGN.                                               C
C                                                                      C
C--- FILES ACCESSED:   MENUS   OBJECTS   DIAL.NETWORK                  C
C                      VIEW.OBJECT   OUTPUT.NETWORK                    C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
CPCCCCCCCCC           PROCESS BLOCK                          BLOCK 0CCC
```

```
C                                                                    C
      CALL PSGRAF
C
C--- SET UP MASTER, MODIFY AND DRAWING MENU
C
      CALL PSFILE('ID60>SU08>ES.DIR>LIB.DIR>MENUS')
C
C--- DEFINE THE INDIVIDUAL OBJECTS (SPRING,BAR,ETC.)
C
      CALL PSFILE('ID60>SU08>ES.DIR>LIB.DIR>OBJECTS')
C
C--- SET UP THE TRANS.,ROT.,SCL. NETWORK.
C
      CALL PSFILE('ID60>SU08>ES.DIR>LIB.DIR>DIAL.NETWORK')
C
C--- SET UP THE MAIN VIEW (VIEW)
C
      CALL PSFILE('ID60>SU08>ES.DIR>LIB.DIR>VIEW.OBJECT')
C
C--- CREATE AN OUTPUT NETWORK
C
      CALL PSFILE('ID60>SU08>ES.DIR>LIB.DIR>OUTPUT.NETWORK')
      PRINT *,'DISPLAY VIEW;'
      CALL PSTERM
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
      RETURN
      END
```

```
      SUBROUTINE MASTER
C                                                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                  C
CDCCCCC              SUBROUTINE DESCRIPTION                    CCCCC
C                                                                  C
C--- THIS SUBROUTINE CONTROLS THE MASTER MENU.                      C
C                                                                  C
C--- PROGRAMMER: JOHN D. REID          DATE: FALL 1983              C
C                                                                  C
C--- CALLED FROM DESGIN.                                            C
C                                                                  C
C--- SUBROUTINES CALLED:   HLIGHT    INPUT    SAVE                  C
C                          CREATE    RESTOR   MODIFY                C
C                                                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                  C
CVCCCCCCCCC          VARIABLE IDENTIFICATION                CCCCCCCC
C                                                                  C
 INSERT COMFIL
C                                                                  C
      HLITE='MST'
C                                                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                  C
CPCCCCCCCCC          PROCESS BLOCK                        BLOCK 0CCC
C                                                                  C
C--- HIGHLIGHT THE MENU THAT IS UNDER CONTROL (MASTER).
C
      CALL HLIGHT
C
C--- READ OPTION
C
100   CALL INPUT
C
C--- CREATE NEW FILE
C
      IF(WORD.EQ.'C') THEN
        CALL CREATE
C
C--- RESTORE OLD FILE
C
      ELSEIF(WORD.EQ.'R') THEN
        CALL RESTOR
C
C--- MODIFY CURRENT FILE
C
      ELSEIF(WORD.EQ.'M') THEN
        CALL MODIFY
C
C--- SAVE CURRENT FILE
C
      ELSEIF(WORD.EQ.'S') THEN
        CALL SAVE
C
```

```
C--- EXIT PROGRAM
C
      ELSEIF(WORD.EQ.'E') THEN
        RETURN
C
C--- BAD OPTION
C
      ELSE
        WRITE(*,910)
      ENDIF
      GOTO 100
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
CFCCCCCCCC           FORMAT STATEMENTS                       BLOCK 9000
C                                                                      C
910   FORMAT(/,'BAD OPTION --- TRY AGAIN.',/)
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
      END
      SUBROUTINE CREATE
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
CDCCCCC              SUBROUTINE DESCRIPTION                       CCCCC
C                                                                      C
C--- THIS SUBROUTINE CREATES A NEW FILE.                               C
C                                                                      C
C--- PROGRAMMER: JOHN D. REID           DATE: FALL 1983                C
C                                                                      C
C--- CALLED FROM:      MASTER      SAVE                                C
C                                                                      C
C--- SUBROUTINES CALLED:   REINIT   DISFIL                             C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
CVCCCCCCCC           VARIABLE IDENTIFICATION                   CCCCCCCC
C                                                                      C
 INSERT COMFIL
C                                                                      C
      EXTERNAL NCHARS
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
CPCCCCCCCC           PROCESS BLOCK                           BLOCK 0CCC
C                                                                      C
C--- IF FILE IS TRUE THEN THERE IS ALREADY A
C    FILE UNDER CONSIDERATION. MUST RE-INITIALIZE.
C
      IF(FILE) THEN
100     WRITE(*,900)
        READ(*,910) IANS
        IF(IANS.EQ.'Y') THEN
          CALL REINIT
```

```
        ELSEIF(IANS.EQ.'N') THEN
          RETURN
        ELSE
          GOTO 100
        ENDIF
      ENDIF
C
C--- INSERT NEW FILENAME AND CHECK TO SEE IF IT
C    ALREADY EXISTS.
C
200   WRITE(*,920)
      READ(*,930) FNAME
      CALL DISFIL
      OPEN(UNIT=12,FILE=FNAME,STATUS='NEW',ERR=800)
      CLOSE(UNIT=12)
      OLDFIL=.FALSE.
      FILE=.TRUE.
      GOTO 999
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
CERCCCCCCC            ERROR HANDLING                         BLOCK 0800
C                                                                      C
800   CALL DBLANK(FNAME)
      WRITE(*,940) FNAME(1:NCHARS(FNAME))
      GOTO 200
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
CFCCCCCCCC            FORMAT STATEMENTS                      BLOCK 9000
C                                                                      C
900   FORMAT('FINISHED WITH CURRENT FILE? (Y OR N) : ', )
910   FORMAT(A2)
920   FORMAT('INPUT NAME OF NEW FILE : ', )
930   FORMAT(A12)
940   FORMAT(A,' ALREADY EXISTS. TRY AGAIN.',/)
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
999   RETURN
      END
      SUBROUTINE RESTOR
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
CDCCCCC               SUBROUTINE DESCRIPTION                      CCCCC
C                                                                      C
C--- THIS SUBROUTINE RESTORES GRAPHICAL DATA FROM THE                  C
C    SPECIFIED FILE.                                                   C
C                                                                      C
C--- PROGRAMMER: JOHN D. REID           DATE: FALL 1983                C
C                                                                      C
C--- CALLED FROM MASTER.                                               C
C                                                                      C
C--- SUBROUTINES CALLED:     REINIT     DISFIL     ADITEM              C
```

```
C                             RESET       CONECT      INCREM          C
C                             DISCON                                  C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CVCCCCCCCC              VARIABLE IDENTIFICATION                CCCCCCCC
C                                                                     C
 INSERT COMFIL
C                                                                     C
      EXTERNAL NCHARS
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CPCCCCCCCC              PROCESS BLOCK                      BLOCK 0CCC
C                                                                     C
C--- IF FILE IS TRUE THEN THERE IS ALREADY A
C    FILE UNDER CONSIDERATION. MUST RE-INITIALIZE.
C
      IF(FILE) THEN
100     WRITE(*,900)
        READ(*,910) IANS
        IF(IANS.EQ.'Y') THEN
          CALL REINIT
        ELSEIF(IANS.EQ.'N') THEN
          RETURN
        ELSE
          GOTO 100
        ENDIF
      ENDIF
C
C--- INPUT OLD FILENAME.
C
      WRITE(*,920)
      READ(*,930) FNAME
      CALL DISFIL
C
C--- OPEN OLD FILENAME
C
      OPEN(UNIT=12,FILE=FNAME,STATUS='OLD',ERR=800)
C
C--- READ IN NUMBER OF ITEMS
C
      READ(12,*) NITEM
C
C--- READ IN THE NITEM'S AND ADD THEM TO THE GRAPHICS.
C
      DO 200 NCITEM=1,NITEM
        READ(12,940) ITMLST(NCITEM)
        READ(12,950) ITMTYP(NCITEM)
        READ(12,*) TRANSX(NCITEM),TRANSY(NCITEM),ROTZ(NCITEM),
     +             SCL(NCITEM),SCLX(NCITEM),SCLY(NCITEM)
        READ(12,955) ITMLBL(NCITEM),LBLPOS(NCITEM,1),
     +                 LBLPOS(NCITEM,2)
        CITEM=ITMLST(NCITEM)
        ITEM=ITMTYP(NCITEM)
```

```
        CALL ADITEM
        CALL PSGRAF
C
C--- ADD LABEL
C
          WRITE(*,957) ITMLBL(NCITEM),LBLPOS(NCITEM,1),
     +                 LBLPOS(NCITEM,2),ITMLBL(NCITEM)
          WRITE(*,958) ITMLBL(NCITEM),CITEM
C
C--- HANDLE SCALING IN X AND Y DIRECTION DIFFERENTLY.
C
        IF(SCLX(NCITEM).NE.1.0) THEN
          WRITE(*,960) CITEM
          WRITE(*,961) SCLX(NCITEM)
          WRITE(*,962)
        ENDIF
        IF(SCLY(NCITEM).NE.1.0) THEN
          WRITE(*,964) CITEM
          WRITE(*,965) SCLY(NCITEM)
          WRITE(*,966)
        ENDIF
C
C--- DISCONNECT SCALING DIALS. THE DIALS ARE
C    DISCONNECTED DOWN HERE BECAUSE THERE MUST
C    BE A DELAY FOR THE ABOVE TO BE EFFECTIVE.
C
        IF(SCLX(NCITEM).NE.1.0 .OR. SCLY(NCITEM).NE.1.0) THEN
          DO 150 IJK=1,400000
150       CONTINUE
          WRITE(*,963) CITEM
          WRITE(*,967) CITEM
          CALL RESET
        ENDIF
        CALL PSTERM
200   CONTINUE
C
C--- READ IN ORIGIN TRANSFORMATIONS
C
      READ(12,*) TRANSX(39),TRANSY(39),SCL(39)
      CITEM='ORGIN'
      CALL PSGRAF
        CALL CONECT
        WRITE(*,980) TRANSX(39)
        WRITE(*,985) TRANSY(39)
        WRITE(*,990) SCL(39)
        CALL INCREM
        DO 300 IJK=1,300000
300     CONTINUE
        CALL RESET
        CALL DISCON
      CALL PSTERM
C
C--- READ IN NUMBER OF EACH ITEM (ie. SPRINGS,MASSES...)
C
      READ(12,*) NUMG,NUMB,NUMS,NUMD
```

```
      READ(12,*) NUMP,NUMM,NUMR,NUMJ
      READ(12,*) NUMF,NUML,NUMV
      CLOSE(UNIT=12)
      OLDFIL=.TRUE.
      FILE=.TRUE.
      GOTO 999
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CERCCCCCCC            ERROR HANDLING                        BLOCK 0800
C                                                                     C
C--- FILE DOES NOT EXIST.                                             C
C                                                                     C
800   CALL DBLANK(FNAME)
      WRITE(*,995) FNAME(1:NCHARS(FNAME))
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CFCCCCCCCC            FORMAT STATEMENTS                     BLOCK 9000
C                                                                     C
900   FORMAT('FINISHED WITH CURRENT FILE? (Y OR N) : ', )
910   FORMAT(A2)
920   FORMAT('INPUT NAME OF OLD FILE : ', )
930   FORMAT(A12)
940   FORMAT(A5)
950   FORMAT(A6)
955   FORMAT(3X,A4,4X,F7.3,4X,F7.3)
957   FORMAT(A3,':=CHARACTERS ',F7.3,',',F7.3,' ''',A3,''';')
958   FORMAT('INCLUDE ',A3,' IN ',A5,';')
960   FORMAT('CONN SCXMAT<1>:<1>',A5,'.SCLX;')
961   FORMAT('SEND ',F7.3,' TO <2>SCALEX;')
962   FORMAT('SEND .00001 TO <1>SCALEX;')
963   FORMAT('DISC SCXMAT<1>:<1>',A5,'.SCLX;')
964   FORMAT('CONN SCYMAT<1>:<1>',A5,'.SCLY;')
965   FORMAT('SEND ',F7.3,' TO <2>SCALEY;')
966   FORMAT('SEND .00001 TO <1>SCALEY;')
967   FORMAT('DISC SCYMAT<1>:<1>',A5,'.SCLY;')
980   FORMAT('SEND ',F7.3,' TO <2>XVALUE;')
985   FORMAT('SEND ',F7.3,' TO <2>YVALUE;')
990   FORMAT('SEND ',F7.3,' TO <2>SCLVAL;')
995   FORMAT(A,' DOES NOT EXIST. CAN NOT RESTORE.',/)
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
999   RETURN
      END
      SUBROUTINE SAVE
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CDCCCCC               SUBROUTINE DESCRIPTION                     CCCCC
C                                                                     C
C--- THIS SUBROUTINE SAVES THE CURRENT WORKFILE.                      C
C                                                                     C
C--- PROGRAMMER: JOHN D. REID          DATE: FALL 1983                C
```

```
C                                                                    C
C--- CALLED FROM MASTER.                                             C
C                                                                    C
C--- SUBROUTINES CALLED:     CREATE     DISFIL                       C
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
CVCCCCCCCC           VARIABLE IDENTIFICATION                 CCCCCCCC
C                                                                    C
 INSERT COMFIL
C                                                                    C
      EXTERNAL NCHARS
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
CPCCCCCCCC           PROCESS BLOCK                         BLOCK OCCC
C                                                                    C
C--- CHECK TO SEE IF FILENAME HAS BEEN SPECIFIED.
C
      IF(.NOT.FILE) THEN
        WRITE(*,900)
        CALL CREATE
      ENDIF
C
C--- IF FILE TO BE SAVED IS OLD ASK USER IF IT IS
C    OK TO OVERWRITE.
C
      IF(OLDFIL) THEN
50      WRITE(*,905) FNAME(1:NCHARS(FNAME))
        READ(*,910) IANS
        IF(IANS.EQ.'OK') GOTO 60
        WRITE(*,920)
        READ(*,930) FNAME
        CALL DISFIL
        OPEN(UNIT=12,FILE=FNAME,STATUS='NEW',ERR=50)
        CLOSE(UNIT=12)
      ENDIF
60    OLDFIL=.TRUE.
C
C--- OPEN THE STORAGE FILE
C
      OPEN(UNIT=12,FILE=FNAME)
C
C--- LOAD THE NUMBER OF ITEMS THAT ARE TO BE SAVED.
C
      ITOTAL=NITEM-NDEL
      WRITE(12,*) ITOTAL
C
C--- LOAD EACH ITEM INTO THE FILE.
C
      DO 100 I=1,NITEM
C
C--- DO NOT ADD DELETED ITEMS.
C
        IF(ITMLST(I).EQ.'DELETE') GOTO 100
```

```
        WRITE(12,940) ITMLST(I)
        WRITE(12,950) ITMTYP(I)
        WRITE(12,960) TRANSX(I),TRANSY(I),ROTZ(I),SCL(I),
     +              SCLX(I),SCLY(I)
        WRITE(12,965) ITMLBL(I),LBLPOS(I,1),LBLPOS(I,2)
100   CONTINUE
C
C--- LOAD THE ORIGIN TRANSFORMATIONS
C
      WRITE(12,970) TRANSX(39),TRANSY(39),SCL(39)
C
C--- LOAD THE COUNTS OF EACH ITEM (ie. SPRINGS,MASSES...)
C
      WRITE(12,*) NUMG,NUMB,NUMS,NUMD
      WRITE(12,*) NUMP,NUMM,NUMR,NUMJ
      WRITE(12,*) NUMF,NUML,NUMV
      CLOSE(UNIT=12)
      WRITE(*,980) FNAME(1:NCHARS(FNAME))
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
CFCCCCCCCC             FORMAT STATEMENTS                     BLOCK 9000
C                                                                      C
900   FORMAT(/,'NO FILE IN MEMORY. PLEASE')
905   FORMAT(A,' ALREADY EXISTS.',/,
     +       'TYPE OK TO OVERWRITE          : ', )
910   FORMAT(A2)
920   FORMAT(/,'INPUT FILENAME TO BE STORED IN : ', )
930   FORMAT(A12)
940   FORMAT(A5)
950   FORMAT(A6)
960   FORMAT(3X,F7.3,4X,F7.3,4X,F8.3,4X,F7.3,4X,F7.3,4X,F7.3)
965   FORMAT(3X,A4,4X,F7.3,4X,F7.3)
970   FORMAT(3X,F7.3,4X,F7.3,4X,F7.3)
980   FORMAT(/,'FILE ',A,' HAS BEEN SAVED.',/)
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
      RETURN
      END
      SUBROUTINE MODIFY
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
CDCCCCC                SUBROUTINE DESCRIPTION                     CCCCC
C                                                                      C
C--- THIS SUBROUTINE CONTROLS THE MODIFY MENU.                         C
C                                                                      C
C--- PROGRAMMER: JOHN D. REID          DATE: FALL 1983                 C
C                                                                      C
C--- CALLED FROM MASTER.                                               C
C                                                                      C
C--- SUBROUTINES CALLED:      HLIGHT      INPUT      LOCKIN            C
C                             LABELS      DRAWNG     GETITM            C
C                             UNLOCK                                   C
```

```
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
CVCCCCCCCCC           VARIABLE IDENTIFICATION                CCCCCCCC
C                                                                    C
 INSERT COMFIL
C
      HLITE='MOD'
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
CPCCCCCCCCC           PROCESS BLOCK                        BLOCK 0CCC
C                                                                    C
C--- HIGHLIGHT THE MODIFY MENU
C
      CALL HLIGHT
C
C--- READ OPTION
C
100   CALL INPUT
C
C--- ADD TO CURRENT FILE
C
      IF(WORD.EQ.'A') THEN
C
C--- CHECK TO SEE IF PREVIOUS ITEM IS LOCKED IN.
C
        IF(.NOT.LOCKED) THEN
          CALL LOCKIN
          CALL LABELS
        ENDIF
        CALL DRAWNG
C
C--- CHANGE CURRENT FILE
C
      ELSEIF(WORD.EQ.'C') THEN
        IF(NITEM.EQ.0) GOTO 200
        IF(.NOT.LOCKED) THEN
          CALL LOCKIN
          CALL LABELS
        ENDIF
C
C--- GET THE ITEM (CITEM) AND THE NUMBER OF THE
C    ITEM (NCITEM) TO BE CHANGED.
C
        CALL GETITM
        IF(NCITEM.EQ.0) THEN
C
C---         PICKED ITEM DOES NOT EXIST IN MEMORY.
C
          WRITE(*,910)
          GOTO 100
        ENDIF
        CALL UNLOCK
C
```

```
C--- LOCKIN CURRENT ITEM (DISABLE TRANS.,ROT.,SCL)
C
      ELSEIF(WORD.EQ.'L') THEN
        IF(.NOT.LOCKED) THEN
          CALL LOCKIN
          CALL LABELS
        ENDIF
C
C--- DELETE AN ITEM FROM THE OBJECT
C
      ELSEIF(WORD.EQ.'D') THEN
        IF(.NOT.LOCKED) THEN
          CALL LOCKIN
          CALL LABELS
        ENDIF
C
C--- GET THE ITEM (CITEM) AND THE NUMBER OF THE
C    ITEM (NCITEM) TO BE DELETED.
C
        CALL GETITM
        IF(NCITEM.EQ.0) THEN
C
C---          PICKED ITEM DOES NOT EXIST IN MEMORY.
C
          WRITE(*,910)
        ELSEIF(NCITEM.EQ.39) THEN
C
C---          DO NOT DELETE ORIGIN
C
          WRITE(*,920)
        ELSE
          ITMLST(NCITEM)='DELETE'
          ITMLBL(NCITEM)='NON'
          CALL PSGRAF
            WRITE(*,9000) CITEM
          CALL PSTERM
C
C--- INCREMENT THE NUMBER OF DELETED ITEMS COUNT
C    AND STORE THE NAME OF THE DELETED ITEM.
C
          NDEL=NDEL+1
          ITMDEL(NDEL)=CITEM
        ENDIF
C
C--- REPLACE AN ITEM IN THE OBJECT WITH ANOTHER
C
      ELSEIF(WORD.EQ.'RP') THEN
        IF(.NOT.LOCKED) THEN
          CALL LOCKIN
        ENDIF
        WRITE(*,900)
C
C--- RETURN AND HLIGHT MASTER MENU.
C
      ELSEIF(WORD.EQ.'R') THEN
```

```
        IF(.NOT.LOCKED) THEN
          CALL LOCKIN
          CALL LABELS
        ENDIF
        HLITE='MST'
        CALL HLIGHT
        RETURN
C
C--- BAD OPTION
C
      ELSE
        WRITE(*,910)
      ENDIF
200   GOTO 100
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
CFCCCCCCCC           FORMAT STATEMENTS                      BLOCK 9000
C                                                                    C
900   FORMAT(//,'OPTION NOT AVAILABLE AT THIS TIME.',//)
910   FORMAT(//,'BAD OPTION --- TRY AGAIN.',//)
920   FORMAT(//,'CAN NOT DELETE ORIGIN.',//)
9000  FORMAT('DELETE ',A5,';')
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
      END
      SUBROUTINE DRAWNG
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
CDCCCCC              SUBROUTINE DESCRIPTION                     CCCCC
C                                                                    C
C--- THIS SUBROUTINE CONTROLS THE DRAWING MENU WHICH IS              C
C    USED TO ADD ITEMS TO THE DESIGN.                                C
C                                                                    C
C--- PROGRAMMER: JOHN D. REID          DATE: FALL 1983               C
C                                                                    C
C--- CALLED FROM MODIFY.                                             C
C                                                                    C
C--- SUBROUTINES CALLED:      HLIGHT      INPUT      ADITEM          C
C                             UNLOCK                                 C
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
CVCCCCCCCC           VARIABLE IDENTIFICATION                 CCCCCCCC
C                                                                    C
 INSERT COMFIL
C
      CHARACTER*1 NUM
      CHARACTER*2 NUM2
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
CPCCCCCCCC           PROCESS BLOCK                          BLOCK 0CCC
```

```
C                                                                          C
C--- HIGHLIGHT THE DRAWING MENU
C
      HLITE='DRW'
      CALL HLIGHT
C
C--- READ OPTION
C
100   CALL INPUT
C
C--- GROUND
C
      IF(WORD.EQ.'G') THEN
        ITEM='GROUND'
        NUMG=NUMG+1
        IF(NUMG.LT.10) THEN
          WRITE(NUM,'(I1)') NUMG
          CITEM='GRND'//NUM
        ELSE
          WRITE(NUM2,'(I2)') NUMG
          CITEM='GRN'//NUM2
        ENDIF
C
C--- BAR
C
      ELSEIF(WORD.EQ.'B') THEN
        ITEM='BAR'
        NUMB=NUMB+1
        IF(NUMB.LT.10) THEN
          WRITE(NUM,'(I1)') NUMB
          CITEM='BARR'//NUM
        ELSE
          WRITE(NUM2,'(I2)') NUMB
          CITEM='BAR'//NUM2
        ENDIF
C
C--- SPRING
C
      ELSEIF(WORD.EQ.'S') THEN
        ITEM='SPRING'
        NUMS=NUMS+1
        IF(NUMS.LT.10) THEN
          WRITE(NUM,'(I1)') NUMS
          CITEM='SPRG'//NUM
        ELSE
          WRITE(NUM2,'(I2)') NUMS
          CITEM='SPR'//NUM2
        ENDIF
C
C--- DAMPER
C
      ELSEIF(WORD.EQ.'D') THEN
        ITEM='DAMPER'
        NUMD=NUMD+1
        IF(NUMD.LT.10) THEN
```

```
          WRITE(NUM,'(I1)') NUMD
          CITEM='DAMP'//NUM
        ELSE
          WRITE(NUM2,'(I2)') NUMD
          CITEM='DAM'//NUM2
        ENDIF
C
C--- PIN
C
      ELSEIF(WORD.EQ.'P') THEN
        ITEM='PIN'
        NUMP=NUMP+1
        IF(NUMP.LT.10) THEN
          WRITE(NUM,'(I1)') NUMP
          CITEM='PINN'//NUM
        ELSE
          WRITE(NUM2,'(I2)') NUMP
          CITEM='PIN'//NUM2
        ENDIF
C
C--- MASS
C
      ELSEIF(WORD.EQ.'M') THEN
        ITEM='MASS'
        NUMM=NUMM+1
        IF(NUMM.LT.10) THEN
          WRITE(NUM,'(I1)') NUMM
          CITEM='MASS'//NUM
        ELSE
          WRITE(NUM2,'(I2)') NUMM
          CITEM='MAS'//NUM2
        ENDIF
C
C--- ROUGH
C
      ELSEIF(WORD.EQ.'R') THEN
        ITEM='ROUGH'
        NUMR=NUMR+1
        IF(NUMR.LT.10) THEN
          WRITE(NUM,'(I1)') NUMR
          CITEM='RUFF'//NUM
        ELSE
          WRITE(NUM2,'(I2)') NUMR
          CITEM='RUF'//NUM2
        ENDIF
C
C--- JOINT
C
      ELSEIF(WORD.EQ.'J') THEN
        ITEM='JOINT'
        NUMJ=NUMJ+1
        IF(NUMJ.LT.10) THEN
          WRITE(NUM,'(I1)') NUMJ
          CITEM='JONT'//NUM
        ELSE
```

```
          WRITE(NUM2,'(I2)') NUMJ
          CITEM='JON'//NUM2
        ENDIF
C
C--- FORCE
C
      ELSEIF(WORD.EQ.'F') THEN
        ITEM='FORCE'
        NUMF=NUMF+1
        IF(NUMF.LT.10) THEN
          WRITE(NUM,'(I1)') NUMF
          CITEM='FORC'//NUM
        ELSE
          WRITE(NUM2,'(I2)') NUMF
          CITEM='FOR'//NUM2
        ENDIF
C
C--- LEFT VELOCITY
C
      ELSEIF(WORD.EQ.'VL') THEN
        ITEM='VLEFT'
        NUML=NUML+1
        IF(NUML.LT.10) THEN
          WRITE(NUM,'(I1)') NUML
          CITEM='VLFT'//NUM
        ELSE
          WRITE(NUM2,'(I2)') NUML
          CITEM='VLF'//NUM2
        ENDIF
C
C--- RIGHT VELOCITY
C
      ELSEIF(WORD.EQ.'VR') THEN
        ITEM='VRIGHT'
        NUMV=NUMV+1
        IF(NUMV.LT.10) THEN
          WRITE(NUM,'(I1)') NUMV
          CITEM='VRGT'//NUM
        ELSE
          WRITE(NUM2,'(I2)') NUMV
          CITEM='VRG'//NUM2
        ENDIF
C
C--- EXIT
C
      ELSEIF(WORD.EQ.'E') THEN
        GOTO 799
C
C--- BAD OPTION
C
      ELSE
        WRITE(*,910)
        GOTO 100
      ENDIF
C
```

```
C--- STORE ITEM NAME, TYPE AND INCREMENT ITEM COUNT.
C
      NITEM=NITEM+1
      NCITEM=NITEM
      ITMLST(NITEM)=CITEM
      ITMTYP(NITEM)=ITEM
C
C--- ADD ITEM TO MAIN OBJECT AND ALLOW
C     FOR ROTATIONS,TRANLATIONS,SCALING.
C
      CALL ADITEM
      NWITEM=.TRUE.
      CALL UNLOCK
C
C--- HIGHLIGHT MODIFY MENU AND RETURN.
C
799   HLITE='MOD'
      CALL HLIGHT
C                                                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                  C
CFCCCCCCCC           FORMAT STATEMENTS                   BLOCK 9000
C                                                                  C
910   FORMAT(//,'BAD OPTION --- TRY AGAIN.',//)
C                                                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                  C
      RETURN
      END
```

```
      SUBROUTINE INPUT
C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CDCCCCC              SUBROUTINE DESCRIPTION                       CCCCC
C                                                                     C
C--- THIS SUBROUTINE GETS THE USERS INPUT                              C
C                                                                     C
C--- PROGRAMMER: JOHN D. REID          DATE: FALL 1983                C
C                                                                     C
C--- CALLED FROM:     MASTER     MODIFY     DRAWNG                    C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CVCCCCCCCC           VARIABLE IDENTIFICATION                   CCCCCCCC
C                                                                     C
 INSERT COMFIL
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CPCCCCCCCC           PROCESS BLOCK                           BLOCK 0CCC
C                                                                     C
C--- READ OPTION
C
      WORD='DUMMY'
      CALL PROMPT('INPUT DESIRED OPTION : ')
      READ(*,900) WORD
      WRITE(*,*) ' '
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CFCCCCCCCC           FORMAT STATEMENTS                       BLOCK 9000
C                                                                     C
900   FORMAT(A32)
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
      RETURN
      END
      SUBROUTINE HLIGHT
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CDCCCCC              SUBROUTINE DESCRIPTION                       CCCCC
C                                                                     C
C--- THIS SUBROUTINE DRAWS A BOX AROUND THE                            C
C    CURRENTLY ACTIVE MENU.                                           C
C                                                                     C
C--- PROGRAMMER: JOHN D. REID          DATE: FALL 1983                C
C                                                                     C
C--- CALLED FROM:     MASTER     MODIFY     DRAWNG                    C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
```

```
CVCCCCCCCC            VARIABLE IDENTIFICATION                CCCCCCCC
C                                                                   C
 INSERT COMFIL
C                                                                   C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                   C
CPCCCCCCCC            PROCESS BLOCK                        BLOCK 0CCC
C                                                                   C
C--- BEGIN STRUCTURE OF THE BOX.
C
      CALL PSGRAF
      PRINT *,'HLIGHT:=BEGINS'
      PRINT *,'  VEC ITEMIZED Z=0 N=5'
C
C--- CHECK TO SEE WHICH MENU IS THE ACTIVE ONE.
C
C--- MASTER MENU.
C
      IF(HLITE.EQ.'MST') THEN
        PRINT *,'P .5,.45 L 1.0,.45 L 1.0,1.0'
        PRINT *,'L .5,1.0 L .5,.45;'
C
C--- MODIFY MENU.
C
      ELSEIF(HLITE.EQ.'MOD') THEN
        PRINT *,'P .5,-.33 L 1.0,-.33 L 1.0,.28'
        PRINT *,'L .5,.28 L .5,-.33;'
C
C--- DRAWING MENU
C
      ELSEIF(HLITE.EQ.'DRW') THEN
        PRINT *,'P .46,-1.0 L 1.0,-1.0 L 1.0,-.42'
        PRINT *,'L .46,-.42 L .46,-1.0;'
      ENDIF
C
C--- COMPLETE STRUCTURE AND DISPLAY HLIGHT.
C
      PRINT *,'ENDS;'
      PRINT *,'DISPLAY HLIGHT;'
      CALL PSTERM
C                                                                   C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                   C
      RETURN
      END
      SUBROUTINE LOCKIN
C                                                                   C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                   C
CDCCCCC               SUBROUTINE DESCRIPTION                    CCCCC
C                                                                   C
C--- THIS SUBROUTINE DISCONNECTS TRANSLATIONS,                      C
C    ROTATIONS AND SCALING FROM THE SPECIFIED                       C
C    OBJECT (CITEM).                                                C
C                                                                   C
```

```
C--- PROGRAMMER: JOHN D. REID          DATE: FALL 1983           C
C                                                                C
C--- CALLED FROM:     MODIFY     POSLBL                          C
C                                                                C
C--- SUBROUTINES CALLED:     GETDAT     DISCON     RESET         C
C                                                                C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                C
CVCCCCCCCCC            VARIABLE IDENTIFICATION           CCCCCCCC
C                                                                C
 INSERT COMFIL
C                                                                C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                C
CPCCCCCCCCC            PROCESS BLOCK                    BLOCK 0CCC
C                                                                C
C--- STORE THE TRANS.,ROT.,AND SCALING FACTORS.
C
      CALL GETDAT
C
      CALL PSGRAF
C
C--- DISCONNECT MOVEMENT CAPIBILITIES.
C
      CALL DISCON
C
C--- RESET ACCUMULATORS ON TRANS., ROT. AND SCALING
C
      CALL RESET
      CALL PSTERM
      LOCKED=.TRUE.
C                                                                C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                C
      RETURN
      END
      SUBROUTINE UNLOCK
C                                                                C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                C
CDCCCCCC               SUBROUTINE DESCRIPTION               CCCCC
C                                                                C
C--- THIS SUBROUTINE CONNECTS TRANSLATIONS,                      C
C    ROTATIONS AND SCALING FOR THE SPECIFIED                     C
C    OBJECT (CITEM).                                             C
C                                                                C
C--- PROGRAMMER: JOHN D. REID          DATE: FALL 1983           C
C                                                                C
C--- CALLED FROM:     MODIFY     DRAWNG     POSLBL               C
C                                                                C
C--- SUBROUTINES CALLED:     CONECT     INCREM                   C
C                                                                C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                C
CVCCCCCCCCC            VARIABLE IDENTIFICATION           CCCCCCCC
```

```
C                                                                    C
 INSERT COMFIL
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
CPCCCCCCCC             PROCESS BLOCK                        BLOCK 0CCC
C                                                                    C
      CALL PSGRAF
C
C--- CONNECT ROT.,TRANS.,SCL. NETWORK.
C
      CALL CONECT
C
C--- INCREMENT TRANS.,ROT. AND SALING BY NEGLIGABLE AMOUNTS.
C    THIS PUTS THE OBJECT IN THE POSITION THE PS300 "THINKS"
C    IT IS IN.
C
      CALL INCREM
      CALL PSTERM
      LOCKED=.FALSE.
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
      RETURN
      END
      SUBROUTINE GETDAT
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
CDCCCCC                SUBROUTINE DESCRIPTION                    CCCCC
C                                                                    C
C--- THIS SUBROUTINE GETS THE TRANSLATIONS, ROTATION                 C
C    AND SCALING FACTORS OF THE CURRENT UNLOCKED OBJECT.             C
C                                                                    C
C--- PROGRAMMER: JOHN D. REID           DATE: FALL 1983              C
C                                                                    C
C--- CALLED FROM LOCKIN.                                             C
C                                                                    C
C--- SUBROUTINES CALLED:     INCREM                                  C
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
CVCCCCCCCC             VARIABLE IDENTIFICATION                CCCCCCCC
C                                                                    C
 INSERT COMFIL
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
CPCCCCCCCC             PROCESS BLOCK                        BLOCK 0CCC
C                                                                    C
C--- CONNECT FACTOR OUTPUTS TO THE REQUIRED NETWORK.
C
      CALL PSGRAF
      PRINT *,'CONN XVALUE<1>:<1>DATA;'
      PRINT *,'CONN YVALUE<1>:<2>DATA;'
```

```
      PRINT *,'CONN ZDIAL<2>:<3>DATA;'
      PRINT *,'CONN SCLVAL<2>:<4>DATA;'
      PRINT *,'CONN SCALEX<1>:<5>DATA;'
      PRINT *,'CONN SCALEY<1>:<6>DATA;'
C
C--- INCREMENT THE X AND Y TRANSLATIONS, Z ROTATION AND
C    THE SCALING FACTORS BY A NEGLIGABLE VALUE.
C
      CALL INCREM
C
C--- TRIGGER SYNCRANIZED DATA FOR OUTPUTS.
C
      PRINT *,'SEND 1 TO <7>DATA;'
C
C--- READ IN FACTORS.
C
      CALL PSTERM
      READ(*,*) TRANSX(NCITEM)
      READ(*,*) TRANSY(NCITEM)
      READ(*,*) ROTZ(NCITEM)
      READ(*,*) SCL(NCITEM)
      READ(*,*) SCLX(NCITEM)
      READ(*,*) SCLY(NCITEM)
      CALL PSGRAF
C
C--- DISCONNECT TRANSFORMATION DATA NETWORK.
C
      PRINT *,'DISC XVALUE<1>:<1>DATA;'
      PRINT *,'DISC YVALUE<1>:<2>DATA;'
      PRINT *,'DISC ZDIAL<2>:<3>DATA;'
      PRINT *,'DISC SCLVAL<2>:<4>DATA;'
      PRINT *,'DISC SCALEX<1>:<5>DATA;'
      PRINT *,'DISC SCALEY<1>:<6>DATA;'
      CALL PSTERM
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
      RETURN
      END
      SUBROUTINE RESET
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
CDCCCCC               SUBROUTINE DESCRIPTION                     CCCCC
C                                                                    C
C--- RESET ACCUMULATORS ON TRANS., ROT. AND SCALING                  C
C    FUNCTIONS ON THE PS300.                                         C
C                                                                    C
C--- PROGRAMMER: JOHN D. REID          DATE: FALL 1983               C
C                                                                    C
C--- CALLED FROM:     RESTOR     LOCKIN                              C
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
CPCCCCCCCC            PROCESS BLOCK                         BLOCK 0CCC
```

```
C                                                                    C
      PRINT *,'SEND 0 TO <2>ZDIAL;'
      PRINT *,'SEND 0 TO <2>XVALUE;'
      PRINT *,'SEND 0 TO <2>YVALUE;'
      PRINT *,'SEND 1 TO <2>SCLVAL;'
      PRINT *,'SEND 1 TO <2>SCALEX;'
      PRINT *,'SEND 1 TO <2>SCALEY;'
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
      RETURN
      END
      SUBROUTINE INCREM
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
CDCCCCC               SUBROUTINE DESCRIPTION                     CCCCC
C                                                                    C
C--- INCREMENT THE X AND Y TRANSLATIONS, Z ROTATION AND              C
C    THE SCALING FACTORS BY A NEGLIGABLE VALUE.                      C
C                                                                    C
C--- PROGRAMMER: JOHN D. REID           DATE: FALL 1983              C
C                                                                    C
C--- CALLED FROM:      RESTOR     UNLOCK     GETDAT                  C
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
CPCCCCCCCC             PROCESS BLOCK                         BLOCK OCCC
C                                                                    C
      PRINT *,'SEND .00001 TO <1>XVALUE;'
      PRINT *,'SEND .00001 TO <1>YVALUE;'
      PRINT *,'SEND .0000001 TO <1>ZDIAL;'
      PRINT *,'SEND .00001 TO <1>SCLVAL;'
      PRINT *,'SEND .00001 TO <1>SCALEX;'
      PRINT *,'SEND .00001 TO <1>SCALEY;'
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
      RETURN
      END
      SUBROUTINE CONECT
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
CDCCCCC               SUBROUTINE DESCRIPTION                     CCCCC
C                                                                    C
C--- THIS SUBROUTINE CONNECTS THE DIAL NETWORK                       C
C    TO THE CURRENT ITEM (CITEM).                                    C
C                                                                    C
C--- PROGRAMMER: JOHN D. REID           DATE: FALL 1983              C
C                                                                    C
C--- CALLED FROM:      RESTOR     UNLOCK                             C
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
```

```
CVCCCCCCCC             VARIABLE IDENTIFICATION                  CCCCCCCC
C                                                                      C
 INSERT COMFIL
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
CPCCCCCCCC             PROCESS BLOCK                          BLOCK 0CCC
C                                                                      C
      WRITE(*,900) CITEM
      WRITE(*,910) CITEM
      WRITE(*,920) CITEM
      WRITE(*,930) CITEM
      WRITE(*,940) CITEM
      WRITE(*,950) CITEM
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
CFCCCCCCCC             FORMAT STATEMENTS                      BLOCK 9000
C                                                                      C
900   FORMAT('CONN XVECT<1>:<1>',A5,'.TRANSX;')
910   FORMAT('CONN YVECT<1>:<1>',A5,'.TRANSY;')
920   FORMAT('CONN ZDIAL<1>:<1>',A5,'.ROTZ;')
930   FORMAT('CONN SCLVAL<1>:<1>',A5,'.SCL;')
940   FORMAT('CONN SCXMAT<1>:<1>',A5,'.SCLX;')
950   FORMAT('CONN SCYMAT<1>:<1>',A5,'.SCLY;')
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
      RETURN
      END
      SUBROUTINE DISCON
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
CDCCCCC                SUBROUTINE DESCRIPTION                      CCCCC
C                                                                      C
C--- THIS SUBROUTINE DISCONNECTS THE DIAL NETWORK                      C
C    FROM THE CURRENT ITEM (CITEM).                                    C
C                                                                      C
C--- PROGRAMMER: JOHN D. REID            DATE: FALL 1983               C
C                                                                      C
C--- CALLED FROM:     RESTOR     LOCKIN                                C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
CVCCCCCCCC             VARIABLE IDENTIFICATION                  CCCCCCCC
C                                                                      C
 INSERT COMFIL
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
CPCCCCCCCC             PROCESS BLOCK                          BLOCK 0CCC
C                                                                      C
      WRITE(*,900) CITEM
      WRITE(*,910) CITEM
```

```
      WRITE(*,920) CITEM
      WRITE(*,930) CITEM
      WRITE(*,940) CITEM
      WRITE(*,950) CITEM
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
CFCCCCCCCC            FORMAT STATEMENTS                      BLOCK 9000
C                                                                      C
900   FORMAT('DISC XVECT<1>:<1>',A5,'.TRANSX;')
910   FORMAT('DISC YVECT<1>:<1>',A5,'.TRANSY;')
920   FORMAT('DISC ZDIAL<1>:<1>',A5,'.ROTZ;')
930   FORMAT('DISC SCLVAL<1>:<1>',A5,'.SCL;')
940   FORMAT('DISC SCXMAT<1>:<1>',A5,'.SCLX;')
950   FORMAT('DISC SCYMAT<1>:<1>',A5,'.SCLY;')
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
      RETURN
      END
      SUBROUTINE REINIT
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
CDCCCCC               SUBROUTINE DESCRIPTION                       CCCCC
C                                                                      C
C--- THIS SUBROUTINE RE-INITIALIZES GRAPHICS                           C
C    AND REQUIRED VARIABLES.                                           C
C                                                                      C
C--- PROGRAMMER: JOHN D. REID          DATE: FALL 1983                 C
C                                                                      C
C--- CALLED FROM:     CREATE     RESTOR                                C
C                                                                      C
C--- SUBROUTINES CALLED:     CMINIT                                    C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
CVCCCCCCCC            VARIABLE IDENTIFICATION                   CCCCCCCC
C                                                                      C
 INSERT COMFIL
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
CPCCCCCCCC            PROCESS BLOCK                          BLOCK 0CCC
C                                                                      C
C--- REMOVE OBJECTS FROM PS300 MEMORY.
C
      CALL PSGRAF
      DO 100 I=1,NITEM
        WRITE(*,900) ITMLST(I)
        WRITE(*,910) ITMLBL(I)
100   CONTINUE
      CALL PSTERM
C
C--- RE-INIT VARIABLES.
```

```
C
      CALL CMINIT
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
CFCCCCCCCC            FORMAT STATEMENTS                        BLOCK 9000
C                                                                       C
900   FORMAT('DELETE ',A5,';')
910   FORMAT('DELETE ',A4,';')
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
      RETURN
      END
      SUBROUTINE DISFIL
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
CDCCCCC               SUBROUTINE DESCRIPTION                        CCCCC
C                                                                       C
C--- THIS SUBROUTINE DISPLAYS THE FILENAME ON THE PS300.                C
C                                                                       C
C--- PROGRAMMER: JOHN D. REID          DATE: FALL 1983                  C
C                                                                       C
C--- CALLED FROM:     CREATE     RESTOR     SAVE                        C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
CVCCCCCCCC            VARIABLE IDENTIFICATION                    CCCCCCCC
C                                                                       C
 INSERT COMFIL
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
CPCCCCCCCC            PROCESS BLOCK                            BLOCK OCCC
C                                                                       C
      CALL PSGRAF
        WRITE(*,900) FNAME
      CALL PSTERM
900   FORMAT('SEND ''FILE : ',A12,''' TO <1>VIEW.ID;')
      RETURN
      END
      SUBROUTINE GETITM
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
CDCCCCC               SUBROUTINE DESCRIPTION                        CCCCC
C                                                                       C
C--- THIS SUBROUTINE DOES THE FOLLOWING:                                C
C       ASKS THE USER FOR A LABEL NAME.                                 C
C       FINDS THE STORAGE LOCATION NUMBER OF THE LABEL.                 C
C          (ie. GIVEN LABEL AND ITMLBL(NCITEM)=LABEL,                   C
C           FIND NCITEM.)                                               C
C       FINDS THE ITEM ASSOCIATED WITH THAT NUMBER.                     C
C                                                                       C
```

```
C--- PROGRAMMER: JOHN D. REID          DATE: FALL 1983              C
C                                                                   C
C--- CALLED FROM MODIFY.                                            C
C                                                                   C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                   C
CVCCCCCCCC           VARIABLE IDENTIFICATION                 CCCCCCCC
C                                                                   C
 INSERT COMFIL
C                                                                   C
      CHARACTER*4 LABEL
C                                                                   C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                   C
CPCCCCCCCC           PROCESS BLOCK                         BLOCK 0CCC
C                                                                   C
C--- INPUT LABEL.
C
      CALL PROMPT('INPUT LABEL NAME : ')
      READ(*,900) LABEL
C
C--- IF LABEL CAN NOT BE FOUND NCITEM IS RETURNED 0.
C
      NCITEM=0
      DO 100 I=1,NITEM
        IF(ITMLBL(I).EQ.LABEL) THEN
          NCITEM=I
          CITEM=ITMLST(NCITEM)
C
C--- IF ITEM HAS BEEN DELETED BEFORE THEN LET
C    NCITEM=0.
C
          IF(CITEM.EQ.'DELETE')  NCITEM=0
          GOTO 999
        ENDIF
100   CONTINUE
C
C--- CHECK FOR ORIGIN
C
      IF(LABEL.EQ.'O') THEN
        NCITEM=39
        CITEM='ORGIN'
      ENDIF
C                                                                   C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                   C
CFCCCCCCCC           FORMAT STATEMENTS                     BLOCK 9000
C                                                                   C
900   FORMAT(A4)
C                                                                   C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                   C
999   RETURN
      END
      SUBROUTINE ADITEM
```

```
C                                                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                  C
CDCCCCC              SUBROUTINE DESCRIPTION                    CCCCC
C                                                                  C
C--- THIS SUBROUTINE CREATES A STRUCTURE (CITEM)                    C
C   THAT CONSISTS OF THE DESIRED OBJECT (ITEM).                     C
C   THE STRUCTURE IS ADDED TO THE MAIN PICTURE.                     C
C                                                                  C
C--- PROGRAMMER: JOHN D. REID          DATE: FALL 1983              C
C                                                                  C
C--- CALLED FROM:     RESTOR     DRAWNG                             C
C                                                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                  C
CVCCCCCCCC           VARIABLE IDENTIFICATION                CCCCCCCC
C                                                                  C
 INSERT COMFIL
C                                                                  C
      CALL PSGRAF
C
C--- CREATE THE DESIRED STRUCTURE
C
        WRITE(*,900) CITEM
          WRITE(*,920) TRANSX(NCITEM)
          WRITE(*,930) TRANSY(NCITEM)
          WRITE(*,940) ROTZ(NCITEM)
          WRITE(*,950)
          WRITE(*,960)
          WRITE(*,970) SCL(NCITEM)
          WRITE(*,980) ITEM
        PRINT *,'ENDS;'
C
C--- ADD STRUCTURE TO MAIN OBJECT
C
      WRITE(*,990) CITEM
      CALL PSTERM
C                                                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                  C
CFCCCCCCCC           FORMAT STATEMENTS                    BLOCK 9000
C                                                                  C
900   FORMAT(A5,':=BEGINS')
920   FORMAT('TRANSX:=TRANSLATE BY ',F8.4,',0,0;')
930   FORMAT('TRANSY:=TRANSLATE BY 0,',F8.4,',0;')
940   FORMAT('ROTZ:=ROT Z ',F9.4,';')
950   FORMAT('SCLX:=SCALE BY 1;')
960   FORMAT('SCLY:=SCALE BY 1;')
970   FORMAT('SCL:=SCALE BY ',F7.4,';')
980   FORMAT('INSTANCE OF ',A6,';')
990   FORMAT('INCLUDE ',A5,' IN VIEW.OBJECT;')
C                                                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                  C
      RETURN
```

```
      END
      SUBROUTINE LABELS
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
CDCCCCC              SUBROUTINE DESCRIPTION                        CCCCC
C                                                                      C
C---  THIS SECTION LABELS A NEWLY ADDED ITEM.                          C
C     AND ALSO UPDATES A CHANGED ITEM'S LABEL POSITION.                C
C                                                                      C
C--- PROGRAMMER: JOHN D. REID          DATE: FALL 1983                 C
C                                                                      C
C--- CALLED FROM MODIFY.                                               C
C                                                                      C
C--- SUBROUTINES CALLED:     POSLBL                                    C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
CVCCCCCCCCC          VARIABLE IDENTIFICATION                    CCCCCCCC
C                                                                      C
 INSERT COMFIL
 INSERT INPTBK
C
C--- LOCAL VARIABLES
C
      LOGICAL NEWLIN,ENDLIN
      CHARACTER*1 INTEMP
      CHARACTER*2 INTMP2
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
CICCCCCCCCC          INITIALIZATION BLOCK                     BLOCK OCCC
C                                                                      C
      MARGIN=80
      ILO=1
      IHI=20
      IVAL=0
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
CPCCCCCCCCC          PROCESS BLOCK                            BLOCK OCCC
C                                                                      C
C--- DO NOT LABEL ORIGIN
C
      IF(NCITEM.EQ.39) THEN
        RETURN
      ENDIF
C
C--- IF THIS IS A NEW ITEM A LABEL MUST BE CREATED.
C
      IF(NWITEM) THEN
C
C--- CONSTRUCT THE LABEL
C
50      CALL PROMPT('INPUT INDEX NUMBER FOR LABEL (1-20) : ')
```

```
        NEWLIN=.TRUE.
        CALL GETIN(IVAL,ILO,IHI,NEWLIN,ENDLIN)
          IF(IVAL.EQ.0) GOTO 50
        IF(IVAL.LT.10) THEN
          WRITE(INTEMP,'(I1)') IVAL
          ITMLBL(NCITEM)=ITEM(1:1)//INTEMP
        ELSE
          WRITE(INTMP2,'(I2)') IVAL
          ITMLBL(NCITEM)=ITEM(1:1)//INTMP2
        ENDIF
C
C--- CHECK TO SEE IF NEW LABEL ALREADY EXISTS.
C
        ITMP1=NITEM-1
        IF(ITMP1.EQ.0) GOTO 70
        DO 60 J=1,ITMP1
          IF(ITMLBL(J).EQ.ITMLBL(NCITEM)) THEN
            WRITE(*,910)
            GOTO 50
          ENDIF
60      CONTINUE
70    ENDIF
C
C--- GET THE X AND Y POSTION OF THE LABEL
C
      CALL POSLBL
C
C--- ADD THE LABEL TO THE ITEM IT REPRESENTS.
C
      CALL PSGRAF
        WRITE(*,920) ITMLBL(NCITEM),LBLPOS(NCITEM,1),
     +               LBLPOS(NCITEM,2),ITMLBL(NCITEM)
        WRITE(*,930) ITMLBL(NCITEM),CITEM
        CALL PROMPT('REMOVE CROSS;')
      CALL PSTERM
      NWITEM=.FALSE.
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CFCCCCCCCC            FORMAT STATEMENTS                      BLOCK 9000
C                                                                     C
910   FORMAT(//,'LABEL ALREADY EXISTS, TRY AGAIN.',//)
920   FORMAT(A3,':=CHARACTERS ',F7.3,',',F7.3,' ''',A3,''';')
930   FORMAT('INCLUDE ',A3,' IN ',A5,';')
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
      RETURN
      END
      SUBROUTINE POSLBL
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CDCCCCC               SUBROUTINE DESCRIPTION                      CCCCC
C                                                                     C
```

```
C--- THIS SUB. FINDS A DESIRED X AND Y POSITION                      C
C    USING THE DIAL NETWORK WITH CROSS-HAIRS ON THE SCREEN.          C
C                                                                    C
C--- PROGRAMMER: JOHN D. REID          DATE: FALL 1983               C
C                                                                    C
C--- CALLED FROM LABELS.                                             C
C                                                                    C
C--- SUBROUTINES CALLED:     UNLOCK     LOCKIN                       C
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
CVCCCCCCCC           VARIABLE IDENTIFICATION                  CCCCCCCC
C                                                                    C
 INSERT COMFIL
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
CICCCCCCCC           INITIALIZATION BLOCK                   BLOCK OCCC
C                                                                    C
      CITEM='CROSS'
      NTEMP=NCITEM
      NCITEM=40
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
CPCCCCCCCC           PROCESS BLOCK                          BLOCK OCCC
C                                                                    C
C--- DISPLAY CROSS-HAIRS AND ALLOW FOR X AND Y TRANSLATION
C
      CALL PSGRAF
        CALL PROMPT('DISPLAY CROSS;')
      CALL PSTERM
      CALL UNLOCK
C
C--- ASK USER TO TYPE OK AFTER HE POSITIONS THE
C    CROSS-HAIRS.
C
      WRITE(*,900)
100   CALL PROMPT('TYPE OK TO CONTINUE : ')
      READ(*,920) IANS
      IF(IANS.NE.'OK') GOTO 100
C
C--- LOCKIN AND GET POSITION OF THE CROSS-HAIRS.
C
      CALL LOCKIN
      LBLPOS(NTEMP,1)=TRANSX(40)
      LBLPOS(NTEMP,2)=TRANSY(40)
C
C--- RESET VARIABLES
C
      NCITEM=NTEMP
      CITEM=ITMLST(NCITEM)
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
```

```
CFCCCCCCCC            FORMAT STATEMENTS                    BLOCK 9000
C                                                                   C
900   FORMAT(//,'USING THE DIALS, POSITION THE CROSS-HAIRS IN THE'
     +     ,/,'DESIRED LOCATION FOR THE LABEL.',/)
920   FORMAT(A2)
C                                                                   C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                   C
      RETURN
      END
```

```
C                                                                   C
C   COMMON FILE COMFIL                                              C
C                                                                   C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                   C
CDCCCCC             COMMON FILE DESCRIPTION                     CCCCC
C                                                                   C
C--- COMMON FILE FOR THE GRAPHIC DESIGN PROGRAM.                     C
C                                                                   C
C--- PROGRAMMER: JOHN D. REID          DATE: FALL 1983              C
C                                                                   C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                   C
CVCCCCCCCC            VARIABLE IDENTIFICATION                 CCCCCCCC
C                                                                   C
C                                                                   C
C  THE FOLLOWING VARIABLES THAT BEGIN WITH 'NUM' REPRESENT          C
C  THE NUMBER OF THAT SPECIFIC ITEM ADDED (AT ANYTIME)              C
C  TO THE DESIGN.                                                   C
C    FOR EXAMPLE: IF NUMS=2 , THEN 2 SPRINGS HAVE BEEN              C
C                 ADDED TO THE DESIGN AT SOME TIME                  C
C                 DURING ITS DEVELOPMENT.                           C
C   NOTE: THESE NUMBERS DO NOT TAKE INTO ACCOUNT ITEMS              C
C         THAT HAVE BEEN DELETED. THEY ARE A RUNNING TOTAL.         C
C    NUMG - GROUNDS                                                 C
C    NUMB - BARS                                                    C
C    NUMS - SPRINGS                                                 C
C    NUMD - DAMPERS                                                 C
C    NUMM - MASSES                                                  C
C    NUMJ - JOINTS                                                  C
C    NUMF - FORCES                                                  C
C    NUMP - PINS                                                    C
C    NUMV - RIGHT VELOCITIES                                        C
C    NUML - LEFT VELOCITIES                                         C
C    NUMR - ROUGH (FRICTION)                                        C
C                                                                   C
C   OTHER VARIABLES                                                 C
C                                                                   C
C    NITEM - TOTAL NUMBER OF ITEMS ADDED                            C
C                                                                   C
```

```
C     NDEL - NUMBER OF ITEMS DELETED                                      C
C                                                                         C
C                                                                         C
C-- LOGICAL VARIABLES                                                     C
C                                                                         C
C     LOCKED - TRUE:  NO TRANSLATION,ROTATIONS OR SCALING                 C
C                     CAPABILITIES ON ANY ITEM.                           C
C              FALSE: TRANS.,ROT., AND SCALING CAPABILITIES               C
C                     ON THE CURRENT ITEM (CITEM).                        C
C                                                                         C
C     FILE   - FALSE: NO FILE NAME HAS BEEN SPECIFIED AS OF YET.          C
C                                                                         C
C     OLDFIL - TRUE:  THE FILE THAT IS BEING WORKED ON IS                 C
C                     OLD. (IT WOULD HAVE TO BE OVERWRITTEN               C
C                     TO BE SAVED)                                        C
C                                                                         C
C     NWITEM - TRUE:  A NEW ITEM HAS BEEN ADDED AND DOES                  C
C                     NOT HAVE A LABEL AS OF YET.                         C
C              FALSE: ITEM UNDER CONSIDERATION ALREADY HAS                C
C                     BEEN LABELLED.                                      C
C                                                                         C
C                                                                         C
C--- ARRAYS                                                               C
C                                                                         C
C     ITMLST(40) - THE PS300 STRUCTURE NAME OF EACH ITEM IN               C
C                  THE DESIGN.                                            C
C                                                                         C
C     ITMTYP(i) - THE TYPE OF ELEMENT ITMLST(i) IS.                       C
C                 (SPRING,BAR,ETC.)                                       C
C                                                                         C
C     ITMLBL(i) - THE LABEL NAME ASSOCIATED WITH ITMLST(i).               C
C                                                                         C
C     LBLPOS(i,2) - THE X AND  Y POSITION OF ITMLBL(i).                   C
C                                                                         C
C                                                                         C
C--- WORK VARIABLES                                                       C
C                                                                         C
C     CITEM - CURRENT ITEM NAME UNDER CONSIDERATION.                      C
C             (CITEM=ITMLST(NCITEM))                                      C
C                                                                         C
C     NCITEM - THE NUMBER OF THE CURRENT ITEM BEING WORKED ON.            C
C                                                                         C
C     ITEM - TYPE OF CURRENT ITEM UNDER CONSIDERATION.                    C
C            (ITEM=ITMTYP(NCITEM))                                        C
C                                                                         C
      CHARACTER*32 WORD
      CHARACTER*12 FNAME
      CHARACTER*6  CITEM,ITEM,ITMLST(40),ITMDEL(40),ITMTYP(40)
      CHARACTER*4 HLITE,ITMLBL(40)
      CHARACTER*2 IANS
      LOGICAL LOCKED,FILE,OLDFIL,NWITEM
      REAL LBLPOS(40,2)
C
      COMMON/BK1/NUMG,NUMS,NUMB,NUMM,NUMR,NUMP,
     +            NUML,NUMV,NUMF,NUMJ,NUMD
```

```
      COMMON/BK2/WORD,CITEM,ITEM,HLITE,ITMTYP,ITMLBL,
     +           NITEM,ITMLST,NDEL,ITMDEL,NCITEM,FNAME
      COMMON/BK3/TRANSX(40),TRANSY(40),ROTZ(40),SCL(40),
     +           SCLX(40),SCLY(40),LBLPOS
      COMMON/BK4/LOCKED,FILE,OLDFIL,NWITEM,IANS
C                                                                  C
C END OF COMFIL CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                  C
```

```
C                                                                  C
C   COMMON FILE COMBG                                              C
C                                                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                  C
CDCCCCC               COMMON FILE DESCRIPTION                 CCCCC
C                                                                  C
C--- COMMON FILE FOR THE BOND GRAPH PROGRAM (BILDBG).             C
C                                                                  C
C--- PROGRAMMER: JOHN D. REID          DATE: FALL 1983             C
C                                                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                  C
CVCCCCCCCC            VARIABLE IDENTIFICATION              CCCCCCCC
C                                                                  C
C-- CHARACTERS
C
C    CNNAM  -  THE CONNECTOR NAME LIST. ( S, D, F, V, R, X )
C    MPNAM  -  THE MOTION POINT NAME LIST. ( M, B, J, G, P )
C    NEWLBL -  THE RE-ORDERED ITEM LIST. THE NEW ORDER IS:
C                 M, B, J, G, P, S, D, F, V, R
C
C-- LOGICAL VARIABLES
C
C    NEWLIN, ENDLIN  -  USED TO HELP READ INPUT IN SUBROUTINES
C                       GETWD, GETRL AND GETIN.
C
C-- REAL AND INTEGER VARIABLES.
C
C    MOTTYP - THE TYPE OF ALLOWABLE MOTION FOR
C             THE SPECIFIED MOTION POINT.
C             MOTTYP(MP I) = 0   NO MOTION
C             MOTTYP(MP I) = 1   X MOTION
C             MOTTYP(MP I) = 2   Y MOTION
C             MOTTYP(MP I) = 3   X AND Y MOTION
C             MOTTYP(MP I) = 4   THETA MOTION
C             MOTTYP(MP I) = 5   X AND THETA MOTION
C             MOTTYP(MP I) = 6   Y AND THETA MOTION
C             MOTTYP(MP I) = 7   X, Y AND THETA MOTION
C
C    NMOT  -  THE NUMBER OF MOTIONS FOR EACH MOTTYP.
C             ex. MP I CAN MOVE IN X AND THETA, THEN
```

```
C               NMOT(MOTTYP(MP I))=NMOT(5) = 2,
C               MP I HAS 2 WAYS IT CAN MOVE.
C
C     XMP,VMP  -  THE POSITION AND VELOCITY VECTOR FOR
C                 THE MOTION POINTS.
C
C     IXMPPT   -  NODE POINTER LIST TO ITS XMP OR VMP.
C
C     XEP,VEP  - THE POSITON AND VELOCITY VECTOR FOR
C                THE END POINTS.
C
C     IXEPPT   - NODE POINTER LIST TO ITS XEP OR VEP.
C
C     XCN,VCN  - THE POSITION AND VELOCITY VECTOR FOR
C                CONNECTORS.
C
C     IXCNPT   - NODE POINTER LIST TO ITS XCN OR VCN.
C
C     MTF  -  TRANSFORMER LIST
C
C     MTFDAT  -  DATA FOR MTF : FUNCTION, CONSTANTS, ARGUMENTS.
C
C     GEOMAS(i,j,1) - THE X-DISTANCE FROM THE CENTER OF MASS i
C                     TO THE ATTACHED END POINT OF CONNECTOR j.
C
C     GEOMAS(i,j,2) - THE Y-DISTANCE FROM THE CENTER OF MASS i
C                     TO THE ATTACHED END POINT OF CONNECTOR j.
C
C     GEOBAR(i,j) - THE DISTANCE FROM THE ROTATION POINT ON BAR i
C                   TO THE ATTACHED END POINT OF CONNECTOR j.
C                                                                   C
C                                                                   C
      PARAMETER (MAXNCN=20)
      PARAMETER (MAXNMP=20)
      PARAMETER (MAXNMS=10)
      PARAMETER (MAXNBR=10)
      PARAMETER (MAXNTF=MAXNMP*4)
C                                                                   C
      CHARACTER*1 ANS
      CHARACTER*4 CNNAM(MAXNCN),MPNAM(MAXNMP),NEWLBL(40)
      CHARACTER*32 WRD
      LOGICAL NEWLIN,ENDLIN
      INTEGER CNONMP(MAXNCN*2),CPTR(MAXNMP+1),CNTYP(MAXNCN),
     +        PNTLBL(40)
      REAL    MTFDAT(MAXNTF,11)
C                                                                   C
      COMMON/BKBG1/ANS,CNNAM,MPNAM,NEWLBL,WRD
      COMMON/BKBG2/NEWLIN,ENDLIN
      COMMON/BKBG3/MPONCN(MAXNCN,2),CNONMP,CPTR,MOTTYP(MAXNMP),
     +             CNTYP,NMOT(0:7),MP1PTR(MAXNMP),PNTLBL,
     +             IEPTYP(MAXNCN),MTF(MAXNTF)
      COMMON/BKBG4/GEOMAS(MAXNMS,MAXNCN,2),
     +             GEOBAR(MAXNBR,MAXNCN),MTFDAT
      COMMON/BKBG5/NMP,NMPWM,NCN,NMS,NBR,NTF,
     +             I,J,NODEO,IXORY,NTFEL,ITFEL(4)
```

```
      COMMON/BKBG6/XMP(MAXNMP*3),VMP(MAXNMP*3),
     +             XEP(MAXNCN*4),VEP(MAXNCN*4),
     +             XCN(MAXNCN),VCN(MAXNCN)
      COMMON/BKBG7/IXMPPT(MAXNMP*3),IXEPPT(MAXNCN*4),
     +             IXCNPT(MAXNCN)
C                                                                     C
C END OF COMBG CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
```

```
C                                                                     C
C   COMMON FILE SYBGBK                                                C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CDCCCCC               COMMON FILE DESCRIPTION                     CCCCC
C                                                                     C
C--- COMMON FILE SYBGBK: SYSTEM BONDGRAPH BLOCK                       C
C                                                                     C
C--- SUPPORTS GRAPDR, BOND GRAPH DATA STRUCTURE                       C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CVCCCCCCCCC             VARIABLE IDENTIFICATION                CCCCCCCC
C                                                                     C
C--- VARIABLES:
C      IELLST  LIST OF NODES BY TYPE NUMBER
C      NBIMX   BONDS INCIDENT ON A GIVEN NODE
C      IBMX    NODES ADJACENT TO A GIVEN BOND
C      NPTR    POINTER TO START OF BONDS IN NBIMX FOR NODE I
C      IELNAM  NAMES OF NODES
C      IBNAM   NAMES OF BONDS
C
C--- NOTE. THIS BLOCK SHOULD BE COORDINATED WITH GREDBK, SINCE
C          THE ARRAYS MATCH AND THE DIMENSIONS SET BY PARAMETER
C          DECLARATIONS SHOULD TOO.
C                                                                     C
      PARAMETER (MNEL=110)
      PARAMETER (MNBD=120)
C
      CHARACTER*4 NTYP(9)
      CHARACTER*32 IELNAM(MNEL),IBNAM(MNBD)
C                                                                     C
      COMMON /SYBGB1/ NEL,NBD,IELLST(MNEL),NBIMX(MNBD*2),
     +                IBMX(MNBD,2),NPTR(MNEL+1),NMCR
      COMMON /SYBGB2/ IELNAM,IBNAM,NTYP
C                                                                     C
C END OF SYBGBK CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
```

```
{ PS300 FILE: MENUS }

{ THIS FILE DOES THE FOLLOWING: }

{   1.  DEFINES THE 3 MENUS: MASTER, MODIFY AND DRAWNG.         }
{   2.  POSITIONS THEM TO THE DESIRED LOCATION ON THE SCREEN. }
{   3.  DISPLAYS THE MENUS ON THE PS300.                        }


MASTER:=BEGINS
  CONTRAST:=SET CONTRAST .5;
  CHARACTER SCALE .024;
  TRANSLATE BY .55,.9,0;
    CHAR 0,0 'C  CREATE NEW FILE';
    CHAR 0,-.1 'R  RESTORE OLD FILE';
    CHAR 0,-.2 'M  MODIFY FILE';
    CHAR 0,-.3 'S  SAVE FILE';
    CHAR 0,-.4  'E  EXIT PROGRAM';
ENDS;
DISPLAY MASTER;


MODIFY:=BEGINS
  CONTRAST:=SET CONTRAST .5;
  CHARACTER SCALE .024;
  TRANSLATE BY .55,.2,0;
    CHAR 0,0 'A  ADD NEW ITEM';
    CHAR 0,-.1 'C  CHANGE ITEM';
    CHAR 0,-.2 'L  LOCK IN ITEM';
    CHAR 0,-.3 'D  DELETE ITEM';
    CHAR 0,-.4  'RP  REPLACE ITEM';
    CHAR 0,-.5 'R  RETURN';
ENDS;
DISPLAY MODIFY;

DRAWNG:=BEGINS
  SET CONTRAST TO .1;
  CHAR SCALE .06;
    SCALE BY .4;
    TRANSLATE BY .525,-.7,0;
    TRANSLATE BY .74,-.5,0;
    CHAR 0,-.03 'G';
    CHAR 0,-.28 'B';
    CHAR 0,-.53 'S';
    CHAR 0,-.78 'D';
    CHAR 0,-1.  'P';
    CHAR 0,-1.21 'M';
   TRANSLATE BY .08,0,0;
     INSTANCE GROUND;
    TRANSLATE BY .2,-.25,0;
     INSTANCE BAR;
    TRANSLATE BY -.2,-.25,0;
     INSTANCE SPRING;
    TRANSLATE BY 0,-.25,0;
     INSTANCE DAMPER;
```

```
    TRANSLATE BY .15,-.25,0;
     INSTANCE PIN;
    TRANSLATE BY .05,-.1875,0;
     INSTANCE MASS;
  TRANSLATE BY .8475,1.1775,0;
    CHARACTER 0,-.03 'R';
    CHARACTER 0,-.25 'J';
    CHARACTER 0,-.53 'F';
    CHAR 0,-.75 'VR';
    CHAR 0,-1.0 'VL';
    CHAR 0,-1.21 'E';
  TRANSLATE BY -.46,.0;
  INSTANCE ROUGH;
  TRANSLATE BY .2,-.225,0;
  INSTANCE JOINT;
  TRANSLATE BY -.1,-.275,0;
  INSTANCE FORCE;
  TRANSLATE BY .05,-.25,0;
  INSTANCE VRIGHT;
  TRANSLATE BY .1,-.25,0;
  INSTANCE VLEFT;
    CHARACTER -.17,-.21 'EXIT';
ENDS;
DISPLAY DRAWNG;
```

```
{ PS300 FILE: VIEW.OBJECT }

{ THIS FILE DOES THE FOLLOWING: }

{   1.  DEFINES THE MAIN GRAPHICS VIEW.                            }
{   2.  DEFINES THE ORIGIN COORDINATE SYSTEM.                      }
{   3.  DEFINES THE CROSS-HAIRS USED TO POSITION LABELS.           }


VIEW:=BEGINS
  CONTRAST:=SET CONTRAST .5;
  CHAR SCALE .05;
  ID:=CHAR -1.,.9 'FILE :              ';
  CHAR SCALE .6;
  OBJECT:= INSTANCE OF ORGIN;
ENDS;


ORGIN:=BEGINS
{ }
{ ALLOW FOR TRANSLATION AND SCALING. }
{ }
 TRANSX:=TRANSLATE BY 0,0,0;
 TRANSY:=TRANSLATE BY 0,0,0;
 SCL:=SCALE BY .75;
  VEC SEPARATE Z=0 N=12
```

```
    -.025,0    .2,0
    0,-.025    0,.2
    0,0        .13,.09
    .115,0     .08,.055
    .08,.055   .083,.04
    .08,.055   .095,.045;
   CHAR SCALE 1.33;
   CHAR .21,-.025 'X';
   CHAR -.025,.2 'Y';
   CHAR .14,.09 '0';
   CHAR .14,.09 '-';
   CHAR -.04,.01 'O';
 ENDS;


 CROSS:=BEGINS
 { }
 { ALLOW FOR X AND Y TRANSLATIONS. }
 { }
  TRANSX:=TRANSLATE BY 0,0,0;
  TRANSY:=TRANSLATE BY 0,0,0;
  VEC SEPARATE Z=0 N=2
    -.05,0    .05,0
    .0,-.05    .0,.05;
 ENDS;




 { PS300 FILE: OBJECTS }

 { THIS FILE DOES THE FOLLOWING: }

 {   1.  DEFINES A CIRCLE TO BE USED AS AN END POINT FOR THE    }
 {       CONECTORS.                                               }
 {   2.  DEFINES THE INDIVIDUAL ITEMS (eg. SPRING, DAMPER)       }


 CIRCLE:=BEGINS
   RATI POLY
     .020,0,0,2
     -.020,-.020,0,-2
     0,.010,0,1
   CHORDS=25;
   RATI POLY
     .020,0,0,-2
     -.020,-.020,0,2
     0,.010,0,-1
   CHORDS=25;
 ENDS;


 GROUND:=BEGINS
    VEC ITEMIZED Z=0 N=16
```

```
     P 0,0       L .4,0
     P .025,-.025  L .05,0
     P .075,-.025  L .1,0
     P .125,-.025  L .15,0
     P .175,-.025  L .2,0
     P .225,-.025  L .25,0
     P .275,-.025  L .3,0
     P .325,-.025  L .35,0;
ENDS;


BAR:=BEGINS
    VEC Z=0 N=5  .2,.00625  -.2,.00625  -.2,-.00625
      .2,-.00625  .2,.00625;
ENDS;


SPRING:=BEGINS
  INSTANCE CIRCLE;
    VEC Z=0 N=10
      0,0  .0875,0  .10625,.05  .14375,-.05
      .18125,.05  .21875,-.05  .25625,.05
      .29375,-.05  .3125,0  .4,0;
  TRANSLATE BY .4,0,0;
  INSTANCE CIRCLE;
ENDS;


DAMPER:=BEGINS
  INSTANCE CIRCLE;
   VEC ITEMIZED Z=0 N=10
     P 0,0     L .175,0
     P .2375,.0375  L .175,.0375  L .175,-.0375  L .2375,-.0375
     P .2,.025  L .2,-.025
     P .2,0      L .4,0;
  TRANSLATE BY .4,0,0;
  INSTANCE CIRCLE;
ENDS;


PIN:=BEGINS
   VEC Z=0 N=4 0,0 .1,0 .05,.07071 0,0;
ENDS;


MASS:=BEGINS
   VEC Z=0 N=5  .0625,.0625  .0625,-.0625  -.0625,-.0625
     -.0625,.0625  .0625,.0625;
ENDS;


ROUGH:=BEGINS
   VEC Z=0 N=18  0,0  .0125,.0125  .0375,-.0125  .0625,.0125
       .0875,-.0125  .1125,.0125  .1375,-.0125  .1625,.0125
       .1875,-.0125  .2125,.0125  .2375,-.0125  .2625,.0125
```

```
        .2875,-.0125  .3125,.0125  .3375,-.0125  .3625,.0125
        .3875,-.0125  .4,0;
ENDS;


JOINT:=BEGINS
   VEC ITEMIZED Z=0 N=8  P -.025,.025  L .025,-.025
      P .025,.025  L .025,-.025  L -.025,-.025
      L -.025,.025  L .025,.025  L -.025,-.025;
ENDS;


FORCE:=BEGINS
   VEC ITEM Z=0 N=7  P 0,0  L .175,0
     P 0,.025  L .175,.025
     P .15,.05  L .2,.0125  L .15,-.025;
  TRANSLATE BY 0,.0125,0;
  INSTANCE CIRCLE;
ENDS;


VRIGHT:=BEGINS
  INSTANCE CIRCLE;
   VEC ITEM Z=0 N=6  P 0,0  L 0,.05  L .1,.05
     P .075,.0625  L .1,.05  L .075,.0375;
ENDS;


VLEFT:=BEGINS
  INSTANCE CIRCLE;
   VEC ITEM Z=0 N=6  P 0,0   L 0,.05  L -.1,.05
     P -.075,.0625  L -.1,.05  L -.075,.0375;
ENDS;




{ PS300 FILE: DIAL.NETWORK }

{ THIS FILE CREATES A DIAL NETWORK THAT IS USED TO TRANSLATE, }
{ ROTATE AND SCALE THE ITEM THAT THE NETWORK IS ATTACHED TO.  }

{ }
{ CREATE THE Z-ROTATION FUNCTION AND SEND OR CONNECT ITS      }
{ INPUTS.                                                     }
ZDIAL:=F:DZROTATE;
SEND 0 TO <2>ZDIAL;
SEND 32 TO <3>ZDIAL;
CONN DIALS<7>:<1>ZDIAL;
{ }
{ CREATE THE X-TRANSLATION FUNCTION AND SEND OR CONNECT ITS   }
{ INPUTS.                                                     }
XVALUE:=F:ACCUMULATE;
SEND 0 TO <2>XVALUE;
```

```
SEND 1 TO <4>XVALUE;
CONN DIALS<5>:<1>XVALUE;
{ }
{ CREATE THE Y-TRANSLATION FUNCTION AND SEND OR CONNECT ITS    }
{ INPUTS.                                                      }
YVALUE:=F:ACCUMULATE;
SEND 0 TO <2>YVALUE;
SEND 1 TO <4>YVALUE;
CONN DIALS<6>:<1>YVALUE;
{ }
{ CREATE THE SCALING FUNCTION AND SEND OR CONNECT ITS INPUTS. }
{ }
SCLVAL:=F:DSCALE;
SEND 1 TO <2>SCLVAL;
SEND 1 TO <3>SCLVAL;
SEND 0.1 TO <5>SCLVAL;
CONN DIALS<8>:<1>SCLVAL;
{ }
{ CREATE THE X AND Y TRANSLATION VECTOR AND CONNECT ITS        }
{ INPUTS.                                                      }
XVECT:=F:XVECTOR;
YVECT:=F:YVECTOR;
CONN XVALUE<1>:<1>XVECT;
CONN YVALUE<1>:<1>YVECT;
{ }
{ CREATE THE X-SCALING NETWORK.                                }
{ }
SCALEX:=F:ACCUMULATE;
SEND 1 TO <2>SCALEX;
SEND 1 TO <4>SCALEX;
SEND .01 TO <6>SCALEX;

 SX2VEC:=F:VECC;
 SEND 1 TO <2>SX2VEC;
 SX3VEC:=F:VECC;
 SEND 1 TO <2>SX3VEC;

SCXMAT:=F:SCALE;

 CONN SX3VEC<1>:<1>SCXMAT;
 CONN SX2VEC<1>:<1>SX3VEC;
 CONN SCALEX<1>:<1>SX2VEC;
CONN DIALS<3>:<1>SCALEX;
{ }
{ CREATE THE Y-SCALING NETWORK.                                }
{ }
SCALEY:=F:ACCUMULATE;
SEND 1 TO <2>SCALEY;
SEND 1 TO <4>SCALEY;
SEND .01 TO <6>SCALEY;

 SY2VEC:=F:CVEC;
 SEND 1 TO <1>SY2VEC;
 SY3VEC:=F:VECC;
 SEND 1 TO <2>SY3VEC;
```

```
SCYMAT:=F:SCALE;

 CONN SY3VEC<1>:<1>SCYMAT;
 CONN SY2VEC<1>:<1>SY3VEC;
 CONN SCALEY<1>:<2>SY2VEC;
CONN DIALS<4>:<1>SCALEY;
{ }
{ SEND A LABEL TO THE APPROPRIATE DIAL.                        }
{ }
SEND 'SCALE X'  TO <1>DLABEL3;
SEND 'SCALE Y'  TO <1>DLABEL4;
SEND 'TRANS X'  TO <1>DLABEL5;
SEND 'TRANS Y'  TO <1>DLABEL6;
SEND 'ROTATE Z' TO <1>DLABEL7;
SEND 'SCALE'    TO <1>DLABEL8;




{ PS300 FILE: OUTPUT.NETWORK }

{ THIS FILE CREATES A NETWORK THAT WILL SEND PS300 OUTPUT      }
{ TO THE TERMINAL SCREEN (HOSTOUT).                            }


OUTPUT:=F:PRINT;
BLANKPAD:=F:CONCATENATEC;
SEND '            ' TO <2>BLANKPAD;
CONN OUTPUT<1>:<1>BLANKPAD;
TOHOST:=F:CONCATENATEC;
SEND CHAR(141) TO <2>TOHOST;
CONN TOHOST<1>:<1>HOSTOUT;
CONN BLANKPAD<1>:<1>TOHOST;
{ }
{ DATA IS A FUNCTION THAT WILL SIMULTANEOUSLY SEND THE         }
{ SIX VALUES STORED IN IT TO THE TERMINAL SCREEN.              }
{ }
DATA:=F:SYNC(7);
CONN DATA<1>:<1>OUTPUT;
CONN DATA<2>:<1>OUTPUT;
CONN DATA<3>:<1>OUTPUT;
CONN DATA<4>:<1>OUTPUT;
CONN DATA<5>:<1>OUTPUT;
CONN DATA<6>:<1>OUTPUT;
```

# APPENDIX B

# TRANSFORMER FUNCTIONS

# APPENDIX B

## TRANSFORMER FUNCTIONS

The functions in this appendix appear in the following sequence:

1. FUNCTION 1
2. FUNCTION 2
3. FUNCTION 3
4. FUNCTION 4
5. FUNCTION 5

```
      FUNCTION 1
C                                                                      C
CDCCCCC                  FUNCTION DESCRIPTION                      CCCCC
C                                                                      C
C--- THIS FUNCTION EVALUATES THE TRANSFORMER BETWEEN A                 C
C    CONNECTOR END POINT X-VELOCITY 1-JUNCTION AND THE                 C
C    THETA-VELOCITY 1-JUNCTION OF THE MASS MOTION POINT IT             C
C    IS ATTACHED TO.                                                   C
C                                                                      C
C--- PROGRAMMER: JOHN D. REID          DATE: FALL 1983                 C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
CVCCCCCCCC               VARIABLE IDENTIFICATION                CCCCCCCC
C                                                                      C
C    TFN - TRANSFORMER NUMBER                                          C
C                                                                      C
C    2 CONSTANTS:   a  -  X-DISTANCE BETWEEN MASS CENTER AND EP.       C
C                   b  -  Y-DISTANCE BETWEEN MASS CENTER AND EP.       C
C                                                                      C
C    1 ARGUMENT:  MPPTR - POINTER TO THE THETA MOTION POINT            C
C                         VALUE IN XMP.                                C
C                                                                      C
C    MTFVAL( TFN )  -  VALUE OF TRANSFORMER TFN.                       C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
CICCCCCCCC               INITIALIZATION BLOCK                 BLOCK CCCC
C                                                                      C
      a = MTFDAT(1,2)
      b = MTFDAT(1,3)
      MPPTR = MTFDAT(1,4)
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
CPCCCCCCCC               PROCESS BLOCK                        BLOCK CCCC
C                                                                      C
      MTFVAL(TFN) = -( a*SIN( XMP(MPPTR) ) + b*COS( XMP(MPPTR) ) )
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      RETURN
      END
      FUNCTION 2
C                                                                      C
CDCCCCC                  FUNCTION DESCRIPTION                      CCCCC
C                                                                      C
C--- THIS FUNCTION EVALUATES THE TRANSFORMER BETWEEN A                 C
C    CONNECTOR END POINT Y-VELOCITY 1-JUNCTION AND THE                 C
C    THETA-VELOCITY 1-JUNCTION OF THE MASS MOTION POINT IT             C
C    IS ATTACHED TO.                                                   C
C                                                                      C
C--- PROGRAMMER: JOHN D. REID          DATE: FALL 1983                 C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
C                                                                    C
CVCCCCCCCC           VARIABLE IDENTIFICATION                 CCCCCCCC
C                                                                    C
C    TFN - TRANSFORMER NUMBER                                        C
C                                                                    C
C    2 CONSTANTS:    a  -  X-DISTANCE BETWEEN MASS CENTER AND EP.   C
C                    b  -  Y-DISTANCE BETWEEN MASS CENTER AND EP.   C
C                                                                    C
C    1 ARGUMENT:  MPPTR - POINTER TO THE THETA MOTION POINT          C
C                         VALUE IN XMP.                              C
C                                                                    C
C    MTFVAL( TFN )  -  VALUE OF TRANSFORMER TFN.                     C
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
CICCCCCCCC           INITIALIZATION BLOCK                  BLOCK CCCC
C                                                                    C
      a = MTFDAT(1,2)
      b = MTFDAT(1,3)
      MPPTR = MTFDAT(1,4)
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
CPCCCCCCCC           PROCESS BLOCK                         BLOCK CCCC
C                                                                    C
      MTFVAL(TFN) = a*COS( XMP(MPPTR) ) - b*SIN( XMP(MPPTR) )
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      RETURN
      END
      FUNCTION 3
C                                                                    C
CDCCCCC                FUNCTION DESCRIPTION                     CCCCC
C                                                                    C
C--- THIS FUNCTION EVALUATES THE TRANSFORMER BETWEEN A               C
C    CONNECTOR END POINT X-VELOCITY 1-JUNCTION AND THE               C
C    THETA-VELOCITY 1-JUNCTION OF THE BAR MOTION POINT IT            C
C    IS ATTACHED TO.                                                 C
C                                                                    C
C--- PROGRAMMER: JOHN D. REID          DATE: FALL 1983               C
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
CVCCCCCCCC           VARIABLE IDENTIFICATION                 CCCCCCCC
C                                                                    C
C    TFN - TRANSFORMER NUMBER                                        C
C                                                                    C
C    1 CONSTANTS:    a  -  DISTANCE BETWEEN BAR ROTATION             C
C                          POINT AND THE CONNECTOR END POINT.        C
C                                                                    C
C    1 ARGUMENT:  MPPTR - POINTER TO THE THETA MOTION POINT          C
C                         VALUE IN XMP.                              C
C                                                                    C
C    MTFVAL( TFN )  -  VALUE OF TRANSFORMER TFN.                     C
```

```
C                                                                 C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                 C
CICCCCCCCC          INITIALIZATION BLOCK                 BLOCK CCCC
C                                                                 C
      a = MTFDAT(1,2)
      MPPTR = MTFDAT(1,3)
C                                                                 C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                 C
CPCCCCCCCC          PROCESS BLOCK                        BLOCK CCCC
C                                                                 C
      MTFVAL(TFN) = - a * SIN( XMP(MPPTR) )
C                                                                 C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      RETURN
      END
      FUNCTION 4
C                                                                 C
CDCCCCC             FUNCTION DESCRIPTION                      CCCCC
C                                                                 C
C--- THIS FUNCTION EVALUATES THE TRANSFORMER BETWEEN A            C
C    CONNECTOR END POINT Y-VELOCITY 1-JUNCTION AND THE            C
C    THETA-VELOCITY 1-JUNCTION OF THE BAR MOTION POINT IT         C
C    IS ATTACHED TO.                                              C
C                                                                 C
C--- PROGRAMMER: JOHN D. REID         DATE: FALL 1983             C
C                                                                 C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                 C
CVCCCCCCCC          VARIABLE IDENTIFICATION                CCCCCCCC
C                                                                 C
C    TFN - TRANSFORMER NUMBER                                     C
C                                                                 C
C    1 CONSTANTS:    a  -  DISTANCE BETWEEN BAR ROTATION          C
C                          POINT AND THE CONNECTOR END POINT.     C
C                                                                 C
C    1 ARGUMENT:  MPPTR - POINTER TO THE THETA MOTION POINT       C
C                          VALUE IN XMP.                          C
C                                                                 C
C    MTFVAL( TFN )  -  VALUE OF TRANSFORMER TFN.                  C
C                                                                 C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                 C
CICCCCCCCC          INITIALIZATION BLOCK                 BLOCK CCCC
C                                                                 C
      a = MTFDAT(1,2)
      MPPTR = MTFDAT(1,3)
C                                                                 C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                 C
CPCCCCCCCC          PROCESS BLOCK                        BLOCK CCCC
C                                                                 C
      MTFVAL(TFN) = a * COS( XMP(MPPTR) )
```

```
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      RETURN
      END
      FUNCTION 5
C                                                                     C
CDCCCCC               FUNCTION DESCRIPTION                       CCCCC
C                                                                     C
C--- THIS FUNCTION CALCULATES THE TRANSFORMERS BETWEEN A              C
C    CONNECTOR AND ITS TWO END POINTS.                                C
C    ALSO CALCULATED ARE THE CONNECTOR DISPLACEMENT AND               C
C    VELOCITY VALUES.                                                 C
C                                                                     C
C--- PROGRAMMER: JOHN D. REID          DATE: FALL 1983                C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CVCCCCCCCC             VARIABLE IDENTIFICATION                  CCCCCCCC
C                                                                     C
C    TFN  -  TRANSFORMER NUMBER THAT CALLED THIS FUNCTION.            C
C                                                                     C
C    XCN  -  RELATIVE DISPLACEMENT OF THE CONNECTOR.                  C
C                                                                     C
C    VCN  -  RELATIVE VELOCITY OF THE CONNECTOR.                      C
C                                                                     C
C    R(1)  -  TRANSFORMER TOWARDS THE X-VELOCITY OF END POINT 1.      C
C                                                                     C
C    R(2)  -  TRANSFORMER TOWARDS THE Y-VELOCITY OF END POINT 1.      C
C                                                                     C
C    R(3)  -  TRANSFORMER TOWARDS THE X-VELOCITY OF END POINT 2.      C
C                                                                     C
C    R(4)  -  TRANSFORMER TOWARDS THE Y-VELOCITY OF END POINT 2.      C
C                                                                     C
C    4 ARGUMENTS: X1 = VALUE OF X-POSITION OF END POINT 1.            C
C                                                                     C
C                 Y1 = VALUE OF Y-POSITION OF END POINT 1.            C
C                                                                     C
C                 X2 = VALUE OF X-POSITION OF END POINT 2.            C
C                                                                     C
C                 Y2 = VALUE OF Y-POSITION OF END POINT 2.            C
C                                                                     C
C              SIMILARLY, VX1 THROUGH VY2 ARE THE VELOCITIES.         C
C                                                                     C
C    1 MODULATED FUNCTION NUMBER = MTFDAT( TFN,6 )                    C
C                                                                     C
C          THE NUMBER CORRESPONDING TO R(n) ABOVE THAT IS             C
C          ASSOCIATED WITH TRANSFORMER TFN.                           C
C                                                                     C
C    1 POINTER = MTFDAT( TFN,7 )  - NUMBER OF POINTERS TO FOLLOW.     C
C                                                                     C
C    CONNECTOR NUMBER = MTFDAT( TFN,8 )                               C
C    POINTERS  -  THE TRANSFORMERS THAT ARE EVALUATED WHEN            C
C                 THIS TRANSFORMER (TFN) IS CALCULATED.               C
C                                                                     C
```

```
C    MFTVAL  -  THE VALUE OF THE TRANSFORMERS.                              C
C                                                                           C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                           C
CICCCCCCCC              INITIALIZATION BLOCK                     BLOCK CCCC
C                                                                           C
      X1 = XEP( MTFDAT( TFN,2 ) )
      Y1 = XEP( MTFDAT( TFN,3 ) )
      X2 = XEP( MTFDAT( TFN,4 ) )
      Y2 = XEP( MTFDAT( TFN,5 ) )
      VX1 = VEP( MTFDAT( TFN,2 ) )
      VY1 = VEP( MTFDAT( TFN,3 ) )
      VX2 = VEP( MTFDAT( TFN,4 ) )
      VY2 = VEP( MTFDAT( TFN,5 ) )
C                                                                           C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                           C
CPCCCCCCCC              PROCESS BLOCK                            BLOCK CCCC
C                                                                           C
C--- CONNECTOR DISPLACEMENT.
C
      XCN( MTFDAT( TFN,8 ) ) = SQRT( (X1-X2)**2 + (Y1-Y2)**2 )
C
C--- TRANSFORMER MODULI.
C
      TMP = 1/L
      R(1) = TMP*(X1-X2)
      R(2) = TMP*(Y1-Y2)
      R(3) = -TMP*(X1-X2)
      R(4) = -TMP*(Y1-Y2)
C
C--- CONNECTOR VELOCITY.
C
      VCN(MTFDAT(TFN,8)) =R(1)*VX1 + R(2)*VY1 + R(3)*VX2 + R(4)*VY2
C
C--- STORE THE TRANSFORMER VALUES.
C
      MTFVAL( TFN ) = R( MTFDAT( TFN,6 ) )
C
C--- STORE THE TRANSFORMERS BY POINTERS.
C
      DO 100 I=1,MTFDAT(TFN,7)-1
C
C-      INTF - TF NUMBER OF POINTER I.
C
        INTF = MTFDAT( TFN,I+8 )
        MTFVAL(INTF) = R( MTFDAT( INTF,6 ) )
100   CONTINUE
C                                                                           C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      RETURN
      END
```

# APPENDIX C

# SOURCE CODE FOR BILDBG

## APPENDIX C

## SOURCE CODE FOR BILDBG

Subroutines in this appendix appear in the following sequence:

| | |
|---|---|
| 1. BILDBG | 12. GETEPS |
| 2. RESTOR | 13. CNTFEP |
| 3. REORDR | 14. BNDNOD |
| 4. INITBG | 15. CNMP3 |
| 5. RESULT | 16. CNMP4 |
| 6. CONECT | 17. CNMP5 |
| 7. GETMP | 18. CNMP6 |
| 8. GETMOT | 19. CNMP7 |
| 9. GETXYT | 20. BGOTF1 |
| 10. GETGEM | 21. BGEPTT |
| 11. BG | 22. BGEPMP |

```
      PROGRAM BILDBG
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CDCCCCC              PROGRAM    DESCRIPTION                      CCCCC
C                                                                     C
C--- THIS PROGRAM CREATES THE LAGRANGIAN BOND GRAPH                   C
C    OF A GRAPHICS FILE CREATED ON THE PS300                          C
C    USING THE GRAPHIC DESIGN PROGRAM.                                C
C                                                                     C
C--- PROGRAMMER: JOHN D. REID          DATE: FALL 1983                C
C                                                                     C
C--- SUBROUTINES CALLED:   RESTOR   REORDR                            C
C                          INITBG   CONECT                            C
C                          GETMOT   GETGEM                            C
C                              BG   RESULT                            C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CVCCCCCCCC           VARIABLE IDENTIFICATION                  CCCCCCCC
C                                                                     C
 INSERT COMFIL
 INSERT INPTBK
 INSERT COMBG
 INSERT SYBGBK
C
      CHARACTER*6 METHOD
      MARGIN=80
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CICCCCCCCC           INITIALIZATION BLOCK                   BLOCK 0CCC
C                                                                     C
C--- TALK TO THE USER.
C
      WRITE(*,900)
      WRITE(*,910)
      WRITE(*,920)
C
C--- READ IN THE GRAPHICAL DATA.
C
      CALL RESTOR(NOGO)
C
C--- IF NOGO=1 EXIT PROGRAM.
C
      IF(NOGO.EQ.1) GOTO 799
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CPCCCCCCCC           PROCESS BLOCK                          BLOCK 0CCC
C                                                                     C
C--- SORT THE ITEM LIST (ITMLBL) INTO THE MOTION POINT LIST
C    (MPNAM) AND THE CONNECTOR LIST (CNNAM).
C
      CALL REORDR
```

```
C
C--- INITIALIZE VARIABLES, THESE VARIABLES DEPEND UPON THE
C    NUMBER OF ITEMS FOUND IN RESTOR AND REORDR ABOVE.
C
      CALL INITBG
C
C--- GET THE CONNECTION NETWORK ( MATRIX ) OF THE
C    SYSTEM.
C
C-- THERE ARE TWO METHODS:
C
C     1. SUBROUTINE CONALT  -  GETS THE MOTION POINTS
C                              ATTACHED TO THE CONNECTORS.
C
C        THE CONALT METHOD IS NOT USED AT THIS TIME
C        BECAUSE IT DOES NOT HANDLE MOTION POINTS
C        CONNECTED TO MOTION POINTS.
C
C
C     2. SUBROUTINE CONECT  -  GETS THE CONNECTORS ATTACHED
C                              TO THE MOTION POINTS.
C
      METHOD='CONECT'
      CALL CONECT
C
C--- GET THE ALLOWABLE MOTION FOR EACH MOTION POINT.
C
      CALL GETMOT
C
C--- GET THE LOCAL GEOMETRY FOR THE MASSES AND BARS.
C    (i.e. DISTANCE FROM MP TO THE END POINT OF ITS CONNECTORS.)
C
      CALL GETGEM
C
C--- BUILD THE BOND GRAPH.
C
      CALL BG
C
C---------- WRITE RESULTS TO A FILE.
C
      CALL RESULT
C
C--- EXIT PROGRAM
C
799   WRITE(*,930)
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
CFCCCCCCCC              FORMAT STATEMENTS                  BLOCK 9000
C                                                                    C
C                                                                    C
900   FORMAT(///,2(/,6X,50('*')),2(/,6X,'**',46X,'**'))
910   FORMAT(6X,'**',10X,'EVANS AND SUTHERLAND PS300',10X,'**',/,
     + 6X,'**',46X,'**',/,6X,'**',12X,'  BOND GRAPH PROGRAM  ',
     + 12X,'**',2(/,6X,'**',46X,'**'))
```

```
920   FORMAT(6X,'**',17X,'DEVELOPED BY',17X,'**',/,
     +    6X,'**',46X,'**',/,
     +    6X,'**',18X,'JOHN REID',19X,'**',/,
     +    6X,'**',46X,'**',/,
     +    6X,'**',18X,'FALL 1983',19X,'**',
     +    2(/,6X,'**',46X,'**'),
     +    2(/,6X,50('*')))
930   FORMAT(////,'*****  YOU ARE NOW LEAVING',
     +        ' BILDBG.  *****',////)
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
      CALL EXIT
      END
      SUBROUTINE RESTOR
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
CDCCCCC                SUBROUTINE DESCRIPTION                   CCCCC
C                                                                    C
C--- THIS SUBROUTINE RESTORES THE GRAPHICAL DATA                     C
C    FROM THE FILE SPECIFIED.                                        C
C                                                                    C
C--- NOTE:  THE FILE MUST FIRST BE CREATED BY THE GRAPHIC            C
C           DESIGN PROGRAM ON THE PS300.                             C
C                                                                    C
C--- PROGRAMMER: JOHN D. REID          DATE: FALL 1983               C
C                                                                    C
C--- CALLED FROM BILDBG.                                             C
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
CVCCCCCCCC             VARIABLE IDENTIFICATION               CCCCCCCC
C                                                                    C
 INSERT COMFIL
C                                                                    C
      EXTERNAL NCHARS
C                                                                    C
C--- IF NOGO = 1   EXIT PROGRAM UPON RETURNING TO BILDBG.            C
C                                                                    C
      NOGO=0
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
CPCCCCCCCC             PROCESS BLOCK                       BLOCK 0CCC
C                                                                    C
C--- INPUT OLD FILENAME.
C
100   WRITE(*,910)
      READ(*,920) FNAME
C
C--- OPEN OLD FILENAME
C
      OPEN(UNIT=12,FILE=FNAME,STATUS='OLD',ERR=800)
C
```

```
C--- READ IN NUMBER OF ITEMS
C
      READ(12,*) NITEM
C
C--- READ IN THE NITEM'S
C
      DO 200 NCITEM=1,NITEM
        READ(12,930) ITMLST(NCITEM)
        READ(12,940) ITMTYP(NCITEM)
        READ(12,*) TRANSX(NCITEM),TRANSY(NCITEM),ROTZ(NCITEM),
     +           SCL(NCITEM),SCLX(NCITEM),SCLY(NCITEM)
        READ(12,950) ITMLBL(NCITEM),LBLPOS(NCITEM,1),
     +               LBLPOS(NCITEM,2)
200   CONTINUE
C
C--- READ IN ORIGIN TRANSFORMATIONS
C
      READ(12,*) TRANSX(39),TRANSY(39),SCL(39)
C
C--- READ IN NUMBER OF EACH ITEM (ie. SPRINGS,MASSES...)
C
      READ(12,*) NUMG,NUMB,NUMS,NUMD
      READ(12,*) NUMP,NUMM,NUMR,NUMC
      READ(12,*) NUMF,NUML,NUMV
      CLOSE(UNIT=12)
      OLDFIL=.TRUE.
      FILE=.TRUE.
      GOTO 999
C                                                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                  C
CERCCCCCCC           ERROR HANDLING                       BLOCK 0800
C                                                                  C
C--- ERROR - OLDFILE DOES NOT EXIST.                               C
C                                                                  C
800   CALL DBLANK(FNAME)
      WRITE(*,960) FNAME(1:NCHARS(FNAME))
      READ(*,970) IANS
      IF(IANS.EQ.'Y') THEN
        GOTO 100
      ELSEIF(IANS.EQ.'N') THEN
        NOGO=1
      ELSE
        WRITE(*,980)
        GOTO 800
      ENDIF
C                                                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                  C
CFCCCCCCCC           FORMAT STATEMENTS                    BLOCK 9000
C                                                                  C
910   FORMAT(//,'INPUT THE NAME OF THE GRAPHIC FILE ',
     +   'TO BE PROCESSED : ',)
920   FORMAT(A12)
930   FORMAT(A5)
```

```
940   FORMAT(A6)
950   FORMAT(3X,A4,4X,F7.3,4X,F7.3)
960   FORMAT(/,A,' DOES NOT EXIST, CAN NOT RESTORE.',
    +      //,'THE FILE MUST FIRST BE CREATED USING THE GRAPHIC',
    +      /,'DESIGN PROGRAM (DESIGN).',//,'WOULD YOU LIKE TO',
    +   ' TRY AGAIN?  (Y OR N) : ',)
970   FORMAT(A1)
980   FORMAT(//,'I DO NOT UNDERSTAND.',//)
C                                                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                  C
999   RETURN
      END
      SUBROUTINE REORDR
C                                                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                  C
CDCCCCC               SUBROUTINE DESCRIPTION                   CCCCC
C                                                                  C
C--- THIS SUBROUTINE RE-ORDERS THE ITMLBL LIST                     C
C    TO A DESIRED ORDER.  WHICH IS:                                C
C                                                                  C
C       M, B, J, G, P, S, D, F, V, R                               C
C                                                                  C
C    PNTLBL IS A POINTER BETWEEN THE NEW ORDER (NEWLBL)            C
C    AND THE OLD ORDER (ITMLBL).                                   C
C                                                                  C
C--- THE MOTION POINT VECTOR ( MPNAM ) AND THE                     C
C    CONNECTION VECTOR ( CNNAM ) ARE ALSO FOUND.                   C
C                                                                  C
C     MOTION POINTS ARE : M, B, J, G AND P                         C
C                                                                  C
C     CONNECTORS ARE    : S, D, F, V AND R                         C
C                                                                  C
C--- PROGRAMMER: JOHN D. REID          DATE: FALL 1983             C
C                                                                  C
C--- CALLED FROM BILDBG.                                           C
C                                                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                  C
CVCCCCCCCC            VARIABLE IDENTIFICATION                CCCCCCCC
C                                                                  C
 INSERT COMFIL
 INSERT COMBG
C                                                                  C
C--- LOCAL VARIABLES.                                              C
C                                                                  C
      CHARACTER*1 LBLLET(10)
      DATA LBLLET/'M','B','J','G','P','S','D','F','V','R'/
      ITEMP=0
C                                                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                  C
CPCCCCCCCC            PROCESS BLOCK                        BLOCK 0CCC
C                                                                  C
```

```
C--- BEGIN LOOP TO PUT IN ORDER OF LBLLET.
C
      DO 200 I=1,10
C
C--- FIND THE NUMBER OF MOTION POINTS ( NMP )
C--- AND THE NUMBER OF M.P.'S WITH MOTION ( NMPWM ).
C
        IF(I.EQ.6) THEN
          NMP=ITEMP
        ELSEIF(I.EQ.4) THEN
          NMPWM=ITEMP
        ENDIF
C
C--- INNER LOOP TO FIND THE ITMLBL'S THAT CORRESPOND
C    TO LBLLET ( STORE IN NEWLBL ).
C
        DO 100 J=1,NITEM
          IF(ITMLBL(J)(1:1).EQ.LBLLET(I)) THEN
            ITEMP=ITEMP+1
            NEWLBL(ITEMP)=ITMLBL(J)
            PNTLBL(ITEMP)=J
C
C-          GET THE CONNECTOR TYPE.
C
            IF(I.EQ.6) THEN
              NCN=NCN+1
              CNTYP(NCN)=1
            ELSEIF(I.EQ.7.OR.I.EQ.10) THEN
              NCN=NCN+1
              CNTYP(NCN)=3
            ELSEIF(I.EQ.8) THEN
              NCN=NCN+1
              CNTYP(NCN)=4
            ELSEIF(I.EQ.9) THEN
              NCN=NCN+1
              CNTYP(NCN)=5
            ENDIF
          ENDIF
100     CONTINUE
200   CONTINUE
C
C--- CALCULATE THE NUMBER OF CONNECTIONS ( NCN )
C
      NCN=NITEM-NMP
C
C--- FIND THE MOTION POINT VECTOR ( MPNAM ) AND THE
C    CONNECTION VECTOR ( CNNAM ).
C
      DO 250 I=1,NITEM
C
C--- MOTION POINTS.
C
        IF(I.LE.NMP) THEN
          MPNAM(I)=NEWLBL(I)
C
```

```
C--- CONNECTIONS.
C
        ELSE
          J=I-NMP
          CNNAM(J)=NEWLBL(I)
        ENDIF
250   CONTINUE
C                                                                 C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                 C
      RETURN
      END
      SUBROUTINE INITBG
C                                                                 C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                 C
CDCCCCC              SUBROUTINE DESCRIPTION                   CCCCC
C                                                                 C
C                                                                 C
C--- THIS SUBROUTINE INITIALIZES THE VARIABLES FOR BILDBG.        C
C                                                                 C
C--- PROGRAMMER: JOHN D. REID          DATE: FALL 1983            C
C                                                                 C
C--- CALLED FROM BILDBG.                                          C
C                                                                 C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                 C
CVCCCCCCCCC          VARIABLE IDENTIFICATION               CCCCCCCC
C                                                                 C
 INSERT COMBG
 INSERT SYBGBK
C                                                                 C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                 C
CPCCCCCCCCC          PROCESS BLOCK                         BLOCK 0CCC
C                                                                 C
      DO 100 I=1,NMP
        MOTTYP(I)=0
        MP1PTR(I)=0
        CPTR(I)=0
100   CONTINUE
      IT1=I+1
      CPTR(IT1)=0
      DO 150 I=1,NMP*3
        IXMPPT(I)=0
        XMP(I)=0
        VMP(I)=0
150   CONTINUE
      DO 200 I=1,MAXNCN
        IEPTYP(I)=0
        MPONCN(I,1)=0
        MPONCN(I,2)=0
        XCN(I)=0
        VCN(I)=0
        IXCNPT(I)=0
```

```
200   CONTINUE
      DO 250 I=1,MAXNCN*2
        CNONMP(I)=0
        DO 240 K=1,MAXNMS
          GEOMAS(K,I,1)=0
          GEOMAS(K,I,2)=0
          GEOBAR(K,I)=0
240     CONTINUE
        IT1=I*2
        IT2=IT1-1
        IXEPPT(IT1)=0
        IXEPPT(IT2)=0
        XEP(IT1)=0
        XEP(IT2)=0
        VEP(IT1)=0
        VEP(IT2)=0
250   CONTINUE
      DO 350 I=1,MAXNTF
        DO 340 K=1,11
          MTFDAT(I,K)=0
340     CONTINUE
350   CONTINUE
C
C--- INITIALIZE THE NMOT ARRAY.
C
      NMOT(0)=0
      NMOT(1)=1
      NMOT(2)=1
      NMOT(3)=2
      NMOT(4)=1
      NMOT(5)=2
      NMOT(6)=2
      NMOT(7)=3
C
      DO 400 I=1,MNEL+1
        NPTR(I)=I
400   CONTINUE
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
      RETURN
      END
```

```
      SUBROUTINE RESULT
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CDCCCCC               SUBROUTINE DESCRIPTION                      CCCCC
C                                                                     C
C---------- WRITES RESULTS TO A FILE.                                 C
C                                                                     C
C--- PROGRAMMER: JOHN D. REID          DATE: FALL 1983                C
C                                                                     C
C--- CALLED FROM BILDBG.                                              C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CVCCCCCCCC            VARIABLE IDENTIFICATION                  CCCCCCCC
C                                                                     C
 INSERT COMBG
 INSERT COMFIL
 INSERT SYBGBK
C                                                                     C
      CHARACTER*15 FILNAM
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CPCCCCCCCC            PROCESS BLOCK                          BLOCK OCCC
C                                                                     C
      WRITE(*,'(1X)')
      CALL PROMPT('INSERT NAME OF FILE FOR STORAGE :')
      READ(*,900) FILNAM
      WRITE(*,'(1X)')
      OPEN(UNIT=12,FILE=FILNAM)
      WRITE(12,'(1X)')
      WRITE(12,910) FNAME
      WRITE(12,'(//)')
C
C--- WRITE THE RE-ORDER LISTS (FROM SUB. REORDR).
C
      DO 50 I=1,NITEM
        WRITE(12,920) ITMLBL(I),NEWLBL(I)
50    CONTINUE
C
C--- THE MPNAM LIST
C
      WRITE(12,'(//)')
      WRITE(12,*) 'THE MPNAM LIST'
      WRITE(12,'(1X)')
      DO 100 I=1,NMP
        WRITE(12,930) I,MPNAM(I),I,MOTTYP(I)
100   CONTINUE
      WRITE(12,'(//)')
C
C--- THE CNNAM LIST.
C
      WRITE(12,*) 'THE CNNAM LIST'
      WRITE(12,'(1X)')
```

```
      DO 150 I=1,NCN
        WRITE(12,940) I,CNNAM(I),I,CNTYP(I),I,IEPTYP(I)
150   CONTINUE
      WRITE(12,'(//)')
C
C--- WRITE OUT MPONCN.
C
      WRITE(12,*) 'MOTION POINTS ON CONNECTORS :'
      WRITE(12,'(1X)')
      DO 200 I=1,NCN
        WRITE(12,950) I,MPONCN(I,1),I,MPONCN(I,2)
200   CONTINUE
C
C-  WRITE OUT CNONMP
C
      WRITE(12,'(//)')
      WRITE(12,*) 'THE CPTR LIST'
      WRITE(12,'(1X)')
      WRITE(12,990) (CPTR(I),I=1,NMP+1)
      WRITE(12,'(//)')
      WRITE(12,*) 'THE CNONMP ARRAY'
      WRITE(12,'(1X)')
      WRITE(12,990) (CNONMP(I),I=1,2*NCN)
      WRITE(12,'(//)')
C
C--- LOCAL GEOMETRY FOR THE MASSES.
C
      WRITE(12,'(//)')
      WRITE(12,*) 'LOCAL GEOMETRY FOR THE MASSES.'
      DO 600 I=1,NMS
        IBEGIN=CPTR(I)
        IEND=CPTR(I+1)-1
        WRITE(12,975) MPNAM(I)
        DO 550 J=IBEGIN,IEND
          ITEMP=CNONMP(J)
          WRITE(12,980) CNNAM(ITEMP),GEOMAS(I,ITEMP,1),
     +                  GEOMAS(I,ITEMP,2)
550     CONTINUE
600   CONTINUE
C
C--- LOCAL GEOMETRY FOR THE BARS.
C
      WRITE(12,'(//)')
      WRITE(12,*) 'LOCAL GEOMETRY FOR THE BARS.'
      DO 700 I=1,NBR
        IBEGIN=CPTR(I+NMS)
        IEND=CPTR(I+NMS+1)-1
        WRITE(12,975) MPNAM(I+NMS)
        DO 650 J=IBEGIN,IEND
          ITEMP=CNONMP(J)
          WRITE(12,985) CNNAM(ITEMP),GEOBAR(I,ITEMP)
650     CONTINUE
700   CONTINUE
C
C--- THE BOND GRAPH
```

```
C
      WRITE(12,'(//)')
      WRITE(12,*) 'THE BOND GRAPH.'
      WRITE(12,'(1X)')
      DO 750 I=1,NBD
        WRITE(12,9000) I,IBMX(I,1),IBMX(I,2)
750   CONTINUE
      WRITE(12,'(1X)')
      DO 800 I=1,NEL
        WRITE(12,9010) I,IELLST(I)
800   CONTINUE
C
C-  WRITE OUT NBIMX.
C
      WRITE(12,'(//)')
      WRITE(12,*) 'THE NPTR LIST'
      WRITE(12,'(1X)')
      WRITE(12,990) (NPTR(I),I=1,NEL+1)
      WRITE(12,'(//)')
      WRITE(12,*) 'THE NBIMX ARRAY'
      WRITE(12,'(1X)')
      WRITE(12,990) (NBIMX(I),I=1,2*NBD+1)
C
C--- POINTER LIST FROM MP 1-JUNCTION NODES TO XMP.
C
      WRITE(12,'(//)')
      WRITE(12,*) 'POINTER FROM MP NODES TO XMP.'
      WRITE(12,'(1X)')
      DO 810 I=1,100
        IF(IXMPPT(I).EQ.0) GOTO 820
        WRITE(12,9020) I,IXMPPT(I)
810   CONTINUE
820   CONTINUE
C
C--- POINTER LIST EP 1-JUNCTION'S TO XEP.
C
      WRITE(12,'(//)')
      WRITE(12,*) 'POINTER FROM EP XMP TO NODE NUMBER'
      WRITE(12,'(1X)')
      DO 830 I=1,NCN*4
        WRITE(12,9030) I,I,IXEPPT(I)
830   CONTINUE
C
C--- POINTER LIST FROM CONNECTOR TO ITS NODE.
C
      WRITE(12,'(//)')
      WRITE(12,*) 'POINTER FROM CONNECTOR TO ITS NODE NUMBER'
      WRITE(12,'(1X)')
      DO 840 I=1,NCN
        WRITE(12,9040) I,I,IXCNPT(I)
840   CONTINUE
C
C--- TRANSFORMERS AND DATA.
C
      WRITE(12,'(//)')
```

```
      WRITE(12,*) 'TRANSFORMERS AND ASSOCIATED DATA.'
      WRITE(12,'(1X)')
      DO 850 I=1,NTF
        WRITE(12,9050) I,MTF(I),MTFDAT(I,1)
        IF(MTFDAT(I,1).EQ.5) THEN
          WRITE(12,9052) MTFDAT(I,2),MTFDAT(I,3),
     +                   MTFDAT(I,4),MTFDAT(I,5),MTFDAT(I,6)
          WRITE(12,9054) MTFDAT(I,7),MTFDAT(I,8),MTFDAT(I,9),
     +                   MTFDAT(I,10),MTFDAT(I,11)
        ELSE
          IF(MTFDAT(I,1).EQ.1.OR.MTFDAT(I,1).EQ.2) THEN
            WRITE(12,9060) MTFDAT(I,2),MTFDAT(I,3),MTFDAT(I,4)
          ELSE
            WRITE(12,9062) MTFDAT(I,2),MTFDAT(I,3)
          ENDIF
        ENDIF
850   CONTINUE
      CLOSE(UNIT=12)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
CFCCCCCCCC           FORMAT STATEMENTS                     BLOCK 9000
900   FORMAT(A15)
910   FORMAT('FILE = ',A12)
920   FORMAT(' ITMLBL : ',A6,'     NEWLBL : ',A6)
930   FORMAT('MPNAM(',I2,') = ',A3,6X,
     +       'MOTTYP(',I2,') = ',I2)
940   FORMAT('CNNAM(',I2,') = ',A3,5X,'CNTYP(',I2,') = ',I2,
     +        5X,'IEPTYP(',I2,') = ',I2)
950   FORMAT('MPONCN(',I2,',1) = ',I2,'          ',
     +        'MPONCN(',I2,',2) = ',I2)
975   FORMAT(//,'MOTION POINT : ',A3)
980   FORMAT(/,'  CONNECTOR : ',A3,3X,'X-DISTANCE = ',F6.2,
     +       4X,'Y-DISTANCE = ',F6.2)
985   FORMAT(/,'  CONNECTOR : ',A3,3X,'DISTANCE = ',F6.2)
990   FORMAT(10(4X,I3))
9000  FORMAT('NODES ATTACHED TO BOND ',I3,' ARE ',
     +        3X,I3,5X,I3)
9010  FORMAT('IELLST(',I3,') = ',I3)
9020  FORMAT('IXMPPT(',I2,') = ',I3)
9030  FORMAT('NODE ASSOCIATED WITH EP ',I2,' = IXEPPT(',I2,
     +       ') = ',I3)
9040  FORMAT('NODE NUMBER FOR CONNECTOR ',I2,
     +       ' = IXCNPT(',I2,') = ',I3)
9050  FORMAT(/,'MTF(',I2,') = ',I3,6X,'FUNCTION NUMBER = ',F2.0)
9052  FORMAT('    ARGUMENTS : ',F3.0,3X,F3.0,3X,F3.0,3X,F3.0,/,
     +       '    MODULATOR FUNCTION : ',F2.0)
9054  FORMAT('    NUMBER OF POINTERS = ',F2.0,/,
     +       5X,'CONNECTOR POINTER = ',F3.0,/,
     +       5X,'MTF POINTERS      : ',F4.0,3X,F4.0,3X,F4.0)
9060  FORMAT('    CONSTANTS : ',F3.0,3X,F3.0,
     +       4X,'ARGUMENT : ',F3.0)
9062  FORMAT('    CONSTANT : ',F3.0,4X,'ARGUMENT : ',F3.0)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
      RETURN
      END
```

```
      SUBROUTINE CONECT
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CDCCCCC              SUBROUTINE DESCRIPTION                       CCCCC
C                                                                     C
C--- THIS SUBROUTINE GETS THE CONNECTION NETWORK OF THE               C
C    SYSTEM USING THE CONNECTORS ON MOTION POINT METHOD.              C
C                                                                     C
C--- PROGRAMMER: JOHN D. REID         DATE: FALL 1983                 C
C                                                                     C
C--- CALLED FROM BILDBG.                                              C
C                                                                     C
C--- SUBROUTINES CALLED:   GETMP                                      C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CVCCCCCCCC           VARIABLE IDENTIFICATION                   CCCCCCCC
C                                                                     C
 INSERT COMBG
C                                                                     C
C--- LOCAL VARIABLES                                                  C
C                                                                     C
      CHARACTER*1 TEMP
      CHARACTER*4 TMPWRD(15)
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CPCCCCCCCC           PROCESS BLOCK                           BLOCK OCCC
C                                                                     C
C--- LOOP THROUGH THE MOTION POINTS ( NMP )
C    TO FIND THE CONNECTORS THAT ARE ATTACHED TO THEM.
C
      CPTR(1)=1
      NRC=0
      DO 300 I=1,NMP
50      NCNPTS=0
        NWRDS=0
        II=I+1
C
C--     ASK THE USER FOR THE ITEMS ATTACHED
C--     TO THE MOTION POINT MPNAM(I).
C
        WRITE(*,900) MPNAM(I)
         NEWLIN=.TRUE.
100      WRD='DEFAULT'
        CALL GETWD(WRD,NEWLIN,ENDLIN)
C
C-      SORT THE INPUT AND STORE IN TMPWRD.
C
        IF(WRD.EQ.'DEFAULT') THEN
C
C-        BAD INPUT.
C
          WRITE(*,910)
```

```
          GOTO 50
        ELSE
C
C-        INPUT IS READABLE.
C
          NWRDS=NWRDS+1
          TMPWRD(NWRDS)=WRD
        ENDIF
C
C--     CHECK FOR END OF INPUT FOR MOTION POINT I.
C
        IF(.NOT.ENDLIN) THEN
          NEWLIN=.FALSE.
          GOTO 100
        ENDIF
C
C---    ASK USER IF THE INPUT IS CORRECT.
C
110     WRITE(*,915) MPNAM(I)
        WRITE(*,916) (TMPWRD(JKL),JKL=1,NWRDS)
        WRITE(*,917)
        READ(*,925) ANS
        IF(ANS.EQ.'N') THEN
          GOTO 50
        ELSEIF(ANS.NE.'Y') THEN
          GOTO 110
        ENDIF
C
C--     LOOP THROUGH TMPWRD IN ORDER TO
C--     CREATE MPONCN AND CNONMP.
C
        DO 200 IWRDN=1,NWRDS
          DO 120 J=1,NCN
            IF(TMPWRD(IWRDN).EQ.CNNAM(J)) THEN
C
C-            CNNAM(J) IS CONNECTED TO MPNAM(I)
C
              ITEMP=CPTR(I)+NCNPTS
              CNONMP(ITEMP)=J
C
C-              FIND THE MPONCN ARRAY AT THE SAME TIME.
C
              CALL GETMP(NCNPTS,TMPWRD,IWRDN)
              GOTO 200
            ENDIF
120       CONTINUE
C
C--       TMPWRD(IWRDN) IS NOT A CONNECTOR.  IS IT A MOTION
C--       POINT OR NOT AN ITEM AT ALL?
C
          DO 140 J=1,NMP
C
C-          INPUT IS A MOTION POINT
C
            IF(TMPWRD(IWRDN).EQ.MPNAM(J)) THEN
```

```
C
C-            ITEM IS CONNECTED TO ITSELF.
C
              IF(TMPWRD(IWRDN).EQ.MPNAM(I)) THEN
              ELSE
C
C-              A RIGID CONNECTOR IS REQUIRED BETWEEN
C               MPNAM(I) AND MPNAM(J) (= TMPWRD(IWRDN)).
C
C-              CHECK TO SEE IF A RIGID CONNECTOR HAS
C               ALREADY BEEN CREATED.
C
                DO 130 K=1,NRC
                  ITEMP=NCN+K
                  IF(MPONCN(ITEMP,1).EQ.I.AND.
     +               MPONCN(ITEMP,2).EQ.J) THEN
                    GOTO 135
                  ENDIF
130             CONTINUE
C
C-              CREATE A RIGID CONNECTOR.
C
                NRC=NRC+1
                ITEMP=NCN+NRC
                WRITE(TEMP,'(I1)') NRC
                CNNAM(ITEMP)='X'//TEMP
C
C-              STORE THE CONNECTOR AS A SPRING TYPE ELEMENT.
C
                CNTYP(ITEMP)=1
C
C-              A RIGID CONNECTOR HAS NOW BEEN MADE - CNNAM(ITEMP)
C               TELL THE USER ABOUT IT.
C
                WRITE(*,920) CNNAM(ITEMP),MPNAM(I),MPNAM(J)
                READ(*,925) ANS
C
C-              UPDATE MPONCN.
C
                MPONCN(ITEMP,1)=J
                MPONCN(ITEMP,2)=I
C
C-              UPDATE CNONMP.
C
135             ITEMP2=CPTR(I)+NCNPTS
                CNONMP(ITEMP2)=ITEMP
                NCNPTS=NCNPTS+1
              ENDIF
              GOTO 200
            ENDIF
140       CONTINUE
C
C--       INPUT IS NOT RECOGNIZED - IGNORE.
C
          WRITE(*,930) TMPWRD(IWRDN)
```

```
          READ(*,925) ANS
C
200     CONTINUE
C
C--     FIND CPTR FOR THE NEXT MOTION POINT (MPNAM(I))..
C
        CPTR(II)=CPTR(I)+NCNPTS
C
C--    LOOP FOR NEXT MOTION POINT.
C
300   CONTINUE
C
C--- THE NUMBER OF CONNECTORS HAS INCREASED BY NRC.
C
      NCN=NCN+NRC
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CFCCCCCCCC           FORMAT STATEMENTS                      BLOCK 9000
C                                                                     C
900   FORMAT(//,'WHICH ITEMS ARE CONNECTED TO ',A3,'? : ', )
910   FORMAT(//,'*** BAD INPUT.  TRY AGAIN. ***')
915   FORMAT(/,'AS I UNDERSTAND IT, THE FOLLOWING ITEMS ARE ',
     +       'CONNECTED TO ',A3,'.',/)
916   FORMAT(5(4X,A3))
917   FORMAT(/,'IS THIS CORRECT? (Y OR N) : ', )
920   FORMAT(//,'A RIGID CONNECTOR ',A2,' HAS BEEN CREATED',/,
     +       'BETWEEN ',A3,' AND ',A3,//,'HIT <RETURN> ',
     +        'TO CONTINUE.', )
925   FORMAT(A1)
930   FORMAT(//,'I DO NOT RECOGNIZE ',A32,/,
     +       'IT WILL BE IGNORED.',//,'HIT <RETURN>',
     +        ' TO CONTINUE.', )
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
      RETURN
      END
      SUBROUTINE GETMP(NCNPTS,TMPWRD,IWRDN)
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CDCCCCC              SUBROUTINE DESCRIPTION                       CCCCC
C                                                                     C
C--- THIS SUBROUTINE IS USED TO FIND THE MOTION POINTS                C
C    ON CONNECTORS ARRAY ( MPONCN ).                                  C
C                                                                     C
C--- PROGRAMMER: JOHN D. REID          DATE: FALL 1983                C
C                                                                     C
C--- CALLED FROM CONECT.                                              C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CVCCCCCCCC           VARIABLE IDENTIFICATION                   CCCCCCCC
C                                                                     C
```

```
C     AT THIS POINT WE KNOW:                                          C
C                                                                     C
C      1. MOTION POINT IS MPNAM(I).                                   C
C      2. CONNECTOR IS CNNAM(J)                                       C
C                                                                     C
 INSERT COMBG
C                                                                     C
      CHARACTER*4 TMPWRD(15)
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CPCCCCCCCC             PROCESS BLOCK                        BLOCK 0CCC
C                                                                     C
C--- THIS IS THE FIRST MP ON THIS CONNECTOR.
C
      IF(MPONCN(J,1).EQ.0) THEN
        NCNPTS=NCNPTS+1
        MPONCN(J,1)=I
C
C--- THIS IS THE SECOND MP ON THIS CONNETOR.
C
      ELSEIF(MPONCN(J,2).EQ.0) THEN
        NCNPTS=NCNPTS+1
        MPONCN(J,2)=I
C
C--- THIS IS THE THIRD MP ON THIS CONNECTOR.
C
C-     ***  ERROR.  CAN NOT HAVE MORE THAN 2.  ***
C
      ELSE
        WRITE(*,900) TMPWRD(IWRDN)
        READ(*,910) ANS
      ENDIF
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CFCCCCCCCC             FORMAT STATEMENTS                    BLOCK 9000
C                                                                     C
900   FORMAT(//,'ALREADY HAVE 2 MOTION POINTS ATTACHED',
     +       ' TO ',A3,'.',/,'IT WILL BE IGNORED.',//,
     +       'HIT <RETURN> TO CONTINUE.', )
910   FORMAT(A1)
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
      RETURN
      END
      SUBROUTINE GETMOT
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CDCCCCC                SUBROUTINE DESCRIPTION                     CCCCC
C                                                                     C
C--- THIS SUBROUTINE GETS THE ALLOWABLE MOTIONS (X,Y, OR THETA)      C
C    FOR EACH MOTION POINT.                                           C
```

```
C                                                                 C
C--- PROGRAMMER: JOHN D. REID          DATE: FALL 1983           C
C                                                                 C
C--- CALLED FROM BILDBG.                                          C
C                                                                 C
C--- SUBROUTINES CALLED:   GETXYT                                 C
C                                                                 C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                 C
CVCCCCCCCC           VARIABLE IDENTIFICATION               CCCCCCCC
C                                                                 C
 INSERT COMBG
C
      RLO=-1.E25
      RHI=1.E25
C                                                                 C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                 C
CPCCCCCCCC           PROCESS BLOCK                         BLOCK OCCC
C                                                                 C
C--- LOOP THROUGH THE MOTION POINTS TO FIND MOTIONS.
C
      DO 300 I=1,NMPWM
C
C--     MOTION POINT IS A MASS
C
        IF(MPNAM(I)(1:1).EQ.'M') THEN
          NMS=NMS+1
C
C-        GET X,Y AND/OR THETA ALLOWABLE MOTIONS.
C
          CALL GETXYT
C
C--     MOTION POINT IS A BAR.
C
        ELSEIF(MPNAM(I)(1:1).EQ.'B') THEN
          NBR=NBR+1
C
C-        GET X,Y AND/OR THETA ALLOWABLE MOTIONS.
C
          CALL GETXYT
C
C--     MOTION POINT IS A JOINT.
C
        ELSEIF(MPNAM(I)(1:1).EQ.'J') THEN
          CALL GETXYT
        ENDIF
300   CONTINUE
C                                                                 C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                 C
      RETURN
      END
      SUBROUTINE GETXYT
C                                                                 C
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CDCCCCC              SUBROUTINE DESCRIPTION                       CCCCC
C                                                                     C
C--- THIS SUBROUTINE FINDS OUT IF THERE IS MOTION                      C
C    IN THE X,Y AND/OR THETA DIRECTION FOR MPNAM(I).                   C
C                                                                     C
C--- PROGRAMMER: JOHN D. REID         DATE: FALL 1983                 C
C                                                                     C
C--- CALLED FROM GETMOT.                                              C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CVCCCCCCCC           VARIABLE IDENTIFICATION                   CCCCCCCC
C                                                                     C
 INSERT COMBG
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CPCCCCCCCC           PROCESS BLOCK                             BLOCK 0CCC
C                                                                     C
C--- JOINTS CAN ONLY MOVE IN X AND Y.
C
50     IF(MPNAM(I)(1:1).EQ.'J') THEN
         WRITE(*,905) MPNAM(I)
       ELSE
         WRITE(*,900) MPNAM(I)
       ENDIF
       NEWLIN=.TRUE.
100    WRD='DEFAULT'
       CALL GETWD(WRD,NEWLIN,ENDLIN)
C
C-     X-MOTION
C
       IF(WRD.EQ.'X') THEN
         MOTTYP(I)=MOTTYP(I)+1
C
C-     Y-MOTION
C
       ELSEIF(WRD.EQ.'Y') THEN
         MOTTYP(I)=MOTTYP(I)+2
C
C-     THETA-MOTION
C
       ELSEIF(WRD.EQ.'T'.OR.WRD.EQ.'THETA') THEN
C
C-       JOINTS CAN NOT MOVE IN THETA.
C
         IF(MPNAM(I)(1:1).EQ.'J') THEN
           WRITE(*,910)
           GOTO 50
         ENDIF
         MOTTYP(I)=MOTTYP(I)+4
C
C-     BAD INPUT
```

```
C
      ELSE
        WRITE(*,910)
        GOTO 50
      ENDIF
C
C--- LOOK FOR END OF INPUT.
C
      IF(.NOT.ENDLIN) THEN
        NEWLIN=.FALSE.
        GOTO 100
      ENDIF
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CFCCCCCCCC             FORMAT STATEMENTS                     BLOCK 9000
C                                                                     C
900   FORMAT(//,'WHICH MOTIONS ARE ALLOWABLE FOR ',A3,
     +       ' — X, Y AND/OR THETA : ', )
905   FORMAT(//,'WHICH MOTIONS ARE ALLOWABLE FOR ',A3,
     +       ' — X AND/OR Y : ', )
910   FORMAT(//,'!!!  BAD INPUT.  TRY AGAIN  !!!')
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
      RETURN
      END
      SUBROUTINE GETGEM
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CDCCCCC                SUBROUTINE DESCRIPTION                     CCCCC
C                                                                     C
C--- THIS SUBROUTINE GETS THE LOCAL GEOMETRY                          C
C    BETWEEN MOTION POINTS (MASSES AND BARS ONLY)                     C
C    AND THE END POINTS OF THE CONNECTORS THAT                        C
C    ARE ATTACHED TO THEM.                                            C
C                                                                     C
C--- PROGRAMMER: JOHN D. REID          DATE: FALL 1983                C
C                                                                     C
C--- CALLED FROM BILDBG.                                              C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CVCCCCCCCC             VARIABLE IDENTIFICATION                 CCCCCCCC
C                                                                     C
 INSERT COMBG
C                                                                     C
      RLO=-1.E25
      RHI=1.E25
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CPCCCCCCCC             PROCESS BLOCK                         BLOCK 0CCC
C                                                                     C
```

```
C--- LOCAL GEOMETRY FOR THE MASSES.
C
      DO 300 I=1,NMS
        IBEGIN=CPTR(I)
        IEND=CPTR(I+1)-1
        DO 200 J=IBEGIN,IEND
          ITEMP=CNONMP(J)
50        NINPUT=0
          WRITE(*,900) MPNAM(I)
          WRITE(*,910) MPNAM(I),CNNAM(ITEMP)
          NEWLIN=.TRUE.
100       RVAL=-1234.5
          CALL GETRL(RVAL,RLO,RHI,NEWLIN,ENDLIN)
C
C-        THERE ARE NO DEFAULT VALUES.
C
          IF(RVAL.EQ.-1234.5) THEN
            WRITE(*,920)
            GOTO 50
          ENDIF
C
C-        INPUT IS GOOD.
C
          NINPUT=NINPUT+1
C
C-        THERE MUST BE ONLY TWO INPUTS.
C
          IF(NINPUT.EQ.3) THEN
            WRITE(*,930)
            READ(*,940) ANS
            GOTO 200
          ENDIF
C
C-        STORE THE INPUT IN GEOMAS.
C
          GEOMAS(I,ITEMP,NINPUT)=RVAL
C
C-        CHECK FOR END OF INPUT.
C
          IF(.NOT.ENDLIN) THEN
            NEWLIN=.FALSE.
            GOTO 100
          ENDIF
C
C-        MUST HAVE TWO INPUTS.
C
          IF(NINPUT.LT.2) THEN
            WRITE(*,950)
            GOTO 50
          ENDIF
200     CONTINUE
300   CONTINUE
C
C--- LOCAL GEOMETRY FOR THE BARS.
C
```

```
      DO 600 I=1,NBR
        IBEGIN=CPTR(I+NMS)
        IEND=CPTR(I+NMS+1)-1
        DO 500 J=IBEGIN,IEND
          ITEMP=CNONMP(J)
350       NINPUT=0
          WRITE(*,960) MPNAM(I+NMS),MPNAM(I+NMS)
          WRITE(*,970) CNNAM(ITEMP)
          NEWLIN=.TRUE.
          RVAL=-1234.5
          CALL GETRL(RVAL,RLO,RHI,NEWLIN,ENDLIN)
C
C-        THERE ARE NO DEFAULT VALUES.
C
          IF(RVAL.EQ.-1234.5) THEN
            WRITE(*,920)
            GOTO 350
          ENDIF
C
C-        INPUT IS GOOD.
C
          GEOBAR(I,ITEMP)=RVAL
C
C-        ONLY ONE INPUT ALLOWED.
C
          IF(.NOT.ENDLIN) THEN
            WRITE(*,980)
            READ(*,940) ANS
          ENDIF
500     CONTINUE
600   CONTINUE
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CFCCCCCCCCC           FORMAT STATEMENTS                     BLOCK 9000
C                                                                     C
900   FORMAT(////,'LET THE CENTER OF ',A3,' BE THE POINT 0,0.')
910   FORMAT(/,'FROM THIS REFERENCE POINT WHAT IS THE ',
     +       'X,Y DISTANCE BETWEEN',/,A3,' AND THE END POINT',
     +       ' OF ',A3,' : ', )
920   FORMAT(//,'THERE ARE NO DEFAULT VALUES.  TRY AGAIN.')
930   FORMAT(//,'ONLY TWO INPUTS ALLOWED.  THE FIRST TWO WILL',
     +       ' BE THE ONES USED.',//,'HIT <RETURN> TO ',
     +       'CONTINUE. ', )
940   FORMAT(A1)
950   FORMAT(//,'YOU MUST ENTER TWO VALUES -- X,Y.  TRY AGAIN.')
960   FORMAT(////,'ASSUME ',A3,' IS PARALLEL TO THE X-AXIS.',/,
     +       'LET THE POINT WHERE ',A3,' ROTATES ABOUT BE 0,0.')
970   FORMAT(/,'FROM THIS REFERENCE POINT, WHAT IS THE ',
     +      'DISTANCE TO THE ENDPOINT OF ',A3,'.',/,
     +      '( INCLUDE + OR - SIGN ) : ', )
980   FORMAT(//,'ONLY ONE INPUT ALLOWED.  THE FIRST ',
     +       'INPUT WILL BE USED.',//,'HIT <RETURN> ',
     +       'TO CONTINUE. ', )
C                                                                     C
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
      RETURN
      END
      SUBROUTINE BG
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
CDCCCCC              SUBROUTINE DESCRIPTION                      CCCCC
C                                                                    C
C--- THIS SUBROUTINE BUILDS THE BOND GRAPH.                          C
C                                                                    C
C--- PROGRAMMER: JOHN D. REID          DATE: FALL 1983               C
C                                                                    C
C--- CALLED FROM BILDBG.                                             C
C                                                                    C
C--- SUBROUTINES CALLED:  GETEPS   CNTFEP                            C
C                          CNMP3    CNMP4                            C
C                          CNMP5    CNMP6                            C
C                          CNMP7   BNDNOD                            C
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
CVCCCCCCCC           VARIABLE IDENTIFICATION                  CCCCCCCC
C                                                                    C
 INSERT COMBG
 INSERT SYBGBK
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
CPCCCCCCCC           PROCESS BLOCK                          BLOCK 0CCC
C                                                                    C
C
C--- CREATE NODES FOR EACH MOTION POINT WITH MOTION.
C---     ( 1-JUNCTIONS )
C
      MP1PTR(1)=1
      DO 100 I=1,NMPWM
        DO 50 J=1,NMOT(MOTTYP(I))
          NEL=NEL+1
          IELLST(NEL)=7
          ITEMP=I+1
          MP1PTR(ITEMP)=NEL+1
50      CONTINUE
C
C-      FIND THE POINTER TO XMP AND VMP.
C
        IELN=MP1PTR(I)
        IF(MOTTYP(I).EQ.1) THEN
          IXMPPT(IELN)=I*3-2
        ELSEIF(MOTTYP(I).EQ.2) THEN
          IXMPPT(IELN)=I*3-1
        ELSEIF(MOTTYP(I).EQ.3) THEN
          IXMPPT(IELN)=I*3-2
          IELN=IELN+1
```

```
          IXMPPT(IELN)=I*3-1
        ELSEIF(MOTTYP(I).EQ.4) THEN
          IXMPPT(IELN)=I*3
        ELSEIF(MOTTYP(I).EQ.5) THEN
          IXMPPT(IELN)=I*3-2
          IELN=IELN+1
          IXMPPT(IELN)=I*3
        ELSEIF(MOTTYP(I).EQ.6) THEN
          IXMPPT(IELN)=I*3-1
          IELN=IELN+1
          IXMPPT(IELN)=I*3
        ELSEIF(MOTTYP(I).EQ.7) THEN
          IXMPPT(IELN)=I*3-2
          IELN=IELN+1
          IXMPPT(IELN)=I*3-1
          IELN=IELN+1
          IXMPPT(IELN)=I*3
        ENDIF
100   CONTINUE
C
C--- ADD THE INERTIA TO EACH MASS.
C
C-      INRTPT = POINTER TO FIRST INERTIA NODE.
C
      INRTPT=NEL+1
      DO 200 I=1,NMS
        DO 150 J=1,NMOT(MOTTYP(I))
          NEL=NEL+1
          NBD=NBD+1
          IELLST(NEL)=2
          IBMX(NBD,1)=MP1PTR(I)+J-1
          IBMX(NBD,2)=NEL
150     CONTINUE
200   CONTINUE
C
C-
C--- LOOK AT EACH CONNECTOR AND THE MOTION POINTS
C--- THAT ARE ATTACHED TO THEM.
C-
C
      DO 500 I=1,NCN
        NTFEL=0
C
C-      LOOK AT THE END POINTS OF CNNAM(I).
C
        CALL GETEPS
C
C--     NOTE: IF THE CONNECTOR IS A RIGID CONNECTOR THE BG
C--           COULD BE SIMPLIFIED FURTHER.  BUT, FOR NOW,
C--           WE WILL TREAT THE RIGID CON. LIKE A SPRING.
C
        IF(CNNAM(I)(1:1).EQ.'X') THEN
          IF(IEPTYP(I).LE.3) THEN
C
C-          NO NODES OR BONDS REQUIRED.
```

```
            GOTO 500
          ENDIF
        ENDIF
C
C--     ONE EP IS FIXED AND THE OTHER EP CAN MOVE IN THE
C--     X OR Y DIRECTION ONLY.
C
        IF(IEPTYP(I).EQ.1) THEN
C
C-        CREATE BG FROM CONNECTOR TO MP.
C
C           CONNECTOR NODE AND POINTER.
C
          NEL=NEL+1
          IELLST(NEL)=CNTYP(I)
          IXCNPT(I)=NEL
C
C           EP OF CONNECTOR 1-JUNCTION AND POINTER.
C
          NEL=NEL+1
          IELLST(NEL)=7
          IF(MOTTYP(MPONCN(I,1)).EQ.1) THEN
            IT1=I*4-3
          ELSEIF(MOTTYP(MPONCN(I,1)).EQ.2) THEN
            IT1=I*4-2
          ELSEIF(MOTTYP(MPONCN(I,2)).EQ.1) THEN
            IT1=I*4-1
          ELSEIF(MOTTYP(MPONCN(I,2)).EQ.2) THEN
            IT1=I*4
          ENDIF
          IXEPPT(IT1)=NEL
C
C           BOND BETWEEN THE CONNECTOR AND ITS EP.
C
          NBD=NBD+1
          IBMX(NBD,1)=NEL
          IBMX(NBD,2)=NEL-1
C
C           BOND BETWEEN END POINT AND MP 1-JUNCTIONS.
C
          NBD=NBD+1
          IF(MOTTYP(MPONCN(I,1)).NE.0) THEN
            IBMX(NBD,1)=MP1PTR(MPONCN(I,1))
          ELSE
            IBMX(NBD,1)=MP1PTR(MPONCN(I,2))
          ENDIF
          IBMX(NBD,2)=NEL
C
C-      THE BG BETWEEN CONNECTOR I AND ITS ATTACHED
C-      MOTION POINTS IS NOT SO SIMPLE.
C
        ELSE
C
C-        BUILD BG FROM CN TO 0-JUNCTION.
C
```

```
C           CONNECTOR NODE AND POINTER.
C
          NEL=NEL+1
          IELLST(NEL)=CNTYP(I)
          IXCNPT(I)=NEL
C           0-JUNCTION
          NEL=NEL+1
          NODE0=NEL
          IELLST(NEL)=6
C           BOND BETWEEN CN AND 0-JUNCTION.
          NBD=NBD+1
          IBMX(NBD,1)=NEL
          IBMX(NBD,2)=NEL-1
C
C-        BUILD BG FROM 0-JUNCTION TO EACH MP.
C-        NEED TO LOOK AT EACH END POINT.
C
          DO 400 J=1,2
            MOTION=MOTTYP(MPONCN(I,J))
C
C-          END POINT FIXED.
C
            IF(MOTION.EQ.0) THEN
C
C-          MOTION IN X,Y OR X ONLY OR Y ONLY.
C
            ELSEIF(MOTION.LE.3) THEN
              CALL CNMP3
C
C-          MOTION IN THETA DIRECTION ONLY.
C
            ELSEIF(MOTION.EQ.4) THEN
              CALL CNMP4
C
C-          MOTION IN X AND THETA DIRECTION ONLY.
C
            ELSEIF(MOTION.EQ.5) THEN
              CALL CNMP5
C
C-          MOTION IN Y AND THETA DIRECTION ONLY.
C
            ELSEIF(MOTION.EQ.6) THEN
              CALL CNMP6
C
C-          MOTION IN X,Y AND THETA DIRECTION ONLY.
C
            ELSEIF(MOTION.EQ.7) THEN
              CALL CNMP7
            ENDIF
400       CONTINUE
C
C-        STORE THE TRANSFORMER DATA BETWEEN THE CONNECTOR
C-        AND ITS END POINTS.
C
          CALL CNTFEP
```

```
        ENDIF
500   CONTINUE
C
C--- CREATE THE NBIMX AND NPTR ARRAYS.
C
      CALL BNDNOD
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
      RETURN
      END
      SUBROUTINE GETEPS
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
CDCCCCC                 SUBROUTINE DESCRIPTION                   CCCCC
C                                                                    C
C--- THIS SUBROUTINE PRE-PROCESSES THE END POINTS THAT               C
C    ARE ASSOCIATED WITH CONNECTOR CNNAM(I).                         C
C                                                                    C
C--- PROGRAMMER: JOHN D. REID          DATE: FALL 1983               C
C                                                                    C
C--- CALLED FROM BG.                                                 C
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
CVCCCCCCCC              VARIABLE IDENTIFICATION                CCCCCCCC
C                                                                    C
C--- STORES THE END POINT TYPE IN IEPTYP(I).                         C
C                                                                    C
 INSERT COMBG
 INSERT SYBGBK
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
CPCCCCCCCC              PROCESS BLOCK                        BLOCK OCCC
C                                                                    C
      MOT1=MOTTYP(MPONCN(I,1))
      MOT2=MOTTYP(MPONCN(I,2))
C
C--- MOT1 = 0  -  NO MOTION FOR END POINT 1.
C
      IF(MOT1.EQ.0) THEN
        IF(MOT2.EQ.0) THEN
          IEPTYP(I)=0
        ELSEIF(MOT2.EQ.1.OR.MOT2.EQ.2) THEN
          IEPTYP(I)=1
        ELSE
          IEPTYP(I)=3
        ENDIF
C
C--- MOT1 = 1  -  X-MOTION ONLY FOR EP 1.
C
      ELSEIF(MOT1.EQ.1) THEN
        IF(MOT2.EQ.0) THEN
```

```
          IEPTYP(I)=1
        ELSEIF(MOT2.EQ.1) THEN
          IEPTYP(I)=4
        ELSEIF(MOT2.EQ.2) THEN
          IEPTYP(I)=5
        ELSE
          IEPTYP(I)=10
        ENDIF
C
C--- MOT1 = 2  -  Y-MOTION ONLY FOR EP 1.
C
      ELSEIF(MOT1.EQ.2) THEN
        IF(MOT2.EQ.0) THEN
          IEPTYP(I)=1
        ELSEIF(MOT2.EQ.1) THEN
          IEPTYP(I)=6
        ELSEIF(MOT2.EQ.2) THEN
          IEPTYP(I)=7
        ELSE
          IEPTYP(I)=11
        ENDIF
C
C--- MOT1 = 3,4,5,6,7  -  MOTION FOR END POINT 1:
C                            X AND Y
C                             THETA
C                            X AND/OR Y, AND THETA
C
      ELSE
        IF(MOT2.EQ.0) THEN
          IEPTYP(I)=2
        ELSEIF(MOT2.EQ.1) THEN
          IEPTYP(I)=8
        ELSEIF(MOT2.EQ.2) THEN
          IEPTYP(I)=9
        ELSE
          IEPTYP(I)=12
        ENDIF
      ENDIF
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
      RETURN
      END
      SUBROUTINE CNTFEP
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CDCCCCC                SUBROUTINE DESCRIPTION                     CCCCC
C                                                                     C
C--- THIS SUBROUTINE STORES THE TRANSFORMER DATA                      C
C    BETWEEN CONNECTOR I AND ITS ASSOCIATED END POINTS.               C
C    IEPTYP(I) DETERMINES THE TYPE OF DATA REQUIRED TO                C
C    BE STORED.                                                       C
C                                                                     C
C--- THE VALUES FOR THE TRANSFORMERS AND CONNECTOR IN THIS            C
```

```
C    SUBROUTINE ARE CALCULATED BY FUNCTION 5.                        C
C                                                                    C
C--- PROGRAMMER: JOHN D. REID          DATE: FALL 1983               C
C                                                                    C
C--- CALLED FROM BG.                                                 C
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
CVCCCCCCCC             VARIABLE IDENTIFICATION                CCCCCCCC
C                                                                    C
C    NTFEL - NUMBER OF TRANSFORMER NODES TO HANDLE.
C
C    ITFEL - THE TRANSFORMER NUMBER FOR EACH NODE.
C
 INSERT COMBG
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
CPCCCCCCCC             PROCESS BLOCK                          BLOCK 0CCC
C                                                                    C
C
C--- SET FUNCTION NUMBER, REQUIRED ARGUMENTS, NUMBER
C--- OF POINTERS TO FOLLOW, AND CONNECTOR POINTER.
C
      DO 100 L=1,NTFEL
        MTFDAT(ITFEL(L),1)=5
        MTFDAT(ITFEL(L),2)=I*4-3
        MTFDAT(ITFEL(L),3)=I*4-2
        MTFDAT(ITFEL(L),4)=I*4-1
        MTFDAT(ITFEL(L),5)=I*4
        MTFDAT(ITFEL(L),7)=NTFEL
        MTFDAT(ITFEL(L),8)=I
100   CONTINUE
C
C----  THE STORAGE DEPENDS ON IEPTYP(I).
C
C
C--- IEPTYP(I) = 2
C
      IF(IEPTYP(I).EQ.2) THEN
C
C-      MODULATOR FUNCTION NUMBER.
C
        MTFDAT(ITFEL(1),6)=1
        MTFDAT(ITFEL(2),6)=2
C
C-      POINTERS TO OTHER TRANSFORMERS.
C
        MTFDAT(ITFEL(1),9)=ITFEL(2)
        MTFDAT(ITFEL(2),9)=ITFEL(1)
C
C--- IEPTYP(I) = 3
C
      ELSEIF(IEPTYP(I).EQ.3) THEN
C
```

```
C-      MODULATOR FUNCTION NUMBER.
C
        MTFDAT(ITFEL(1),6)=3
        MTFDAT(ITFEL(2),6)=4
C
C-      POINTERS TO OTHER TRANSFORMERS.
C
        MTFDAT(ITFEL(1),9)=ITFEL(2)
        MTFDAT(ITFEL(2),9)=ITFEL(1)
C
C--- IEPTYP(I) = 4
C
      ELSEIF(IEPTYP(I).EQ.4) THEN
C
C-      MODULATOR FUNCTION NUMBER.
C
        MTFDAT(ITFEL(1),6)=1
        MTFDAT(ITFEL(2),6)=3
C
C-      POINTERS TO OTHER TRANSFORMERS.
C
        MTFDAT(ITFEL(1),9)=ITFEL(2)
        MTFDAT(ITFEL(2),9)=ITFEL(1)
C
C--- IEPTYP(I) = 5
C
      ELSEIF(IEPTYP(I).EQ.5) THEN
C
C-      MODULATOR FUNCTION NUMBER.
C
        MTFDAT(ITFEL(1),6)=1
        MTFDAT(ITFEL(2),6)=4
C
C-      POINTERS TO OTHER TRANSFORMERS.
C
        MTFDAT(ITFEL(1),9)=ITFEL(2)
        MTFDAT(ITFEL(2),9)=ITFEL(1)
C
C--- IEPTYP(I) = 6
C
      ELSEIF(IEPTYP(I).EQ.6) THEN
C
C-      MODULATOR FUNCTION NUMBER.
C
        MTFDAT(ITFEL(1),6)=2
        MTFDAT(ITFEL(2),6)=3
C
C-      POINTERS TO OTHER TRANSFORMERS.
C
        MTFDAT(ITFEL(1),9)=ITFEL(2)
        MTFDAT(ITFEL(2),9)=ITFEL(1)
C
C--- IEPTYP(I) = 7
C
      ELSEIF(IEPTYP(I).EQ.7) THEN
```

```
C
C-      MODULATOR FUNCTION NUMBER.
C
        MTFDAT(ITFEL(1),6)=2
        MTFDAT(ITFEL(2),6)=4
C
C-      POINTERS TO OTHER TRANSFORMERS.
C
        MTFDAT(ITFEL(1),9)=ITFEL(2)
        MTFDAT(ITFEL(2),9)=ITFEL(1)
C
C--- IEPTYP(I) = 8
C
      ELSEIF(IEPTYP(I).EQ.8) THEN
C
C-      MODULATOR FUNCTION NUMBER.
C
        MTFDAT(ITFEL(1),6)=1
        MTFDAT(ITFEL(2),6)=2
        MTFDAT(ITFEL(3),6)=3
C
C-      POINTERS TO OTHER TRANSFORMERS.
C
        MTFDAT(ITFEL(1),9)=ITFEL(2)
        MTFDAT(ITFEL(1),10)=ITFEL(3)
        MTFDAT(ITFEL(2),9)=ITFEL(1)
        MTFDAT(ITFEL(2),10)=ITFEL(3)
        MTFDAT(ITFEL(3),9)=ITFEL(1)
        MTFDAT(ITFEL(3),10)=ITFEL(2)
C
C--- IEPTYP(I) = 9
C
      ELSEIF(IEPTYP(I).EQ.9) THEN
C
C-      MODULATOR FUNCTION NUMBER.
C
        MTFDAT(ITFEL(1),6)=1
        MTFDAT(ITFEL(2),6)=2
        MTFDAT(ITFEL(3),6)=4
C
C-      POINTERS TO OTHER TRANSFORMERS.
C
        MTFDAT(ITFEL(1),9)=ITFEL(2)
        MTFDAT(ITFEL(1),10)=ITFEL(3)
        MTFDAT(ITFEL(2),9)=ITFEL(1)
        MTFDAT(ITFEL(2),10)=ITFEL(3)
        MTFDAT(ITFEL(3),9)=ITFEL(1)
        MTFDAT(ITFEL(3),10)=ITFEL(2)
C
C--- IEPTYP(I) = 10
C
      ELSEIF(IEPTYP(I).EQ.10) THEN
C
C-      MODULATOR FUNCTION NUMBER.
C
```

```
      MTFDAT(ITFEL(1),6)=1
      MTFDAT(ITFEL(2),6)=3
      MTFDAT(ITFEL(3),6)=4
C
C-    POINTERS TO OTHER TRANSFORMERS.
C
      MTFDAT(ITFEL(1),9)=ITFEL(2)
      MTFDAT(ITFEL(1),10)=ITFEL(3)
      MTFDAT(ITFEL(2),9)=ITFEL(1)
      MTFDAT(ITFEL(2),10)=ITFEL(3)
      MTFDAT(ITFEL(3),9)=ITFEL(1)
      MTFDAT(ITFEL(3),10)=ITFEL(2)
C
C--- IEPTYP(I) = 11
C
     ELSEIF(IEPTYP(I).EQ.11) THEN
C
C-    MODULATOR FUNCTION NUMBER.
C
      MTFDAT(ITFEL(1),6)=2
      MTFDAT(ITFEL(2),6)=3
      MTFDAT(ITFEL(3),6)=4
C
C-    POINTERS TO OTHER TRANSFORMERS.
C
      MTFDAT(ITFEL(1),9)=ITFEL(2)
      MTFDAT(ITFEL(1),10)=ITFEL(3)
      MTFDAT(ITFEL(2),9)=ITFEL(1)
      MTFDAT(ITFEL(2),10)=ITFEL(3)
      MTFDAT(ITFEL(3),9)=ITFEL(1)
      MTFDAT(ITFEL(3),10)=ITFEL(2)
C
C--- IEPTYP(I) = 12
C
     ELSEIF(IEPTYP(I).EQ.12) THEN
C
C-    MODULATOR FUNCTION NUMBER.
C
      MTFDAT(ITFEL(1),6)=1
      MTFDAT(ITFEL(2),6)=2
      MTFDAT(ITFEL(3),6)=3
      MTFDAT(ITFEL(4),6)=4
C
C-    POINTERS TO OTHER TRANSFORMERS.
C
      MTFDAT(ITFEL(1),9)=ITFEL(2)
      MTFDAT(ITFEL(1),10)=ITFEL(3)
      MTFDAT(ITFEL(1),11)=ITFEL(4)
      MTFDAT(ITFEL(2),9)=ITFEL(1)
      MTFDAT(ITFEL(2),10)=ITFEL(3)
      MTFDAT(ITFEL(2),11)=ITFEL(4)
      MTFDAT(ITFEL(3),9)=ITFEL(1)
      MTFDAT(ITFEL(3),10)=ITFEL(2)
      MTFDAT(ITFEL(3),11)=ITFEL(4)
      MTFDAT(ITFEL(4),9)=ITFEL(1)
```

```
        MTFDAT(ITFEL(4),10)=ITFEL(2)
        MTFDAT(ITFEL(4),11)=ITFEL(3)
      ENDIF
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
      RETURN
      END
      SUBROUTINE BNDNOD
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CDCCCCC               SUBROUTINE DESCRIPTION                     CCCCC
C                                                                     C
C--- THIS SUBROUTINE CREATES THE BONDS ATTACHED                       C
C    TO A GIVEN NODE ARRAY ( NBIMX ) AND ITS                          C
C    POINTER ( NPTR ).                                                C
C                                                                     C
C--- PROGRAMMER: JOHN D. REID          DATE: FALL 1983                C
C                                                                     C
C--- CALLED FROM BG.                                                  C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CVCCCCCCCC            VARIABLE IDENTIFICATION                 CCCCCCCC
C                                                                     C
 INSERT SYBGBK
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CPCCCCCCCC            PROCESS BLOCK                           BLOCK OCCC
C                                                                     C
C--- LOOP THROUGH THE BONDS TO LOOK AT ATTACHED NODES.
C
      DO 300 IBND=1,NBD
C
C--     LOOK AT EACH NODE.
C
        DO 200 INOD=1,2
          NODE=IBMX(IBND,INOD)
          IPT=NPTR(NODE)
          IF(NBIMX(IPT).EQ.0) THEN
            NBIMX(IPT)=IBND
          ELSE
C
C-          INCREMENT NBIMX.
C
            IBEGIN=NPTR(NEL+1)
            IEND=IPT+1
            DO 100 K=IBEGIN,IEND,-1
              NBIMX(K)=NBIMX(K-1)
100         CONTINUE
C
C-          STORE THE BOND.
C
```

```
            NBIMX(IPT)=IBND
C
C-          INCREMENT POINTER - NPTR.
C
            DO 150 K=NODE+1,NEL+1
              NPTR(K)=NPTR(K)+1
150         CONTINUE
          ENDIF
200     CONTINUE
300   CONTINUE
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
      RETURN
      END
      SUBROUTINE CNMP3
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CDCCCCC                 SUBROUTINE DESCRIPTION                    CCCCC
C                                                                     C
C--- THIS SUBROUTINE DOES THE FOLLOWING:                              C
C                                                                     C
C    KNOWN: CONNECTOR ( CNNAM(I) ) IS CONNECTED TO MOTION             C
C           POINT ( MPONCN(I,J) ).                                    C
C           THIS MOTION POINT IS OF MOTTYP 1, 2 OR 3.                 C
C                                                                     C
C    CREATE THE BG FROM THE CONNECTOR 0-JUNCTION ( NODE0 )            C
C    TO THIS MOTION POINT'S 1-JUNCTION.                               C
C                                                                     C
C--- PROGRAMMER: JOHN D. REID          DATE: FALL 1983                C
C                                                                     C
C--- CALLED FROM BG.                                                  C
C                                                                     C
C--- SUBROUTINES CALLED:   BG0TF1                                     C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CVCCCCCCCC              VARIABLE IDENTIFICATION                CCCCCCCC
C                                                                     C
 INSERT COMBG
 INSERT SYBGBK
C                                                                     C
      IT1=0
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CPCCCCCCCC              PROCESS BLOCK                       BLOCK 0CCC
C                                                                     C
C--- IXORY WILL TELL US WHICH EP 1-JUNCTION IS
C    BEING CREATED. ( X OR Y )
C
50    IF(MOTTYP(MPONCN(I,J)).EQ.1) THEN
        IXORY=1
      ELSEIF(MOTTYP(MPONCN(I,J)).EQ.2) THEN
```

```
          IXORY=2
        ELSE
          IF(IT1.EQ.0) THEN
            IXORY=1
          ELSE
            IXORY=2
          ENDIF
        ENDIF
C
C--- BUILD BG FROM CONNECTOR 0-JUNCTION TO END POINT.
C
        CALL BGOTF1
C
C--- BUILD BG FROM END POINT TO MOTION POINT.
C
        NBD=NBD+1
        IF(J.EQ.1) THEN
          IBMX(NBD,1)=MP1PTR(MPONCN(I,J))+IT1
          IBMX(NBD,2)=NEL
        ELSE
          IBMX(NBD,1)=NEL
          IBMX(NBD,2)=MP1PTR(MPONCN(I,J))+IT1
        ENDIF
C
C--- FOR MOTTYP 3 THERE ARE TWO MP'S.
C
        IF(MOTTYP(MPONCN(I,J)).EQ.3) THEN
          IF(IT1.EQ.1) GOTO 999
          IT1=1
          GOTO 50
        ENDIF
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
999     RETURN
        END
        SUBROUTINE CNMP4
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
CDCCCCC                  SUBROUTINE DESCRIPTION                   CCCCC
C                                                                     C
C--- THIS SUBROUTINE DOES THE FOLLOWING:                              C
C                                                                     C
C    KNOWN: CONNECTOR ( CNNAM(I) ) IS CONNECTED TO MOTION             C
C           POINT ( MPONCN(I,J) ).                                    C
C           THIS MOTION POINT IS OF MOTTYP 4.                         C
C                                                                     C
C    CREATE THE BG FROM THE CONNECTOR 0-JUNCTION ( NODEO )            C
C    TO THIS MOTION POINT'S 1-JUNCTION.                               C
C                                                                     C
C--- PROGRAMMER: JOHN D. REID          DATE: FALL 1983                C
C                                                                     C
C--- CALLED FROM BG.                                                  C
C                                                                     C
```

```
C--- SUBROUTINES CALLED:   BGOTF1   BGEPTT                          C
C                                                                   C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                   C
CVCCCCCCCC            VARIABLE IDENTIFICATION               CCCCCCCC
C                                                                   C
 INSERT COMBG
 INSERT SYBGBK
C                                                                   C
      IT1=MP1PTR(MPONCN(I,J))
C                                                                   C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                   C
CPCCCCCCCC            PROCESS BLOCK                       BLOCK 0CCC
C                                                                   C
C--- BUILD BG FROM NODE0 TO MP THROUGH THE X END POINT
C    AND Y END POINT OF CONNECTOR I.
C
      DO 200 K=1,2
        IXORY=K
C
C-      BG FROM NODE0 TO CONNECTOR END POINT.
C
        CALL BGOTF1
C
C-      BG FROM END POINT TO THETA MOTION POINT.
C
        IFUN=K
        IMPN=IT1
        CALL BGEPTT(IFUN,IMPN)
200   CONTINUE
C                                                                   C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                   C
      RETURN
      END
      SUBROUTINE CNMP5
C                                                                   C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                   C
CDCCCCC               SUBROUTINE DESCRIPTION                   CCCCC
C                                                                   C
C--- THIS SUBROUTINE DOES THE FOLLOWING:                            C
C                                                                   C
C    KNOWN: CONNECTOR ( CNNAM(I) ) IS CONNECTED TO MOTION           C
C           POINT ( MPONCN(I,J) ).                                  C
C           THIS MOTION POINT IS OF MOTTYP 5.                       C
C                                                                   C
C    CREATE THE BG FROM THE CONNECTOR 0-JUNCTION ( NODE0 )          C
C    TO THIS MOTION POINT'S 1-JUNCTION.                             C
C                                                                   C
C--- PROGRAMMER: JOHN D. REID          DATE: FALL 1983              C
C                                                                   C
C--- CALLED FROM BG.                                                C
C                                                                   C
```

```
C--- SUBROUTINES CALLED:   BGOTF1   BGEPTT   BGEPMP                 C
C                                                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                  C
CVCCCCCCCC            VARIABLE IDENTIFICATION               CCCCCCCC
C                                                                  C
 INSERT COMBG
 INSERT SYBGBK
C                                                                  C
      IT1=MP1PTR(MPONCN(I,J))
C                                                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                  C
CPCCCCCCCC            PROCESS BLOCK                         BLOCK 0CCC
C                                                                  C
C--- BG FROM CONNECTOR 0-JUNCTION ( NODE0 ) TO
C    X AND THETA MP VIA X END POINT.
C
C-     BG FROM NODE0 TO X EP.
C
      IXORY=1
      CALL BGOTF1
C
C-     BG FROM X EP TO X AND THETA MP.
C
      IFUN=1
      IMPN1=IT1
      IMPN2=IT1+1
      CALL BGEPMP(IFUN,IMPN1,IMPN2)
C
C--- BG FROM NODE0 TO THETA MP VIA Y END POINT.
C
C-     BG FROM NODE0 TO Y EP.
C
      IXORY=2
      CALL BGOTF1
C
C--    BG FROM Y EP TO THETA MP.
C
      IFUN=2
      IMPN=IT1+1
      CALL BGEPTT(IFUN,IMPN)
C                                                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                  C
      RETURN
      END
      SUBROUTINE CNMP6
C                                                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                  C
CDCCCCC               SUBROUTINE DESCRIPTION                   CCCCC
C                                                                  C
C--- THIS SUBROUTINE DOES THE FOLLOWING:                           C
C                                                                  C
```

```
C     KNOWN: CONNECTOR ( CNNAM(I) ) IS CONNECTED TO MOTION          C
C            POINT ( MPONCN(I,J) ).                                 C
C            THIS MOTION POINT IS OF MOTTYP 6.                      C
C                                                                   C
C     CREATE THE BG FROM THE CONNECTOR 0-JUNCTION ( NODE0 )         C
C     TO THIS MOTION POINT'S 1-JUNCTION.                            C
C                                                                   C
C--- PROGRAMMER: JOHN D. REID          DATE: FALL 1983              C
C                                                                   C
C--- CALLED FROM BG.                                                C
C                                                                   C
C--- SUBROUTINES CALLED:   BGOTF1   BGEPTT   BGEPMP                 C
C                                                                   C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                   C
CVCCCCCCCC            VARIABLE IDENTIFICATION               CCCCCCCC
C                                                                   C
 INSERT COMBG
 INSERT SYBGBK
C                                                                   C
      IT1=MP1PTR(MPONCN(I,J))
C                                                                   C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                   C
CPCCCCCCCC            PROCESS BLOCK                         BLOCK 0CCC
C                                                                   C
C--- BG FROM NODE0 TO THETA MP VIA X END POINT.
C
C-    BG FROM NODE0 TO X EP.
C
      IXORY=1
      CALL BGOTF1
C
C-    BG FROM X EP TO THETA MP.
C
      IFUN=1
      IMPN=IT1+1
      CALL BGEPTT(IFUN,IMPN)
C
C--- BG FROM CONNECTOR 0-JUNCTION ( NODE0 ) TO
C     Y AND THETA MP VIA Y END POINT.
C
C-    BG FROM NODE0 TO Y EP.
C
      IXORY=2
      CALL BGOTF1
C
C-    BG FROM Y EP TO Y AND THETA MP.
C
      IFUN=2
      IMPN1=IT1
      IMPN2=IT1+1
      CALL BGEPMP(IFUN,IMPN1,IMPN2)
C                                                                   C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
C                                                                       C
      RETURN
      END
      SUBROUTINE CNMP7
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
CDCCCCC                SUBROUTINE DESCRIPTION                       CCCCC
C                                                                       C
C--- THIS SUBROUTINE DOES THE FOLLOWING:                                C
C                                                                       C
C    KNOWN: CONNECTOR ( CNNAM(I) ) IS CONNECTED TO MOTION               C
C           POINT ( MPONCN(I,J) ).                                      C
C           THIS MOTION POINT IS OF MOTTYP 7.                           C
C                                                                       C
C    CREATE THE BG FROM THE CONNECTOR 0-JUNCTION ( NODE0 )              C
C    TO THIS MOTION POINT'S 1-JUNCTION.                                 C
C                                                                       C
C--- PROGRAMMER: JOHN D. REID          DATE: FALL 1983                  C
C                                                                       C
C--- CALLED FROM BG.                                                    C
C                                                                       C
C--- SUBROUTINES CALLED:   BGOTF1   BGEPMP                              C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
CVCCCCCCCC             VARIABLE IDENTIFICATION                   CCCCCCCC
C                                                                       C
 INSERT COMBG
 INSERT SYBGBK
C                                                                       C
      IT1=MP1PTR(MPONCN(I,J))
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
CPCCCCCCCC             PROCESS BLOCK                            BLOCK 0CCC
C                                                                       C
C--- BG FROM CONNECTOR 0-JUNCTION ( NODE0 ) TO
C    X AND THETA MP VIA X END POINT.
C
C-    BG FROM NODE0 TO X EP.
C
      IXORY=1
      CALL BGOTF1
C
C-    BG FROM X EP TO X AND THETA MP.
C
      IFUN=1
      IMPN1=IT1
      IMPN2=IT1+2
      CALL BGEPMP(IFUN,IMPN1,IMPN2)
C
C--- BG FROM CONNECTOR 0-JUNCTION ( NODE0 ) TO
C    Y AND THETA MP VIA Y END POINT.
C
```

```
C-    BG FROM NODEO TO Y EP.
C
      IXORY=2
      CALL BGOTF1
C
C-    BG FROM Y EP TO Y AND THETA MP.
C
      IFUN=2
      IMPN1=IT1+1
      IMPN2=IT1+2
      CALL BGEPMP(IFUN,IMPN1,IMPN2)
C                                                                 C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                 C
      RETURN
      END
      SUBROUTINE BGOTF1
C                                                                 C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                 C
CDCCCCC               SUBROUTINE DESCRIPTION                  CCCCC
C                                                                 C
C--- THIS SUBROUTINE BUILDS THE BG FROM THE CONNECTOR             C
C    0-JUNCTION TO ONE OF ITS END POINTS THROUGH A                C
C    TRANSFORMER.                                                 C
C                                                                 C
C--- PROGRAMMER: JOHN D. REID          DATE: FALL 1983            C
C                                                                 C
C--- CALLED FROM:   CNMP3   CNMP4   CNMP5                         C
C                   CNMP6   CNMP7                                 C
C                                                                 C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                 C
CVCCCCCCCC            VARIABLE IDENTIFICATION              CCCCCCCC
C                                                                 C
 INSERT COMBG
 INSERT SYBGBK
C                                                                 C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                 C
CPCCCCCCCC            PROCESS BLOCK                     BLOCK OCCC
C                                                                 C
C--- BG FROM 0-JUNCTION ( NODEO ) TO TRANSFORMER.
C
      NBD=NBD+1
C
C-      TRANSFORMER NODE AND POINTERS.
C
      NEL=NEL+1
      IELLST(NEL)=8
      NTF=NTF+1
      MTF(NTF)=NEL
      NTFEL=NTFEL+1
      ITFEL(NTFEL)=NTF
C
```

```
C-      BOND BETWEEN NODEO AND TF.
      IF(J.EQ.1) THEN
        IBMX(NBD,1)=NEL
        IBMX(NBD,2)=NODEO
      ELSE
        IBMX(NBD,1)=NODEO
        IBMX(NBD,2)=NEL
      ENDIF
C
C--- BG FROM TF TO END POINT.
C
      NBD=NBD+1
C
C-      1-JUNCTION END POINT AND POINTER.
C
      NEL=NEL+1
      IELLST(NEL)=7
      IF(J.EQ.2) IXORY=IXORY+2
      IT2=(I-1)*4+IXORY
      IXEPPT(IT2)=NEL
C-      BOND BETWEEN TF AND 1-JUNCTION.
      IF(J.EQ.1) THEN
        IBMX(NBD,1)=NEL
        IBMX(NBD,2)=NEL-1
      ELSE
        IBMX(NBD,1)=NEL-1
        IBMX(NBD,2)=NEL
      ENDIF
C                                                                 C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                 C
      RETURN
      END
      SUBROUTINE BGEPTT(IFUN,IMPN)
C                                                                 C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                 C
CDCCCCC              SUBROUTINE DESCRIPTION                   CCCCC
C                                                                 C
C--- THIS SUBROUTINE BUILDS THE BOND GRAPH BETWEEN AN             C
C    END POINT 1-JUNCTION ( NEL ) AND THE MOTION POINT            C
C    THETA 1-JUNCTION ( IMPN ) THAT IS CONNECTED THROUGH          C
C    A TRANSFORMER ASSOCIATED WITH FUNCTION IFUN.                 C
C                                                                 C
C--- PROGRAMMER: JOHN D. REID          DATE: FALL 1983            C
C                                                                 C
C--- CALLED FROM:   CNMP4   CNMP5   CNMP6                         C
C                                                                 C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                 C
CVCCCCCCCC           VARIABLE IDENTIFICATION               CCCCCCCC
C                                                                 C
 INSERT COMBG
 INSERT SYBGBK
C                                                                 C
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
CPCCCCCCCC            PROCESS BLOCK                        BLOCK OCCC
C                                                                    C
C-    TRANSFORMER BETWEEN END POINT AND THETA MOTION POINT.
C
      NEL=NEL+1
      IELLST(NEL)=8
C
C-    STORE THE TRANSFORMER AND ITS DATA.
C
      NTF=NTF+1
      MTF(NTF)=NEL
C
C-    MP IS A MASS: FUNCTION IFUN  ( IFUN=1 OR 2 )
C                   2 CONSTANTS
C                   1 ARGUMENT
C
      MPN=MPONCN(I,J)
      IF(MPN.LE.NMS) THEN
        MTFDAT(NTF,1)=IFUN
        MTFDAT(NTF,2)=GEOMAS(MPN,I,1)
        MTFDAT(NTF,3)=GEOMAS(MPN,I,2)
        MTFDAT(NTF,4)=MPN*3
C
C-    MP IS A BAR: FUNCTION IFUN+2
C                  1 CONSTANT
C                  1 ARGUMENT
C
      ELSE
        MTFDAT(NTF,1)=IFUN+2
        MTFDAT(NTF,2)=GEOBAR(MPN-NMS,I)
        MTFDAT(NTF,3)=MPN*3
      ENDIF
C
C-    BONDS FROM EP TO MP 1-JUNCTIONS.
C
      NBD=NBD+1
      IF(J.EQ.1) THEN
        IBMX(NBD,1)=NEL
        IBMX(NBD,2)=NEL-1
        NBD=NBD+1
        IBMX(NBD,1)=IMPN
        IBMX(NBD,2)=NEL
      ELSE
        IBMX(NBD,1)=NEL-1
        IBMX(NBD,2)=NEL
        NBD=NBD+1
        IBMX(NBD,1)=NEL
        IBMX(NBD,2)=IMPN
      ENDIF
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
      RETURN
```

```
      END
      SUBROUTINE BGEPMP(IFUN,IMPN1,IMPN2)
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
CDCCCCC              SUBROUTINE DESCRIPTION                       CCCCC
C                                                                      C
C--- THIS SUBROUTINE BUILDS A BOND GRAPH FROM AN                       C
C    END POINT TO ITS ASSOCIATED MOTION POINT                          C
C    1-JUNCTIONS ( X OR Y, AND THETA ).                                C
C                                                                      C
C--- PROGRAMMER: JOHN D. REID          DATE: FALL 1983                 C
C                                                                      C
C--- CALLED FROM:   CNMP5   CNMP6   CNMP7                              C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
CVCCCCCCCC            VARIABLE IDENTIFICATION                  CCCCCCCC
C                                                                      C
 INSERT COMBG
 INSERT SYBGBK
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
CPCCCCCCCC            PROCESS BLOCK                          BLOCK 0CCC
C                                                                      C
C-    BG FROM EP TO X OR Y, AND THETA MP.
C
C-      0-JUNCTION
      NEL=NEL+1
      IELLST(NEL)=6
C
C-    TRANSFORMER BETWEEN END POINT AND THETA MOTION POINT.
C
      NEL=NEL+1
      IELLST(NEL)=8
C
C-    STORE THE TRANSFORMER AND ITS DATA.
C
      NTF=NTF+1
      MTF(NTF)=NEL
C
C-    MP IS A MASS: FUNCTION IFUN
C                   2 CONSTANTS
C                   1 ARGUMENT
C
      MPN=MPONCN(I,J)
      IF(MPN.LE.NMS) THEN
        MTFDAT(NTF,1)=IFUN
        MTFDAT(NTF,2)=GEOMAS(MPN,I,1)
        MTFDAT(NTF,3)=GEOMAS(MPN,I,2)
        MTFDAT(NTF,4)=MPN*3
C
C-    MP IS A BAR: FUNCTION IFUN+2
C                  1 CONSTANT
```

```
C                    1 ARGUMENT
C
      ELSE
        MTFDAT(NTF,1)=IFUN+2
        MTFDAT(NTF,2)=GEOBAR(MPN-NMS,I)
        MTFDAT(NTF,3)=MPN*3
      ENDIF
C
C-      REQUIRED BONDS.
C
      NBD=NBD+1
      IF(J.EQ.1) THEN
        IBMX(NBD,1)=NEL-1
        IBMX(NBD,2)=NEL-2
        NBD=NBD+1
        IBMX(NBD,1)=IMPN1
        IBMX(NBD,2)=NEL-1
        NBD=NBD+1
        IBMX(NBD,1)=NEL
        IBMX(NBD,2)=NEL-1
        NBD=NBD+1
        IBMX(NBD,1)=IMPN2
        IBMX(NBD,2)=NEL
      ELSE
        IBMX(NBD,1)=NEL-2
        IBMX(NBD,2)=NEL-1
        NBD=NBD+1
        IBMX(NBD,1)=NEL-1
        IBMX(NBD,2)=IMPN1
        NBD=NBD+1
        IBMX(NBD,1)=NEL-1
        IBMX(NBD,2)=NEL
        NBD=NBD+1
        IBMX(NBD,1)=NEL
        IBMX(NBD,2)=IMPN2
      ENDIF
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
      RETURN
      END
```

# APPENDIX D

## SPRING-MASS-DAMPER RESULTS

# APPENDIX D

## SPRING-MASS-DAMPER RESULTS

The following is the data base results from BILDBG for the spring-mass-damper system discussed in Chapter 4.1.

EXAMPLE: SPRING-MASS-DAMPER SYSTEM

| | | | |
|---|---|---|---|
| ITMLBL : | M1 | NEWLBL : | M1 |
| ITMLBL : | B1 | NEWLBL : | B1 |
| ITMLBL : | S1 | NEWLBL : | S1 |
| ITMLBL : | D1 | NEWLBL : | D1 |
| ITMLBL : | V1 | NEWLBL : | F1 |
| ITMLBL : | F1 | NEWLBL : | V1 |

THE MPNAM LIST

| | |
|---|---|
| MPNAM( 1) = M1 | MOTTYP( 1) = 1 |
| MPNAM( 2) = B1 | MOTTYP( 2) = 1 |

THE CNNAM LIST

| | | |
|---|---|---|
| CNNAM( 1) = S1 | CNTYP( 1) = 1 | IEPTYP( 1) = 4 |
| CNNAM( 2) = D1 | CNTYP( 2) = 3 | IEPTYP( 2) = 4 |
| CNNAM( 3) = F1 | CNTYP( 3) = 4 | IEPTYP( 3) = 1 |
| CNNAM( 4) = V1 | CNTYP( 4) = 5 | IEPTYP( 4) = 1 |

MOTION POINTS ON CONNECTORS :

| | |
|---|---|
| MPONCN( 1,1) = 1 | MPONCN( 1,2) = 2 |
| MPONCN( 2,1) = 1 | MPONCN( 2,2) = 2 |
| MPONCN( 3,1) = 1 | MPONCN( 3,2) = 0 |

MPONCN( 4,1) = 2          MPONCN( 4,2) = 0

THE CPTR LIST

| | | |
|---|---|---|
| 1 | 4 | 7 |

THE CNONMP ARRAY

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 1 | 2 | 4 | 0 | 0 |

LOCAL GEOMETRY FOR THE MASSES.

MOTION POINT : M1

CONNECTOR : S1    X-DISTANCE = -1.00    Y-DISTANCE = 1.00

CONNECTOR : D1    X-DISTANCE = -1.00    Y-DISTANCE = -1.00

CONNECTOR : F1    X-DISTANCE = 1.00    Y-DISTANCE = 0.00

LOCAL GEOMETRY FOR THE BARS.

MOTION POINT : B1

CONNECTOR : S1    DISTANCE = 1.00

CONNECTOR : D1    DISTANCE = -1.00

CONNECTOR : V1    DISTANCE = 2.00

THE BOND GRAPH.

NODES ATTACHED TO BOND 1 ARE 1 3
NODES ATTACHED TO BOND 2 ARE 5 4
NODES ATTACHED TO BOND 3 ARE 6 5
NODES ATTACHED TO BOND 4 ARE 7 6
NODES ATTACHED TO BOND 5 ARE 1 7
NODES ATTACHED TO BOND 6 ARE 5 8
NODES ATTACHED TO BOND 7 ARE 8 9
NODES ATTACHED TO BOND 8 ARE 9 2
NODES ATTACHED TO BOND 9 ARE 11 10
NODES ATTACHED TO BOND 10 ARE 12 11
NODES ATTACHED TO BOND 11 ARE 13 12
NODES ATTACHED TO BOND 12 ARE 1 13
NODES ATTACHED TO BOND 13 ARE 11 14

NODES ATTACHED TO BOND 14 ARE 14 15
NODES ATTACHED TO BOND 15 ARE 15 2
NODES ATTACHED TO BOND 16 ARE 17 16
NODES ATTACHED TO BOND 17 ARE 1 17
NODES ATTACHED TO BOND 18 ARE 19 18
NODES ATTACHED TO BOND 19 ARE 2 19

IELLST( 1) = 7
IELLST( 2) = 7
IELLST( 3) = 2
IELLST( 4) = 1
IELLST( 5) = 6
IELLST( 6) = 8
IELLST( 7) = 7
IELLST( 8) = 8
IELLST( 9) = 7
IELLST( 10) = 3
IELLST( 11) = 6
IELLST( 12) = 8
IELLST( 13) = 7
IELLST( 14) = 8
IELLST( 15) = 7
IELLST( 16) = 4
IELLST( 17) = 7
IELLST( 18) = 5
IELLST( 19) = 7

THE NPTR LIST

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 5 | 8 | 9 | 10 | 13 | 15 | 17 | 19 | 21 |
| 22 | 25 | 27 | 29 | 31 | 33 | 34 | 36 | 37 | 39 |

THE NBIMX ARRAY

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 17 | 12 | 5 | 1 | 19 | 15 | 8 | 1 | 2 | 6 |
| 3 | 2 | 4 | 3 | 5 | 4 | 7 | 6 | 8 | 7 |
| 9 | 13 | 10 | 9 | 11 | 10 | 12 | 11 | 14 | 13 |
| 15 | 14 | 16 | 17 | 16 | 18 | 19 | 18 | 0 | |

POINTER FROM MP NODES TO XMP.

IXMPPT( 1) = 1
IXMPPT( 2) = 4

POINTER FROM EP XMP TO NODE NUMBER

NODE ASSOCIATED WITH EP 1 = IXEPPT( 1) = 7

```
NODE ASSOCIATED WITH EP  2 = IXEPPT( 2) =   0
NODE ASSOCIATED WITH EP  3 = IXEPPT( 3) =   9
NODE ASSOCIATED WITH EP  4 = IXEPPT( 4) =   0
NODE ASSOCIATED WITH EP  5 = IXEPPT( 5) =  13
NODE ASSOCIATED WITH EP  6 = IXEPPT( 6) =   0
NODE ASSOCIATED WITH EP  7 = IXEPPT( 7) =  15
NODE ASSOCIATED WITH EP  8 = IXEPPT( 8) =   0
NODE ASSOCIATED WITH EP  9 = IXEPPT( 9) =  17
NODE ASSOCIATED WITH EP 10 = IXEPPT(10) =   0
NODE ASSOCIATED WITH EP 11 = IXEPPT(11) =   0
NODE ASSOCIATED WITH EP 12 = IXEPPT(12) =   0
NODE ASSOCIATED WITH EP 13 = IXEPPT(13) =  19
NODE ASSOCIATED WITH EP 14 = IXEPPT(14) =   0
NODE ASSOCIATED WITH EP 15 = IXEPPT(15) =   0
NODE ASSOCIATED WITH EP 16 = IXEPPT(16) =   0


 POINTER FROM CONNECTOR TO ITS NODE NUMBER

NODE NUMBER FOR CONNECTOR  1 = IXCNPT( 1) =   4
NODE NUMBER FOR CONNECTOR  2 = IXCNPT( 2) =  10
NODE NUMBER FOR CONNECTOR  3 = IXCNPT( 3) =  16
NODE NUMBER FOR CONNECTOR  4 = IXCNPT( 4) =  18


 TRANSFORMERS AND ASSOCIATED DATA.


MTF( 1) =   6       FUNCTION NUMBER = 5.
   ARGUMENTS :  1.    2.    3.    4.
   MODULATOR FUNCTION : 1.
   NUMBER OF POINTERS = 2.
     CONNECTOR POINTER =  1.
     MTF POINTERS       :   2.     0.     0.

MTF( 2) =   8       FUNCTION NUMBER = 5.
   ARGUMENTS :  1.    2.    3.    4.
   MODULATOR FUNCTION : 3.
   NUMBER OF POINTERS = 2.
     CONNECTOR POINTER =  1.
     MTF POINTERS       :   1.     0.     0.

MTF( 3) =  12       FUNCTION NUMBER = 5.
   ARGUMENTS :  5.    6.    7.    8.
   MODULATOR FUNCTION : 1.
   NUMBER OF POINTERS = 2.
     CONNECTOR POINTER =  2.
     MTF POINTERS       :   4.     0.     0.

MTF( 4) =  14       FUNCTION NUMBER = 5.
   ARGUMENTS :  5.    6.    7.    8.
   MODULATOR FUNCTION : 3.
   NUMBER OF POINTERS = 2.
     CONNECTOR POINTER =  2.
     MTF POINTERS       :   3.     0.     0.
```

# APPENDIX E

# MASS-SPRING PENDULUM RESULTS

## APPENDIX E

## MASS-SPRING PENDULUM RESULTS

The following is the data base results from BILDBG for the mass-spring pendulum discussed in Chapter 4.2.

EXAMPLE: MASS-SPRING PENDULUM

```
 ITMLBL :    G1         NEWLBL :    M1
 ITMLBL :    P1         NEWLBL :    G1
 ITMLBL :    S1         NEWLBL :    P1
 ITMLBL :    M1         NEWLBL :    S1
 ITMLBL :    F1         NEWLBL :    F1
```

THE MPNAM LIST

```
MPNAM( 1) = M1        MOTTYP( 1) =  3
MPNAM( 2) = G1        MOTTYP( 2) =  0
MPNAM( 3) = P1        MOTTYP( 3) =  0
```

THE CNNAM LIST

```
CNNAM( 1) = S1      CNTYP( 1) =  1       IEPTYP( 1) =  2
CNNAM( 2) = F1      CNTYP( 2) =  4       IEPTYP( 2) =  2
CNNAM( 3) = X1      CNTYP( 3) =  1       IEPTYP( 3) =  0
```

MOTION POINTS ON CONNECTORS :

```
MPONCN( 1,1) =  1          MPONCN( 1,2) =  3
MPONCN( 2,1) =  1          MPONCN( 2,2) =  0
MPONCN( 3,1) =  3          MPONCN( 3,2) =  2
```

THE CPTR LIST

```
     1       3       4       6
```

THE CNONMP ARRAY

```
     1       2       3       1       3       0
```

LOCAL GEOMETRY FOR THE MASSES.

MOTION POINT : M1

```
  CONNECTOR : S1    X-DISTANCE =    0.00    Y-DISTANCE =    0.00

  CONNECTOR : F1    X-DISTANCE =    0.00    Y-DISTANCE =    0.00
```

THE BOND GRAPH.

```
NODES ATTACHED TO BOND   1 ARE      1       3
NODES ATTACHED TO BOND   2 ARE      2       4
NODES ATTACHED TO BOND   3 ARE      6       5
NODES ATTACHED TO BOND   4 ARE      7       6
NODES ATTACHED TO BOND   5 ARE      8       7
NODES ATTACHED TO BOND   6 ARE      1       8
NODES ATTACHED TO BOND   7 ARE      9       6
NODES ATTACHED TO BOND   8 ARE     10       9
NODES ATTACHED TO BOND   9 ARE      2      10
NODES ATTACHED TO BOND  10 ARE     12      11
NODES ATTACHED TO BOND  11 ARE     13      12
NODES ATTACHED TO BOND  12 ARE     14      13
NODES ATTACHED TO BOND  13 ARE      1      14
NODES ATTACHED TO BOND  14 ARE     15      12
NODES ATTACHED TO BOND  15 ARE     16      15
NODES ATTACHED TO BOND  16 ARE      2      16

IELLST(  1) =   7
IELLST(  2) =   7
IELLST(  3) =   2
IELLST(  4) =   2
IELLST(  5) =   1
IELLST(  6) =   6
IELLST(  7) =   8
IELLST(  8) =   7
IELLST(  9) =   8
IELLST( 10) =   7
IELLST( 11) =   4
IELLST( 12) =   6
IELLST( 13) =   8
```

```
IELLST( 14) =   7
IELLST( 15) =   8
IELLST( 16) =   7
```

THE NPTR LIST

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 4 | 7 | 8 | 9 | 10 | 13 | 15 | 17 | 19 |
| 21 | 22 | 25 | 27 | 29 | 31 | 33 | | | |

THE NBIMX ARRAY

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 13 | 6 | 1 | 16 | 9 | 2 | 1 | 2 | 3 | 7 |
| 4 | 3 | 5 | 4 | 6 | 5 | 8 | 7 | 9 | 8 |
| 10 | 14 | 11 | 10 | 12 | 11 | 13 | 12 | 15 | 14 |
| 16 | 15 | 0 | | | | | | | |

POINTER FROM MP NODES TO XMP.

```
IXMPPT( 1) =   1
IXMPPT( 2) =   2
```

POINTER FROM EP XMP TO NODE NUMBER

```
NODE ASSOCIATED WITH EP  1 = IXEPPT( 1) =   8
NODE ASSOCIATED WITH EP  2 = IXEPPT( 2) =  10
NODE ASSOCIATED WITH EP  3 = IXEPPT( 3) =   0
NODE ASSOCIATED WITH EP  4 = IXEPPT( 4) =   0
NODE ASSOCIATED WITH EP  5 = IXEPPT( 5) =  14
NODE ASSOCIATED WITH EP  6 = IXEPPT( 6) =  16
NODE ASSOCIATED WITH EP  7 = IXEPPT( 7) =   0
NODE ASSOCIATED WITH EP  8 = IXEPPT( 8) =   0
NODE ASSOCIATED WITH EP  9 = IXEPPT( 9) =   0
NODE ASSOCIATED WITH EP 10 = IXEPPT(10) =   0
NODE ASSOCIATED WITH EP 11 = IXEPPT(11) =   0
NODE ASSOCIATED WITH EP 12 = IXEPPT(12) =   0
```

POINTER FROM CONNECTOR TO ITS NODE NUMBER

```
NODE NUMBER FOR CONNECTOR  1 = IXCNPT( 1) =   5
NODE NUMBER FOR CONNECTOR  2 = IXCNPT( 2) =  11
NODE NUMBER FOR CONNECTOR  3 = IXCNPT( 3) =   0
```

TRANSFORMERS AND ASSOCIATED DATA.

```
MTF( 1) =   7       FUNCTION NUMBER = 5.
   ARGUMENTS :  1.    2.    3.    4.
   MODULATOR FUNCTION : 1.
   NUMBER OF POINTERS = 2.
     CONNECTOR POINTER =  1.
     MTF POINTERS      :  2.     0.     0.

MTF( 2) =   9       FUNCTION NUMBER = 5.
   ARGUMENTS :  1.    2.    3.    4.
   MODULATOR FUNCTION : 2.
   NUMBER OF POINTERS = 2.
     CONNECTOR POINTER =  1.
     MTF POINTERS      :  1.     0.     0.

MTF( 3) =  13       FUNCTION NUMBER = 5.
   ARGUMENTS :  5.    6.    7.    8.
   MODULATOR FUNCTION : 1.
   NUMBER OF POINTERS = 2.
     CONNECTOR POINTER =  2.
     MTF POINTERS      :  4.     0.     0.

MTF( 4) =  15       FUNCTION NUMBER = 5.
   ARGUMENTS :  5.    6.    7.    8.
   MODULATOR FUNCTION : 2.
   NUMBER OF POINTERS = 2.
     CONNECTOR POINTER =  2.
     MTF POINTERS      :  3.     0.     0.
```

# APPENDIX F

# VEHICLE SUSPENSION RESULTS

# APPENDIX F

## VEHICLE SUSPENSION RESULTS

The following is the data base results from BILDBG for the vehicle suspension discussed in Chapter 4.3.

EXAMPLE: VEHICLE SUSPENSION

```
ITMLBL :   S2        NEWLBL :   M1
ITMLBL :   D2        NEWLBL :   M2
ITMLBL :   J2        NEWLBL :   J2
ITMLBL :   S1        NEWLBL :   J1
ITMLBL :   D1        NEWLBL :   S2
ITMLBL :   J1        NEWLBL :   S1
ITMLBL :   M1        NEWLBL :   S3
ITMLBL :   S3        NEWLBL :   D2
ITMLBL :   D3        NEWLBL :   D1
ITMLBL :   V2        NEWLBL :   D3
ITMLBL :   V1        NEWLBL :   F1
ITMLBL :   F1        NEWLBL :   F2
ITMLBL :   M2        NEWLBL :   V2
ITMLBL :   F2        NEWLBL :   V1
```

THE MPNAM LIST

```
MPNAM( 1) = M1       MOTTYP( 1) =  2
MPNAM( 2) = M2       MOTTYP( 2) =  6
MPNAM( 3) = J2       MOTTYP( 3) =  2
MPNAM( 4) = J1       MOTTYP( 4) =  2
```

THE CNNAM LIST

```
CNNAM( 1) = S2      CNTYP( 1) =  1      IEPTYP( 1) =  9
CNNAM( 2) = S1      CNTYP( 2) =  1      IEPTYP( 2) =  9
CNNAM( 3) = S3      CNTYP( 3) =  1      IEPTYP( 3) = 11
```

```
CNNAM( 4) = D2       CNTYP( 4) =  3     IEPTYP( 4) =  9
CNNAM( 5) = D1       CNTYP( 5) =  3     IEPTYP( 5) =  9
CNNAM( 6) = D3       CNTYP( 6) =  3     IEPTYP( 6) = 11
CNNAM( 7) = F1       CNTYP( 7) =  4     IEPTYP( 7) =  1
CNNAM( 8) = F2       CNTYP( 8) =  4     IEPTYP( 8) =  2
CNNAM( 9) = V2       CNTYP( 9) =  5     IEPTYP( 9) =  1
CNNAM(10) = V1       CNTYP(10) =  5     IEPTYP(10) =  1
```

MOTION POINTS ON CONNECTORS :

```
MPONCN( 1,1) =  2          MPONCN( 1,2) =  3
MPONCN( 2,1) =  2          MPONCN( 2,2) =  4
MPONCN( 3,1) =  1          MPONCN( 3,2) =  2
MPONCN( 4,1) =  2          MPONCN( 4,2) =  3
MPONCN( 5,1) =  2          MPONCN( 5,2) =  4
MPONCN( 6,1) =  1          MPONCN( 6,2) =  2
MPONCN( 7,1) =  1          MPONCN( 7,2) =  0
MPONCN( 8,1) =  2          MPONCN( 8,2) =  0
MPONCN( 9,1) =  3          MPONCN( 9,2) =  0
MPONCN(10,1) =  4          MPONCN(10,2) =  0
```

THE CPTR LIST

```
     1      4     11     14     17
```

THE CNONMP ARRAY

```
     3      6      7      2      5      8      3      6      1      4
     1      4      9      2      5     10      0      0      0      0
```

LOCAL GEOMETRY FOR THE MASSES.

MOTION POINT : M1

```
  CONNECTOR : S3    X-DISTANCE =   0.00    Y-DISTANCE =  -0.50

  CONNECTOR : D3    X-DISTANCE =   0.00    Y-DISTANCE =  -0.50

  CONNECTOR : F1    X-DISTANCE =   0.00    Y-DISTANCE =   0.00
```

MOTION POINT : M2

```
  CONNECTOR : S1    X-DISTANCE =  -3.00    Y-DISTANCE =   0.00

  CONNECTOR : D1    X-DISTANCE =  -3.00    Y-DISTANCE =   0.00
```

F 3

| | | | | |
|---|---|---|---|---|
| CONNECTOR : F2 | X-DISTANCE = | 0.00 | Y-DISTANCE = | 0.00 |
| CONNECTOR : S3 | X-DISTANCE = | 1.00 | Y-DISTANCE = | 0.00 |
| CONNECTOR : D3 | X-DISTANCE = | 1.00 | Y-DISTANCE = | 0.00 |
| CONNECTOR : S2 | X-DISTANCE = | 3.00 | Y-DISTANCE = | 0.00 |
| CONNECTOR : D2 | X-DISTANCE = | 3.00 | Y-DISTANCE = | 0.00 |

THE BOND GRAPH.

```
NODES ATTACHED TO BOND   1 ARE    1    6
NODES ATTACHED TO BOND   2 ARE    2    7
NODES ATTACHED TO BOND   3 ARE    3    8
NODES ATTACHED TO BOND   4 ARE   10    9
NODES ATTACHED TO BOND   5 ARE   11   10
NODES ATTACHED TO BOND   6 ARE   12   11
NODES ATTACHED TO BOND   7 ARE   13   12
NODES ATTACHED TO BOND   8 ARE    3   13
NODES ATTACHED TO BOND   9 ARE   14   10
NODES ATTACHED TO BOND  10 ARE   15   14
NODES ATTACHED TO BOND  11 ARE   16   15
NODES ATTACHED TO BOND  12 ARE    2   16
NODES ATTACHED TO BOND  13 ARE   17   16
NODES ATTACHED TO BOND  14 ARE    3   17
NODES ATTACHED TO BOND  15 ARE   10   18
NODES ATTACHED TO BOND  16 ARE   18   19
NODES ATTACHED TO BOND  17 ARE   19    4
NODES ATTACHED TO BOND  18 ARE   21   20
NODES ATTACHED TO BOND  19 ARE   22   21
NODES ATTACHED TO BOND  20 ARE   23   22
NODES ATTACHED TO BOND  21 ARE   24   23
NODES ATTACHED TO BOND  22 ARE    3   24
NODES ATTACHED TO BOND  23 ARE   25   21
NODES ATTACHED TO BOND  24 ARE   26   25
NODES ATTACHED TO BOND  25 ARE   27   26
NODES ATTACHED TO BOND  26 ARE    2   27
NODES ATTACHED TO BOND  27 ARE   28   27
NODES ATTACHED TO BOND  28 ARE    3   28
NODES ATTACHED TO BOND  29 ARE   21   29
NODES ATTACHED TO BOND  30 ARE   29   30
NODES ATTACHED TO BOND  31 ARE   30    5
NODES ATTACHED TO BOND  32 ARE   32   31
NODES ATTACHED TO BOND  33 ARE   33   32
NODES ATTACHED TO BOND  34 ARE   34   33
NODES ATTACHED TO BOND  35 ARE    1   34
NODES ATTACHED TO BOND  36 ARE   32   35
NODES ATTACHED TO BOND  37 ARE   35   36
NODES ATTACHED TO BOND  38 ARE   36   37
NODES ATTACHED TO BOND  39 ARE   37    3
NODES ATTACHED TO BOND  40 ARE   32   38
NODES ATTACHED TO BOND  41 ARE   38   39
```

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| NODES | ATTACHED | TO | BOND | 42 | ARE | 39 | 40 |
| NODES | ATTACHED | TO | BOND | 43 | ARE | 40 | 2 |
| NODES | ATTACHED | TO | BOND | 44 | ARE | 40 | 41 |
| NODES | ATTACHED | TO | BOND | 45 | ARE | 41 | 3 |
| NODES | ATTACHED | TO | BOND | 46 | ARE | 43 | 42 |
| NODES | ATTACHED | TO | BOND | 47 | ARE | 44 | 43 |
| NODES | ATTACHED | TO | BOND | 48 | ARE | 45 | 44 |
| NODES | ATTACHED | TO | BOND | 49 | ARE | 46 | 45 |
| NODES | ATTACHED | TO | BOND | 50 | ARE | 3 | 46 |
| NODES | ATTACHED | TO | BOND | 51 | ARE | 47 | 43 |
| NODES | ATTACHED | TO | BOND | 52 | ARE | 48 | 47 |
| NODES | ATTACHED | TO | BOND | 53 | ARE | 49 | 48 |
| NODES | ATTACHED | TO | BOND | 54 | ARE | 2 | 49 |
| NODES | ATTACHED | TO | BOND | 55 | ARE | 50 | 49 |
| NODES | ATTACHED | TO | BOND | 56 | ARE | 3 | 50 |
| NODES | ATTACHED | TO | BOND | 57 | ARE | 43 | 51 |
| NODES | ATTACHED | TO | BOND | 58 | ARE | 51 | 52 |
| NODES | ATTACHED | TO | BOND | 59 | ARE | 52 | 4 |
| NODES | ATTACHED | TO | BOND | 60 | ARE | 54 | 53 |
| NODES | ATTACHED | TO | BOND | 61 | ARE | 55 | 54 |
| NODES | ATTACHED | TO | BOND | 62 | ARE | 56 | 55 |
| NODES | ATTACHED | TO | BOND | 63 | ARE | 57 | 56 |
| NODES | ATTACHED | TO | BOND | 64 | ARE | 3 | 57 |
| NODES | ATTACHED | TO | BOND | 65 | ARE | 58 | 54 |
| NODES | ATTACHED | TO | BOND | 66 | ARE | 59 | 58 |
| NODES | ATTACHED | TO | BOND | 67 | ARE | 60 | 59 |
| NODES | ATTACHED | TO | BOND | 68 | ARE | 2 | 60 |
| NODES | ATTACHED | TO | BOND | 69 | ARE | 61 | 60 |
| NODES | ATTACHED | TO | BOND | 70 | ARE | 3 | 61 |
| NODES | ATTACHED | TO | BOND | 71 | ARE | 54 | 62 |
| NODES | ATTACHED | TO | BOND | 72 | ARE | 62 | 63 |
| NODES | ATTACHED | TO | BOND | 73 | ARE | 63 | 5 |
| NODES | ATTACHED | TO | BOND | 74 | ARE | 65 | 64 |
| NODES | ATTACHED | TO | BOND | 75 | ARE | 66 | 65 |
| NODES | ATTACHED | TO | BOND | 76 | ARE | 67 | 66 |
| NODES | ATTACHED | TO | BOND | 77 | ARE | 1 | 67 |
| NODES | ATTACHED | TO | BOND | 78 | ARE | 65 | 68 |
| NODES | ATTACHED | TO | BOND | 79 | ARE | 68 | 69 |
| NODES | ATTACHED | TO | BOND | 80 | ARE | 69 | 70 |
| NODES | ATTACHED | TO | BOND | 81 | ARE | 70 | 3 |
| NODES | ATTACHED | TO | BOND | 82 | ARE | 65 | 71 |
| NODES | ATTACHED | TO | BOND | 83 | ARE | 71 | 72 |
| NODES | ATTACHED | TO | BOND | 84 | ARE | 72 | 73 |
| NODES | ATTACHED | TO | BOND | 85 | ARE | 73 | 2 |
| NODES | ATTACHED | TO | BOND | 86 | ARE | 73 | 74 |
| NODES | ATTACHED | TO | BOND | 87 | ARE | 74 | 3 |
| NODES | ATTACHED | TO | BOND | 88 | ARE | 76 | 75 |
| NODES | ATTACHED | TO | BOND | 89 | ARE | 1 | 76 |
| NODES | ATTACHED | TO | BOND | 90 | ARE | 78 | 77 |
| NODES | ATTACHED | TO | BOND | 91 | ARE | 79 | 78 |
| NODES | ATTACHED | TO | BOND | 92 | ARE | 80 | 79 |
| NODES | ATTACHED | TO | BOND | 93 | ARE | 81 | 80 |
| NODES | ATTACHED | TO | BOND | 94 | ARE | 3 | 81 |
| NODES | ATTACHED | TO | BOND | 95 | ARE | 82 | 78 |
| NODES | ATTACHED | TO | BOND | 96 | ARE | 83 | 82 |

```
NODES ATTACHED TO BOND  97 ARE     84        83
NODES ATTACHED TO BOND  98 ARE      2        84
NODES ATTACHED TO BOND  99 ARE     85        84
NODES ATTACHED TO BOND 100 ARE      3        85
NODES ATTACHED TO BOND 101 ARE     87        86
NODES ATTACHED TO BOND 102 ARE      4        87
NODES ATTACHED TO BOND 103 ARE     89        88
NODES ATTACHED TO BOND 104 ARE      5        89

IELLST(  1) =   7
IELLST(  2) =   7
IELLST(  3) =   7
IELLST(  4) =   7
IELLST(  5) =   7
IELLST(  6) =   2
IELLST(  7) =   2
IELLST(  8) =   2
IELLST(  9) =   1
IELLST( 10) =   6
IELLST( 11) =   8
IELLST( 12) =   7
IELLST( 13) =   8
IELLST( 14) =   8
IELLST( 15) =   7
IELLST( 16) =   6
IELLST( 17) =   8
IELLST( 18) =   8
IELLST( 19) =   7
IELLST( 20) =   1
IELLST( 21) =   6
IELLST( 22) =   8
IELLST( 23) =   7
IELLST( 24) =   8
IELLST( 25) =   8
IELLST( 26) =   7
IELLST( 27) =   6
IELLST( 28) =   8
IELLST( 29) =   8
IELLST( 30) =   7
IELLST( 31) =   1
IELLST( 32) =   6
IELLST( 33) =   8
IELLST( 34) =   7
IELLST( 35) =   8
IELLST( 36) =   7
IELLST( 37) =   8
IELLST( 38) =   8
IELLST( 39) =   7
IELLST( 40) =   6
IELLST( 41) =   8
IELLST( 42) =   3
IELLST( 43) =   6
IELLST( 44) =   8
IELLST( 45) =   7
IELLST( 46) =   8
```

```
IELLST( 47) =   8
IELLST( 48) =   7
IELLST( 49) =   6
IELLST( 50) =   8
IELLST( 51) =   8
IELLST( 52) =   7
IELLST( 53) =   3
IELLST( 54) =   6
IELLST( 55) =   8
IELLST( 56) =   7
IELLST( 57) =   8
IELLST( 58) =   8
IELLST( 59) =   7
IELLST( 60) =   6
IELLST( 61) =   8
IELLST( 62) =   8
IELLST( 63) =   7
IELLST( 64) =   3
IELLST( 65) =   6
IELLST( 66) =   8
IELLST( 67) =   7
IELLST( 68) =   8
IELLST( 69) =   7
IELLST( 70) =   8
IELLST( 71) =   8
IELLST( 72) =   7
IELLST( 73) =   6
IELLST( 74) =   8
IELLST( 75) =   4
IELLST( 76) =   7
IELLST( 77) =   4
IELLST( 78) =   6
IELLST( 79) =   8
IELLST( 80) =   7
IELLST( 81) =   8
IELLST( 82) =   8
IELLST( 83) =   7
IELLST( 84) =   6
IELLST( 85) =   8
IELLST( 86) =   5
IELLST( 87) =   7
IELLST( 88) =   5
IELLST( 89) =   7
```

THE NPTR LIST

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 5 | 13 | 28 | 31 | 34 | 35 | 36 | 37 | 38 |
| 42 | 44 | 46 | 48 | 50 | 52 | 55 | 57 | 59 | 61 |
| 62 | 66 | 68 | 70 | 72 | 74 | 76 | 79 | 81 | 83 |
| 85 | 86 | 90 | 92 | 94 | 96 | 98 | 100 | 102 | 104 |
| 107 | 109 | 110 | 114 | 116 | 118 | 120 | 122 | 124 | 127 |
| 129 | 131 | 133 | 134 | 138 | 140 | 142 | 144 | 146 | 148 |
| 151 | 153 | 155 | 157 | 158 | 162 | 164 | 166 | 168 | 170 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 172 | 174 | 176 | 179 | 181 | 182 | 184 | 185 | 188 | 190 |
| 192 | 194 | 196 | 198 | 201 | 203 | 204 | 206 | 207 | 209 |

THE NBIMX ARRAY

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 89 | 77 | 35 | 1 | 98 | 85 | 68 | 54 | 43 | 26 |
| 12 | 2 | 100 | 94 | 87 | 81 | 70 | 64 | 56 | 50 |
| 45 | 39 | 28 | 22 | 14 | 8 | 3 | 102 | 59 | 17 |
| 104 | 73 | 31 | 1 | 2 | 3 | 4 | 15 | 9 | 5 |
| 4 | 6 | 5 | 7 | 6 | 8 | 7 | 10 | 9 | 11 |
| 10 | 13 | 12 | 11 | 14 | 13 | 16 | 15 | 17 | 16 |
| 18 | 29 | 23 | 19 | 18 | 20 | 19 | 21 | 20 | 22 |
| 21 | 24 | 23 | 25 | 24 | 27 | 26 | 25 | 28 | 27 |
| 30 | 29 | 31 | 30 | 32 | 40 | 36 | 33 | 32 | 34 |
| 33 | 35 | 34 | 37 | 36 | 38 | 37 | 39 | 38 | 41 |
| 40 | 42 | 41 | 44 | 43 | 42 | 45 | 44 | 46 | 57 |
| 51 | 47 | 46 | 48 | 47 | 49 | 48 | 50 | 49 | 52 |
| 51 | 53 | 52 | 55 | 54 | 53 | 56 | 55 | 58 | 57 |
| 59 | 58 | 60 | 71 | 65 | 61 | 60 | 62 | 61 | 63 |
| 62 | 64 | 63 | 66 | 65 | 67 | 66 | 69 | 68 | 67 |
| 70 | 69 | 72 | 71 | 73 | 72 | 74 | 82 | 78 | 75 |
| 74 | 76 | 75 | 77 | 76 | 79 | 78 | 80 | 79 | 81 |
| 80 | 83 | 82 | 84 | 83 | 86 | 85 | 84 | 87 | 86 |
| 88 | 89 | 88 | 90 | 95 | 91 | 90 | 92 | 91 | 93 |
| 92 | 94 | 93 | 96 | 95 | 97 | 96 | 99 | 98 | 97 |
| 100 | 99 | 101 | 102 | 101 | 103 | 104 | 103 | 0 | |

POINTER FROM MP NODES TO XMP.

IXMPPT( 1) = 2
IXMPPT( 2) = 5
IXMPPT( 3) = 6
IXMPPT( 4) = 8
IXMPPT( 5) = 11

POINTER FROM EP XMP TO NODE NUMBER

NODE ASSOCIATED WITH EP 1 = IXEPPT( 1) = 12
NODE ASSOCIATED WITH EP 2 = IXEPPT( 2) = 15
NODE ASSOCIATED WITH EP 3 = IXEPPT( 3) = 0
NODE ASSOCIATED WITH EP 4 = IXEPPT( 4) = 19
NODE ASSOCIATED WITH EP 5 = IXEPPT( 5) = 23
NODE ASSOCIATED WITH EP 6 = IXEPPT( 6) = 26
NODE ASSOCIATED WITH EP 7 = IXEPPT( 7) = 0
NODE ASSOCIATED WITH EP 8 = IXEPPT( 8) = 30
NODE ASSOCIATED WITH EP 9 = IXEPPT( 9) = 0
NODE ASSOCIATED WITH EP 10 = IXEPPT(10) = 34
NODE ASSOCIATED WITH EP 11 = IXEPPT(11) = 36
NODE ASSOCIATED WITH EP 12 = IXEPPT(12) = 39

```
NODE ASSOCIATED WITH EP 13 = IXEPPT(13) =  45
NODE ASSOCIATED WITH EP 14 = IXEPPT(14) =  48
NODE ASSOCIATED WITH EP 15 = IXEPPT(15) =   0
NODE ASSOCIATED WITH EP 16 = IXEPPT(16) =  52
NODE ASSOCIATED WITH EP 17 = IXEPPT(17) =  56
NODE ASSOCIATED WITH EP 18 = IXEPPT(18) =  59
NODE ASSOCIATED WITH EP 19 = IXEPPT(19) =   0
NODE ASSOCIATED WITH EP 20 = IXEPPT(20) =  63
NODE ASSOCIATED WITH EP 21 = IXEPPT(21) =   0
NODE ASSOCIATED WITH EP 22 = IXEPPT(22) =  67
NODE ASSOCIATED WITH EP 23 = IXEPPT(23) =  69
NODE ASSOCIATED WITH EP 24 = IXEPPT(24) =  72
NODE ASSOCIATED WITH EP 25 = IXEPPT(25) =   0
NODE ASSOCIATED WITH EP 26 = IXEPPT(26) =  76
NODE ASSOCIATED WITH EP 27 = IXEPPT(27) =   0
NODE ASSOCIATED WITH EP 28 = IXEPPT(28) =   0
NODE ASSOCIATED WITH EP 29 = IXEPPT(29) =  80
NODE ASSOCIATED WITH EP 30 = IXEPPT(30) =  83
NODE ASSOCIATED WITH EP 31 = IXEPPT(31) =   0
NODE ASSOCIATED WITH EP 32 = IXEPPT(32) =   0
NODE ASSOCIATED WITH EP 33 = IXEPPT(33) =   0
NODE ASSOCIATED WITH EP 34 = IXEPPT(34) =  87
NODE ASSOCIATED WITH EP 35 = IXEPPT(35) =   0
NODE ASSOCIATED WITH EP 36 = IXEPPT(36) =   0
NODE ASSOCIATED WITH EP 37 = IXEPPT(37) =   0
NODE ASSOCIATED WITH EP 38 = IXEPPT(38) =  89
NODE ASSOCIATED WITH EP 39 = IXEPPT(39) =   0
NODE ASSOCIATED WITH EP 40 = IXEPPT(40) =   0


 POINTER FROM CONNECTOR TO ITS NODE NUMBER

NODE NUMBER FOR CONNECTOR  1 = IXCNPT( 1) =   9
NODE NUMBER FOR CONNECTOR  2 = IXCNPT( 2) =  20
NODE NUMBER FOR CONNECTOR  3 = IXCNPT( 3) =  31
NODE NUMBER FOR CONNECTOR  4 = IXCNPT( 4) =  42
NODE NUMBER FOR CONNECTOR  5 = IXCNPT( 5) =  53
NODE NUMBER FOR CONNECTOR  6 = IXCNPT( 6) =  64
NODE NUMBER FOR CONNECTOR  7 = IXCNPT( 7) =  75
NODE NUMBER FOR CONNECTOR  8 = IXCNPT( 8) =  77
NODE NUMBER FOR CONNECTOR  9 = IXCNPT( 9) =  86
NODE NUMBER FOR CONNECTOR 10 = IXCNPT(10) =  88


 TRANSFORMERS AND ASSOCIATED DATA.

MTF( 1) =  11      FUNCTION NUMBER = 5.
   ARGUMENTS :  1.    2.    3.    4.
   MODULATOR FUNCTION : 1.
   NUMBER OF POINTERS = 3.
     CONNECTOR POINTER =  1.
     MTF POINTERS      :   3.     5.     0.
```

```
MTF( 2) =  13       FUNCTION NUMBER = 1.
   CONSTANTS :  3.     0.     ARGUMENT :  6.

MTF( 3) =  14       FUNCTION NUMBER = 5.
   ARGUMENTS :  1.     2.     3.     4.
   MODULATOR FUNCTION : 2.
   NUMBER OF POINTERS = 3.
     CONNECTOR POINTER =  1.
     MTF POINTERS       :   1.     5.     0.

MTF( 4) =  17       FUNCTION NUMBER = 2.
   CONSTANTS :  3.     0.     ARGUMENT :  6.

MTF( 5) =  18       FUNCTION NUMBER = 5.
   ARGUMENTS :  1.     2.     3.     4.
   MODULATOR FUNCTION : 4.
   NUMBER OF POINTERS = 3.
     CONNECTOR POINTER =  1.
     MTF POINTERS       :   1.     3.     0.

MTF( 6) =  22       FUNCTION NUMBER = 5.
   ARGUMENTS :  5.     6.     7.     8.
   MODULATOR FUNCTION : 1.
   NUMBER OF POINTERS = 3.
     CONNECTOR POINTER =  2.
     MTF POINTERS       :   8.    10.     0.

MTF( 7) =  24       FUNCTION NUMBER = 1.
   CONSTANTS : -3.     0.     ARGUMENT :  6.

MTF( 8) =  25       FUNCTION NUMBER = 5.
   ARGUMENTS :  5.     6.     7.     8.
   MODULATOR FUNCTION : 2.
   NUMBER OF POINTERS = 3.
     CONNECTOR POINTER =  2.
     MTF POINTERS       :   6.    10.     0.

MTF( 9) =  28       FUNCTION NUMBER = 2.
   CONSTANTS : -3.     0.     ARGUMENT :  6.

MTF(10) =  29       FUNCTION NUMBER = 5.
   ARGUMENTS :  5.     6.     7.     8.
   MODULATOR FUNCTION : 4.
   NUMBER OF POINTERS = 3.
     CONNECTOR POINTER =  2.
     MTF POINTERS       :   6.     8.     0.

MTF(11) =  33       FUNCTION NUMBER = 5.
   ARGUMENTS :  9.    10.    11.    12.
   MODULATOR FUNCTION : 2.
   NUMBER OF POINTERS = 3.
     CONNECTOR POINTER =  3.
     MTF POINTERS       :  12.    14.     0.
```

```
MTF(12) =  35       FUNCTION NUMBER = 5.
   ARGUMENTS :  9.   10.   11.   12.
   MODULATOR FUNCTION : 3.
   NUMBER OF POINTERS = 3.
     CONNECTOR POINTER =  3.
     MTF POINTERS       :  11.    14.     0.

MTF(13) =  37       FUNCTION NUMBER = 1.
   CONSTANTS :  1.    0.    ARGUMENT :  6.

MTF(14) =  38       FUNCTION NUMBER = 5.
   ARGUMENTS :  9.   10.   11.   12.
   MODULATOR FUNCTION : 4.
   NUMBER OF POINTERS = 3.
     CONNECTOR POINTER =  3.
     MTF POINTERS       :  11.    12.     0.

MTF(15) =  41       FUNCTION NUMBER = 2.
   CONSTANTS :  1.    0.    ARGUMENT :  6.

MTF(16) =  44       FUNCTION NUMBER = 5.
   ARGUMENTS : 13.   14.   15.   16.
   MODULATOR FUNCTION : 1.
   NUMBER OF POINTERS = 3.
     CONNECTOR POINTER =  4.
     MTF POINTERS       :  18.    20.     0.

MTF(17) =  46       FUNCTION NUMBER = 1.
   CONSTANTS :  3.    0.    ARGUMENT :  6.

MTF(18) =  47       FUNCTION NUMBER = 5.
   ARGUMENTS : 13.   14.   15.   16.
   MODULATOR FUNCTION : 2.
   NUMBER OF POINTERS = 3.
     CONNECTOR POINTER =  4.
     MTF POINTERS       :  16.    20.     0.

MTF(19) =  50       FUNCTION NUMBER = 2.
   CONSTANTS :  3.    0.    ARGUMENT :  6.

MTF(20) =  51       FUNCTION NUMBER = 5.
   ARGUMENTS : 13.   14.   15.   16.
   MODULATOR FUNCTION : 4.
   NUMBER OF POINTERS = 3.
     CONNECTOR POINTER =  4.
     MTF POINTERS       :  16.    18.     0.

MTF(21) =  55       FUNCTION NUMBER = 5.
   ARGUMENTS : 17.   18.   19.   20.
   MODULATOR FUNCTION : 1.
   NUMBER OF POINTERS = 3.
     CONNECTOR POINTER =  5.
     MTF POINTERS       :  23.    25.     0.

MTF(22) =  57       FUNCTION NUMBER = 1.
   CONSTANTS : -3.    0.    ARGUMENT :  6.
```

```
MTF(23) =  58      FUNCTION NUMBER = 5.
   ARGUMENTS : 17.   18.   19.   20.
   MODULATOR FUNCTION : 2.
   NUMBER OF POINTERS = 3.
     CONNECTOR POINTER =  5.
     MTF POINTERS      :  21.    25.     0.

MTF(24) =  61      FUNCTION NUMBER = 2.
   CONSTANTS : -3.    0.    ARGUMENT :  6.

MTF(25) =  62      FUNCTION NUMBER = 5.
   ARGUMENTS : 17.   18.   19.   20.
   MODULATOR FUNCTION : 4.
   NUMBER OF POINTERS = 3.
     CONNECTOR POINTER =  5.
     MTF POINTERS      :  21.    23.     0.

MTF(26) =  66      FUNCTION NUMBER = 5.
   ARGUMENTS : 21.   22.   23.   24.
   MODULATOR FUNCTION : 2.
   NUMBER OF POINTERS = 3.
     CONNECTOR POINTER =  6.
     MTF POINTERS      :  27.    29.     0.

MTF(27) =  68      FUNCTION NUMBER = 5.
   ARGUMENTS : 21.   22.   23.   24.
   MODULATOR FUNCTION : 3.
   NUMBER OF POINTERS = 3.
     CONNECTOR POINTER =  6.
     MTF POINTERS      :  26.    29.     0.

MTF(28) =  70      FUNCTION NUMBER = 1.
   CONSTANTS :  1.    0.    ARGUMENT :  6.

MTF(29) =  71      FUNCTION NUMBER = 5.
   ARGUMENTS : 21.   22.   23.   24.
   MODULATOR FUNCTION : 4.
   NUMBER OF POINTERS = 3.
     CONNECTOR POINTER =  6.
     MTF POINTERS      :  26.    27.     0.

MTF(30) =  74      FUNCTION NUMBER = 2.
   CONSTANTS :  1.    0.    ARGUMENT :  6.

MTF(31) =  79      FUNCTION NUMBER = 5.
   ARGUMENTS : 29.   30.   31.   32.
   MODULATOR FUNCTION : 1.
   NUMBER OF POINTERS = 2.
     CONNECTOR POINTER =  8.
     MTF POINTERS      :  33.     0.     0.

MTF(32) =  81      FUNCTION NUMBER = 1.
   CONSTANTS :  0.    0.    ARGUMENT :  6.
```

```
MTF(33) =  82       FUNCTION NUMBER = 5.
   ARGUMENTS : 29.   30.   31.   32.
   MODULATOR FUNCTION : 2.
   NUMBER OF POINTERS = 2.
     CONNECTOR POINTER =  8.
     MTF POINTERS      :  31.     0.     0.

MTF(34) =  85       FUNCTION NUMBER = 2.
   CONSTANTS :  0.    0.    ARGUMENT :  6.
```