Innovative Computer Aided Diagramming

by

Daniel R. Wolfe

Project Report submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of
Master of Science
in
Computer Science

APPROVED:

John A. N. Lee, Chairman

James D. Arthur

J. Patrick Bixler

February, 1989

Blacksburg, Virginia

Innovative Computer Aided Diagramming

by

Daniel R. Wolfe

John A. N. Lee, Chairman

Computer Science

(ABSTRACT)

Computer aided diagramming (CAD) tools today are based on the approach of a fixed symbol into which text is fitted. This approach leads to shortfalls in human interfacing and tool development. The inverse view of entering the text and then drawing the symbol around it provides the basis for an innovative approach that is more natural and user friendly. This report documents benefits realized from the new approach in the development of a CAD tool referred to as DiagramEdit.

Acknowledgements

I would like to thank Dr. John A. N. Lee and Mr. Richard P. Cullen for the help they have given me throughout this work. I would also like to thank Dr. James D. Arthur and Dr. J. Patrick Bixler for their service on my committee.

This project report is dedicated to my loving parents.

Table of Contents

List of Figures

Chapter 1. Introduction

The fundamental drawing techniques employed by computer aided diagramming (CAD) tools today are based on a very limited approach that leads to shortfalls in human interfacing and tool development. This approach is based on the view of a fixed symbol into which text is fitted. The inverse view of entering the text and then drawing the symbol around it provides the basis for an innovative approach that is more natural and user friendly. This approach originated in research conducted at the Naval Surface Warfare Center (NSWC) on this subject.

Researchers at NSWC examined several CAD tools and concluded that their human interfaces were cumbersome and unfriendly. The approach employed by these tools justified the need for a CAD tool based on the new approach. The tool that was developed is referred to as DiagramEdit. DiagramEdit is a free form CAD tool that evolved from an extensive study of natural and efficient human-computer interface techniques. It was discovered during the development of DiagramEdit that a tool based on the new approach is not difficult to implement.

This report focuses on the benefits realized from this new approach in the development of DiagramEdit. Chapter 2 contains a review of the drawing techniques in three CAD tools on the market today that run on the same hardware as DiagramEdit. Chapter 3 contains a review of the drawing techniques in DiagramEdit and compares them to those employed by the tools reviewed in Chapter 2. Chapter 4 provides an account of the development of DiagramEdit and includes several of the algorithms used. Chapter 5 discusses the applications of DiagramEdit. The report concludes with a list of extensions that would enhance the capabilities of DiagramEdit in Chapter 6. A comprehensive user's guide is provided in Appendix A.

Chapter 2. A Review of Existing CAD Tools

DiagramEdit runs on an IBM PC or compatible system. Thus, three CAD tools on the market today that run on the same hardware are reviewed below. Each review consists of a description of the screen layout and the drawing of a diagram followed by a discussion of the drawing techniques and the drawbacks and/or advantages of these techniques. The three tools chosen for review are Interactive EasyFlow by HavenTree Software Limited, the Yourdon Analyst/Designer Toolkit by Yourdon, Inc., and AdaGRAPH by the Analytic Sciences Corporation.

2.1 Interactive EasyFlow

Interactive EasyFlow is an on-screen interactive graphics program that allows one to compose and modify flowcharts and organization charts.

2.1.1 Screen Layout

Once Interactive EasyFlow is running, three windows and a Status Bar appear on the screen. Their content is described below.

In the upper left corner is the Status Bar. This is in inverted characters and indicates at all times what Interactive EasyFlow is doing. When it says 'Interactive EasyFlow', it means that Interactive EasyFlow is waiting for a command.

The small window in the upper left area of the screen is the Text/Message window. User text (for example, text to go inside shapes) is entered here and messages from Interactive EasyFlow also appear here.

The small window in the lower left area of the screen is the Current Shape window. This window shows the content of the Current Shape area of the chart. At startup this window is empty.

The large window in the right area of the screen is the Chart window. This window shows part or all of the chart being developed, usually in dot form. This means that each character in the actual chart shows up on the screen as a single dot. With this type of display the words inside the shapes cannot be read. That is the purpose of the Current Shape window. However, it does allow one to see the overall layout of the chart, and the relationship between shapes and lines.

The rectangular block in the Chart window is the Shape cursor. It shows the current position in the chart. Whenever the Shape cursor is on the screen, the cursor arrow keys can be used to move it around the chart.

With Interactive EasyFlow, charts are composed on a grid that is 16 charting shapes wide by 16 charting shapes high. The rows of the grid are assigned numbers from one to 16, while the columns of the grid are assigned letters 'A' to 'P'. As the cursor keys are used to move the Shape cursor around the chart, the Status Bar shows its current location. At startup the Status Bar shows 'Interactive EasyFlow A1', indicating that the Shape cursor is in the upper-left corner of the chart page in grid location 'A1'.

2.1.2 Drawing a Diagram

The 'F3' key is used to create a text symbol. When this key is pressed, the Status Bar shows 'Shape Request', a small Character cursor appears in the Text/Message window, and Interactive EasyFlow waits for text to be entered. The first word of text specifies the type of shape, and the rest is the text to appear inside the shape. For example, if the 'P' key is pressed, the 'P' appears in the Text/Message

window, and a process shape appears in the Current Shape window.

As more text is typed, it appears in both windows. In the Text/Message window it appears simply as one character after another, but in the Current Shape window it appears centered within the shape. This allows one to see how the text will appear inside the shape in the final chart.

The first word of text specifies the type of shape to be used. There are over forty standard charting shapes supplied with Interactive EasyFlow. The four most commonly used shapes are the terminal, process, decision, and input-output shapes. These and several others are available in three sizes: small, medium and large. The base shape code gives the medium size shape, prefixing the shape code with 'S' gives the small version, and 'L' gives the large version.

When text is typed inside shapes, Interactive EasyFlow automatically formats and centers the text before placing it inside the shape. When creating lines of text to go inside a shape, Interactive EasyFlow attempts to avoid the edges of the shape in order to give the text a pleasing appearance. If a large amount of text is to go inside a shape, then

Interactive EasyFlow will use the full area inside a shape, edges and all.

Interactive EasyFlow does give some control over the formatting of text in shapes, but not a great deal. With full manual formatting, Interactive EasyFlow does not attempt to center the lines vertically within the shape. In addition, depending on which formatting character was used, lines of text are either left justified, centered or right justified when placed in the shape. The edges of the shape are not avoided either. When formatting under manual control, the edges have to be avoided by a user.

The procedure for modifying existing shapes in Interactive EasyFlow is very similar to that used for creating new shapes. The Shape cursor is moved to the desired shape and the 'F3' key is pressed. Text can then be edited as previously described.

A shape is deleted from the chart by pressing the 'F3' key and deleting all text from the Text/Message window. When the 'Return' key is pressed, the shape is deleted. Another method for deleting the shape is to move the Shape cursor to the desired shape and press the 'Shift'-'F3' keys.

The 'F4' key is used to draw a line. When this key is pressed, the Status Bar shows 'Line Request' and the Shape cursor vanishes from the Chart window. It is replaced by a small blinking block called the Line cursor. The cursor arrow keys are used to move the Line cursor from the shape where the line request was initiated, to the destination shape.

A line is drawn on the screen as follows. When the Line cursor is moved around the screen, a thick line is left so it can be seen from where the line request came. This thick line is converted into a thin line at the end of the line request operation. Unlike the Shape cursor, which moves a full shape location at a time, the Line cursor moves only half a shape location at a time. This is necessary so that lines can run between adjacent shapes. If a mistake is made when requesting a line, then the cursor arrow keys can be used to back up and correct the line. The line request operation is terminated when the Line cursor runs into an existing shape. Another method of terminating a line request is to move the Line cursor to a place where a shape could go, but which is presently empty, and press the 'F3' key. The editor is then entered and a new shape is created.

As soon as the line request operation is complete, Interactive EasyFlow will proceed to reroute the line and convert it into a thin line. Therefore, no matter how much wandering around the chart was done during a line request, Interactive EasyFlow pays attention only to how the line request exited the 'From' shape and entered the 'To' shape. As a chart evolves, Interactive EasyFlow automatically changes the line routing to take advantage of the current chart layout. There is also the option of turning automatic rerouting off.

Most flowcharts involve decisions, and decisions are typically represented using a diamond decision shape. The Shape key is used to create a decision shape, and the Line key to create lines, but some method is needed for putting conditional text like 'No' or 'Yes' on the lines. This is handled by Interactive EasyFlow as part of the line request process. Each line can have only one conditional text word associated with it, and that word can only be four characters long. Since conditional text is short, the only editing facility available is the 'Backspace' key which backs up and deletes the previous character. The cursor keys are not used for editing in this case because they are being used to move the Line cursor.

Interactive EasyFlow can draw solid and dashed lines. It defaults to solid lines. When in the midst of creating a line, the line type can be changed by pressing the 'Alt'-'L' keys.

In addition to creating lines, lines can be edited by moving the Shape cursor to the shape where the line to be edited originates, pressing the 'F4' key, and pressing a cursor arrow key to indicate the line to be edited. A thick line will sprout up along the path of the line, stopping just short of the destination shape. The line can then be directed towards a different destination.

In Interactive EasyFlow, there is no way to specify a line request run into a line that eventually goes to a destination. Instead, the line request must run to its ultimate destination, ignoring totally any other lines which may already exist in the chart. If lines can be merged, then Interactive EasyFlow will notice this when it reroutes the chart, and it will merge the lines appropriately. There is no way to prevent this merging of lines.

Lines exiting from the current shape are deleted with the 'Shift'-'F4' keys. When these keys are pressed, the Text/Message window contains the message 'Indicate

direction' followed by a list of directions that correspond to lines exiting from this shape. A cursor arrow key is pressed to indicate the particular line to be deleted. Only lines that exit the current shape can be deleted in this manner. To delete lines that enter the current shape, the Shape cursor must be moved to the shape where the lines originate and they must be deleted from there. If a shape is deleted, all lines associated with that shape both entering and exiting are deleted as well.

With Interactive EasyFlow, charts are done on a grid which is 16 shape locations wide by 16 shape locations high, and there are only certain locations where shapes and lines can go. As a result, there are a couple of things to note about the grid:

1. Flow-lines connecting shapes cannot cross.

2. There is room between adjacent shapes for one and only one flow-line.

3. Flow-lines can go through the positions where shapes are allowed, provided there is no actual shape in that location.

Interactive EasyFlow provides several other functions such as zooming and panning to view portions of a chart and a cutting and pasting facility to do various things with an arbitrary block of shapes. It also provides the capabilities for storing a chart on disk, loading a chart from disk, and printing a chart. More information on these functions and others can be found in [5].

2.1.3 Discussion of Drawing Techniques

Interactive EasyFlow is adequate for drawing flowcharts and organization charts. However, the drawing approach is based on the view of a fixed symbol into which text is fitted. As a result, the drawing techniques are unnatural and awkward. Several of the drawbacks in these techniques are discussed below.

Interactive EasyFlow provides charting shapes in three different sizes. Thus, text has to be fitted inside a shape. In some cases a user may have to go to a larger size shape just to fit one more word. A user should not have to be concerned whether text will fit inside a shape.

When text is entered, Interactive EasyFlow formats and centers text before placing it inside a shape. In many

cases a user does not want text to be automatically formatted. Interactive EasyFlow does give some control over the formatting of text, but not a great deal. A user should have total control over the formatting of text in shapes.

Interactive EasyFlow has a cumbersome method for entering conditional text as part of the line request process. It should be done in the same manner as a text symbol.

The use of separate windows for text entry and text formatting is not necessary. As text is typed into the Text/Message window, it also appears in the Current Shape window centered within the shape. If Interactive EasyFlow could allow a user to format text in the Text/Message window as it is being entered, one less window would be needed. As a result, a user could view more of the chart.

The automatic rerouting of flow-lines by Interactive EasyFlow is not necessarily an advantage. In most cases a user does not care for the result of the rerouting and must redraw the flow-lines anyway. Fortunately, this option can be turned off.

The fixed 16 by 16 grid in Interactive EasyFlow severely restricts the placement of flow-lines in a chart. For

example, flow-lines connecting shapes cannot cross. As another example, there is room for one and only one flow-line between adjacent shapes.

There is no way in Interactive EasyFlow to specify a line request run into a line that eventually goes to a destination. Instead, the line request must run to its ultimate destination, ignoring totally any other lines which may already exist in the chart. If lines can be merged, then Interactive EasyFlow merges the lines appropriately when it reroutes the chart. A user cannot prevent this occurrence. Instead, a user should have the freedom to specify a line request run into a line that eventually goes to a destination.

A user is burdened with using zoom and pan functions to view portions of a chart with Interactive EasyFlow. Only the overall layout of the chart can be viewed at one time, and not the text inside the shapes. A user should be able to view the layout of a chart and the text inside the shapes at the same time.

2.2 Yourdon Analyst/Designer Toolkit

The Yourdon Analyst/Designer Toolkit (Workbench) provides Software Engineers with support in all stages of the planning and design of large and small scale systems. Using a graphics mouse, the Workbench allows a user to create a variety of diagrams within a menu driven environment.

2.2.1 Screen Layout

The editing screen appears when the Workbench is invoked from DOS. The areas of the screen are described below.

The message banner is located at the top of the screen, and it provides basic information about the diagrams with which a user is working. The left side of the banner displays the name of the diagram which is active. Error messages also appear in the message banner.

The drawing region is located in the center of the screen, and it is reserved for editing and viewing diagrams.

The navigation port is located in the upper right-hand corner of the screen. Used with the available functions, the port allows the scale of all or part of a specified diagram to be changed.

The icon drum is located on the right side of the screen, and it consists of two columns of graphic symbols called icons. These are the symbols which are available for diagramming.

The Menu Bar appears on the line below the drawing region, and it lists the different families of commands: EDIT, VIEW, FILE, and PARM (parameters).

The Function Bar appears on the bottom line of the screen, and it list all of the commands which are used to create and edit diagrams. The list of commands displayed depends on the function selected from the Menu Bar.

The two scrolling arrows allow access to icons which are not displayed on the screen. The scrolling arrows are located beneath the icon drum.

The Text Input Region appears at the bottom of the screen, superimposed over the Function Bar. Whenever text is entered, it is first previewed in the Text Input Region.

Two session control options are located in the lower left corner of the screen. The CLEAR option allows previously executed functions to be cancelled. The HELP option

provides additional information about either a command, icon or error message.

2.2.2 Drawing a Diagram

A symbol is drawn in the following way. The mouse cursor is positioned on the desired icon and the select button is pressed which highlights the icon. The mouse cursor is then moved somewhere in the drawing region and the select button is pressed again. The chosen icon is drawn at the location of the mouse cursor. The same result can be achieved by first choosing one or more points in the drawing region and then selecting the icon. By choosing multiple points, several copies of an icon can be placed on the screen simultaneously.

The cancel button undoes the last action(s) previously executed. The right button (on a 2 button mouse; the middle button on a three button mouse) is the cancel button. For example, when an icon is highlighted indicating that it has been selected, the cancel button can be pressed to cancel the icon choice. The cancellation is indicated by the disappearance of the highlight on the icon. The cancel button can also be used to remove an icon after it has been

added to a diagram. This is done by placing the mouse cursor on the icon and pressing the cancel button.

The evaluate button is used to execute a Workbench function without having to reselect it after each execution. Pressing the right button on a three button mouse (or both buttons on a two button mouse) activates the evaluate mode.

Text is added to a diagram in the following way. Many icons have text markers which appear as small asterisks in or near the associated icon. These markers indicate the regions on a diagram which are reserved for text. Text is entered by moving the mouse cursor to a text marker and pressing the select button. Next, the mouse cursor is moved to the EDIT function on the Menu Bar and the select button is pressed again. Next, the mouse cursor is moved to the TEXT function on the Function Bar and the select button is pressed a third time. The TEXT function is highlighted and two parallel lines replace the asterisk in the drawing region. As text is entered, the lines separate to indicate how much space the text occupies. The text which is entered temporarily replaces the Function Bar. The 'Return' key is pressed to indicate that text entry is complete. A new line is created by pressing the 'Alt'-'I' keys.

The type of symbol determines the number of text fields. For example, a process symbol in a data flow diagram contains two reserved text fields. The upper field represents the process number and can include only one or two consecutive digits. The lower text field represents the process name. This field accepts entries of any length and may contain multiple lines.

To enter the number and text, the mouse is used to select first the top and then the bottom asterisk. The EDIT function is selected from the Menu Bar, the TEXT function from the Function Bar, and then a process number is entered. The 'Return' key is then pressed and the process name is entered. A new line is created by pressing the 'Alt'-'I' keys. Pressing the 'Return' key again completes the text entry.

Text can also be entered that is not associated with specific symbols in a diagram. This is done by using the Free Form Text Icons, marked on the icon drum as RTEXT, LTEXT, and CTEXT. These three TEXT functions write text on the screen and justify it in relation to the mouse cursor. RTEXT right justifies the text, LTEXT left justifies the text, and CTEXT centers the text about the mouse cursor.

The mouse cursor is moved to any location in the drawing region and the select button is pressed. Text is entered as described above. To insert more than one line of text, the 'Alt'-'I' keys are pressed after the first line of text.

When the EDIT function is selected from the Menu Bar, the Workbench provides other functions on the Function Bar besides the TEXT function. For example, a user can select functions to either change the size, copy, move or delete an icon. Whenever one of these functions is performed on an icon, the associated text is also effected. These same functions are also available for a group of symbols.

A line is drawn with the Workbench in the following way. The mouse cursor is positioned on the line icon and the select button is pressed which highlights the icon. The mouse cursor is then moved onto a symbol in the drawing region and drawing is done by selecting the endpoints of the line segments that make up the line. The Workbench senses that a line is complete when the mouse cursor is moved to the symbol at which the line is to terminate. The line is then drawn. If the line needs to be rerouted, the Workbench provides a RERT function on the Function Bar. The Workbench

does not allow lines to be routed into other lines that eventually go to a destination.

A grid function is also available for diagramming. The grid allows for the creation of diagrams with straight line accuracy by causing the mouse cursor to jump from one point to another. The grid is turned on by selecting the PARM function on the Menu Bar.

The FILE function on the Menu Bar allows basic file maintenance operations to be performed such as saving and loading diagrams.

Once a diagram has been created, the Workbench can be used to verify whether the diagram adheres to established diagramming rules pertaining to a specific type of diagram. This is called diagram verification and is based on a set of predetermined rules that govern each of the symbols and connecting lines in a diagram. More information on this function and others can be found in [7-8].

2.2.3 Discussion of Drawing Techniques

The drawing approach in the Workbench is also based on the view of a fixed symbol into which text is fitted. As a

result, the drawing techniques are again unnatural and awkward. The Workbench does have one advantage in that it does allow a user to draw free form diagrams with an optional grid. There are, however, several drawbacks which are discussed below.

The creation of a text symbol is exceedingly cumbersome and inefficient with the Workbench since it takes five selections with the mouse:

1. To select an icon.

2. To select a position(s) in the drawing region.

3. To select a text marker inside the symbol, which appears as an asterisk.

4. To select the EDIT function from the Menu Bar.

5. To select the TEXT function from the Function Bar.

After these five selections, a user can only see the line of text that is being entered. In addition, a user must watch two parallel lines separate to indicate how much space the text occupies. To enter more lines of text, the 'Alt'-'I' keys must be used.

Modifying a text symbol is another burden on a user with the Workbench. If text needs to be added to a symbol, the size of the text will most likely exceed the size of the icon. As a result, the size of the icon must be increased, creating more work for a user. Also, if there are any connectors connected to the symbol, they remain attached to the icon. This is also the case when an icon is moved. A user may not care for the rerouting and have to redraw the connectors.

The Workbench does not allow connectors to be routed into other connectors that eventually go to a destination. This can result in a diagram that is difficult to read if there are a number of connectors drawn from or to a symbol.

A user is again burdened with zooming, panning and scrolling functions to view portions of a diagram with the Workbench. Instead, a user should be able to view the entire layout of a diagram and the text inside the symbols at the same time.

2.3 AdaGRAPH

AdaGRAPH is a graphical, structured design and development tool for Ada systems developers. AdaGRAPH implements a graphical design language called the Process Abstraction

Language (PAL). This graphical language is higher level than Ada, but is similar in syntax to the Ada language. AdaGRAPH automates George W. Cherry's PAMELA methodology.

2.3.1 Screen Layout

There are four mechanisms by which a user can communicate with AdaGRAPH: the icons, the drop-down menus, the dialog and alert boxes, and the function keys. The icon menu window is a pictorial list of all the operations that can be performed on that graph. An icon is selected with a mouse. After a project has been opened, there are four drop-down menus available from the Menu Bar which is located across the top of the screen. These menus are titled FILE, VIEW, OVERVIEW, and AdaGRAPH. The commands on each drop-down menu are accessed using the mouse. Dialog boxes are specialized cases of windows that are used to convey information to AdaGRAPH. Finally, function keys can be used in AdaGRAPH to speed up work. They are equivalent to making a selection from a drop-down menu.

When AdaGRAPH is invoked from DOS, the screen shows the Project Information Screen dialog box. After a valid DOS directory name is entered, AdaGRAPH creates the DOS

directory to associate the project with that name. After clicking an 'Okay' button with the mouse, the screen appears with two windows. A large window on the right contains the project name and a graphical representation of an executable block, or a shadow object representing the Ada Run Time System. A smaller window on the left side of the screen contains a menu of all the AdaGRAPH icons.

2.3.2 Drawing a Diagram

A text symbol is drawn by positioning the mouse cursor on an icon and pressing the select button. It will become highlighted to indicate that it has been selected. Next, the mouse cursor is moved to a location on the screen and the select button is pressed again. The icon will be drawn at the location of the cursor.

AdaGRAPH does not allow icons to overlap. If more space is needed for drawing, scrolling techniques must be used. There are two scroll bars, two sliders, and four scroll arrows along the bottom line and last column of the screen in AdaGRAPH. The scroll bars represent the size of the file in the window, and the sliders represent the portion of the file that is currently visible. This gives a user a general

idea of the size of a file and the current position. AdaGRAPH provides methods for scrolling in each direction.

The next step in drawing a text symbol is entering the text. This is done by using the mouse cursor to select the labeling icon and the icon in which the text is to be entered. After entering the text in a dialog box, AdaGRAPH will place the text inside the appropriate icon. The exact location at which the text will be positioned inside the icon cannot be controlled; AdaGRAPH determines where within the icon the text is placed.

A symbol is erased by selecting the erase icon and moving the mouse cursor until it is anywhere inside the symbol to be erased. Pressing the select button again causes the symbol to disappear. Next, the mouse cursor is moved to a blank spot on the icon menu and the select button is pressed. This deactivates the erase icon.

Connectors between text symbols are drawn by selecting a data flow icon from the icon menu window and entering the symbol in which the arrow is to originate. When the select button is pressed, the symbol becomes highlighted and a Port Selector dialog box appears on the screen. This box has seven little boxes on each side of it. These little boxes

are ports; they indicate positions on a symbol where arrows can be connected. When a port is selected with the mouse cursor, it becomes highlighted. The selection of a port is completed by clicking an 'Okay' button with the mouse. The same procedure is followed with the destination symbol.

AdaGRAPH provides two options for placing bends in data flow arrows. One, simply by indicating the origin and destination symbols, AdaGRAPH will automatically provide the necessary bends. Two, the mouse cursor can be used to select the desired bends prior to selecting the destination symbol. AdaGRAPH only draws 90-degree data flows and does not allow data flows to connect to other data flows.

Once the data flow arrows have been drawn, they need to be labeled. The initial labeling of arrows is a procedure similar to that for labeling text symbols. When the labeling icon is selected, AdaGRAPH adds a small diamond-shaped handle to each arrow. This is to provide an area in which to point when selecting arrows. By positioning the mouse cursor in the handle of an arrow and typing text into the dialog box, AdaGRAPH places the label on the data flow arrow.

By using the mouse cursor to select the labeling icon and the data flow arrow or text symbol, labels can be modified. When this is done, the Pick a Label Operation dialog box appears. This box contains dialog boxes to either add, delete, rename, change the appearance, or cancel a label.

Data flow arrow handles are erased by selecting the refresh icon from the icon menu. Data flow arrows are erased using the exact same procedure as described above for text symbols.

A file is saved by selecting the file icon with the mouse cursor. The icon becomes highlighted for a brief time while AdaGRAPH stores the information related to the graph. More information on this icon and others can be found in [1-2].

2.3.3 Discussion of Drawing Techniques

AdaGRAPH also has an unnatural and awkward user interface that requires the use of a mouse. However, it is an improvement over the Workbench in that all of the text can be seen as it is entered. As was the case in Interactive EasyFlow and the Workbench, the drawing approach in AdaGRAPH is based on the view of a fixed symbol into which text is fitted. One advantage AdaGRAPH does have is that it allows

a user to draw free form diagrams. There are, however, several drawbacks which are discussed below.

AdaGRAPH does not allow a user to control where text will be placed within a symbol. Instead, AdaGRAPH determines where within the icon the text is placed. This inflexibility can force a user to have to format text as it is entered for proper placement in an icon.

Modifying a text symbol is another burden on a user with AdaGRAPH. If text needs to be added to a symbol, the size of the text will most likely exceed the size of the icon. As a result, the size of the icon must be increased, creating more work for a user. Also, if there are any connectors connected to the symbol, they would have to be rerouted.

AdaGRAPH does not allow free form text with the labeling icon. Instead, it must always be associated with a data flow. This limits the placement of text in a diagram.

AdaGRAPH does not allow data flows to connect to other data flows. This limits flexibility in drawing and can cause a large number of data flows to enter or exit from an icon, thereby making a diagram difficult to read.

The use of ports to indicate positions on a symbol where data flows can exist limits the number of data flows that can enter or exit an icon.

With AdaGRAPH a user is again burdened with zooming, panning and scrolling to view portions of a diagram. A user should be able to view the entire layout of a diagram and the text inside the symbols at the same time.

Chapter 3. A Review of DiagramEdit

Each of the CAD tools reviewed in the previous chapter is based on the approach of a fixed symbol into which text is fitted. This approach results in human interfaces that are cumbersome and unfriendly thereby justifying the need for a CAD tool based on the inverse approach. DiagramEdit is based on the approach of entering text first and then drawing the symbol around it. It turns out that a CAD tool developed in this way offers many benefits in the human interface. The review of DiagramEdit presented below follows the same format as those in the previous chapter.

DiagramEdit is designed to aid in the drawing and modification of a wide range of diagram types used in system and software engineering, some of which are: block, functional flow, operational sequence, state, finite state, data structure, tree, and data flow.

3.1 Screen Layout

In DiagramEdit all of the IBM PC screen except for the bottom line is available for either creating, modifying, or viewing diagrams. This is the diagram area and is 24 rows

by 80 columns in size. The bottom line is reserved for prompts, menus and messages.

3.2 Drawing a Diagram

When DiagramEdit is invoked from DOS, the main menu appears on the bottom line of the screen as shown in Figure 1. An option is selected by pressing the key corresponding to the highlighted character(s). DiagramEdit provides the following options on the main menu:

New -- Clears a diagram from the diagram area and initializes DiagramEdit to start a new diagram. Includes an option for starting the new diagram with a text symbol chosen from the diagram currently in the diagram area.

Draw -- Initiates the drawing of a new diagram or the modification of an existing diagram.

Save -- Saves the diagram currently in the diagram area on the logged drive/directory.

Load Blk/Lst -- Loads a diagram from the logged drive/directory. The diagram desired may be selected from a diagram currently on the screen (load from Block/Symbol) or from a list of the file names on the logged drive/directory.

Print -- Prepares the screen for printing by clearing the bottom line. Printing the file name is optional. Note that the screen may be printed at any time via the DOS "Shift-Print Screen" function. However, unless this option is used the entire bottom line will be included on the hard copy.

Help -- Displays the Help screen for the main menu. A reproduction of this screen is shown in Figure 2.

Esc -- Exits DiagramEdit and returns control to DOS in the current logged drive/directory.

When the 'D' key is pressed to select the draw option on the main menu, the draw display appears on the screen as shown in Figure 3. A triangular cursor called the Draw/Select cursor is positioned in the middle of the diagram area. The cursor can be moved horizontally and vertically with the cursor arrow keys and towards the upper left, upper right, lower left and lower right with the 'Home', 'PgUp', 'End' and 'PgDn' keys, respectively. The cursor moves one character space each time a movement key is pressed. The number of character spaces that the cursor moves can be increased by pressing the 'Tab' key and decreased by pressing the 'Shift'-'Tab' keys. The current number of

character spaces to be skipped is displayed by the 'S#' indicator on the bottom line of the screen. The cursor's current row-column position is displayed by the 'R# C#' indicators.

As indicated on the bottom line, the 'F10' key is used to bring up help screens. The content of the screen will depend on the drawing status. At this point the Help screen shown in Figure 4 will appear. The help screens are meant to serve as a reminder of the keys used to perform the various functions and are not intended as a tutorial.

A text symbol is drawn by moving the Draw/Select cursor anywhere in the diagram area to position the symbol. When the 'F1' key is pressed, the symbol type appears on the screen with a Text cursor inside. At this point either text can be entered and/or modified inside the symbol, the symbol can be moved, or the type can be changed. As text is entered and/or modified, the symbol type surrounding the text expands and contracts according to the size of the text. The 'F1' key is pressed again to complete the symbol. When the symbol is completed, the Draw/Select cursor appears in place of the Text cursor.

Figure 5 illustrates the drawing of a text symbol. Note that the Draw/Select cursor is gone and that the Text cursor appears inside the symbol shown. The Text cursor is only used for text editing and symbol movement. When the 'F10' key is pressed at this point, the Help screen shown in Figure 6 will appear. The creation/modification of text can be seen in Figure 7. Figure 8 shows the completion of the text symbol after the 'F1' key has been pressed.

The symbol type can be changed by pressing the 'F3' key while the Text cursor is active. When the 'F3' key is pressed, the type changes, automatically stepping through all of the types and cycling back to the first type. A type is selected by pressing the 'F3' key a second time. The twelve text symbol types available in DiagramEdit are shown in Figure 9.

Figure 10 illustrates the drawing of a second text symbol in the diagram. With the Text cursor active, the 'F3' key is pressed and the symbol type changes. When the desired symbol type appears the 'F3' key is pressed a second time. Figure 11 shows that the symbol type chosen is a rectangle.

A text symbol is moved in the diagram area with the Text cursor active. The symbol is moved horizontally by

positioning the cursor at the beginning or end of any line and pressing the left or right cursor key. Vertical movement is done by positioning the cursor at the top or bottom line and pressing the up or down cursor key. The symbol will move the number of character spaces displayed by the 'S#' indicator. In addition, the 'R# C#' indicators will be updated to reflect the movement.

An existing text symbol can be modified by positioning the Draw/Select cursor anywhere on the text of the symbol. When the 'F1' key is pressed, the Text cursor appears in place of the Draw/Select cursor and the same functions as described above can be performed. In addition to being modified, existing symbols can also be copied and deleted.

Existing text symbols are copied by positioning the Draw/Select cursor on the text of a symbol and pressing the 'F4' key. The cursor is then moved to the desired position in the diagram area and the 'F4' key is pressed again. The copy will be drawn in the diagram area with the upper left-most character positioned coincident with the Draw/Select cursor.

Existing text symbols are deleted by positioning the Draw/Select cursor on the text of a symbol and pressing the

'F5' key. The symbol and all attached connectors are deleted from the screen.

A connector symbol is drawn by positioning the Draw/Select cursor anywhere on the text of one symbol and pressing the 'F2' key. When the 'F10' key is pressed at this point, the Help screen shown in Figure 12 will appear. As the cursor is moved to the other text symbol, the connector is automatically drawn. When the cursor is positioned on the text of the other symbol, the 'F2' key is pressed again to indicate completion of the connector. DiagramEdit then clips the connector to the edges of the text symbols and draws the head(s). As a result of clipping, precise positioning of the Draw/Select cursor at the start or finish is not a concern. The path of the connector between the symbols is left as drawn. Connectors can be drawn between any combination of text, connector and line symbols.

The connector symbol type is selected with the 'F3' key either before or after the connector is started. When pressed before, the connector/line symbol type indicator on the bottom line will cycle through the four connector and two line symbol types shown in Figure 13. When used after the connector is started, only the connector types will be

cycled. In either case, when the desired symbol type appears, the 'F3' key is pressed again to complete the selection of the symbol type.

The drawing of a connector is shown in Figure 14 prior to clipping. The symbol type displayed at the bottom of the screen is the solid unidirectional connector. The connector is started by positioning the Draw/Select cursor on the text of the symbol on the left and pressing the 'F2' key. After pressing the down cursor arrow key several times, a solid line is produced trailing the ~~Draw/Select~~ cursor. The line is extended to the text of the lower right symbol with the right arrow cursor key, and the 'F2' key is pressed to complete the connector. After the connector is clipped, the diagram appears as shown in Figure 15. As mentioned in the above paragraph, the connector type could have been changed at any time with the 'F3' key while the connector was being drawn.

Connector symbol types can be changed at any time in a manner analogous to text symbols. The Draw/Select cursor is positioned on the symbol and the 'F3' key is pressed. The symbol as well as the indicator at the bottom of the screen will cycle through the four connector types. When the

desired type appears, the 'F3' key is pressed again and cycling will stop with the new symbol displayed.

Connectors are deleted in the same way as text symbols. The Draw/Select cursor is positioned on the symbol and the 'F5' key is pressed. The connector and all connectors attached to it are cleared. Associated text and/or line symbols are not effected.

There are two line symbols available in DiagramEdit, solid and dashed. Line symbols are drawn, deleted and the symbol type is selected and changed in the same way as connectors. The only difference between connector and line symbols is that lines can start and finish at any position in the diagram area. As a result, no clipping ever occurs when a line is completed. Figure 16 shows an example of a completed, dashed line.

3.3 Discussion of Drawing Techniques

As a result of the approach of drawing the symbol around the text, DiagramEdit has a natural and friendly user interface that has many advantages. They are discussed below.

DiagramEdit eliminates a user's concern for the amount of text to be entered by expanding and contracting the symbol as the text is entered. With this technique only the size of the IBM PC screen limits the size of the text thereby introducing the concept of a virtual symbol. The technique also makes DiagramEdit as close as possible to diagrammatic word processing.

With DiagramEdit a user has the option to either edit text, change the symbol type, or move a symbol all at the same time. This technique eliminates the awkward separation/integration of text entry functions found in the CAD tools reviewed in the previous chapter. It also eliminates the distinction between drawing and modifying a text symbol.

A user has complete control of text formatting with DiagramEdit and can always see how the symbol and its text will look in the completed diagram. Text is edited with a subset of TURBO Pascal Editor commands.

Interactive EasyFlow uses a fixed grid for placement of symbols and connections between symbols. Fixed grids restrict symbol placement in diagrams. DiagramEdit, on the other hand, uses a character grid for placing symbols and

connections between symbols. Placement using a character grid provides the maximum flexibility. If the grid is any smaller, say pixel size, slower text symbol movement and connector/line drawing would result.

DiagramEdit allows the whole diagram to be viewed at all times including the text inside the symbols. Each of the tools reviewed in the previous chapter, on the other hand, provides zoom and pan functions to view areas of a diagram. This is an inconvenience for a user. DiagramEdit provides the maximum amount of screen space for modifying and viewing diagrams with only the bottom line reserved for menus and messages.

DiagramEdit works on the minimum IBM PC or compatible and requires no special hardware such as a mouse or light pen. Not even a hard disk is required. This allows for more widespread use of the tool. Many people claim that using cursor keys is easier than using a mouse because they do not have to move their hands from the keyboard.

Diagrams are printed with the standard print screen program provided with DOS. This program prints diagrams on an 8 1/2 by 11 piece of paper which allows for better organization of

a set of diagrams. As a result, parts of a single diagram do not have to be taped together.

DiagramEdit provides connection freedom in that connectors can connect any combination of text, connector and line symbols. This freedom can eliminate the need to draw a large number of connectors to/from the same symbol.

When any symbol is deleted, DiagramEdit deletes all of the attached connectors. This eliminates the need for a user to delete them individually.

DiagramEdit supports a simple method for drawing a hierarchy of diagrams with a user only having to know the name of the top-level diagram. When the new option is selected, a user can either start the new diagram with no symbols or a text symbol that is selected from the diagram currently in the diagram area. When a text symbol is selected, the diagram area is cleared and the symbol is redrawn in the upper left corner of the screen. In addition, DiagramEdit creates a name from the text. If the new diagram is saved with the created name, the load from block option can later be used to load the lower-level diagram in the hierarchy. This same procedure can also be followed to extend diagrams on the same level.

The text symbols in DiagramEdit are based on lines and characters. Therefore, a user can position most of the text symbols so that they touch each other. This gives a user the option to eliminate connectors.

When a diagram is saved with DiagramEdit, the file is stored in an ASCII format on disk. The exact format is described in the next chapter. Most CAD tools store diagram files so that they cannot be read. Since DiagramEdit stores diagram files in ASCII, they can be processed by other programs. An example of such a program is given in Chapter 5.

Chapter 4. The Development of DiagramEdit

This account of the development of DiagramEdit covers important aspects in the design and implementation of the tool. It includes a discussion of the environment in which the tool was developed, the structure of the diagram file, and the algorithms used for drawing symbols, editing text, and clipping connectors.

4.1 Development Environment

DiagramEdit was developed with the TURBO Pascal Version 3.01A Software Development Environment running on an IBM PC with a color graphics monitor. The IBM PC was chosen as the hardware for development because of its widespread availability, low cost and graphics capability. The TURBO System was chosen as the software development environment because it provides a fast compiler, high resolution graphics, and Turtlegraphics procedures that can be called to manipulate a cursor.

High resolution graphics mode allows for the drawing of symbols around text and the line segments in connectors/lines. In high resolution graphics mode the IBM

PC screen displays 80 columns by 25 rows of characters and 640 horizontal by 200 vertical pixels. Character position (1,1) is in the upper left corner and (80,25) is in the lower right corner of the screen. 640 horizontal by 200 vertical pixels give graphics X-coordinates between 0 and 639 and graphics Y-coordinates between 0 and 199. Graphics position (0,0) is in the upper left corner and (639,199) is in the lower right corner of the screen. A character is 8 by 8 pixels.

A line is drawn on the screen by calling the TURBO Pascal procedure 'Draw(X1,Y1,X2,Y2,Color)' which draws a line between the graphics positions specified by (X1,Y1) and (X2,Y2) in the color specified by 'Color'. All parameters are integer expressions. 'Color' is set to zero to draw in the color of the background screen (black) or to one to draw in the color of the text.

TURBO Pascal's Turtlegraphics procedures are called to manipulate the Draw/Select cursor. Turtlegraphics is based on the idea of a 'turtle' that can 'walk' a given distance and turn through a specified angle, drawing a line as it goes along. Turtlegraphics and high resolution graphics are used simultaneously in DiagramEdit.

The TURBO Turtlegraphics procedures operate on turtle coordinates. The turtle's home position (0,0) in this coordinate system is always in the middle of the active window, with positive values stretching to the right and upwards, and negative values stretching to the left and downwards. X-coordinates range from -319 to 320 and Y-coordinates range from -99 to 100.

4.2 Draw/Select Cursor Movement

The Draw/Select cursor (turtle) is manipulated in DiagramEdit with four of the TURBO Turtlegraphics procedures. Those procedures are 'ShowTurtle', 'HideTurtle', 'SetHeading(Angle)' and 'SetPosition(X,Y)'. The 'ShowTurtle' procedure displays the turtle as a small triangle. The turtle is initially 'hidden', so this procedure is the first one called. When the turtle is displayed, the 'HideTurtle' procedure is called to hide the turtle so that it is not shown on the screen. The procedure 'SetHeading(Angle)' turns the turtle to the angle specified by the integer expression 'Angle'. 0 is upwards, and increasing angles represent clockwise rotation. If 'Angle' is not in the range 0..359, it is converted into a number in that range. Four integer constants are pre-defined to

easily turn the turtle in the four main directions: North = 0 (up), East = 90 (right), South = 180 (down), and West = 270 (left). The turtle is moved by calling the procedure 'SetPosition(X,Y)' with the coordinates given by the integer expressions 'X' and 'Y'.

The Draw/Select cursor is moved in the diagram area using the four cursor arrow keys and the 'Home', 'PgUp', 'PgDn' and 'End' keys. The cursor arrow keys turn the turtle in the four main directions described above. The other keys turn the turtle in four directions specified by user-defined constants in DiagramEdit: NorthEast = 45 (PgUp), SouthEast = 135 (PgDn), SouthWest = 225 (End), and NorthWest = 315 (Home).

The turtle is initially pointing North and is set to move one character space or 8 pixels from the position (-4,8). When a cursor movement key is pressed, the turtle is moved by (1) adding or subtracting 8 from the appropriate turtle coordinate(s), (2) calling the 'SetHeading(Angle)' procedure to face the turtle in the direction specified by the movement key, and (3) calling the 'SetPosition(X,Y)' procedure with the new turtle position. For example, if the turtle position is (-12,8) and the right arrow key is

pressed, the 'SetHeading(Angle)' procedure is called with 'Angle' = 90 to turn the turtle to the right and the 'SetPosition(X,Y)' procedure is called to position the turtle at (-4,8).

The value 8 that is added or subtracted from the turtle coordinate(s) is changed by pressing the 'Tab' key to add character spaces or by pressing the 'Shift'-'Tab' keys to subtract character spaces. For example, if the 'Tab' key is pressed two times, then the Draw/Select cursor will move 24 pixels or 3 character spaces.

Note that if the number of character spaces that the Draw/Select cursor is to be moved exceeds the boundary of the diagram area, DiagramEdit positions the turtle to appear on the opposite side. This is done by calling the 'SetPosition(X,Y)' procedure with the appropriate turtle coordinates.

If a connector/line is being drawn as the Draw/Select cursor moves, then a line trails the cursor. In the case where the connector/line is solid, the 'Draw(X1,Y1,X2,Y2,Color)' procedure is called to draw a line from the old position of the cursor to the new position. On the other hand, if the connector/line is dashed, then the 'Draw(X1,Y1,X2,Y2,Color)'

procedure is called to draw a line(s) from the old position of the cursor to the new position every 4 pixels.

4.3 The Symbol Table

The symbol table for a diagram contains the text, connector and line symbol records in the order in which they were drawn. A record contains the minimum information necessary to draw a symbol. Symbol records are read into memory from a diagram file on disk when a diagram is loaded and written to a diagram file on disk from memory when a diagram is saved. The data structures used are described below.

MS -- A constant equal to the maximum number of symbols that can be created in a diagram.

MC -- A constant equal to the maximum number of characters that can be entered in a text symbol.

Symbol(MS) -- An integer array containing the symbol number of either a text (1-12), connector (21-24) or line (25-26) symbol.

RowOrFrom(MS) -- An integer array containing either the row position of the upper left-most character if the symbol is a text symbol, the index of the symbol in the symbol table on

the 'From' side of a connector symbol, or a zero if the symbol is a line symbol.

ColOrTo(MS) -- An integer array containing either the column position of the upper left-most character if the symbol is a text symbol, the index of the symbol in the symbol table on the 'To' side of a connector symbol, or a zero if the symbol is a line symbol.

Info(MS,MC) -- An integer array containing the text of a text symbol in ASCII integer form or the graphics coordinates of a connector or line symbol.

The example diagram in Figure 17 contains text, connector and line symbols. The information in the diagram file on disk corresponding to the example diagram is shown below.

```
1 3 3 65 110 121 32 116 101 120 116 32 116 121 112 101 100
32 105 110 116 111 13 97 32 116 101 120 116 32 115 121 109
98 111 108 46 -1
3 12 34 65 110 121 32 116 101 120 116 32 116 121 112 101 100
13 105 110 116 111 32 97 110 111 116 104 101 114 13 116 101
120 116 32 115 121 109 98 111 108 46 -1
21 1 2 83 34 83 35 83 99 261 99 -1
26 0 0 0 0 187 147 283 147 283 163 323 163 323 147 419 147
-1
```

All of the above records are of type integer and are terminated with a -1. The following paragraphs describe the

fields within a record and the data structures in which they are stored.

The first integer of each record is either the symbol number of a text symbol (1-12), a connector symbol (21-24), or a line symbol (25-26). Each symbol number is stored in the array 'Symbol'. The second integer of each record is either the row number of the upper left-most character of a text symbol, the index of the symbol in the symbol table on the 'From' side of a connector symbol, or a zero if the symbol is a line. The second integer is stored in the array 'RowOrFrom'. The third integer of each record is similar to the second. It is either the column number of the upper left-most character of a text symbol, the index of the symbol in the symbol table on the 'To' side of a connector symbol, or a zero if the symbol is a line. The third integer is stored in the array 'ColOrTo'. The remaining integers in each record are the characters of a text symbol in ASCII form, or the pairs of graphics coordinates of a connector or line symbol. The integers are stored in the array 'Info'.

The text symbol in the upper left corner of the diagram in Figure 17 corresponds to the first record in the symbol

table. This symbol is a text symbol of type 1 and the upper left-most character of the text is positioned on row 3 and column 3. These fields are followed by the text in ASCII integer form.

The connector drawn between the two text symbols in the diagram corresponds to the third record in the symbol table. This connector has a symbol type of 21. The next two integers indicate that the connector is drawn from the first to the second symbol in the table. Since the connector has two line segments, the parameters 83,34,83,99,1 and 83,99,26,99,1 are used in the 'Draw(X1,Y1,X2,Y2,Color)' procedure to draw the connector on the screen. However, if the symbol type had been 23 or 24, the parameters for the first procedure call to 'Draw(X1,Y1,X2,Y2,Color)' would be 83,35,83,99,1 instead. This is a result of the difference of the size of a head not drawn on the 'From' side of undirectional connectors (21-22) and drawn on the 'From' side of bidirectional connectors (23-24). Thus, for unidirectional connectors, drawing starts with the first pair of coordinates and for bidirectional connectors, drawing starts with the second pair of coordinates. Both pairs of coordinates are saved in case the symbol type of the connector is changed.

The record for a line is similar to a connector except the second through the fifth integers are zeros. This is evident from the line in the diagram which corresponds to the fourth record in the symbol table. This line has a symbol type of 25. Since lines do not connect symbols, the second and third integers are set to zero to indicate that they are not used. The fourth and fifth integers are also set to zero since heads are not drawn on the ends of lines. Therefore, the drawing of lines always starts with the second pair of coordinates.

4.4 Text Symbol Drawing

All of the text symbol types available in DiagramEdit are composed of different combinations of lines based on a rectangle. Graphics position (H1,V1) is in the upper left corner and (H2,V2) is in the lower right corner of the rectangle. The coordinates in these graphics positions as well as their midpoints, H3 and V3, are used to draw the different symbol types. The graphics coordinates used to draw the twelve text symbol types available in DiagramEdit are shown in Figures 18-21. Before drawing a symbol around the text, the graphics coordinates of a symbol are computed with the following equations:

```
H1 = ColOrTo * 8 - 9
V1 = RowOrFrom * 8 - 9
H2 = MaxCol * 8 - 1
V2 = MaxRow * 8 - 1
H3 = (H1 + H2) Div 2
V3 = (V1 + V2) Div 2
```

'RowOrFrom' and 'ColOrTo' contain the row-column position of the upper left-most character of the text. 'MaxCol' contains the number of the right-most column on the longest line of text and 'MaxRow' contains the row number of the last line of text. The 'Div' operator is used to perform integer division.

Based on the symbol type chosen, DiagramEdit calls the TURBO Pascal 'Draw(X1,Y1,X2,Y2,Color)' procedure to draw the lines of a symbol. For example, to draw symbol type 2 shown in Figure 19, the following procedure calls are made:

```
Draw(H1,V1,H2,V1,1),
Draw(H2,V1,H2+8,V1+V3,1),
Draw(H2+8,V1+V3,H2,V2,1),
Draw(H2,V2,H1,V2,1), and
Draw(H1,V2,H1,V1,1).
```

4.5 The Text Editor

A text symbol is selected for editing when the 'F1' key is pressed. Figure 6 shows the keys available in the text editor. If the Draw/Select cursor is positioned in a clear area of the screen, then a new text symbol is created. On the other hand, an existing symbol is modified when the cursor is positioned anywhere on its text. DiagramEdit determines where the cursor is positioned by comparing the graphics coordinates of each text symbol with those of the cursor. If the X-coordinate of the cursor lies within 'H1' and 'H2' and the Y-coordinate lies within 'V1' and 'V2' of a text symbol, then that symbol is modified. Otherwise, a new symbol is created.

The text editor is implemented as a large character array that encompasses the entire screen boundary. Text is manipulated with pointers that contain the beginning character row and column numbers, the ending character row number, and each ending character column number of the text. Execution of the text editor is the same regardless of whether a text symbol is new or already exists. Only the initialization is different.

To see how the pointers are used, the Up arrow key will be used as an example. (The other Text cursor and symbol movement keys are implemented in a similar way.) The Up arrow key moves the Text cursor up one row or moves the text symbol up the number of rows specified by 'S#' on the bottom line of the screen. If the cursor is on the top line of the text, then the symbol will be moved up. Otherwise, the cursor will be moved up one row and remain in the same column.

To move the Text cursor up one row, it is first deleted by printing the character at its row-column position. Second, one is subtracted from the row number of the cursor. The column number is then compared to the ending character column number of the new row. If the column number is greater, then it is set to the ending character column number of the new row. The cursor is then redrawn at the new row-column position.

To move the text up the number of rows specified by 'S#', the number of character spaces to be skipped is subtracted from the beginning character row number. If the result is greater than or equal to one, then the text is moved up in the following way. First, the symbol surrounding the text

is erased by drawing it in black. Second, the number of rows to be skipped is subtracted from the row number of the Text cursor and from the beginning and ending character row numbers. Third, the rows of characters are copied and printed in their new position, and the remaining rows of characters are set to null and blanks are printed. Fourth, the graphics coordinates of the symbol are computed and the symbol is redrawn. Finally, the cursor is redrawn at its new row-column position.

4.6 Connector/Line Symbol Drawing

A connector/line symbol is started by pressing the 'F2' key. If the symbol to be drawn is a line, then zero is stored in 'RowOrFrom'. Otherwise, DiagramEdit determines which symbol the connector to be drawn is coming from. When the symbol being checked is a text symbol, the graphics coordinates of the symbol are compared with those of the Draw/Select cursor. If the X-coordinate of the cursor lies within 'H1' and 'H2' and the Y-coordinate lies within 'V1' and 'V2' of a text symbol, then the index of that symbol in the symbol table is stored in 'RowOrFrom'. On the other hand, when the symbol being checked is a connector or line symbol, then a line segment intersection procedure is called to determine

if the cursor lies on a connector or line. The details of this procedure are discussed in the next section.

After storing a value in 'RowOrFrom', the graphics coordinates of the Draw/Select cursor are stored in 'Info'. The coordinates are also stored each time the cursor changes direction. When the 'F2' key is pressed again to end a connector/line, the same procedure described above occurs to store a value in 'ColOrTo'. Also, the graphics coordinates of the cursor are stored in 'Info', and if necessary, clipping occurs and a head(s) is drawn. Clipping is performed with a line segment intersection procedure described below.

4.7 A Line Segment Intersection Procedure

A line segment intersection procedure is called in DiagramEdit in two different cases. Each case is described below.

In the first case the procedure is called during clipping to determine which line segment of a text symbol a connector intersects (see Figure 15). Each line segment of the text symbol is checked against the line segment of the connector for intersection. When a point of intersection is found,

clipping occurs when the line segment of a connector is redrawn to the point of intersection (see Figure 16).

In the second case the procedure is called to determine if the Draw/Select cursor lies on a segment of a connector/line. Before calling the procedure, DiagramEdit determines whether the graphics coordinates of the cursor lie within the endpoints of the line segment of a connector/line. When this is the case, the procedure is called to further determine if the cursor lies within the endpoints of a line segment of a connector/line.

The cursor is only a graphics position on the screen and the procedure excludes the case where the candidate line segments are colinear. To circumvent this problem, three pixels are added to and subtracted from the X- and Y-coordinates of the graphics position to form two new lines. The coordinates of the endpoints of each line and those of the line segment of a connector/line are used as parameters to the procedure to determine if they intersect.

The line segment intersection procedure below solves the problem: given two arbitrary line segments in the plane, determine (1) if the line segments intersect and (2) the point of intersection, if any. The procedure is

computationally simple and is applicable even if one of the line segments is vertical. It is important to note that this procedure does not require that special consideration be given to the case where one (or both) line segment is vertical. Most procedures require that vertical line segments be given additional consideration (e.g., slope-intercept form). The additional effort required for this case is clearly to be avoided. The execution time of the procedure Line_Segment_Intersection is on the order of N log N and excludes the case where the candidate line segments are colinear.

```
Procedure Line_Segment_Intersection
Input line_segment_L1:  (X1,Y1),(X2,Y2)
Input line_segment_L2:  (X3,Y3),(X4,Y4)
Output intersection_point: (Xint,Yint)
Output Intersect
S1:   Set Intersect to FALSE
S2:   Compute:  X1 = X1 * 0.417
                Y1 = Y1 * 2.4
                X2 = X2 * 0.417
                Y2 = Y2 * 2.4
                X3 = X3 * 0.417
                Y3 = Y3 * 2.4
```

```
                X4 = X4 * 0.417
                Y4 = Y4 * 2.4
S3:   Compute:  A1 = X2 - X1
                A2 = Y2 - Y1
                A3 = X3 - X4
                A4 = Y3 - Y4
S4:   Compute:  Delta = A1 * A4 - A2 * A3
S5:   If Delta is zero, exit
S6:   Compute:  A5 = X3 - X1
                A6 = Y3 - Y1
S7:   Compute:  Lambda = (A5 * A4 - A3 * A6) / Delta
S8:   If Lambda is outside [0,1], exit.
S9:   Compute:  Mu = (A1 * A6 - A5 * A2) / Delta
S10:  If Mu is outside [0,1], exit.
S11:  Compute intersection_point:
      Xint = Round(((1.0 - Mu) * X3 + Mu * X4) / 0.417)
      Yint = Round(((1.0 - Mu) * Y3 + Mu * Y4) / 2.4)
      Set Intersect to TRUE
end Line_Segment_Intersection
```

The method developed to solve the line segment intersection problem is based on a vector scheme [6]. The input parameters to the procedure are the coordinates of the first line segment (X1,Y1), (X2,Y2), and the coordinates of the

second line segment (X3,Y3), (X4,Y4). The output parameters from the procedure are the point of intersection (Xint,Yint) and the flag 'Intersect'. If 'Intersect' has the value TRUE, then the point of intersection is valid.

Note the use of the constants 2.4 (12/5) and 0.417 (5/12). Since the IBM PC screen is not symmetric, the aspect ratio is used to determine the spacing of horizontal, vertical and diagonal points. This accounts for the use of the constants that determine how many horizontal points (0.417) are equal to how many vertical points (2.4). At the beginning of the procedure, both X-coordinates of each line segment are multiplied by 0.417 and both Y-coordinates are multiplied by 2.4 to convert them into the scaled coordinates which are used in the calculations. A similar action occurs at the end of the procedure when X- and Y-coordinates of the point of intersection are divided by 0.417 and 2.4, respectively, to convert them back to graphics coordinates. The 'Round' function is used because graphics coordinates are integers.

Chapter 5. Applications of DiagramEdit

DiagramEdit has been applied in several different ways to date. The tool has been used at NSWC to aid in the drawing and modification of diagrams used in system and software design. In addition, it has also been used as part of a Diagrammatic Language System (DLS) developed at NSWC to interface between a user and a Diagrammatic Language Translator (DLT). The DLT is based on a diagrammatic language and performs as a compiler to detect and report errors in diagrams and to convert sets of diagrams into Pascal. The code produced by the system can then be compiled and executed. The details of these applications are presented below.

5.1 System Design

An example of the application of DiagramEdit in the design of a system appears in a paper written by Richard P. Cullen of NSWC entitled "Combat Systems Architecture Evolution to a 2010 Goal." In this paper Cullen uses DiagramEdit to illustrate series of evolutionary steps that can lead to the realization of a form of a Combat Systems Architecture in

the 2010 time frame. The diagrams in this paper are shown in Figures 22-29. These diagrams are examples of the many types of diagrams that can be drawn with DiagramEdit.

5.2 Software Design

An example of the use of DiagramEdit in software design took place in an experiment conducted at Mary Washington College (MWC). The experiment was conducted to determine if using a computer aided diagrammatic approach to design offered advantages to undergraduate students in the several stages of preparing and implementing programs. Statistically significant results were obtained from this experiment which favored the use of a computer aided diagrammatic design method. The greatest benefit was realized in the later stages of coding and implementation rather than in the design stage itself. Overall, the diagrammatic classes completed assignments in 26% less time than the classes using a textual approach without any measurable loss of quality. The research was a joint effort of NSWC and MWC. The results of the experiment are contained in [4].

5.3 Diagrammatic Language Translation

In the following application, DiagramEdit is used as part of a DLS to interface between a user and a DLT. The interface to the DLT is through the diagram files stored on disk. Based on a diagrammatic language, the DLT processes the diagram files and performs as a compiler to detect and report errors in diagrams and to convert sets of diagrams into Pascal. The code produced by the system can then be compiled and executed. Figures 30-36 show an example of a valid set of diagrams in the diagrammatic language that solve a projectile motion problem.

The DLS arose from an effort initiated at NSWC to explore the potential of diagrammatic language forms to reduce the effort involved in the development and life support of large, embedded computer programs. As a result of this project, a serial diagrammatic language was devised that differed radically from conventional languages. Statistics from a number of test cases showed an average of a four to one advantage over conventional language [4]. The lexicon and number of syntactic elements is also greatly reduced. Perhaps most important, however, is that diagrammatic form is based on, and very close to, diagrams people commonly use to design computer programs.

Experimentation and paper analysis conducted as part of the independent research effort demonstrated the feasibility of implementing the serial language using standard graphic monitors, interpreting the diagrams and translating them into executable code. The work progressed to the point where a transition to independent exploratory development was warranted to determine the full requirements and methods for translating diagrammatic languages into executable code, and also to develop and demonstrate the capability in a development model.

The model that was developed is the DLS. The DLS was designed to provide a user with a completely diagrammatic working environment with no exposure to conventional text-based languages. The design provides the capability to create and modify diagrams (functions of DiagramEdit), manage diagram files, translate and debug single or multiple programs, generate executable code, and control program execution. Diagnostics at all stages are presented in the original diagram context. The DLS runs on standard IBM PCs under DOS and is written entirely in TURBO Pascal. DiagramEdit can be invoked from the DLS or from DOS.

Other than limiting the symbols available to a user and ensuring that a drawing is mechanically correct, there are no language specific functions in DiagramEdit. The tool is applicable to a wide variety of languages. Diagrams can be drawn with a wide degree of freedom in a manner very analogous to pencil and paper.

Chapter 6. Future Work

Several new functions are recommended for DiagramEdit in the future. Its modular design and flexible data structures allow new functions to be added with relative ease. The recommended functions are described below.

When a diagram is loaded or saved, only the currently logged drive/directory is accessed. This is also the case when DiagramEdit program files are activated with the 'Chain' procedure. A function is needed on the main menu that would allow the logged drive/directory to be changed so that diagram files can be loaded from a different drive/directory. In addition, when the load from list option is chosen from the main menu, the currently logged drive should be displayed in the upper left corner of the screen with the diagram file listing.

Since the option to load a diagram from a block is available, the option to save a file from a block should also be provided in DiagramEdit. This function would be particularly useful when the highest-level diagram in a hierarchy of diagrams is saved. A name could be created from the text of the symbol chosen.

An installation program could be written to allow a user to customize the keys used for the text editor in DiagramEdit. This would allow users unfamiliar with the TURBO Pascal Editor to redefine the keys.

Another function that could be included in the installation program is the creation of user-defined text symbol types. A user could view diagrams showing the dimensions of the twelve text symbols already available and enter the dimensions of a new text symbol. DiagramEdit would draw the new symbol on the screen, and if it was acceptable to a user, it would be included with the other symbols. By the same token, the capability should exist for deleting any of the twelve text symbols or any new text symbols created.

With the text editor, text symbols can currently be moved horizontally and vertically with the cursor arrow keys. The 'Home', 'End', 'PgUp', and 'PgDn' keys should also be used to move the symbol diagonally according to the 'S#' indicator.

Regarding movement, in many cases a user wants to move a whole diagram or just a group of symbols. If the whole diagram is to be moved, a function key could be pressed, and the cursor movement keys would be used to move the diagram

according to the 'S#' indicator. If a group of symbols is to be moved, a function key would be pressed when the Draw/Select cursor is positioned in the upper left and lower right corners of the group. DiagramEdit would then draw a box around the group of symbols and the cursor would be moved to the new position. A function key would be pressed again to complete the move. Only symbols appearing completely within the box would be moved. This same procedure could be used to copy a group of symbols.

Backtracking is a function that would be useful when drawing connectors or lines. Many times a user draws a line segment too long and must abort the drawing of a whole connector/line. Backtracking would enable a user to move the Draw/Select cursor back over the line segment, erasing the segment as it moves. Erasing would continue until the end of the line segment is reached. If the direction of the Draw/Select cursor changes or the complete line segment is erased, then drawing would continue.

Converting the DiagramEdit program code to execute under TURBO Pascal Version 5.0 would offer several advantages over Version 3.0. Some of the advantages are faster compilation

speed, more efficient code generation, and support for EGA and VGA compatible monitors.

The use of a bit mapping algorithm for clipping would eliminate the large amount of code needed to determine the point of intersection between connector line segments and the line segments of text symbols. A bit mapping algorithm would enable logic to be used instead of mathematics thereby increasing the speed of finding a point of intersection.

When the load from block option is used to load diagrams arranged in a hierarchy or on the same level, a method does not exist for travelling back through the levels from a lower-level diagram to a higher-level diagram. As a user travels back up to the highest level diagram, DiagramEdit could save the file names so that the load from list option positions the cursor to point to the corresponding file name in the hierarchy.

Instead of storing symbol information in separate arrays when a diagram is loaded, the information could be stored in one large array with another array containing the index of where the information starts for each symbol. This would eliminate wasted space in each of those separate arrays. In addition, text symbols could contain virtually as much text

and connectors/lines could have virtually as many segments as desired.

REFERENCES

1. Analytic Sciences Corporation: AdaGRAPH Reference Manual Version 1.2, 1987.

2. Analytic Sciences Corporation: AdaGRAPH User's Guide Version 1.2, 1987.

3. Borland International Inc.: Turbo Pascal Version 3.0 Reference Manual, 1985.

4. Cullen, R. P., E. C. Ackermann and W. R. Pope: "Computer Aided Program Design Experiments: Diagrammatic Versus Textual Method," prepared for publication in the proceedings of the 1989 Annual ACM Computer Conference.

5. HavenTree Software Limited: Interactive EasyFlow User's Manual, 1986.

6. Meyers, B. C. and C. F. Fennemore: "A Note On The Problem Of Intersecting Line Segments": NSWC TN 86-219, September, 1986.

7. Yourdon Inc.: Software Engineering Workbench Core Reference Manual, 1987.

8. Yourdon Inc.: Software Engineering Workbench User's Guide, 1987.

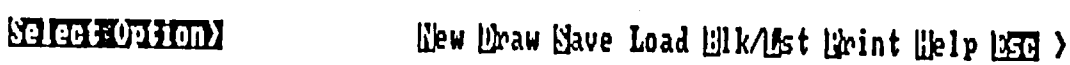


Figure 1. The Main Menu

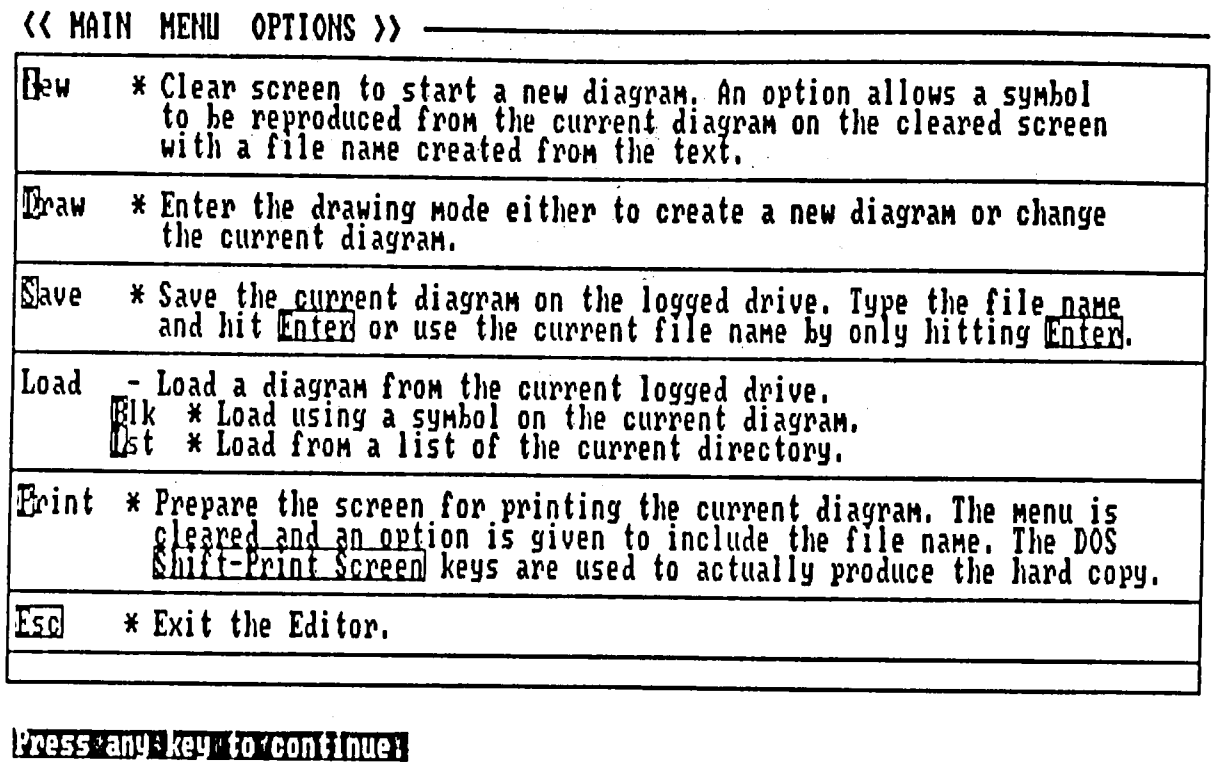


Figure 2. The Main Menu Help Screen

Figure 3. The Draw Display

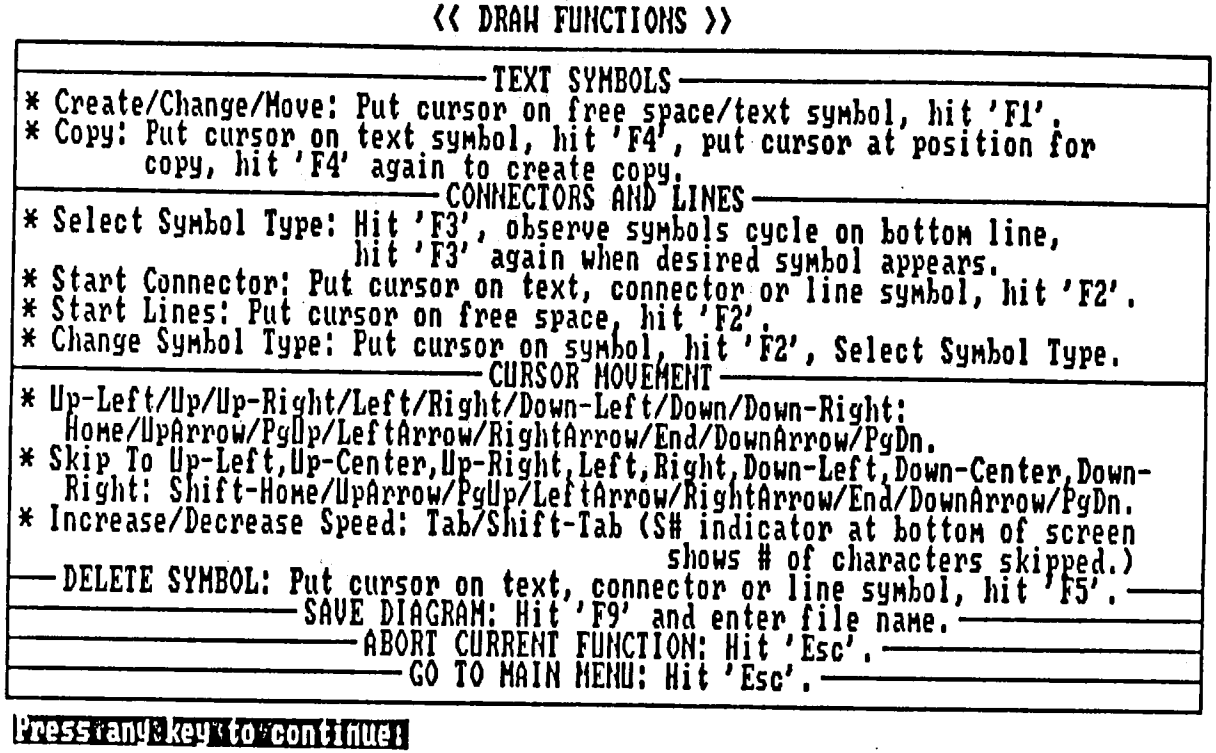


Figure 4. The Draw Display Help Screen

Edit Text Symbol F1 Help Insert —▪ R3 C3 S1 >

Figure 5. The Drawing of a Text Symbol

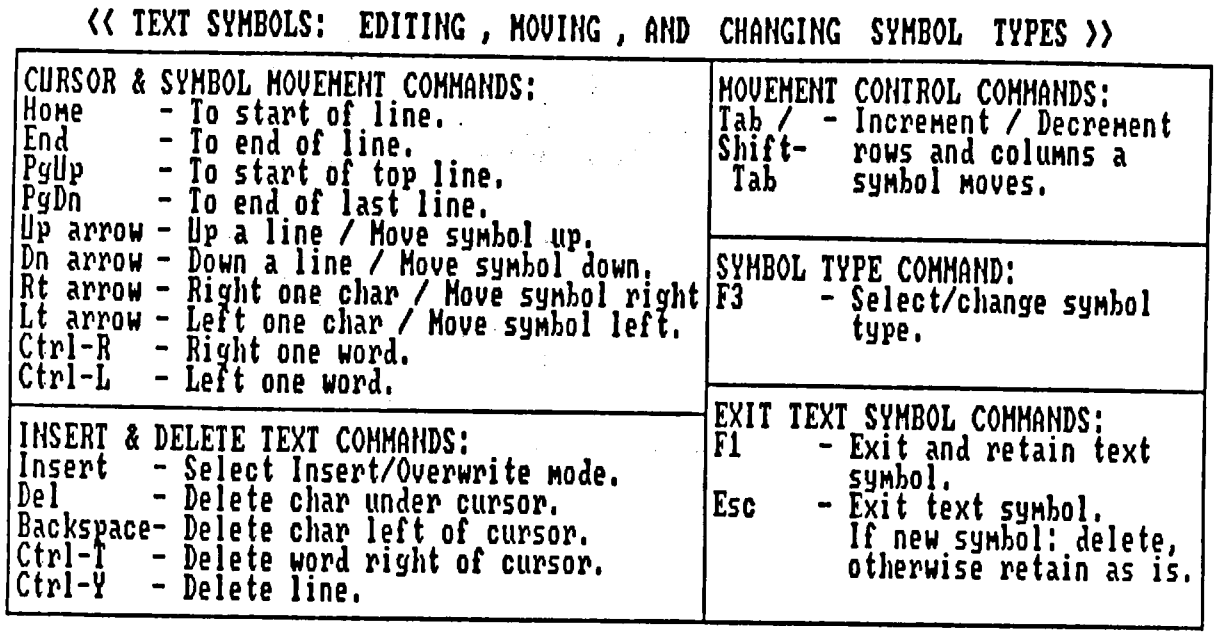
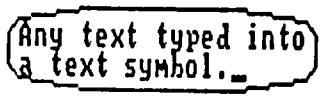


Figure 6. The Text Symbol Editor Help Screen



Edit a text symbol F1 Help Insert R4 C17 S1 >

Figure 7. The Entry/Modification of Text in a Text Symbol

Draw diagram Help R4 C17 S1

Figure 8. The Completion of a Text Symbol

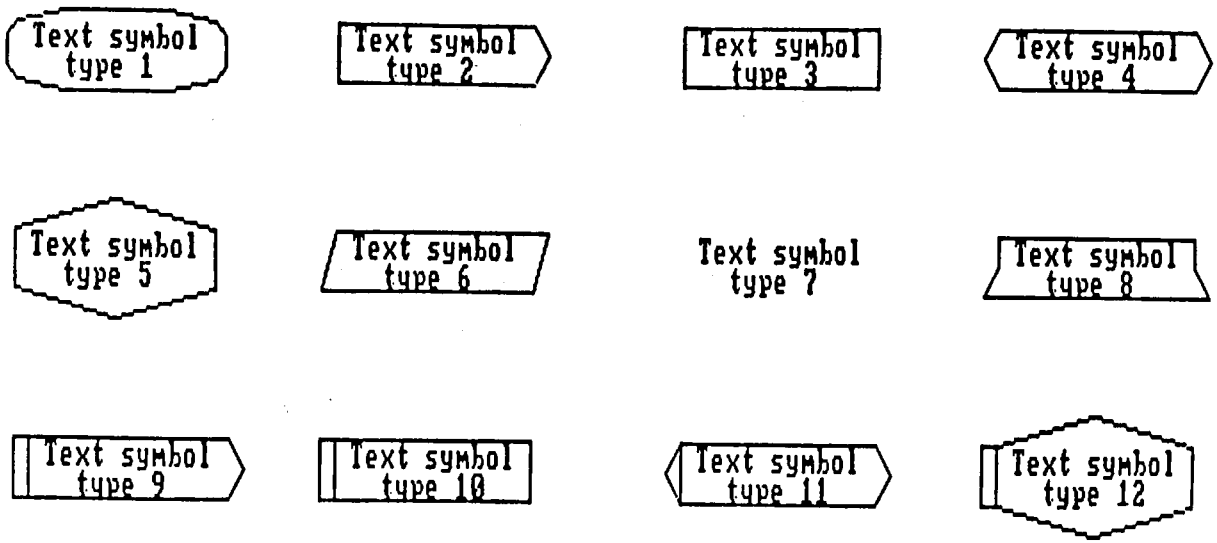


Figure 9. The Twelve Text Symbol Types

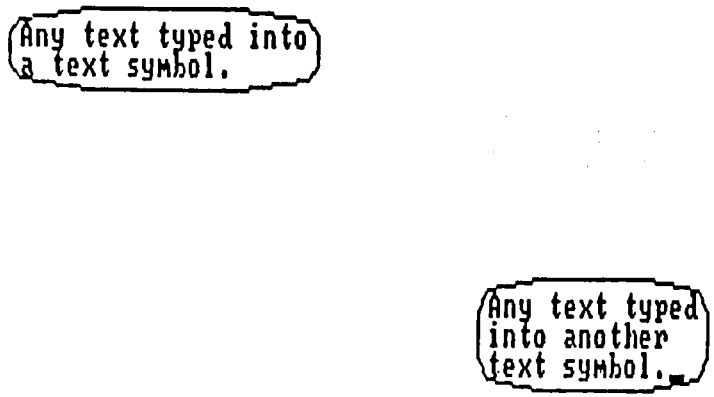


Figure 10. The Drawing of a Second Text Symbol

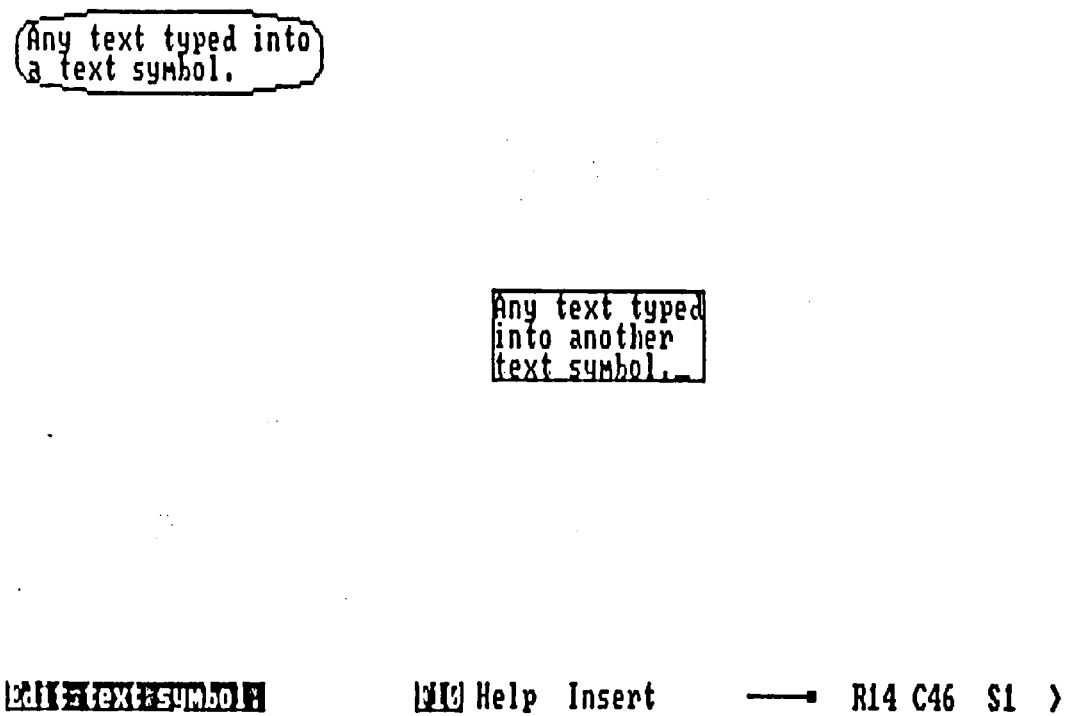


Figure 11. The Selection of a Text Symbol Type

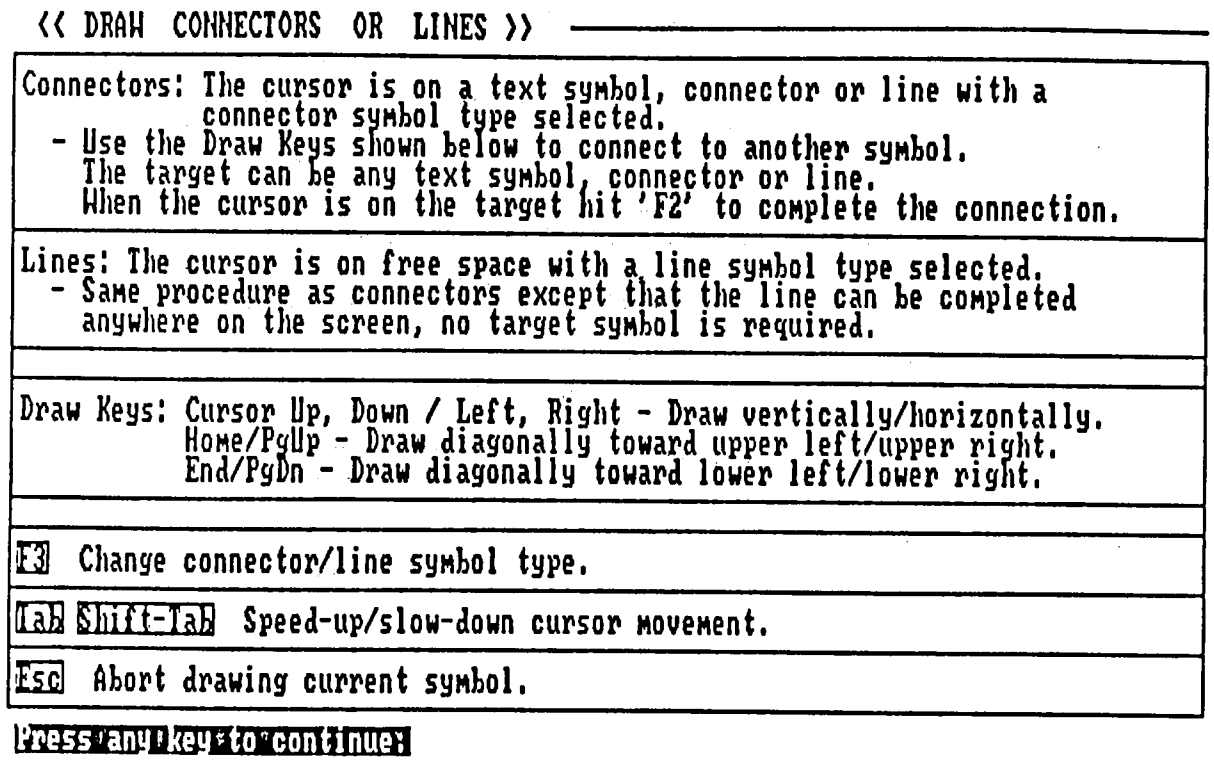


Figure 12. The Connector/Line Drawing Help Screen

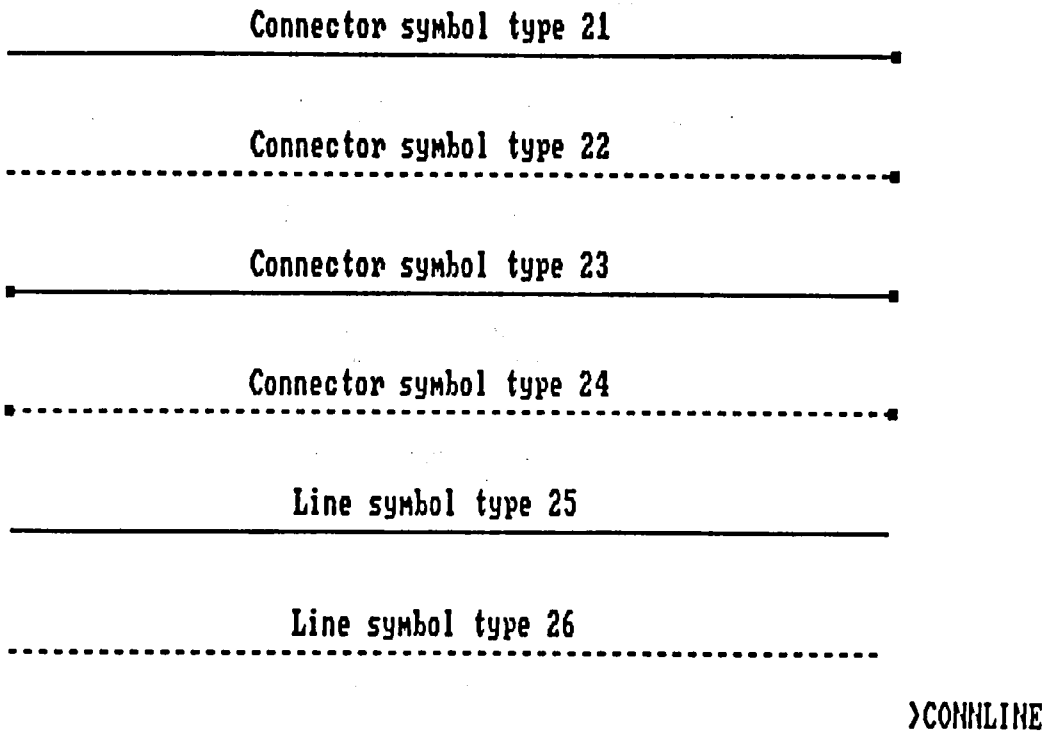


Figure 13. The Four Connector and Two Line Symbol Types

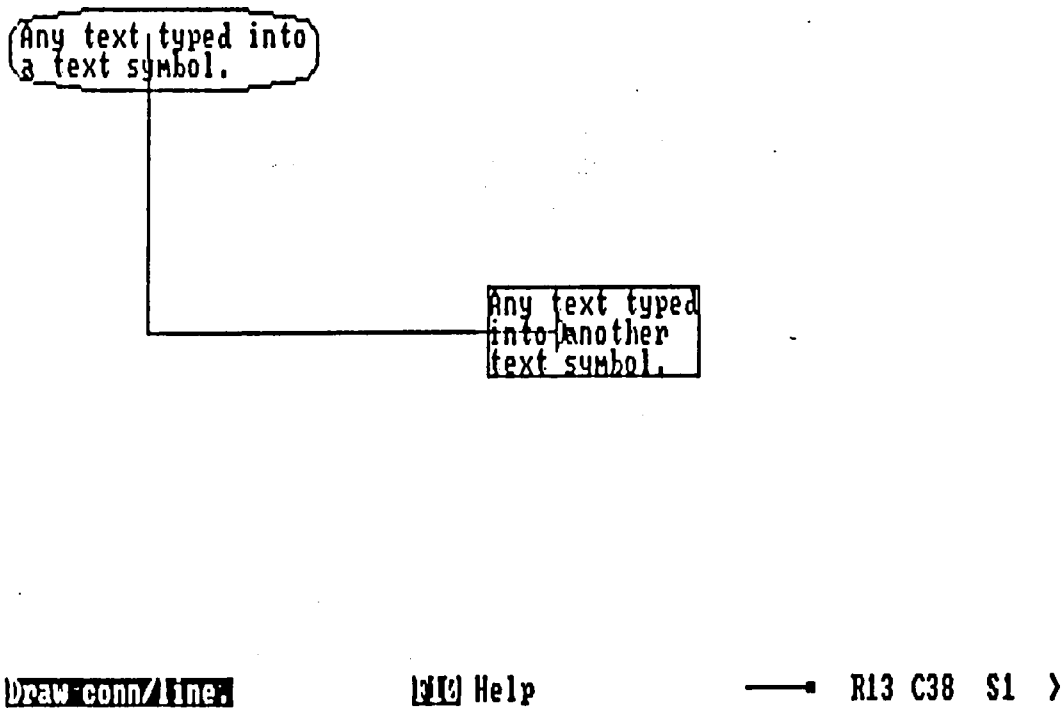
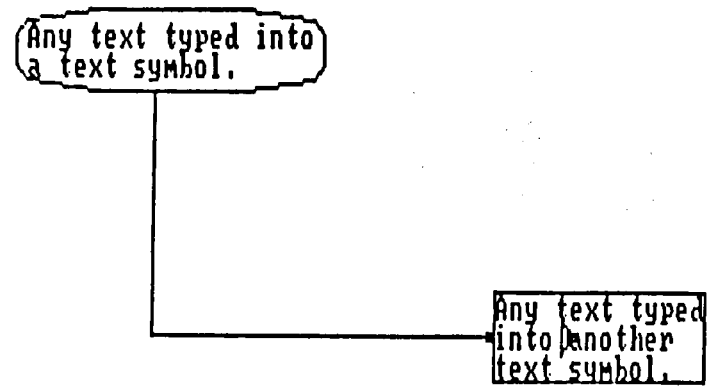


Figure 14. The Drawing of a Connector



Draw diagram. F10 Help R13 C38 S1

Figure 15. The Completion of a Connector

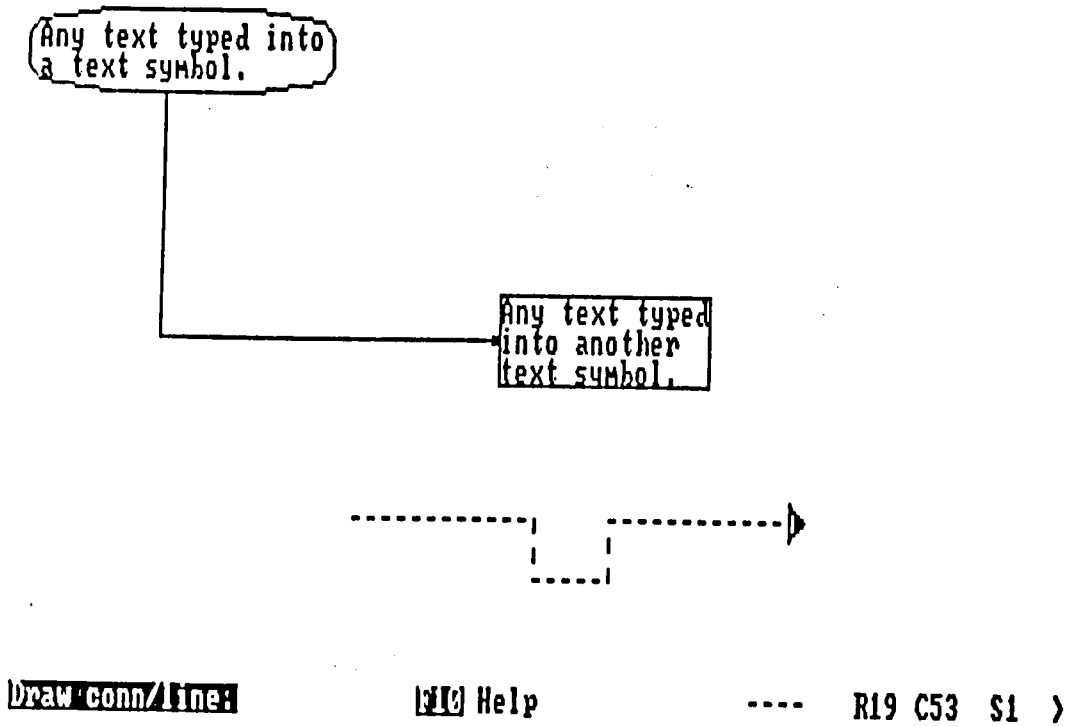
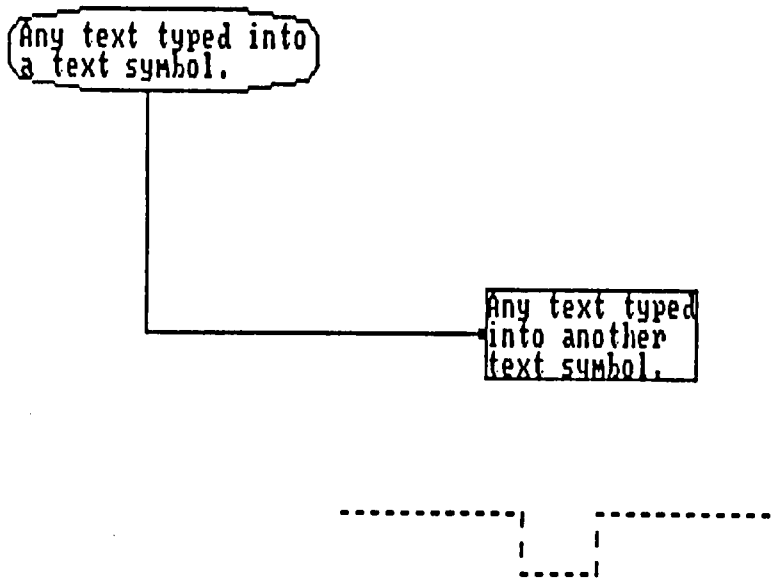
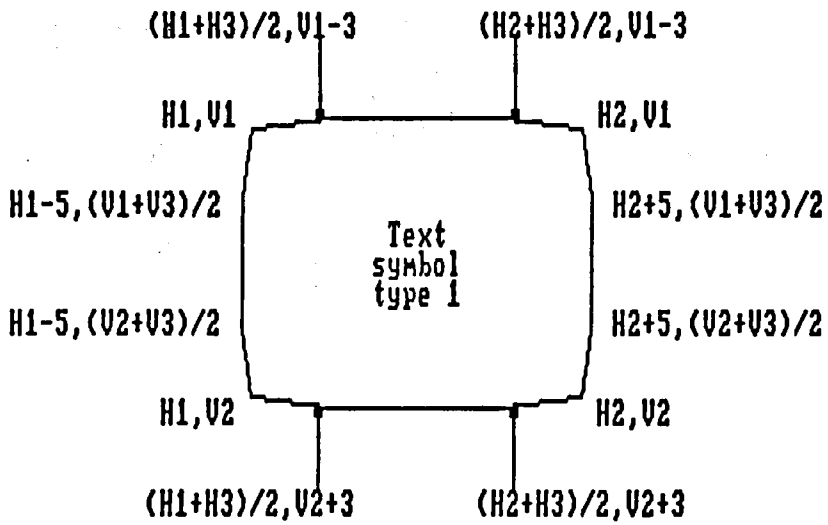


Figure 16. The Completion of a Line



>EXAMPLE

Figure 17. An Example Diagram



>SYMBOL1

Figure 18. The Graphics Coordinates of Text Symbol Type 1

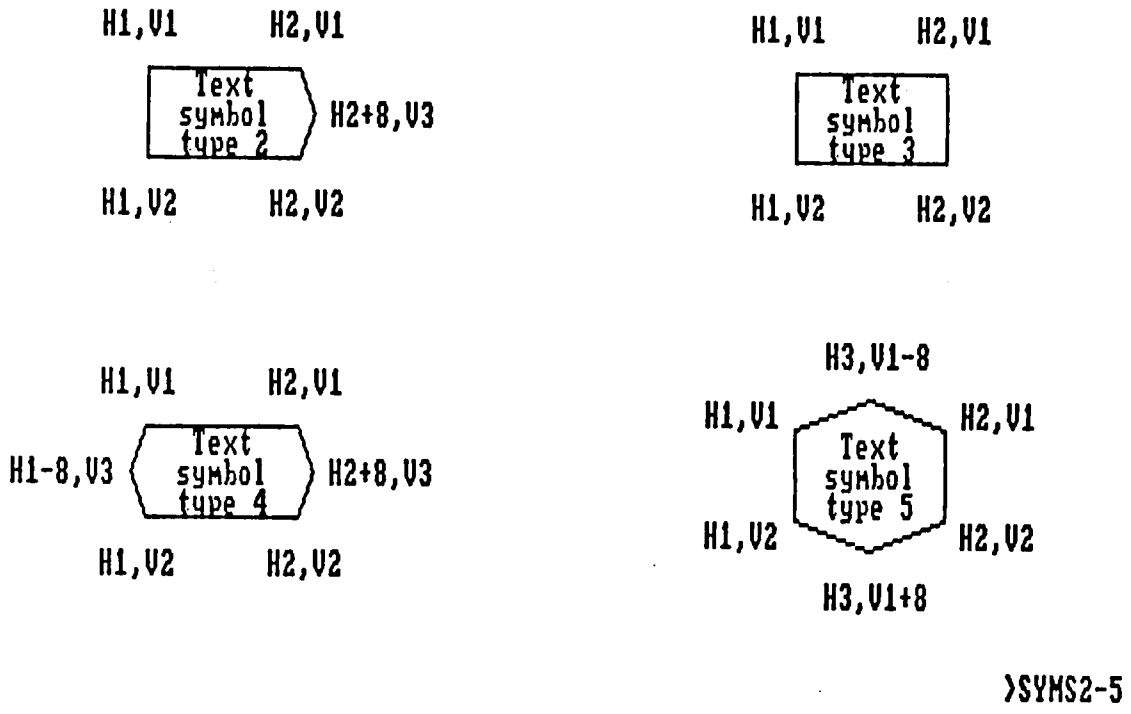


Figure 19. The Graphics Coordinates of Text Symbol Types 2-5

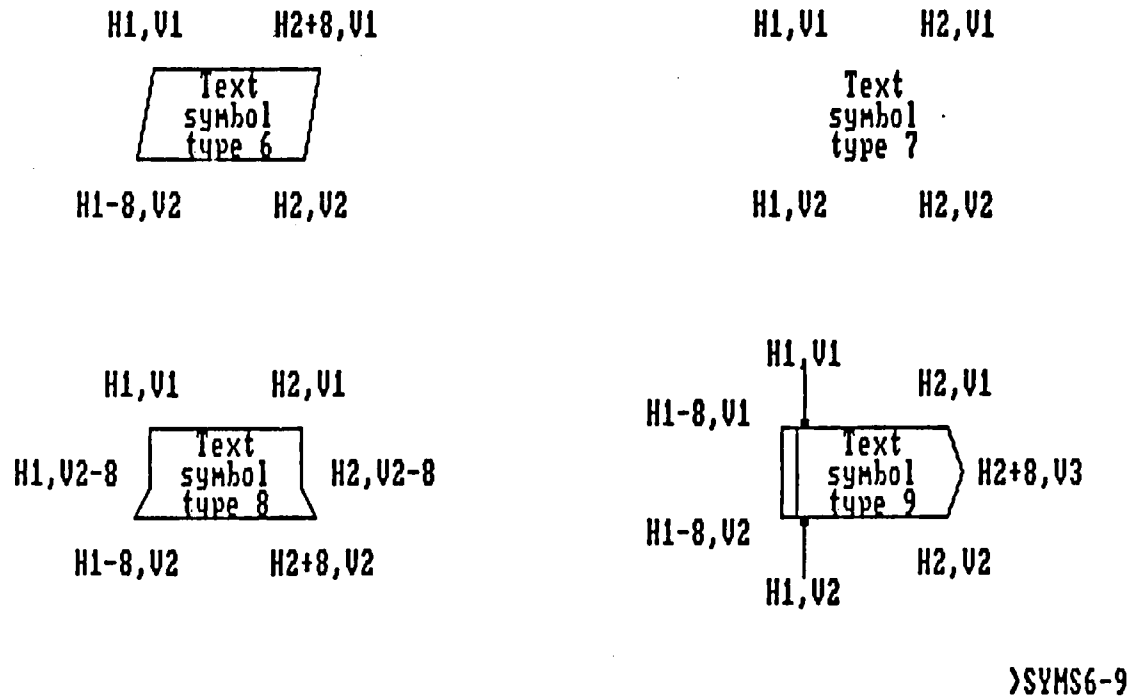


Figure 20. The Graphics Coordinates of Text Symbol Types 6-9

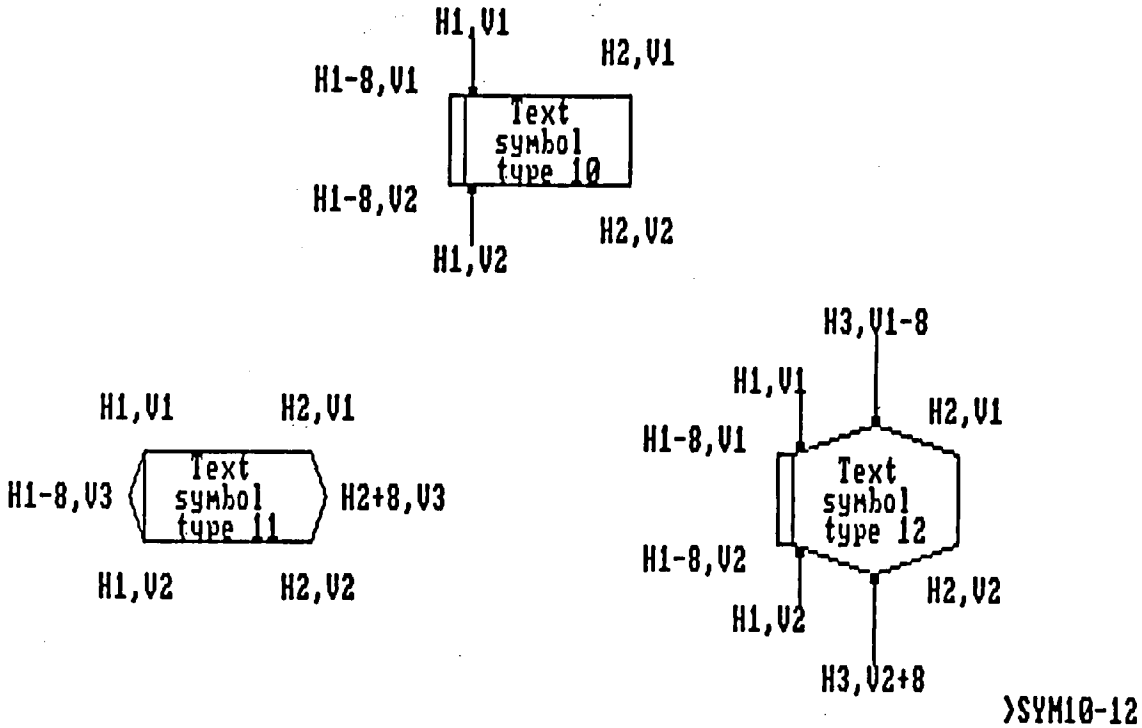
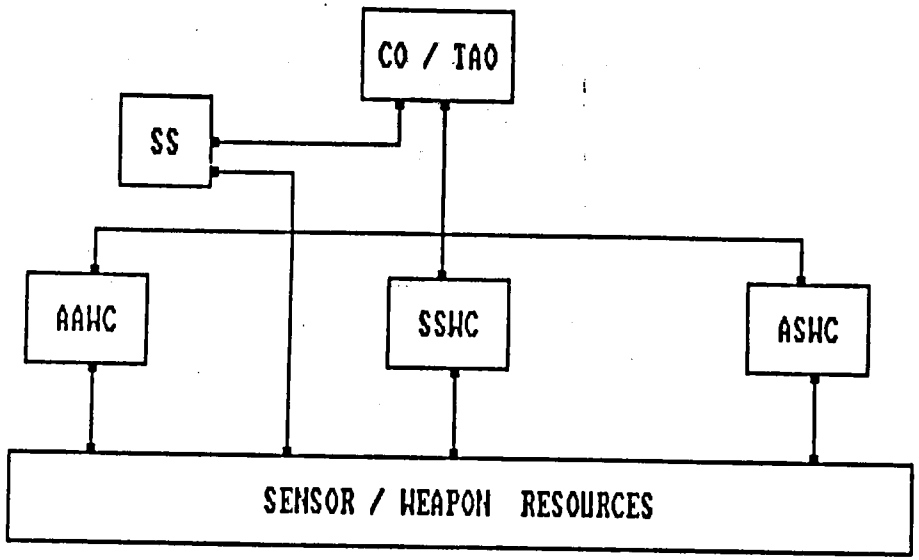


Figure 21. The Graphics Coordinates of Text Symbol Types 10-12

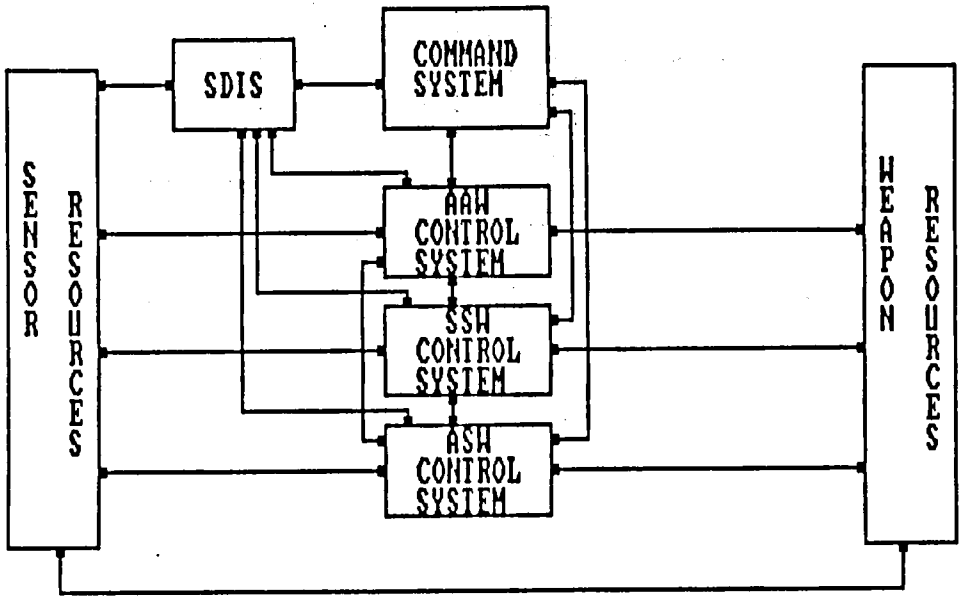
Figure 1 COMBAT SYSTEM ARCHITECTURE : BATTLE ORGANIZATION



>RAS-BL

Figure 22. Diagram #1 of a System Design

Figure 2 COMBAT SYSTEM ARCHITECTURE : CONFIGURATION



>RAS-CON

Figure 23. Diagram #2 of a System Design

Figure 3 CURRENT BATTLE ORGANIZATION SUPERSET

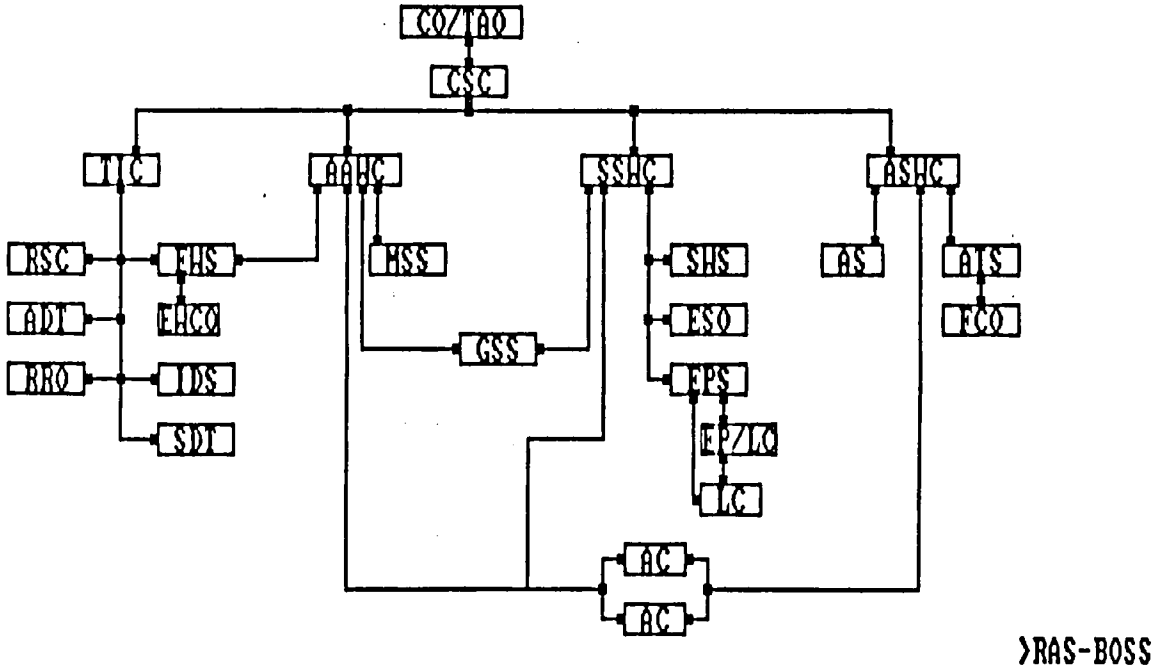


Figure 24. Diagram #3 of a System Design

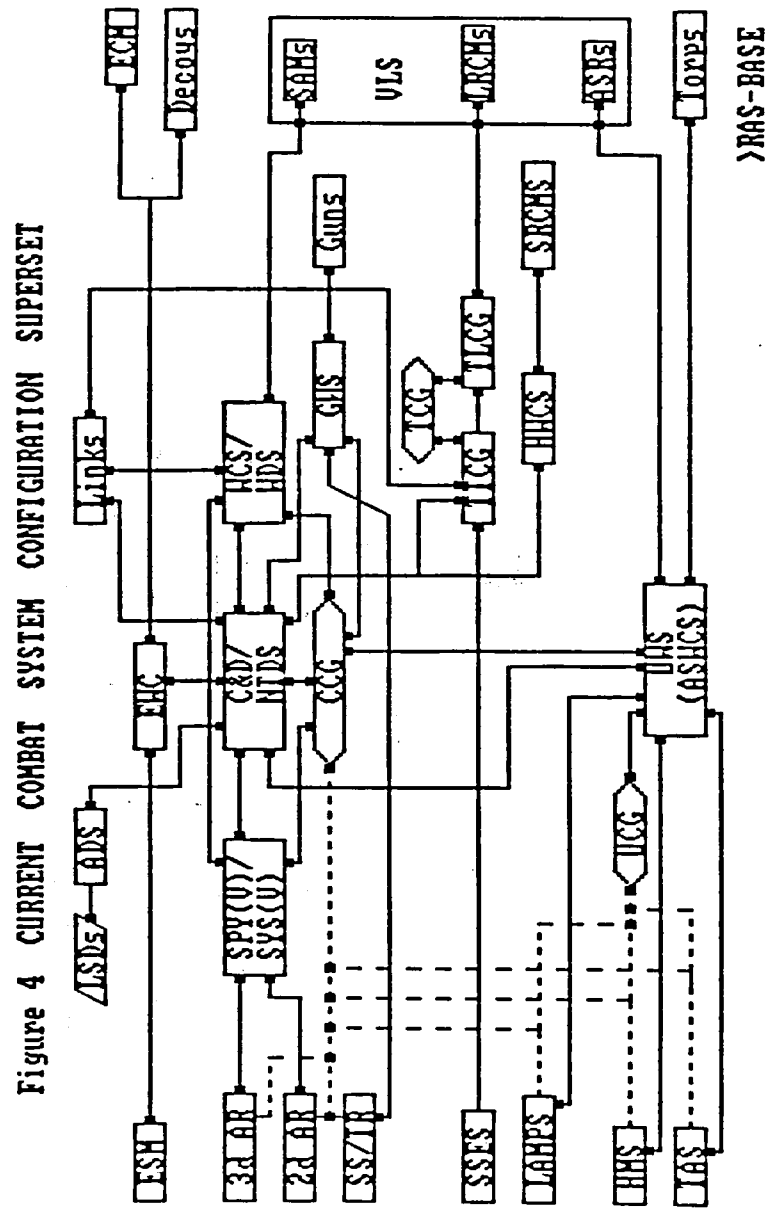


Figure 25. Diagram #4 of a System Design

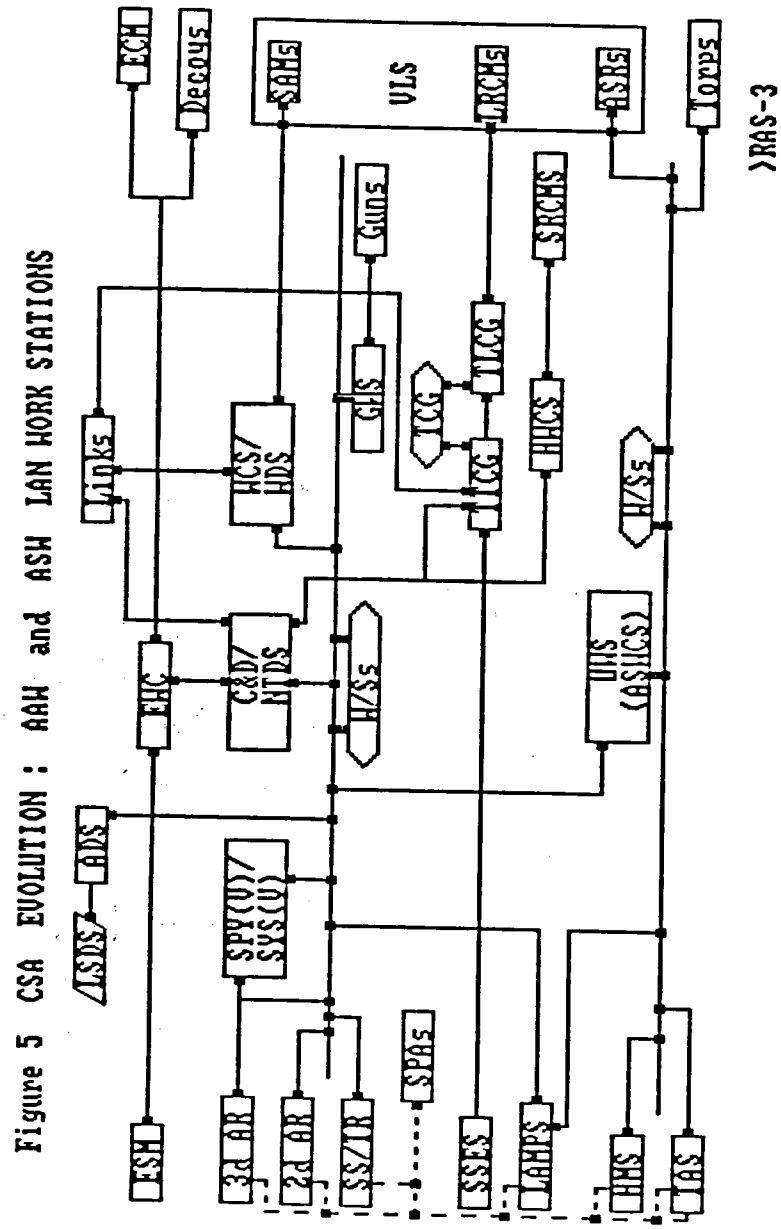


Figure 26. Diagram #5 of a System Design

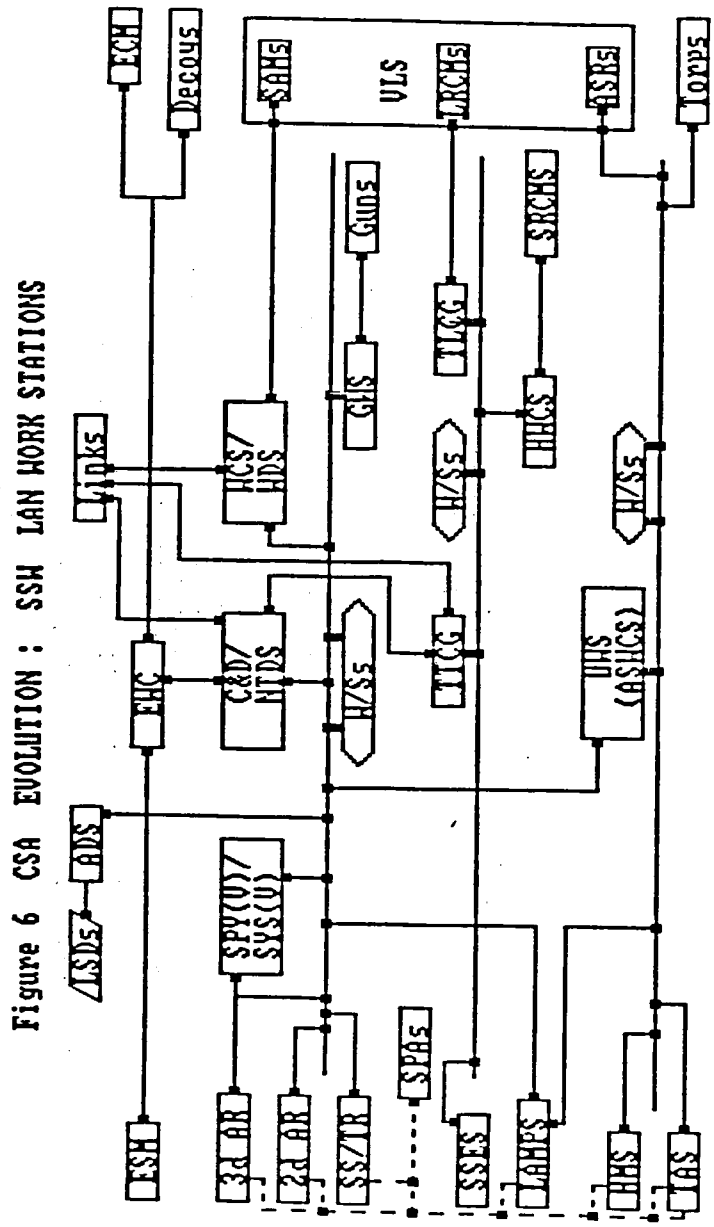


Figure 27. Diagram #6 of a System Design

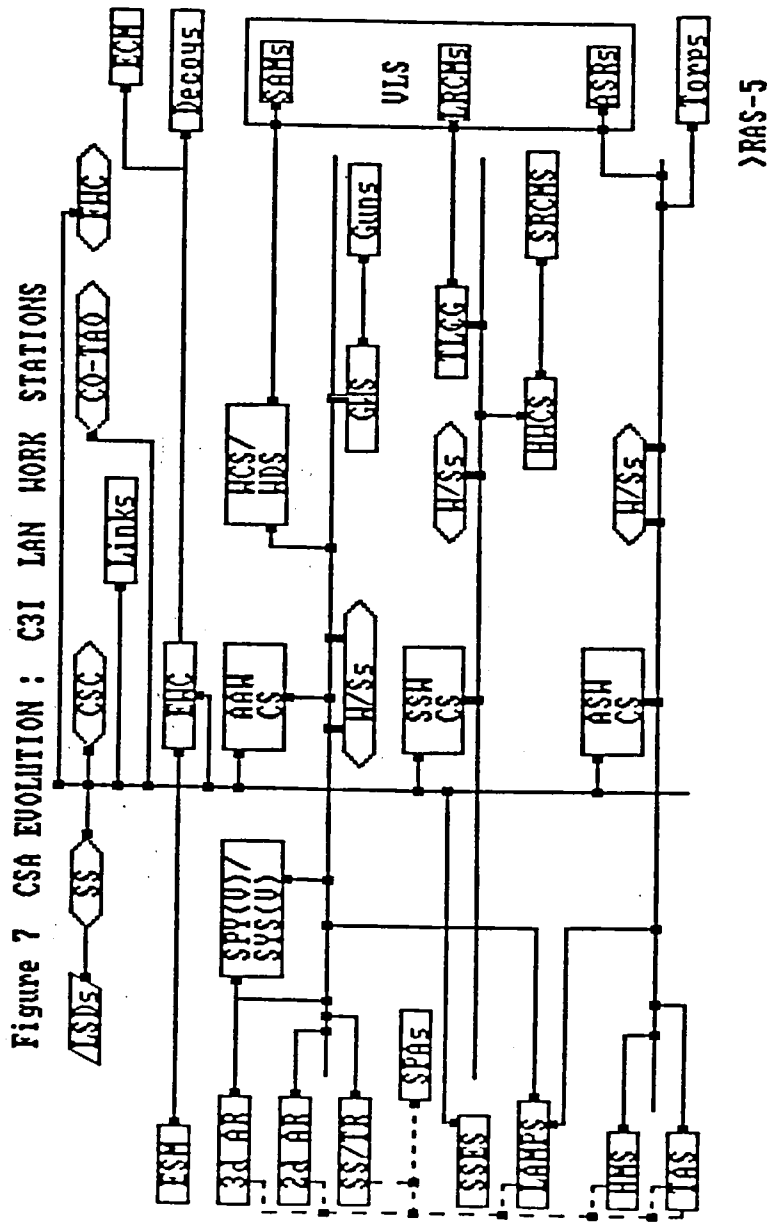


Figure 28. Diagram #7 of a System Design

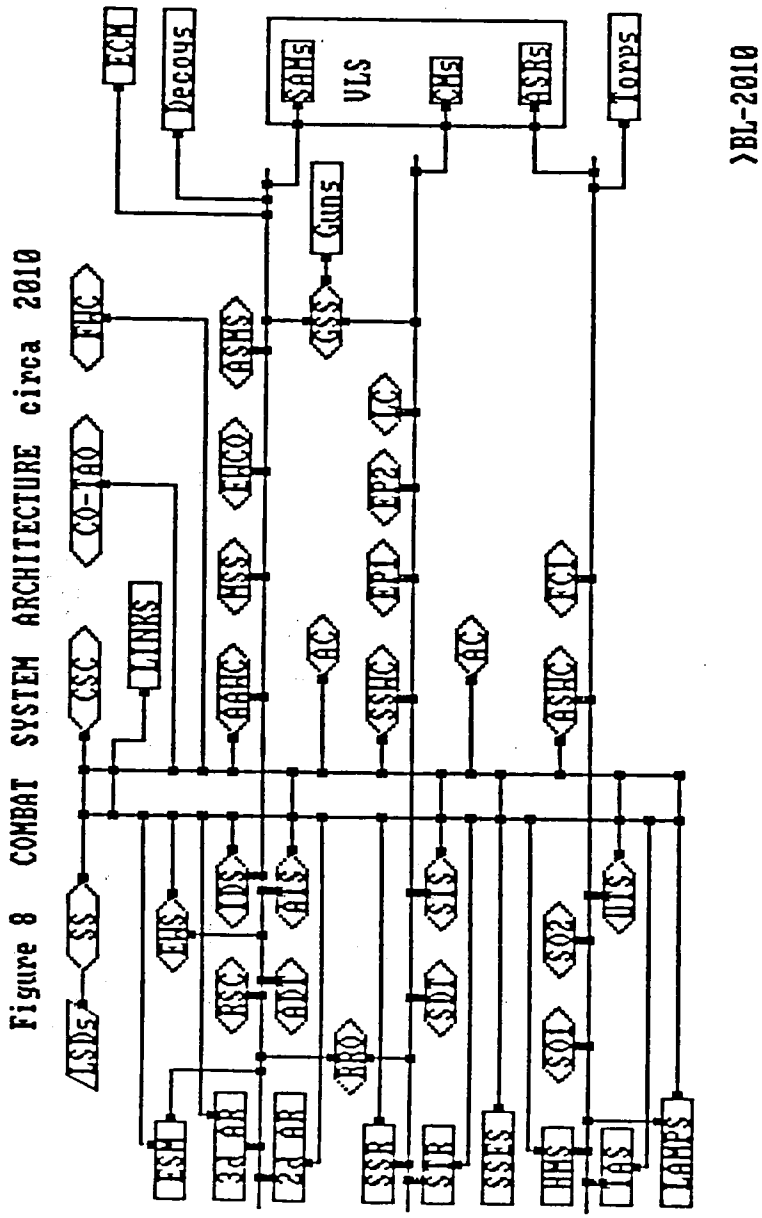


Figure 29. Diagram #8 of a System Design

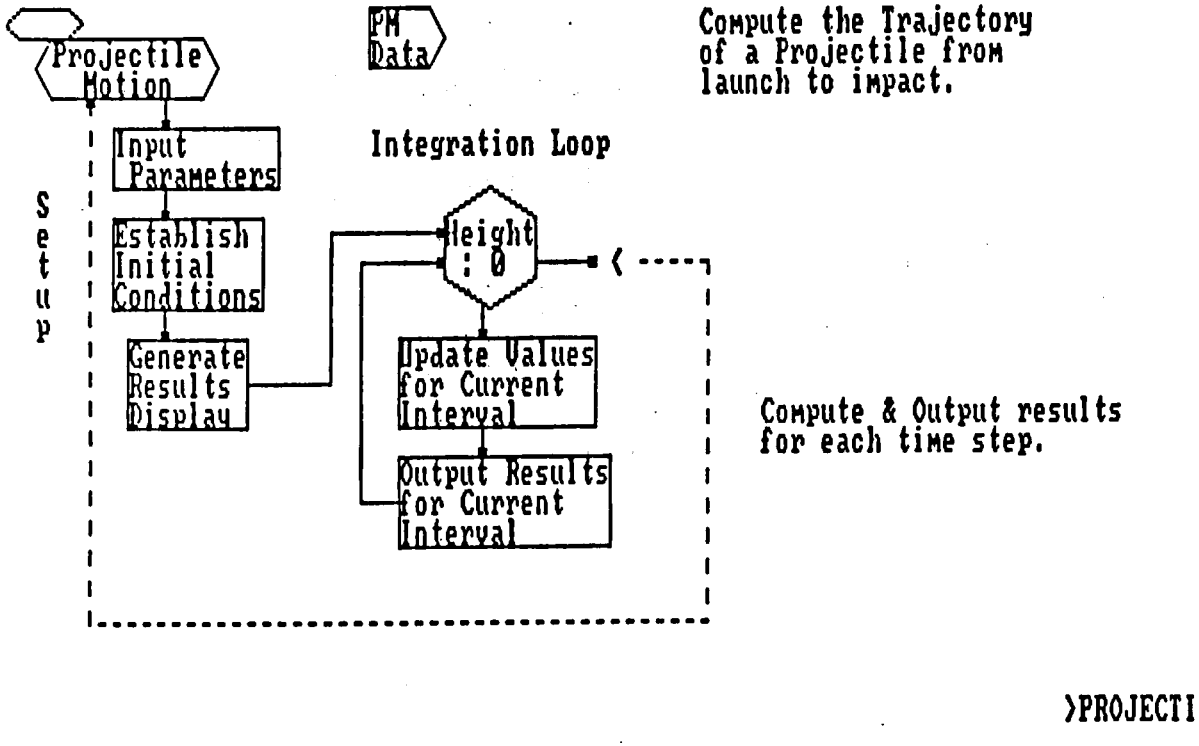


Figure 30. Diagram #1 of a Diagrammatic Language Program

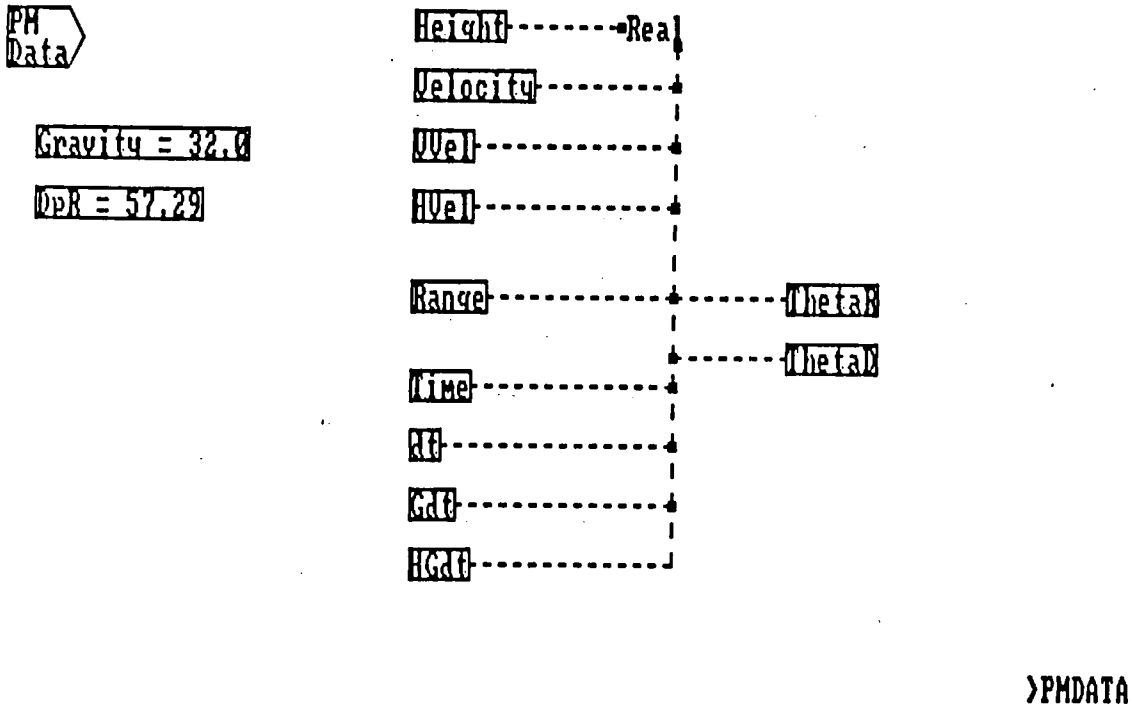


Figure 31. Diagram #2 of a Diagrammatic Language Program

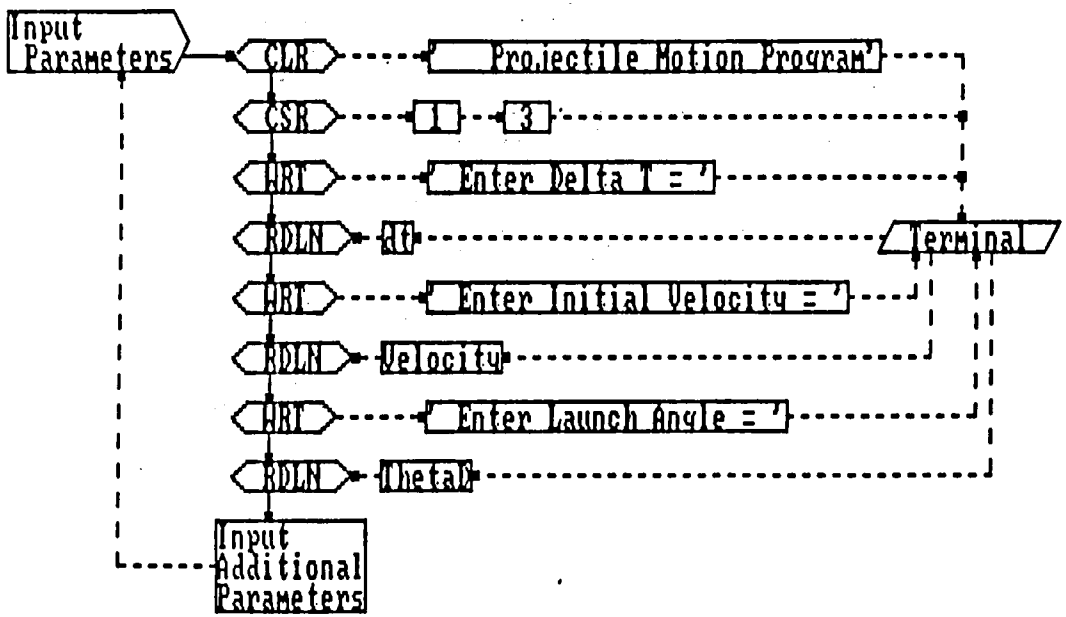
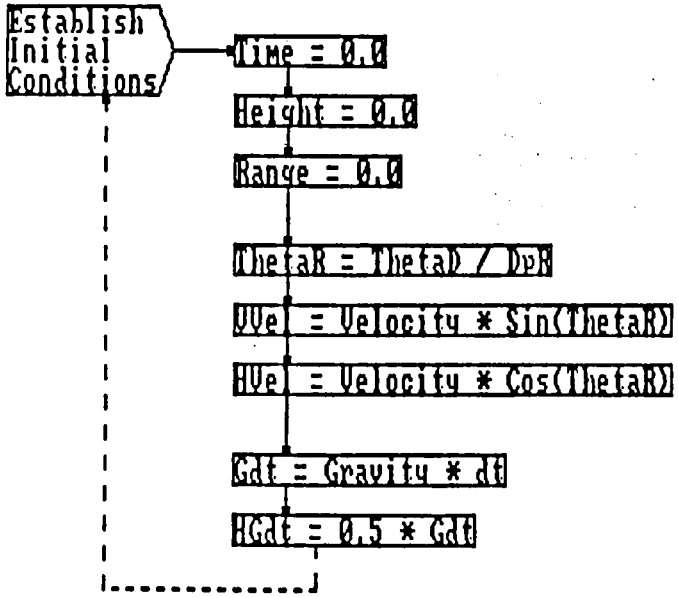
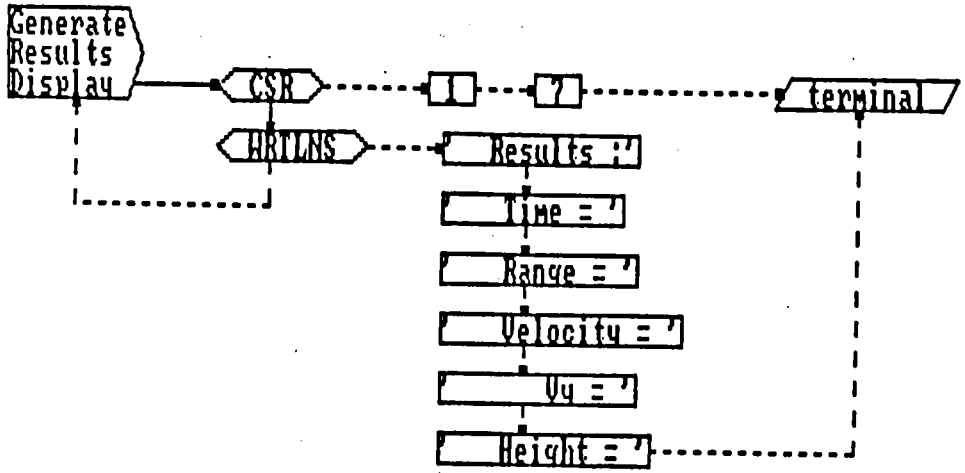


Figure 32. Diagram #3 of a Diagrammatic Language Program



>ESTABLIS

Figure 33. Diagram #4 of a Diagrammatic Language Program



>GENERATE

Figure 34. Diagram #5 of a Diagrammatic Language Program

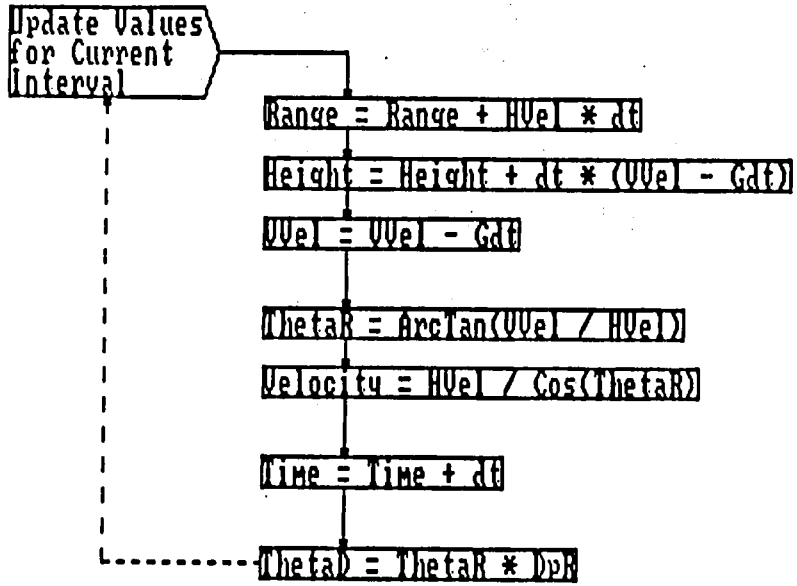


Figure 35. Diagram #6 of a Diagrammatic Language Program

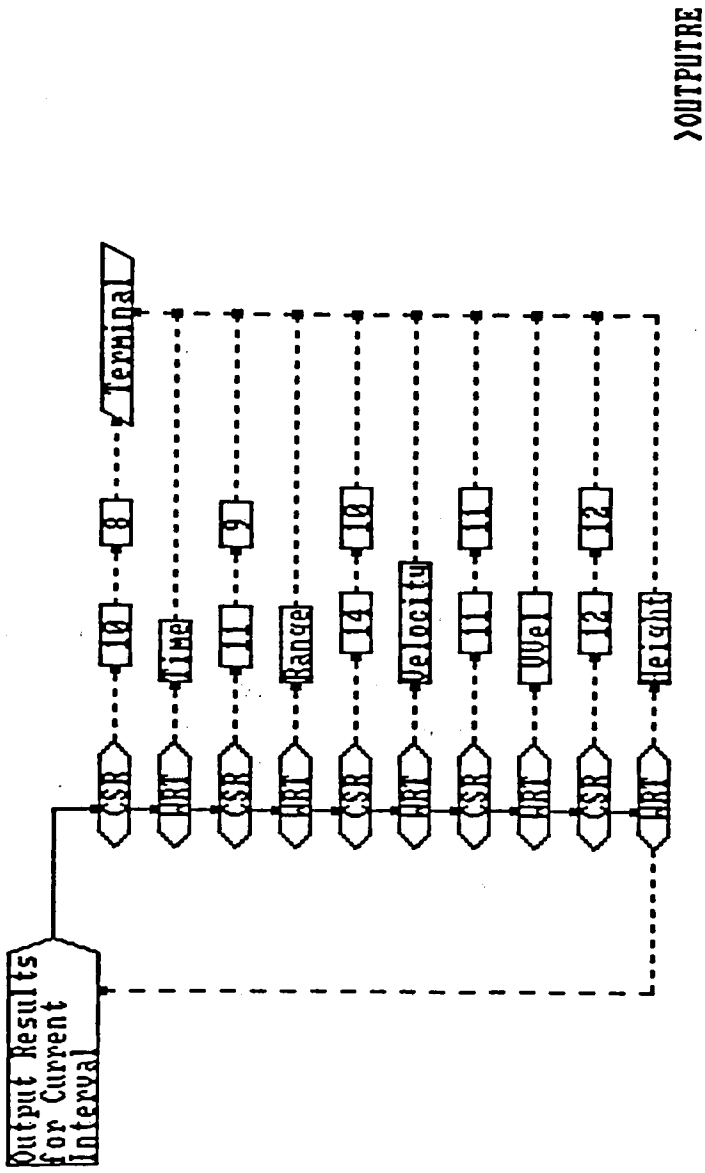


Figure 36. Diagram #7 of a Diagrammatic Language Program

Appendix A. DiagramEdit User's Guide

A.1 Environment

DiagramEdit runs under DOS 3.1 on any IBM PC or compatible with 256K or more of memory and CGA compatibility. There are no hard disk or RAM disk requirements. A single floppy disk is all that is needed to load and use DiagramEdit. All functions are performed with the keyboard so there is no need for special equipment such as a mouse, light pen or ball tab. Hard copy is obtained via the DOS "Shift-Print Screen" function.

A.2 Installation

There are no special procedures required to install DiagramEdit on a machine. The distribution diskette contains the eight files listed below.

DE.COM

DEMAIN.CHN

DEDRAW.CHN

DECLIP.CHN

EDCONLIN.HLP

EDDRAW.HLP

EDMAIN.HLP

EDTEXT.HLP

Depending on the machine configuration, the files may be copied to the hard disk or run directly from the diskette drive. If sufficient extended memory is available, optimum execution speed is realized on any PC series machine by operating from RAM disk (VDISK.SYS). Prior to running DiagramEdit, the DOS 'GRAPHICS' command must be entered if printouts of diagrams are desired.

A.3 Initialization

DiagramEdit is invoked by first establishing the DOS drive/directory where the distribution files reside and then entering the command 'DE' in response to the DOS prompt. The screen will be cleared and the display shown in Figure 1 will appear.

A.4 Screen Layout

All of the IBM PC screen except for the bottom line shown in Figure 1 is available for either creating, modifying, or viewing diagrams. This is the diagram area and is 24 rows

by 80 columns in size. The bottom line is separated into three fields. On the left is a field reserved for prompts and messages. The field in the center is used to display the main menu shown in Figure 1 and as a diagramming aid when the draw option is in effect. The third field which is on the right following the '>' symbol is always used for entry and/or display of the file name of the diagram currently in the diagram area. In Figure 1 the field is blank indicating that there is no diagram active at this time. This is always the case when DiagramEdit has just been initialized.

The prompt/message field always indicates the current state of DiagramEdit, the existence of an error condition, a request for a user entry, or a prompt for the next user action. In Figure 1 a message is displayed in the field indicating that DiagramEdit is ready to accept the selection of a main menu option.

A.5 Main Menu Option Summary

The options available on the main menu are briefly described below with more detail provided later in this user's guide.

An option is selected by pressing the key corresponding to the highlighted character(s).

New -- Clears a diagram from the diagram area and initializes DiagramEdit to start a new diagram. Includes an option for starting a new diagram with a text symbol chosen from the diagram currently in the diagram area.

Draw -- Initiates the drawing of a new diagram or the modification of an existing diagram.

Save -- Saves the diagram currently in the diagram area on the logged drive/directory.

Load Blk/Lst -- Loads a diagram from the logged drive/directory. The diagram desired may be selected from a diagram currently in the diagram area (load from Block/Symbol) or from a list of the file names on the logged drive/directory.

Print -- Prepares the diagram area of the screen for printing by removing the prompt/message and menu fields. Printing of the file name field is optional. Note that the screen may be printed at any time via the DOS "Shift-Print Screen" function. However, unless this option is used the entire bottom line will be included on the hard copy.

Help -- Displays the Help screen for the main menu. A reproduction of this screen is shown in Figure 2.

Esc -- Exits DiagramEdit and returns control to DOS in the current logged drive/directory.

A.6 Drawing a Diagram

By pressing the 'D' key to select the draw option on the main menu, the display shown in Figure 3 will appear on the screen. A triangular cursor, called the Draw/Select cursor is positioned in the middle of the diagram area. The 'Draw diagram.' message on the left of the bottom line indicates that a diagram may be drawn or modified.

As indicated on the bottom line to the right of the prompt/message field, the 'F10' key is used to bring up help screens. The content of the screen will depend on the drawing status. At this point or at any time the 'Draw diagram.' message is displayed, the Help screen shown in Figure 4 will appear. This screen serves as a reminder of the keys used to perform the various functions and is not intended as a tutorial.

The other items appearing on the bottom line to the left of the file name field will be explained later when appropriate.

A.6.1 Draw/Select Cursor Movement

At this point the Draw/Select cursor can be moved anywhere in the diagram area to position a text symbol that is to be drawn. Movement is effected horizontally and vertically using the four cursor arrow keys. The 'Home', 'PgUp', 'End', and 'PgDn' keys will direct the cursor toward the upper left, upper right, lower left and lower right, respectively. The cursor moves one character space at a time and will wrap around when the border of the diagram area is exceeded. The Row-Column indicator, 'R# C#', at the bottom of the screen will change to reflect the current position of the cursor.

The cursor can be made to move more quickly by pressing the 'Tab' key one or more times. Each time the 'Tab' key is pressed, the cursor will be set to move an additional character space when one of the cursor movement keys is used. The 'S#' indicator at the bottom of the screen reflects the number of characters that will be skipped. The

maximum number of characters that can be skipped is ten. Movement can be slowed down after use of the 'Tab' key by pressing the 'Shift'-'Tab' keys. This reduces the number of character spaces skipped by one each time they are used. The minimum number of character spaces that can be skipped is one.

The cursor can also be moved directly to eight fixed positions around the diagram area regardless of the initial position or speed setting. This is done using the 'Shift' key in conjunction with either the 'Home', Up Arrow, 'PgUp', Left Arrow, Right Arrow, 'End', Down Arrow, or 'PgDn' keys. The key combinations move the cursor to the row-column positions: 4-10, 4-40, 4-70, 12-10, 12-70, 20-10, 20-40 and 20-70, respectively.

A.6.2 Text Symbol Drawing and Movement

A text symbol is created when the 'F1' key has been pressed. The screen will appear as shown in Figure 5 after the cursor has been moved to the position 'R3 C3' and the 'F1' key has been pressed. The message 'Edit text symbol.' indicates that either text can be entered and/or modified, the symbol type can be selected or changed, or the symbol can be moved

to another position. Note in Figure 5 that the Draw/Select cursor is gone and that a bar cursor appears inside the symbol shown. This is referred to as the Text cursor and is only used for text editing and symbol movement.

The Help screen shown in Figure 6 will appear when the 'F10' key is pressed. As mentioned before, the information is provided as a reminder. A more definitive description of the available functions is provided below.

The creation/modification of text can be seen by entering text as shown in Figure 7. If part of the symbol is erased, it will be restored by subsequent actions. Note that the symbol automatically expands as text is entered. The symbol will also automatically contract when text is deleted. Text can be entered in a symbol up to five lines below (row 29) and twenty characters to the right (column 100) of the diagram area. Therefore, if a key is pressed to insert a character in column 100 or the 'Return' key is pressed while the cursor is on row 29, the key is ignored. Also, if a character is inserted in the middle of a line causing the last character on the line to move past column 100, the character that moves past column 100 is lost.

If the number of characters entered in a text symbol reaches five less than the maximum and a key is pressed to insert a character, the message 'Five characters left.' is displayed until the next key is pressed. This action will cause the message 'Edit text symbol.' to be redisplayed. If the number of characters reaches the maximum, the message 'No more characters.' is displayed when any key is pressed to insert a character. When any other key is pressed, the message 'Edit text symbol.' is redisplayed. At this point a character must be deleted before another one can be inserted.

Movement of symbols about the diagram area is done with the Text cursor active in a symbol as it is now. A symbol is moved horizontally by positioning the Text cursor at the beginning or end of any line and pressing the left or right cursor key. Vertical movement is done by positioning the cursor at the top or bottom line and pressing the up or down cursor key. A symbol will move the number of character spaces last established with the 'Tab' and/or 'Shift'-'Tab' keys each time a cursor key is pressed. These keys can also be used at this time to speed up or slow down movement of a symbol. Note that the Row-Column indicator at the bottom of the screen will reflect the movement, showing the position

of the Text cursor. If a cursor key is pressed to move a symbol to a position where the column number would be not be within 1 and 100 and the row number would not be within 1 and 29, then the key is ignored.

When the text has been entered and a symbol is positioned as desired, the 'F1' key is pressed to indicate that a symbol is complete as shown in Figure 8. Note that the Text cursor changes back to the Draw/Select cursor staying at the same position in the diagram area and the 'Draw diagram.' message is displayed. If a symbol is positioned so that a portion of the text extends onto another symbol or outside the diagram area, it can be entered with the 'F1' key and moved off to another position. However, if the text is completely out of the diagram area, then the message 'Symbol out of bounds.' is displayed briefly before being replaced by the 'Draw diagram.' message. When this occurs, the symbol cannot be recovered.

A text symbol that has been initiated may also be aborted by pressing the 'Esc' key. The symbol will be cleared from the diagram area, the Draw/Select cursor will be displayed at the same position, and the 'Draw diagram.' message will be displayed.

If five symbols less than the maximum number of symbols have been created in a diagram and the 'F1' key is pressed to create a new text symbol, the message 'Five symbols left.' is displayed briefly before being replaced by the 'Edit text symbol.' message. If the number of symbols in a diagram reaches the maximum, the message 'No more symbols.' is displayed briefly before being replaced by the 'Draw diagram.' message. At this point a symbol must be deleted before a new one can be initiated.

A.6.3 Changing and Repositioning Text Symbols

A text symbol can be modified or moved at any time by positioning the Draw/Select cursor anywhere on the text within the symbol and pressing the 'F1' key. The cursor will change to the Text cursor allowing any of the functions described previously to be performed. When the changes have been made, the symbol is exited by again pressing the 'F1' key. Note that the 'F1' key acts as a flip-flop, switching in and out of text symbols. If the cursor is positioned in a clear area and the 'F1' key is pressed, then a new text symbol is initiated. If it is located on the text of an existing symbol, then that symbol is entered for modification or movement.

If the 'Esc' key is pressed while in a previously created text symbol, then the effect will be the same as pressing the 'F1' key. In other words, the 'Esc' key will only abort the creation of a new symbol and not the modification of an existing symbol.

A.6.4 Selecting and Changing Text Symbol Types

There are twelve text symbol types available in DiagramEdit as illustrated in Figure 9. Symbol types are selected in text mode with the 'F3' key. To see how this is done, a second symbol is created in the diagram area at position 'R10 C30' as shown in Figure 10. With the Text cursor active, the 'F3' key is pressed and the symbol type changes, automatically stepping through all of the symbols and cycling back to the first symbol type. The desired symbol is selected, which in the case shown in Figure 11 is a rectangle, by pressing the 'F3' key a second time. The cycling stops with the symbol displayed and the cursor position unchanged. The symbol type selection can be done at any time after creation of a symbol is initiated.

The symbol type that appears when a symbol is being created is always the same as the last type selected. When a symbol is reentered, the type changes to that of the symbol.

Changing the symbol type after a symbol is completed can be done at any time by reentering the symbol, selecting the new symbol type with the 'F3' key, and then exiting the symbol. Note that the text can be changed at the same time and/or the symbol can be moved to a new position. If either the type, size or position of a symbol changes and connectors (described shortly) are attached to that symbol, then they are cleared when the 'F1' key is pressed to complete the symbol.

A.6.5 Deleting Text Symbols

Completed text symbols are deleted by positioning the Draw/Select cursor on the text of a symbol and pressing the 'F5' key. The symbol and all attached connectors are cleared from the diagram area. Care should be exercised when using this function since the symbols cleared from the diagram area cannot be recovered.

If the cursor is not positioned on a text symbol (or on a connector or on a line), then the message 'Invalid delete.'

will appear. The cursor can be repositioned for another try or another function can be used.

A.6.6 Copying Text Symbols

Completed text symbols are copied by positioning the Draw/Select cursor on the text of a symbol and pressing the 'F4' key. The message 'Symbol selected for copy.' will appear. The cursor is then moved to the desired position for the copy and the 'F4' key is pressed again. The copy will be drawn in the diagram area with the upper left-most character positioned coincident with the cursor. The 'Draw diagram.' message will then be displayed and the cursor will remain where it is. If a symbol is copied over another symbol or extends out of the diagram area, it can be entered with the 'F1' key and moved off to another position. In the case where the copy extends off the diagram area and outside of row 29 and/or column 100, the message 'Invalid copy.' is displayed and the symbol may be copied elsewhere on the screen.

If the cursor is not positioned on a text symbol, then the message 'Invalid symbol for copy.' will appear. The cursor

can be repositioned for another try or another function can be used.

The 'Esc' key may be pressed, after a symbol has been selected for copy with the 'F4' key, to abort the action. The 'Draw diagram.' message is displayed and the cursor can be repositioned for another copy or another function can be used.

A symbol that has been copied becomes a completed symbol in its own right. It bears no relationship to the original as far as DiagramEdit is concerned. The copy can be modified, moved, copied or deleted as though it was an original symbol.

If five symbols less than the maximum number of symbols have been created in a diagram and the 'F4' key is pressed to copy a text symbol, the message 'Five symbols left.' is displayed briefly before being replaced by the 'Symbol selected for copy.' message. If the number of symbols in a diagram reaches the maximum, the message 'No more symbols.' is displayed briefly before being replaced by the 'Draw diagram.' message. At this point a symbol must be deleted before a new one can be initiated.

A.6.7 Connecting Symbols

A connector between two text symbols is started by positioning the Draw/Select cursor on the text of a 'From' symbol and pressing the 'F2' key. The cursor is moved to a 'To' symbol with the cursor movement keys. The connector is automatically drawn as the cursor progresses. When the cursor is on the text of the 'To' symbol, the 'F2' key is pressed again to indicate completion of the connector. The connector is then clipped to the edges of the text symbols and the head(s) are drawn. Therefore, precise positioning of the cursor at the start or finish is not a concern. The path of the connector between the symbols will be left as drawn.

After the 'F2' key is pressed to start a connector, the Help screen shown in Figure 12 will appear when the 'F10' key is pressed. As mentioned before, the information is provided as a reminder. A more definitive description of the available functions is provided below.

The connector symbol type is selected before or after the connector is started. The 'F3' key is used in both cases. When used before, pressing the 'F3' key will cause the connector/line symbol type indicator on the bottom line to

cycle through the four connector and two line types shown in Figure 13. When used after the connector is started, only the connector types will be cycled. (Line drawing is described later.) In either case, when the desired symbol type appears, the 'F3' key is pressed to complete selection of the symbol.

The drawing of a connector is shown in Figure 14 prior to clipping. The symbol type displayed on the bottom of the screen is the solid unidirectional connector. The connector is started by positioning the Draw/Select cursor on the text of the symbol on the left and pressing the 'F2' key. The 'Draw conn/line.' message will appear in response to this action. After pressing the down arrow cursor key several times, a solid line is produced trailing the Draw/Select cursor. The line is extended into the text of the lower right symbol with the right arrow cursor key, and the 'F2' key is pressed to complete the connector. After the connector is clipped, the diagram appears as shown in Figure 15. The Draw/Select cursor remains where it was and the 'Draw diagram.' message is displayed. As mentioned in the above paragraph, the connector type could have been changed at any time with the 'F3' key while the connector was being drawn.

Note that the connector just drawn has two line segments. The maximum number of line segments that DiagramEdit allows is ten. When the tenth segment is started, the message 'Last line segment.' is displayed. If a connection is not completed, the 'C/L too long. Press Esc.' message appears. Pressing the 'Esc' key causes DiagramEdit to delete the connector, leave the Draw/Select cursor at the same position, and display the 'Draw diagram.' message.

When the 'F2' key is pressed to start a connector and the Draw/Select cursor is not positioned on the text of a symbol, then the message 'Invalid conn/line.' will appear. At this point the cursor can be repositioned for another try or another function can be used. The same message will appear again if the 'F2' key is pressed a second time and a line segment has not been drawn or the cursor is not positioned on the text of a symbol after at least one line segment has been drawn. In either case the connector is not deleted and can be continued until the cursor is positioned on the text of a symbol.

The 'Esc' key can be pressed to abort the drawing of a connector at any time before pressing the 'F2' key. The line segment(s) drawn will be erased, the cursor will remain

where it was last positioned, and the 'Draw diagram.' message will be displayed.

In addition to drawing connectors between two text symbols as discussed above, connectors can be drawn:

1. between two line symbols,
2. from a text or line symbol to a connector,
3. from a connector to a text or line symbol,
4. between two connectors,
5. from a text or line symbol to the same text or line symbol, and
6. from a connector to the same connector.

If five symbols less than the maximum number of symbols have been created in a diagram and the 'F2' key is pressed to initiate the drawing of a connector, the message 'Five symbols left.' is displayed briefly before being replaced by the 'Draw conn/line.' message. If the number of symbols in a diagram reaches the maximum, the message 'No more symbols.' is displayed briefly before being replaced by the

'Draw diagram.' message. At this point a symbol must be deleted before a new one can be initiated.

A.6.8 Changing and Deleting Connector Symbols

Connector symbol types can be changed at any time in a manner analogous to text symbols. The Draw/Select cursor is positioned on the symbol and the 'F3' key is pressed. The symbol as well as the indicator at the bottom of the screen will change at a fixed rate, cycling through the four connector types. When the desired symbol type appears, the 'F3' key is pressed again. The cycling will stop with the new symbol type displayed. Connectors attached to the connector being changed are not effected.

Connectors are deleted in the same way as text symbols. The Draw/Select cursor is positioned on the symbol and the 'F5' key is pressed. The connector and all connectors attached to it are cleared. Associated text and/or line symbols are not effected.

A.6.9 Drawing, Changing and Deleting Line Symbols

There are two lines symbol types available in DiagramEdit, solid and dashed. These symbol types are selected in

exactly the same way as connectors when the 'Draw diagram.' message is displayed. The 'F3' key is pressed and the connector/line indicator cycles repeatedly through the connector and line symbol types. When the desired line symbol type appears, the 'F3' key is pressed a second time and the cycling stops on that type.

A line is drawn by positioning the Draw/Select cursor at any starting position in the diagram area and pressing the 'F2' key. The message 'Draw conn/line.' will appear in response. The line is now drawn using the cursor movement keys. When any ending position is reached, the 'F2' key is pressed a second time to specify completion of the symbol. The 'Draw diagram.' message appears in response to the action. An example of a completed, dashed line is shown in Figure 16. Note that no clipping ever occurs when a line is completed.

If a line type appears at the bottom of the screen, and the 'F2' key is pressed two times in succession, then the messsage 'Invalid conn/line.' will appear and the line may be continued.

The drawing of a line symbol can be aborted at any time by pressing the 'Esc' key. The line will be erased, the

Draw/Select cursor will remain at its current position, and the 'Draw diagram.' message will be displayed.

A line symbol type can be changed at any time while it is being drawn. The 'F3' key is pressed and the line drawn to the current point as well as the connector/line indicator will switch back and forth between the two line types. When the desired type appears, the 'F3' key is pressed again to make the selection. The switching stops and the line drawing can continue.

A line symbol type is changed after completion in exactly the same way as connectors. The Draw/Select cursor is positioned anywhere on the line and the 'F3' key is pressed. The line and connector/line indicator will switch between the two line types until the 'F3' key is pressed again to make the selection.

Deletion of a line symbol is the same as for connectors. The Draw/Select cursor is positioned on the symbol and the 'F5' key is pressed. The line and any attached connectors are erased from the diagram area.

If five symbols less than the maximum number of symbols have been created in a diagram and the 'F2' key is pressed to

initiate the drawing of a line, the message 'Five symbols left.' is displayed briefly before being replaced by the 'Draw conn/line.' message. If the number of symbols in a diagram reaches the maximum, the message 'No more symbols.' is displayed briefly before being replaced by the 'Draw diagram.' message. At this point a symbol must be deleted before a new one can be initiated.

A.7 Saving Diagrams

Diagrams can be saved from the main menu or when the draw display is active. When the main menu is displayed, the 'S' key is used to initiate a save. From the draw display the 'F9' key is used. In both cases the procedure is the same. After the option is selected, the 'Input file name.' message appears. A Bar cursor is positioned after the '>' symbol on the extreme right hand side of the bottom line. A valid DOS file name, without an extension, can now be typed into the field. When the 'Return' key is pressed, the diagram in the diagram area is stored on the currently logged drive/directory with extension '.SYM' automatically appended to the file name by DiagramEdit. The 'Diagram is being saved.' message appears briefly before being replaced by the

'Select Option>' or 'Draw diagram.' message, as appropriate. The file name remains on the screen as entered.

If the file name already exists on the currently logged drive/directory, the warning message 'File already exists.' will appear briefly, followed by the query 'Overwrite (Y/N)?'. If the response is the 'Y' key (Yes), then the file will be overwritten with the current diagram. A 'N' key (No) key response will cause DiagramEdit to abort the save operation.

When a file name is already displayed on the screen from a previous 'Save', 'Load' or 'New' operation, it can be reused or a different file name can be entered. If the current file name is to be used, then only the 'Return' key needs to be pressed. A different file name is specified by following the procedure for entering a file name. When the first character of the name is typed, the old file name is cleared.

The save operation can be aborted at any time before or during typing of the file name by pressing the 'Esc' key. This causes DiagramEdit to terminate the save and display the 'Select Option>' or 'Draw diagram.' message, as appropriate. If a previously displayed file name is being

overwritten when the 'Esc' key is pressed, then the original name is restored to the screen.

If an I/O error occurs during the saving of a diagram, DiagramEdit will abort the save operation and display an I/O error message briefly before replacing it with the 'Select Option>' or 'Draw diagram.' message, as appropriate. A list of the I/O errors that can occur during the saving of a diagram appear at the end of this user's guide.

A.8 The New Option

The main menu 'New' option is used to start a new diagram when a diagram is already present in the diagram area. The option is selected by pressing the 'N' key. If the current diagram has not been saved, then the prompt 'Save diagram (Y/N)?' is displayed and the diagram can be saved or not. The save is performed as described above.

If the diagram had previously been saved or after the response to the save prompt under this option has been made, the 'Select text symbol (Y/N)?' prompt is displayed by DiagramEdit. This feature provides the option of choosing a text symbol to be reproduced on the new, cleared diagram area. The option is invoked by a 'Y' (Yes) key response to

the prompt. The Draw/Select cursor will appear where it was last positioned and the 'Select symbol for new.' message will be displayed. A text symbol is selected by positioning the cursor on the text of the symbol and pressing the 'F1' key. DiagramEdit will reproduce the symbol in the new, cleared diagram area, construct a file name from the text of the symbol, display it on the bottom line of the screen, and display the 'Select Option>' message. If the cursor is not placed on the text of a symbol or DiagramEdit cannot construct a valid file name from the text of the symbol selected, the message 'Invalid symbol selected.' is displayed and the cursor can be positioned for another try.

If the 'N' (No) key is pressed in response to the 'Select text symbol (Y/N)?' prompt or the 'Esc' key is pressed after a 'Y' (Yes) response, DiagramEdit will clear the whole screen, reinitialize itself, and display the main menu and the 'Select Option>' message.

A.9 The Load Option

The main menu 'Load' option is used to load a diagram from the logged drive/directory. The diagram desired may be selected from a diagram currently on the display (load from

Block/Symbol) or from a list of the file names on the logged drive/directory. The option to load from a block is selected by pressing the 'B' key, while the option to load from a list of file names on the logged drive/directory is selected by pressing the 'L' key. If the current diagram has not been saved when either key is pressed, then the prompt 'Save diagram (Y/N)?' is displayed and the diagram can be saved or not. The save is performed as described earlier.

When the 'B' key is pressed to load a diagram from a block/symbol, the Draw/Select cursor appears where it was last positioned and the message 'Select symbol for load.' is displayed. A text symbol is selected by positioning the cursor on the text of the symbol and pressing the 'F1' key. DiagramEdit will construct a file name from the text of the symbol, display the file name on the bottom line of the screen, and load the diagram from the logged drive/directory. The message 'Diagram is being loaded.' will also appear briefly before being replaced by the 'Select Option>' message.

If the cursor is not placed on the text of a symbol or DiagramEdit cannot construct a valid file name from the

text, the message 'Invalid symbol selected.' is displayed. Also, if DiagramEdit cannot find the file on the logged drive/directory, the message 'File does not exist.' is displayed. In either case, if a previous file name is overwritten, the old file name is restored and the cursor can be repositioned for another try.

When the 'L' key is pressed to load a diagram from a list of file names on the logged drive/directory (.SYM files), the list is displayed in the diagram area along with the 'Select file to load.' message. The Draw/Select cursor is also displayed pointing to the file name in the upper left corner of the diagram area. The cursor arrow keys are used to position the cursor to point to the file name of the diagram to be loaded. A file name is selected by pressing the 'Return' key. DiagramEdit will load the diagram from the logged drive/directory, briefly displaying the message 'Diagram is being loaded.', before replacing it with the 'Select Option>' message. Note that if the load from list option is entered again, the cursor will be displayed at the point it was previously positioned.

The 'Esc' key may be pressed after the 'B' key or the 'L' key to abort the loading of a diagram. In the case of it

being pressed after the 'B' key, DiagramEdit will remove the Draw/Select cursor from the diagram area and display the 'Draw diagram.' message. In the case of the 'Esc' key being pressed after the 'L' key, DiagramEdit will remove the list of file names from the diagram area, redraw the previous diagram if any, and display the 'Draw diagram.' message.

If an I/O error occurs during the loading of a diagram, DiagramEdit will abort the load operation and display an I/O error message briefly before clearing the whole screen, reinitializing itself, and displaying the main menu and the 'Select Option>' message. A list of the I/O errors that can occur during the loading of a diagram appear at the end of this user's guide.

A.10 Leaving the Editor

The 'Esc' key is used to return from the draw display to the main menu and to return from the main menu to DOS in the logged drive/directory. When returning from the main menu to DOS, if the current diagram has not been saved, then the prompt 'Save diagram (Y/N)?' is displayed and the diagram can be saved or not. The save is performed as described earlier. Control then returns to DOS.

A.11 Drawing Techniques

By employing several simple drawing techniques based on the flexibility of the functions presented above, DiagramEdit can be used to draw diagrams in a more efficient manner. Several of these techniques are presented below.

Draw groups of text symbols first, before drawing connectors. Since movement of text symbols is easy, position is not as much of a concern. The diagram area can be used as a scratchpad for drawing text symbols. After the groups of text symbols have been modified and/or moved to the desired positions, the connectors can be drawn. There are two reasons for drawing this way. One, most people decide what information they want in the text symbols and where they should be positioned, before deciding how to connect them. Two, usually when text symbols are modified/moved, any connectors attached to them are deleted and then must be redrawn, which takes time. Automatic redrawing of connectors was considered for DiagramEdit, but was rejected since these algorithms usually draw connectors in a path that is not acceptable to a user. Thus, connectors would have to be redrawn anyway.

Use the the 'Shift' key in conjunction with the cursor keys to jump to different positions in the diagram area. This technique can save a great deal of keystrokes when the Draw/Select cursor is being positioned on a symbol.

To create a new text symbol similar to one that already exists, make a copy of it and then modify it. In many cases, this will save time since text can usually be modified more easily than created, and the symbol may already be of the desired type. Similar to this technique is a shortcut for creating a text symbol of the same type as one that already exists. By positioning the Draw/Select cursor on a text symbol of the desired type and pressing the 'F1' key twice, the next symbol will be created with that type also.

Use the null symbol to place comments in a diagram. A group of symbols can be commented by drawing a block around them using a line symbol and placing comments inside the block.

Use the new, save and load from block options to draw diagrams in a hierarchy as follows. A hierarchy of diagrams is started by drawing the top-level diagram and selecting the new option. When the prompt 'Select text symbol (Y/N)?'

is displayed, respond by pressing the 'Y' (Yes) key to select a text symbol. After a text symbol is selected with the 'F1' key, DiagramEdit reproduces the symbol on the new, cleared diagram area and creates a file name from the text. This diagram can now drawn and modified, and when it is complete, the save option can be used to save the diagram. By pressing the 'Return' key, the diagram will be saved under the created name. This process can be continued to draw diagrams at the next level down or at the same level. When the diagram at the top level is loaded later using the load from list option, the load from block option can be used to move down through the hierarchy. The load from list option can then be used again by pressing the 'Return' key to reload the top-level diagram. This enables a user to only have to remember the file name of the top-level diagram.

A.12 I/O Errors

The messages shown below are displayed when an I/O error occurs during the loading or saving of a diagram. The messages appear next to their error numbers which are in hexadecimal.

```
01  'File does not exist.'
02  'File not open for input.'
03  'File not open for output.'
04  'File not open.'
05  'Cannot read from file.'
06  'Cannot write to file.'
10  'Error in numeric format.'
20  'N/A on logical device.'
21  'N/A in direct mode.'
22  'N/A on standard files.'
90  'Record length mismatch.'
91  'Seek beyond end of file'
99  'Unexpected end of file.'
F0  'Disk write error.'
F1  'Directory is full.'
F2  'File size overflow.'
F3  'Too many open files.'
FF  'File disappeared.'
```

The vita has been removed from the scanned document