ERRATA

Jackson, James H., SELMA: A Conversational System for the Graphical Specification of Markovian Queueing Networks.

Page 18, 9th line from bottom should read:
which are associated with the merge. Whenever an item is available...

Page 19, 8th line through 5th line from bottom should read:
probability 0.3. Whenever the server in Figure 8d is not full, an item is placed in it from the bottom queue if the bottom queue is not empty, or from the top queue if this queue is not empty and the bottom queue is empty.

Page 24: Replace with Corrected Figure 10 (see next page).

Page 29, Line 3 should read:
phase and results phase to provide this function. When this light button is referenced with...

Page 32, 8th line from the bottom should read:
For the example in Figure 13, this result is shown in Figure 14a. QAS then returns the information necessary to display the graph, and SELMA generates the graph and displays it on these axes, as shown in Figure 14b.

Page 34, 7th line from the bottom should read:
DETAIL light button allows the user to specify the maximum and minimum...

Page 37, 12th line from the bottom should read:
are used to save the current model on a file at the IBM 360/67 and to retrieve...

6th line from the bottom should read:
command is terminated by a carriage return from the keyboard.)

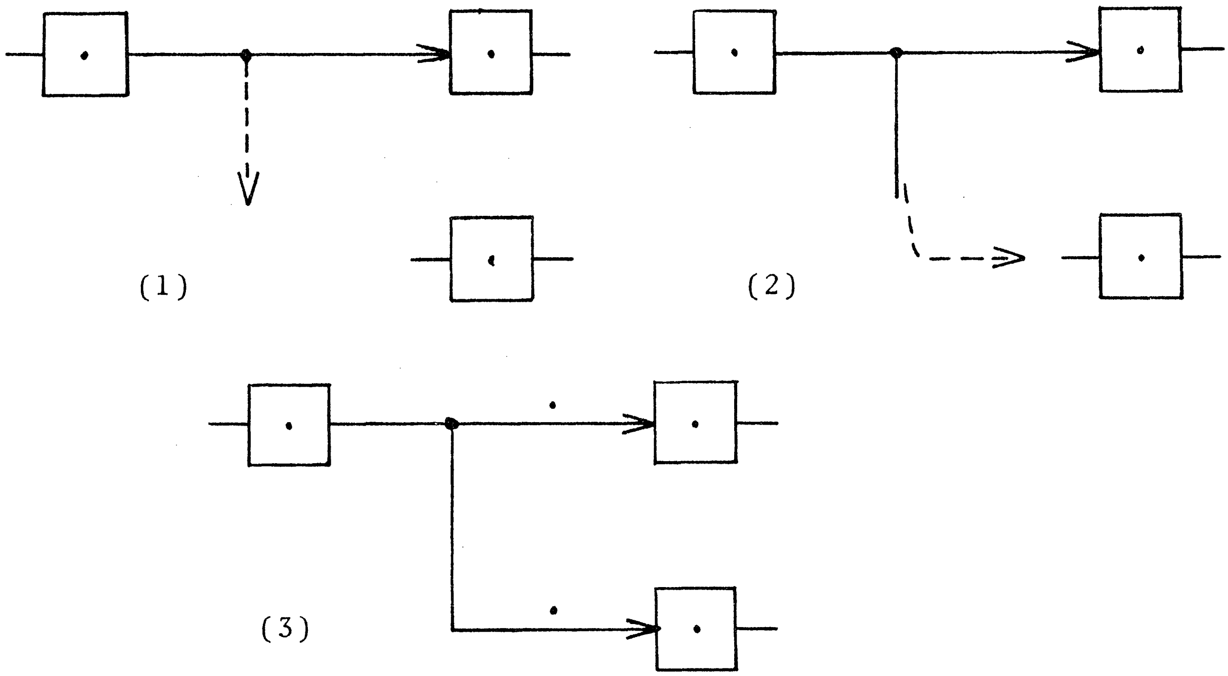
Page 40, 4th line from the bottom should read:
the SEL Executive System until the command exchanger has removed...

Page 44, Line 8 should read:
01   02   T,T[2]   Display single value

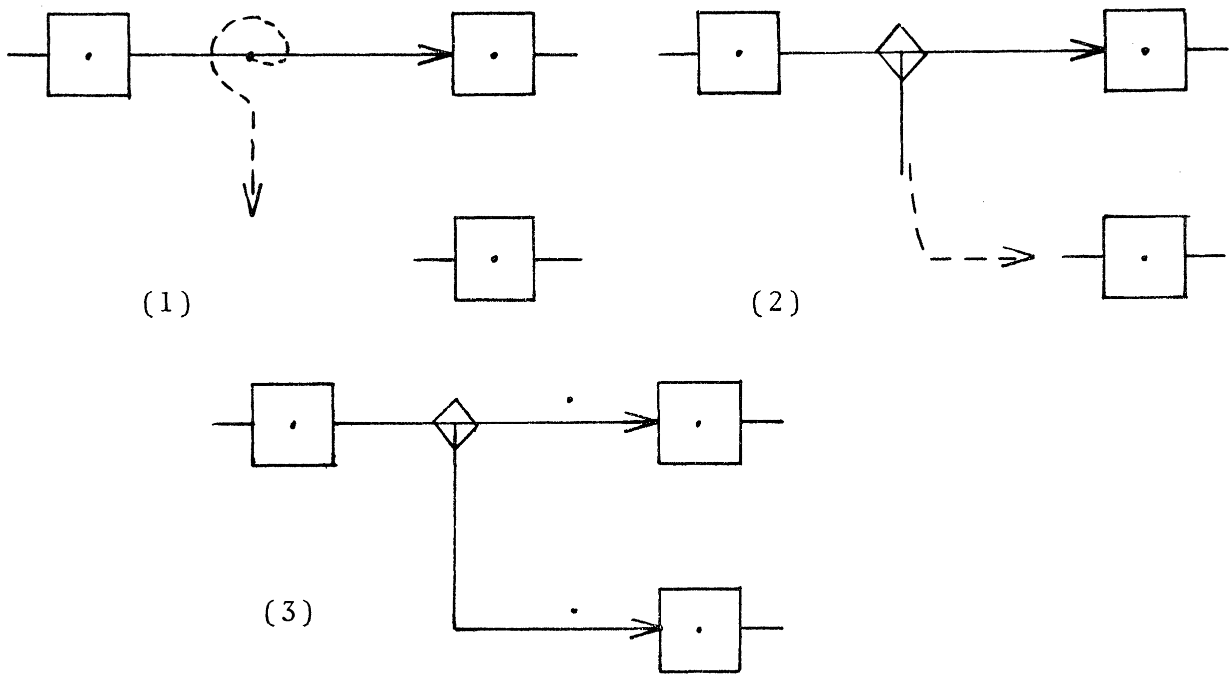
Footnote 2 should appear as follows:
[2]Abscissa value, ordinate value

Footnote 4 should not appear.

Page 65:
The word "EXIT" in the figure should be written "Exit".



a. Random Branch



b. Priority Branch

Corrected Figure 10.

Generation of Branches

THE UNIVERSITY OF MICHIGAN

Technical Report 23

SELMA: A Conversational System for the Graphical Specification of Markovian Queueing Networks

James H. Jackson

CONCOMP: Research in Conversational Use of Computers
F. H. Westervelt, Director
ORA Project 07449

supported by:

ADVANCED RESEARCH PROJECTS AGENCY
DEPARTMENT OF DEFENSE
WASHINGTON, D. C.

CONTRACT NO. DA-49-083 OSA-3050
ARPA ORDER NO. 716

administered through:

OFFICE OF RESEARCH ADMINISTRATION ANN ARBOR

October 1969

ABSTRACT

This report discusses the design and use of the Systems Engineering Laboratory's Markovian Analyzer (SELMA) system for a DEC 339 computer display terminal. This system provides interactive graphics support for a program which was developed concurrently for the IBM 360/67 to analyze a class of Markovian queueing networks. Special features of the system include handling of all graphic operations at the terminal and recognition of patterns of motion of the light pen to provide a human-oriented drawing capability.

TABLE OF CONTENTS

LIST OF FIGURES

1. Introduction

This report discusses the design and use of the Systems Engineering Laboratory's Markovian Analyzer (SELMA) system for a DEC 339 [1] computer display terminal. The purpose of this system is to provide interactive graphic support for the QAS program [2], which was developed concurrently for the IBM 360/67 to analyze a class of Markovian queueing models. SELMA consists of a set of user tasks which are executed in a multiprogrammed fashion under the control of the SEL Executive System [3].

A simplified representation of the tasks which comprise SELMA is shown in Figure 1. Each task is represented by a rectangle, whereas each of the major display structures is represented by a circle. Each directed path on the diagram represents a flow of information from one unit to another.

Whenever SELMA is operating, both the command exchanger task and the keyboard interpreter task are continuously executed. The function of the command exchanger is to communicate with the QAS program in the IBM 360/67 via a 201A dataphone. Because the bandwidth of this dataphone is somewhat limited (2000 bits/sec), all graphic operations are performed locally in the display terminal, and all data sent to QAS or received from QAS are formatted into blocks called commands. Each command represents a macroscopic operation, such as creating or destroying an element, and it contains only the required control information, rather than an excessive

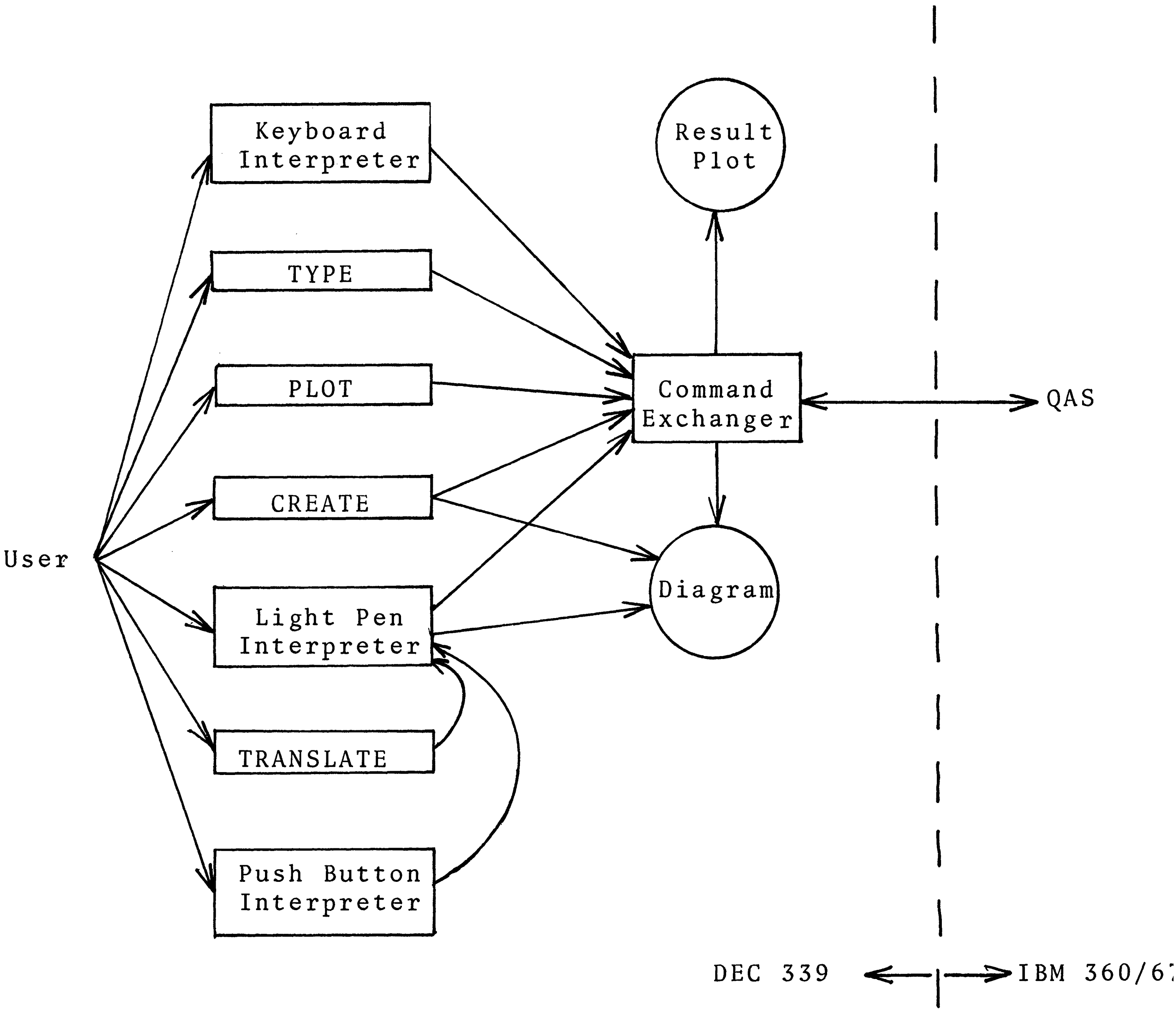


Figure 1

Simplified SELMA Structure

amount of graphic information. The basic function of the keyboard interpreter is to provide a direct path between the user and the command exchanger for certain relatively infrequent operations such as destroying the current model or terminating execution.

The other tasks which are shown in Figure 1 are scheduled for execution in response to user inputs. The TYPE and PLOT tasks are scheduled in response to references to light buttons whose function is to request results from QAS. These tasks do not modify the diagram of the current model; they merely instruct the command exchanger to send appropriate commands to QAS. The CREATE task is scheduled in response to a reference to any one of several light buttons, each of which is used to create an element of a particular type. This task must both give the command exchanger information to send to QAS and modify the diagram to include the symbol for the created element. The other functions which are required in order to draw the network are handled by the light pen interpreter task. This task is scheduled in response to a reference to any part of the diagram with the light pen. The function which it performs is determined by the part of the diagram which is referenced, and, in many cases, by recognition of subsequent patterns of light pen motion. This recognition of patterns of light pen motion allows the user to express his intentions in a human-oriented fashion and it eliminates the need to clutter the screen with other light buttons or to sequence through a large set of light button menus. The TRANSLATE task and the push button interpreter task are used to modify the behavior of the light pen interpreter. The TRANSLATE task, which is invoked by a light button, modifies the behavior of the light pen interpreter so that the entire 75"× 75" "paper" on which the diagram is drawn may be moved under the display screen. In this way, a diagram which is larger then the display screen may be drawn. The push button interpreter accepts numerical parameter values from the push buttons and modifies the behavior of the light pen interpreter so that these values may be assigned at various attachment points on the diagram.

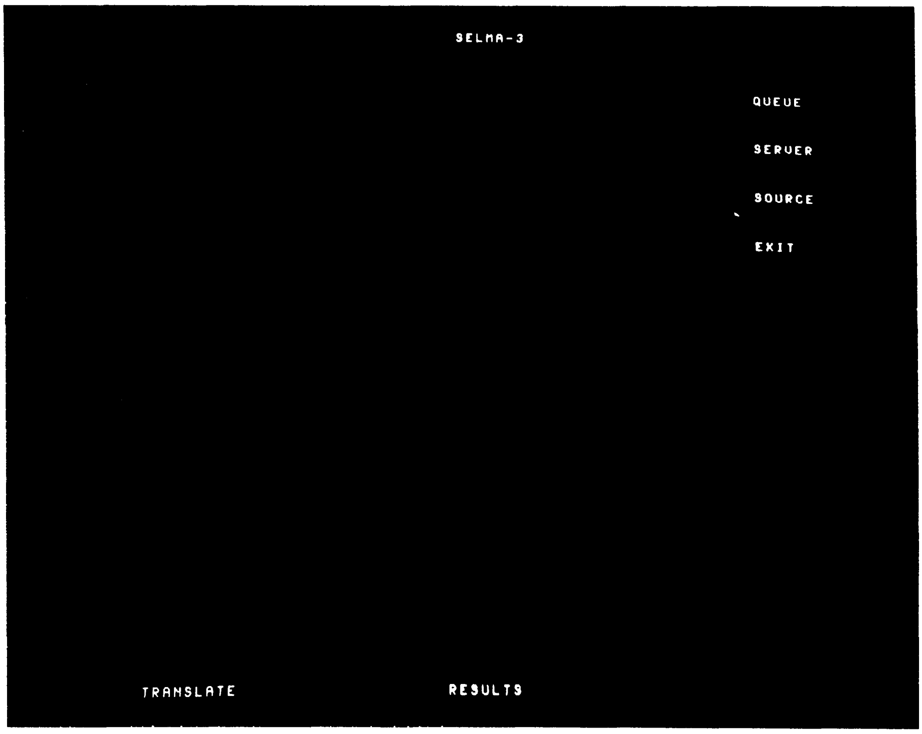
In the sections which follow, usage of the system, as well as the operations performed by the various tasks, is described. In Section 2, usage of the system is described, together with the operation of the light pen interpreter. The command exchanger is described in detail in Section 3, and, in Section 4, the display structure which is interrogated in order to form many of the commands which are sent to QAS is described. In Section 5, a feature of the command exchanger which should be useful while implementing future modifications of SELMA and/or QAS is described, and some forseeable modifications are discussed in Section 6. A listing of SELMA is supplied as an appendix.

2. Usage of SELMA and QAS

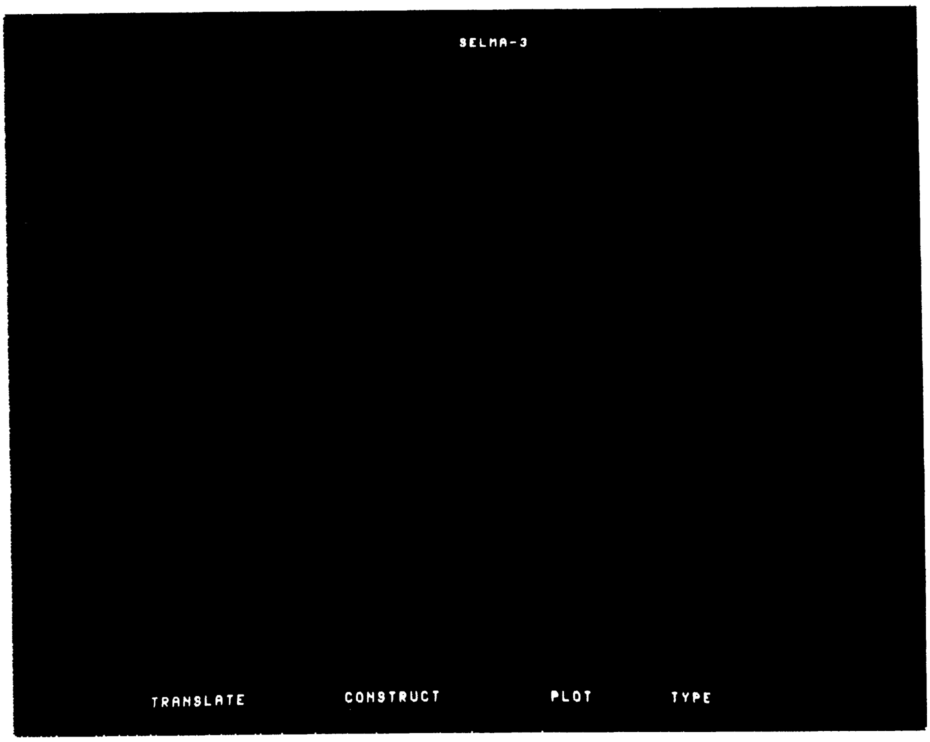
Whenever SELMA is to be used in conjunction with QAS, the command to run QAS must be given to the central time-sharing system before SELMA is started. This restriction is necessary because the SELMA command exchanger contains no provision for communicating with the command language interpreter of the time-sharing system. SELMA may then be started by scheduling and running the single task at the location whose address appears on the SELMA binary tape. (This address is not specified here because it is subject to change without notice.)

Just after SELMA is started, the light buttons shown in Figure 2a appear on the screen. When these light buttons are present, SELMA is said to be in the construction phase, and the light pen interpreter is adjusted so that a diagram of a queueing model may be drawn or modified. The other two phases indicated in Figure 2 are the results phase (Figure 2b) and the plot phase (Figure 2c). The results phase allows the user to request results, whereas the plot phase displays a graph of the last requested result and allows the user to expand sections of the graph and to request labels for points of interest on the graph.

Transitions between SELMA phases are depicted by Figure 3. In this figure, each phase is represented by a circle, and the light buttons which produce transitions between phases are indicated by

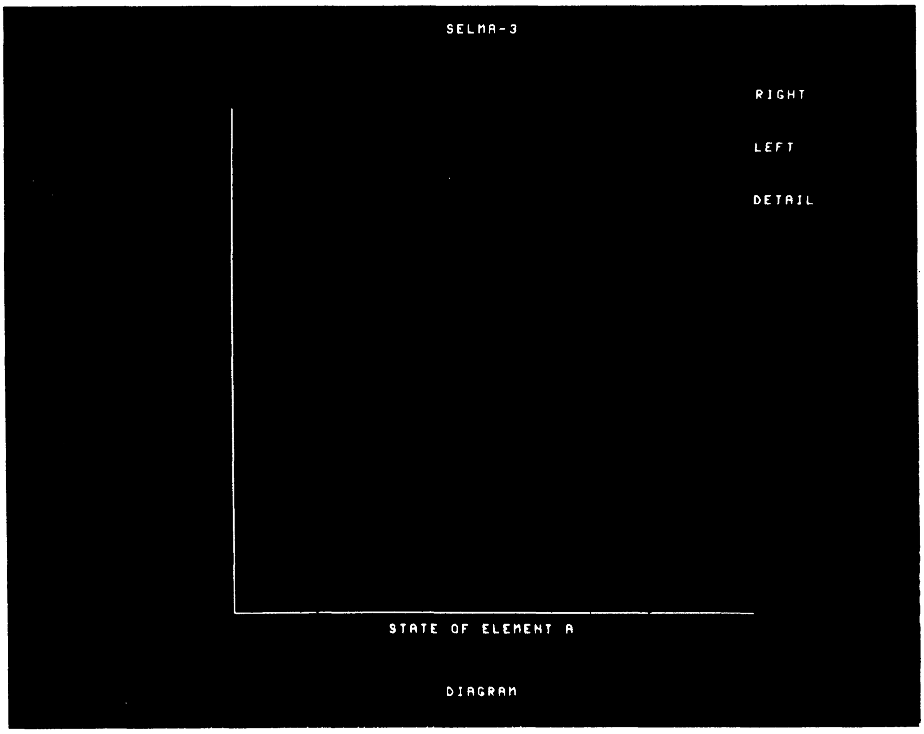


a. Construction Phase



b. Results Phase

Figure 2
SELMA Phases



c. Plot Phase

Figure 2
SELMA Phases (cont.)

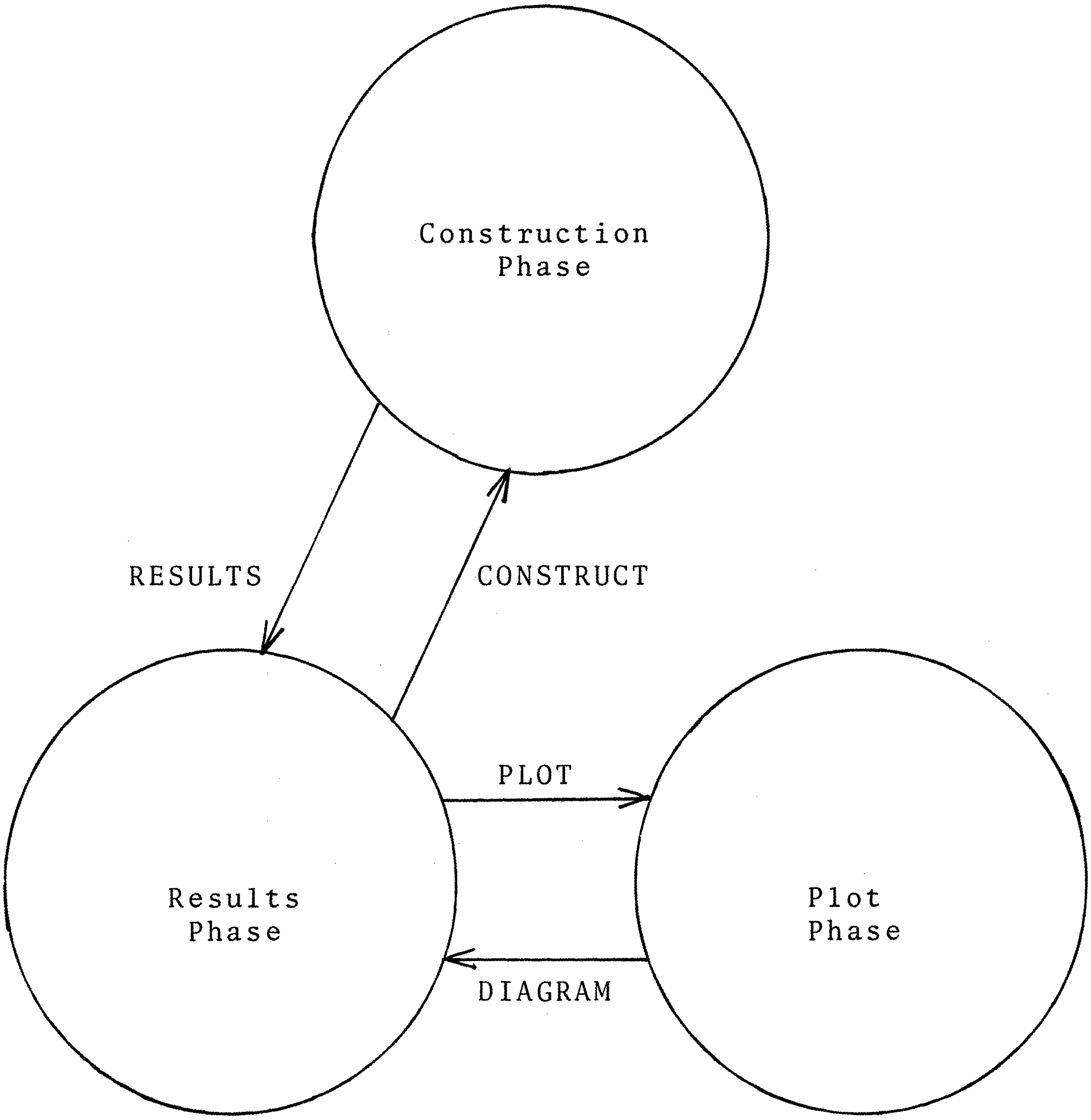


Figure 3

Phase Transitions

labels on directed paths between phases.

SELMA is placed in the construction phase when it is started so that the user may begin the construction of a new model. Before any other steps in the construction can be performed, at least one element must be created. An element is created by selecting the light button in the upper right-hand corner of the screen (Figure 2a) which denotes its type. A tracking cross appears with the element so that the element may be moved to any desired position on the screen.

The various elements which may be created are shown in Figure 4. The interpretation of each of these elements has been described previously [4,5]. Each element symbol includes one or more ports, i.e., attachment points for connections to other elements. In addition, an element symbol may have a body and one or more numerical parameters. Since no value is assumed for the parameter of an element when the element is created, the position of each parameter in a newly created symbol is indicated by a dot, called a parameter dot.

The creation of new symbols is the only operation required for the construction of a model which requires the use of light buttons. All other operations are performed with the aid of the light pen interpreter. As mentioned in the introduction, the action to be taken by the light pen interpreter is determined from the part of the diagram which is referenced and often from a recognition of subsequent patterns of motion of the light pen. The part of the diagram which is

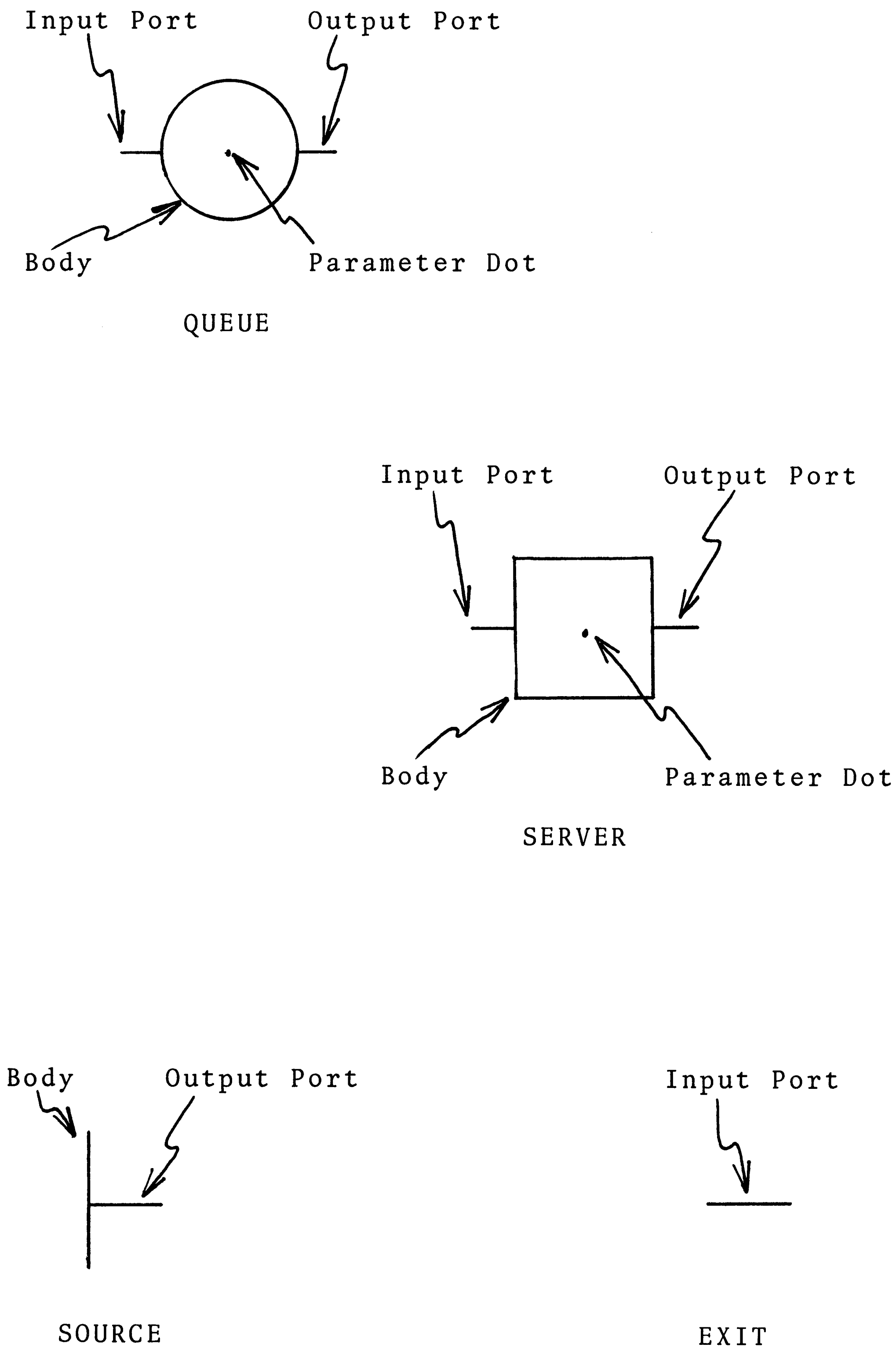


Figure 4

Element Symbols

referenced is easily determined from the display structure (Section 4) through the use of subroutines available in the executive system. However, since the executive system provides no facility for recognizing patterns of light pen motion, this latter determination is made completely within SELMA.

The method employed by SELMA to recognize patterns of light pen motion may be described with the aid of Figure 5. Whenever a light pen reference is made to a part of the diagram which requires that subsequent patterns of motion of the light pen be recognized, tracking is started at the coordinates of the light pen, and one or more threshold patterns, i.e., patterns of imaginary lines of the form shown in Figure 5, are placed on the screen. The number of patterns and the coordinates and size of each pattern are determined by the part of the diagram which is referenced. (The size of each pattern is specified by a parameter d.)

Whenever an element symbol is referenced with the light pen while SELMA is in the construction phase (and the light pen interpreter has not been modified by the TRANSLATE task or the push button interpreter), two threshold patterns are placed at the coordinates of the symbol as shown in Figure 6a, and a tracking cross is placed at the coordinates of the light pen. If the symbol has connected ports, the fragment of the diagram which consists of this symbol and all connections and symbols associated with it are moved with the tracking cross. Otherwise, the symbol may be either moved or deleted,

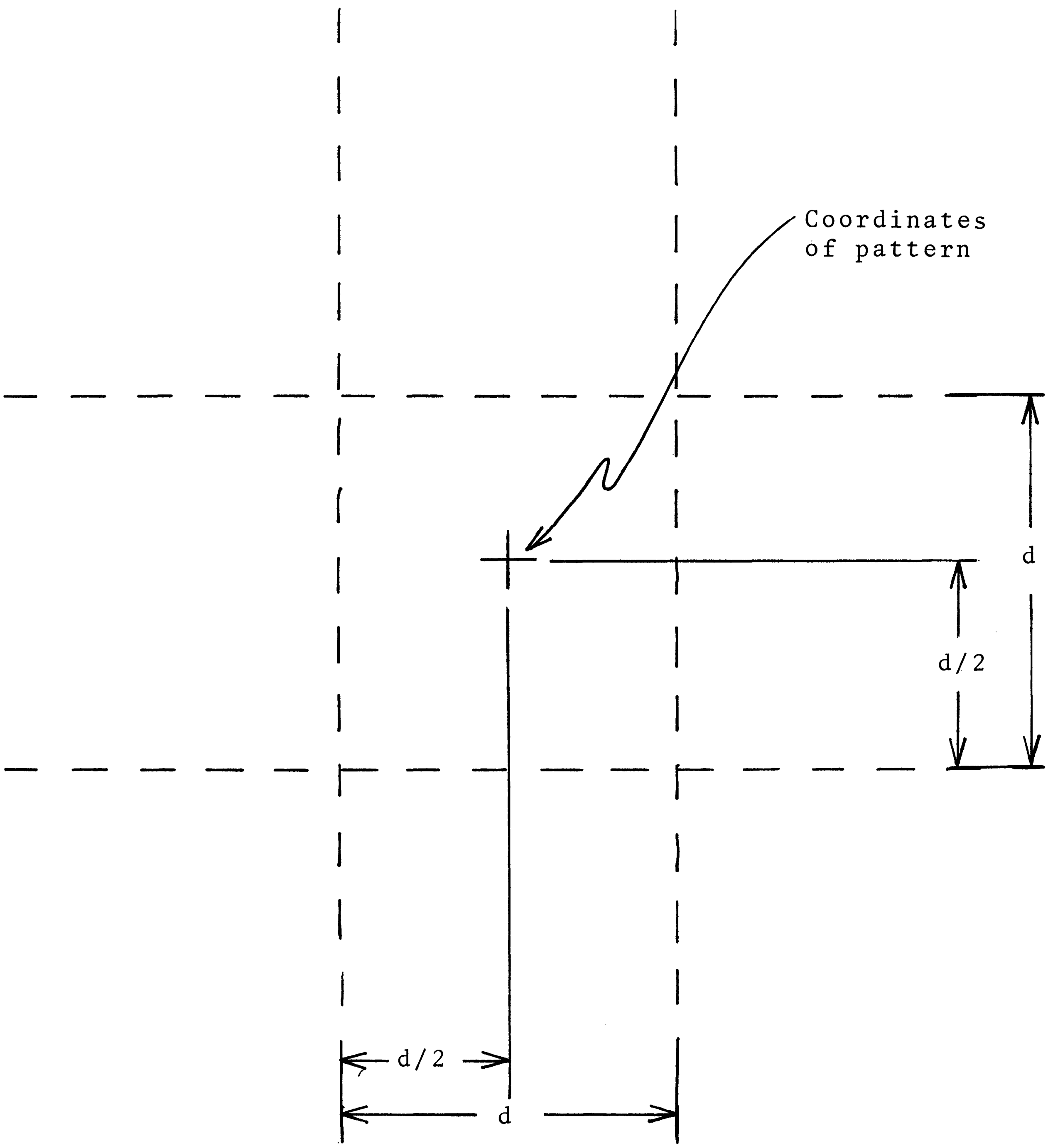
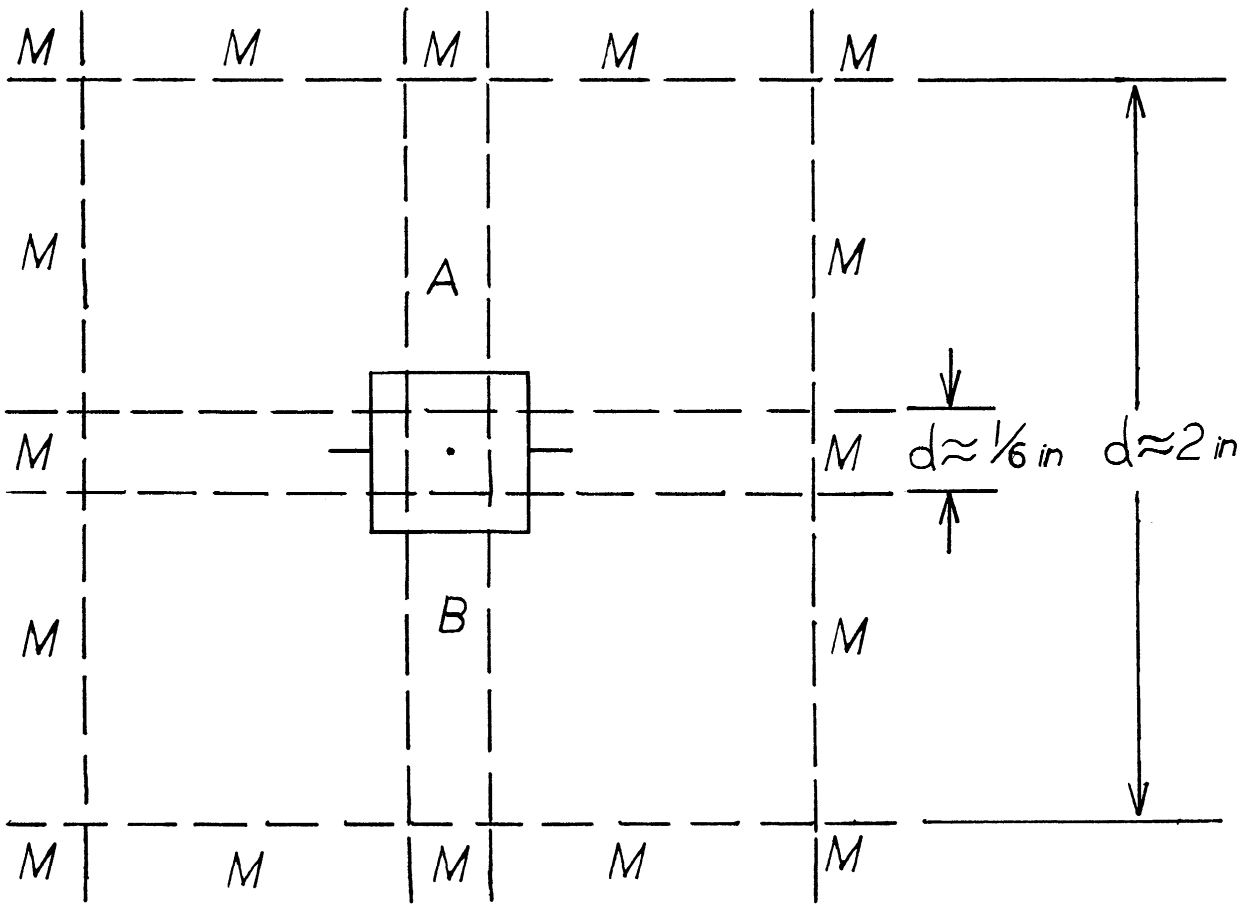
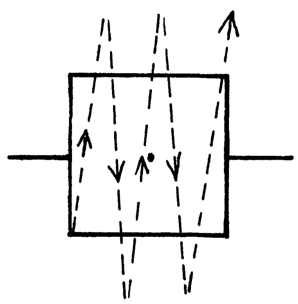


Figure 5

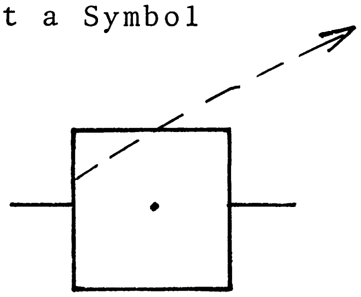
Threshold Pattern for Light Pen Motion Recognition



a. Threshold Patterns About a Symbol



b. Deleting a Symbol

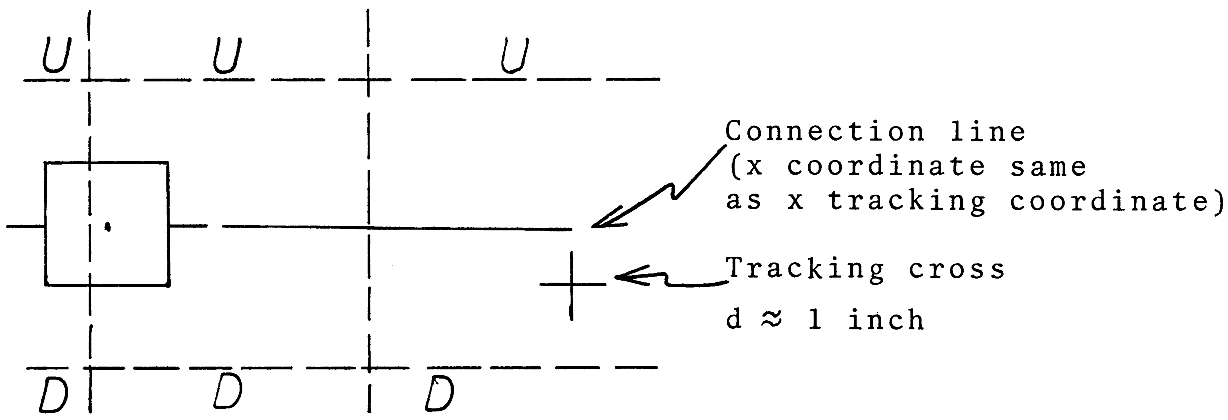


c. Moving a Symbol

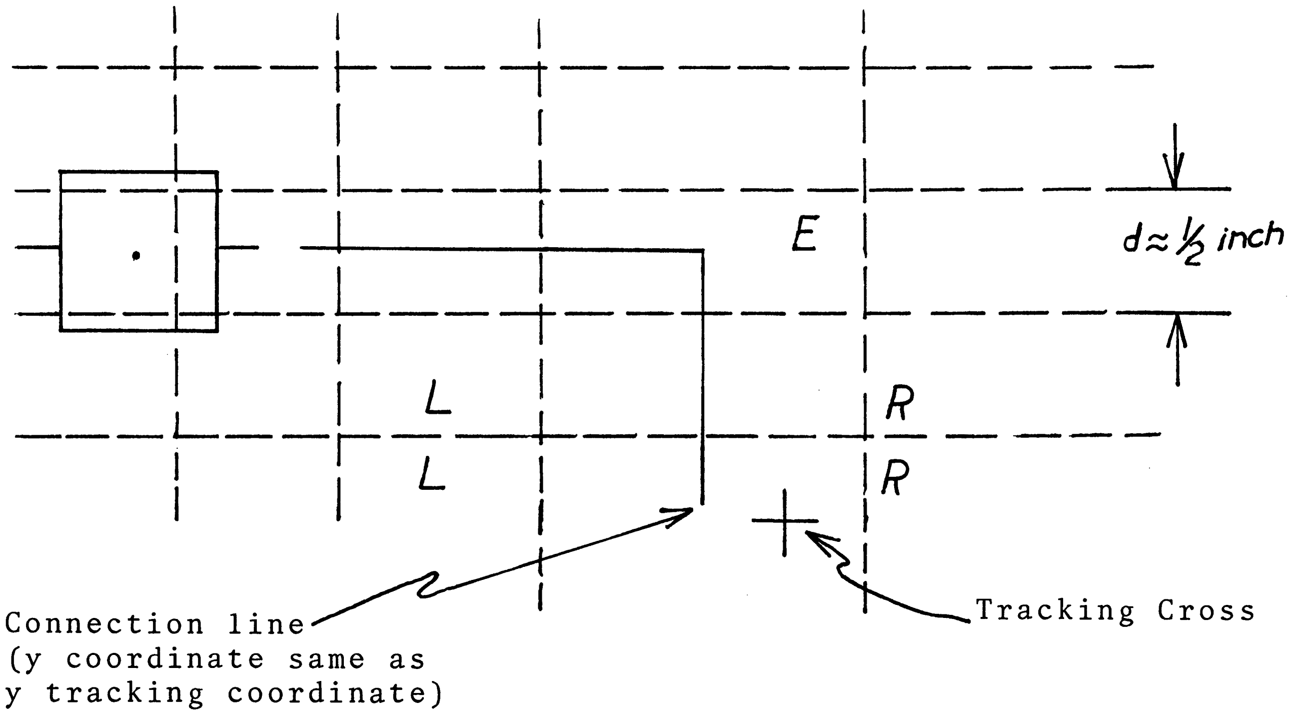
Figure 6

Light Pen Motion on Unconnected Element Symbols depending on subsequent motion of the light pen. In Figure 6a, the significant regions of the screen which are bounded by the lines of the threshold pattern and/or the edge of the screen are labeled A, B, or M. The decision to delete or move the symbol is made by recognizing transitions among these regions. The symbol is deleted if the pen is stroked vertically across the symbol as shown in Figure 6b. This motion is recognized as a sequence of five transitions between the A and B regions without entering an M region. The symbol is moved with the light pen if the light pen is moved into a M region. This latter motion is depicted in Figure 6c.

In the above discussion of deleting and moving symbols, a reference to a symbol with the light pen was assumed to be a reference to any part of the symbol other than an output port. When an unconnected output port is referenced on any symbol (regardless of whether or not the symbol has connected ports), the drawing of a connection line is begun as shown in Figure 7a. A threshold pattern is established at the coordinates of the output port. The connection line is continuously updated as shown until the light pen is moved into either a U or D region, at which time a new vertical segment is added to the connection line (Figure 7b). A smaller threshold pattern is then established at the coordinates of the output port. The original threshold pattern is moved to the newly created corner of the connection line. If the light pen is now moved into the E region, the vertical segment of the connection line is deleted, the smaller threshold pattern is deleted,



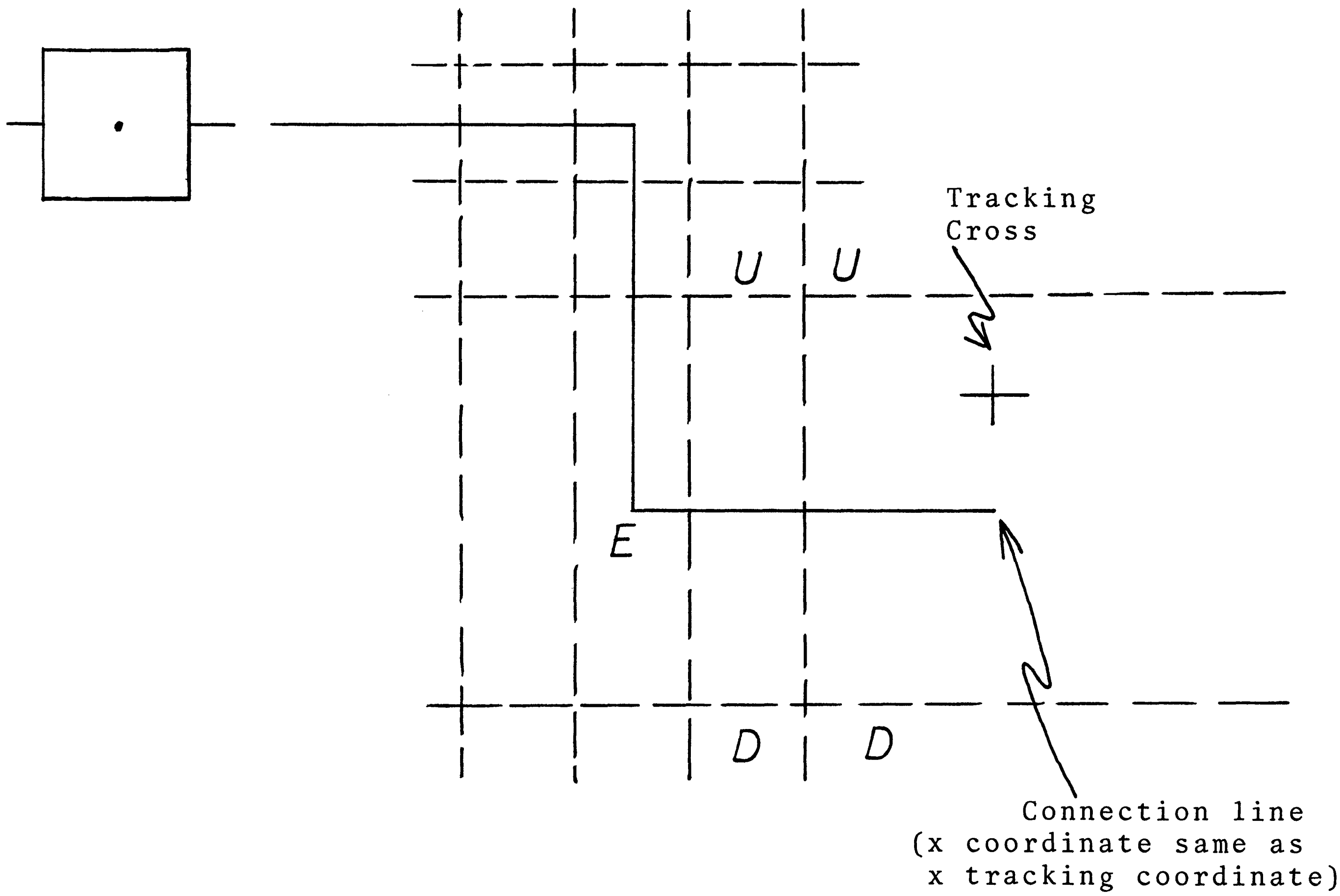
a. Start of Connection Line at Output Port



b. After First Corner

Figure 7

Drawing a Connection Line



c. After Second Corner

Figure 7

Drawing a Connection Line (Cont.)

the larger threshold pattern is moved to the coordinates of the output port, and action proceeds again as depicted in Figure 7a. If the light pen is moved into an L or R region, a second horizontal segment is added to the connection line as shown in Figure 7c. The small threshold pattern is moved to the next to last corner, and the larger threshold pattern is moved to the newly created corner. Again, if the light pen is moved into the E region, the last segment of the connection line is deleted and action proceeds according to Figure 7b. If the light pen is moved into a U or D region, a second vertical segment is added to the connection line. The operation proceeds in this manner until the drawing of the connection line is terminated by either of the following events:

1. The connection line is deleted by loss of tracking,
2. The connection line is used to form or modify a connection.

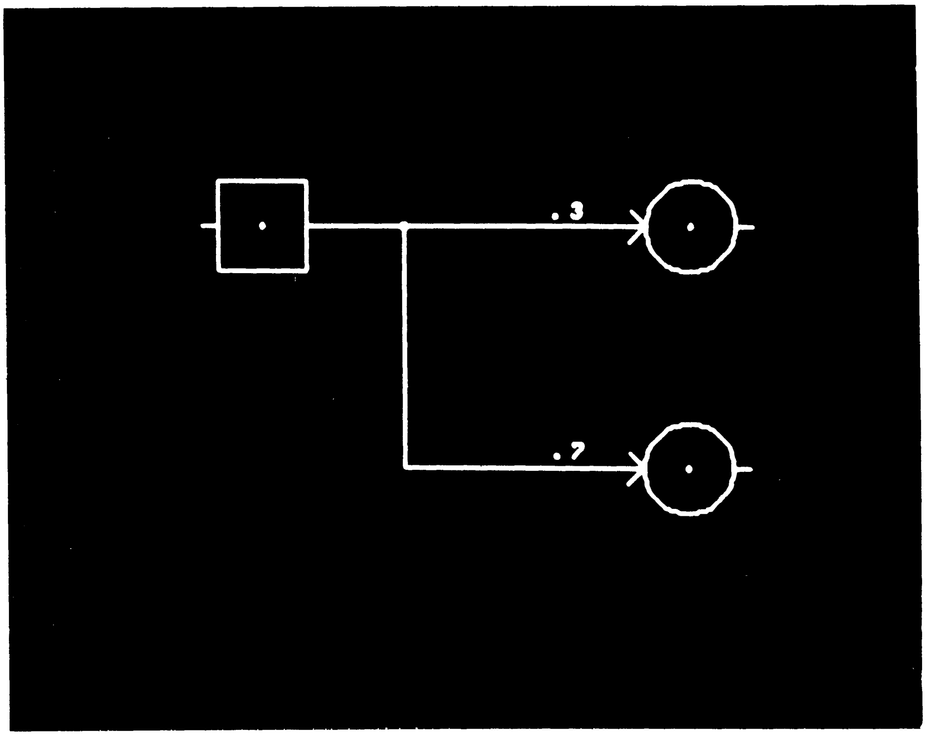
Connection lines may be used to form the following five types of connections:

1. Simple connection,
2. Random branch,
3. Priority branch,
4. Random merge, and
5. Priority merge.

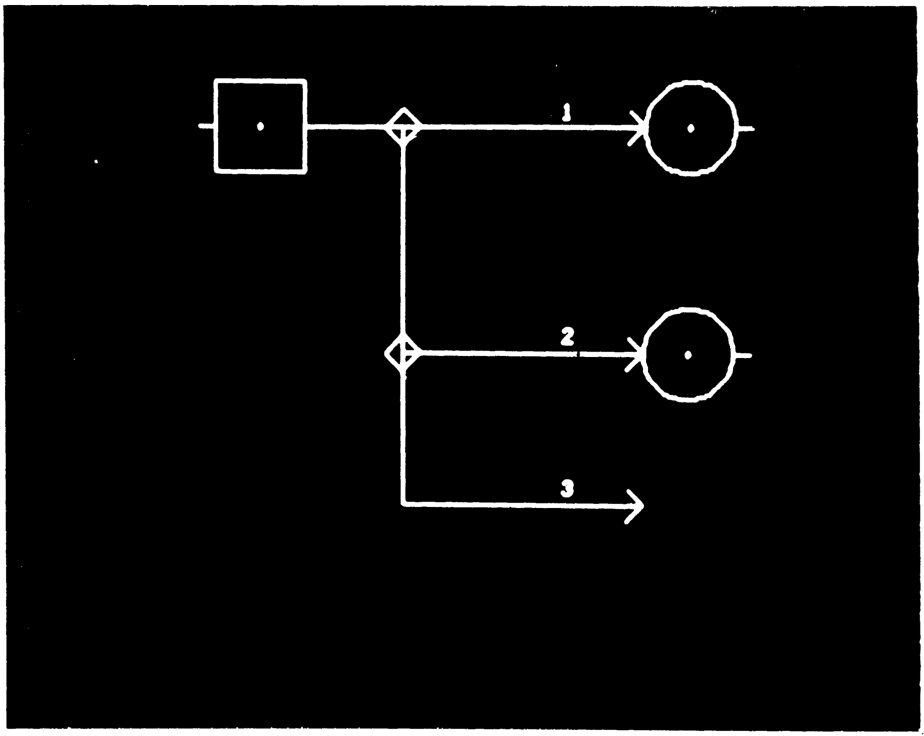
The first of these, the simple connection, associates an output port of an element with an input port of either the same or another element. An item flows from the output port through the connection to the input port whenever an item is available at the output port and the input port can accept it. A branch associates one output port with several input ports. Whenever an item is available at the output port and at least one input port can accept it, it flows through the branch to one of the input ports which can accept it. If the branch is a random branch, no input port can accept the item unless all input ports associated with the branch can accept it. Whenever all of these ports can accept the item, one of them is selected randomly in accordance with probabilities assigned to each of them. If the branch is a priority branch, an available item flows through the branch if any input port can accept it, and priorities assigned to the input ports determine which port receives the item. A merge associates several output ports with one input port. Whenever the input port can accept an item and an item becomes available at at least one output port, one of the available items flows through the merge to the input port. If the merge is a random merge, an item is not available at an output port unless items are available at all output ports which are associated with the branch. Whenever an item is available at all of these output ports and the input port can accept an item, one of the available items is selected randomly in accordance with probabilities assigned to each output port. If the merge is a priority merge, an available item flows through the merge if the input port can accept it, and priorities assigned to each output port determine which item is selected.

For both branches and merges, selection of ports on the basis of probability is indicated in a SELMA diagram by a dot (·), and selection of ports on the basis of priority is indicated by a diamond (<>). Some examples of branches and merges which are drawn with these symbols are shown in Figure 8. (The arrowheads on each branch are actually connected input ports, which differ in appearance from unconnected input ports.) In Figure 8a, whenever the server has finished servicing an item and neither of the queues is full, the item from the server is placed in the top queue with probability 0.3 or in the bottom queue with probability 0.7. The server in Figure 8b cannot hold an item after it has been serviced, for there is always an input port which can accept it. If the top queue is not full and the server finishes servicing an item, the item is placed in the top queue. Otherwise, if the top queue is full and the bottom queue is not, the item is placed in the bottom queue. If both queues are full, the item leaves the system through the sink. In Figure 8c, whenever the server is empty and no queue is empty, an item is placed in the server from the top queue with probability 0.2, from the middle queue with probability 0.5, or from the bottom queue with probability 0.3. Whenever the right-hand queue in Figure 8d is not full, an item is placed in it from the bottom left-hand queue if the bottom left-hand queue is not empty, or from the top left-hand queue if this queue is not empty and the bottom left-hand queue is empty.

All connections which are drawn with SELMA are originally generated as simple connections, which later may be extended to form branches or merges. A simple connection is drawn by starting a connection line at an unconnected output port and terminating it at an unconnected



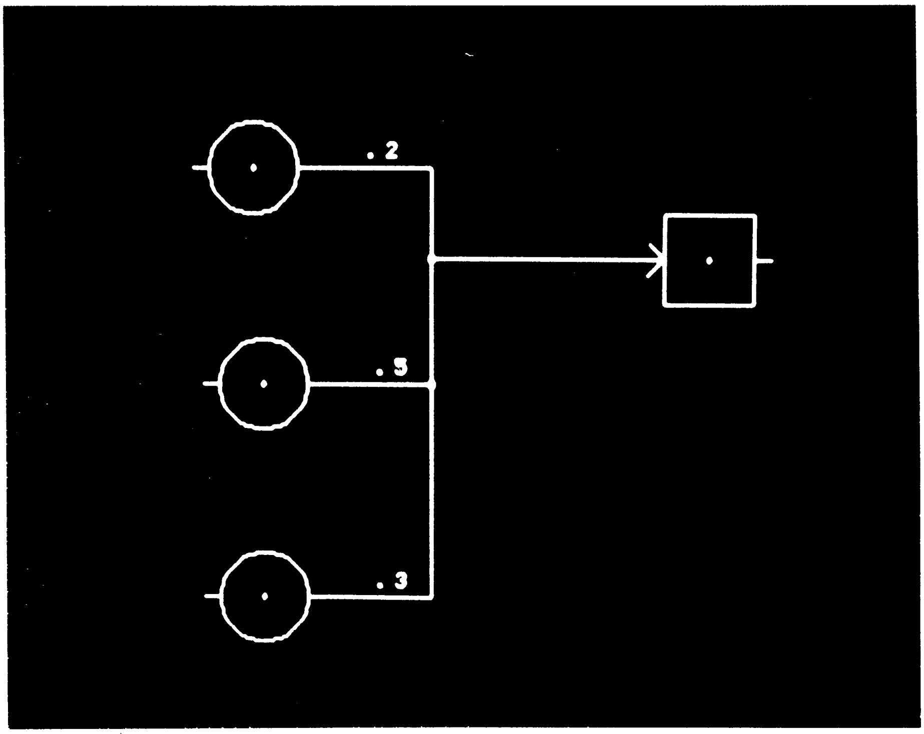
a. Random Branch



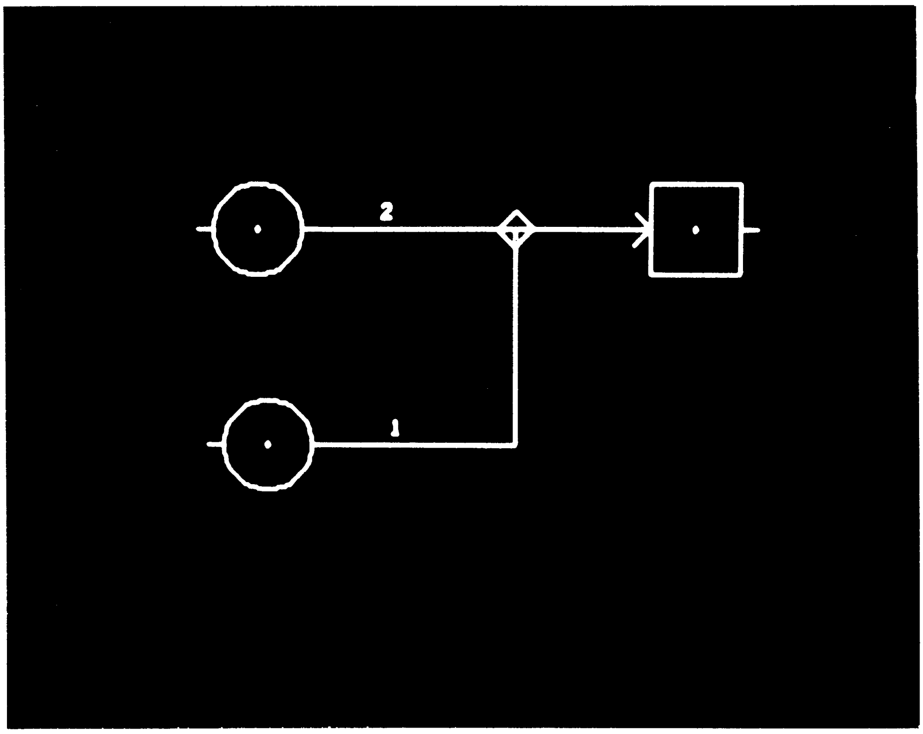
b. Priority Branch

Figure 8

Examples of Branches and Merges



c. Random Merge



d. Priority Merge

Figure 8 (Cont.)

input port by passing the light pen over the input port while drawing the connection line. SELMA will then stop tracking the light pen and will convert the connection line to a simple connection.

Whenever a simple connection is to be converted to a branch, the type of branch (i.e., random or priority) must be established. This is accomplished by recognizing a pattern of light pen motion. In addition, a simple connection may be deleted by stroking vertically across it in the same manner that one would use to delete an element symbol. The threshold patterns which are involved are shown in Figure 9. When the light pen is aimed at a connection, a dot appears on the connection at the coordinates at which the light pen was detected. If the connection is a simple connection and the light pen is then moved into an M region without entering a W, an X, a Y, and a Z region or moving back and forth five times between A and B regions, a connection line is started at this point, and the dot remains to indicate that the connection is to be converted to a random branch. The conversion is performed when the connection line is completed by passing the pen over an unconnected input port. This operation is shown in Figure 10a. If the light pen enters a W, an X, a Y, and a Z region without entering an M region or moving back and forth five times between A and B regions, the dot is converted to a diamond to indicate that the connection is to be converted to a priority branch. When the light pen is subsequently moved into an M region, a connection line is started in the center of the diamond. This operation is indicated in Figure 10b. If the light pen is moved back and

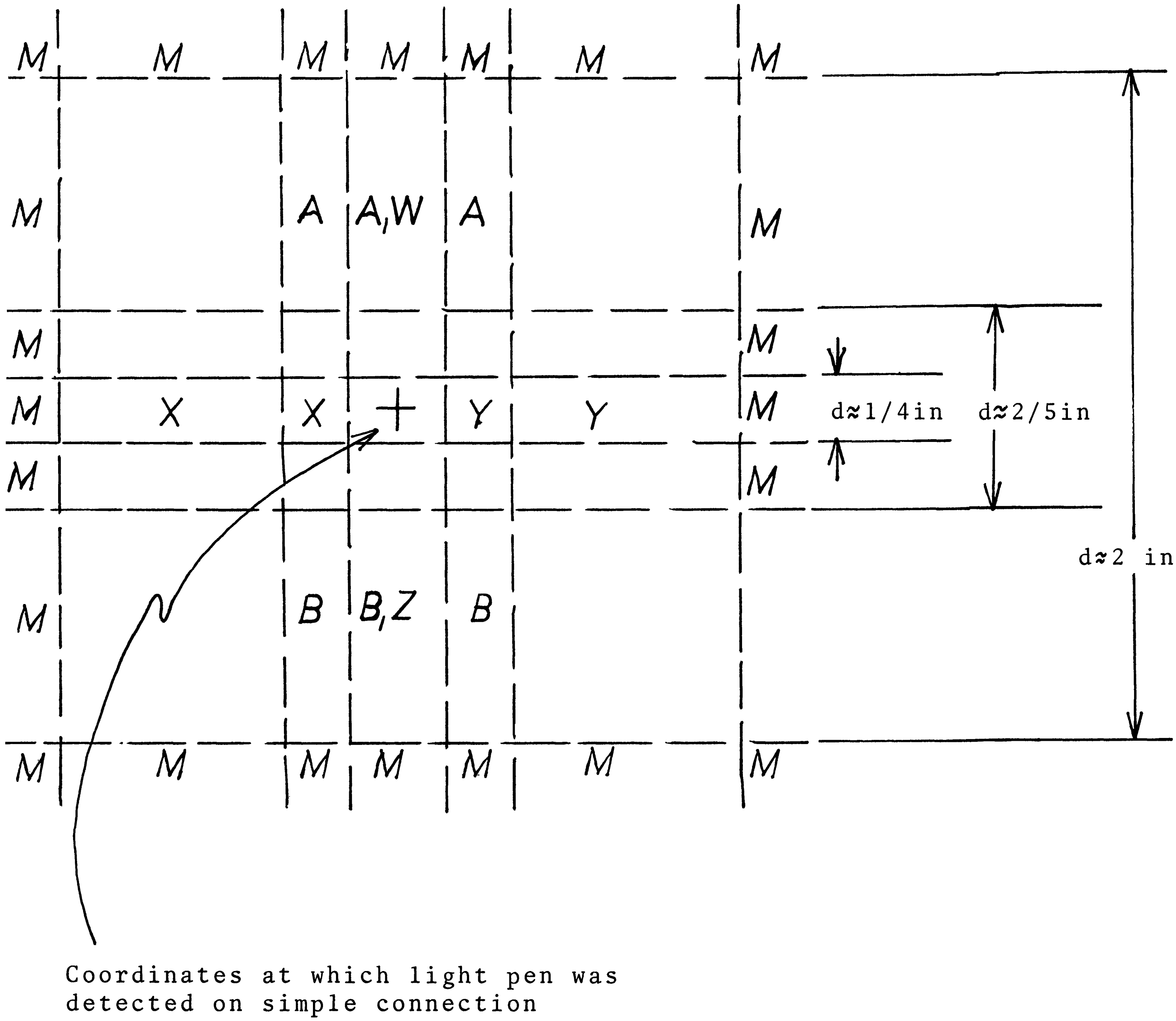
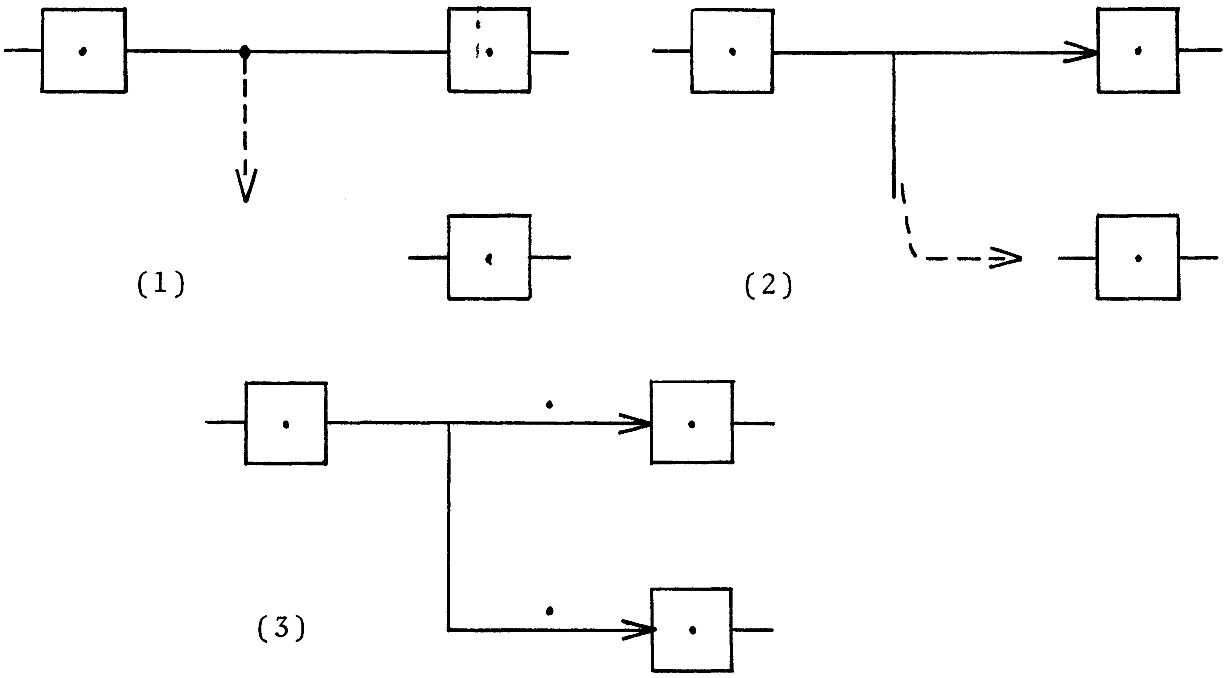
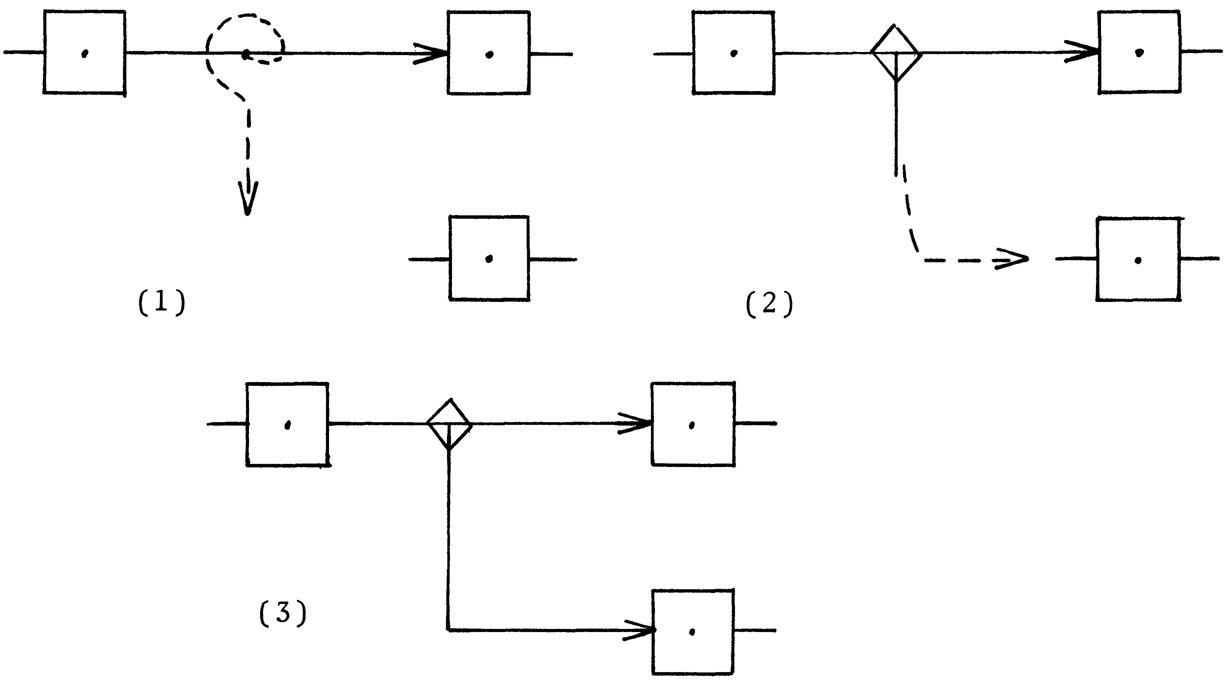


Figure 9

Threshold Patterns for Generating Branches



a. Random Branch



b. Priority Branch

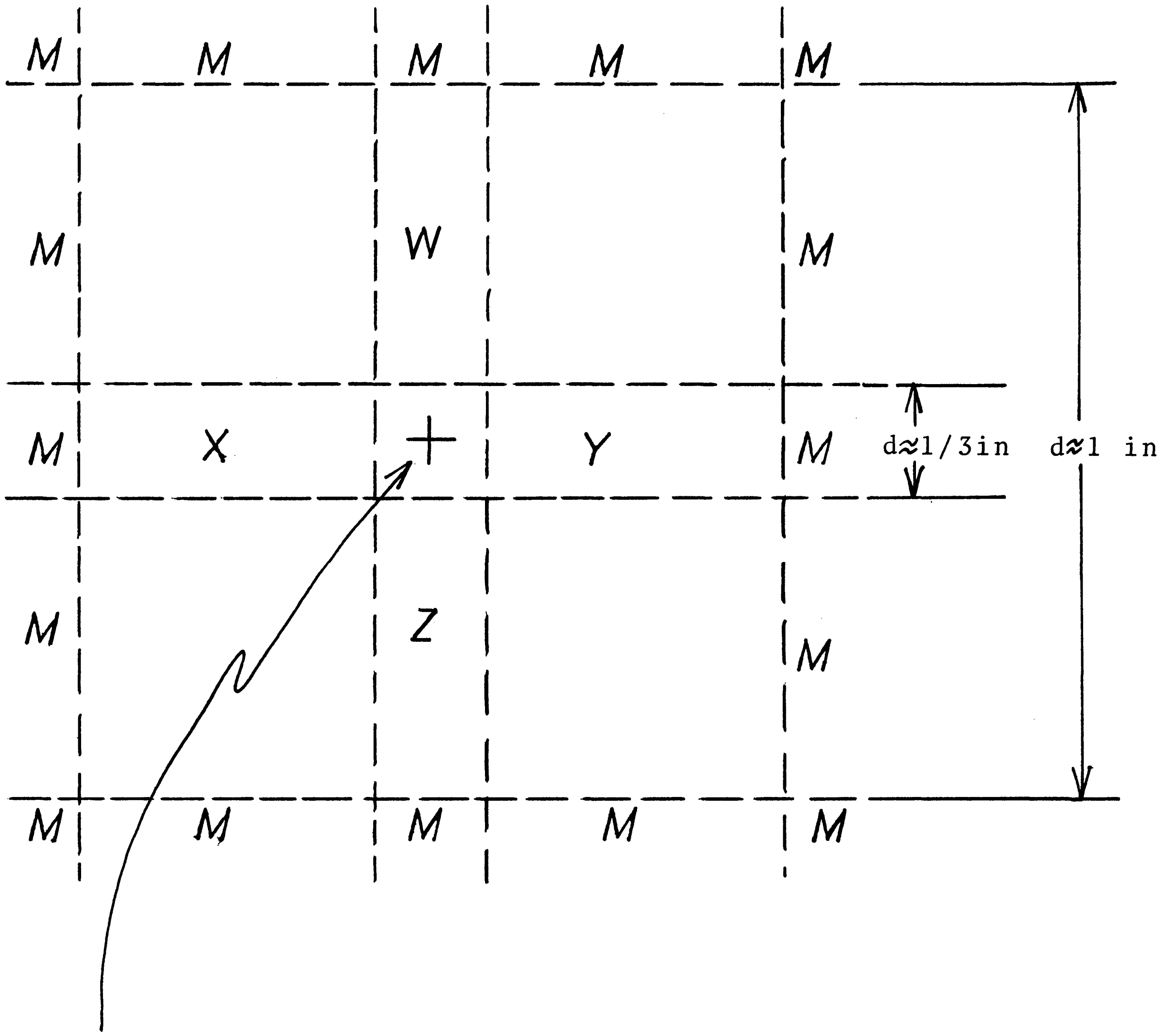
Figure 10

Generation of Branches

forth five times between A and B regions without entering an M region or a W, an X, a Y, and a Z region, the connection is deleted. This action for deleting simple connections is the same as that for deleting symbols, and it applies to branches and merges as well.

A merge, like a branch, is also formed by adding a connection line to a simple connection. However, the connection line to be added is drawn from an output port to the simple connection, rather than from the simple connection to an input port. The type of merge (i.e., random or priority) is determined by recognizing light pen motion as the connection line is terminated. The thresholds involved in this operation are shown in Figure 11. If the light pen enters the W, X, Y, and Z regions without entering an M region, tracking of the light pen is terminated by SELMA, and a priority merge is formed. If the user removes the light pen near the simple connection without entering an M region, a random merge is formed. These operations are shown in Figure 12. If the light pen is moved into an M region, the threshold pattern is removed, and drawing of the connection line continues as through no simple connection had been encountered.

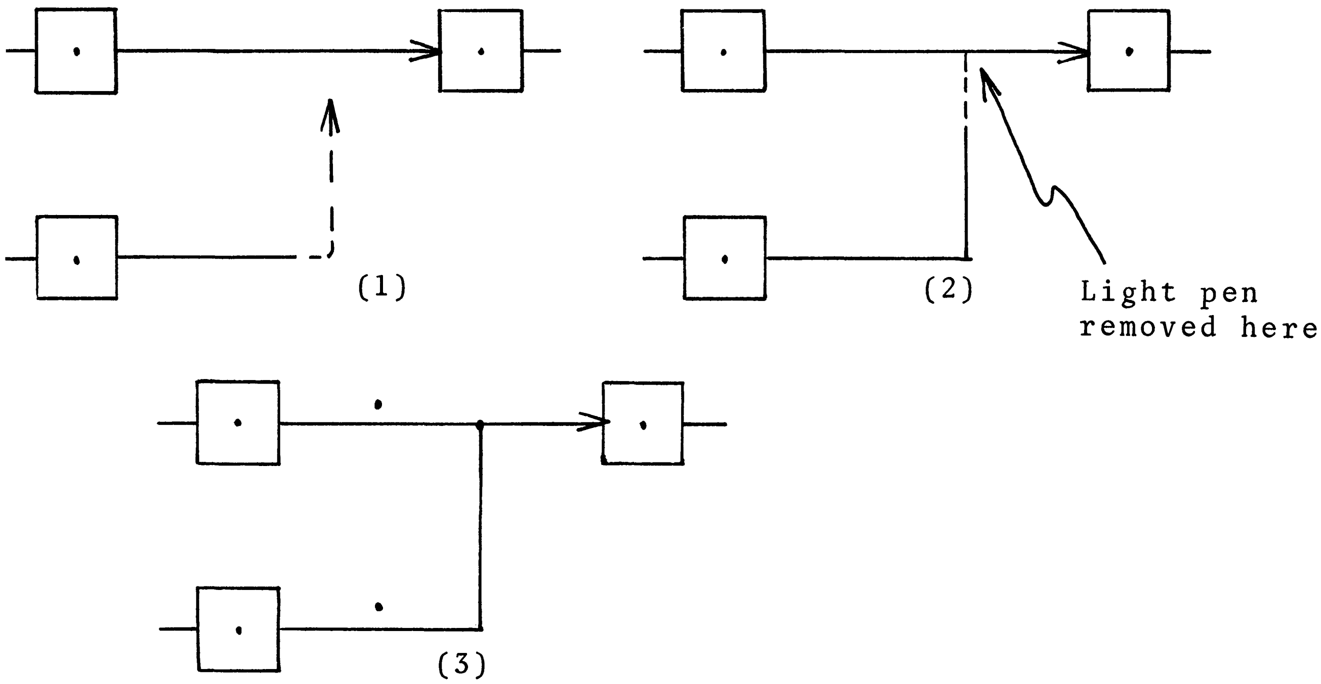
Once a branch or a merge is formed, more element ports may be associated with it by drawing additional connection lines. A connection line to be added to a branch is drawn from the branch to an input port, whereas a connection line to be added to a merge is drawn from an output port to the merge. No threshold pattern is required to determine whether a dot or diamond is to be placed at the intersection of the new



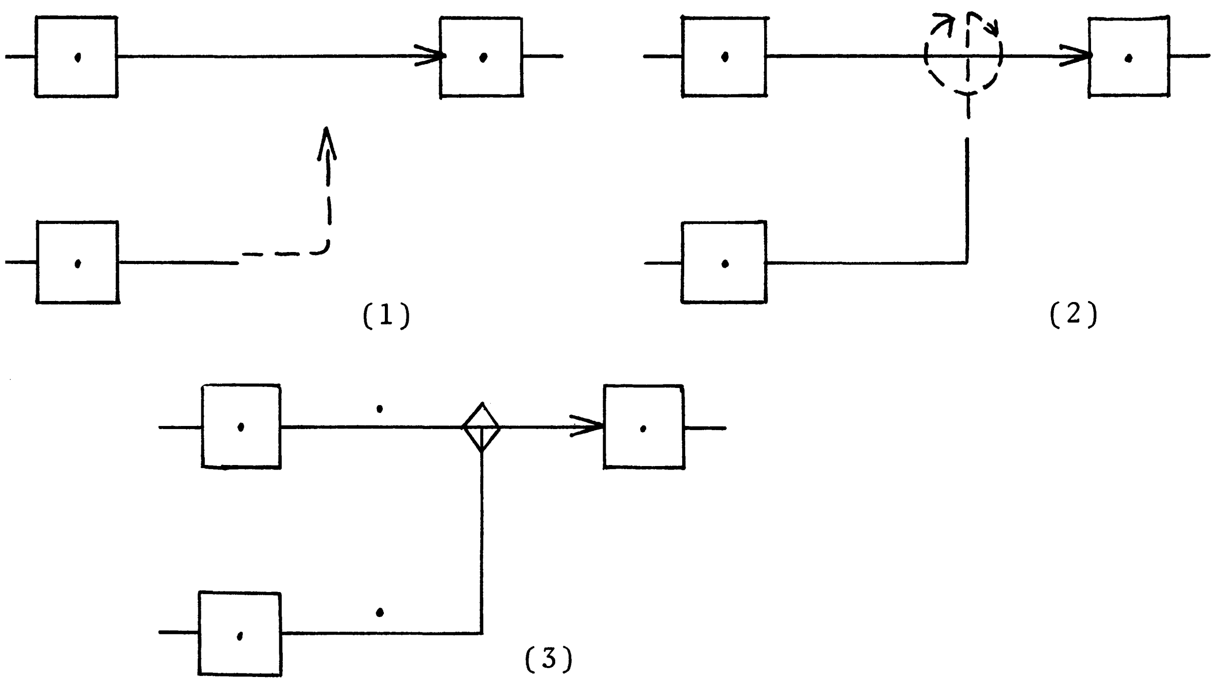
Coordinates at which light pen was detected on simple connection

Figure 11

Threshold Patterns for Generating Merges



a. Random Merge



b. Priority Merge

Figure 12

Generation of Merges

connection line with the branch or merge, since the type of branch or merge is already established.

The values of the probabilities and priorities which are associated with branches and merges, as well as the values of the parameters which are associated with certain symbols, are assigned through the use of push buttons. Push buttons 0 through 9 represent the decimal digits 0 through 9, respectively. Push button 10 represents a decimal point, and push button 11 is used to delete erroneous parameter values. A parameter value is specified by supplying a sequence of from one to four characters. The value is displayed in the lower right-hand corner of the screen as it is inputted, and the light pen interpreter is modified so that all parameter dots and parameter values, but no other parts of the diagram, are sensitive to the light pen. If the light pen is now aimed at a parameter dot or parameter value, the parameter dot or parameter value is replaced with the new parameter value which was obtained from the push buttons, and the light pen interpreter is restored to its original state.

Two modes of parameter values may be assigned: integer and real. The function of each parameter value in the diagram determines its mode, e.g., a queue length must be an integer, a probability must be a real parameter, etc. SELMA refuses to accept a parameter value of the wrong mode.

Occasionally, a model will be so complex that its diagram will not fit onto the display screen. For this reason, the diagram is considered to be drawn on a 75"× 75" piece of "paper" which may be moved to various positions under the screen. The TRANSLATE light button is provided in the lower left-hand corner of the screen in the construction phase to provide this function. When this light button is referenced with the light pen, the light pen interpreter is put into a translate mode, and the TRANSLATE light button is modified to indicate this fact. A subsequent reference to this light button will restore the light pen interpreter and the TRANSLATE light button to their original states. (The creation of a new element will also produce this effect.) When the light pen interpreter is in the translate mode, a subsequent reference to a point on the diagram causes tracking to be started and the corresponding point on the "paper" to be affixed to the tracking cross. The "paper" may then be moved by moving the light pen.

When a diagram is thought to be complete, SELMA may be put into the results phase (Figure 2b) via a reference to the RESULTS light button. When SELMA is in this phase, the user may request a steady-state probability density function either for any single element or for the entire model. The light pen interpreter is modified when this phase is entered so that the only function which can be performed by referencing the diagram without further modifying the light pen interpreter (via the TRANSLATE light button or by assignment of parameter values) is the identification of elements.

When a density function is to be requested for a particular element, that element may be identified by referencing it with the light pen. A box of dotted lines is placed around the element and a one-character alphabetic name is assigned to the element as shown in Figure 13. If a different element is subsequently selected before the result request is completed by referencing either the PLOT or TYPE light button (described below), the box is moved to that element, and the name associated with the box is changed to the name of the new element. A reference to the box with the light pen will remove it from the diagram. When no box appears around any element, the entire model, rather than any particular element, is identified.

The probability density function for the item (i.e., either the entire model or a particular element) identified in this manner may be obtained in either of two forms: (1) as a histogram on the display screen, or (2) as a table on the teletype. A histogram is obtained through a reference to the PLOT light button, and a typed table is obtained through a reference to the TYPE light button.

When a result is requested, it will not be returned if the diagram is not complete or if contradictory parameter values exist. The diagram is not complete under either of the following conditions:

(1) At least one port in the diagram is not connected, or

(2) At least one parameter dot remains on the diagram.

Contradictory parameter values exist if either of the following conditions is true:

(1) Two equal priorities are assigned to one priority branch or merge, or

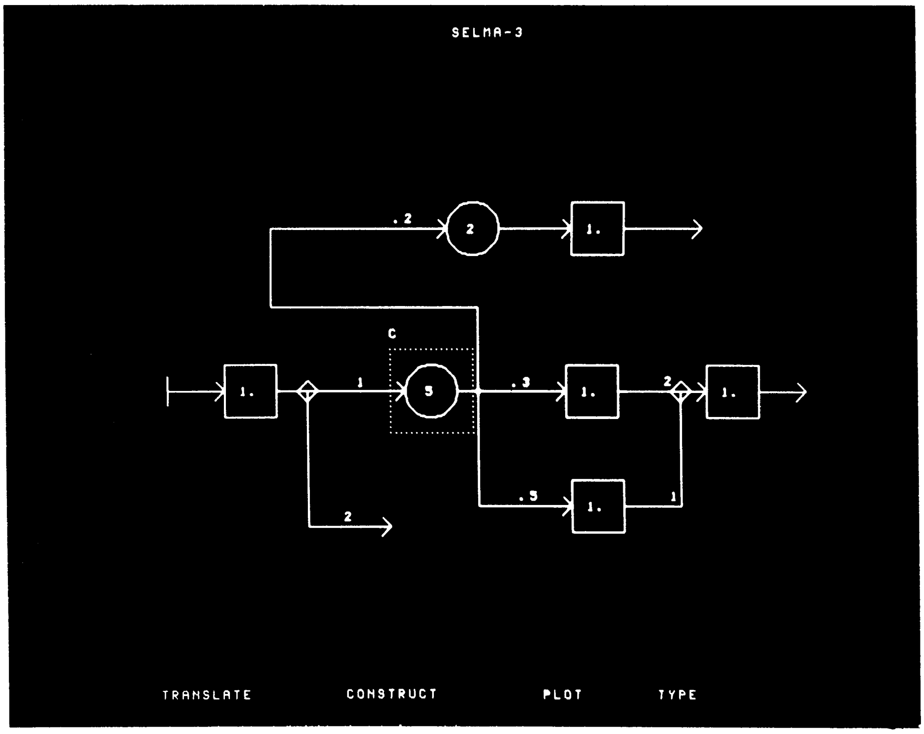


Figure 13

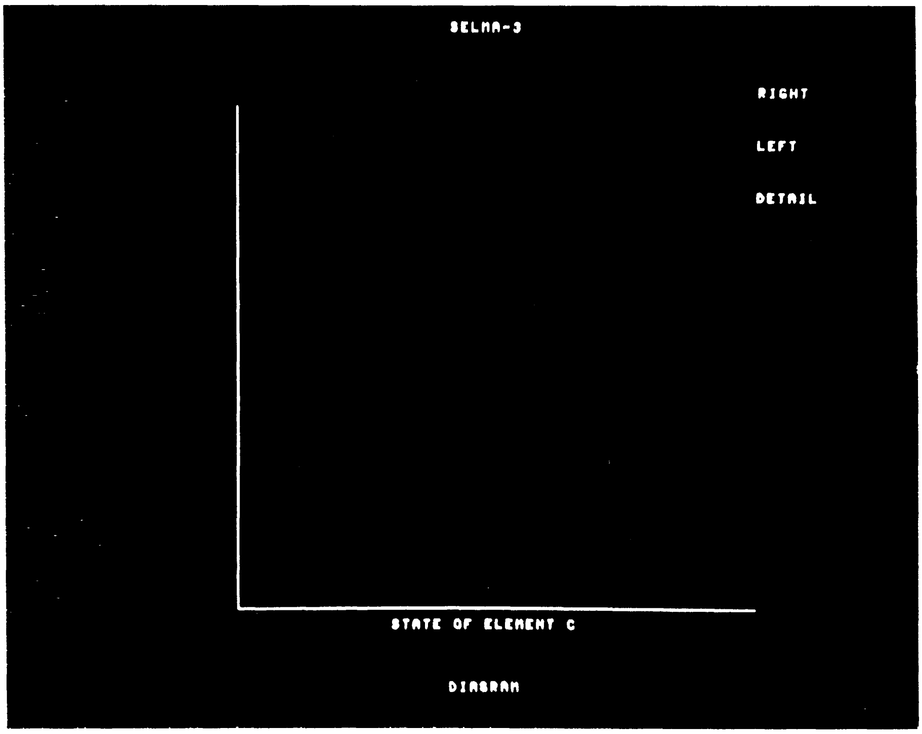
Identification of an Element

(2) The probabilities assigned to a random branch or merge do not sum to 1.0.

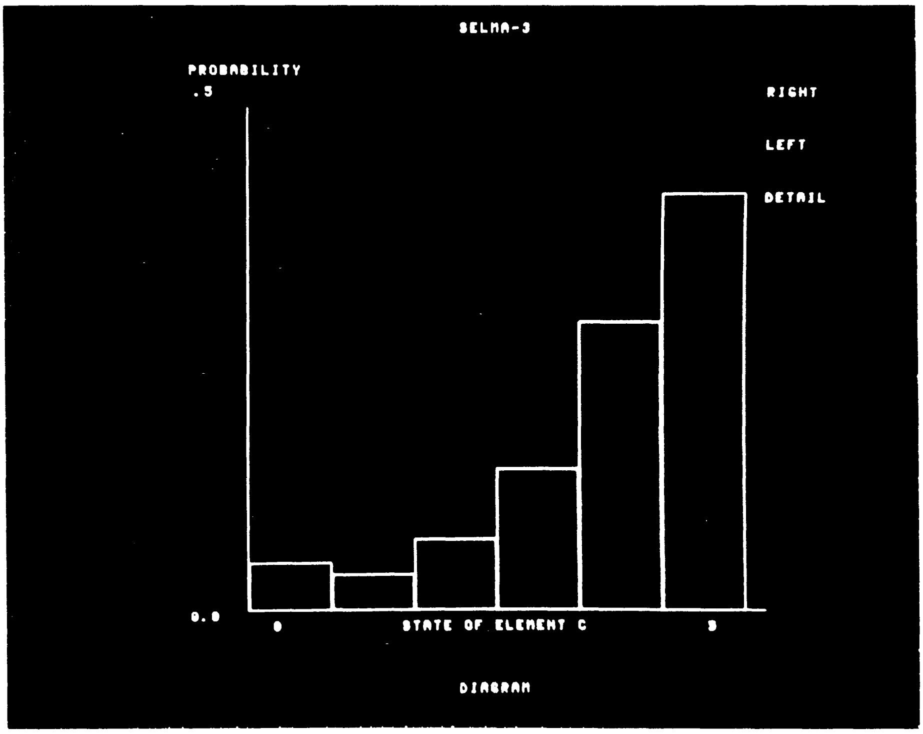
If any of these four conditions is true, SELMA is forced into the construction phase, a large flashing arrow is placed on the screen to indicate the location of the error, and an appropriate comment is typed on the teletype. The flashing arrow disappears from the screen when the error is corrected. However, if the error involves contradictory parameter values, the arrow may not point to the value which the user decides to change to correct the problem. In this event, the parameter which is being indicated should be reassigned the same value to delete the arrow and thus avoid confusion on subsequent errors.

If the result which was requested was a histogram and none of the above errors is present in the diagram, SELMA enters the plot phase (Figure 2c). The diagram is removed from the screen, and axes for the graph are displayed, together with a label for the abscissa. For the example in Figure 13, this result is shown in Figure 14b.

Only the maximum and minimum values of the abscissa and ordinate which could be represented without scaling the graph are represented on the final plot. However, the user may request numerical values from QAS for any particular bar in the histogram by pointing to that bar with the light pen. (Subsequent references to other bars will erase previous values so that only the last one is displayed.) The result of pointing to the fourth bar in Figure 14b is shown in Figure 15a.



a. Before Graph Is Returned from QAS

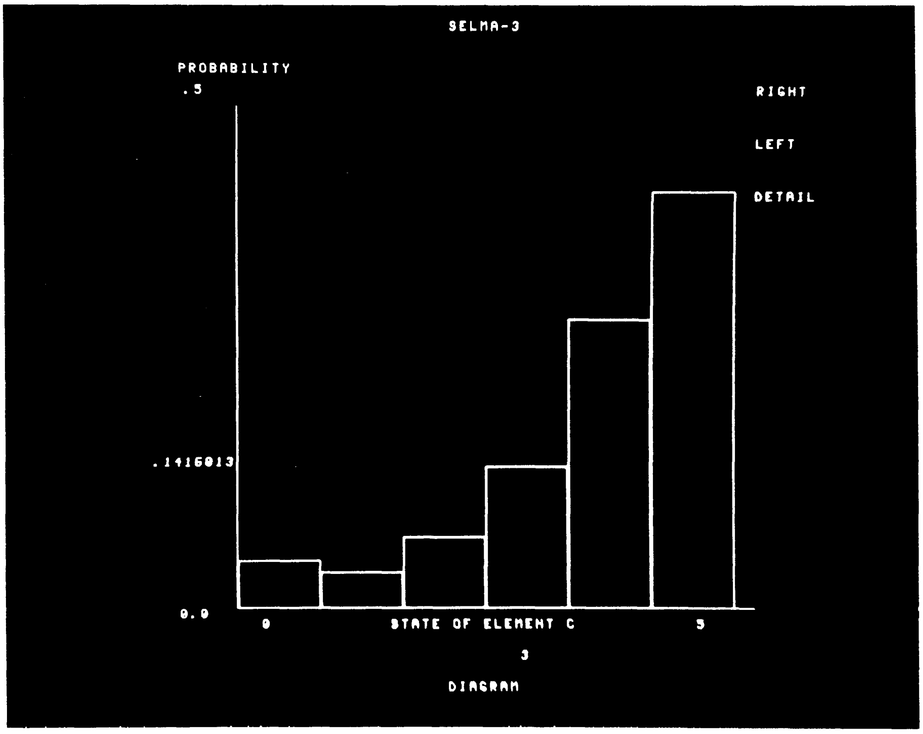


b. After Graph Is Returned from QAS

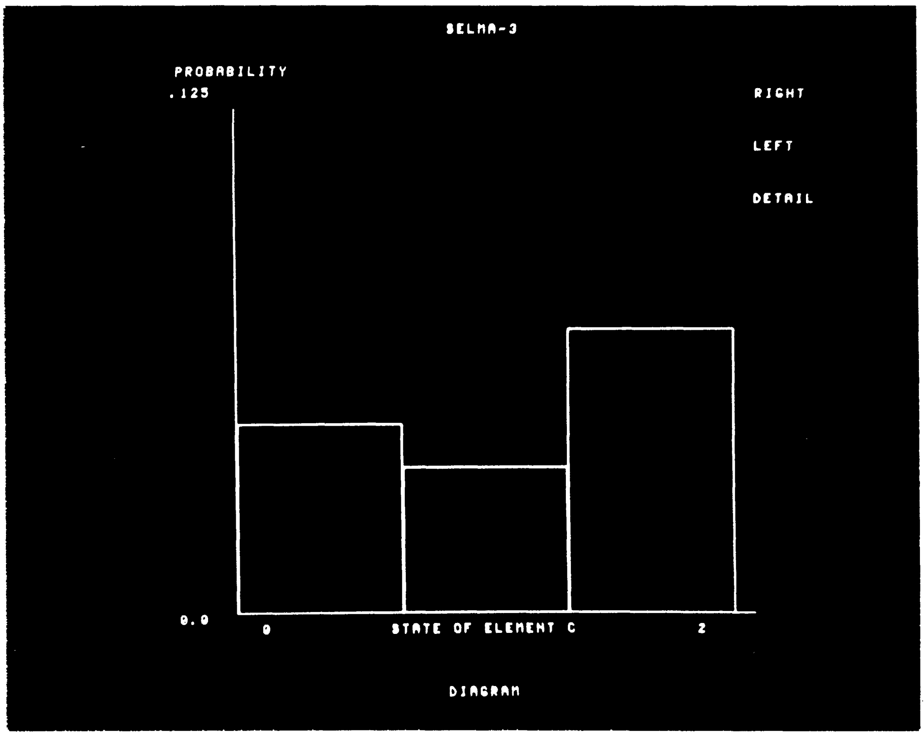
Figure 14

Result Plot For Model in Figure 13

In addition to facility for obtaining numerical values for points on the graph, facility is also provided for looking at various sections of the graph. This facility is necessary because a maximum of 50 abscissa values may appear in one graph (limited by local storage and resolution of the display) and some results require more than 50 abscissa values to plot in their entirety. It is also a convenience, since detailed examination of a graph with a wide range of ordinate values or a large number of abscissa values is more difficult without it. Various sections of the graph may be displayed through the use of the RIGHT, LEFT, and DETAIL light buttons. If the RIGHT light button is referenced, and then the bar at a particular abscissa is referenced, the plot is modified so that that abscissa, together with all abscissas of greater value (up to a total of 50 abscissa values) is plotted. In similar manner, a plot involving an abscissa, together with all abscissas of smaller value (up to a total of 50 values) may be obtained through the use of the LEFT light button. For example, the graph obtained by applying the LEFT light button to the abscissa 2 in Figure 15a is shown in Figure 15b. The DETAIL light allows the user to specify the maximum and minimum abscissas he wishes to plot. After the DETAIL light button is referenced, these two abscissas are specified by referencing the two bars of the graph which correspond to them. For all of these operations, the fact that a particular light button has been selected is indicated by the removal of all other light buttons from the screen until the new graph is completely specified.



a. Display of Numerical Values



b. Expansion of a Section of the Graph

Figure 15

Modifications of a Result Plot

As indicated by the above discussion, all of the very common operations which are involved in the specification of a queueing model, with the exception of parameter value input, are performed with the light pen only. This type of operation was chosen because the user can usually generate faster input with the light pen than with any other input device. (Input of parameter values with the light pen was found to be awkward, so push buttons were used for this purpose.) However, there are some operations which are performed very infrequently, and, should they be performed accidentally, compensating for their effect would be difficult. In particular, destroying the current model in order to begin a new model, and terminating SELMA are such operations. These operations are performed through the use of the keyboard, which is a relatively inaccessable device to the user. In this way, accidental performance of these operations is less likely than it would be if these operations were invoked with the light pen.

All operations performed from the keyboard are initiated by commands which are invoked by a single character. The keyboard interpreter responds to each command by typing a word or phrase which is a more complete description of the command. A command may be deleted before it is effective by typing a rubout in place of any subsequent character. The command which destroys the current model in preparation for beginning a new one is the following:

CLEAR? OK

(The underlined characters are those typed by the user.) Although the character "C" would be sufficient for specifying this operation, a confirmation (accomplished by typing "O") is required to help avoid accidental performance of this operation. The fact that the confirmation is required is indicated by the question mark (?). If the user wishes not to confirm the operation, he may type "N" for "NO". Similarly, the following command, which terminates SELMA (and QAS), requires confirmation:

ESCAPE? OK

Two other keyboard commands which represent infrequent operations which cannot easily be reversed once they are performed are saving the current model on a file at the IBM 360/67 and retrieving a queueing model from a file. These two commands are the following:

SAVE ON FILE _ _ _ _

GET FILE _ _ _ _

Each command is completed by typing a file name, or a file name followed by a space, in turn followed by the word "ALL". (The command is terminated by a carriage-return typed on the keyboard.) In the former case, any results which have been computed are not saved, but in the latter case they are saved. The user is prevented from performing any other operations while a model is being saved or retrieved. Retrieval of a model effects a "CLEAR" command before the model is actually retrieved.

Although saving or retrieving a model is costly to perform accidentally, no provision for explicit confirmation is provided. The file name which is required to complete each command serves as a confirmation. If an illegal file name is specified, a comment to this effect will be typed, and QAS will ignore the command.

Some users have found the keyboard, rather than the push buttons, to be a more suitable device for inputting parameter values. For this reason, a command to substitute the teletype for the push buttons and a command to specify a parameter value from the keyboard are provided. The device from which parameter values are to be obtained is specified by the command

VALUES FROM { PUSH BUTTONS | TELETYPE }

(The default device is the push buttons.) While the teletype is the input device for parameter values, the following command may be given to specify a parameter value:

PARAMETER: _ _ _ _

This command is completed by typing a number which consists of from one to eight characters, followed by a carriage return. All of the characters are sent to QAS, but only the first four are displayed if more than four are specified. Real values are restricted to be less than or equal to 999.9999 to avoid misrepresentation of the decimal point on the display.

Two other keyboard commands are provided for purposes of debugging SELMA and/or QAS. However, these commands are not normally available to the user, since they deal with dataphone commands between SELMA and QAS. They are described in Section 5, following a detailed description of the command exchanger in Section 3 and a description of the data structure used to represent the topology of the network in Section 4.

3. The Command Exchanger

In order to be effective, the frequently used operations described in the previous section must provide rapid response to user inputs. If all programmed operations were performed in sequence, the low data rate of the dataphone would pose a threat to this response time in that the transmission of commands to inform QAS of certain inputs might delay responses to subsequent inputs. For this reason, the response task for each user input which requires communication with QAS merely places the commands to be sent to QAS into a buffer. A separate task, the command exchanger, then reads commands from this buffer and sends them to QAS. Since the command exchanger and response tasks for user inputs are executed asynchronously, the rate of dataphone transmission does not seriously affect the rate of response to user inputs.

Since the dataphone under consideration is a half-duplex dataphone, SELMA and QAS must not transmit simultaneously. For this reason, SELMA and QAS alternately transmit records, with QAS sending the first record. Since QAS sends the first record, QAS need not be executing before SELMA is started. Since a record directed to the SELMA command exchanger is not acknowledged by the SEL executive system until the command exchanger has removed it from the dataphone input buffer, SELMA need not be executing before execution of QAS begins.

Each dataphone record which is sent from SELMA to QAS or from QAS to SELMA contains exactly one command. The general format of a command is shown in Figure 16. The command is delimited by bytes whose value is FF_{16}, and all other bytes in the command have values less than FF_{16}. (Although the beginning and end of the record are sufficient to delimit the command, the two delimiting bytes were included in the format to facilitate a possible future program modification to permit transmission of multiple commands per record.) The first byte after the initial delimiter specifies the "phase" of the command, and the byte which follows it identifies the command within that phase. (The command phase is not related to the phase of SELMA.) The combination of the phase and command bytes identifies the command, and, consequently, specifies the format of the data bytes.

The formats of the commands which are accepted by QAS are indicated by Figure 17, and the formats of the commands which are accepted by SELMA are indicated by Figure 18. The abbreviations for various groups of data bytes are interpreted as follows:

CN Connection name. A number between 129 and 254 which identifies a particular connection. (1 byte)

CPN Connection port number. A number between 1 and 254 which specifies a particular port of the connection specified by an associated connection name. (1 byte)

CT Connection type. A number between 129 and 254 which specifies the type of connection (e.g., simple

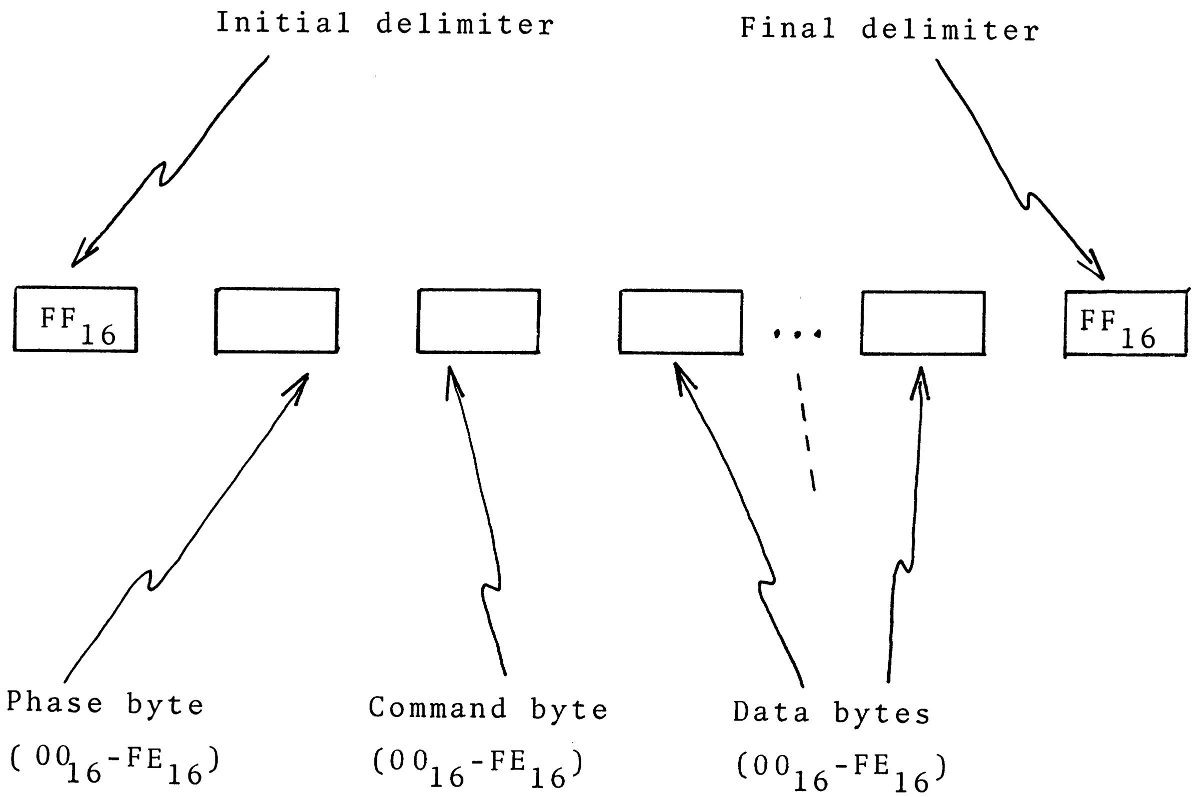


Figure 16

Command Format

Phase Byte	Command Byte	Data Bytes	Function
00	00	-	Null
00	01	-	Patch-up[1]
00	02	-	Call system
00	03	-	Call error[1]
00	04	-	Call MTS[1]
00	05	-	Initialize
00	06	-	Wipe out QAS output buffer[1]
01	00	EN or CN, ET or CT, GPV	Create element
01	01	EN or CN	Destroy element or connection
01	02	EN or CN, PN, PV	Assign parameter value
01	03	CN, CPN, EN, EPN	Connect
01	04	EN, EPN	Disconnect
01	05	EN or CN, GPN, GPV	Alter generation parameter value
02	00	IL, PV[2]	Compile and solve[1]
03	00	EN[3]	Plot results
03	01	SN, SC	Modify plot
03	02	SN	Get value for graph
03	03	EN[3]	Type results
04	00	FN	Begin saving model
04	01	-	Terminate saving model
04	02	FN	Retrieve model from file
04	03	SELMA phase 02 command	Save command to be returned

Figure 17
Commands Accepted by QAS

[1] This command is not currently generated by SELMA.

[2] Parameter value specifies convergence factor (see QAS report).

[3] An element name 00 specifies the entire model.

Phase Byte	Command Byte	Data Bytes	Function
00	00	-	Null
00	01	(See text)	Patch-up
00	02	-	End of file
01	00	XC, T, T, T, T, T[1]	Set up graph
01	01	YC, YC, . . . , YC	Plot values
01	02	T, T[4]	Display single value
02	00	YC, XC	Create fragment
02	01	YC, XC, EN, 00, LET	Create element
02	01	YC, XC, CN, 00, WC	Create connection
02	02	DW, DW, . . . , DW	Load connection segment
02	03	-	Insert connection leaf
02	04	YC, XC, PV[3]	Assign parameter
02	05	EN, EPN	Connect
03	00	-	Send model to be saved

Figure 18

Commands Accepted by SELMA

---

[1]Distance between abscissas, minimum x label, maximum x label, minimum y label, maximum y label, y axis label.

[2]Ordinate label.

[3]Missing parameter value indicates unassigned parameter.

[4]Abscissa value, ordinate value.

connection, random branch or merge, priority branch or merge). (1 byte)

DW Display file word. Two bytes, each of which has a value from 0 through 127. These two bytes are decoded to form an 18-bit word to be loaded into core by SELMA according to the following scheme:

(1) The low-order 7 bits of each byte are concatenated to form a 14-bit number.

(2) The two high-order bits of the 14-bit number are placed into positions 0 and 1 of the 18-bit word, and the remaining 12 bits are placed into positions 6-17 of the 18-bit word. Positions 2-5 of the 18-bit word are set to zero.

EN Element name. A number between 1 and 127 which identifies a particular element. (1 byte)

EPN Element port number. A number between 1 and 254 which specifies a particular port of the element specified by an associated element name. (1 byte)

ET Element type. A number between 1 and 127 which specifies the type of element (e.g., queue, server, or source or exit). (1 byte)

FN File name. A sequence of bytes whose values are 6-bit character codes [3 p. 17] which represent a file name. (1 to 16 bytes)

GPN Generation parameter number. A number between 1 and 254 which identifies a particular generation parameter (i.e., parameter which modifies an element or connection type). (1 byte)

GPV Generation parameter value. A number between 1 and 254 which represents the value of a generation parameter. (1 byte)

IL Iteration limit. Two bytes, each of which has a value between 0 and 127. A number between 1 and 16,383 which represents the maximum number of iterations which will be performed whenever a model is solved is obtained by concatenating the low-order 7 bits of the two bytes.

LET Local element type. A number between 1 and 254 which identifies a graphical symbol for an element type. This number is not necessarily the same as the corresponding element type, for several graphical symbols may be associated with one element type (e.g., source and exit represent the same element type). (1 byte)

PN Parameter number. A number between 1 and 254 which identifies a parameter for an element or connection. (1 byte)

PV Parameter value. A sequence of bytes whose values are 6-bit character codes which represent a non-negative real or integer number. (1 to 8 bytes)

SC State count. A number between 1 and 254 which represents the number of points to be plotted on a graph by QAS. Fifty points are plotted if this number is either zero or greater than 50. (1 byte)

SN State number. A number between 0 and 16,383 which represents an abscissa on a graph plotted by QAS. This number is coded in the same way that an iteration limit (IL) is coded. (2 bytes)

T Text item. A sequence of bytes, the first of which has a value which is the number of bytes which follow it, and the remainder of which are 6-bit character codes. (2-16 bytes)

WC Word count. A number between 1 and 16,383 which represents the size of storage block required to load a connection leaf when retrieving a model from a file. This number is coded in the same way that an iteration limit (IL) is coded. (2 bytes)

XC An abscissa between -8192 and 8191. This coordinate is coded in the same way that an iteration limit (IL) is coded. However, the 14-bit number represented is interpreted as a two's complement number, rather than as an unsigned positive number. (2 bytes)

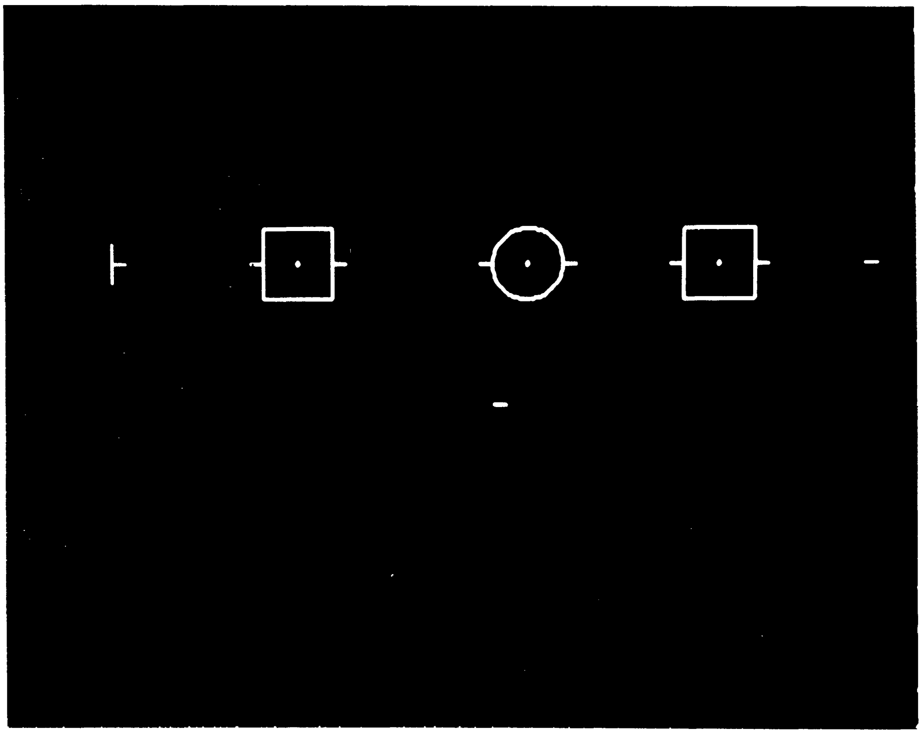
YC An ordinate between -8192 and 8191. This number is coded in the same way an abscissa is coded. (2 bytes)

Several of the commands described in Figure 17 and Figure 18 affect the exchange of other commands. The null command (which may be sent in either direction) is provided so that one party can allow the other to transmit without directing it to perform any other function. However, if each party were to send null commands whenever it had nothing else to send, the maximum dataphone traffic would occur when both SELMA and QAS were idle. Since all external inputs are directed to SELMA, rather than to both SELMA and QAS, this redundant dataphone traffic is eliminated by having SELMA not send a null command unless it expects a non-null reply from QAS. In this way, no communication takes place when both parties are idle, and SELMA is the next party to transmit whenever both parties are idle.

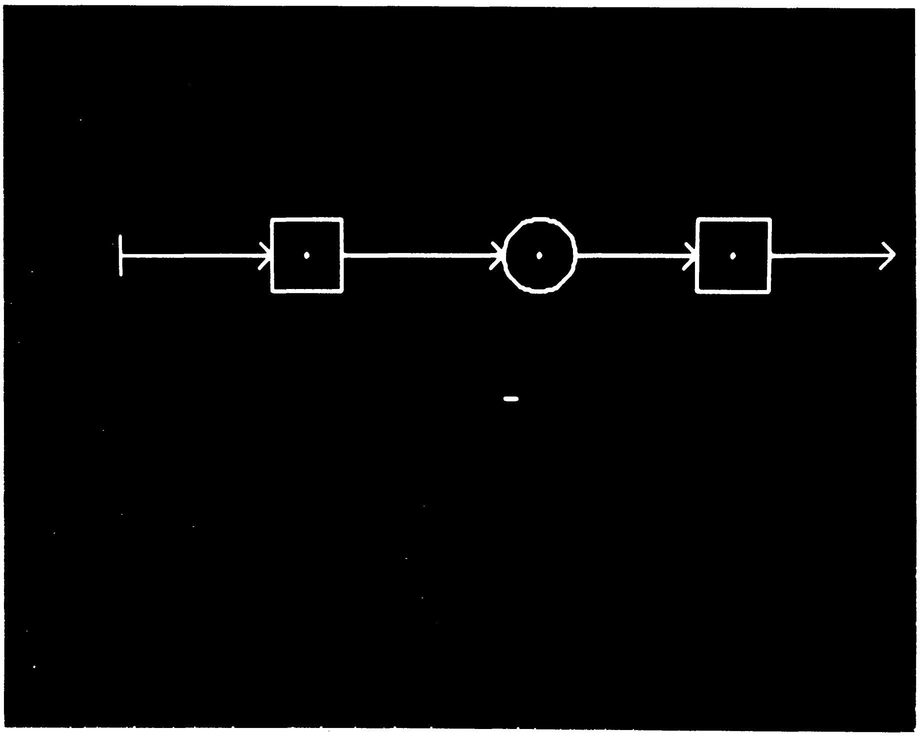
The patch-up command which is generated by QAS is transmitted to SELMA in response to receiving a command from SELMA which cannot be performed. The data bytes for the patch-up command form a command (except for delimiter bytes) to be returned to QAS to replace the offensive command. Some of the data bytes for the replacement command are set to zero or are not present to indicate to SELMA that these are to be filled in before the replacement command is sent. SELMA may then either complete the replacement command and send it to QAS, or it may send an entirely different command, as though it had never sent the illegal command.

Currently, SELMA responds to all patch-up commands by informing the user of the error which has occurred, and not responding to QAS. The user is then free to correct the error in any way he desires. In order to prevent one such error from being diagnosed as a series of errors, the user is prevented from proceeding whenever his activity generates a command to be sent to QAS which could result in a patch-up. (Such a command is generated whenever a model is saved or retrieved or whenever a result is requested.) This is accomplished by setting an "end-of-file switch" in SELMA and disabling all inputs until it is reset. This switch is then reset by either an end-of-file command from QAS (indicating the end of a successful series of transmissions) or by a patch-up command (indicating that the operation could not be performed).

As an example of a typical exchange of commands between SELMA and QAS, consider the operations depicted by Figure 19. Before any of these operations are performed, SELMA and QAS are started. Since QAS must send the first command, but it has no operation to perform (since SELMA has not informed it of any input), QAS initially sends a null command to SELMA. SELMA then initializes QAS, and



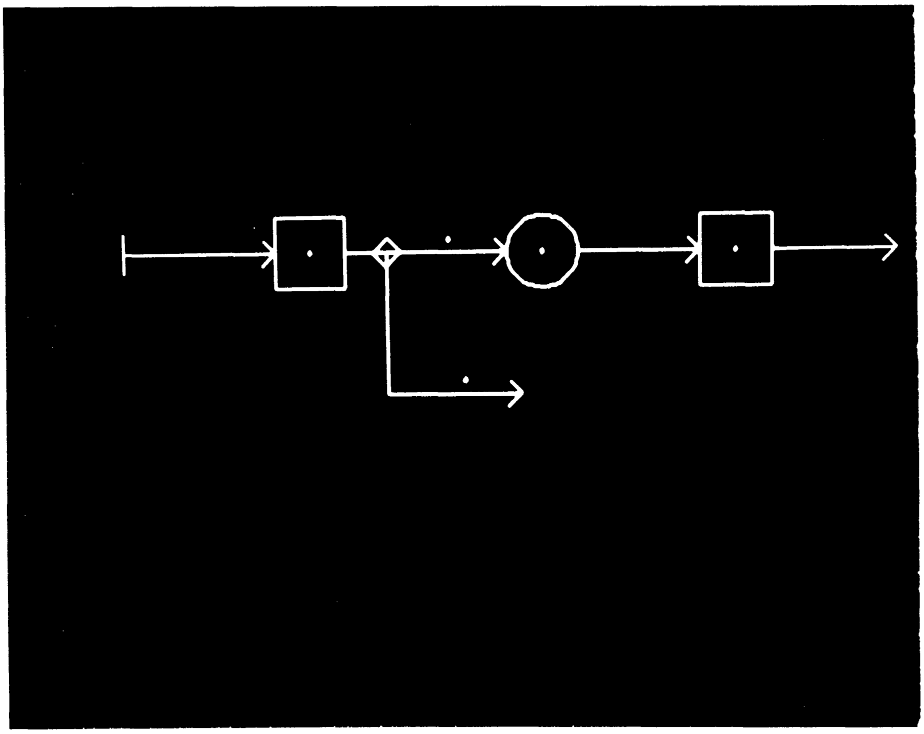
a. Creation of Elements



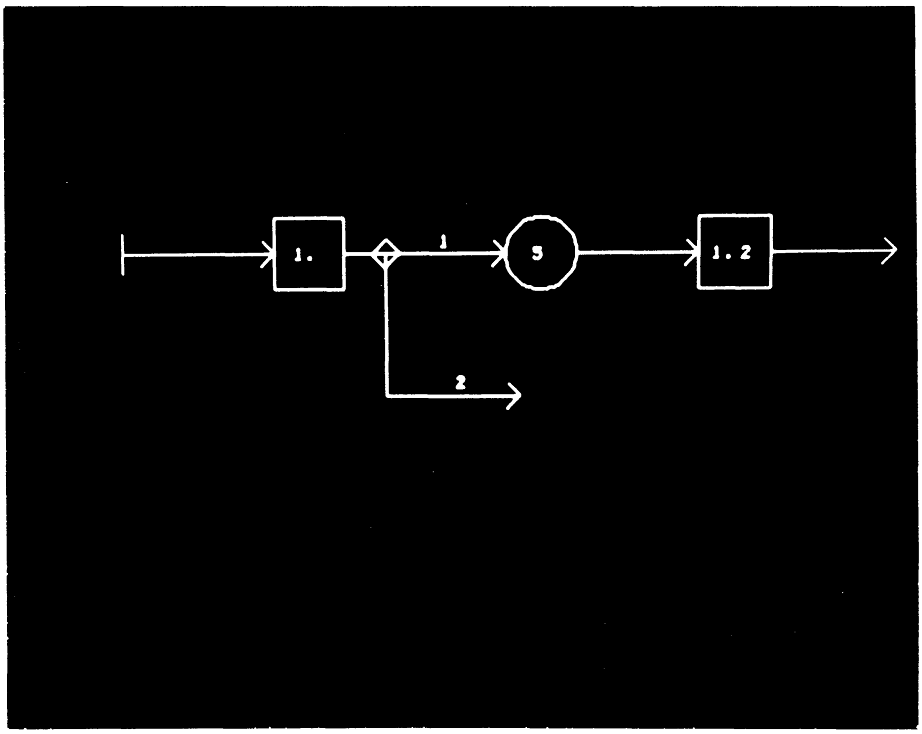
b. Addition of Simple Connections

Figure 19

Typical Construction of a Model



c. Generation of Priority Branch



d. Assignment of Parameters

Figure 19 (Cont.)

QAS replies with a second null command. This exchange of commands may be depicted (in hexadecimal notation) as follows:

QAS:	0000	(null)
SELMA:	0005	(initialize QAS)
QAS:	0000	(null)

(The delimiter bytes are not shown.) Then, when the elements shown in Figure 19a are created, the following commands are exchanged:

SELMA:	01000103	(create source)
QAS:	0000	
SELMA:	01000202	(create server)
QAS:	0000	
SELMA:	01000301	(create queue)
QAS:	0000	
SELMA:	01000402	(create server)
QAS:	0000	
SELMA:	01000503	(create exit)
QAS:	0000	
SELMA:	01000603	(create exit)
QAS:	0000	

Each of the commands transmitted by SELMA completely specifies the creation of an element. However, three commands are required to completely specify the creation of each of the simple connections in Figure 19b: one to create the connection, and one to connect each of its ports to element ports. The commands involved when the simple connections in Figure 19b are created are the following:

SELMA:	01008181	(create connection)
QAS:	0000	
SELMA:	010381010101	(connect)
QAS:	0000	
SELMA:	010381020201	(connect)
QAS:	0000	
SELMA:	01008281	(create connection)
QAS:	0000	
SELMA:	010382010202	(connect)
QAS:	0000	
SELMA:	010382020301	(connect)
QAS:	0000	
SELMA:	01008381	(create connection)
QAS:	0000	
SELMA:	010383010302	(connect)
QAS:	0000	
SELMA:	010383020401	(connect)
QAS:	0000	
SELMA:	01008481	(create connection)
QAS:	0000	
SELMA:	010384010402	(connect)
QAS:	0000	

SELMA:	010384020501	(connect)
QAS:	0000	

For purposes of illustration, assume that the user now indicates that he wants a plot of the probability distribution for the queue in Figure 19b. Clearly, the diagram in Figure 19b is incomplete, so a patch-up command will be returned by QAS. Since QAS checks the structure of the model before it checks its parameters, the patch-up command will refer to the unconnected exit. The following exchange of commands results from the user's attempt to obtain this solution:

SELMA:	030003	(plot results)
QAS:	0001010300000601	(patch-up)

The user, upon being requested via a flashing arrow on the screen and a teletype comment to connect the unconnected port, then draws the priority branch shown in Figure 19c. Since the former simple connection must be replaced with a new branch connection, the following commands are exchanged:

SELMA:	01040202	(disconnect)
QAS:	0000	
SELMA:	01040301	(disconnect)
QAS:	0000	
SELMA:	010182	(destroy connection)
QAS:	0000	
SELMA:	0100828202	(create priority branch)
QAS:	0000	

SELMA:	010382010202	(connect)
QAS:	0000	
SELMA:	010382020301	(connect)
QAS:	0000	
SELMA:	010382030601	(connect)
QAS:	0000	

Now that the request for correction has been satisfied, the user again requests a probability distribution for the queue. However, the diagram is still not complete, for not all (in fact, none) of the parameter values have been assigned. QAS complains about the parameter in the server which receives inputs from the source. The following commands are exchanged:

SELMA:	030003	(plot results)
QAS:	000101020201	(patch-up)

In response to this second complaint, the user assigns parameter values as shown in Figure 19d. The commands involved are the following:

SELMA:	010202010131	(assign 1.)
QAS:	0000	
SELMA:	0102820101	(assign 1)
QAS:	0000	
SELMA:	0102820202	(assign 2)
QAS:	0000	
SELMA:	0102030105	(assign 5)

QAS:	0000	
SELMA:	01020401013102	(assign 1.2)
QAS:	0000	

The diagram is now complete, and the user once again requests a probability distribution for the queue. This time, a plot is returned. The commands which are involved are the following:

SELMA:	030003	(plot results)
QAS:	0100100001000105030031000231 050B191B180B0A0B1215121D22	(set up graph)
SELMA:	0000	(null)
QAS:	0101510D1E64195B153D1178037C	(results)
SELMA:	0000	(null)
QAS:	0002	(end of file)

Now that a solution has been obtained, the user decides to save both the model and the results of solution on the file S, and then terminate SELMA and QAS with the ESCAPE command. The following commands are exchanged as a result of these operations:

SELMA:	04001C3E0A1515	(begin saving of file S)
QAS:	0300	(send model to be saved)
SELMA:	04030200000C7F45	(save)
QAS:	0000	(null)
SELMA:	040302017F66002E060004	etc.
QAS:	0000	
SELMA:	04030201000000053040002	

QAS: 0000

SELMA: 040302010000005304000 2

QAS: 0000

SELMA: 040302047F7C7F74013102

QAS: 0000

SELMA: 04030201000000360300 01

QAS: 0000

SELMA: 040302047F7C7F7C05

QAS: 0000

SELMA: 04030201000000 6B050004

QAS: 0000

SELMA: 0403020100000018020002

QAS: 0000

SELMA: 040302047F7C7F780131

QAS: 0000

SELMA: 0403020100000000010003

QAS: 0000

SELMA: 04030201000000 1F82000021

QAS: 0000

SELMA: 04030202450222402451080010082461000814181C181C081408003824511000000010000000186800001000003440680834000018082451100000001000004840001848 2C00

QAS: 0000

SELMA: 04030203

QAS:	0000
SELMA:	02040003002301
QAS:	0000
SELMA:	040302047F1C001802
QAS:	0000
SELMA:	040302050202
QAS:	0000
SELMA:	040302050301
QAS:	0000
SELMA:	040302050601
QAS:	0000
SELMA:	0403020100000005A8400000A
QAS:	0000
SELMA:	0403020201012240245110000000100000444000 18442C00
QAS:	0000
SELMA:	04030203
QAS:	0000
SELMA:	040302050402
QAS:	0000
SELMA:	040302050501
QAS:	0000
SELMA:	0403020100000003D8300000A
QAS:	0000

SELMA: 04030202010122402451100000001000004440 0018442C00

QAS: 0000

SELMA: 04030203

QAS: 0000

SELMA: 040302050302

QAS: 0000

SELMA: 040302050401

QAS: 0000

SELMA: 0403020100000028100000A

QAS: 0000

SELMA: 04030202010122402451100000001000004440 0018442C00

QAS: 0000

SELMA: 04030203

QAS: 0000

SELMA: 040302050101

QAS: 0000

SELMA: 040302050201

QAS: 0000

SELMA: 0401 (terminate saving)

QAS: 0000

SELMA: 0002 (call system)

The command which terminates the save operation (0401 from SELMA) resets the end-of-file switch in SELMA which is set by the initial save command (0400 from SELMA). Consequently, no end-of-file command (0002 from QAS) is needed from QAS. This special condition is consistent with the design of QAS, for QAS produces no non-null output and therefore does not generate the end-of-file command.

The reader should note that some of the commands which are saved when a model is saved on a file do not contain all of the information necessary to reconstruct the model. For example, the "assign parameter" command (0204) which is saved does not specify to which element it refers or which parameter of that element is being assigned. Such information however, may be derived from the order in which commands are saved in the file. The process of obtaining this information is described in the following section, following a description of the display structure which is used to represent the topology of the network.

4. The Display Structure

In order that SELMA be able to generate the commands described in the previous section or to interpret those received from QAS, the topology of the network diagram must be known to SELMA at all stages of its construction. This topology is stored in a display structure consisting of levels and leaves maintained by the executive system [3 p. 35]. The arrangement of these levels and leaves for this application may be described in terms of sets.

A set of element symbols which are connected to each other, together with the symbols for the connections which connect them, is treated as a unit, called a fragment. A completed diagram of a queueing network consists of exactly one fragment. However, the diagram may consist of several fragments before it has been completed. The diagram, then, is considered to be a set of fragments, or

$$D = \{F_i\},$$

where D denotes the diagram and F_i denotes the i^{th} fragment in the diagram. Each fragment, in turn, consists of symbols for elements and connections, or

$$F_i = \{S_{ij}\},$$

where S_{ij} denotes the j^{th} symbol in the i^{th} fragment. The reader should note that, for topological purposes, the symbol for an element is no different than a symbol for a connection.

Each symbol is defined by a 3-tuple which consists of a leaf, a set of ports, and a set of parameters, i.e.,

$$S_{ij} = (\ell_{ij}, P_{ij}, Q_{ij}),$$

where ℓ_{ij} is the leaf which draws the symbol S_{ij} (except for its ports and parameters), P_{ij} is an ordered set of ports, and Q_{ij} is an ordered set of parameters. If S_{ij} is the symbol for an element, ℓ_{ij} represents a leaf which may be used to help define other element symbols, whereas, if S_{ij} is the symbol for a connection, ℓ_{ij} is used to help define only S_{ij}. The reason for this condition is that all elements of the same type have identical appearance, whereas connections of the same type almost always do not have the same appearance.

The set of ports for the symbol S_{ij} may be represented by the m_{ij}-tuple

$$P_{ij} = (p_{ij}^{1}, p_{ij}^{2}, \ldots, p_{ij}^{m_{ij}})$$

and the set of parameters for S_{ij} may be represented by the n_{ij}-tuple

$$Q_{ij} = (q_{ij}^{1}, q_{ij}^{2}, \ldots, q_{ij}^{n_{ij}}).$$

Since every element and connection must have at least one port, m_{ij} is a positive integer, whereas, since some elements and connections have no parameters, n_{ij} is a non-negative integer. Even if $n_{ij} = 0$, Q_{ij} is considered to exist, but with dimension zero.

The only items in the topology which may be used in more than one place in the diagram are the leaves which draw element symbols and ports. Multiple uses of leaves which draw element symbols have been described above. When a port is used as part of more than one symbol, one of the symbols represents an element, and one of the symbols represents the connection which associates that port with other element ports. Clearly, both the connection symbol and the element symbol must be in the same fragment. To summarize,

$$p_{ij}^{r} = p_{k\ell}^{s} \Rightarrow k=i,$$

and either (1) S_{ij} is an element symbol and $S_{i\ell}$ is a connection symbol, or (2) S_{ij} is a connection symbol and $S_{i\ell}$ is an element symbol.

In order to illustrate a typical display structure for a network diagram, Figure 20 shows an incomplete network diagram and Figure 21 shows the display structure for that diagram. (All rectangles in the display structure diagram represent leaves, and all other symbols represent levels. The ordering of the attributes of each level is indicated by arranging these attributes in order from left to right in the display structure diagram.) Different leaves are used for connected and unconnected ports so that SELMA may easily ignore connected ports while connections are being drawn. Except for the leaves which display connection symbols and the leaves which display parameter values, only one copy of each leaf which displays a particular graphic element exists in the structure. This leaf is then inserted into as many levels as required to represent all of the occurrences of the graphic element which it displays.

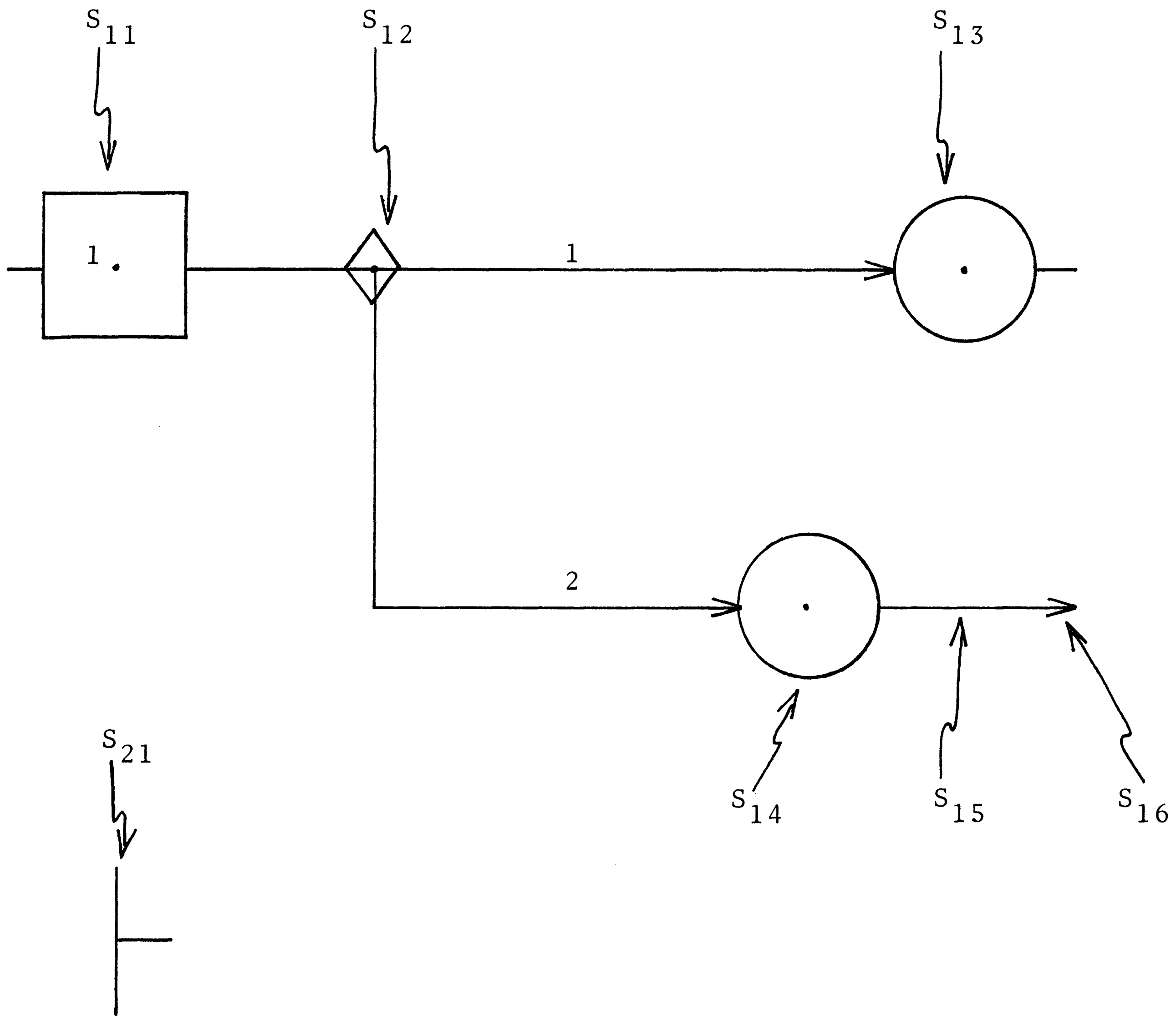


Figure 20

An Incomplete Network Diagram

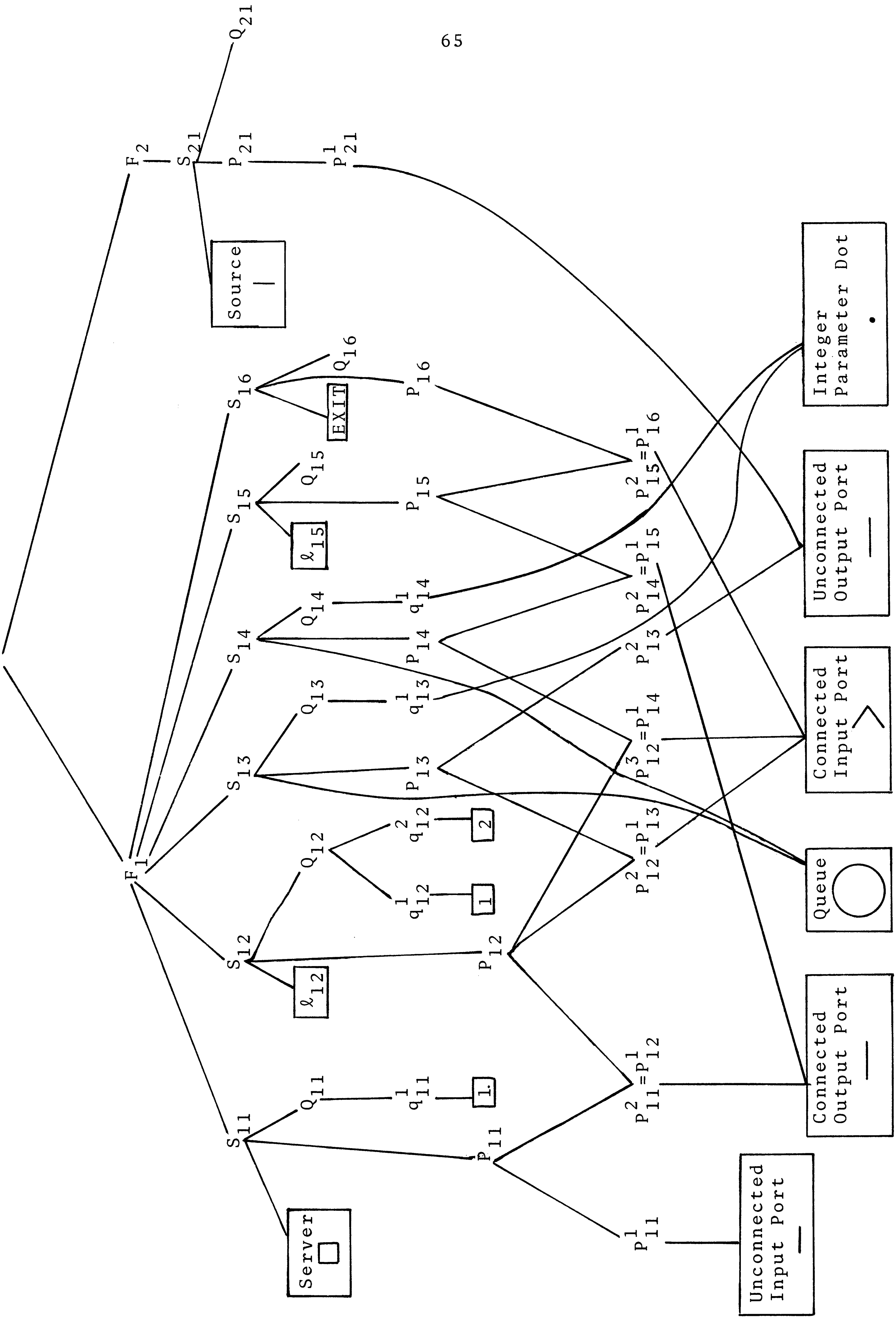


Figure 21. Display Structure for Diagram in Figure 20

As mentioned in the previous section, one of the many uses of the display structure for the network diagram is the interpretation of commands received by SELMA during the retrieval of a saved model. In general, the following information is not explicit in these commands, but is implied by the order in which they are received:

(1) The command which creates an element or connection symbol does not specify to which fragment that symbol belongs.

(2) The command which inserts a connection leaf does not specify into which symbol level that leaf should be inserted.

(3) The command which assigns a parameter value does not specify which parameter of which symbol is being assigned.

(4) The command which connects a connection port to an element port does not specify which port of which connection is being connected.

The information which is missing from these four commands is obtained from the order in which the commands are received as follows:

(1) A newly created symbol is inserted into the last fragment which was created.

(2) A connection leaf is inserted into the last symbol which was created.

(3) A parameter is assigned to the last symbol which was created. Parameters are received in their reverse order, so inserting them successively into a Q_{ij} level

will order them properly.

(4) A connection is made between a port of the last symbol created (which must represent a connection) and the element port specified by the command. Connect commands are received in the reverse order that connection ports are to assume, so inserting element ports successively into a P_{ij} level to generate connection ports will order the connection ports properly.

The display structure which represents the network diagram is part of another display structure whenever it is active (i.e., in the construction phase and results phase). The form of this display structure for the construction phase is shown in Figure 22. Each light button consists of a leaf inserted into a level (denoted by LB). All of the LB levels are, in turn, inserted into a single level to facilitate destroying all of them when SELMA changes phases. The scratch level is used for highly temporary displays such as connection lines while they are being drawn, arrows to identify patch-up errors, etc. Whenever the plot phase is entered, the diagram level (D) is removed from the highest active level [3 p. 35] and replaced with a level into which are inserted the display structure components necessary to display a graph on the screen. This level is then again replaced with the diagram level when SELMA enters the construction or results phase.

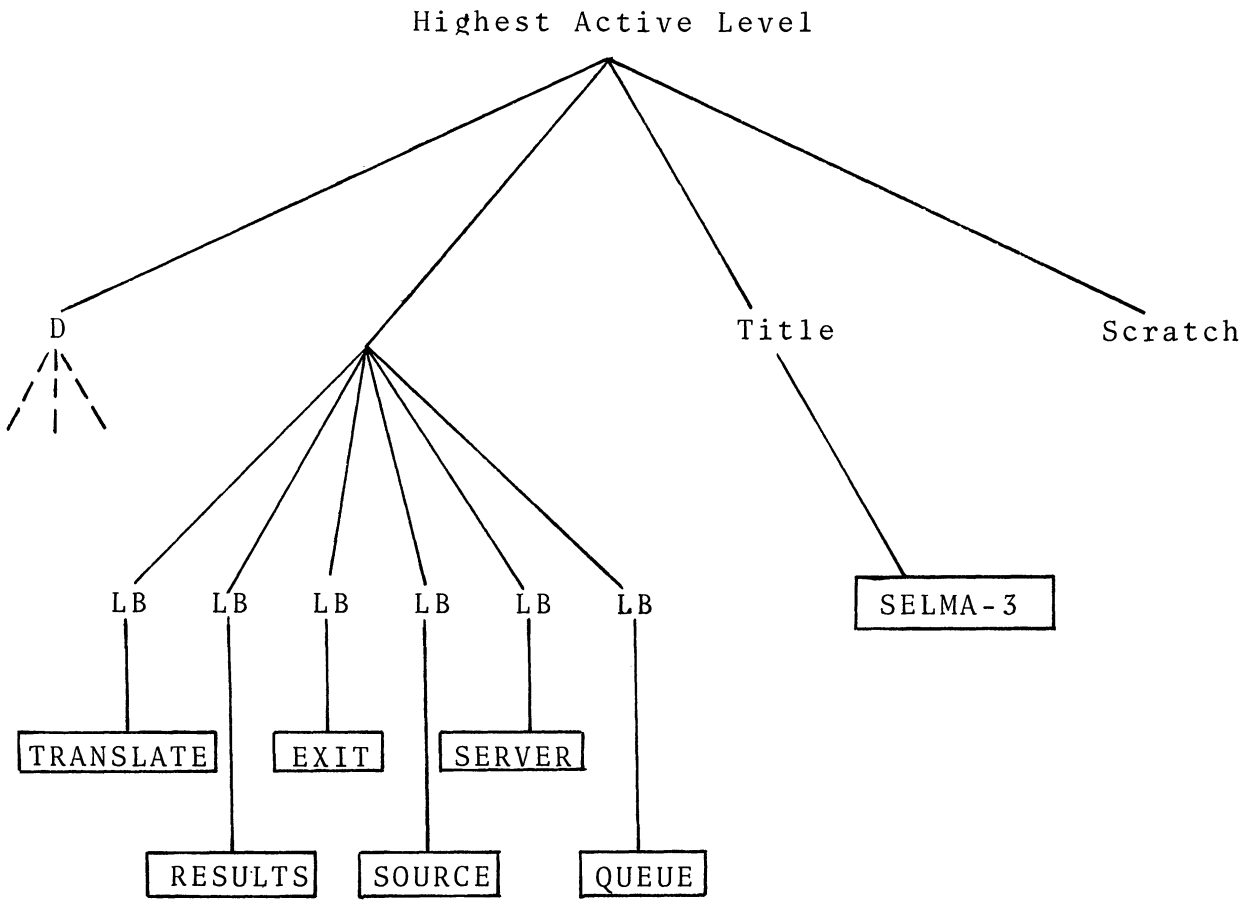


Figure 22

Construction Phase Display Structure

5. Special Command Exchanger Feature

In addition to those features described in Section 3, a special feature is included in the command exchanger to facilitate testing of future modifications to SELMA which affect the command repertoire. Whenever SELMA is started, but the dataphone is not connected to another party, the teletype is substituted for the QAS program. This is accomplished by the addition of a command to the keyboard interpreter to permit simulated input from QAS and by the routing of all command output to the teleprinter (to be typed in hexadecimal format).

The command which is added to the keyboard interpreter is of the following form:

QAS: [_ _ _ _]

The command is begun by typing the letter "Q" on the keyboard. A command is then typed (except for delimiter bytes) on the keyboard in hexadecimal notation. If a carriage return is typed instead of the command, a null command is assumed. Otherwise, an even number of hexadecimal digits must be typed, followed by the carriage return. (In all cases, the right bracket is typed in response to the carriage return.) If a phase byte is explicitly inputted, a carriage return will be ignored until the command byte is inputted. The command may be deleted at any time before it is completed by typing a null character [3 p. 17].

Because the command exchanger actually interprets the QAS keyboard command, commands between the user and SELMA via teletype are subject to the same restrictions that govern commands between QAS and SELMA. The first command via teletype must be inputted by the user, and only one command is typed by SELMA until the next command is inputted via keyboard. For example, assume that SELMA is started with the data set disconnected. Only after the user supplies the command

QAS: [0000]

does SELMA respond by typing

SELMA: [0005]

to indicate that QAS should be initialized. (The zeros in the QAS keyboard command were not underlined here because it is assumed that all null commands are inputted simply by typing a carriage return.) If the user then again inputs a null command, no response from SELMA is produced until the user performs some action which generates one or more commands. The first of these generated commands is then typed, and the others are retained until more QAS commands are inputted.

Since the only communication between SELMA and QAS is in the form of commands, the feature described above permits complete testing of new versions of SELMA without the use of the IBM 360/67. This feature of the command exchanger is intended only for this purpose, and should not be used by users who are attempting to solve queueing models.

6. Foreseeable Modifications

The system described in this report was designed to demonstrate the usefulness of graphical input for the specification of queueing networks to a computer. It was not intended to be a tool for the queueing analyst. However, with a few modifications, the system could serve the latter purpose as well.

The elements which are available in SELMA (i.e., queue, server, source, and exit) may not be sufficient for the specification of many models. In fact, various users may disagree on what elements are sufficient. Since a universally acceptable set of elements is difficult, if not impossible, to derive, a definition capability for elements is desirable in SELMA. Since the "menu" of elements in SELMA is table-driven, the inclusion of this facility in SELMA would not be difficult. However, a corresponding definition facility for QAS might be difficult to implement.

Another type of definition facility could allow definition of elements which would be equivalent to existing fragments. This definition facility could be supported exclusively by the display terminal, since the creation of a copy of such an element could be described to QAS by the sequence of commands required to generate the corresponding fragment. Each element defined in this way could be represented by a symbol such as one which is used to represent an element in the current version of SELMA. Much larger models than can presently be accommodated in the display terminal could then be accommodated because the detail of each fragment which represents a composite element would not have to be stored locally once the equivalent element was defined.

For SELMA/QAS to be a useful tool, results should be available in forms other than probability distributions for individual state variables. A suitable generalization of this type of result might be a conditional probability distribution for an algebraic function of state variables, where the algebraic function and the conditions which affect the distribution are specified by the user. Such distributions could be further processed by QAS to produce conditional expectations of functions of state variables.

References

1. 339 Programmed Buffered Display User's Handbook, DEC-09-I6FA-D, Digital Equipment Corporation, Maynard, Massachusetts, May 1968.

2. Randall, L. S., I. S. Uppal, G. A. Mc Clain, and J. F. Blinn, Implementation of the Queue Analyzer System (QAS) on the IBM 360/67, Technical Report 22, Concomp Project; also SEL 43, Systems Engineering Laboratory, University of Michigan, Ann Arbor, Michigan, to be published.

3. Jackson, James H., An Executive System for a DEC 339 Computer Display Terminal, Technical Report 15, Concomp Project; also SEL-32, Systems Engineering Laboratory, University of Michigan, Ann Arbor, December 1968.

4. Wallace, V. L., and K. B. Irani, Network Models for the Conversational Design of Stochastic Service Systems, Technical Report 13, Concomp Project; also SEL-30, Systems Engineering Laboratory, University of Michigan, Ann Arbor, November 1968.

5. Wallace, V. L., and K. B. Irani, A System for the Solution of Simple Stochastic Networks, Technical Report 14, Concomp Project; also SEL 31, Systems Engineering Laboratory, University of Michigan, Ann Arbor, to be published.

Appendix

Program Listing

```
       $TITLE                  SELMA/QAS COMMAND EXCHANGER

QASEOR $EQU 27                 END OF RECORD

* QAS IS A TASK WHICH COMMUNICATES WITH THE /360 QAS PROGRAM.

QAS    DZM QASCM+2
       DZM QASCMR+2
       DZM QASCMS
       LAC =QASQ
       JMS* =Q.C
       LAC =QAST4
       DAC QASB
       LAC =740000
       DAC QASR+1
       LAW 610
       JMS* =T.A
       LAW 10
       JMS* =T.R
QASCMD JMS QASR
       SAD =377
       SKP
       JMP QASCMD
       JMS QASR
       DAC QASP
       JMS QASR
       DAC QASC
       LAC QASPHS
       TAD QASP
       SMA
       JMP QASE
       LAC =QASPHS+1
       TAD QASP
       DAC QAST
       LAC* QAST
       DAC QAST
       LAC* QAST
       TAD QASC
       SMA
       JMP QASE
```

```
        LAC QAST
        TAD QASC
        TAD =1
        DAC QAST
        XCT* QAST

QASE    LAW 10
        JMS* =T.A
        LAC =QAST1
        JMS* =L.T
        LAC QASP
        SAD =377
        XOR =400
        SPA
        JMP KBDRPY
        JMS QASBYT
        LAC QASC
        SAD =377
        XOR =400
        SPA
        JMP KBDRPY
        JMS QASBYT
        JMS QASR
        SAD =377
        JMP *-2
        SPA
        JMP KBDRPY
        JMS QASBYT
        JMP *-6

Q0000   JMS QASR
        SMA
        JMP Q0000
        LAC QASQ+1
        SAD QASQ+2
        JMP Q00002
        LAC =377
        JMS QASS
        DZM Q00004
        LAC =QASQ
```

```
        JMP *+3
Q00001  LAC =QASQ
        DAC Q00004
        JMS* =Q.F
        JMP Q00003
        DAC Q00005
        LRS 11
        AND =377
        JMS QASS
        LAC Q00005
        LRS 11
        AND =400
        SZA
        JMP *+22
        LAC Q00005
        AND =377
        JMS QASS
        LAC Q00004
        SZA
        JMP *+10
        LAC Q00005
        SAD S0002
        JMP *+14
        SAD S0003
        JMP *+12
        SAD S0004
        JMP *+10
        LAC Q00005
        AND =400
        SNA
        JMP Q00001
        LAC =377
        JMS QASS
        JMP QASCMD
        LAC =377
        JMS QASS
        LAC QASR+1
        SAD =740000
        LAW 600
        SAD *-1
```

```
       JMS* =T.R
       JMS* =T.F
       JMP Q0000+3
Q00002 JMS* =T.P
       JMP Q00001
Q00003 JMS* =T.P

Q0002  DZM QASCMS
       JMP Q0000
```

```
* READ BYTE FROM COMMAND SOURCE
* CALLING SEQUENCE:
*       JMS QASR
*       ----                    (RETURN)
* AC CONTENT ON RETURN:
*   BYTE FROM COMMAND SOURCE

QASR    $DC 0
        $DC 0
        JMS* =B.FI
        SKP
        JMP* QASR
        LAW 10
        JMS* =T.A
        LAC =QAST2
        JMS* =L.T
        LAW 610
        JMS* =T.R
        LAC =560000+QASR1
        DAC QASR+1
QASR1   JMS KBDBYT
        JMP* QASR
```

```
* SEND BYTE TO COMMAND SINK
* CALLING SEQUENCE:
*       JMS QASS
*       ----                    (RETURN)
* AC CONTENT ON ENTRY:
*    BYTE TO BE SENT

QASS    $DC 0
        DAC QASS2
        JMS* =B.FO
        JMP QASS1
        LAC QASS2
        JMS QASBRK
        SAD =QAST4
        LAW QASEOR
        SAD *-1
        JMS* =B.FO
        NOP
        JMP* QASS
QASS1   LAC QASS2
        JMS QASBYT
        JMP* QASS
```

```
* GET BRACKET TEXT
* CALLING SEQUENCE:
*       JMS QASBRK
*       ----                        (RETURN)
* AC CONTENT ON ENTRY:
*   BYTE TO BE TESTED
* AC CONTENT ON RETURN:
*   GIVEN BYTE IF GIVEN BYTE WAS NOT 'FF'
*   POINTER TO BRACKET TEXT LIST IF GIVEN BYTE WAS 'FF'

QASBRK $DC 0
       SAD =377
       SKP
       JMP* QASBRK
       LAC QASB
       SAD =QAST4
       JMP *+3
       LAC =QAST4
       SKP
       LAC =QAST3
       DAC QASB
       JMP* QASBRK
```

```
* TYPE BYTE ON TELEPRINTER
* CALLING SEQUENCE:
*       JMS QASBYT
*       ----                    (RETURN)
* AC CONTENT ON ENTRY:
*    BYTE TO BE TYPED

QASBYT $DC 0
       SAD =377
       JMP *+10
       AND =377
       LRS 4
       ALSS 2
       LLS 4
       XOR =770000
       JMS* =B.T
       JMP* QASBYT
       JMS QASBRK
       SAD =QAST4
       JMP *+4
       LAW 10
       JMS* =T.A
       LAC =QAST3
       JMS* =L.T
       LAC QASB
       SAD =QAST3
       JMP* QASBYT
       LAW 10
       JMS* =T.R
       JMP* QASBYT
```

```
* PUT OUTPUT COMMAND ON QUEUE
* CALLING SEQUENCE:
*       JMS QASCM
*       ----                      (RETURN)
* AC CONTENT ON ENTRY:
*    POINTER TO COMMAND

QASCM   $DC 0
        JMS* =T.L
        $DC 0
        DAC QASCM3
QASCM1  LAC* QASCM3
        LMQ
        LAC =QASQ
        JMS* =Q.A
        JMP QASCM2
        LAC* QASCM3
        ISZ QASCM3
        AND =400400
        SNA
        JMP QASCM1
        JMS* =T.U
        $DC QASCM
        JMP QASCM1
QASCM2  JMS* =T.P
```

```
* REQUEST QAS OUTPUT
* CALLING SEQUENCE:
*       JMS QASCMR
*       ----                    (RETURN)
* AC CONTENT ON ENTRY:
*    POINTER TO COMMAND

QASCMR $DC 0
       JMS* =T.L
       $DC 0
       DAC QASCM4
       LAC QASCMS
       SZA+CLC
       JMP *+7
       DAC QASCMS
       LAC QASCM4
       JMS QASCM
       JMS* =T.U
       $DC QASCMR
       JMP QASCMR+4
       JMS* =T.P
```

```
* TEXT LISTS

QAST1   $DC 7
        $TEXT "*** ILLEGAL COMMAND ["

QAST2   $DC 31
        $TEXT "*** DATA SET DISCONNECTE"
        $DC 157475
        $TEXT "*** COMMAND SOURCE & SINK SWITCHED TO TELETYP"
        $DC 167475

QAST3   $DC 3
        $TEXT "SELMA: ["

QAST4   $DC 1
        $DC 537475

QASQ    $DC *+400
        $DS 400
```

```
* RECEIVED COMMAND DECODING TABLES

QASPHS $DC -4
       $DC QAS00                 CONTROL PHASE
       $DC QAS01                 DISPLAY RESULTS PHASE
       $DC QAS02                 REGENERATION PHASE
       $DC QAS03                 SAVE PHASE

QAS00  $DC -3
       JMP Q0000                 NULL
       JMP Q0001                 PATCH-UP
       JMP Q0002                 END OF FILE

QAS01  $DC -3
       JMP Q0100                 SET UP GRAPH
       JMP Q0101                 PLOT VALUES
       JMP Q0102                 DISPLAY SINGLE VALUE

QAS02  $DC -6
       JMP Q0200                 CREATE FRAGMENT
       JMP Q0201                 CREATE ELEMENT OR CONNECTION
       JMP Q0202                 LOAD CONNECTION SEGMENT
       JMP Q0203                 INSERT CONNECTION LEAF
       JMP Q0204                 ASSIGN PARAMETER
       JMP Q0205                 CONNECT

QAS03  $DC -1
       JMP Q0300                 SEND MODEL TO BE SAVED
```

```
* TRANSMITTED COMMAND TABLES

* CONTROL PHASE
S0000  $DC 400              NULL COMMAND
S0001  $DC 401              PATCH-UP
S0002  $DC 402              CALL SYSTEM
S0003  $DC 403              CALL ERROR
S0004  $DC 404              CALL MTS
S0005  $DC 405              INITIALIZE
S0006  $DC 406              WIPE OUT /360 OUTPUT BUFFER

* GENERATION PHASE
S0100  $DC 1000             CREATE ELEMENT OR CONNECTION
       $DC 0                NAME/TYPE
       $DC 0                GEN PAR VALUES (NO. 1 & 2)
S0101  $DC 1001             DESTROY ELEMENT OR CONNECTION
       $DC 0                NAME (HIGH ORDER)
S0102  $DC 1002             ASSIGN PARAMETER VALUE
       $DC 0                NAME/PARAMETER NUMBER
       $DS 4                PARAMETER VALUE (6-BIT CODE)
S0103  $DC 1003             CONNECT
       $DC 0                CON NAME/CON PORT NUMBER
       $DC 0                ELEMENT NAME/ELEMENT PORT NUMBER
S0104  $DC 1004             DISCONNECT
       $DC 0                ELEMENT NAME/ELEMENT PORT NUMBER
S0105  $DC 1005             ALTER GENERATION PARAMETER
       $DC 0                NAME/GENERATION PARAMETER NUMBER
       $DC 0                GEN PAR VALUE (HIGH ORDER)

* SOLVE PHASE
S0200  $DC 2000             COMPILE & SOLVE
       $DC 0                NUMBER OF ITERATIONS
       $DS 4                CONVERGENCE FACTOR (6-BIT CODE)

* RESULTS PHASE
S0300  $DC 3000             PLOT RESULTS
       $DC 0                ELT NO. (0=ALL) (HIGH ORDER)

S0301  $DC 3001             MODIFY PLOT
       $DC 0                FIRST STATE NUMBER
```

```
        $DC 0                   COUNT (HIGH ORDER)

S0302   $DC 3002                GET SINGLE VALUE
        $DC 0                   STATE NUMBER

S0303   $DC 3003                TYPE RESULTS
        $DC 0                   ELT NO. (0=ALL) (HIGH ORDER)

* DOCUMENTATION PHASE
S0400   $DC 4000                BEGIN SAVE
        $DS 14                  FILE NAME
S0401   $DC 4401                END SAVE
S0402   $DC 4002                RETRIEVE
        $DS 14                  FILE NAME
S0403   $DC 4003                SAVE RETURN COMMAND
        $DS 42                  RETURN COMMAND
```

```
       $TITLE                  SELMA RESPONSE TO QAS COMMANDS
Q0001  DZM QASCMS
       JMS QASR
       SAD =1
       JMP Q00011
       SAD =4
       SKP
       JMP QASE
       LAC =Q0001F
       JMS* =L.T
       JMS QASR
       SNA
       JMP *+3
       LAC =Q0001G
       JMS* =L.T
       JMP Q0000
       LAC =Q0001S
       JMP *-3
Q00011 LAC =KBD1
       DAC KBDTBL
       LAC =PBT
       JMS* =P.T
       JMS QASR
       SAD =2
       JMP Q00012
       JMS QASR
       JMS QASR
       LAC =Q0001C
       SKP
Q00012 LAC =Q0001P
       DAC Q0001T
       JMS QASR
       TAD =NAME
       DAC Q0001N
       LAC* Q0001N
       JMS ATTR
       $DC 0
       CLA
       JMS ATTR
```

```
$DC 0
DAC Q0001L
LAC Q0001T
SAD =Q0001C
JMP *+5
CLA
JMS ATTR
$DC 0
DAC Q0001L
JMS QASR
SZA
JMP *+4
LAC =Q0001R
DAC Q0001T
LAC =1
DAC Q0001N
LAC Q0001L
JMS ATTR
$DC 0
LMQ
LAC Q0001L
JMS PENCNT
CMA
TAD Q0001N
DAC Q0001N
LAC Q0001L
SKP
CLA
JMS ATTR
$DC 0
ISZ Q0001N
JMP *-4
DAC Q0001L
LAC Q0001T
JMS* =L.T
LAC Q0001L
JMS ATTR
$DC 0
DAC Q0001T
LMQ
```

```
LAC Q0001L
JMS* =S.TR
$DC 0
LAC =Q0001A
LMQ
LAC Q0001L
JMS* =S.TI
$DC 0                    *
LAC Q0001T
LMQ
LAC Q0001L
JMS* =S.TI
$DC 0                    *
LAC =Q0001E
JMS* =L.T
LAC =CON
JMS* =T.S
JMP Q0000
```

```
* SET UP GRAPH

Q0100   JMS PLOCLR
        JMS QASR
        ALSS 7
        DAC PLOINC
        JMS QASR
        XOR PLOINC
        LRSS 5
        DAC PLOINC
        JMS PLOLBL
        $DC -320
        $DC -250
        JMS PLOLBL
        $DC -320
        $DC 250
        JMS PLOLBL
        $DC -310
        $DC -340
        JMS PLOLBL
        $DC 310
        $DC -340
        JMS PLOLBL
        $DC 330
        $DC -300
Q01001  LAC =S0000
        JMS QASCM
        JMP Q0000
```

* PLOT VALUES

```
Q0101  JMS QASR
       SAD =377
       JMP Q01001
       ALSS 7
       DAC Q01011
       JMS QASR
       XOR Q01011
       LRSS 5
       XOR =4000
       DAC Q01011
       LAC =1
       DAC* PLOPTR
       ISZ PLOPTR
       LAC Q01011
       DAC* PLOPTR
       ISZ PLOPTR
       ISZ PLOPTR
       LAW 4000
       DAC* PLOPTR
       ISZ PLOPTR
       CLC
       TAD PLOINC
       DAC* PLOPTR
       ISZ PLOPTR
       LAC Q01011
       XOR =2000
       DAC* PLOPTR
       LAC PLOPTR
       TAD =3
       DAC PLOPTR
       JMP Q0101
```

```
* DISPLAY SINGLE VALUE

Q0102  LAC LVLVX
       LMQ
       LAC LVLSCR
       JMS* =S.TR
       JMP *+3
       LAC LVLVX
       JMS CHEW
       LAC LVLVY
       LMQ
       LAC LVLSCR
       JMS* =S.TR
       JMP *+3
       LAC LVLVY
       JMS CHEW
       LAC PENGPI
       LRS 3
       AND =77
       MUL
PLOINC $DC 0
       LACQ
       TAD =-300
       RAL
       TAD PLOINC
       RAR
       DAC *+3
       JMS PLOLBL
       $DC -350
       $DC 0
       DAC LVLVX
       JMS PLOLBL
Q01021 $DC 0
       $DC -340
       DAC LVLVY
       JMS* =D.E
       JMP Q0000
```

```
* CREATE FRAGMENT

Q0200   JMS* =S.TL
        $DC 0                    *
        DAC Q02FRG
        LMQ
        LAC LVLDGM
        JMS* =S.TI
        $DC 0                    *
        LAC Q02FRG
        JMS Q02CRD
Q02001  LAC =S0000
        JMS QASCM
        JMP Q0000
```

```
* CREATE ELEMENT OR CONNECTION

Q0201   JMS* =S.TL
        $DC 0                 *
        DAC Q02PAR
        JMS LVL
        $DC Q02FRG
        $DC 0
        $DC 0
        $DC 0
        $DC 0                 *
        DAC Q02SYM
        JMS Q02CRD
        JMS* =S.TL
        $DC 0                 *
        DAC Q02PRT
        LMQ
        LAC Q02SYM
        JMS* =S.TI
        $DC 0                 *
        JMS QASR
        TAD =NAME
        DAC Q02TMP
        LAC Q02SYM
        DAC* Q02TMP
        LAW -NAME
        TAD Q02TMP
        AND =200
        SZA
        JMP Q02013
        JMS QASR
        JMS QASR
        TAD =Q0201T-1
        DAC Q02TMP
        LAC* Q02TMP
        DAC Q02TMP
Q02011  LAC* Q02TMP
        SMA
        JMP Q02012
        ALS 2
```

```
      LRSS 12
      DAC *+14
      LACQ
      LRSS 12
      DAC *+12
      LAC* Q02TMP
      AND =600000
      SAD =400000
      LAC =INP
      SAD =600000
      LAC =OUT
      JMS LVL
      $DC Q02PRT
      $DC 0
      $DC 0
      $DC 500
      $DC 0                  *
      ISZ Q02TMP
      JMP Q02011
Q02012 LAC Q02TMP
      TAD =1
      LMQ
      LAC Q02SYM
      JMS* =S.TI
      $DC 0                  *
      JMP Q02001
Q02013 JMS QASR
      JMS QASR
      ALSS 7
      DAC Q02TMP
      JMS QASR
      XOR Q02TMP
      JMS CORGET
      $DC 0                  *
      DAC Q02BLK
      DAC Q02PTR
      LAW 10
      LMQ
      LAC Q02PRT
      JMS* =S.LP
```

```
        JMP Q02001

Q0201T  $DC DIC1+10
        $DC DIC2+10
        $DC DIC3+7
        $DC DIC4+7
```

```
* LOAD CONNECTION LEAF SEGMENT

Q0202  JMS QASR
       SAD =377
       JMP Q02001
       LRSS 5
       ALSS 4
       LLS 5
       ALSS 7
       DAC Q02TMP
       JMS QASR
       XOR Q02TMP
       DAC* Q02PTR
       ISZ Q02PTR
       JMP Q0202
```

```
* INSERT CONNECTION LEAF

Q0203  LAC Q02BLK
       TAD =1
       LMQ
       LAC Q02SYM
       JMS* =S.TI
       $DC 0                    *
       JMP Q02001
```

```
* ASSIGN PARAMETER

Q0204  LAC Q02SYM
       JMS ATTR
       $DC 0
       TAD =-1
       DAC Q02TMP
       LAC* Q02TMP
       AND =377
       SAD =1
       LAC =PARINT
       SAD =202
       LAC =PARINT
       SAD =PARINT
       SKP
       LAC =PAR
       DAC Q02MOD
       JMS* =S.TL
       $DC 0                 *
       DAC Q02TMP
       LMQ
       LAC Q02PAR
       JMS* =S.TI
       $DC 0                 *
       LAC Q02TMP
       JMS Q02CRD
       LAW 500
       LMQ
       LAC Q02TMP
       JMS* =S.LP
       LAC =Q02BUF+1
       DAC Q02PTR
       DZM Q02BUF
       JMS QASR
       SAD =377
       JMP Q02042
Q02041 XOR =777700
       DAC* Q02PTR
       ISZ Q02BUF
       ISZ Q02PTR
```

```
        JMS QASR
        SAD =377
        SKP
        JMP Q02041
        LAC =Q02BUF
        JMS* =L.D
        $DC 0                    *
        DAC Q02BLK
        LAC* Q02MOD
        AND =400000
        XOR =2010
        DAC* Q02BLK
        LAC Q02BUF
        ALSS 14
        XOR* Q02BLK
        DAC* Q02BLK
        LAC Q02BLK
        SKP
Q02042  LAC Q02MOD
        LMQ
        LAC Q02TMP
        JMS* =S.TI
        $DC 0                    *
        JMP Q02001

Q02BUF  $DS 5
```

```
* CONNECT

Q0205   JMS  QASR
        TAD  =NAME
        DAC  Q02TMP
        LAC* Q02TMP
        JMS  ATTR
        $DC  0
        CLA
        JMS  ATTR
        $DC  0
        DAC  Q02PTR
        JMS  ATTR
        $DC  0
        DAC  Q02TMP
        LMQ
        LAC  Q02PTR
        JMS  PENCNT
        CMA
        DAC  Q02BLK
        JMS  QASR
        TAD  Q02BLK
        DAC  Q02BLK
        LAC  Q02PTR
        SKP
Q02051  CLA
        JMS  ATTR
        $DC  0
        DAC  Q02TMP
        ISZ  Q02BLK
        JMP  Q02051
        LAC  Q02TMP
        JMS  ATTR
        $DC  0
        DAC  Q02PTR
        LMQ
        LAC  Q02TMP
        JMS* =S.TR
        $DC  0
        LAC  Q02PTR
```

```
SAD =INP
LAC =INPCON
SAD =OUT
LAC =OUTCON
LMQ
LAC Q02TMP
JMS* =S.TI
$DC 0                    *
LAC Q02TMP
LMQ
LAC Q02PRT
JMS* =S.TI
$DC 0                    *
JMP Q02001
```

```
* SAVE DIAGRAM

Q0300  DZM KBDSS
       JMP Q0000



* TRANSLATE LEVEL TO DATAPHONE COORDINATES
* CALLING SEQUENCE:
*       JMS Q02CRD
*       ----                    (RETURN)
* AC CONTENT ON ENTRY:
*   POINTER TO LEVEL TO BE TRANSLATED

Q02CRD $DC 0
       DAC Q02CR2
       JMS Q02CR1
       JMS* =S.LY
       JMS Q02CR1
       JMS* =S.LX
       JMP* Q02CRD

Q02CR1 $DC 0
       JMS QASR
       ALSS 7
       DAC Q02CR3
       JMS QASR
       XOR Q02CR3
       ALS 4
       LRSS 26
       LAC Q02CR2
       JMP* Q02CR1
```

```
Q0001A $DC 500
       $DC 6302
       VEC
       $DC 2010
       $DC 10
       $DC 6020
       $DC 40
       $DC 2020
       $DC 2020
       $DC 4040
       $DC 2020
       $DC 6200
       $DC 200
       $DC 210
       $DC 6210
       $DC 6301
       POP
```

```
Q0001F $DC 7
       $TEXT "*** BAD FILE NAME"
       $DC 747577

Q0001G $DC 11
       $TEXT "*** RETRIEVAL CANCELLED"
       $DC 747577

Q0001S $DC 13
       $TEXT "*** SAVE OPERATION CANCELLED"
       $DC 747577

Q0001C $DC 11
       $TEXT "*** PORT NOT CONNECTED"
       $DC 747577

Q0001P $DC 7
       $TEXT "*** BAD PARAMETER"
       $DC 747577

Q0001R $DC 15
       $TEXT "*** PROBABILITIES DO NOT SUM TO 1.0"
       $DC 747577

Q0001E $DC 13
       $TEXT "*** RESULT REQUEST CANCELLED"
       $DC 747577
```

```
        $TITLE                  SELMA KEYBOARD INTERPRETER

KBD     LAW -7
        DAC KBDI
        LAC =KBDI
        DAC KBDTBL
        LAW 30
        JMS* =T.A
KBDINT  LAW 10
        JMS* =T.R
        JMS KBDXCT
        LAC KBDTBL /PAR
```

```
* CLEAR COMMAND

KBDC   JMS KBDTP
       LAC =KBDCT+400000 /PAR
       LAC =KBDCCM
       JMS* =T.S
       LAC LVLDGM
       JMS CHEWA
       JMS* =X.T
       LAW -400
       JMS CORZRO
       LAC =NAME /PAR
       JMP KBDINT

KBDCCM LAC =S0005
       JMS QASCM
       JMS* =T.F
```

```
* ESCAPE COMMAND

KBDE    JMS KBDTP
        LAC =KBDET+400000 /PAR
        LAC LVLHAL
        JMS CHEWA
        CLA
        JMS* =D.P
        JMS* =X.T
        LAC =KBDECM
        JMS* =T.S
        LAC QASR+1
        SAD =740000
        JMP *+12
        LAC QASQ+1
        SAD QASQ+2
        LAC QASR
        AND =77777
        SAD =Q0000+1
        JMP *+4
        LAC =KBD3
        DAC KBDTBL
        JMP KBDINT
        CLA
        JMS* =P.T
        LAW 33
        JMS* =T.R
        JMS* =T.F

KBDECM  LAC =S0002
        JMS QASCM
        JMS* =T.F
```

```
* GET COMMAND

KBDG    JMS KBDTP
        LAC =KBDGT /PAR
        JMS KBDFL
        $DC S0402+1
        LAC =PBTR+2
        JMS* =P.T
        CLA
        JMS* =D.P
        LAW 10
        LMQ
        LAC LVLBUT
        JMS* =S.LP
        JMS* =X.S
        JMS* =T.P
        LAC LVLDGM
        JMS CHEWA
        LAW -400
        JMS CORZRO
        LAC =NAME /PAR
        LAC =S0005
        JMS QASCM
        LAC =KBDINT
        JMS* =T.S
        LAC =S0402
        JMS QASCMR
KBDG1   LAC QASCMS
        SZA
        JMP KBDG2
        LAC TRAPEN
        JMS PENSET
        LAC =PBT
        JMS* =P.T
        LAC LVLBUT
        CLQ
        JMS* =S.LP
        LAC KBDFSV
        DAC KBDTBL
        JMS* =T.F
```

```
        JMP KBDG1
KBDG2   JMS* =T.P
```

```
* PARAMETER COMMAND

KBDP    LAC KBDVSW
        SNA
        JMP KBDINT+2
        JMS KBDTP
        LAC =KBDPT  /PAR
        DZM KBDPM
        LAC KBDTBL
        DAC KBDPSV
        LAC =KBD5
        DAC KBDTBL
        LAC TRAPEN
        DAC KBDPLP
        LAC =PENPAR
        JMS PENSET
        JMS KBDP1
        JMS KBDP1
        JMS KBDP1
        JMS KBDP1
        LAC KBDPM
        SNA
        JMP *+5
        JMS KBDP1
        JMS KBDP1
        JMS KBDP1
        JMS KBDP1
        LAC =KBD6
        DAC KBDTBL
        JMS KBDP1

KBDP1   $DC 0
        JMS KBDXCT
        LAC KBDTBL  /PAR

KBDP2   XOR =777700
        ISZ KBDPP
        DAC* KBDPP
        ISZ KBDPB
        JMS* =B.T
```

```
        JMP* KBDP1

KBDP3   LAC KBDPM
        SNA+CLC
        JMP *+4
        TAD KBDP1
        DAC KBDP1
        JMP* KBDP1
        LAW 17761
        DAC KBDPM
        JMP KBDP2+1

KBDP4   DZM KBDPB
        LAC =KBDPB
        DAC KBDPP
        LAC KBDPLP
        JMS PENSET
        LAC KBDPSV
        DAC KBDTBL
        JMP KBDCM2

KBDPB   $DC 0
        $DC 777777
        $DC 777777
        $DC 777777
        $DC 777777
        $DC 777777
        $DC 777777
        $DC 777777
        $DC 777777
```

```
* QAS COMMAND

KBDQ    LAC QASR+1
        SAD =740000
        JMP KBDINT
        JMS KBDTP
        LAC =KBDQT /PAR
        JMS KBDCM
        LAC =KBDQQ /PAR
        LAW -6
        DAC KBDI
        LAC QASQ+2
        SAD QASQ+1
        JMP KBDINT
        SAD QASQ
        LAC =QASQ+2
        TAD =1
        DAC KBDQ1
        LAC* KBDQ1
        SAD S0002
        SKP
        SAD S0003
        SKP
        SAD S0004
        SKP
        JMP KBDINT
        LAW 31
        JMS* =T.R
        JMS* =T.F

KBDQQ   $DC *+47
        $DS 47
```

```
* REPLY COMMAND

KBDR    JMS KBDTP
        LAC =KBDRT /PAR
        JMS KBDCM
        LAC =KBDRQ /PAR
        LAW 10
        JMS* =T.R
        LAC KBDRSV
        DAC KBDTBL
        JMS KBDR2
        DAC KBDSW
        JMS KBDR2
        SZA
        DZM KBDSW
        DAC QASP
        JMS KBDR2
        SAD =2
        SKP
        SAD =3
        SKP
        SAD =4
        SKP
        DZM KBDSW
        DAC QASC
        LAC KBDSW
        SNA
        JMP KBDR1
        LAC =QASQ
        JMS* =Q.C
        LAC =S0000
        TAD QASC
        JMS QASCM
        LAC LVLHAL
        JMS CHEWA
        CLA
        JMS* =D.P
        JMS* =X.T
        LAW 21
        JMS* =T.R
```

```
        JMS* =T.F
KBDR1   LAC =377
        JMS QASS
        LAC QASP
        JMS QASS
        LAC QASC
        JMS QASS
        JMS KBDR2
        SPA
        JMP KBDINT+2
        JMS QASS
        JMP *-4

KBDR2   $DC 0
        LAC =KBDRQ
        JMS* =Q.F
        $DC 0
        JMP* KBDR2

KBDRQ   $DC *+47
        $DS 47
```

```
* SAVE COMMAND

KBDS    JMS KBDTP
        LAC =KBDST /PAR
        JMS KBDFL
        $DC S0400+1
        LAC =PBTR+2
        JMS* =P.T
        CLA
        JMS* =D.P
        LAW 10
        LMQ
        LAC LVLBUT
        JMS* =S.LP
        JMS* =X.S
        JMS* =T.P
        LAC =KBDINT
        JMS* =T.S
        LAC =S0400
        DAC KBDSS
        JMS QASCMR
        LAC QASCMS
        SNA
        JMP KBDG1
        LAC KBDSS
        SNA
        JMP *+4
        SKP
        JMP *-7
        JMS* =T.P
        LAC =KBDSFQ
        JMS* =Q.C
        LAC LVLDGM
KBDS1   JMS ATTR
        JMP KBDS2
        LMQ
        LAC =KBDSFQ
        JMS* =Q.A
        $DC 0
        CLA
```

```
        JMP KBDS1
KBDS2   LAC =KBDSFQ
        JMS* =Q.F
        JMP KBDS11
        DAC KBDSF
        LAC =2000
        DAC S0403+1
        LAC KBDSF
        JMS KBDCRD
        LAC S0403+3
        XOR =400
        DAC S0403+3
        LAC =S0403
        JMS QASCM
        LAC =KBDSQ1
        JMS* =Q.C
        LAC =KBDSQ2
        JMS* =Q.C
        LAC KBDSF
KBDS3   JMS ATTR
        JMP KBDS4
        LMQ
        LAC =KBDSQ1
        JMS* =Q.A
        $DC 0
        CLA
        JMP KBDS3
KBDS4   LAC =KBDSQ1
        JMS* =Q.F
        JMP KBDS6
        DAC KBDSS
        JMS ATTR
        $DC 0
        TAD =-1
        DAC KBDSTP
        LAC* KBDSTP
        AND =200
        SNA
        JMP KBDS5
        LAC KBDSS
```

```
        LMQ
        LAC =KBDSQ2
        JMS* =Q.A
        $DC 0
        JMP KBDS4
KBDS5   LAC =2001
        DAC S0403+1
        LAC KBDSS
        JMS KBDCRD
        LAC KBDSS
        JMS NAMGET
        $DC 0
        ALSS 11
        DAC S0403+4
        LAC KBDSTP
        SAD =DIC1+12
        LAC =401000
        SAD =DIC2+12
        LAC =402000
        SAD =DIC3+10
        LAC =403000
        SAD =DIC4+10
        LAC =404000
        DAC S0403+5
        LAC =S0403
        JMS QASCM
        JMS KBDPAR
        JMP KBDS4
KBDS6   LAC =KBDSQ2
        JMS* =Q.F
        JMP KBDS2
        DAC KBDSS
        LAC =2001
        DAC S0403+1
        LAC KBDSS
        JMS KBDCRD
        LAC KBDSS
        JMS NAMGET
        $DC 0
        ALSS 11
```

```
        DAC S0403+4
        LAC KBDSS
        JMS ATTR
        $DC 0
        TAD =-2
        DAC KBDSTP
        LAC* KBDSTP
        ISZ KBDSTP
        TAD =-1
        DAC KBDSC
        JMS KBDCOD
        XOR =400
        DAC S0403+5
        LAC =S0403
        JMS QASCM
        LAC =2002
        DAC S0403+1
        LAC KBDSC
        CMA
        TAD =1
        DAC KBDSC
KBDS7   LAC =S0403+2
        DAC KBDSP
        LAW -41
        DAC KBDSWC
KBDS8   LAC* KBDSTP
        ISZ KBDSTP
        LRS 14
        RTR
        RTR
        LLS 14
        JMS KBDCOD
        DAC* KBDSP
        ISZ KBDSP
        ISZ KBDSC
        SKP
        JMP KBDS9
        ISZ KBDSWC
        JMP KBDS8
        CLC
```

```
        TAD KBDSP
        DAC KBDSP
        LAC* KBDSP
        XOR =400
        DAC* KBDSP
        LAC =S0403
        JMS QASCM
        JMP KBDS7
KBDS9   CLC
        TAD KBDSP
        DAC KBDSP
        LAC* KBDSP
        XOR =400
        DAC* KBDSP
        LAC =S0403
        JMS QASCM
        LAC =2403
        DAC S0403+1
        LAC =S0403
        JMS QASCM
        JMS KBDPAR
        LAC =KBDSQ3
        JMS* =Q.C
        LAC =2005
        DAC S0403+1
        LAC KBDSS
        JMS ATTR
        $DC 0
        CLA
        JMS ATTR
        $DC 0
        DAC KBDSSV
        JMS ATTR
        JMP KBDS10
        LMQ
        LAC =KBDSQ3
        JMS* =Q.I
        $DC 0
        CLA
        JMP *-7
```

```
KBDS10 LAC =KBDSQ3
       JMS* =Q.F
       JMP KBDS6
       DAC KBDSP
       LMQ
       LAC KBDSSV
       JMS* =S.TR
       $DC 0
       LAC =*+5
       LMQ
       LAC KBDSP
       JMS* =S.LU
       JMS* =T.F
       LAC KBDSP
       JMS* =S.LN
       LAC KBDSP
       LMQ
       LAC KBDSSV
       JMS* =S.TI
       $DC 0
       LAW 2
       JMS* =D.O
       $DC 0
       DAC KBDSYM
       LAW 1
       JMS* =D.O
       $DC 0
       DAC KBDPRT
       JMS* =D.E
       LAC KBDSYM
       JMS NAMGET
       $DC 0
       ALSS 11
       DAC S0403+2
       LAC KBDSP
       LMQ
       LAC KBDPRT
       JMS PENCNT
       XOR =400
       XOR S0403+2
```

```
        DAC  S0403+2
        LAC  =S0403
        JMS  QASCM
        JMP  KBDS10
KBDS11  LAC  =S0401
        JMS  QASCM
        DZM  QASCMS
        JMP  KBDGI

KBDSFQ  $DC  *+30
        $DS  30

KBDSQ1  $DC  *+30
        $DS  30

KBDSQ2  $DC  *+15
        $DS  15

KBDSQ3  $DC  *+15
        $DS  15
```

```
* VALUE COMMAND

KBDV    JMS KBDTP
        LAC =KBDVT /PAR
        JMS* =B.K
        SAD =31
        JMP *+6
        SAD =35
        JMP *+4
        SAD =77
        JMP KBDCM2
        JMP *-7
        TAD =-31
        DAC KBDVSW
        TAD =KBDVTP
        SAD =KBDVTP+4
        TAD =KBDVTT-KBDVTP-4
        JMS* =L.T
        JMP KBDINT
```

```
* RESPONSE TO ILLEGAL COMMAND FROM QAS

KBDRPY LAC =537475
       JMS* =B.T
       LAW 10
       JMS* =T.R
       LAC KBDTBL
       DAC KBDRSV
       LAC =KBD2
       DAC KBDTBL
       CLC
       DAC KBDSW
       LAC KBDSW
       SAD =377
       JMP Q0000+3
       SZA
       JMP *+3
       JMP QASCMD
       JMP *-6
       JMS* =T.P
```

```
* RETURN BYTE FROM LAST KEYBOARD COMMAND
* CALLING SEQUENCE:
*       JMS KBDBYT
*       ----                     (RETURN)
* AC CONTENT ON RETURN:
*   BYTE IN BITS 10-17; BITS 0-9 CLEAR (UNLESS END OF RECORD)
*** CALLED FROM QAS TASK ONLY ***

KBDBYT $DC 0
       LAW -6
       SAD KBDI
       JMP *+4
       SKP
       JMP KBDBYT+1
       JMS* =T.P
       LAC =KBDQQ
       JMS* =Q.F
       $DC 0
       SMA
       JMP* KBDBYT
       LAW -7
       DAC KBDI
       CLC
       JMP* KBDBYT
```

```
* INTERPRET KEYBOARD CHARACTER
* CALLING SEQUENCE:
*       JMS KBDXCT
*       LAC ---                 (LAC POINTER TO KEYBOARD TABLE)

KBDXCT $DC 0
       JMS* =B.K
       DAC KBDXCI
       XCT* KBDXCT
       DAC KBDPTR
       LAC* KBDPTR
       DAC KBDCNT
       LAC KBDXCI
       ISZ KBDPTR
       SAD* KBDPTR
       JMP *+5
       ISZ KBDPTR
       ISZ KBDCNT
       JMP *-5
       JMP KBDXCT+1
       ISZ KBDPTR
       XCT* KBDPTR
```

```
* TYPE KEYBOARD RESPONSE AND OBTAIN CONFIRMATION IF NECESSARY
* CALLING SEQUENCE:
*       JMS KBDTP
*       LAC ----                  (LAC POINTER TO RESPONSE TEXT LIST;
*                                  BIT 0 SET IF CONFIRMATION REQUIRED)
*       ----                      (RETURN)

KBDTP   $DC 0
        LAW 10
        JMS* =T.A
        XCT* KBDTP
        JMS* =L.T
        XCT* KBDTP
        ISZ KBDTP
        SMA
        JMP* KBDTP
KBDTP1  JMS KBDXCT
        LAC =KBD4 /PAR
KBDTP2  LAC =KBDOK
        JMS* =L.T
        JMP* KBDTP
KBDTP3  LAC =KBDNO
        JMS* =L.T
        JMP KBDINT
```

```
* READ COMMAND FROM KEYBOARD
* CALLING SEQUENCE:
*       JMS KBDCM
*       LAC ----                  (LAC POINTER TO SINK QUEUE)
*       ----                      (RETURN)

KBDCM   $DC 0
        XCT* KBDCM
        JMS* =Q.C
        LAC =377
        LMQ
        XCT* KBDCM
        JMS* =Q.A
        $DC 0
        JMS KBDCM4
        JMP KBDCM3
        JMS KBDCM4
        JMP KBDCM2
        JMS KBDCM4
        SKP
        JMP *-2
KBDCM1  LAC =377
        LMQ
        XCT* KBDCM
        JMS* =Q.A
        JMP KBDCM2
        CLQ+CMQ
        XCT* KBDCM
        JMS* =Q.A
        JMP KBDCM2
        LAC =537475
        JMS* =B.T
        ISZ KBDCM
        JMP* KBDCM
KBDCM2  LAC =KBDDEL
        JMS* =L.T
        JMP KBDINT
KBDCM3  LAC =KBDNUL
        JMS* =L.T
        CLQ
```

```
        XCT* KBDCM
        JMS* =Q.A
        $DC 0
        CLQ
        XCT* KBDCM
        JMS* =Q.A
        $DC 0
        JMP KBDCM1

KBDCM4  $DC 0
        JMS KBDCM5
        JMP* KBDCM4
        DAC KBDCM7
        JMS KBDCM5
        JMS* =T.P
        LRS 4
        LAC KBDCM7
        LRSS 16
        XCT* KBDCM
        JMS* =Q.A
        JMP KBDCM2
        ISZ KBDCM4
        JMP* KBDCM4

KBDCM5  $DC 0
        JMS* =B.K
        SAD =74
        JMP* KBDCM5
        SAD =77
        JMP KBDCM2
        TAD =-20
        SMA
        JMP KBDCM5+1
        AND =777717
        DAC KBDCM6
        JMS* =B.T
        LAC KBDCM6
        AND =17
        ISZ KBDCM5
        JMP* KBDCM5
```

```
* GET FILE NAME FROM KEYBOARD
* CALLING SEQUENCE:
*       JMS KBDFL
*       $DC ----              (POINTER TO 16-BYTE VECTOR)
*       ----                  (RETURN)

KBDFL   $DC 0
        LAC* KBDFL
        DAC KBDFL3
        DAC KBDFL4
        LAW -14
        DAC KBDFL5
        DAC KBDFL6
        LAC =76076
        DAC* KBDFL3
        ISZ KBDFL3
        ISZ KBDFL5
        JMP *-3
        JMS* =B.K
        SAD =77
        JMP KBDCM2
        SAD =74
        JMP KBDCM2
        XOR =400
        ALSS 11
        DAC* KBDFL4
        LRS 11
        AND =77
        XOR =777700
        JMS* =B.T
KBDFL1  JMS* =B.K
        SAD =77
        JMP KBDCM2
        SAD =74
        JMP KBDFL2
        XOR* KBDFL4
        XOR =400400
        DAC* KBDFL4
        AND =77
        XOR =777700
```

```
        JMS* =B.T
        ISZ KBDFL6
        SKP
        JMP KBDFL2
        JMS* =B.K
        SAD =77
        JMP KBDCM2
        SAD =74
        JMP KBDFL2
        DAC KBDFL7
        LAC* KBDFL4
        AND =177177
        DAC* KBDFL4
        ISZ KBDFL4
        LAC KBDFL7
        XOR =400
        ALSS 11
        DAC* KBDFL4
        LAC KBDFL7
        XOR =777700
        JMS* =B.T
        JMP KBDFL1
KBDFL2  LAW 17475
        JMS* =B.T
        LAC KBDTBL
        DAC KBDFSV
        LAC =KBD3
        DAC KBDTBL
        ISZ KBDFL
        JMP* KBDFL
```

```
* SEND PARAMETERS
* CALLING SEQUENCE:
*       JMS KBDPAR
*       ----                        (RETURN)

KBDPAR $DC 0
       LAC =KBDSQ3
       JMS* =Q.C
       LAC =2004
       DAC S0403+1
       LAC KBDSS
       JMS ATTR
       $DC 0
       CLA
       JMS ATTR
       $DC 0
       CLA
       JMS ATTR
       JMP* KBDPAR
       JMS ATTR
       JMP *+7
       LMQ
       LAC =KBDSQ3
       JMS* =Q.I
       $DC 0
       CLA
       JMP *-7
KBDPA1 LAC =KBDSQ3
       JMS* =Q.F
       JMP* KBDPAR
       DAC KBDSP
       JMS KBDCRD
       LAC KBDSP
       JMS ATTR
       $DC 0
       DAC KBDSP
       SAD =PARINT
       JMP KBDPA3
       SAD =PAR
       JMP KBDPA3
```

```
ISZ KBDSP
LAC* KBDSP
LRS 3
AND =77000
XOR =400000
DAC S0403+4
ISZ KBDSP
LAC* KBDSP
SAD =761121
JMP KBDPA2
ISZ KBDSP
LAC* KBDSP
LRS 14
AND =77
XOR =400400
XOR S0403+4
DAC S0403+4
ISZ KBDSP
LAC* KBDSP
SAD =761121
JMP KBDPA2
ISZ KBDSP
LAC* KBDSP
LRS 3
AND =77000
XOR =400000
DAC S0403+5
LAC S0403+4
XOR =400
DAC S0403+4
ISZ KBDSP
LAC* KBDSP
SAD =761121
JMP KBDPA2
ISZ KBDSP
LAC* KBDSP
LRS 14
AND =77
XOR =400400
XOR S0403+5
```

```
       DAC S0403+5
KBDPA2 LAC =S0403
       JMS QASCM
       JMP KBDPA1
KBDPA3 LAC S0403+3
       XOR =400
       DAC S0403+3
       JMP KBDPA2
```

```
* SET UP COORDINATES FOR TRANSMISSION
* CALLING SEQUENCE:
*       JMS KBDCRD
*       ----                    (RETURN)
* AC CONTENT ON ENTRY:
*    POINTER TO LEVEL WHOSE COORDINATES ARE TO BE SENT

KBDCRD $DC 0
       TAD =4
       DAC KBDCR1
       LAC* KBDCR1
       JMS* =C.CB
       JMS KBDCOD
       DAC S0403+2
       ISZ KBDCR1
       LAC* KBDCR1
       JMS* =C.CB
       JMS KBDCOD
       DAC S0403+3
       JMP* KBDCRD
```

```
* ENCODE 14 BITS OF INFORMATION FOR TRANSMISSION
* CALLING SEQUENCE:
*       JMS KBDCOD
*       ----                    (RETURN)
* AC CONTENT ON ENTRY:
*   INFORMATION TO BE ENCODED IN BITS 4-17
* AC CONTENT ON RETURN:
*   TWO-BYTE WORD OF ENCODED INFORMATION

KBDCOD $DC 0
       LRS 7
       ALS 2
       LLS 7
       AND =177177
       JMP* KBDCOD
```

```
* KEYBOARD TABLES

KBD1    $DC -7
        $TEXT "OOC"
        JMP KBDC
        $TEXT "OOE"
        JMP KBDE
        $TEXT "OOG"
        JMP KBDG
        $TEXT "OOP"
        JMP KBDP
        $TEXT "OOS"
        JMP KBDS
        $TEXT "OOV"
        JMP KBDV
        $TEXT "OOQ"
        JMP KBDQ

KBD2    $DC -1
        $TEXT "OOR"
        JMP KBDR

KBD3    $DC -1
        $TEXT "OOQ"
        JMP KBDQ

KBD4    $DC -3
        $TEXT "OOO"
        JMP KBDTP2
        $TEXT "OON"
        JMP KBDTP3
        $DC 77
        JMP KBDTP3

KBD5    $DC -14
        $DC 0
        JMP KBDP2
        $DC 1
        JMP KBDP2
        $DC 2
```

```
        JMP KBDP2
        $DC 3
        JMP KBDP2
        $DC 4
        JMP KBDP2
        $DC 5
        JMP KBDP2
        $DC 6
        JMP KBDP2
        $DC 7
        JMP KBDP2
        $DC 10
        JMP KBDP2
        $DC 11
        JMP KBDP2
        $TEXT "00."
        JMP KBDP3
        $DC 77
        JMP KBDP4

KBD6    $DC -1
        $DC 77
        JMP KBDP4
```

```
* TEXT LISTS

KBDCT   $DC 3
        $TEXT "CLEAR? "

KBDET   $DC 3
        $TEXT "ESCAPE? "

KBDGT   $DC 3
        $TEXT "GET FILE "

KBDPT   $DC 4
        $TEXT "PARAMETER: "

KBDQT   $DC 2
        $TEXT "QAS: ["

KBDRT   $DC 3
        $TEXT "REPLY: ["

KBDST   $DC 5
        $TEXT "SAVE ON FILE "

KBDVT   $DC 4
        $TEXT "VALUES FROM "

KBDVTP  $DC 5
        $TEXT "PUSH BUTTONS"
        $DC 747577

KBDVTT  $DC 4
        $TEXT "TELETYPE"
        $DC 747577

KBDOK   $DC 2
        $TEXT "OK"
        $DC 747577

KBDNO   $DC 2
        $TEXT "NO"
```

```
        $DC 747577

KBDDEL  $DC 5
        $TEXT " -- DELETED"
        $DC 747577

KBDNUL  $DC 2
        $TEXT "0000"
```

```
        $TITLE                    SELMA PUSH BUTTON INTERPRETER
PBT     LAC KBDVSW
        SZA
        JMP PBTR
        JMS* =P.R
        SAD =1
        JMP PBTR
        CLA
        JMS* =P.T
        DZM KBDPM
        LAC KBDTBL
        DAC KBDPSV
        LAC =KBD6
        DAC KBDTBL
        LAC TRAPEN
        DAC KBDPLP
        LAC =PENPAR
        JMS PENSET
        LAW -400
        LMQ
        LAC LVLSCR
        JMS* =S.LY
        LAC =320
        LMQ
        LAC LVLSCR
        JMS* =S.LX
        JMS PBTI
        JMS PBTI
        JMS PBTI
        JMS PBTI
        JMS PBTI
PBTD    LAC LVLSCR
        JMS CHEWA
        DZM KBDPB
        LAC =KBDPB
        DAC KBDPP
        LAC KBDPLP
        JMS PENSET
        LAC KBDPSV
```

```
        DAC KBDTBL
PBTR    LAC =PBT
        JMS* =P.T
        CLA
        JMS* =P.S
        JMS* =P.E
        JMS* =T.F

PBT1    $DC 0
        JMS* =P.E
        JMS* =P.R
        SZA
        JMP *+7
        LAC KBDPB
        SNA
        JMP PBTR
        SKP
        JMP *-7
        JMS* =T.P
        SAD =1
        JMP PBTD
        NORM
        LACS
        TAD =777637
        SAD =777712
        TAD =47
        SAD KBDPM
        JMP PBT2
        SAD =777761
        DAC KBDPM
        ISZ KBDPP
        DAC* KBDPP
        ISZ KBDPB
        LAC LVLSCR
        JMS CHEWA
        LAC =KBDPB
        JMS* =L.D
        $DC 0                   *
        LMQ
        LAC LVLSCR
```

```
        JMS* =S.TI
        $DC 0                        *
        CLA
        JMS* =P.S
        JMP* PBT1
PBT2    CLA
        JMS* =P.S
        JMP PBT1+1
```

```
        $TITLE                  SELMA CONSTRUCTION FRAME

CON     JMS* =X.S
        JMP PENEND
        LAC LVLBUT
        JMS CHEWA
        LAC LVLDGM
        LMQ
        LAC LVLHAL
        JMS* =S.TR
        NOP
        LAC LVLRES
        LMQ
        LAC LVLDGM
        JMS* =S.TR
        JMP *+3
        LAC LVLRES
        JMS CHEW
        LAC LVLDGM
        LMQ
        LAC LVLHAL
        JMS* =S.TI
        $DC 0                   *
        LAC =310
        DAC CONY
        LAC =DIC1
        DAC CONDIC
        TAD =1
        DAC *+5
        TAD =1
        TAD* *+3
        DAC *+7
        JMS BUTX
        $DC 0
        $DC LVLBUT
CONY    $DC 0
        $DC 320
        $DC 500
        $DC 0
        $DC 0                   *
```

```
        LAW -50
        TAD CONY
        DAC CONY
        LAC* CONDIC
        SZA
        JMP CONY-11
        JMS BUTX
        $DC RESTXT
        $DC LVLBUT
        $DC -400
        $DC -34
        $DC 500
        $DC RES
        $DC 0                   *
        JMS BUTX
        $DC SELT2
        $DC LVLBUT
        $DC -400
        $DC -400
        $DC 500
        $DC TRA
        $DC 0                   *
        DAC LVLTRA
        LAC =PENINT
        JMS PENSET
        LAW -100
        JMS* =N.C
        JMP PENEND

* CREATE NEW SYMBOL

CONCRE  $DC 0
        DAC CONDIC
        JMS* =X.S
        JMP PENEND
        JMS PENHIT
        LAC TRAPEN
        JMS PENSET
        JMS* =S.TL
        JMP PENEND
```

```
        DAC LVLFRG
        LAC PENY
        CMA
        TAD PENYL
        CMA
        LRSS 25
        LAC LVLFRG
        JMS* =S.LY
        LAC PENX
        CMA
        TAD PENXL
        CMA
        LRSS 25
        LAC LVLFRG
        JMS* =S.LX
        JMS* =S.TL
        JMP CON8
        DAC LVLSYM
        LMQ
        LAC LVLFRG
        JMS* =S.TI
        JMP CON7
CON1    LAC* CONDIC
        SPA
        JMP CON3
        JMS* =S.TL
        JMP CON8
        DAC LVLTMP
        LMQ
        LAC LVLSYM
        JMS* =S.TI
        JMP CON6
CON2    LAC* CONDIC
        SPA
        JMP CON3
        ALS 2
        LRSS 12
        DAC *+13
        LACQ
        LRSS 12
```

```
        DAC *+11
        LAC* CONDIC
        AND =600000
        SNA
        LAC =PAR-PARINT+200000
        TAD =PARINT-200000
        JMS LVL
        $DC LVLTMP
        $DC 0
        $DC 0
        $DC 500
        JMP CON8
        ISZ CONDIC
        JMP CON2
CON3    JMS* =S.TL
        JMP CON8
        DAC LVLTMP
        LMQ
        LAC LVLSYM
        JMS* =S.TI
        JMP CON6
CON4    LAC* CONDIC
        SMA
        JMP CON5
        ALS 2
        LRSS 12
        DAC *+14
        LACQ
        LRSS 12
        DAC *+12
        LAC* CONDIC
        AND =600000
        SAD =400000
        LAC =INP
        SAD =600000
        LAC =OUT
        JMS LVL
        $DC LVLTMP
        $DC 0
        $DC 0
```

```
        $DC 500
        JMP CON8
        ISZ CONDIC
        JMP CON4
CON5    LAC CONDIC
        TAD =1
        LMQ
        LAC LVLSYM
        JMS* =S.TI
        JMP CON8
        LAC LVLFRG
        LMQ
        LAC LVLDGM
        JMS* =S.TI
        JMP CON8
        LAC =PENMV
        JMS* =T.S
        LAC* CONDIC
        JMS NAMDEF
        LAC LVLSYM /PAR
        XCT* CONCRE /PAR
        $DC 0
        JMS* =T.F
CON6    LAC LVLTMP
        SKP
CON7    LAC LVLSYM
        JMS CHEW
CON8    LAC LVLFRG
        JMS CHEW
        JMP PENEND

CONTXT  $DC 3
        $TEXT "CONSTRUCT"

* CREATE ELEMENT WITH NO GENERATION PARAMETERS

CONNUL  $DC 0
        LAC S0100+1
        XOR =400
        DAC S0100+1
```

```
        LAC =S0100
        JMS QASCM
        JMP* CONNUL

* MOVE FRAGMENT LEVEL WITH TRACKING CROSS

PENMV   JMS TRAC
        LAC LVLFRG /PAR
        LAW -40
        JMS* =N.C
        JMP PENEND

* INTERPRET HIT ON ELEMENT OR CONNECTION

PENINT  JMS PENLVL
        LAC LVLEAF
        SAD =OUT
        JMP PENCON
        LAC LVLSYM
        JMS ATTR
        $DC 0
        TAD =-1
        DAC PENTMP
        LAC* PENTMP
        AND =200
        SNA+CLA
        JMP *+14
        JMS* =D.0
        $DC 0
        TAD =-1
        SAD PENTMP
        SKP
        JMP PENEND
        LAC* PENTMP
        AND =200000
        TAD =140
        DAC PENCI1+1
        JMP PENCIN
        JMS PENHIT
        LAC LVLFRG
```

```
        JMS  ATTR
        $DC  0
        CLA
        JMS  ATTR
        SKP
        JMP  PENMV
        LAW  -5
        DAC  PENINC
        DZM  PENIND
        LAC  LVLSYM
        JMS  COORDS
        $DC  0
        DAC  PENY
        LACQ
        DAC  PENX
PENTHR  JMS  PENREG
        $DC  160
        AND  =757
        SZA
        JMP  PENMV
        JMS  PENREG
        $DC  10
        AND  =202
        SNA
        JMP  *+7
        SAD  PENIND
        JMP  *+5
        DAC  PENIND
        ISZ  PENINC
        SKP
        JMP  PENDEL
        JMS* =X.S
        JMP  *+3
        JMP  PENEND
        JMP  PENTHR
        JMS* =T.P

* DELETE SYMBOL

PENDEL JMS* =X.T
```

```
        LAC LVLFRG
        LMQ
        LAC LVLDGM
        JMS* =S.TR
        $DC 0
        LAC LVLSYM
        JMS NAMGET
        $DC 0
        DAC PENTMP
        XOR =400
        LLS 11
        DAC S0101+1
        LAC =S0101
        JMS QASCM
        LAC =NAME
        TAD PENTMP
        DAC PENTMP
        DZM* PENTMP
        LAC LVLFRG
        JMS CHEW
        JMP PENEND

* TERMINATE CONNECTION LINE

PENLIN  JMS PENLVL
        LAC LVLEAF
        SAD =INP
        JMP PENINP
        LAC LVLSYM
        SAD PENSYM
        JMP PENEND
        JMS ATTR
        $DC 0
        TAD =-1
        DAC PENTMP
        LAC* PENTMP
        SPA
        JMP PENEND
        AND =200
        SNA
```

```
        JMP PENEND
        DAC LINTYP
        JMS* =T.F
PENINP  LAC LINV
        DAC PENTMP
        ISZ PENTMP
        LAC* PENTMP
        SMA
        JMP PENEND
        DAC LINTYP
        LAC =Q0001A
        LMQ
        LAC LVLOUT
        JMS* =S.TR
        NOP
        JMS* =X.T
        JMS* =T.F

* DRAW CONNECTION FROM OUTPUT PORT

PENCON  LAC =Q0001A
        LMQ
        LAC LVLOUT
        JMS* =S.TR
        NOP
        JMS PENHIT
        LAC LVLFRG
        DAC PENFRG
        JMS COORDS
        $DC 0
        DAC PENFY
        LACQ
        DAC PENFX
        LAC LVLSYM
        DAC PENSYM
        LAC LVLPRT
        DAC PENPRT
        LAC LVLOUT
        DAC PENOUT
        JMS COORDS
```

```
$DC 0
DAC PENY
LACQ
DAC PENX
JMS LINDRW
JMP PENEND
SKP
JMP PENIB
LAC LVLOUT
JMS COORDS
$DC 0
DAC PENYT
LACQ
DAC PENXT
JMS LINADJ
JMS* =S.TL
$DC 0                    *
DAC LVLFRG
LAC LINE
LMQ
LAC LVLSCR
JMS* =S.TR
$DC 0
LAC LINY
CMA
TAD PENFY
CMA
LRSS 3
DAC *+12
LAC LINX
CMA
TAD PENFX
CMA
LRSS 3
DAC *+5
LAC LVLFRG
JMS LVL
$DC PENFRG
$DC 0
$DC 0
```

```
$DC 0
$DC 0                    *
DAC PENFRG
LAW 10
LMQ
LAC LVLFRG
JMS* =S.LP
LAC PENOUT
LMQ
LAC LVLFRG
JMS* =S.TI
$DC 0                    *
LAC LVLOUT
LMQ
LAC LVLFRG
JMS* =S.TI
$DC 0                    *
LAC LINE
LMQ
LAC PENFRG
JMS* =S.TI
$DC 0                    *
LAC =OUT
LMQ
LAC PENOUT
JMS* =S.TR
$DC 0
LAC =OUTCON
LMQ
LAC PENOUT
JMS* =S.TI
$DC 0                    *
LAC =INP
LMQ
LAC LVLOUT
JMS* =S.TR
$DC 0
LAC =INPCON
LMQ
LAC LVLOUT
```

```
JMS* =S.TI
$DC 0                    *
LAC =201
DAC S0100+1
JMS NAMDEF
LAC PENFRG /PAR
JMS CONNUL /PAR
$DC 0
LAC PENFRG
JMS NAMGET
$DC 0
ALSS 11
DAC S0103+1
ISZ S0103+1
LAC PENSYM
JMS NAMGET
$DC 0
ALSS 11
DAC S0103+2
LAC PENOUT
LMQ
LAC PENPRT
JMS PENCNT
XOR S0103+2
XOR =400
DAC S0103+2
LAC =S0103
JMS QASCM
ISZ S0103+1
LAC LVLSYM
JMS NAMGET
$DC 0
ALSS 11
DAC S0103+2
LAC LVLOUT
LMQ
LAC LVLPRT
JMS PENCNT
XOR S0103+2
XOR =400
```

```
        DAC S0103+2
        LAC =S0103
        JMS QASCM
        LAW -40
        JMS* =N.C
        JMP PENEND

* DETERMINE WHETHER TO DELETE CONNECTION OR DRAW OUTPUT BRANCH

PENCIN JMS PENHIT
        LAC PENY
        LRSS 23
        LAC LVLSCR
        JMS* =S.LY
        LAC PENX
        LRSS 23
        LAC LVLSCR
        JMS* =S.LX
        LAC =RAND
        LMQ
        LAC LVLSCR
        JMS* =S.TI
        $DC 0                *
        LAW -5
        DAC PENINC
        DZM PENIND
        DZM PENTYP
PENCI1 JMS PENREG
        $DC 0
        AND =757
        SZA
        JMP PENOB
        JMS PENREG
        $DC 20
        AND =252
        LMQ
        LAC PENTYP
        OMQ
        DAC PENTYP
        SAD =252
```

```
       JMP PENOB
       JMS PENREG
       $DC 24
       AND =202
       SNA
       JMP *+7
       SAD PENIND
       JMP *+5
       DAC PENIND
       ISZ PENINC
       SKP
       JMP PENCDL
       JMS* =X.S
       JMP PENCI2
       LAC LVLSCR
       JMS CHEWA
       JMP PENEND
       JMP PENCI1
PENCI2 JMS* =T.P

* DELETE CONNECTION

PENCDL LAC LVLSCR
       JMS CHEWA
       JMS* =X.T
       CLA
       JMS* =D.P
       LAC LVLFRG
       TAD =4
       DAC PENTMP
       LAC* PENTMP
       JMS* =C.CB
       DAC PENY
       ISZ PENTMP
       LAC* PENTMP
       JMS* =C.CB
       DAC PENX
       LAC LVLSYM
       DAC PENSYM
       LMQ
```

```
        LAC LVLFRG
        JMS* =S.TR
        $DC 0
        LAC PENSYM
        JMS ATTR
        $DC 0
        CLA
        JMS ATTR
        $DC 0
        DAC PENPRT
PENCD1  LAC PENPRT
        JMS ATTR
        JMP PENCD2
        DAC PENOUT
        LMQ
        LAC PENPRT
        JMS* =S.TR
        $DC 0
        LAC PENOUT
        JMS ATTR
        $DC 0
        DAC PENTMP
        LMQ
        LAC PENOUT
        JMS* =S.TR
        $DC 0
        LAC PENTMP
        SAD =OUTCON
        LAC =OUT
        SAD =INPCON
        LAC =INP
        LMQ
        LAC PENOUT
        JMS* =S.TI
        $DC 0              *
        JMS PENCDS
        JMS PENLVL
        LAC LVLSYM
        JMS NAMGET
        $DC 0
```

```
        ALSS 11
        DAC S0104+1
        LAC LVLOUT
        LMQ
        LAC LVLPRT
        JMS PENCNT
        XOR S0104+1
        XOR =400
        DAC S0104+1
        LAC =S0104
        JMS QASCM
        JMP PENCD1
PENCD2  LAC PENSYM
        JMS NAMGET
        $DC 0
        DAC PENTMP
        XOR =400
        ALSS 11
        DAC S0101+1
        LAC =S0101
        JMS QASCM
        LAC =NAME
        TAD PENTMP
        DAC PENTMP
        DZM* PENTMP
        LAC PENSYM
        JMS CHEW
        LAC =PENCQ1
        JMS* =Q.C
        LAC =PENCQ2
        JMS* =Q.C
PENCD3  LAC LVLFRG
        JMS ATTR
        JMP PENCD8
        DAC PENSYM
        JMS* =S.TL
        $DC 0                      *
        DAC PENFRG
        LMQ
        LAC LVLDGM
```

```
        JMS* =S.TI
        $DC 0                   *
        LAC PENY
        LMQ
        LAC PENFRG
        JMS* =S.LY
        LAC PENX
        LMQ
        LAC PENFRG
        JMS* =S.LX
        LAC PENSYM
PENCD4  LMQ
        LAC LVLFRG
        JMS* =S.TR
        JMP PENCD7
        LAC PENSYM
        LMQ
        LAC PENFRG
        JMS* =S.TI
        $DC 0                   *
        LAC PENSYM
        JMS ATTR
        $DC 0
        CLA
        JMS ATTR
        $DC 0
        DAC PENPRT
PENCD5  LAC PENPRT
        JMS ATTR
        JMP PENCD6
        DAC PENOUT
        LMQ
        LAC PENPRT
        JMS* =S.TR
        $DC 0
        LAC PENOUT
        LMQ
        LAC =PENCQ1
        JMS* =Q.I
        $DC 0
```

```
       LAC PENOUT
       JMS ATTR
       $DC 0
       SAD =INP
       JMP PENCD5
       SAD =OUT
       JMP PENCD5
       JMS PENCDS
       LAW 2
       JMS* =D.0
       $DC 0
       LMQ
       LAC =PENCQ2
       JMS* =Q.I
       $DC 0
       JMP PENCD5
PENCD6 LAC =PENCQ1
       JMS* =Q.F
       JMP PENCD7
       LMQ
       LAC PENPRT
       JMS* =S.TI
       $DC 0                *
       JMP PENCD6
PENCD7 LAC =PENCQ2
       JMS* =Q.F
       JMP PENCD3
       DAC PENSYM
       JMP PENCD4
PENCD8 LAC LVLFRG
       LMQ
       LAC LVLDGM
       JMS* =S.TR
       $DC 0
       LAC LVLFRG
       JMS CHEW
       LAC TRAPEN
       JMS PENSET
       JMP PENEND
```

```
PENCDS $DC 0
       LAC =*+6
       LMQ
       LAC PENOUT
       JMS* =S.LU
       JMS* =D.E
       JMS* =T.F
       LAC PENOUT
       JMS* =S.LN
       JMP* PENCDS

PENCQ1 $DC *+34
       $DS 34
PENCQ2 $DC *+34
       $DS 34
```

$TITLE SELMA CONSTRUCTION FRAME (CONT.)

```
* DRAW OUTPUT BRANCH

PENOB   LAC LVLFRG
        DAC PENFRG
        JMS COORDS
        $DC 0
        DAC PENFY
        LACQ
        DAC PENFX
        LAC LVLSYM
        DAC PENSYM
        JMS ATTR
        $DC 0
        TAD =-1
        DAC PENTMP
        LAC* PENTMP
        SAD =201
        JMP *+6
        RAL
        SPA+RAR
        JMP PENCI1+16
        TAD =400
        JMP PENOB1
        LAC PENTYP
        SAD =252
        LAC =401202
        SAD =401202
        SKP
        LAC =401203
PENOB1  DAC PENTYP
        AND =377
        SAD =203
        JMP PENOB2
        LAC LVLSCR
        JMS CHEWA
        LAC =PRIO
        LMQ
        LAC LVLSCR
```

```
        JMS* =S.TI
        $DC 0                     *
PENOB2  LAC LVLPRT
        DAC PENPRT
        LAC LVLOUT
        DAC PENOUT
        JMS LINDRW
        SKP
        JMP *+4
        LAC LVLSCR
        JMS CHEWA
        JMP PENEND
        LAC LVLOUT
        JMS COORDS
        $DC 0
        DAC PENYT
        LACQ
        DAC PENXT
        JMS LINADJ
        LAC LVLFRG
        LMQ
        LAC PENFRG
        DAC LVLFRG
        LACQ
        DAC PENFRG
        LAC LVLSYM
        LMQ
        LAC PENSYM
        DAC LVLSYM
        LACQ
        DAC PENSYM
        LAC LVLPRT
        LMQ
        LAC PENPRT
        DAC LVLPRT
        LACQ
        DAC PENPRT
        LAC LVLOUT
        LMQ
        LAC PENOUT
```

```
        DAC LVLOUT
        LACQ
        DAC PENOUT
        JMP PENBCH

* TERMINATE MULTI-INPUT BRANCH

PENIB   LAC LVLSYM
        JMS ATTR
        $DC 0
        TAD =-1
        DAC PENTMP
        LAC* PENTMP
        SAD =201
        LAC LINTYP
        TAD =400
        DAC PENTYP
        LAC LINV
        DAC PENTMP
        ISZ PENTMP
        LAC* PENTMP
        SPA+CLC
        JMP *+6
        TAD LINV
        DAC LINV
        ISZ LINR
        LAC =1
        SKP
        LAW -3
        DAC PENTMP
        LAC PENX
        DAC PENXT
        CMA
        TAD LINX
        CMA
        JMS LINSTR
        LAC PENTMP
        TAD LINV
        DAC LINV
        CLC
```

```
        TAD LINR
        DAC LINR
        LAC PENY
        DAC PENYT
        CMA
        TAD LINY
        CMA
        JMS LINSTR
        LAC LVLFRG
        SAD PENFRG
        JMP PENBCH
        JMS COORDS
        $DC 0
        CMA
        TAD PENFY
        CMA
        LRSS 3
        DAC PENTMP
        LLS 25
        CMA
        TAD PENFX
        CMA
        LRSS 25
        LAC PENTMP
        JMS PENMOV
        LAC PENFRG /PAR
        LAC LVLFRG /PAR
        LAC LVLFRG
        LMQ
        LAC LVLDGM
        JMS* =S.TR
        $DC 0
        LAC LVLFRG
        JMS CHEW
        JMP PENBCH

* ADD BRANCH TO CONNECTION & INFORM QAS

PENBCH LAC LVLSYM
        JMS ATTR
```

```
        $DC 0
        TAD =-1
        DAC PENBC
        LAC* PENBC
        SAD =201
        JMP PENBC3
        LAC LVLSYM
        JMS NAMGET
        $DC 0
        ALSS 11
        XOR =1
        DAC S0105+1
        LAC PENTYP
        RAL
        AND =377000
        XOR =400000
        DAC S0105+2
        LAC =S0105
        JMS QASCM
PENBC1  CLA
        JMS* =D.P
        LAC =*+6
        LMQ
        LAC LVLSYM
        JMS* =S.LU
        JMS* =D.E
        JMS* =T.F
        LAC LVLSYM
        JMS* =S.LN
        LAC PENTYP
        SMA
        JMP *+5
        LAC LINY
        DAC PENYT
        LAC LINX
        DAC PENXT
        LAC LVLSYM
        JMS COORDS
        $DC 0
        DAC PENYB
```

```
CMA
TAD PENYT
CMA
LRSS 1
DAC PENYT
LLS 23
DAC PENXB
CMA
TAD PENXT
CMA
LRSS 1
DAC PENXT
CLC
TAD PENBC
DAC PENBCC
LAW -2
TAD LINE
DAC PENBCB
LAC PENTYP
AND =1
SNA+CLA
LAW -3
TAD =-4
DAC PENBCN
SAD =-7
LAC =PRIO-RAND-4
TAD =RAND+5
DAC PENTMP
LAC PENBCN
CMA
TAD =1
TAD* PENBCB
TAD* PENBCC
JMS CORGET
$DC 0                    *
DAC PENBCT
DAC PENBCP
LAC PENTYP
DAC* PENBCP
ISZ PENBCP
```

```
       LAC =200500
       DAC* PENBCP
       ISZ PENBCP
       LAC =201121
       DAC* PENBCP
       ISZ PENBCP
       LAC PENYT
       JMS* =C.BC
       XOR =2000
       DAC* PENBCP
       ISZ PENBCP
       LAC PENXT
       JMS* =C.BC
       XOR =6000
       DAC* PENBCP
       ISZ PENBCP
       LAC PENBCN
       JMS CORMOV
       LAC PENBCP /PAR
       LAC PENTMP /PAR
       LAC PENBCN
       CMA
       TAD =1
       TAD PENBCP
       DAC PENBCP
       LAC LINE
       TAD =1
       DAC PENTMP
       LAC PENTYP
       SMA
       JMP PENBC6
       LAW -4
PENBC2 TAD* PENBCB
       DAC PENBCN
       CMA
       TAD =1
       JMS CORMOV
       LAC PENBCP /PAR
       LAC PENTMP /PAR
       LAC PENBCN
```

```
TAD PENBCP
DAC PENBCP
TAD =-1
DAC PENTMP
LAC* PENTMP
AND =3777
DAC* PENTMP
LAC PENYT
JMS* =C.BC
DAC* PENBCP
ISZ PENBCP
LAC PENXT
JMS* =C.BC
XOR =4000
DAC* PENBCP
ISZ PENBCP
LAC PENBC
TAD =2
DAC PENTMP
LAW -4
TAD* PENBCC
CMA
JMS CORMOV
LAC PENBCP /PAR
LAC PENTMP /PAR
LAC PENBC
TAD =1
LMQ
LAC LVLSYM
JMS* =S.TR
$DC 0
LAC PENBCT
TAD =1
LMQ
LAC LVLSYM
JMS* =S.TI
$DC 0                    *
LAC LVLSCR
JMS CHEWA
LAC PENBC
```

```
JMS CORFRE
LAC PENOUT
JMS ATTR
$DC 0
DAC PENTMP
LMQ
LAC PENOUT
JMS* =S.TR
$DC 0
LAC PENTMP
SAD =INP
LAC =INPCON
SAD =OUT
LAC =OUTCON
LMQ
LAC PENOUT
JMS* =S.TI
$DC 0                    *
LAC LVLSYM
JMS ATTR
$DC 0
DAC PENTMP
CLA
JMS ATTR
$DC 0
DAC LVLPRT
CLA
JMS ATTR
$DC 0
DAC LVLTMP
LAC PENTYP
AND =1
SNA+CLA
LAC =PARINT-PAR
TAD =PAR
DAC PENTMP
LAC =*+6
LMQ
LAC PENOUT
JMS* =S.LU
```

```
        JMS* =D.E
        JMS* =T.F
        LAC PENOUT
        JMS* =S.LN
        LAC TRAPEN
        JMS PENSET
        LAC PENOUT
        LMQ
        LAC LVLPRT
        JMS* =S.TI
        $DC 0                   *
        LAC PENOUT
        JMS COORDS
        $DC 0
        TAD =20
        CMA
        TAD PENYB
        CMA
        LRSS 1
        DAC PENYT
        LLS 23
        DAC PENXT
        LAC PENTYP
        SPA+CLA
        LAW -200
        TAD =100
        TAD PENXT
        CMA
        TAD PENXB
        CMA
        LRSS 1
        DAC PENXT
        LAC PENTMP
        JMS LVL
        $DC LVLTMP
PENYT   $DC 0
PENXT   $DC 0
        $DC 500
        $DC 0                   *
        LAC LVLSYM
```

```
       JMS  NAMGET
       $DC  0
       ALSS  11
       DAC  S0103+1
       LAC  PENOUT
       LMQ
       LAC  LVLPRT
       JMS  PENCNT
       XOR  S0103+1
       DAC  S0103+1
       LAC  PENSYM
       JMS  NAMGET
       $DC  0
       ALSS  11
       DAC  S0103+2
       LAC  PENOUT
       LMQ
       LAC  PENPRT
       JMS  PENCNT
       XOR  S0103+2
       LAC  PENOUT
       LMQ
       LAC  PENPRT
       JMS  PENCNT
       XOR  S0103+2
       XOR  =400
       DAC  S0103+2
       LAC  =S0103
       JMS  QASCM
       LAW  -40
       JMS*  =N.C
       JMP  PENEND

PENBC3 CLA
       JMS*  =D.P
       LAC  =10
       DAC  PENYB
       LAC  =40
       DAC  PENXB
       LAC  LVLSYM
```

```
        JMS ATTR
        $DC 0
        DAC PENTMP
        CLA
        JMS ATTR
        $DC 0
        DAC LVLPRT
        CLA
        JMS ATTR
        JMP PENBC5
        DAC LVLTMP
        LAC PENTYP
        AND =1
        SNA+CLA
        LAC =PARINT-PAR
        TAD =PAR
        DAC PENTMP
        LAC PENTYP
        SMA
        JMP PENBC4
        CLC
        TAD PENBC
        DAC PENBCC
        LAW -3
        TAD* PENBCC
        TAD PENBCC
        DAC PENBCP
        LAC* PENBCP
        JMS* =C.CB
        CMA
        TAD PENYB
        DAC PENYB
        ISZ PENBCP
        LAC* PENBCP
        JMS* =C.CB
        CMA
        TAD =-100
        TAD PENXB
        DAC PENXB
PENBC4  LAC PENTMP
```

```
        JMS LVL
        $DC LVLTMP
PENYB   $DC 0
PENXB   $DC 0
        $DC 500
        $DC 0                    *
        LAC LVLPRT
        DAC PENBCP
        JMS ATTR
        $DC 0
        DAC LVLTMP
        CLA
        JMS ATTR
        $DC 0
        DAC PENTMP
        LMQ
        LAC PENBCP
        JMS* =S.TR
        $DC 0
        LAC LVLTMP
        LMQ
        LAC PENBCP
        JMS* =S.TR
        $DC 0
        LAC LVLSYM
        DAC PENBSV
        JMS NAMGET
        $DC 0
        ALSS 11
        DAC S0103+1
        XOR =400000
        DAC S0101+1
        LAC =*+6
        LMQ
        LAC PENTMP
        JMS* =S.LU
        JMS* =D.E
        JMS* =T.F
        LAC PENTMP
        JMS* =S.LN
```

```
JMS PENLVL
LAC LVLSYM
JMS NAMGET
$DC 0
ALSS 11
DAC S0104+1
LAC PENTMP
LMQ
LAC LVLPRT
JMS PENCNT
XOR =400
XOR S0104+1
DAC S0104+1
DAC S0103+2
LAC =S0104
JMS QASCM
LAC =*+6
LMQ
LAC LVLTMP
JMS* =S.LU
JMS* =D.E
JMS* =T.F
LAC LVLTMP
JMS* =S.LN
JMS PENLVL
LAC LVLSYM
JMS NAMGET
$DC 0
ALSS 11
DAC S0104+1
LAC LVLTMP
LMQ
LAC LVLPRT
JMS PENCNT
XOR =400
XOR S0104+1
DAC S0104+1
LAC =S0104
JMS QASCM
LAC =S0101
```

```
       JMS QASCM
       LAC PENTYP
       AND =377
       XOR S0101+1
       XOR =400000
       DAC S0100+1
       LAC =402000
       DAC S0100+2
       LAC =S0100
       JMS QASCM
       ISZ S0103+1
       LAC PENTYP
       SMA
       ISZ S0103+1
       LAC =S0103
       JMS QASCM
       LAC S0104+1
       DAC S0103+2
       LAC S0103+1
       XOR =3
       DAC S0103+1
       LAC =S0103
       JMS QASCM
       LAC PENBSV
       DAC LVLSYM
       LAC PENTMP
       LMQ
       LAC PENBCP
       JMS* =S.TI
       $DC 0                   *
       LAC LVLTMP
       LMQ
       LAC PENBCP
       JMS* =S.TI
       $DC 0                   *
       JMP PENBC1
PENBC5 LAC PENTMP
       LMQ
       LAC LVLSYM
       JMS* =S.TR
```

```
        $DC 0
        LAC LVLPRT
        LMQ
        LAC LVLSYM
        JMS* =S.TR
        $DC 0
        JMS* =S.TL
        $DC 0                   *
        DAC LVLTMP
        LMQ
        LAC LVLSYM
        JMS* =S.TI
        $DC 0                   *
        LAC LVLPRT
        LMQ
        LAC LVLSYM
        JMS* =S.TI
        $DC 0                   *
        LAC PENTMP
        LMQ
        LAC LVLSYM
        JMS* =S.TI
        $DC 0                   *
        JMP PENBC3+22
PENBC6  LAC =201121
        DAC* PENBCP
        ISZ PENBCP
        LAC* PENTMP
        ISZ PENTMP
        SMA
        JMP *-3
        DAC* PENBCP
        ISZ PENBCP
        LAC* PENTMP
        ISZ PENTMP
        AND =3777
        DAC* PENBCP
        ISZ PENBCP
        LAC LINE
        TAD =2
```

```
DAC PENTMP
LAW -7
JMP PENBC2
```

```
        $TITLE                    SELMA RESULTS FRAME

RES     JMS* =X.S
        JMP PENEND
        LAC LVLBUT
        JMS CHEWA
        LAC LVLDGM
        LMQ
        LAC LVLHAL
        JMS* =S.TR
        NOP
        LAC LVLDGM
        LMQ
        LAC LVLHAL
        JMS* =S.TI
        $DC 0                     *
        JMS BUTX
        $DC SELT2
        $DC LVLBUT
        $DC -400
        $DC -400
        $DC 500
        $DC TRA
        $DC 0                     *
        DAC LVLTRA
        LAC =PENID
        JMS PENSET
        JMS BUTX
        $DC CONTXT
        $DC LVLBUT
        $DC -400
        $DC -160
        $DC 500
        $DC CON
        $DC 0                     *
        JMS BUTX
        $DC PLOTXT
        $DC LVLBUT
        $DC -400
        $DC 50
```

```
        $DC 500
        $DC PLO
        $DC 0                       *
        JMS BUTX
        $DC RESTYP
        $DC LVLBUT
        $DC -400
        $DC 200
        $DC 500
        $DC REST
        $DC 0                       *
        DAC LVLTYP
        LAW -67
        DAC PENIDT+1
        DZM LVLRES
        LAW -100
        JMS* =N.C
        JMP PENEND

* IDENTIFY SYMBOL

ENID    JMS PENLVL
        LAC LVLSYM
        JMS NAMGET
        $DC 0
        DAC PENIDN
        AND =200
        SZA
        JMP PENEND
        LAC PENIDN
        TAD =-67
        DAC PENIDT+1
        LAC LVLRES
        LMQ
        LAC LVLDGM
        JMS* =S.TR
        JMP *+3
        LAC LVLRES
        JMS CHEW
        JMS PENHIT
```

```
        LAC LVLDGM
        JMS COORDS
        $DC 0
        CMA
        DAC PENIDY
        LACQ
        CMA
        DAC PENIDX
        LAC LVLSYM
        JMS COORDS
        $DC 0
        TAD =121
        TAD PENIDY
        LRSS 1
        DAC PENIDY
        LLS 23
        TAD =-77
        TAD PENIDX
        LRSS 1
        DAC PENIDX
        JMS BUTX
        $DC PENIDT
        $DC LVLDGM
PENIDY  $DC 0
PENIDX  $DC 0
        $DC 540
        $DC PENIDC
        $DC 0                   *
        DAC LVLRES
        LAC =RESBOX
        LMQ
        LAC LVLRES
        JMS* =S.TI
        $DC 0                   *
        JMS* =X.S
        JMS* =T.P
        JMP PENEND

* REMOVE ELEMENT LABEL
```

```
PENIDC LAC LVLRES
       LMQ
       LAC LVLDGM
       JMS* =S.TR
       $DC 0
       LAC LVLRES
       JMS CHEW
       LAW -67
       DAC PENIDT+1
       JMP PENEND

REST   LAC PENIDT+1
       TAD =67
       XOR =400
       LLS 11
       DAC S0303+1
       LAC LVLTYP
       LMQ
       LAC LVLBUT
       JMS* =S.TR
       $DC 0
       LAW -100
       JMS* =N.C
       LAC LVLTYP
       LMQ
       LAC LVLBUT
       JMS* =S.TI
       $DC 0                    *
       LAC =S0303
       JMS QASCM
       JMP PENEND

PENIDT $DC 1
       $DC 0

RESTXT $DC 3
       $TEXT "RESULTS"

RESTYP $DC 2
       $TEXT " TYPE"
```

```
RESBOX $DC 700
       VEC
       $DC 2002
       $DC 0
       $DC 6020
       $DC 0
       $DC 4000
       $DC 20
       $DC 4020
       $DC 0
       $DC 4000
       $DC 2020
       $DC 2
       $DC 4000
       POP
```

```
        $TITLE                  SELMA PLOT FRAME

PLO     JMS* =X.S
        JMP PENEND
        LAC =PBTR+2
        JMS* =P.T
        DZM LVLVX
        DZM LVLVY
        LAC KBDTBL
        DAC PLOKBD
        LAC =KBD3
        DAC KBDTBL
        LAC LVLBUT
        JMS CHEWA
        LAC LVLDGM
        LMQ
        LAC LVLHAL
        JMS* =S.TR
        NOP
        LAC LVLRES
        LMQ
        LAC LVLDGM
        JMS* =S.TR
        JMP *+3
        LAC LVLRES
        JMS CHEW
        DZM LVLRES
        LAC =PENGPH
        JMS* =D.P
        LAC =PLOAX
        LMQ
        LAC LVLBUT
        JMS* =S.TI
        $DC 0                   *
        JMS BUTX
        $DC PLODGM
        $DC LVLBUT
        $DC -400
        $DC -34
        $DC 500
```

```
        $DC  DGM
        $DC  0                  *
        DAC  LVLESC
        LAW  -67
        SAD  PENIDT+1
        JMP  PLOALL
        LAC  PENIDT+1
        XOR  =420100
        DAC  PLOLBE+6
        JMS  BUTX
        $DC  PLOLBE
        $DC  LVLBUT
        $DC  -320
        $DC  -110
        $DC  500
        $DC  PENEND
        $DC  0                  *
        JMP  *+11
PLOALL  JMS  BUTX
        $DC  PLOLBA
        $DC  LVLBUT
        $DC  -320
        $DC  -124
        $DC  500
        $DC  PENEND
        $DC  0                  *
        JMS  BUTX
        $DC  PLORT
        $DC  LVLBUT
        $DC  310
        $DC  320
        $DC  500
        $DC  PLORTT
        $DC  0                  *
        DAC  LVLPR
        JMS  BUTX
        $DC  PLOLT
        $DC  LVLBUT
        $DC  240
        $DC  320
```

```
       $DC 500
       $DC PLOLTT
       $DC 0                  *
       DAC LVLPL
       JMS BUTX
       $DC PLODT
       $DC LVLBUT
       $DC 170
       $DC 320
       $DC 500
       $DC PLODTT
       $DC 0                  *
       DAC LVLPD
       JMS PLOCLR
       LAW -300
       LMQ
       LAC LVLSCR
       JMS* =S.LX
       LAW -300
       LMQ
       LAC LVLSCR
       JMS* =S.LY
       LAC PENIDT+1
       TAD =467
       ALS 11
       DAC S0300+1
       LAC =S0300
       JMS QASCMR
       DZM PLOST
       LAC QASCMS
       SNA
       JMP PENEND
       SKP
       JMP *-4
       JMS* =T.P

DGM    LAC PLOKBD
       DAC KBDTBL
       LAC =PBT
       JMS* =P.T
```

```
        JMS PLOCLR
        JMP RES

* MODIFY PLOT

PLODTT  LAC LVLPD
        SKP
PLORTT  LAC LVLPR
        SKP
PLOLTT  LAC LVLPL
        DAC LVLPLO
        JMS* =S.LN
        LAW 10
        LMQ
        LAC LVLESC
        JMS* =S.LP
        LAW 10
        LMQ
        LAC LVLPD
        SAD LVLPLO
        SKP
        JMS* =S.LP
        LAW 10
        LMQ
        LAC LVLPR
        SAD LVLPLO
        SKP
        JMS* =S.LP
        LAW 10
        LMQ
        LAC LVLPL
        SAD LVLPLO
        SKP
        JMS* =S.LP
        LAC =PENSHF
        JMS* =D.P
        LAC =400000
        DAC S0301+2
        DZM PLOSW
        JMS* =D.E
```

```
       LAC PLOSW
       SZA
       JMP *+4
       SKP
       JMP *-4
       JMS* =T.P
       LAC LVLPLO
       SAD LVLPD
       JMP PLOD
       SAD LVLPR
       JMP *+20
       LAC PLOCNT
       TAD =-61
       TAD PLOST
       SPA
       CLA
       DAC PLOBEG
       CMA
       TAD PLOST
       TAD =2
       TAD PLOCNT
       ALSS 11
       XOR =400000
       DAC S0301+2
       LAC PLOBEG
       JMP *+3
       LAC PLOCNT
       TAD PLOST
       DAC PLOST
       JMS KBDCOD
       DAC S0301+1
PLOEND LAC LVLPLO
       SAD LVLPD
       LAC =PLODTT
       SAD LVLPR
       LAC =PLORTT
       SAD LVLPL
       LAC =PLOLTT
       LMQ
       LAC LVLPLO
```

```
        JMS* =S.LL
        LAW 500
        LMQ
        LAC LVLPD
        JMS* =S.LP
        LAW 500
        LMQ
        LAC LVLPR
        JMS* =S.LP
        LAW 500
        LMQ
        LAC LVLPL
        JMS* =S.LP
        LAW 500
        LMQ
        LAC LVLESC
        JMS* =S.LP
        LAC =PENGPH
        JMS* =D.P
        LAC =S0301
        JMS QASCMR
        LAC QASCMS
        SNA
        JMP PENEND
        SKP
        JMP *-4
        JMS* =T.P
PLOD    JMS PENHIT
        JMS* =X.S
        JMS* =T.P
        LAC PLOCNT
        DAC PLOSAV
        DZM PLOSW
        JMS* =D.E
        LAC PLOSW
        SZA
        JMP *+4
        SKP
        JMP *-4
        JMS* =T.P
```

```
        LAC PLOCNT
        CMA
        TAD PLOSAV
        CMA
        ALS 11
        XOR =400000
        SPA
        JMP *+6
        CMA
        TAD =2000
        DAC S0301+2
        LAC PLOCNT
        JMP *+4
        TAD =1000
        DAC S0301+2
        LAC PLOSAV
        TAD PLOST
        DAC PLOST
        JMS KBDCOD
        DAC S0301+1
        JMP PLOEND

* DISPLAY VALUE

PENGPH JMS* =D.A
        TAD =-PLOGPH-3
        AND =777770
        DAC PENGP1
        LRSS 3
        TAD PLOST
        JMS KBDCOD
        XOR =400
        DAC S0302+1
        LAC =S0302
        JMS QASCM
        LAC PENGP1
        TAD =PLOGPH+4
        DAC Q01021
        LAC* Q01021
        AND =1777
```

```
        TAD =-300
        DAC Q01021
        JMS* =T.F

* SHIFT GRAPH

PENSHF  JMS* =D.A
        TAD =-PLOGPH-3
        LRSS 3
        DAC PLOCNT
        CLC
        DAC PLOSW
        JMS* =T.F
```

```
* CLEAR GRAPH
* CALLING SEQUENCE:
*       JMS PLOCLR
*       ----                      (RETURN)

PLOCLR $DC 0
       LAC LVLSCR
       JMS CHEWA
       LAW -1000
       DAC PLOC1
       LAC =PLOGPH+2
       DAC PLOC2
       DZM* PLOC2
       ISZ PLOC2
       ISZ PLOC1
       JMP *-3
       LAC =PLOGPH
       LMQ
       LAC LVLSCR
       JMS* =S.TI
       $DC 0                      *
       LAC =PLOGPH+3
       DAC PLOPTR
       JMP* PLOCLR

PLOGPH $DC 560
       VEC
       $DS 1000
       $DC 0
       $DC 4000
       POP
```

```
* DISPLAY LABEL FROM DATAPHONE BUFFER
* CALLING SEQUENCE:
*       JMS PLOLBL
*       $DC ----                    (Y COORDINATE)
*       $DC ----                    (X COORDINATE)
*       ----                        (RETURN)

PLOLBL $DC 0
       LAC* PLOLBL
       ISZ PLOLBL
       TAD =300
       DAC PLOLBY
       JMS QASR
       DAC PLOLBB
       CMA
       TAD =1
       DAC PLOLBC
       CLL
       ALS 2
       TAD* PLOLBL
       ISZ PLOLBL
       TAD =300
       DAC PLOLBX
       LAC =PLOLBB+1
       DAC PLOLBP
       JMS QASR
       XOR =777700
       DAC* PLOLBP
       ISZ PLOLBP
       ISZ PLOLBC
       JMP *-5
       JMS BUTX
       $DC PLOLBB
       $DC LVLSCR
PLOLBY $DC 0
PLOLBX $DC 0
       $DC 500
       $DC PENEND
       $DC 0                        *
       JMP* PLOLBL
```

PLOLBB $DS 20

```
PLOTXT $DC 2
       $TEXT "PLOT"

PLODGM $DC 3
       $TEXT "DIAGRAM"

PLOLBE $DC 6
       $TEXT "STATE OF ELEMEN"
       $DC 0

PLOLBA $DC 7
       $TEXT "STATE OF ENTIRE MODEL"

PLORT  $DC 2
       $TEXT "RIGHT"

PLOLT  $DC 2
       $TEXT "LEFT"

PLODT  $DC 2
       $TEXT "DETAIL"

PLOTT  $DC 2
       $TEXT "TYPE"
```

```
PLOAX   $DC 500
        VEC
        $DC 300
        $DC 2300
        $DC 6600
        $DC 0
        $DC 4000
        $DC 620
        $DC 300
        $DC 6320
        POP
```

```
        $TITLE                 SELMA GLOBAL DISPLAY TASKS

* TERMINATE DISPLAY SERVICE

PENEND LAC QASCM+2
       SZA
       JMP *+4
       JMS* =D.E
       JMS* =T.F
       JMP PENEND
       JMS* =T.P

* PARAMETER DISPLAY TASK

PENPAR LAC KBDPB
       SNA
       JMP PENEND
       SAD =1
       SKP
       JMP *+4
       LAC KBDPB+1
       SAD =777761
       JMP PENEND
       JMS PENLVL
       LAC LVLEAF
       SNA
       JMP PENEND
       SAD =PAR
       JMP *+10
       SAD =PARINT
       JMP *+6
       LAC* LVLEAF
       AND =7777
       SAD =2010
       SKP
       JMP PENEND
       LAC* LVLEAF
       XOR KBDPM
       SPA
       JMP PENEND
```

```
LAC LVLSYM
JMS NAMGET
$DC 0
ALSS 11
DAC S0102+1
LAC LVLOUT
LMQ
LAC LVLPRT
JMS PENCNT
XOR S0102+1
DAC S0102+1
LAC LVLEAF
LMQ
LAC LVLOUT
JMS* =S.TR
$DC 0
LAC KBDPB
TAD =-4
SMA
CLA
TAD =4
DAC KBDPB
LAC* KBDPP
XOR =400
DAC* KBDPP
LAC =KBDPB
DAC PENPA3
LAW -4
DAC PENPA1
LAC =S0102+2
DAC PENPA2
ISZ PENPA3
LAC* PENPA3
ALSS 11
ISZ PENPA3
XOR* PENPA3
XOR =77077
DAC* PENPA2
ISZ PENPA2
ISZ PENPA1
```

```
JMP *-11
LAC =S0102
JMS QASCM
LAC* KBDPP
XOR =400
DAC* KBDPP
LAC* LVLEAF
LRS 14
AND =7
CMA
DAC PENPA1
LAC LVLOUT
TAD =4
DAC PENPA2
LAC LVLEAF
JMS CHEWA
LAC LVLEAF
SAD =PAR
JMP *+4
SAD =PARINT
SKP
JMP *+7
LAC* PENPA2
JMS* =C.CB
TAD =-4
LMQ
LAC LVLOUT
JMS* =S.LY
LAC PENPA1
TAD KBDPB
CMA+CLL
ALS 2
DAC PENPA3
ISZ PENPA2
LAC* PENPA2
JMS* =C.CB
TAD PENPA3
LMQ
LAC LVLOUT
JMS* =S.LX
```

```
       LAC =KBDPB
       JMS* =L.D
       JMP PENREJ
       DAC LVLEAF
       LMQ
       LAC LVLOUT
       JMS* =S.TI
       JMP PENCHW
       LAC =Q0001A
       LMQ
       LAC LVLOUT
       JMS* =S.TR
       NOP
       LAC* LVLEAF
       AND =7777
       DAC* LVLEAF
       LAC KBDPM
       AND =40
       XOR KBDPB
       ALSS 14
       XOR* LVLEAF
       DAC* LVLEAF
       JMP PENREJ+10
PENREJ LAC KBDPM
       SMA+CLA
       LAC =PARINT-PAR
       TAD =PAR
       LMQ
       LAC LVLOUT
       JMS* =S.TI
       $DC 0
       LAC KBDPLP
       JMS PENSET
       LAC KBDPSV
       DAC KBDTBL
       DZM KBDPB
       LAC =KBDPB
       DAC KBDPP
       LAC LVLSCR
       JMS CHEWA
```

```
        LAC KBDVSW
        SNA
        JMP *+5
        LAW 17475
        JMS* =B.T
        LAW 10
        JMS* =T.R
        LAW -40
        JMS* =N.C
        JMP PENEND
PENCHW  LAC LVLEAF
        JMS CHEW
        JMP PENREJ

* TRANSLATION TASKS

TRA     JMS* =X.S
        JMP PENEND
        LAC TRAS
        SZA
        JMP *+11
        LAW -360
        DAC TRAS
        LMQ
        LAC LVLTRA
        JMS* =S.LY
        LAC =PENTRA
        JMS* =D.P
        JMP *+3
        LAC TRAPEN
        JMS PENSET
        LAW -40
        JMS* =N.C
        JMP PENEND

PENTRA  JMS PENHIT
        JMS TRAC
        LAC LVLDGM /PAR
        LAC LVLDGM
```

```
JMS COORDS
$DC 0
DAC PENYL
LACQ
DAC PENXL
JMP PENEND
```

```
        $TITLE                       SELMA GRAPHICAL TABLES
* SYMBOL DICTIONARY

DIC1    $DC DIC2
        $DC 2
        $TEXT "QUEUE"
        LAC =*+3
        JMS CONCRE
        JMS CONNUL   /PAR
        $DC 200000                   INTEGER PARAMETER AT (0,0) #1
        $DC 400354                   INPUT PORT AT (-24,0) #1
        $DC 600034                   OUTPUT PORT AT (34,0) #2
        $DC 1                        QUEUE
        $DC 500
        VEC
        $DC 0
        $DC 4024
        INCR
        $DC 7252
        $DC 5372
        $DC 5352
        $DC 7373
        $DC 6354
        $DC 5374
        $DC 5374
        $DC 7464
        $DC 5574
        $DC 5554
        $DC 7575
        $DC 6556
        $DC 5576
        $DC 5576
        $DC 5676
        $DC 5657
        $DC 7657
        $DC 5677
        $DC 7767
        $DC 5057
        $DC 7057
```

```
        $DC 7070
        $DC 6051
        $DC 7051
        $DC 5071
        $DC 7161
        $DC 5251
        $DC 7251
        $DC 7252
        $DC 1400
        VEC
        $DC 0
        $DC 6024
        POP

DIC2    $DC DIC3
        $DC 2
        $TEXT "SERVER"
        LAC =*+3
        JMS CONCRE
        JMS CONNUL  /PAR
        $DC 0                REAL PARAMETER AT (0,0) #1
        $DC 400354           INPUT PORT AT (-24,0) #1
        $DC 600034           OUTPUT PORT AT (34,0) #2
        $DC 2                SERVER
        $DC 500
        VEC
        $DC 24
        $DC 24
        $DC 4000
        $DC 2050
        $DC 6050
        $DC 0
        $DC 4000
        $DC 50
        $DC 4050
        $DC 0
        $DC 2024
        $DC 6024
        POP
```

```
DIC3    $DC DIC4
        $DC 2
        $TEXT "SOURCE"
        LAC =*+3
        JMS CONCRE
        JMS CONNUL  /PAR
        $DC 600010          OUTPUT PORT AT (10,0) #1
        $DC 3               SOURCE
        $DC 500
        SVEC
        $DC 1400
        $DC 7400
        $DC 7300
        $DC 1340
        POP

DIC4    $DC 0
        $DC 2
        $TEXT "EXIT"
        LAC =*+3
        JMS CONCRE
        JMS CONNUL  /PAR
        $DC 400000          INPUT PORT AT (0,0) #1
        $DC 3               EXIT
        $DC 500
        POP
```

```
* GRAPHICAL PORTS

* OUTPUT PORT (UNCONNECTED)
OUT     SVEC
        $DC 30
        $DC 4050
        POP
* OUTPUT PORT (CONNECTED)
OUTCON SVEC
        $DC 30
        $DC 4050
        POP
* INPUT PORT  (UNCONNECTED)
INP     SVEC
        $DC 30
        $DC 4050
        POP
* INPUT PORT  (CONNECTED)
INPCON SVEC
        $DC 4727
        $DC 3721
        $DC 4707
        $DC 0141
        POP
* UNASSIGNED REAL PARAMETER
PAR     $DC 401111
        $DC 5255
        $DC 5751
        $DC 400
        POP
* UNASSIGNED INTEGER PARAMETER
PARINT INCR
        $DC 5255
        $DC 5751
        $DC 400
        POP
* RANDOM BRANCH NODE
RAND    $DC 200500
        $DC 201111
        $DC 5164
```

```
        $DC 6660
        $DC 5204
        $DC 203000
* PRIORITY BRANCH NODE
PRIO    $DC 200500
        $DC 201141
        $DC 10
        $DC 5030
        $DC 7030
        $DC 7010
        $DC 5010
        $DC 70
        $DC 203000
```

```
      $TITLE                SELMA LINE DRAWING SUBROUTINES

* DRAW CONNECTION LINE
* CALLING SEQUENCE:
*      JMS LINDRW
*      ----                 (RETURN IF LINE NOT COMPLETED)
*      ----                 (RETURN IF LINE ENDED ON PORT)
*      ----                 (RETURN IF LINE ENDED ON CONNECTION)
* LINE STARTS AT COORDINATES IN PENX AND PENY

LINDRW $DC 0
       LAC PENY
       DAC LINY
       LRSS 23
       LAC LVLSCR
       JMS* =S.LY
       LAC PENX
       DAC LINX
       LRSS 23
       LAC LVLSCR
       JMS* =S.LX
       LAC =12
       JMS CORGET
       JMP* LINDRW
       DAC LINE
       ISZ LINE
       DAC LINTMP
       LAC =201
       DAC* LINTMP
       ISZ LINTMP
       LAC =200500
       DAC* LINTMP
       ISZ LINTMP
       LAC =201121
       DAC* LINTMP
       ISZ LINTMP
       LAC =4000
       DAC* LINTMP
       ISZ LINTMP
       DZM* LINTMP
```

```
        ISZ LINTMP
        LAC LINTMP
        DAC LINVEC
        LAC =4000
        DAC* LINTMP
        ISZ LINTMP
        DZM* LINTMP
        ISZ LINTMP
        LAC LINTMP
        DAC LINRET
        LAC =400000
        DAC* LINTMP
        ISZ LINTMP
        LAC =4000
        DAC* LINTMP
        ISZ LINTMP
        LAC =203000
        DAC* LINTMP
        LAC LINE
        LMQ
        LAC LVLSCR
        JMS* =S.TI
        JMP LINDEL
        DZM LINTYP
        LAC =PENLIN
        JMS* =D.P
        JMS* =D.E
        JMP LINXL
LINDEL  LAC LINE
        JMS CHEW
        JMP *+3
LINCHW  LAC LVLSCR
        JMS CHEWA
        JMS* =D.D
        LAC TRAPEN
        JMS PENSET
        JMP* LINDRW
LINXL   LAC LINVEC
        DAC LINV
        ISZ LINV
```

```
        LAC  LINRET
        DAC  LINR
        ISZ  LINR
LINXM   SKP
        SKP
        JMS*  =T.P
        JMS*  =X.X
        CMA
        TAD  LINX
        CMA
        JMS  LINSTR
        JMS  PENREG
        $DC  100
        AND  =707
        SZA
        JMP  LINXA
        JMS  PENREG
        $DC  40
        AND  =555
        SNA
        JMP  LINXS
LINONE  LAC  LINTYP
        SAD  =200
        JMS  LINCON
        JMS*  =X.S
        JMP  LINXM
        LAC  LINTYP
        SNA
        JMP  LINCHW
        LAC  =PENINT
        JMS  PENSET
        ISZ  LINDRW
        JMP*  LINDRW
LINXA   LAW  -5
        TAD  LINV
        SAD  LINE
        SKP
        JMP  *+13
        JMS  PENREG
        $DC  100
```

```
        AND =555
        SZA
        JMP *+6
        LAC PENX
        CMA
        TAD LINX
        CMA
        JMS LINSTR
        LAC* LINV
        JMS* =C.CB
        RCL
        TAD PENX
        DAC PENX
        LAC LINE
        LMQ
        LAC LVLSCR
        JMS* =S.TR
        $DC 0
        LAC LINE
        JMS LINEXT
        DAC LINE /PAR
        DAC LINVEC /PAR
        DAC LINRET /PAR
        JMP LINDEL
        LAC LINE
        LMQ
        LAC LVLSCR
        JMS* =S.TI
        JMP LINDEL
        JMP LINYL
LINXS   LAW -5
        TAD LINV
        SAD LINE
        JMP LINONE
        LAC PENX
        CMA
        TAD LINX
        CMA
        JMS LINSTR
        LAC LINE
```

```
        LMQ
        LAC LVLSCR
        JMS* =S.TR
        $DC 0
        LAC LINE
        JMS LINTRM
        DAC LINE /PAR
        DAC LINVEC /PAR
        DAC LINRET /PAR
        JMP LINDEL
        LAC LINE
        LMQ
        LAC LVLSCR
        JMS* =S.TI
        JMP LINDEL
        LAC* LINVEC
        XOR =2000
        JMS* =C.CB
        RCL
        TAD PENY
        DAC PENY
LINYL   LAC LINVEC
        DAC LINV
        LAC LINRET
        DAC LINR
LINYM   SKP
        SKP
        JMS* =T.P
        JMS* =X.Y
        CMA
        TAD LINY
        CMA
        JMS LINSTR
        JMS PENREG
        $DC 100
        AND =555
        SZA
        JMP LINYA
        JMS PENREG
        $DC 40
```

```
        AND =707
        SNA
        JMP LINYS
        LAC LINTYP
        SAD =200
        JMS LINCON
        JMS* =X.S
        JMP LINYM
        JMP LINCHW
LINYA   LAC* LINV
        JMS* =C.CB
        RCL
        TAD PENY
        DAC PENY
        LAC LINE
        LMQ
        LAC LVLSCR
        JMS* =S.TR
        $DC 0
        LAC LINE
        JMS LINEXT
        DAC LINE /PAR
        DAC LINVEC /PAR
        DAC LINRET /PAR
        JMP LINDEL
        LAC LINE
        LMQ
        LAC LVLSCR
        JMS* =S.TI
        JMP LINDEL
        JMP LINXL
LINYS   LAC PENY
        CMA
        TAD LINY
        CMA
        JMS LINSTR
        LAC LINE
        LMQ
        LAC LVLSCR
        JMS* =S.TR
```

```
        $DC 0
        LAC LINE
        JMS LINTRM
        DAC LINE /PAR
        DAC LINVEC /PAR
        DAC LINRET /PAR
        JMP LINDEL
        LAC LINE
        LMQ
        LAC LVLSCR
        JMS* =S.TI
        JMP LINDEL
        LAC LINVEC
        DAC LINV
        ISZ LINV
        LAC* LINV
        XOR =2000
        JMS* =C.CB
        RCL
        TAD PENX
        DAC PENX
        JMP LINXL+3
LINCON  $DC 0
        LAC PENX
        DAC PENXT
        LAC PENY
        DAC PENYT
        JMS PENHIT
        JMS* =X.S
        JMP LINREG-1
        LAC =200603
LINDUN  DAC LINTYP
        LAC TRAPEN
        JMS PENSET
        JMS* =X.T
        ISZ LINDRW
        ISZ LINDRW
        JMP* LINDRW
        DZM LINTYP
LINREG  JMS PENREG
```

```
$DC 140
AND =20
SZA
JMP *+11
DZM LINTYP
LAC PENXT
DAC PENX
LAC PENYT
DAC PENY
ISZ LINCON
JMS* =D.E
JMP* LINCON
JMS PENREG
$DC 20
AND =252
LMQ
LAC LINTYP
OMQ
DAC LINTYP
SAD =252
JMP *+5
JMS* =X.S
JMP *+6
JMS* =D.D
JMP LINDUN-1
LAC =200602
JMP LINDUN
JMP LINREG
JMS* =T.P
```

```
* STORE COORDINATE IN LINE BLOCK
* CALLING SEQUENCE:
*       JMS LINSTR
*       ----                    (RETURN)
* AC CONTENT ON ENTRY:
*    DESIRED COORDINATE AT END OF LINE RELATIVE TO BEGINNING

LINSTR  $DC 0
        AND =777770
        LRSS 1
        DAC LIN1
        JMS* =C.BC
        XOR =2000
        DAC LIN2
        LAC* LINR
        JMS* =C.CB
        DAC LIN3
        LAC* LINV
        JMS* =C.CB
        TAD LIN3
        TAD LIN1
        JMS* =C.BC
        XOR* LINV
        AND =3777
        XOR* LINV
        DAC* LINV
        LAC* LINR
        AND =774000
        XOR LIN2
        DAC* LINR
        JMP* LINSTR
```

```
* EXTEND NON-ACTIVE LINE BLOCK
* CALLING SEQUENCE:
*       JMS LINEXT
*       DAC ----                (DAC POINTER TO NEW BLOCK)
*       DAC ----                (DAC POINTER TO LAST VECTOR)
*       DAC ----                (DAC POINTER TO RETURN VECTOR)
*       ----                    (RETURN IF NOT ENOUGH STORAGE)
*       ----                    (RETURN)
* AC CONTENT ON ENTRY:
*   POINTER TO BLOCK TO BE EXTENDED (SECOND LOCATION)

LINEXT $DC 0
       TAD =-1
       DAC LIN1
       DAC LIN2
       TAD =-1
       DAC LIN3
       LAC* LIN3
       TAD =1
       JMS CORGET
       JMP LINE1
       DAC LIN3
       TAD =1
       XCT* LINEXT
       ISZ LINEXT
       LAC* LIN2
       SPA
       JMP *+5
       DAC* LIN3
       ISZ LIN2
       ISZ LIN3
       JMP *-6
       LAC LIN3
       XCT* LINEXT
       ISZ LINEXT
       LAC =4000
       DAC* LIN3
       ISZ LIN3
       DZM* LIN3
       ISZ LIN3
```

```
        LAC LIN3
        XCT* LINEXT
        LAC* LIN2
        ISZ LIN2
        DAC* LIN3
        ISZ LIN3
        LAC* LIN2
        DAC* LIN3
        ISZ LIN3
        LAC =203000
        DAC* LIN3
        LAC LIN1
        JMS CORFRE
        SKP
LINE1   ISZ LINEXT
        ISZ LINEXT
        ISZ LINEXT
        JMP* LINEXT
```

```
* SHORTEN NON-ACTIVE LINE BLOCK
* CALLING SEQUENCE:
*       JMS LINTRM
*       DAC ----              (DAC  POINTER TO NEW BLOCK)
*       DAC ----              (DAC POINTER TO LAST VECTOR)
*       DAC ----              (DAC POINTER TO RETURN VECTOR)
*       ----                  (RETURN IF NOT ENOUGH STORAGE)
*       ----                  (RETURN)
* AC CONTENT ON ENTRY:
*   POINTER TO LINE BLOCK (SECOND LOCATION)

LINTRM $DC 0
       TAD =-1
       DAC LIN1
       DAC LIN2
       TAD =-1
       DAC LIN3
       LAW -3
       TAD* LIN3
       JMS CORGET
       JMP LINT1
       DAC LIN3
       TAD =1
       XCT* LINTRM
       ISZ LINTRM
       LAC* LIN2
       SPA
       JMP *+5
       DAC* LIN3
       ISZ LIN2
       ISZ LIN3
       JMP *-6
       LAW -4
       TAD LIN3
       XCT* LINTRM
       ISZ LINTRM
       TAD =2
       DAC LIN3
       XCT* LINTRM
       LAC* LIN2
```

```
        ISZ LIN2
        DAC* LIN3
        ISZ LIN3
        LAC* LIN2
        DAC* LIN3
        ISZ LIN3
        LAC =203000
        DAC* LIN3
        LAC LIN1
        JMS CORFRE
        SKP
LINT1   ISZ LINTRM
        ISZ LINTRM
        ISZ LINTRM
        JMP* LINTRM
```

```
* ADJUST END OF LINE
* CALLING SEQUENCE:
*       JMS LINADJ
*       ----                    (RETURN)
* LINV & LINR SET TO X VECTOR ON ENTRY

LINADJ $DC 0
       LAC PENXT
       CMA
       TAD LINX
       CMA
       JMS LINSTR
       LAC LVLFRG
       SAD PENFRG
       JMP LINA1
       JMS COORDS
       $DC 0
       TAD PENY
       CMA
       TAD PENFY
       TAD PENYT
       CMA
       LRSS 3
       DAC PENTMP
       LLS 25
       CMA
       TAD PENFX
       CMA
       LRSS 25
       LAC PENTMP
       JMS PENMOV
       LAC PENFRG /PAR
       LAC LVLFRG /PAR
       LAC LVLFRG
       LMQ
       LAC LVLDGM
       JMS* =S.TR
       $DC 0
       LAC LVLFRG
       JMS CHEW
```

```
        JMP* LINADJ
LINAI   LAW -3
        TAD LINV
        DAC LINV
        CLC
        TAD LINR
        DAC LINR
        LAC PENYT
        CMA
        TAD LINY
        CMA
        JMS LINSTR
        JMP* LINADJ
```

```
      $TITLE               SELMA DISPLAY SUPPORT SUBROUTINES
* GENERATE INTERMEDIATE LEVEL
* CALLING SEQUENCE:
*       JMS LVL
*       $DC ----             (LOC CONTAINING POINTER TO OWNER)
*       $DC ----             (Y COORDINATE)
*       $DC ----             (X COORDINATE)
*       $DC ----             (DISPLAY PARAMETER)
*       ----                 (RETURN IF DISPLAY STORAGE EXCEEDED)
*       ----                 (RETURN)
* AC CONTENTS ON ENTRY:
*    POINTER TO ATTRIBUTE
* AC CONTENTS ON RETURN:
*    POINTER TO CREATED LEVEL

LVL     $DC 0
        JMS* =T.1
        $DC 0
        DAC LVL4
        JMS* =S.TL
        JMP LVL3
        DAC LVL5
        LAC LVL4
        LMQ
        LAC LVL5
        JMS* =S.TI
        JMP LVL2
        LAC* LVL+2
        DAC LVL4
        ISZ LVL+2
        LAC* LVL+2
        LMQ
        LAC LVL5
        JMS* =S.LY
        ISZ LVL+2
        LAC* LVL+2
        LMQ
        LAC LVL5
        JMS* =S.LX
```

```
        ISZ LVL+2
        LAC* LVL+2
        LMQ
        LAC LVL5
        JMS* =S.LP
        LAC LVL5
        LMQ
        LAC* LVL4
        JMS* =S.TI
        JMP LVL1
        LAC LVL5
        JMP LVL3+2
LVL1    LAC LVL5
        JMS ATTR
        $DC 0
        LMQ
        LAC LVL5
        JMS* =S.TR
        $DC 0
        LAC LVL5
        JMS* =S.TD
        $DC 0
        JMP LVL3+3
LVL2    LAC LVL5
        JMS* =S.TD
        $DC 0
LVL3    ISZ LVL+2
        ISZ LVL+2
        ISZ LVL+2
        ISZ LVL+2
        JMS* =T.U
        $DC LVL
```

```
* FORM LIGHT BUTTON FROM PREDEFINED STRUCTURE
* CALLING SEQUENCE:
*       JMS BUTN
*       $DC ----             (LOC CONTAINING POINTER TO OWNER)
*       $DC ----             (Y COORDINATE)
*       $DC ----             (X COORDINATE)
*       $DC ----             (DISPLAY PARAMETER)
*       $DC ----             (SERVICE TASK ADDRESS)
*       ----                 (RETURN IF DISPLAY STORAGE EXCEEDED)
*       ----                 (RETURN IF SUCCESSFUL)
* AC CONTENTS ON ENTRY:
*   POINTER TO STRUCTURE FOR BUTTON DISPLAY
* AC CONTENTS ON RETURN:
*   POINTER TO LIGHT BUTTON LEVEL

BUTN    $DC 0
        JMS* =T.L
        $DC 0
        DAC BUTN3
        LAW -4
        DAC BUTN4
        LAC =BUTN1
        DAC BUTN5
        LAC* BUTN+2
        DAC* BUTN5
        ISZ BUTN+2
        ISZ BUTN5
        ISZ BUTN4
        JMP *-5
        LAC BUTN3
        JMS LVL
BUTN1   $DC 0
        $DC 0
        $DC 0
        $DC 0
        JMP BUTN2
        DAC BUTN3
        LAC* BUTN+2
        LMQ
        LAC BUTN3
```

```
        JMS* =S.LL
        LAC BUTN3
        ISZ BUTN+2
BUTN2   ISZ BUTN+2
        JMS* =T.U
        $DC BUTN
```

```
* CREATE TEXT LIGHT BUTTON
* CALLING SEQUENCE:
*       JMS BUTX
*       $DC ----            (ADDRESS OF TEXT LIST)
*       $DC ----            (LOC CONTAINING POINTER TO OWNER)
*       $DC ----            (Y COORDINATE)
*       $DC ----            (X COORDINATE)
*       $DC ----            (DISPLAY PARAMETER)
*       $DC ----            (SERVICE TASK ADDRESS)
*       ----                (RETURN IF DISPLAY STORAGE EXCEEDED)
*       ----                (RETURN IF SUCCESSFUL)
* AC CONTENTS ON RETURN:
*   POINTER TO LIGHT BUTTON LEVEL

BUTX    $DC 0
        JMS* =T.L
        $DC 0
        LAC* BUTX+2
        JMS* =L.D
        JMP BUTX4
        DAC BUTX7
        LAW -6
        DAC BUTX5
        LAC =BUTX2
        DAC BUTX6
BUTX1   ISZ BUTX+2
        ISZ BUTX6
        ISZ BUTX5
        SKP
        JMP BUTX2-1
        LAC* BUTX+2
        DAC* BUTX6
        JMP BUTX1
        LAC BUTX7
BUTX2   JMS BUTN
        $DC 0
        $DC 0
        $DC 0
        $DC 0
        $DC 0
```

```
        JMP BUTX3+2
        ISZ BUTX+2
BUTX3   JMS* =T.U
        $DC BUTX
        LAC BUTX7
        JMS* =S.LL
        JMP BUTX3
BUTX4   LAC BUTX+2
        TAD =6
        DAC BUTX+2
        JMP BUTX3
```

```
* DESTROY ALL LEVELS, NODES, AND TEXT LEAVES IN A DISPLAY
* STRUCTURE
* CALLING SEQUENCE:
*       JMS CHEW
*       ----                    (RETURN)
* AC CONTENTS ON ENTRY:
*    POINTER TO MAXIMUM ELEMENT IN THE STRUCTURE TO BE CHEWED
* THE MAXIMUM ELEMENT SPECIFIED MUST OWN ALL LEVELS WHICH OWN
* ELEMENTS OF THE STRUCTURE.

CHEW    $DC 0
        JMS* =T.L
        $DC 0
        DAC CHEW6
        LAC =CHEWQ
        JMS* =Q.C
CHEW1   LAC* CHEW6
        SNA
        JMP CHEW5
        AND =7777
        SAD =2010
        JMP CHEW4
        LAC CHEW6
        CMA
        TAD FREE
        CMA
        SMA
        JMP CHEW35
        LAC CHEW6
        AND =70000
        SAD =10000
        SKP
        JMP CHEW5
CHEW2   LAC CHEW6
        JMS ATTR
        JMP CHEW3
        DAC CHEW7
        LMQ
        LAC CHEW6
        JMS* =S.TR
```

```
       $DC 0
       LAC CHEW7
       LMQ
       LAC =CHEWQ
       JMS* =Q.A
       $DC 0
       JMP CHEW2
CHEW3  LAC CHEW6
       JMS* =S.TD
       $DC 0
       JMP CHEW5
CHEW35 CLC
       TAD CHEW6
       JMS CORFRE
       JMP CHEW5
CHEW4  LAC CHEW6
       JMS* =L.L
CHEW5  LAC =CHEWQ
       JMS* =Q.F
       JMP *+3
       DAC CHEW6
       JMP CHEW1
       JMS* =T.U
       $DC CHEW

CHEWQ  $DC *+200
       $DS 200
```

```
* CHEW ATTRIBUTES
* CALLING SEQUENCE
*       JMS CHEWA
*       ----                    (RETURN)
* AC CONTENT ON ENTRY:
*   POINTER TO LEVEL WHOSE ATTRIBUTES ARE TO BE CHEWED

CHEWA   $DC 0
        JMS* =T.L
        $DC 0
        DAC CHEWA1
        LAC CHEWA1
        JMS ATTR
        JMP *+11
        DAC CHEWA2
        LMQ
        LAC CHEWA1
        JMS* =S.TR
        $DC 0
        LAC CHEWA2
        JMS CHEW
        JMP CHEWA+4
        JMS* =T.U
        $DC CHEWA
```

```
* FIND ATTRIBUTE OF LEVEL
* CALLING SEQUENCE:
*       JMS ATTR
*       ----                    (RETURN IF NO MORE ATTRIBUTES)
*       ----                    (RETURN IF ATTRIBUTE FOUND)
* AC CONTENTS ON ENTRY:
*   ADDRESS OF LEVEL -- GET POINTER TO FIRST ATTRIBUTE
*   ZERO             -- GET POINTER TO NEXT ATTRIBUTE

ATTR    $DC 0
        SNA
        JMP ATTR1
        TAD =7
        DAC ATTR2
        LAC* ATTR2
        DAC ATTR2
        LAC* ATTR2
        AND =777770
        SAD =762010
        SKP
        JMP* ATTR
        ISZ ATTR2
        LAC* ATTR2
        ISZ ATTR
        JMP* ATTR
ATTR1   LAC ATTR2
        TAD =2
        JMP ATTR+4
```

```
* CHECK TRACKING THRESHOLDS
* CALLING SEQUENCE:
*       JMS THRS-              (THRSX OR THRSY)
*       $DC ----               (REFERENCE COORDINATE C)
*       $DC ----               (THRESHOLD VALUE T)
*       ----                   (Z > C+T-1)
*       ----                   (C-T < Z < C+T)
*       ----                   (Z < C-T+1)
* THRSY -- TEST Y TRACKING COORDINATE
* THRSX -- TEST X TRACKING COORDINATE
* "Z" DENOTES EITHER THE X OR Y TRACKING COORDINATE

THRSX   $DC 0
        JMS* =X.X
        CMA
        TAD* THRSX
        ISZ THRSX
        TAD* THRSX
        CMA
        SMA
        JMP THRSX1
        TAD* THRSX
        TAD* THRSX
        SPA
        ISZ THRSX
        ISZ THRSX
THRSX1  ISZ THRSX
        JMP* THRSX

THRSY   $DC 0
        LAC THRSY
        DAC THRSX
        JMS* =X.Y
        JMP THRSX+2
```

```
* FIND COORDINATES OF OWNER LEVEL ON LAST DISPLAY INTERRUPT
* CALLING SEQUENCE:
*       JMS COORDS
*       ----                   (RETURN IF LEVEL NOT AN OWNER)
*       ----                   (RETURN)
* AC CONTENT ON ENTRY:
*    POINTER TO OWNER LEVEL
* AC CONTENT ON RETURN:
*    Y COORDINATE
* MQ CONTENT ON RETURN:
*    X COORDINATE

COORDS $DC 0
       JMS* =T.L
       $DC 0
       DAC COOLVL
       LAC =COOQ
       JMS* =Q.C
       DZM COOSW
       DZM COOCNT
       LAC COOCNT
       JMS* =D.0
       JMP *+16
       DAC COOTMP
       SAD COOLVL
       DAC COOSW
       LAC COOSW
       SNA
       JMP *+6
       LAC COOTMP
       LMQ
       LAC =COOQ
       JMS* =Q.I
       $DC 0
       ISZ COOCNT
       JMP *-17
       LAC COOSW
       SNA
       JMP COOFAL
       LAC =3
```

```
        DAC COOSKL
        DZM COOX
        DZM COOY
COONXT  LAC =COOQ
        JMS* =Q.F
        JMP COODUN
        TAD =2
        DAC COOTMP
        LAC* COOTMP
        LRS 10
        AND =1
        SNA+CLA+CLL
        JMP *+3
        LLS 2
        DAC COOSKL
        LAC COOSKL
        XOR =640700
        DAC COOS
        ISZ COOTMP
        ISZ COOTMP
        LAC* COOTMP
        JMS* =C.CB
        CLL
COOS    $DC 0
        TAD COOY
        DAC COOY
        ISZ COOTMP
        LAC* COOTMP
        JMS* =C.CB
        CLL
        XCT COOS
        TAD COOX
        DAC COOX
        JMP COONXT
COODUN  LAC COOX
        LMQ
        LAC COOY
        ISZ COORDS+2
COOFAL  JMS* =T.U
        $DC COORDS
```

```
COOQ    $DC *+103
        $DS 103
```

```
* MOVE LEVEL WITH TRACKING COORDINATES
* CALLING SEQUENCE:
*       JMS TRAC
*       LAC ----                 (LAC POINTER TO LEVEL)
*       ----                     (RETURN)

TRAC    $DC 0
        LAC PENY
        DAC TRACY
        LAC PENX
        DAC TRACX
        XCT* TRAC
        TAD =4
        DAC TRACTP
        LAC* TRACTP
        JMS* =C.CB
        LLSS 3
        DAC TRACYL
        ISZ TRACTP
        LAC* TRACTP
        JMS* =C.CB
        LLSS 3
        DAC TRACXL
TRAC1   JMS* =X.X
        LMQ
        CMA
        TAD TRACX
        CMA
        DAC TRACXI
        LACQ
        DAC TRACX
        JMS* =X.Y
        LMQ
        CMA
        TAD TRACY
        CMA
        DAC TRACYI
        LACQ
        DAC TRACY
        LAC TRACXI
```

```
TAD TRACXL
DAC TRACXL
LRSS 25
XCT* TRAC
JMS* =S.LX
LAC TRACYI
TAD TRACYL
DAC TRACYL
LRSS 25
XCT* TRAC
JMS* =S.LY
JMS* =X.S
JMP *+4
ISZ TRAC
JMP* TRAC
JMP TRACI
JMS* =T.P
```

```
       $TITLE                SELMA MISCELLANEOUS SUBROUTINES

* DEFINE SYMBOL NAME & SEND CREATE COMMAND TO /360
* CALLING SEQUENCE:
*      JMS NAMDEF
*      LAC ----              (LAC POINTER TO SYM OR CON LVL)
*      JMS ----              (JMS TO GEN PAR SUBROUTINE)
*      ----                  (RETURN IF NO NAME AVALIABLE)
*      ----                  (RETURN IF SUCCESSFUL)
* AC CONTENT ON ENTRY:
*   TYPE NUMBER IN BITS 10-17

NAMDEF $DC 0
       JMS* =T.L
       $DC 0
       AND =377
       DAC S0100+1
       AND =200
       TAD =NAME+1
       DAC NAM4
       LAW -177
       DAC NAM5
NAM1   LAC* NAM4
       SNA
       JMP NAM2
       ISZ NAM4
       ISZ NAM5
       JMP NAM1
       JMP NAM3
NAM2   XCT* NAMDEF+2
       DAC* NAM4
       LAC NAM4
       TAD =-NAME
       CLQ
       LLS 11
       XOR S0100+1
       DAC S0100+1
       ISZ NAMDEF+2
       XCT* NAMDEF+2
NAM3   ISZ NAMDEF+2
```

```
ISZ  NAMDEF+2
JMS*  =T.U
$DC  NAMDEF
```

```
* FIND SYMBOL NAME
* CALLING SEQUENCE:
*       JMS NAMGET
*       ----                      (RETURN IF NO CORRESPONDING NAME)
*       ----                      (RETURN IF SUCCESSFUL)
* AC CONTENT ON ENTRY:
*    POINTER TO SYMBOL
* AC CONTENT ON RETURN:
*    NAME OF SYMBOL IN BITS 10-17, BITS 0-9 CLEAR

NAMGET $DC 0
       AND =77777
       DAC NAM6
       LAC =NAME+1
       DAC NAM4
       LAW -377
       DAC NAM5
       LAC* NAM4
       AND =77777
       SAD NAM6
       JMP *+5
       ISZ NAM4
       ISZ NAM5
       JMP *-6
       JMP* NAMGET
       LAC NAM4
       TAD =-NAME
       ISZ NAMGET
       JMP* NAMGET

NAME   $DS 400
```

```
* SET LIGHT PEN SERVICE & STOP TRANSLATION
* CALLING SEQUENCE:
*       JMS PENSET
*       ----                    (RETURN)
* AC CONTENT ON ENTRY:
*    POINTER TO NEW LIGHT PEN SERVICE TASK

PENSET $DC 0
       DAC TRAPEN
       JMS* =D.P
       LAW -400
       DZM TRAS
       LMQ
       LAC LVLTRA
       JMS* =S.LY
       JMP* PENSET
```

```
* READ DISPLAY COORDINATES AND START TRACKING
* CALLING SEQUENCE:
*       JMS PENHIT
*       ----                    (RETURN)
* X & Y COORDINATES ARE STORED IN PENX & PENY BY THIS SUBROUTINE

PENHIT $DC 0
       JMS* =D.Y
       TAD =4
       AND =777770
       DAC PENY
       TAD =700
       SPA
       JMP PENEND
       JMS* =D.X
       TAD =4
       AND =777770
       DAC PENX
       LMQ
       LAC PENY
       JMS* =X.I
       JMP* PENHIT
```

```
* DETERMINE LIGHT PEN REGION RELATIVE TO PENX AND PENY
* CALLING SEQUENCE:
*       JMS PENREG
*       $DC ----              (THRESHOLD VALUE)
*       ----                  (RETURN)
* AC CONTENT ON RETURN:
*   ALL BITS 0 EXCEPT ONE WHICH DENOTES REGION:
*                 15    16    17
*                 12    13    14
*                  9    10    11

PENREG $DC 0
       LAC* PENREG
       ISZ PENREG
       DAC PENY+1
       DAC PENX+1
       DZM PENTMP
       JMS THRSY
PENY   $DC 0
       $DC 0
       ISZ PENTMP
       ISZ PENTMP
       LAC PENTMP
       TAD PENTMP
       TAD PENTMP
       XOR =660500
       DAC *+7
       JMS THRSX
PENX   $DC 0
       $DC 0
       ISZ *+3
       ISZ *+2
       LAC =400
       $DC 0
       JMP* PENREG
```

```
* MOVE ALL ATTRIBUTES FROM ONE LEVEL INTO ANOTHER
* CALLING SEQUENCE:
*       JMS PENMOV
*       LAC ----             (LAC POINTER TO "TO" LEVEL)
*       LAC ----             (LAC POINTER TO "FROM" LEVEL)
*       ----                 (RETURN)
* AC CONTENT ON ENTRY:
*    Y DISPLACEMENT TO BE ADDED TO ALL MOVED LEVELS
* MQ CONTENT ON ENTRY:
*    X DISPLACEMENT TO BE ADDED TO ALL MOVED LEVELS

PENMOV $DC 0
       DAC PENMY
       LACQ
       DAC PENMX
       XCT* PENMOV
       DAC PENSNK
       ISZ PENMOV
       XCT* PENMOV
       DAC PENSOU
       ISZ PENMOV
PENM1  LAC PENSOU
       JMS ATTR
       JMP* PENMOV
       DAC PENMT
       LMQ
       LAC PENSOU
       JMS* =S.TR
       $DC 0
       LAC PENMT
       TAD =4
       DAC PENMC
       LAC* PENMC
       JMS* =C.CB
       TAD PENMY
       LMQ
       LAC PENMT
       JMS* =S.LY
       ISZ PENMC
       LAC* PENMC
```

```
JMS* =C.CB
TAD PENMX
LMQ
LAC PENMT
JMS* =S.LX
LAC PENMT
LMQ
LAC PENSNK
JMS* =S.TI
$DC 0                   *
CLA
JMP PENM1
```

```
* COUNT ATTRIBUTE POSITION IN LEVEL
* CALLING SEQUENCE:
*       JMS PENCNT
*       ----                    (RETURN)
* AC CONTENT ON ENTRY:
*   POINTER TO OWNER LEVEL
* MQ CONTENT ON ENTRY:
*   POINTER TO ATTRIBUTE WHOSE POSITION IS TO BE DETERMINED
* AC CONTENT ON RETURN:
*   POSITION OF ATTRIBUTE IN LEVEL RELATIVE TO LAST ATTRIBUTE

PENCNT  $DC 0
        DAC PENCT
        LACQ
        DAC PENCA
        DZM PENCC
        LAC PENCT
        SKP
PENC1   CLA
        ISZ PENCC
        JMS ATTR
        JMP PENC2
        SAD PENCA
        SKP
        JMP PENC1
        LAC PENCC
        DAC PENCCS
        ISZ PENCC
        JMP PENC1
PENC2   LAC PENCCS
        CMA
        TAD PENCC
        JMP* PENCNT
```

```
* READ ALL NECESSARY LEVEL ADDRESSES
* CALLING SEQUENCE:
*       JMS PENLVL
*       ----                      (RETURN)

PENLVL  $DC 0
        LAC =PENLSW
        DAC PENLP
        DZM PENLSW
        LAC =10
        DAC PENLT
        JMS* =D.0
        JMP *+5
        DAC* PENLP
        LAC PENLSW
        SAD LVLDGM
        ISZ PENLP
        CLC
        TAD PENLT
        SMA
        JMP *-12
        DZM* PENLP
        ISZ PENLP
        LAC PENLP
        SAD =LVLOUT+4
        JMP* PENLVL
        JMP *-5

PENLSW  $DC 0
LVLFRG  $DC 0
LVLSYM  $DC 0
LVLPRT  $DC 0
LVLOUT  $DC 0
LVLEAF  $DC 0
        $DS 2
```

```
* COLLECT GARBAGE
* CALLING SEQUENCE:
*       JMS CORGRB
*       ----                    (RETURN)

CORGRB  $DC 0
        DZM CORL1
        LAC FREE
        DAC CORL
        LAC* CORL
        SPA
        JMP *+6
CORG1   TAD CORL
        SAD =40000
        JMP CORG3
        DAC CORL
        JMP *-7
        CMA
        TAD =1
        TAD CORL
        DAC CORL2
        LAC CORL
        DAC CORL3
        ISZ CORL
        LAC CORL1
        DAC* CORL
        LAC CORL3
        DAC CORL1
        LAC CORL2
        SAD =40000
        JMP CORG3
        DAC CORL
CORG2   LAC* CORL
        SMA
        JMP CORG1
        TAD* CORL1
        DAC* CORL1
        LAC* CORL
        CMA
        TAD =1
```

```
        TAD CORL
        SAD =40000
        JMP CORG3
        DAC CORL
        JMP CORG2
CORG3   LAC CORL1
        DAC CORCHN
        JMP* CORGRB
```

```
* GET CORE BLOCK
* CALLING SEQUENCE:
*       JMS CORGET
*       ----                      (RETURN IF BLOCK NOT AVAILABLE)
*       ----                      (RETURN)
* AC CONTENT ON ENTRY:
*   SIZE OF DESIRED BLOCK
* AC CONTENT ON RETURN:
*   ADDRESS OF BLOCK

CORGET  $DC 0
        SNA+SPA
        JMP* CORGET
        DAC COR2
        JMS COR1
        SKP
        JMP COR3
        JMS CORGRB
        LAC COR2
        JMS COR1
        JMP* CORGET
COR3    ISZ CORGET
        JMP* CORGET
COR1    $DC 0
        DAC CORSIZ
        ISZ CORSIZ
        LAC =CORCHN
        DAC CORL1
        LAC CORCHN
        SNA
        JMP* COR1
        DAC CORL
COR4    LAC* CORL
        TAD CORSIZ
        SPA+SNA
        JMP COR5
        ISZ CORL
        LAC CORL
        DAC CORL1
        LAC* CORL
```

```
        SNA
        JMP* COR1
        DAC CORL
        JMP COR4
COR5    DAC CORT1
        SAD =-1
        JMP COR4+4
        SNA
        JMP COR6
        LAC CORSIZ
        DAC* CORL
        TAD CORL
        DAC* CORL1
        DAC CORL1
        LAC CORT1
        DAC* CORL1
        ISZ CORL1
COR7    ISZ CORL
        LAC* CORL
        DAC* CORL1
        LAC CORL
        ISZ COR1
        JMP* COR1
COR6    LAC* CORL
        CMA
        TAD =1
        DAC* CORL
        JMP COR7
```

```
* FREE ALL ALLOCATABLE CORE
* CALLING SEQUENCE:
*       JMS CORINT
*       ----                    (RETURN)

CORINT $DC 0
       LAC FREE
       DAC CORCHN
       TAD =-40000
       DAC* CORCHN
       LAC CORCHN
       TAD =1
       DAC CORT
       DZM* CORT
       JMP* CORINT
```

```
* FREE CORE BLOCK
* CALLING SEQUENCE:
*       JMS CORFRE
*       ----                    (RETURN)
* AC CONTENT ON ENTRY:
*    ADDRESS OF BLOCK TO BE FREED

CORFRE $DC 0
       DAC CORT
       LAC CORCHN
       DAC* CORT
       LAC CORT
       TAD =-1
       DAC CORCHN
       LAC* CORCHN
       CMA
       TAD =1
       DAC* CORCHN
       JMP* CORFRE
```

```
* MOVE A BLOCK OF CORE
* CALLING SEQUENCE:
*       JMS CORMOV
*       LAC ----                 (LAC ADDRESS OF "TO" BLOCK)
*       LAC ----                 (LAC ADDRESS OF "FROM" BLOCK)
*       ----                     (RETURN)
* AC CONTENT ON ENTRY:
*    TWO'S COMPLEMENT OF LENGTH OF BLOCK

CORMOV $DC 0
       DAC CORCNT
       XCT* CORMOV
       DAC CORSNK
       ISZ CORMOV
       XCT* CORMOV
       DAC CORSOU
       ISZ CORMOV
       LAC* CORSOU
       DAC* CORSNK
       ISZ CORSNK
       ISZ CORSOU
       ISZ CORCNT
       JMP *-5
       JMP* CORMOV
```

```
* STORE ZEROS THROUGHOUT CORE BLOCK
* CALLING SEQUENCE:
*       JMS CORZRO
*       LAC ----              (LAC ADDRESS OF CORE BLOCK)
*       ----                  (RETURN)
* AC CONTENT ON ENTRY:
*   TWO'S COMPLEMENT OF LENGTH OF BLOCK

CORZRO $DC 0
       DAC CORCNT
       XCT* CORZRO
       DAC CORSNK
       DZM* CORSNK
       ISZ CORSNK
       ISZ CORCNT
       JMP *-3
       JMP* CORZRO
```

```
        $TITLE                  SELMA INITIALIZATION

SELMA   LAW 633
        JMS* =T.A
        LAC =SELT1
        JMS* =L.T
        LAC =637475
        JMS* =B.T
        LAW 630
        JMS* =T.R
        LAC =PBT
        JMS* =P.T
        JMS* =P.E
        DZM KBDPB
        LAC =KBDPB
        DAC KBDPP
        DZM KBDVSW
        JMS* =S.LH
        DAC LVLHAL
        JMS* =S.TL
        $DC 0                   *
        DAC LVLSCR
        LMQ
        LAC LVLHAL
        JMS* =S.TI
        $DC 0                   *
        LAW 500
        LMQ
        LAC LVLSCR
        JMS* =S.LP
        JMS BUTX
        $DC SELT1
        $DC LVLHAL
        $DC 370
        $DC -34
        $DC 500
        $DC PENEND
        $DC 0                   *
        JMS* =S.TL
        $DC 0                   *
```

```
        DAC LVLBUT
        LMQ
        LAC LVLHAL
        JMS* =S.TI
        $DC 0                   *
        JMS* =S.TL
        $DC 0                   *
        DAC LVLDGM
        LAW 60
        LMQ
        LAC LVLDGM
        JMS* =S.LP
        JMS CORINT
        LAW -400
        JMS CORZRO
        LAC =NAME /PAR
        DZM PENXL
        DZM PENYL
        DZM NAMDEF+2
        DZM BUTN+2
        DZM BUTX+2
        DZM LVL+2
        DZM CHEWA+2
        DZM CHEW+2
        DZM COORDS+2
        LAC =QAS
        JMS* =T.S
        LAC =KBD
        JMS* =T.S
        LAC =CON
        JMS* =T.S
        SKP
        SKP
        JMS* =T.P
        LAC =S0005
        JMS QASCM
        JMS* =T.F

SELTI   $DC 3
        $TEXT "SELMA-3"
```

```
SELT2   $DC 6
        $TEXT "TRANSLATE"
        $DC 747576
        $TEXT "   ON"

FREE    $DC 36200              BEGINNING OF FREE CORE

        $END
```

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

ORIGINATING ACTIVITY *(Corporate author)* THE UNIVERSITY OF MICHIGAN CONCOMP PROJECT	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP

REPORT TITLE

SELMA: A CONVERSATIONAL SYSTEM FOR THE GRAPHICAL SPECIFICATION MARKOVIAN QUEUEING NETWORKS

DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

Technical Report 23

AUTHOR(S) *(First name, middle initial, last name)*

JAMES H. JACKSON

REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
October, 1969	73	5

CONTRACT OR GRANT NO. DA-49-083 OSA 3050 PROJECT NO.	9a. ORIGINATOR'S REPORT NUMBER(S) Technical Report 23
	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)* SEL Technical Report 45

DISTRIBUTION STATEMENT

Qualified requesters may obtain copies of this report from DDC

SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY Advanced Research Projects Agency

ABSTRACT

This report discusses the design and use of the Systems Engineering Laboratory's Markovian Analyzer (SELMA) system for a DEC 339 computer display terminal. This system provides interactive graphics support for a program which was developed concurrently for the IBM 360/67 to analyze a class of Markovian queueing networks. Special features of the system include handling of all graphic operations at the terminal and recognition of patterns of motion of the light pen to provide a human-oriented drawing capability.

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Interactive computer graphics Remote display terminal Network diagrams Queueing models DEC 339						