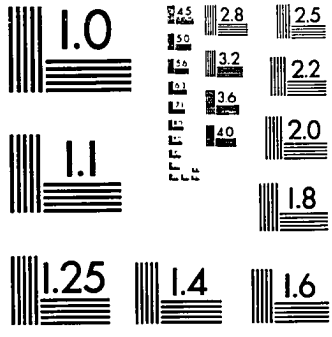


Humphreys, George Edward

A MICROCOMPUTER-AIDED DRAFTING SYSTEM

The University of Oklahoma D. ENGR. 1985

THE UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

A MICROCOMPUTER-AIDED DRAFTING SYSTEM

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

degree of

DOCTOR OF ENGINEERING

by

GEORGE EDWARD HUMPHREYS

Norman, Oklahoma

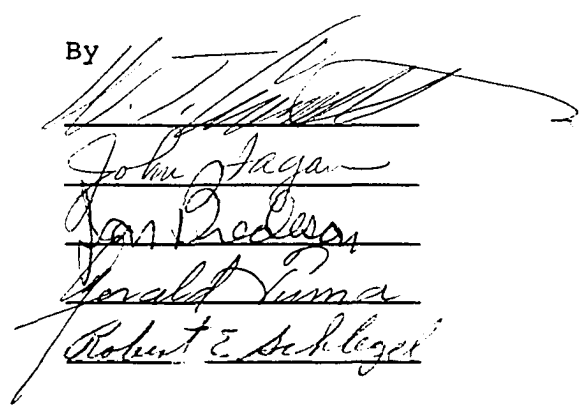
1985

A MICROCOMPUTER-AIDED DRAFTING SYSTEM

A DISSERTATION

APPROVED FOR THE SCHOOL OF ELECTRICAL ENGINEERING

AND COMPUTER SCIENCE



ACKNOWLEDGEMENTS

The author hereby acknowledges the help which he has received from the members of his graduate committee. Dr. W.T.Cronenwett, chairman of the committee, has been especially helpful throughout the project.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

ABSTRACT

This microcomputer-aided drafting system (MADS) is easily learned and used. Most of the communication with the user is done through menus and ample opportunity is provided for recovery from mistakes. It provides the user with a facility for generating high quality electronic schematics which can be easily changed or updated and scaled to fit various sizes of paper. A user who is familiar with electronic drafting can produce a usable drawing within an hour or two of his first introduction to the system. The general principle of operation for this system is to enter a schematic into the system with the keyboard and the graphic controls while the drawing is displayed on a video screen at various scale factors.

A MICROCOMPUTER-AIDED DRAFTING SYSTEM

CHAPTER I

INTRODUCTION

The development of the microcomputer-aided drafting system described here was motivated by the experience of the author while practicing as a design and development engineer in an industrial setting in 1979 and 1980. The fundamental difficulty that the author encountered involved the maintenance of reasonably clean schematics that could accurately describe the state of the product that was currently being developed. The process required keeping a master copy, the original, of each drawing, from which working copies were made to be used and modified in the laboratory as the development of the product continued. The modifications made in the laboratory were then edited into the master. The editing process was tedious and time-consuming because it involved erasing and redrawing, or drawing and pasting. To have the changes made by the drafting department was not a viable alternative because the

department was busy and time was required to allow the editing process to be fitted into the schedule. In this particular case, each engineer on the design team was responsible for the maintenance of several drawings of different subsystems. For example, the author was responsible for the schematic of a large-scale integrated circuit although only some of the circuits on the schematic were designed by him. Each of the engineers had to update drawings to reflect changes that were made by other engineers as well as himself. The author, therefore, was interested in finding a computer system that could answer such a need. Unfortunately, the least expensive drafting systems available then cost at least one hundred thousand dollars. No one had tried to develop a truly inexpensive computer-aided drafting system at that time. When it later happened that the author's graduate program required a design project, an inexpensive drafting system immediately came to mind.

After some discussion, it was decided that a microcomputer-aided drafting system would be a worthwhile project if the system could produce high-quality drawings and if the system could be designed to be as general as possible. The project was somewhat complicated by the lack of suitable microcomputers on the market at that time as well as by the extreme budgetary restrictions imposed by

the design goal of keeping the system inexpensive and the fact that the author financed the entire project with his own resources.

The first microcomputer-aided drafting system (MADS) that was developed as part of this project was much more general than is the present version. It would communicate with the user in any units, would scale and rotate drawing elements, and would also do several other less important things that the present system will not do. That system, however, was difficult to use just because it was versatile and, therefore, required that the user regularly provide it with parameters. It was, therefore, decided that MADS be revised. The primary emphasis was on developing a "user-friendly" system that could be used to generate and update only electronic schematics. The system described here is the result of that revision.

When the author started this project, very little printed information on the subject was available. Much of the development work, therefore, was based on intuition and the author's experience with video display systems in industry. Since then, more information has become available as more and more companies have produced similar commercial products. The material included in the bibliography provides either very general principles concerning

man-machine interfaces and computer graphic systems or extremely specific information concerning the hardware and software used during the development of this system.

Chapter II is intended to be removed from this paper and used alone as an operator's manual.

CHAPTER II

OPERATION MANUAL

Introduction

This microcomputer-aided drafting system (MADS) is easily learned and used. It provides the user with a facility for generating high quality electronic schematics which can be easily changed or updated and scaled to fit various sizes of paper. Furthermore, a user who is familiar with electronic drafting can produce a usable drawing within an hour or two of his first introduction to the system. After power is turned on for the computer, the drafting system is started by pressing the 'D' key and then entering the current date with the keyboard.

The general principle of operation for this system is to enter a schematic into the system with the keyboard and the graphic controls while the drawing is displayed on a video screen at various scale factors. While developing the schematic, the user can either enlarge any part of the

drawing by a factor of as much as sixteen, or take a wide-angle view of the entire drawing at any time. It is also quite easy to produce evenly spaced lines according to standard practice since the system will place drawing elements on one-eighth-inch centers in most cases.

Function Menu

The first menu displayed in this primarily menu-driven system is the function menu shown in Figure 1. A function is chosen by moving the alphanumeric cursor up or down using the arrow keys and pressing the ENTER key with the cursor located next to the desired function. This menu is self-explanatory; since it is the entry point to the various functions of the system, each of the entries will be explained in the order in which it appears. Four of the functions of this menu begin by asking the user to enter the name of a drawing. If, for any reason, the user wishes to return to the menu without executing the function, he can do so by pressing the enter key when asked for the name. This is true for most of the cases in which the user is asked to supply an alphabetic or numeric parameter to the system. The exceptions to this rule will be pointed out as they are encountered.

HERE ARE YOUR CHOICES
( ) QUIT
( ) EDIT AN OLD DRAWING
( ) PLOT A DRAWING
( ) START A NEW DRAWING
( ) DEFINE A DRAWING ELEMENT
POSITION CURSOR ON CHOICE AND PRESS ENTER

Figure 1. Function Menu

Quit

The quit function simply returns control of the computer to the Basic language system. Type RUN and press the enter key to return to MADS.

Edit An Old Drawing

MADS checks the name of the drawing entered by the user to make sure that the drawing is already in the file which is currently in the machine. It then loads an extensive program into the computer and initializes several files before displaying the element menu shown in Figure 2. The heading of the menu tells the user whether the system is generating solid lines or dashed lines and the distance that is assumed between dots on the screen. The entries in the menu are as follows.

SOLID LINE MODE
DELTA = 1/16 INCH
ELEMENT MENU

POSITION TEXT CURSOR AND PRESS ENTER

( ) GRID LINE	( ) TOP HALF CIRCLE
( ) RECTANGLE	( ) LEFT HALF CIRCLE
( ) CIRCLE	( ) BOTTOM HALF CIRCLE
( ) HORIZONTAL TEXT	( ) RIGHT HALF CIRCLE
( ) VERTICAL TEXT	( ) UPPER RIGHT 1/4 CIRCLE
( ) DETAIL LINE	( ) UPPER LEFT 1/4 CIRCLE
( ) MACRO ELEMENT	( ) LOWER LEFT 1/4 CIRCLE
( ) CURSOR POSITION	( ) LOWER RIGHT 1/4 CIRCLE
( ) ERASE LAST ITEM	( ) ERASE LAST MACRO ITEM
( ) ZOOM	( ) ERASE MODE(WITH ZOOM)
( ) WIDE ANGLE	( ) DASHED LINE MODE
( ) QUIT	( ) DASHED LINE OFF

Figure 2. Drawing Menu

Grid Line. The grid line function will allow the user to draw straight lines on one-eighth-inch centers by moving the graphic cursor and pressing the red button on the trackball. To draw the first of a sequence of lines, it is necessary to indicate both ends of the line. Another line can be appended to the first by moving the cursor to another point and pressing the red button on the trackball. The drafting system will draw a line from the end of the last entered end of the first line to the most recently entered point. The user can return to the menu by pressing the red button twice at the same location. Since the alphanumeric cursor comes to rest on the grid line function, it is inevitable that a user will enter this function

inadvertently. If both ends of the first line are indicated at the same location, the system will leave a dot on the graphic screen but nothing will be recorded on the drawing, and the system will return to the menu. This is a handy way to determine the location of grid points.

Rectangle. The rectangle function is used to place rectangles on the drawing. The user is directed to position the graphic cursor on the lower left corner of the rectangle and press the red button. He then indicates the upper right corner with the same procedure. Both corners will be placed on the nearest grid point.

Circle. The circle function allows, as would be expected, the placement of circles on the schematic. The system asks for the radius and the center of the circle. The user can generate a series of circles, i.e., circuit nodes, by indicating the centers of the circles one at a time. To return to the menu, the user presses the red button in the same place twice. If, when asked for the circle radius, the user simply presses the enter key, a default value will be assumed by the system which is a correct value for creating circuit nodes. This description of circle generation also applies to half circles and quarter circles.

Horizontal Text. Labels containing nine characters or less can be put on a drawing with this function. The user must enter a code determining the size of the print to be used and the location of the label. The code is a number from zero to fifteen which corresponds to print ranging in size from less than one-tenth of an inch to approximately seven-sixteenths of an inch. Multiple labels with the same size print can be entered without going back to the menu by moving the graphic cursor to a new location after entering each label. Pressing the red button at the same location twice will return the system to the menu.

Vertical Text. Vertical labels are the same as horizontal labels except that the label reads vertically from the bottom to the top of the page.

Detail Line. This function is the same as grid line except that it will place lines exactly where directed without rounding to grid points.

Macro Element. A macro element is a collection of primitive drawing elements previously defined and constructed as described below on page 15. A macro element can be placed on the drawing as though it were a single element. Examples would be transistors or logic gates. Placement on the drawing of macros such as resistors or capacitors that are normally connected between nodes is

effected by indicating the end points between which the macro should be connected. Macros without external connections or with arbitrary connection points, such as logic gates, are placed on the drawing by indicating the center of the element. A macro can be placed in several different locations without returning to the menu for each of them. Press the red button twice in the same location to return to the menu.

Cursor Position. This function allows the user to learn the coordinates of the graphic cursor in inches measured from the lower left corner of the drawing.

Erase Last Item. This function erases the last simple item placed on the drawing. If the last item placed on the drawing was a macro, the last element placed as part of the macro will be erased.

Erase Last Macro Item. This function erases the last macro placed on the drawing. Simple elements placed after the last macro will be erased. IF THERE ARE NO MACROS ON THE DRAWING, THE ENTIRE DRAWING WILL BE ERASED.

Erase Mode(with Zoom). When the erase mode is called for, the system first goes through the process of asking whether the user wants to restrict the view of the drawing to which the erase mode will be applied, since this

usually reduces the time needed to erase the necessary items. The screen is then cleared and the elements of the drawing are put on the screen one at a time. After each element is drawn, the user decides if it should be erased. If it is to be erased, the item is wiped off the screen and the user is asked to verify the action. If the user indicates that it should not be erased, the item is redrawn and the system continues with the next element. Macro elements are considered to be an entity in this routine and are erased as such.

Zoom. This is a photographic term which refers to the process of viewing a smaller part of a scene. This function allows the user to fill the screen with a small part of a drawing, thereby permitting more detail to be displayed. The user indicates the lower left corner and the upper right corner of the desired view. The lower left corner of the new view will be the grid point nearest to the point specified by the user. The upper right corner of the new view will be chosen so that the upper right corner specified will be included in the new view. The new view will include as small an area of the drawing as possible, and the horizontal and vertical distance between dots on the screen will be a power of two. The user can zoom in on any part of the view on the screen. For example, in Figure 3, if P1 is indicated as the lower left corner of a new view

and P2 is indicated as the upper right, then Figure 4 will be the new view. If P3 is then chosen to be the lower left of a new view and P4 is the upper right, Figure 5 will be the new view.

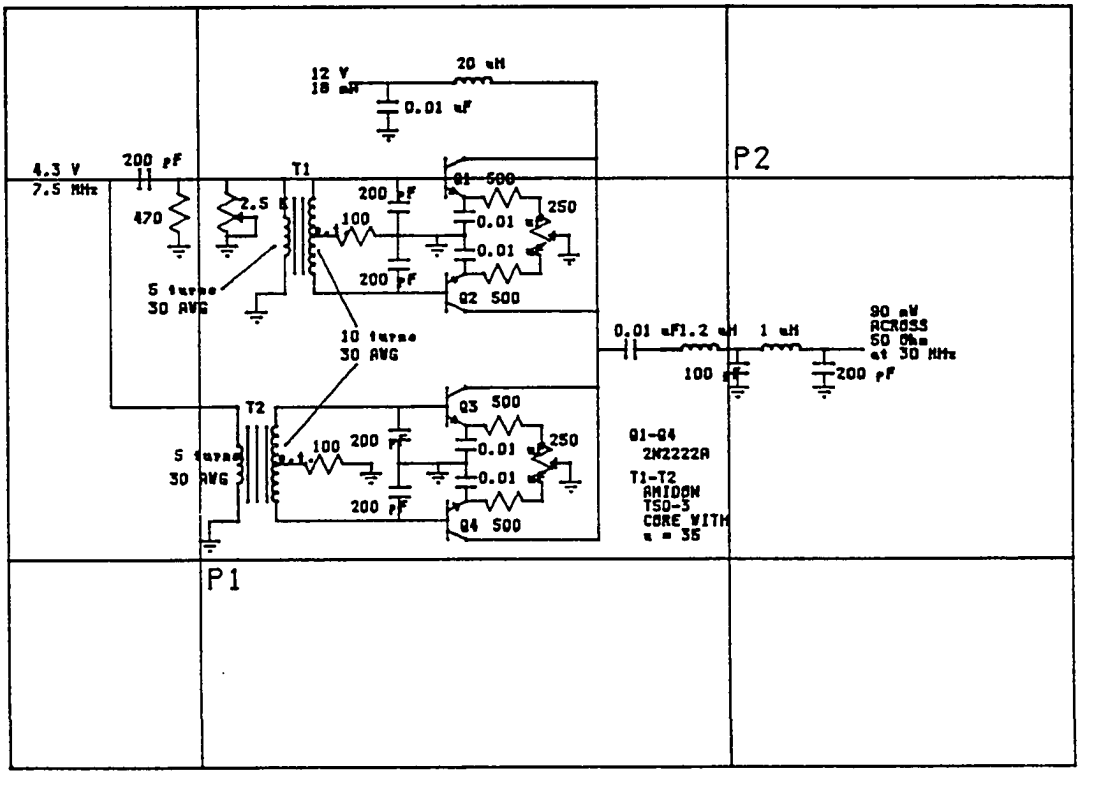


Figure 3. A Drawing At Full-Size

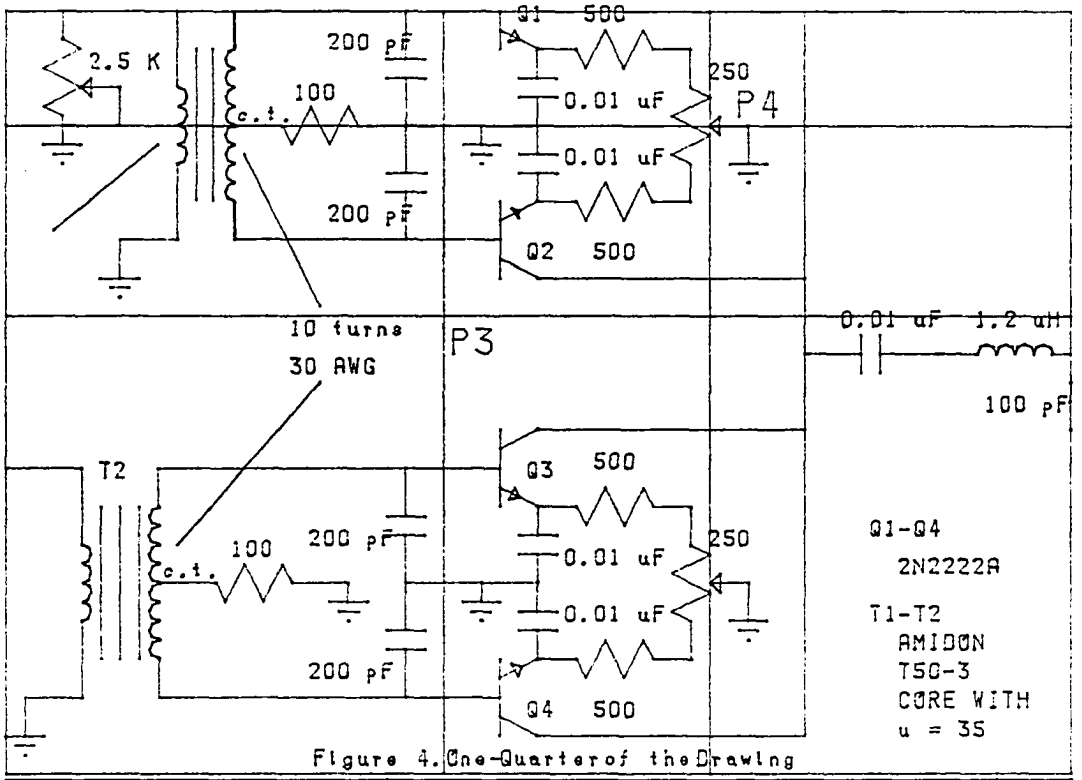


Figure 4. One-Quarter of the Drawing

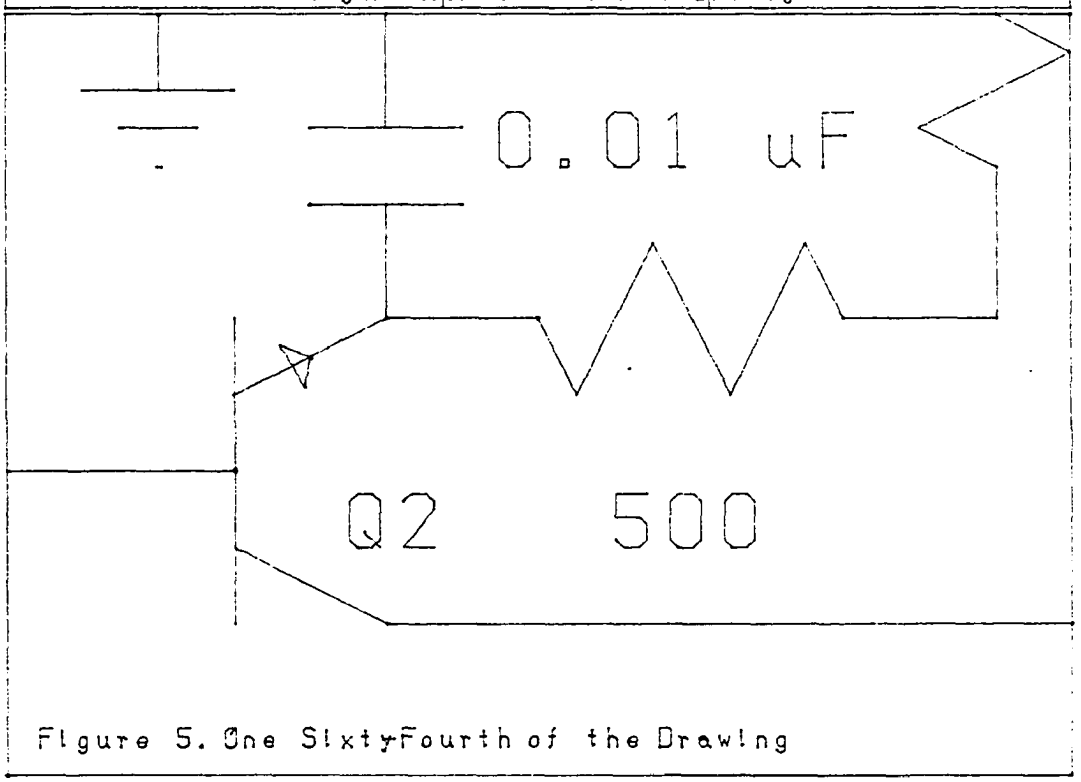


Figure 5. One Sixty-Fourth of the Drawing

Wide Angle. This is a photographic term which refers to the process of viewing a larger part of a scene. This function simply draws the entire schematic on the screen.

Quit. This function returns the system to the function menu.

Dashed Line Mode. All the elements put on the drawing in this mode will be plotted with broken lines. They will, however, appear broken only on the plotter and not on the video screen, therefore, it is important that this mode be used with caution.

Dashed Line Off. This function puts the system back into the solid line mode.

Plot a Drawing

MADS asks for the name of the drawing to be plotted and checks to see that it is in the file before displaying the menu shown in Figure 6. This menu operates in the same way as the others in this system. A plot operation can be temporarily halted by pressing the escape(ESC) key. It may have to be pressed several times before the plotter will respond.

Quit. This entry returns the system to the function menu.

```
WHAT DO YOU WANT TO DO?
POSITION THE CURSOR AND PRESS ENTER

( ) QUIT
( ) PLOT THE ENTIRE DRAWING
( ) PLOT THE DRAWING IN THE LOWER LEFT QUARTER
( ) PLOT THE DRAWING IN THE LOWER RIGHT QUARTER
( ) PLOT THE DRAWING IN THE UPPER LEFT QUARTER
( ) PLOT THE DRAWING IN THE UPPER RIGHT QUARTER
( ) SPECIAL HANDLING
```

Figure 6. Plot Function Menu

Plot the Drawing in Quarters. The drawing will be scaled and plotted on the indicated part of the plot bed.

Special Handling. The user will be prompted through a procedure in which any part of the drawing can be plotted on any part of the plot bed. This feature allows drawings to be scaled up or down to fit specific spaces.

Start a New Drawing

In this function, the system asks for the name of the new drawing and verifies that it is a new drawing before going to the same program used in the Edit a Drawing function.

Define a Drawing Element

This is the function in which macro elements are defined and constructed. The user must enter the name of the element using not more than nine printable characters. If the entered name is not already in the macro list, the system displays a drawing element menu, shown in Figure 7, which is quite similar to the one used in editing a drawing. The differences between these menus are due to the differences in the tasks to be accomplished with each of them. A macro element, for example, cannot be part of another macro; therefore, there are no macro references in the menu used to define macros.

ELEMENT MENU

( ) REFERENCE DOT	( ) TOP HALF CIRCLE
( ) CURSOR POSITION	( ) LEFT HALF CIRCLE
( ) LINE	( ) BOTTOM HALF CIRCLE
( ) RECTANGLE	( ) RIGHT HALF CIRCLE
( ) CIRCLE	( ) UPPER RIGHT 1/4 CIRCLE
( ) HORIZONTAL TEXT	( ) UPPER LEFT 1/4 CIRCLE
( ) VERTICAL TEXT	( ) LOWER LEFT 1/4 CIRCLE
( ) ERASE LAST ITEM	( ) LOWER RIGHT 1/4 CIRCLE
( ) QUIT	

POSITION CURSOR AND PRESS ENTER

Figure 7. Macro Element Menu

CHAPTER III

TECHNICAL DETAILS

Hardware

The microcomputer used in the construction of this drafting system is a SS50C bus system manufactured by Southwest Technical Products in San Antonio, Texas. The basic computer uses a Motorola MC6809 microprocessor as a central processor and includes 56 kilobytes of random access memory (RAM), two eight-inch flexible-disk drives, and a dumb terminal. The standard practice of refering to 1024 bytes as a kilobyte is used in this paper.

Additional equipment which was necessary to fulfill the drafting functions of the system includes a digital plotter and a video display monitor. Furthermore, it was also necessary to replace 8 kilobytes of the original computer RAM with a circuit board containing 6 kilobytes of graphic RAM and 2 kilobytes of standard RAM. This graphic memory board was designed and constructed by the author. A schematic of the board is shown in Appendix I.

The design of the graphic RAM circuit is relatively standard practice and it is quite reliable although somewhat old-fashioned. In this design, 6144 bytes of RAM are displayed on a video screen under the control of a Motorola MC6845 CRT controller. This is done in a "bit-mapped" fashion in which each bit of the memory corresponds to an individual point on the screen that will be light if the bit is a one, and dark if the bit is a zero. This arrangement yields a graphic display with 256 points, usually called pixels for picture elements, in the horizontal direction, and 192 pixels in the vertical direction. The microprocessor has priority over the CRT controller and can read or write the graphic memory at any time. This makes the system more efficient but causes a small amount of interference on the screen during line drawing. The graphic cursor is constructed entirely with hardware components, and no software support is required to hold or display the cursor location. The horizontal and vertical outputs of a trackball drive up-down counters which contain the addresses of the horizontal and vertical cursors. The program can determine the states of these counters by reading two input ports, which is, in this case, a simple memory read operation because the MC6809 uses memory-mapped input and output.

Software

All of the MADS programs run under the Flex operating system which comes from Technical System Consultants (TSC) of Chapel Hill, North Carolina, and are written in TSC's extended Basic Precompiler or 6809 Mnemonic Assembler. The programs that interact with the user and the programs that deal with the disk files are all written in Basic while Assembler is used for the graphic video screen drawing routines. The Assembler subroutines require almost 4 kilobytes of memory after conversion to machine language and are loaded into the computer memory immediately after the loading of the operating system. These routines are not moved or changed in any way by the Basic programs although parameters are transfered in both directions.

Appendix II is a list of all the subroutine names with their functions used in the system. Appendix III contains complete listings of all the programs and subroutines used in the system.

Basic Programs

As can be seen from the system flowchart in Figure 8, MADS makes extensive use of the chaining capability of Basic. In a system that supports chaining, a program that is

being executed can request that another program be loaded from the disk. The second program is loaded into the space previously occupied by the first program and executed. Chaining allows a system as complicated as MADS to run on a computer as small as this one; i.e., one with 38 kilobytes of memory, which is what is left for Basic out of the original 56 kilobytes after reserving 8 kilobytes for Flex, 6 kilobytes for graphics, and 4 kilobytes for Assembler.

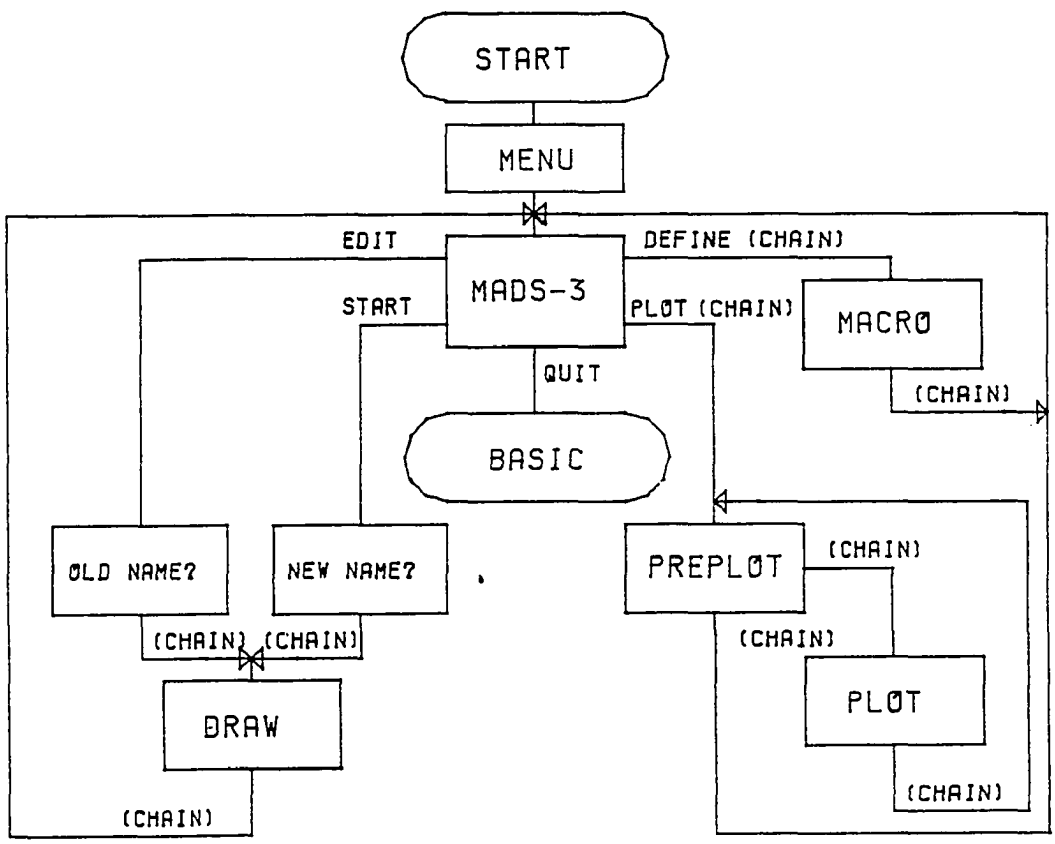


Figure 8. System Flow Chart

Mads-3. This program generates the function menu (Figure 1, Chapter II), collects parameters from the user, and calls the other programs.

Draw. This program generates the drawing element menu (Figure 2, Chapter II), and interacts with the user to form the drawing file.

Preplot. This program generates the plot menu (Figure 6, Chapter II), and interacts with the user to collect plot parameters.

Plotter. This program plots a drawing.

Macro. This program generates the macro element menu (Figure 7, Chapter II), and interacts with the user to form the macro element files.

Menu. This program draws the first twelve macro elements across the top and down the right side of the graphic video screen.

Disk Files

As many as four disk files can be active at any time in MADS. The most simple one is used for passing parameters between Basic programs during chaining and is named PARA. The other three are more complicated and

contain information pertaining to either individual drawings or to macro elements. These three are: the macro directory (MAC_DIR), the macro elements (MAC_ELEM), and the drawing being developed or plotted, which is named by the user. All three of them are organized into records with each record containing fourteen sub-records, and with each sub-record containing one element of the file. In each of these files an element is represented by a ten-byte alphanumeric field and four sixteen-bit parameters. The parameters have different meanings for different elements. For example, if the element is a line, the four parameters contain the coordinates of the two end-points of the line. If the element is a rectangle, the parameters are the coordinates of the lower left corner and the base and altitude of the rectangle. In the drawing file and the macro element file, almost all the file elements are also drawing elements. Each sub-record of the macro directory file, however, contains the name of the macro, the record number of the macro element file where the description of the macro begins, the sub-record number of the first element of the macro description, and a code that indicates whether the macro is to have connections to the external circuit.

Coordinates

The user's perception of the coordinate system associated with a drawing in MADS is one in which the lower left corner of the drawing is the zero point in both the horizontal and vertical directions. A typical drawing is 14 inches wide and 10 inches tall, although they can be 28 inches wide and 20 inches tall. The plotter is an integer machine that has a coordinate system based on tenths of a millimeter. It ranges from zero to 3600 in the horizontal direction and from zero to 2600 in the vertical direction. There are 254 tenths of a millimeter in one inch; this is a fortunate coincidence because multiplying a parameter in the user's coordinate system by 256 will yield a number that is quite close to the coordinate system of the plotter. We are, of course, more interested in 256 than in 254 because 256 is a power of two and provides the basis for a purely binary coordinate system in which most calculations can be done with integer arithmetic. For this reason, the drawing coordinates are stored in the disk file as sixteen-bit numbers with the binary point in the middle. This yields seven bits of positive integer data and eight bits of fractional data. A drawing parameter, as stored in the file, is bounded by zero and 128 inches because the largest number that can be stored in the file is 32767. The drawing has a resolution of 0.00390625 inches. In practice,

however, the upper bound on the size of a drawing is 28 inches and the largest number that will be stored on the disk is 7168.

The user must perceive himself as having the ability to move the graphic video screen close to any point on the drawing to see greater detail. To provide this perception, the scaling factor which relates the screen coordinates to the file coordinates must be a variable which can be changed, at least indirectly, by the user. Furthermore, since binary calculations are convenient, it is useful to employ the old English binary system of measures in which an inch is successively divided by two to obtain more precise measurements. This was implicit in the decision to scale the user's coordinates into file coordinates by multiplying them with 256. It happens that 224 pixels in the horizontal and 160 pixels in the vertical directions are ideal for representing parts of an inch ranging from 256 pixels to the inch to 8 pixels to the inch. This choice of screen coordinates also provides space at the top and right side of the screen to display some of the macro elements drawn at a scale of 32 pixels to the inch. The video screen coordinates range from zero to 255 pixels in the horizontal direction and from zero to 191 in the vertical direction.

Drawing On The Video Screen

The flowchart of the program Draw, which performs the graphic video screen drawing and interacts with the user to form the drawing file, is shown in Figure 9.

Macro Elements. When the user requests a macro element, MADS prints the names of all the macros on the text screen with a number associated with each of them. The user enters the number of the desired macro with the keyboard and MADS enters the name of the macro as well as the number of elements in the macro into the drawing file. The individual elements of the macro are then brought out of the macro element file, scaled and translated, drawn on the graphic screen, and entered into the drawing file.

Zoom and Wide Angle. Both of these functions require that the graphic video screen be thought of as a local coordinate system located within a global coordinate system which is the drawing. The video screen is expanded and contracted with respect to the drawing and is moved around the drawing surface as though it were a window. The wide-angle function is quite straight forward in that it calls for the entire drawing to appear on the screen. The zoom function is more interesting. Determining the area that the user wants to have expanded is a matter of asking the user an appropriate set of questions. From the

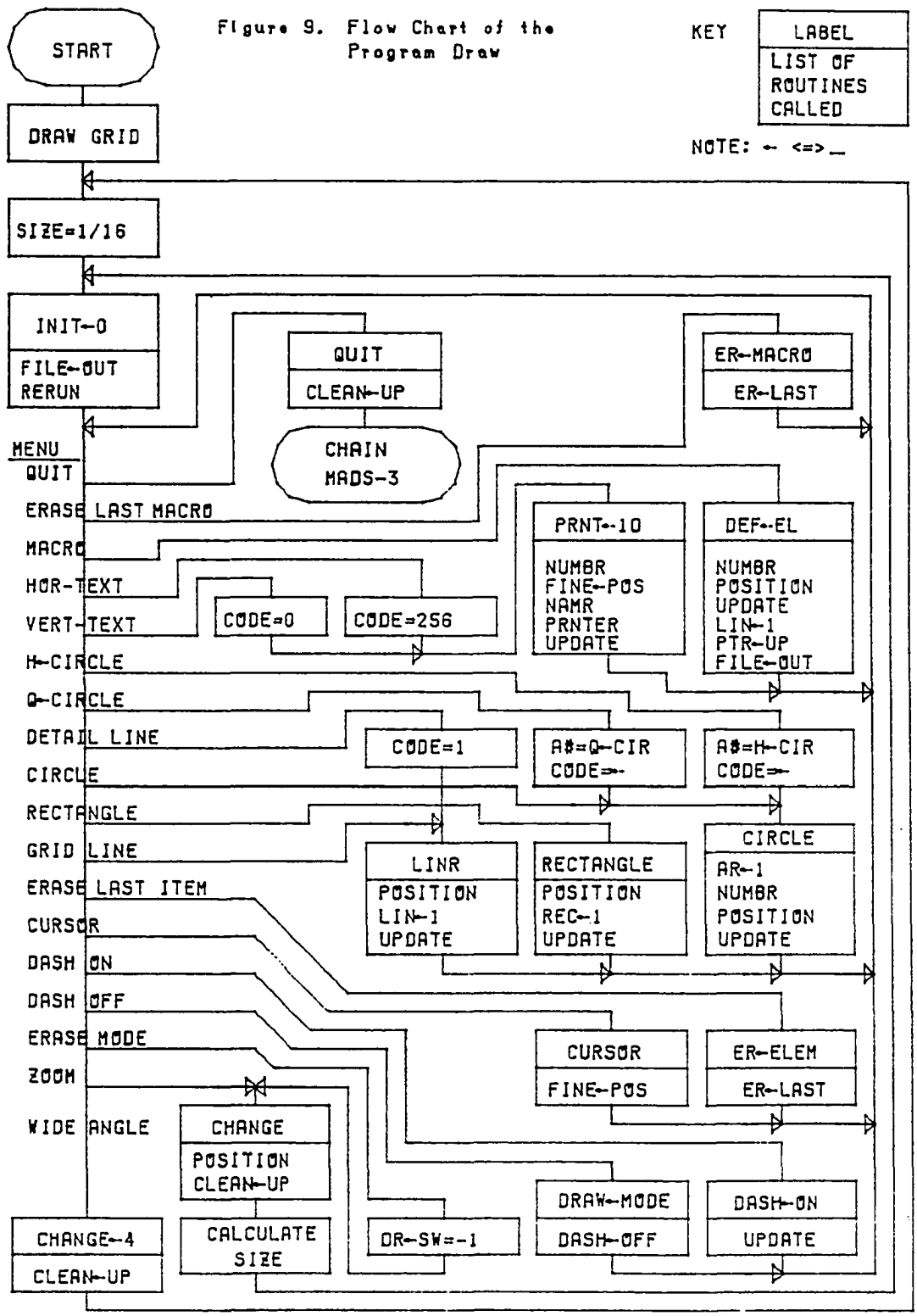


Figure 9. Flow Chart of the Program Draw

boundaries of that area the appropriate scaling and translation factors can be calculated with geometry. Then it is necessary to determine, for each drawing element in the file, whether any or all of an element is contained in the area to be displayed. The problem can best be illustrated by an example. Consider a straight line. Its position on the drawing is remembered by storing its end points. It is necessary to determine whether the line crosses the area of the drawing which coincides with the graphic screen, and if it does, to determine the end points of the visible line in graphic screen coordinates. Fortunately, both of these tasks can be accomplished more or less simultaneously by a subroutine which executes quickly most of the time although it is rather large. This routine was also not difficult to write although it was tedious. The program listings in Appendix IV contain this subroutine in Basic in the program titled Plotter and in Assembler in the collection of subroutines titled Main.

Erase Last Item. Each time that a drawing element is added to the drawing, a pair of variables are incremented. These variables contain the number of the last record and the number of the last record element. When the Erase-Last-Item function is commanded, the last item put into the drawing file is processed as though it were being drawn on the screen in the normal fashion except that the

line which is drawn is dark, not light. The pointer variables are then decremented, thereby removing the element from the file. There is a hazard involved with the use of this function. If it is requested when the last element of a macro is the last element in the drawing file, that element of the macro will be erased. If a simple element is the next addition to the drawing, that element will be treated as part of the macro by the system. If a macro element is the next addition to the drawing, the macro name will become part of the previous macro. The individual elements of the added macro will then become simple elements of the drawing and will no longer be treated as an associated group by the system. This hazard is not a problem but it can be quite confusing to the user during an erase-mode operation.

Erase Last Macro Item. This function utilizes the same subroutine used by the Erase-Last-Item function. It simply continues to call the subroutine and decrement the last element pointer until it finds a drawing element name that is not specifically defined within the system.

Erase Mode. This is the most complicated function to program in the entire project, primarily because it cannot be written as a separate routine but has to be incorporated into the routine which brings elements out of

the file and draws them on the screen. This is done in the system of programming logic sometimes called the "bucket full of worms" method. This method is hard to use and debug, but can produce small programs if properly applied. Such a drastic step is considered necessary because the final version of the program which requires the erase mode virtually fills all the available memory space.

When the erase mode is requested, the system first executes the zoom function because it has been learned through experience that the erase operation is much more efficient if the drawing area to be considered is restricted as much as possible. MADS draws each element of the drawing on the screen and asks the user if it is to be erased. If the answer is 'yes', the element is removed from the screen and the user is asked to verify this action. If the answer is again 'yes', the element is removed from the drawing file. If the answer to the first question is 'no', the system moves on to the next element. If the answer to the second question is 'no', the element is redrawn on the screen before the system moves on to the next element. Macro elements are treated in the same way as single elements in that the entire macro element is drawn and wiped out and redrawn if necessary.

Plotting a Drawing

Two Basic programs do the plot functions. One of them, PREPLOT, communicates with the user to collect parameters to be used during the plot operation. While collecting parameters, it is sometimes necessary for PREPLOT to draw on the graphic video screen. While performing the special handling function, for example, PREPLOT requests that the user indicate the boundaries of the drawing area to be plotted. PREPLOT draws a rectangle around the area indicated and asks the user to verify this action before continuing with the function. The other one, PLOTTER, uses the parameters supplied by PREPLOT to plot the drawing. Since one of the features of this system is the ability to plot any part of the drawing on any part of the plot bed, the dynamic range of the calculations required of the program PLOTTER can be quite large. For this reason, floating point calculations are extensively utilized in PLOTTER.

CHAPTER IV

CONCLUSION

Summary

The microcomputer-aided drafting system described in this paper is easily learned and used. Most of the communication with the user is done through menus and ample opportunity is provided for recovery from mistakes. An important feature of MADS is the ability to draw a collection of simple circuit elements and name the collection to form a macro element which can then be treated as though it were a simple element. Another feature is the ease with which the user can implement the standard electronic drafting practice of drawing connecting lines on a regularly spaced grid system to insure a uniform appearance. The primary advantage of this system over manual methods is the ease with which drawings can be updated with MADS as compared with manual methods.

The resolution of the graphic video screen is not especially good although it is sufficiently high to cause

few problems for the user during the development of a drawing.

The primary shortcoming of MADS is the speed with which it draws on the graphic video screen. Since drawing elements must be stored on the flexible disk due to RAM restrictions, quite a lot of time may be needed to complete a complex drawing on the screen.

Future Possiblities

The speed problem could be improved by at least an order of magnitude by installing a rigid-disk drive and compatible operating system. This action, however, would increase the total cost of the system by approximately fifty percent which somewhat contradicts the original purpose of the project. The drawing rate would be increased even more dramatically by rewriting all the Basic programs in Assembler. If the program which now occupies 38 kilobytes could be reduced to 10 or 12 kilobytes, which is a conservative estimate, enough space would be left in RAM to store a drawing of at least 1500 elements. There would be a delay while the drawing was transfered from disk to memory at the start of an edit and from memory to disk at the end of an edit, but intermediate operations such as zoom and erase should be extremely fast. Another approach to

speeding up intermediate operations while waiting for the flexible disk at the start and finish of the edit is based on adding data RAM to the computer. RAM could be mounted on a circuit board which would plug in to the secondary input-output bus of an SS50C computer. Three of the four addresses associated with each input-output port would be used to transfer address and data to and from the added RAM. The fourth address would be used to write commands to the RAM unit and read the status of the RAM unit. This approach would allow the development of a drawing containing at least 4000 elements if 64 kilobytes of RAM were used.

BIBLIOGRAPHY

SELECTED REFERENCES

Dent, Joseph B.; Devens, W.George; Marvin, Frank F.; and Trent, Harold F. Fundamentals of Engineering Graphics. New York: Macmillan, 1983.

Harrington, Steven. Computer Graphics. New York: McGraw-Hill, 1983.

Leventhal, Lance. 6809 Assembly Language Programming. New York: Osborne/McGraw-Hill, 1981.

McCormick, Ernest J. Human Factors in Engineering Design. New York: McGraw-Hill, 1976.

M6800 Microprocessor Applications Manual. Phoenix, AZ: Motorola, 1975.

The Complete Motorola Microcomputer Data Library. Phoenix, AZ: Motorola, 1978.

Newman, William M. and Robert F. Sproull. Principles of Interactive Computer Graphics. New York: McGraw-Hill, 1979.

Shneiderman, Ben. Software Psychology. Cambridge, MA: Winthrop, 1980.

Extended Basic User's Manual. Chapel Hill, NC: Technical Systems Consultants Inc., 1979.

Basic Precompiler User's Manual. Chapel Hill, NC: Technical Systems Consultants Inc., 1979.

FLEX Assembler. Chapel Hill, NC: Technical Systems Consultants Inc., 1980.

The MOS Memory Data Book. Houston: Texas Instruments, 1980.

Tuma, Jan J. Engineering Mathematics Handbook. 2nd ed. New York: McGraw-Hill, 1979.

Instruction Manual for DIGI-PLOT Model WX4671. Tokyo: Watanabe Instruments Corp., 1980.

APPENDIX I

SCHEMATIC

The schematic on the next two pages is of the graphic RAM and associated circuitry which was designed and constructed as part of this project. The schematic was prepared on this system and would normally be plotted on a larger sheet of paper. Its size is reduced here, which is a feature of MADS.

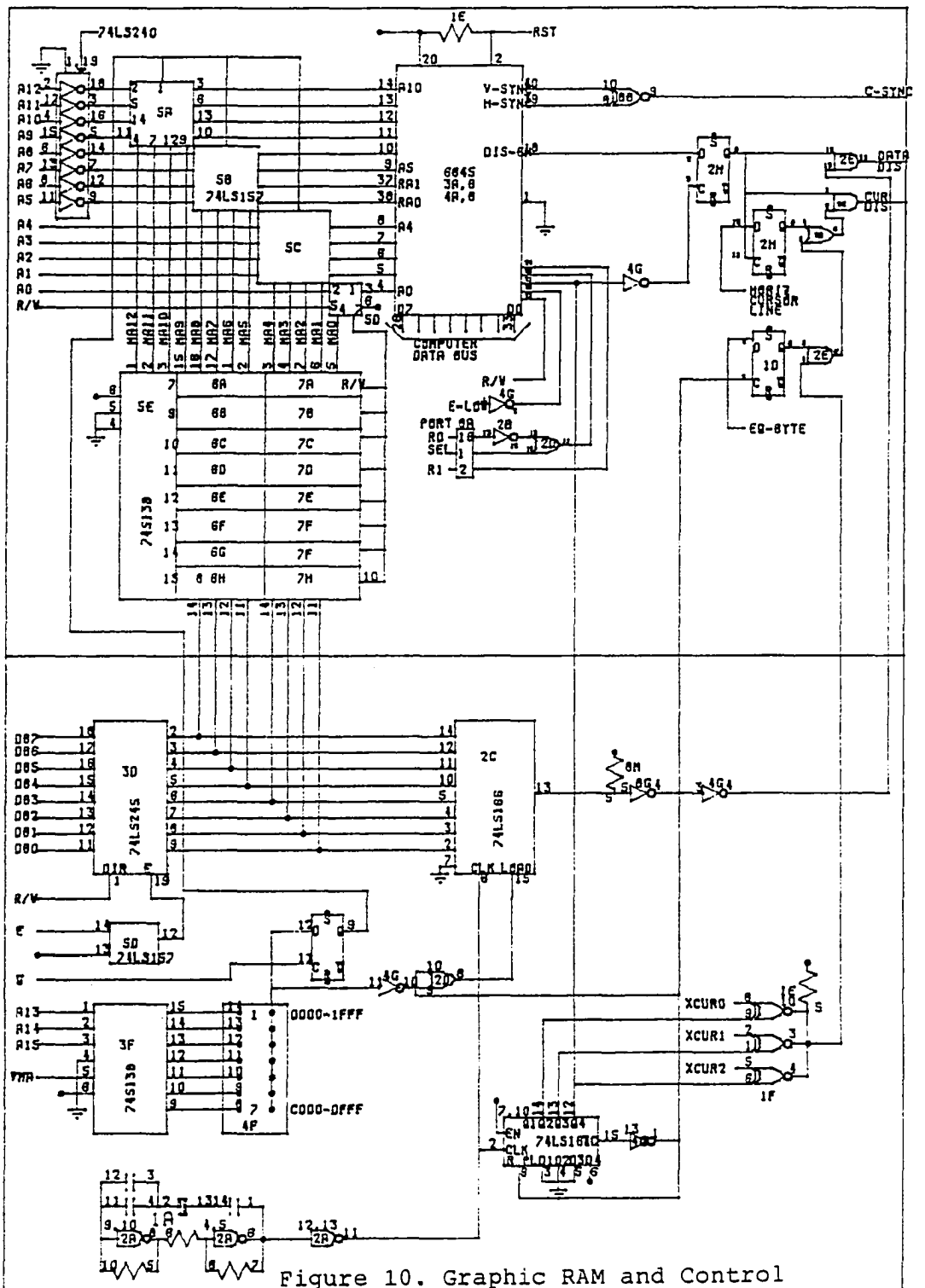
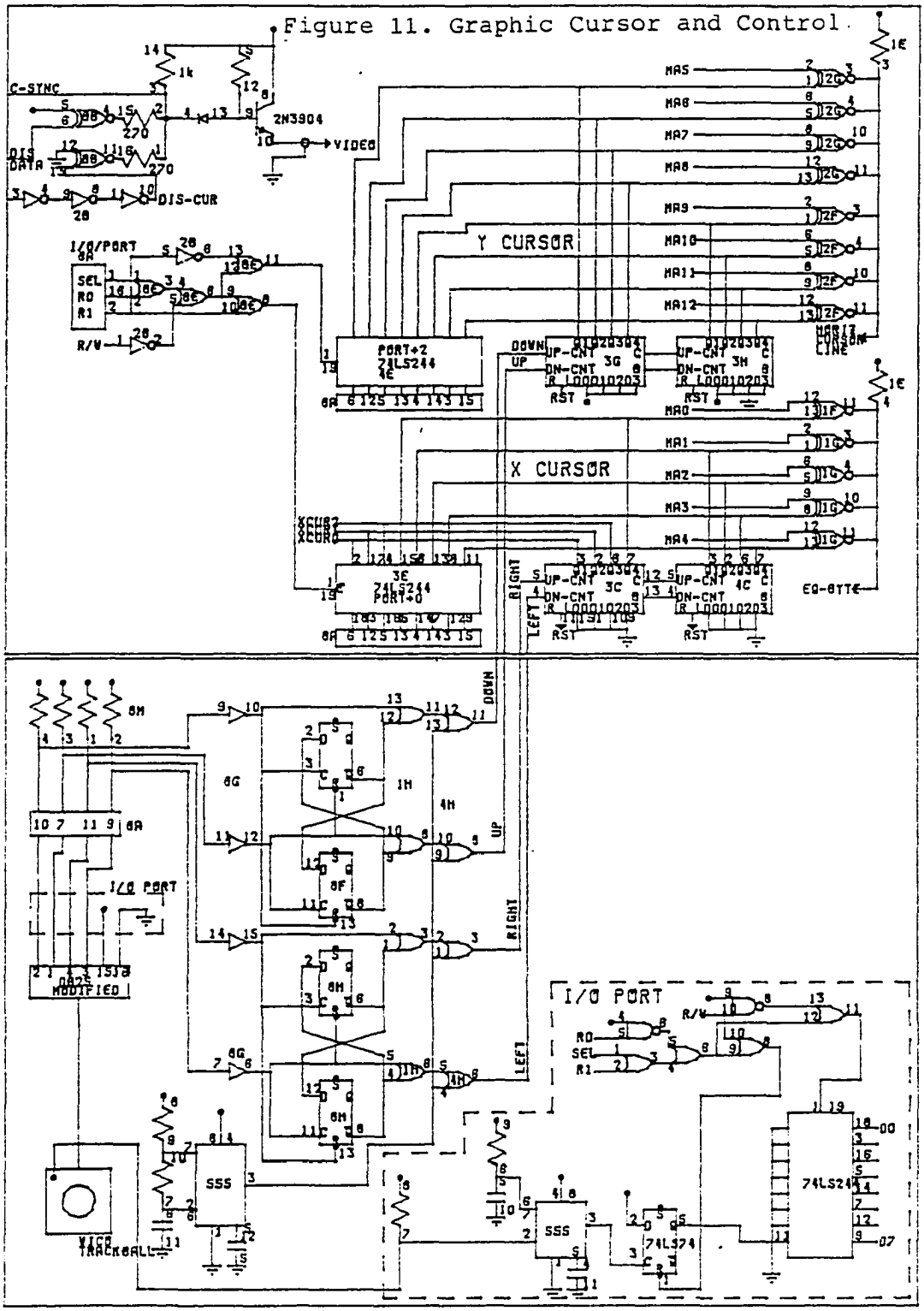


Figure 10. Graphic RAM and Control

Figure 11. Graphic Cursor and Control



APPENDIX II

SUBROUTINES IN MADS

Subroutines Written In Basic

Ar_1. This routine determines whether any part of an arc appears on the video screen. It calls assembler routines Q_cir, H_cir, and Cir.

Clean up. This routine fills the current sector of the drawing file with "0"s.

Dash off. This routine writes "DASH_OFF" into the current file element name. It calls the Basic routine Update.

Er last. This routine erases the last item in the drawing file from the screen and from the file. It calls the Basic routine File_out.

File out. This routine decodes file element names and starts macro operations. It calls the Basic routines Lin_1, Rec_1, and Ar_1 as well as the assembler routine Prnter.

Fine pos. This routine reads the graphic cursor position without rounding. It calls the Basic routine Red_b.

Lin 1. This routine is a connection to the window routine. It calls the Basic routine Win_con.

Namr. This routine collects valid alphabetic parameters.

Numbr. This routine collects valid numeric parameters.

Position. This routine reads the graphic cursor position and rounds up to the nearest grid point. It calls the Basic routine Red_b.

Prnter. This routine draws a rectangle on the screen to indicate the area to be occupied by printed material and prints a label in the rectangle if the label will fit. It calls the Basic routine Rec_1 and the assembler routine Prntr.

Ptr up. This routine updates the drawing file element pointer in macro operations.

Rec 1. This routine draws rectangles on the screen. It calls the Basic routine Win_con.

Red b. This routine reads the red button on the trackball.

Rerun. This routine controls indexing through the drawing element file on some macro operations.

Update. This routine writes the current element name and parameters into the drawing file and increments the file pointers.

Win con. This routine checks lines for boundary conditions. It calls the assembler routine Windo.

Subroutines Written In Assembler

Math Package. This is a collection of routines that perform several mathematical functions with forty-eight bit numbers. The functions performed include addition, subtraction, multiplication, division, and square-root extraction. Numerous calls are made within these routines.

Window. This routine determines whether any part of a line should appear on the graphic screen and the values of the end points of the visible portion of the line in screen coordinates. It calls Graph as well as several of the routines in the mathematics package.

Graph. This routine plots a straight line on the video screen. Input values must be in screen coordinates. It calls Plot, Horiz, Vert, and Divide.

Plot. This routine plots a point on the video screen. Input values must be in screen coordinates. It calls Find.

Find. This routine determines the address of the byte into which a one must be written to plot a point on the screen and the bit which must be turned on in that byte.

Horiz. This routine plots a horizontal line on the video screen. It calls Find.

Vert. This routine plots a vertical line on the video screen. It calls Find.

Divide. This routine calculates an unsigned 8-bit fractional quotient.

Clear. This routine clears the entire screen.

Wipe. This routine clears the drawing part of the screen.

Prntr. This routine prints alphanumeric characters on the video screen. It calls Graph.

Cir. This routine supervises the drawing of a full circle on the screen. It calls C_init, Sid_op, Quad_1, Quad_2, Quad_3, and Quad_4.

H cir. This routine supervises the drawing of a half circle on the screen. It calls C_init, Sid_op, Quad_1, Quad_2, Quad_3, and Quad_4.

Q cir. This routine supervises the drawing on a quarter circle on the screen. It calls C_init, Sid_op, Quad_1, Quad_2, Quad_3, and Quad_4.

C init. This routine scales the radius and the center point values from file coordinates to screen coordinates. It calls Scale and L_mult in the mathematics package.

Quad 1, Quad 2, Quad 3, and Quad 4. Each of these routines calculates the points of a circle in one quadrant. These routines call Plot.

APPENDIX III

PROGRAM LISTINGS

Programs in Basic

Mads_3

```
$SCALE  6
**
          POINTR = DPEEK(52267) - 2
          CLEAR = HEX("9C7D")
          EXEC, "TTYSET,WD=0,DP=0"
*         GOSUB SCRN_WIP
*
*
**
DRIVER    PRINT CHR$(26)
          FOR I% = 1 TO 8
               PRINT
          NEXT I%
          PRINT TAB(25);'HERE ARE YOUR CHOICES '
          PRINT TAB(25);' ( ) QUIT '
          PRINT TAB(25);' ( ) EDIT AN OLD DRAWING '
          PRINT TAB(25);' ( ) PLOT A DRAWING '
          PRINT TAB(25);' ( ) START A NEW DRAWING '
          PRINT TAB(25);' ( ) DEFINE A DRAWING ELEMENT '
          PRINT TAB(19);\
               'POSITION CURSOR ON CHOICE AND PRESS RETURN';
          FOR I% = 1 TO 34
          PRINT CHR$(08);
          NEXT I%
          FOR I% = 1 TO 5
               PRINT CHR$(11);
          NEXT I%
```

```
DR_5        UP% = 0: DN% = 0
DR_10       A$ = INCH$(0)
            IF A$ = CHR$(13) GOTO DR_40
            IF A$ <> CHR$(11) GOTO DR_20
                 UP% = UP% + 1
                 GOTO DR_10
DR_20       IF A$ <> CHR$(10) GOTO DR_10
                 DN% = DN% + 1
                 GOTO DR_10
DR_40       B% = 1 + DN% - UP%
            IF  B% = 1  GOTO  QUIT_F
            IF  B% = 2  GOTO  OLD_F
            IF  B% = 3  GOTO  PLOT_F
            IF  B% = 4  GOTO  NEW_F
            IF  B% = 5  GOTO  MACRO_F
            IF B% <= 0  GOTO DR_50
                 FOR I% = 1 TO B--1
                      PRINT CHR$(11);
                 NEXT I%
                 GOTO DR_60
DR_50        FOR I% = B%+1 TO I
                 PRINT CHR$(10);
            NEXT I%
DR_60       FOR I% = 1 TO 27
                 PRINT CHR$(12);
            NEXT I%
            GOTO DR_5
*
*
*
NEW_F       PRINT
            PRINT CHR$(26)
NEW_1       GOSUB NAMR
            IF CR_SW% = 1 GOTO DRIVER
            ON ERROR GOTO NEW_10
            A$ = A$ + '.DRW'
            OPEN OLD A$ AS 1
            GET #1, RECORD 1
            PRINT TAB(27); 'THIS FILE ALREADY EXISTS.'
            PRINT CHR$(27); CHR$(89)
            CLOSE 1
            GOTO NEW_1
NEW_1_5     PRINT
            FOR I% = 1 TO 10
                 PRINT
            NEXT I%
            PRINT TAB(24)7'WHICHDDRAWING SPACE DO YOU WANT?'
            PRINT TAB(33);'( ) 14"W BY 10"H'
            PRINT TAB(33);'( ) 28"W BY 20"H'
            PRINT TAB(22);\
                  'POSITION THE CURSOR AND PRESS RETURN';
```

```
            PRINT CHR$(11);CHR$(11);
            FOR I% = 1 TO 24
                 PRINT CHR$(8);
            NEXT I%
            UP% =0: DN% = 0
NEW_4       B$ = INCH$(0)
            IF B$ = CHR$(13) GOTO NEW_6
            IF B$ <> CHR$(11) GOTO NEW_5
                 UP% = UP% + 1
                 GOTO NEW_4
NEW_5       IF B$ <> CHR$(10) GOTO NEW_4
                 DN% = DN% + 1
                 GOTO NEW_4
NEW_6       B% = 1 + DN% - UP%
            IF B% > 0 AND B% < 3 GOTO NEW_8
                 PRINT CHR$(26);"LET'S TRY AGAIN";
                 GOTO NEW_1_5
NEW_8       FIELD #1, 10 AS ELEM$, 2 AS P0$, 2 AS P1$,\
                                    2 AS P2$, 2 AS P3$
            LSET ELEM$ = "0000000000"
            LSET P0$ = CVT%$(1):REM  RECORD #  [N%]
            LSET P1$ = CVT%$(0):REM  ELEMENT # [D_PTR%]
            LSET P2$ = CVT%$(B%)
            LSET P3$ = CVT%$(0)
            PUT #1, RECORD 1
            CLOSE 1
            OPEN NEW "PARA" AS 1
            PRINT #1, A$
            CLOSE 1
            CHAIN "0.DRAW"
NEW_10      IF ERR <> 4 THEN ON ERROR GOTO
            CLOSE 1
            OPEN NEW A$ AS 1
            RESUME NEW_1_5
*
*
***
OLD_F       PRINT
            GOSUB OLD_1
            IF CR_SW% = 1 GOTO DRIVER
            CHAIN "0.DRAW"
*
*
*
PLOT_F      PRINT
            GOSUB OLD_1
            IF CR_SW% = 1 GOTO DRIVER
            CHAIN "0.PREPLOT"
*
*
**
```

```
MACRO_F     PRINT
            ON ERROR GOTO MAC_10
            OPEN OLD "MAC_DIR" AS 1
            GET #1, RECORD 1
            CLOSE 1
MAC_5       CHAIN "0.MACRO"
MAC_10      IF ERR <> 4 THEN ON ERROR GOTO
            CLOSE 1
            OPEN NEW "MAC_DIR" AS 1
            FIELD #1,10 AS ELEM$,2 AS P0$,2 AS P1$,2 AS P2$
            LSET ELEM$ = 'ZZZZZZZZZZ'
            LSET P0$ = CVT%$(1)
            LSET P1$ = CVT%$(0)
            LSET P2$ = CVT%$(0)
            PUT #1, RECORD 1
            CLOSE 1
            OPEN NEW "MAC_ELEM" AS 1
            FIELD #1, 10 AS ELEM$
            LSET ELEM$ = '0000000000'
            PUT #1, RECORD 1
            CLOSE 1
            CHAIN "0.MACRO"
*
*
*
QUIT_F      EXEC, "TTYSET,DP=23,PS=Y"
            PRINT CHR$(26);\
                  'KEY IN "RUN" AND PRESS ENTER TO RETURN'
            END
*
*
SCRN_WIP    DPOKE POINTR,CLEAR
            DUM%=USR(0)
            RETURN
*
*
OLD_1       PRINT CHR$(26)
            EXEC, 'CAT,.DRW'
            PRINT
            PRINT
OLD_2       GOSUB NAMR
            IF CR_SW% = 1 THEN RETURN
            ON ERROR GOTO OLD_10
            A$ = A$ + '.DRW'
            OPEN OLD A$ AS 1
            GET #1, RECORD 1
OLD_5       CLOSE 1
            OPEN NEW "PARA" AS 1
            PRINT #1, A$
            CLOSE  1
            RETURN
```

```
OLD_10    IF  ERR<>4   THEN  ON ERROR GOTO
          CLOSE 1
          PRINT "File can not be found."
          RESUME OLD_2
*
NAMR      PRINT
          PRINT
          PRINT TAB(33);'DRAWING NAME?'
          PRINT
          FOR I% = 1 TO 35
               PRINT CHR$(12);
          NEXT I%
NAMR_5    A$ = ''
          CR_SW% = 0
          B$ = INCH$(0)
          IF B$ = CHR$(13) GOTO NAMR_40
          IF B$ > CHR$(64) AND B$ < CHR$(91) \
          OR B$ > CHR$(97) AND B$ < CHR$(123) \
          GOTO NAMR_10
            PRINT CHR$(10);
            PRINT 'THE FIRST CHARACTER MUST BE @LPHABETIC.';
            PRINT CHR$(11);
            FOR I% = 1 TO 40
                 PRINT CHR$(8);
            NEXT I%
            PRINT ' ';CHR$(8);
            GOTO NAMR_5
NAMR_10   A$ = A$ + B$
          I% = 2
NAMR_12        B$ = INCH$(0)
               IF B$ = CHR$(13) GOTO NAMR_40
               IF B$ <> CHR$(8)  GOTO NAMR_15
                    IF LEN(A$) <= 1  GOTO NAMR_5
                    A$ = LEFT$(A$,LEN(A$)-1)
                    PRINT  ' ';CHR$(8);
                    I% = I% - 1
                    GOTO NAMR_12
NAMR_15             IF B$ > CHR$(64) AND B$ < CHR$(91) \
                    OR B$ > CHR$(97) AND B$ < CHR$(123) \
                    OR B$ > CHR$(47) AND B$ < CHR$(58) \
                    OR B$ = CHR$(45) OR  B$ = CHR$(95) \
                    GOTO NAMR_20
                         PRINT CHR$(10);CHR$(10);
                         PRINT ' LETTERS  NUMBERS  -  AND _';
                         PRINT CHR$(11);CHR$(11);
                         FOR J% = 1 TO 28
                              PRINT CHR$(8);
                         NEXT J%
                         PRINT ' ';CHR$(8);
                         GOTO NAMR_12
NAMR_20               A$ = A$ + B$
```

```
            I% = I% + 1
            IF I% <= 8 GOTO NAMR_12
            B$ = INCH(0)
            IF B$ = CHR$(13) GOTO NAMR_40
            IF B$ <> CHR$(8) GOTO NAMR_30
                 PRINT ' '; CHR$(8);
                 I% = I% - 1
                 GOTO NAMR_12
NAMR_30     PRINT CHR$(10)
            PRINT 'ONLY EIGHT CHARACTERS ARE ALLOWED '
            A$ = LEFT$(A$,8)
            PRINT 'THE NAME IS '; A$
            PRINT 'DO YOU WANT TO START OVER(Y-N)?';
            B$ = INCH$(0)
            PRINT
            IF B$ <> 'Y' THEN RETURN
                 GOTO NAMR
NAMR_40     PRINT
            IF LEN(A$) = 0 THEN CR_SW% = 1
            RETURN
            END
```

Draw

```
$SCALE  6
            DIM P%(3), ELEM$(13), P$(3,13), MP$(3,13) :\
            DIM M_ELEM$(13), NO_EL$(13), ROT_CN$(13) :\
            DIM NAM$(13), REC$(13), EL_PTR$(13) :\
            ON ERROR GOTO END_OF_FILE
            MESSAGE$ = \
            'REPEAT THE LAST POSITION TO RETURN TO THE MENU':\
            CUR_MESGE$ = \
             'POSITION THE GRAPHIC CURSOR \
             AND PRESS THE RED BUTTON AT THE ':\
            GOTO ENVIRN
END_OF_FILE'    IF ERR <> 7 THEN ON ERROR GOTO
                PRINT 'THIS DISKETTE IS FULL':\
                RESUME QUIT
DRAW_1      DR_SW% = 1:\
            MODE$ = 'SOLID LINE'
SW_TEST     CODE% = 0:\
            BND_SW% = 0
DRAW_OUT    PRINT CHR$(26):\
            POKE ERASE,0:\
            ERASE% = 0:\
            PRINT TAB(33); MODE$; ' MODE':\
            PRINT TAB(29); 'DELTA = 1/'; 256/SIZE%; ' INCH':\
            DN% = 1:\
            RIGHT% = 1:\
```

```
            PRINT  TAB(32); 'ELEMENT MENU':\
            PRINT
            PRINT TAB(20); \
                  'POSITION TEXT CURSOR AND PRESS ENTER ':\
            PRINT TAB(21); CHR$(27);'1';TAB(43);CHR$(27);'1':\
            PRINT:\
            PRINT TAB(20); '( ) GRID LINE'; \
                  TAB(41); '( ) TOP    HALF CIRCLE'
            PRINT TAB(20); '( ) RECTANGLE'; \
                  TAB(41); '( ) LEFT   HALF CIRCLE':\
            PRINT TAB(20); '( ) CIRCLE'; \
                  TAB(41); '( ) BOTTOM HALF CIRCLE'
            PRINT TAB(20); '( ) HORIZONTAL TEXT'; \
                  TAB(41); '( ) RIGHT  HALF CIRCLE'
            PRINT TAB(20); '( ) VERTICAL TEXT';\
                  TAB(41); '( ) UPPER RIGHT 1/4 CIRCLE':\
            PRINT TAB(20); '( ) DETAIL LINE';\
                  TAB(41); '( ) UPPER LEFT  1/4 CIRCLE'
            PRINT TAB(20); '( ) MACRO ELEMENT';\
                  TAB(41); '( ) LOWER LEFT  1/4 CIRCLE':\
            PRINT TAB(20); '( ) CURSOR POSITION'; \
                  TAB(41); '( ) LOWER RIGHT 1/4 CIRCLE'
            PRINT TAB(20); '( ) ERASE LAST ITEM';\
                  TAB(41); '( ) ERASE LAST MACRO ITEM':\
            PRINT TAB(20); '( ) ZOOM'; \
                  TAB(41); '( ) ERASE MODE(WITH ZOOM)'
            PRINT TAB(20); '( ) WIDE ANGLE'; \
                  TAB(41); '( ) DASHED LINE MODE':\
            PRINT TAB(20); '( ) QUIT';\
                  TAB(41); '( ) DASHED LINE OFF':\
            FOR I% = 1 TO 12:\
                 PRINT CHR$(11);:\
            NEXT I%:\
            PRINT TAB(20); '(';
DRAW_5      B$ = INCH$(0):\
            IF B$ >= CHR$(08) AND B$ <= CHR$(13)  GOTO DRAW_10
                 PRINT CHR$(8);:\
                 GOTO DRAW_5
DRAW_10     IF B$ = CHR$(13) GOTO DRAW_20
            IF B$ = CHR$(12) THEN RIGHT% = RIGHT% + 1
            IF B$ = CHR$(11) THEN DN% = DN% - 1
            IF B$ = CHR$(10) THEN DN% = DN% + 1
            IF B$ = CHR$(09) THEN RIGHT% = 21
            IF B$ = CHR$(08) THEN RIGHT% = RIGHT% - 1
            GOTO DRAW_5
DRAW_20     IF RIGHT% > 60 OR RIGHT% < 0 \
                 OR DN% > 12  OR  DN% < 1 GOTO DRAW_OUT
            PRINT CHR$(26):\
            IF RIGHT% > 20 GOTO DRAW_30
                 ON DN% GOTO \
                 LINR,RECTANGL,CIRCLE,PRNT_H,PRNT_V,DET_LINR,\
```

```
                DEF_EL,CURSOR,ER_ELEM,CHANGE,WIDE_A,QUIT
DRAW_30     IF DN% >= 5  GOTO DRAW_40
                P%(3) = DN% - 1:\
                A$ = 'H_CIR':\
                GOTO CRCL_10
DRAW_40     IF DN% > 8 GOTO DRAW_50
                P%(3) = DN% - 5:\
                A$ = 'Q_CIR':\
                GOTO CRCL_10
DRAW_50     ON DN%-8 GOTO ER_MACRO,ERASER,DASH_ON,DRAW_MODE
DET_LINR        CODE% = 1:\
            GOTO LINR
DEF_EL         PRINT CHR$(26); TAB(12); 'ELEMENT LIST':\
            PRINT:\
            CTR% = 0 :\
            CODE% = 18:\
            FOR M_DIR_REC% = 1 TO  LAST_M_DIR_REC%:\
                GET #3, RECORD M_DIR_REC%:\
                FOR M_DIR_PTR% =  0 TO 13:\
                    IF NAM$(M_DIR_PTR%) = '0000000000' OR \
                       NAM$(M_DIR_PTR%) = 'ZZZZZZZZZZ'  \
                       GOTO D_EL_5
                       PRINT TAB(12);\
                            CTR%; SPC(2); NAM$(M_DIR_PTR%):\
                       NO_MAC% = NO_MAC% + 1:\
                       CTR% = CTR% + 1:\
                       IF CTR% <= CODE% GOTO D_EL_5
                            CODE% = CODE% + 18:\
  PRINT 'KEY IN THE NUMBER OF THE ELEMENT AND PRESS ENTER':\
  PRINT 'OR PRESS ONLY ENTER TO CONTINUE ';:\
                            GOSUB NUMBR
                            IF CR_SW% = 0 GOTO D_EL_50
                              PRINT
D_EL_5          NEXT M_DIR_PTR%:\
            NEXT M_DIR_REC%:\
            PRINT 'KEY IN THE NUMBER OF THE ELEMENT';:\
            PRINT ' AND PRESS ENTER':\
            PRINT 'OR PRESS ONLY ENTER TO CONTINUE ';:\
            CR_SW% = 0:\
            GOSUB NUMBR
            IF CR_SW% = 1 GOTO DRAW_OUT
D_EL_50     PRINT:\
            B% = VAL(C$):\
            IF B% < 0 OR B% > NO_MAC% GOTO DRAW_OUT
D_EL_55     CTR% = 0:\
            FOR M_DIR_REC% = 1 TO  LAST_M_DIR_REC%:\
                GET #3, RECORD M_DIR_REC%:\
                FOR M_DIR_PTR% =  0 TO 13:\
                    IF NAM$(M_DIR_PTR%) = '0000000000'\
                    OR NAM$(M_DIR_PTR%) = 'ZZZZZZZZZZ'\
                    GOTO D_EL_60
```

```
                    IF CTR% = B% GOTO D_EL_70
                         CTR% = CTR% + 1
D_EL_60          NEXT M_DIR_PTR%:\
            NEXT M_DIR_REC%:\
            PRINT "CAN'T FIND NUMBER"; B%:\
            GOTO DRAW_OUT
D_EL_70     X0% = X_MAX% + 1: X_1% = X0%
D_EL_72     M_ELEM_REC% = CVT$%(REC$(M_DIR_PTR%)):\
            M_ELEM_PTR% = CVT$%(EL_PTR$(M_DIR_PTR%)):\
            NO_ELEM% = CVT$%(NO_EL$(M_DIR_PTR%)):\
            ROT_CN% = CVT$%(ROT_CN$(M_DIR_PTR%)):\
            GET #2, RECORD  M_ELEM_REC%:\
            PRINT CHR$(26); MESSAGE$
D_EL_75     A$ = NAM$(M_DIR_PTR%):\
            FOR K%=1 TO 3:P%(K%)=0:NEXT K%:\
            IF ROT_CN% < 256 GOTO D_EL_80
                 PRINT CUR_MESGE$; 'ELEMENT CENTER';:\
                 GOSUB POSITION
                 IF X0% = X% AND Y0% = Y% GOTO SW_TEST
                 X0%    = X%:\
                 Y0%    = Y%:\
                 P%(0) = NO_ELEM%:\
                 GOSUB UPDATE
                 CTR% = 1:\
                 GOTO D_EL_90
D_EL_80     PRINT CUR_MESGE$:\
            PRINT   'LEFT OR TOP CONNECTION';:\
            GOSUB POSITION
            IF X_1% = X% AND Y_1% = Y% GOTO SW_TEST
            X_1%    = X%:\
            Y_1%    = Y%:\
            DPOKE POINTR,PLOT:\
            DPOKE X_LOC,(X_1%-X_LOW%)/SIZE%:\
            DPOKE Y_LOC,(Y_1%-Y_LOW%)/SIZE%:\
            DUM% = USR(0):\
            PRINT 'RIGHT OR BOTTOM CONNECTION';:\
            GOSUB POSITION
            IF X_1% = X% AND Y_1% = Y% GOTO SW_TEST
            P%(0) = NO_ELEM% + 1:\
            GOSUB UPDATE
            X_2%    = X%:\
            Y_2%    = Y%:\
            X0% = (X_1% + X_2%)/2:\
            Y0% = (Y_1% + Y_2%)/2:\
            A$ = 'LINE':\
            P%(0) = X_1%:\
            P%(1) = Y_1%:\
            P%(2) = CVT$%(MP$(0,M_ELEM_PTR%)) + X0%:\
            P%(3) = CVT$%(MP$(1,M_ELEM_PTR%)) + Y0%:\
            GOSUB LIN_1
            GOSUB UPDATE
```

```
            P%(0) = X_2%:\
            P%(1) = Y_2%:\
            P%(2) = CVT$%(MP$(2,M_ELEM_PTR%)) + X0%:\
            P%(3) = CVT$%(MP$(3,M_ELEM_PTR%)) + Y0%:\
            GOSUB LIN_1
            GOSUB UPDATE
            GOSUB PTR_UP
            CTR% = 2
D_EL_90     A$ = M_ELEM$(M_ELEM_PTR%):\
            FOR J% = 0 TO 3:\
                  P%(J%) = CVT$%(MP$(J%,M_ELEM_PTR%)):\
            NEXT J%:\
            P%(0) = P%(0) + X0%:\
            P%(1) = P%(1) + Y0%:\
            IF A$ <> 'LINE'  GOTO  D_EL_120
                  P%(2) = P%(2) + X0%:\
                  P%(3) = P%(3) + Y0%:\
                  GOSUB LIN_1
                  GOSUB UPDATE
                  GOSUB PTR_UP
                  GOTO D_EL_300
D_EL_120    GOSUB FILE_OUT
            GOSUB UPDATE
            GOSUB PTR_UP
D_EL_300    CTR% = CTR% + 1:\
            IF CTR% <= NO_ELEM%  GOTO  D_EL_90
            GOTO D_EL_72
PTR_UP      M_ELEM_PTR% = M_ELEM_PTR% + 1:\
            IF M_ELEM_PTR% <= 13  THEN RETURN
            IF CTR% = NO_ELEM%  THEN RETURN
                  M_ELEM_PTR% = 0:\
                  M_ELEM_REC% = M_ELEM_REC% + 1:\
                  GET #2, RECORD M_ELEM_REC%:\
                  RETURN
ERASER      DR_SW% = -1:\
            IF MODE$=  'DASHED LINE' THEN GOSUB DASH_OFF
            PRINT CHR$(26):\
            GOTO CHANGE
DRAW_MODE   GOSUB DASH_OFF
            GOTO DRAW_1
DASH_ON     MODE$ = 'DASHED LINE':\
            A$ = 'DASH_ON':\
            GOSUB UPDATE
            GOTO DRAW_OUT
DASH_OFF    A$ = 'DASH_OFF':\
            GOSUB UPDATE
            RETURN
CURSOR      PRINT'PRESS THE RED BUTTON':\
            PRINT' FOR THE CURSOR POSITION AT A NEW LOCATION':\
            PRINT' AT THE SAME LOCATION FOR THE MENU':\
            X1% = X_MAX% + 1
```

```
CURS_10    GOSUB FINE_POS
           A = X%:\
           B = Y%:\
           A =  A/256:\
           B =  B/256:\
           IF X% = X1% AND Y% = Y1% GOTO DRAW_OUT
           PRINT CHR$(11); "X=";INT(A); (A-INT(A))*8; '/8';:\
           PRINT "  Y=";INT(B); (B-INT(B))*8; '/8';\
                 CHR$(27); CHR$(84);:\
           X1% = X%:\
           Y1% = Y%:\
           GOTO CURS_10
ER_ELEM    GOSUB ER_LAST
           PRINT:\
           PRINT 'SHALL WE ERASE ANOTHER ONE(Y-N)? ';:\
           B$ = INCH$(0):\
           IF B$ = 'Y' OR B$ = 'y' GOTO ER_ELEM
           GOTO SW_TEST
ER_MACRO   DR_SW% = -1
ER_M_10    GOSUB ER_LAST
           IF MACRO_SW% = 0 GOTO ER_M_10
                MACRO_SW% = 0:\
                GOTO DRAW_1
ER_LAST    IF N% > 1 GOTO ER_L_10
           IF D_PTR% = 0 GOTO SW_TEST
ER_L_10    ERASE% = 255:\
           A$ = ELEM$(D_PTR%):\
           FOR I% = 0 TO 3:\
                P%(I%) = CVT$%(P$(I%,D_PTR%)):\
           NEXT I%:\
           GOSUB FILE_OUT
           NO_ELEM% = NO_ELEM% - 1:\
           IF D_PTR% = 0  GOTO ER_L_20
                D_PTR% = D_PTR% - 1:\
                RETURN
ER_L_20    D_PTR% = 13:\
           PUT #1, RECORD N%
           N% = N% - 1:\
           GET #1, RECORD N%
           RETURN
LINR       PRINT CUR_MESGE$; 'START POINT';:\
           IF CODE%=0 THEN GOSUB POSITION ELSE GOSUB FINE_POS
           P%(0) = X%:\
           P%(1) = Y%:\
           S1% = 0:\
           DPOKE POINTR,PLOT:\
           DPOKE X_LOC,(P%(0)-X_LOW%)/SIZE%:\
           DPOKE Y_LOC,(P%(1)-Y_LOW%)/SIZE%:\
           DUM% = USR(0):\
           PRINT CUR_MESGE$; 'END';
LINE_R_0   IF CODE%=0 THEN GOSUB POSITION ELSE GOSUB FINE_POS
```

```
            PRINT CHR$(11);:\
            P%(2) = X%:\
            P%(3). = Y%:\
            IF P%(0) = P%(2) AND P%(1) = P%(3)  GOTO SW_TEST
LINE_R_2    GOSUB LIN_1
            IF   BND_SW% = 1  GOTO SW_TEST
            A$ = 'LINE':\
            GOSUB UPDATE
            P%(0) = P%(2):\
            P%(1) = P%(3):\
            IF S1% = 1 GOTO LINE_R_0
            PRINT:\
            S1% = 1:\
            PRINT 'PRESS THE RED BUTTON':\
            PRINT ' AT A NEW LOCATION FOR ANOTHER LINE':\
            PRINT ' AT THE SAME LOCATION FOR THE MENU';:\
            GOTO LINE_R_0
RECTANGL    PRINT CUR_MESGE$:\
            PRINT  "LOWER LEFT CORNER OF RECTANGLE ";:\
            GOSUB POSITION
            P%(0) = X%:\
            P%(1) = Y%:\
            DPOKE POINTR,PLOT:\
            DPOKE X_LOC,(P%(0)-X_LOW%)/SIZE%:\
            DPOKE Y_LOC,(P%(1)-Y_LOW%)/SIZE%:\
            DUM% = USR(0):\
            PRINT   'UPPER RIGHT CORNER ';:\
            GOSUB POSITION
            P%(2) = X%:\
            P%(3) = Y%:\
            P%(2) = ABS(P%(0) - P%(2)):\
            P%(3) = ABS(P%(1) - P%(3)):\
            GOSUB REC_1
            A$ = 'RECT':\
            IF  BND_SW% = 0  THEN GOSUB UPDATE
                GOTO SW_TEST
CIRCLE      P%(3) = 0:\
            A$ = 'CIRCLE'
CRCL_10     PRINT MESSAGE$
CRCL_20     PRINT"WHAT IS THE RADIUS?";:\
            GOSUB NUMBR
            PRINT:\
            IF CR_SW% = 0 GOTO CRCL_30
                 A$ = 'N6':\
                 P%(2) = 12:\
                 GOTO CRCL_40
CRCL_30     IF A < 255 AND A > 0 GOTO CRCL_35
                 PRINT 'THE RADIUS MUST BE <= 255':\
                 GOTO CRCL_20
CRCL_35     P%(2) = A*256
CRCL_40     PRINT CUR_MESGE$; "CIRCLE CENTER";:\
```

```
            GOSUB POSITION
            IF P%(0) = X% AND P%(1) = Y% GOTO SW_TEST
            P%(0) = X%:\
            P%(1) = Y%:\
            DPOKE POINTR,PLOT:\
            DPOKE X_LOC,(P%(0)-X_LOW%)/SIZE%:\
            DPOKE Y_LOC,(P%(1)-Y_LOW%)/SIZE%:\
            DUM% = USR(0):\
            GOSUB AR_1
            IF  DR_SW% <= 0  GOTO CRCL_40
                 IF DR_SW% = 1 THEN GOSUB UPDATE
                GOTO CRCL_40
PRNT_H      CODE% = 0:\
            GOTO PRNT_5
PRNT_V      CODE% = 256
PRNT_5      PRINT MESSAGE$
PRNT_10     PRINT'LETTER SIZE(0-15) ';:\
            GOSUB NUMBR
            S1% = A:\
            IF S1% > 15 OR S1% < 0 GOTO PRNT_10
            CODE% = CODE% + S1%:\
            P%(2) = 0:\
            P%(3) = CODE%:\
            CODE% = 0:\
            P%(0) = X_MAX% + 1:\
            PRINT
PRNT_20     PRINT CUR_MESGE$:\
            PRINT  'LOWER LEFT OF 1st CHARACTER ';:\
            GOSUB FINE_POS
            IF P%(0) = X% AND P%(1) = Y% GOTO SW_TEST
            P%(0) =  X%:\
            P%(1) =  Y%
            PRINT'LABEL(9 CHAR) ';:\
            LENGTH% = 9:\
            GOSUB NAMR
            B$ = A$:\
            A$ = 'P' + B$:\
            GOSUB PRNTER
            GOSUB UPDATE
            GOTO PRNT_20
CHANGE      PRINT CUR_MESGE$:\
            PRINT 'LOWER LEFT CORNER OF THE NEW VIEW';:\
            GOSUB POSITION
            X1%   = X%:\
            Y1%   = Y%:\
            PRINT  'UPPER RIGHT CORNER OF THE NEW VIEW';:\
            GOSUB POSITION
            X = ABS(X% - X1%):\
            Y = ABS(Y% - Y1%):\
            IF X <> 0 AND Y <> 0 GOTO CHA_10
                 PRINT 'MUST HAVE DISTINCT POINTS':\
```

```
            GOTO CHANGE
CHA_10      IF 224/X >= 160/Y THEN SCRATCH = X \
                              ELSE SCRATCH = Y*1.4
            SIZE = 0.5
SIZ_1       SIZE = SIZE*2 :\
            LOC_W% = SCRATCH/SIZE :\
            IF LOC_W% <= CRT_X%  GOTO SIZ_8
            IF SIZE = 32 GOTO SIZ_8
               GOTO SIZ_1
SIZ_8       SIZE% = SIZE:\
            DPOKE SIZ, SIZE%:\
            X_LOW% = X1%:\
            Y_LOW% = Y1%:\
            X_HIGH% = CRT_X%*SIZE% + X_LOW%:\
            IF  X_HIGH% <= X_MAX%  GOTO  CHANGE_3
                X_HIGH% = X_MAX%:\
                X_LOW% = X_HIGH% - CRT_X%*SIZE%:\
                IF X_LOW% < 0  THEN X_LOW% = 0
CHANGE_3    Y_HIGH% = CRT_Y%*SIZE% + Y_LOW%:\
            IF  Y_HIGH% <= Y_MAX%  GOTO  CHANGE_4
                Y_HIGH% = Y_MAX%:\
                Y_LOW% = Y_HIGH% - CRT_Y%*SIZE%:\
                IF Y_LOW% < 0  THEN Y_LOW% = 0
CHANGE_4    GOSUB CLEAN_UP
            GOTO INIT_100
WIDE_A        GOSUB CLEAN_UP
            GOTO INIT_0
POSITION    GOSUB RED_B
            PRINT:\
            X%=INT((PEEK(XCUR)*SIZE%+16+X_LOW%)/32)*32:\
            Y%=INT(((191-PEEK(YCUR))*SIZE%+16+Y_LOW%)/32)*32:\
            IF X% <= X_HIGH% AND X% >= X_LOW%\
                AND Y% <= Y_HIGH% AND Y% >= Y_LOW% THEN RETURN
            PRINT 'NOT A VALID COORDINATE, TRY AGAIN';:\
            GOTO POSITION
FINE_POS    GOSUB RED_B
            PRINT:\
            X% = PEEK(XCUR)*SIZE%+X_LOW%:\
            Y% = (191-PEEK(YCUR))*SIZE%+Y_LOW%:\
            IF X% <= X_HIGH% AND X% >= X_LOW%\
                AND Y% <= Y_HIGH% AND Y% >= Y_LOW% THEN RETURN
            PRINT 'NOT A VALID COORDINATE, TRY AGAIN':\
            GOTO FINE_POS
RED_B       POKE BUTTON, 0
RED_10      IF PEEK(BUTTON) = HEX("80") THEN RETURN
               GOTO RED_10
UPDATE      D_PTR% = D_PTR% + 1:\
            IF D_PTR% <= 13  GOTO UP_5
               PUT #1, RECORD N%:\
               N% = N% + 1:\
               D_PTR% = 0
```

```
UP_5        LSET ELEM$(D_PTR%) = A$:\
            FOR I% = 0 TO 3:\
                 LSET P$(I%,D_PTR%) = CVT%$(P%(I%)):\
            NEXT I%:\
            RETURN
QUIT        GOSUB CLEAN_UP
            GET #1, RECORD 1:\
            LSET P$(1,0) = CVT%$(D_PTR%):\
            LSET P$(0,0) = CVT%$(N%):\
            PUT #1, RECORD 1:\
            CLOSE 1:\
            CLOSE 2:\
            CLOSE 3:\
            EXEC,'TTYSET,PS=Y':\
            CHAIN "0.MADS-3"
CLEAN_UP    A$ = '0000000000':\
            IF D_PTR% = 13 GOTO CLN_5
            FOR J% = D_PTR%+1 TO 13:\
                 LSET ELEM$(J%) = A$:\
                 FOR I% = 0 TO 3:\
                      LSET P$(I%,J%) = CVT%$(0):\
                 NEXT I%:\
            NEXT J%
CLN_5       PUT #1, RECORD N%:\
            RETURN
INIT_0      SIZE% = 16*SCALE%:\
            SIZE = SIZE%:\
            DPOKE SIZ, SIZE%:\
            X_LOW% = 0:\
            Y_LOW% = 0:\
            X_HIGH% = CRT_X% * SIZE%:\
            Y_HIGH% = CRT_Y%*SIZE%
INIT_100    DPOKE POINTR, WIPE:\
            DUM% = USR (0):\
            DPOKE X_LO,X_LOW%:\
            DPOKE Y_LO,Y_LOW%:\
            DPOKE X_HI,X_HIGH%:\
            DPOKE Y_HI,Y_HIGH%:\
            P%(0) = X_LOW%:\
            P%(1) = Y_LOW%:\
            P%(2) = X_HIGH% - X_LOW%:\
            P%(3) = Y_HIGH% - Y_LOW%:\
            GOSUB REC_1
            DPOKE CRT_X, CRT_X%:\
            DPOKE CRT_Y, CRT_Y%:\
            DPOKE POINTR, PLOT:\
            FOR X1% = 0 TO X_HIGH% STEP 256:\
                 FOR Y1% = 0 TO Y_HIGH% STEP 256:\
                      IF X1% < X_LOW% OR Y1% < Y_LOW% \
                           GOTO INIT_105
                      DPOKE X_LOC,(X1% - X_LOW%)/SIZE%:\
```

```
                    DPOKE Y_LOC,(Y1% - Y_LOW%)/SIZE%:\
                    DUM% = USR(0)
INIT_105          NEXT Y1%:\
               NEXT X1%:\
               IF D_PTR% = 0  AND  N% = 1  GOTO DRAW_1
               MACRO_SW% = 0:\
               FOR J_REC% = 1 TO  N%:\
                    GET #1, RECORD J_REC%:\
                    IF J_REC% = 1  THEN  M% = 1\
                              ELSE M% = 0
                    FOR I_ELEM% = M% TO 13:\
                         A$ = ELEM$(I_ELEM%):\
                         IF A$ = '0000000000' OR \
                            A$ = 'ZZZZZZZZZZ'  GOTO INIT_300
                         FOR K% = 0 TO 3:\
                              P%(K%) = CVT$%(P$(K%,I_ELEM%)):\
                         NEXT K%:\
                         GOSUB FILE_OUT
                         IF DR_SW% <> -1  GOTO INIT_300
                         IF MACRO_SW% = 0 GOTO INIT_110
                              CTR% = CTR% + 1:\
                              IF CTR% < NO_ELEM% GOTO INIT_300
                              IF PEEK(WIN_SW) <> 0  GOTO INIT_115
                                   MACRO_SW% = 0:\
                                   GOTO INIT_300
INIT_110                 IF PEEK(WIN_SW) = 0  GOTO INIT_300
                         C$ = A$
INIT_115                 POKE WIN_SW, 0:\
                         PRINT:\
                         PRINT C$;'--ERASE[(Y)ES-(N)O-(Q)UIT] ';:\
                         B$ = INCH$(0):\
                         IF B$ = 'Y' OR B$ = 'y'  GOTO INIT_200
                            IF B$ = 'N' OR B$ = 'n'  GOTO INIT_150
                                 DR_SW% = 1
INIT_150                    MACRO_SW% = 0:\
                              GOTO INIT_300
INIT_200                 ERASE% = 255:\
                         IF MACRO_SW% = 0 THEN GOSUB FILE_OUT \
                                          ELSE GOSUB RERUN
                         ERASE% = 0:\
                         PRINT:\
                         PRINT 'VERIFY(Y-N) ';:\
                         B$ = INCH$(0):\
                         IF B$ = 'Y' OR B$ = 'y' GOTO INIT_250
                            IF MACRO_SW% = 0 THEN GOSUB FILE_OUT\
                                             ELSE GOSUB RERUN
                            MACRO_SW% = 0
                            GOTO INIT_300
INIT_250                 IF MACRO_SW% = 0 GOTO INIT_290
                              I1% = ELEM_PTR%:\
                              JJ% = REC_PTR%:\
```

```
                        IF II% <> 0 GOTO INIT_260
                             GET #1, RECORD JJ%-1:\
                             LSET ELEM$(13)='ZZZZZZZZZZ':\
                             PUT #1, RECORD JJ%-1:\
                             GET #1, RECORD JJ%:\
                             GOTO INIT_265
INIT_260                    GET #1, RECORD JJ%:\
                            LSET ELEM$(II%-1) = 'ZZZZZZZZZZ'
INIT_265                    FOR CTR% = 1 TO NO_ELEM%:\
                            LSET ELEM$(II%) = 'ZZZZZZZZZZ':\
                            II% = II% + 1:\
                            IF II% <= 13 GOTO INIT_270
                                 PUT #1, RECORD JJ%:\
                                 II% = 0:\
                                 JJ% = JJ% + 1:\
                                 GET #1, RECORD JJ%
INIT_270                    NEXT CTR%:\
                            PUT #1, RECORD JJ%:\
                            GET #1, RECORD J_REC%:\
                            MACRO_SW% = 0:\
                            GOTO INIT_300
INIT_290                LSET ELEM$(I_ELEM%) = 'ZZZZZZZZZZ':\
                        PUT #1, RECORD J_REC%
INIT_300            NEXT I_ELEM%:\
                NEXT J_REC%:\
                GOTO DRAW_1
RERUN           II% = ELEM_PTR%:\
                JJ% = REC_PTR%:\
                GET #1, RECORD JJ%:\
                FOR CTR% = 1 TO NO_ELEM%:\
                     A$ = ELEM$(II%):\
                     IF A$ = '0000000000' OR \
                        A$ = 'ZZZZZZZZZZ'  GOTO RER_15
                     FOR K% = 0 TO 3:\
                          P%(K%) = CVT$%(P$(K%,II%)):\
                     NEXT K%:\
                     GOSUB FILE_OUT
RER_15               II% = II% + 1:\
                     IF II% <= 13 GOTO RER_20
                          II% = 0:\
                          JJ% = JJ% + 1:\
                          GET #1, RECORD JJ%
RER_20          NEXT CTR%:\
                RETURN
FILE_OUT        POKE ERASE, ERASE%:\
                IF  A$ = 'LINE'  GOTO L_1
                IF  A$ = "RECT"  GOTO R_1
                IF  A$ = 'N6' THEN A$ = 'CIRCLE'
                IF  A$ = "CIRCLE" GOTO A_1
                IF  A$ = 'H_CIR'  GOTO A_1
                IF  A$ = "Q_CIR"  GOTO A_1
```

```
            IF A$ = 'DASH_ON' OR A$ = 'DASH_OFF' THEN RETURN
            IF LEFT$(A$,1) = 'P' GOTO P_1
            IF DR_SW% <> -1 THEN RETURN
            IF MACRO_SW% = 1 THEN RETURN
            MACRO_SW% = 1:\
            C$ = A$:\
            NO_ELEM% = P%(0):\
            CTR% = -1:\
            IF I_ELEM% = 13 GOTO FILE_10
                 ELEM_PTR% = I_ELEM% + 1:\
                 REC_PTR% = J_REC%:\
                 RETURN
FILE_10     ELEM_PTR% = 0:\
            REC_PTR% = J_REC% + 1:\
            RETURN
L_1         GOSUB  LIN_1
            RETURN
R_1         GOSUB  REC_1
            RETURN
A_1         GOSUB  AR_1
            RETURN
P_1         GOSUB  PRNTER
            RETURN
LIN_1       X1%=P%(0):\
            Y1%=P%(1):\
            X2%=P%(2):\
            Y2%=P%(3):\
            GOSUB WIN_CON
            RETURN
REC_1       IF P%(3) <= SIZE% THEN REC_5
                 IF P%(2) <= SIZE% THEN REC_2
                      BASE%= P%(2):\
                      HEIGHT%= P%(3):\
                      X1%=P%(0):\
                      Y1%=P%(1):\
                      X2%=X1% + BASE%:\
                      Y2%=Y1%
REC_1_4               GOSUB WIN_CON
                      X1%=X2%:\
                      Y2%=Y1% + HEIGHT%
REC_1_6               GOSUB WIN_CON
                      Y1%=Y2%:\
                      X2%=X1% - BASE%
REC_1_7               GOSUB WIN_CON
                      X1%=X2%:\
                      Y1%=Y2%:\
                      Y2%=Y1% - HEIGHT%
REC_1_8               GOSUB WIN_CON
                      RETURN
REC_2            IF P%(3) <= SIZE% THEN REC_4
                      X1%=P%(0):\
```

```
                    Y1%=P%(1):\
                    X2% = P%(0):\
                    Y2% = P%(1) + HEIGHT
REC_3               GOSUB WIN_CON
                    RETURN
REC_4           X1%=P%(0):\
                Y1%=P%(1):\
                X2%=P%(0):\
                Y2%=P%(1):\
                GOSUB WIN_CON
                RETURN
REC_5       IF P%(2) <= SIZE%  GOTO REC_4
            X1%=P%(0):\
            Y1%=P%(1):\
            X2%=P%(0) + BASE:\
            Y2%=P%(1):\
            GOSUB WIN_CON
            RETURN
WIN_CON     IF X1% > X_MAX%  OR  X1% < 0  GOTO BND_ERR
            IF X2% > X_MAX%  OR  X2% < 0  GOTO BND_ERR
            IF Y1% > Y_MAX%  OR  Y1% < 0  GOTO BND_ERR
            IF Y2% > Y_MAX%  OR  Y2% < 0  GOTO BND_ERR
            DPOKE G_X1, X1%:\
            DPOKE G_Y1, Y1%:\
            DPOKE G_X2, X2%:\
            DPOKE G_Y2, Y2%:\
            DPOKE POINTR,WINDO:\
            POKE ERASE, ERASE%:\
            DUM% = USR(0):\
            RETURN
BND_ERR     BND_SW% = 1:\
            RETURN
AR_1        X1% = P%(0):\
            Y1% = P%(1):\
            RAD% = P%(2):\
            CODE% = P%(3):\
            S1% = ABS(X_HIGH% - X1%):\
            S2% = ABS(X_LOW% - X1%)
            S3% = ABS(Y_HIGH% - Y1%):\
            S4% = ABS(Y_LOW% - Y1%):\
            IF X1% < X_LOW%  GOTO AR_16
                 IF X1% > X_HIGH%  GOTO AR_10
                      IF Y1% < Y_LOW%  GOTO AR_8
                           IF Y1% > Y_HIGH%  GOTO AR_6
AR_4                            IF   RAD% > SQR(S1%^2 + S3%^2)\
                                AND  RAD% > SQR(S1%^2 + S4%^2)\
                                AND  RAD% > SQR(S2%^2 + S3%^2)\
                                AND  RAD% > SQR(S2%^2 + S4%^2)\
                                        THEN AR_120\
                                        ELSE  AR_98
AR_6                        IF RAD% < S3% THEN AR_120
```

```
                        IF    RAD% > S3% + 230*SIZE%    \
                       OR CODE% = 0  THEN  RETURN
                            GOTO AR_98
AR_8                 IF  RAD% < S4%  THEN AR_120
                     IF  RAD% > S3% + 230*SIZE%  THEN AR_120
                     IF CODE% = 2 GOTO AR_120
                          GOTO AR_98
AR_10          IF Y1% > Y_HIGH%  GOTO AR_14
                     IF Y1% < Y_LOW%  GOTO AR_12
                          IF RAD% <S1% THEN AR_120
                          IF  RAD% >  S1% + 273*SIZE% \
                                  THEN AR_120
                          IF CODE% = 3 GOTO AR_120
                            GOTO AR_98
AR_12                SCRATCH%= SQR(S1%^2 + S4%^2):\
                     IF RAD% < SCRATCH%   THEN AR_120
                     IF RAD% > SCRATCH%+320*SIZE% THEN AR_120
                      IF CODE% >= 2  GOTO AR_120
                          GOTO AR_98
AR_14                 SCRATCH%= SQR(S1%^2 + S3%^2):\
                      IF RAD% < SCRATCH%              THEN  AR_120
                      IF RAD% > SCRATCH% + SIZE%*320 THEN AR_120
                      IF CODE% = 3  GOTO AR_120
                         GOTO AR_98
AR_16          IF Y1% >Y_HIGH%  GOTO AR_20
                     IF Y1% < Y_LOW%  GOTO AR_18
                          IF RAD% < S2% THEN AR_120
                          IF  RAD% > S2% + 273*SIZE%    THEN AR_120
                           IF  CODE% = 1  GOTO AR_120
                          GOTO AR_98
AR_18                SCRATCH%= SQR(S2%^2 + S4%^2):\
                     IF RAD% < SCRATCH%            THEN AR_120
                     IF RAD% > SCRATCH% + 320*SIZE% THEN AR_120
                      IF CODE% = 1 OR CODE% =2  GOTO AR_120
                         GOTO AR_98
AR_20           SCRATCH%= SQR(S2%^2 + S3%^2):\
                IF RAD% < SCRATCH%             THEN AR_120
                IF RAD% > SCRATCH% + 320*SIZE% THEN AR_120
                 IF  CODE% =1  GOTO AR_120
AR_98           DPOKE RADIUS, RAD%:\
                DPOKE X1, X1%:\
                DPOKE Y1, Y1%:\
                POKE WIN_SW, 1:\
                IF A$ <> 'CIRCLE'  AND A$ <> 'N6' GOTO AR_110
                     DPOKE POINTR, CIR:\
                     DUM% = USR(0):\
                     RETURN
AR_110          IF A$ = 'H_CIR'  GOTO AR_115
                     DPOKE POINTR, Q_CIR:\
                     DPOKE CODE, P%(3):\
                     DUM% = USR(0):\
```

```
            RETURN
AR_115      DPOKE POINTR, H_CIR:\
            DPOKE CODE, P%(3):\
            DUM% = USR(0):\
            RETURN
AR_120      POKE WIN_SW, 0:\
            RETURN
PRNTER      CODE% = P%(3):\
            FOR I% = 1 TO 9:\
                 IF RIGHT$(A$,1) <> ' ' GOTO PRN_5
                      A$ = LEFT$(A$,LEN(A$)-1):\
            NEXT I%
PRN_5       IF CODE% > 16      GOTO PRN_10
                 P%(2) = (LEN(A$)-1)*8*CODE%:\
                 P%(3) = 8*CODE%:\
                 GOSUB REC_1
                 P%(2) = 0:\
                 P%(3) = CODE%:\
                 IF 8*CODE%/SIZE% < 9 THEN RETURN
                 X1% = (P%(0)-X_LOW%)/SIZE% + 1:\
                 GOTO PRN_20
PRN_10      CODE% = CODE% - 256:\
            P%(2) = 8*CODE%:\
            P%(0) = P%(0) - P%(2):\
            P%(3) = 8*(LEN(A$)-1)*CODE%:\
            GOSUB REC_1
            P%(0) = P%(0) + P%(2):\
            P%(2) = 0:\
            P%(3) = CODE% + 256:\
            IF 8*CODE%/SIZE% < 9 THEN RETURN
            X1% = (P%(0)-X_LOW%)/SIZE% - 1
PRN_20      DPOKE POINTR, PRNTR:\
            Y1% = (P%(1)-Y_LOW%)/SIZE% + 1:\
            IF X1%  <  0 \
               OR Y1%  <  0 \
               OR X1%  > 218 \
               OR Y1%  > 153  THEN  RETURN
                DPOKE X0, X1%:\
                DPOKE Y0, Y1%
PRN_30      DPOKE CODE, P%(3):\
            DPOKE CHPTR, PTR(A$):\
            DUM% = USR(0):\
            RETURN
NAMR        A$ = '':\
            CR_SW% = 0:\
            I% = 1
NAMR_12     B$ = INCH$(0):\
            IF B$ = CHR$(13) GOTO NAMR_40
            IF B$ <> CHR$(8)  GOTO NAMR_15
                 IF LEN(A$) <= 1  GOTO NAMR
                 A$ = LEFT$(A$,LEN(A$)-1):\
```

```
                PRINT  ' ';CHR$(8);:\
                I% = I% - 1:\
                GOTO NAMR_12
NAMR_15     IF B$ > CHR$(31) AND B$ < CHR$(127) GOTO NAMR_20
                IF B$ = CHR$(10) THEN PRINT CHR$(11);
                IF B$ = CHR$(11) THEN PRINT CHR$(10);
                IF B$ = CHR$(12) THEN PRINT CHR$( 8);
                GOTO NAMR_12
NAMR_20      A$ = A$ + B$:\
            I% = I% + 1:\
            IF I% <= LENGTH% GOTO NAMR_12
            B$ = INCH$(0):\
            IF B$ = CHR$(13) GOTO NAMR_40
            IF B$ <> CHR$(8) GOTO NAMR_30
                PRINT ' '; CHR$(8);:\
                I% = I% - 1:\
                GOTO NAMR_12
NAMR_30     PRINT CHR$(10);CHR$(7);:\
            PRINT 'ONLY'; LENGTH%; ' CHARACTERS ARE ALLOWED ':\
            A$ = LEFT$(A$,LENGTH%):\
            PRINT 'THE TEXT IS '; A$:\
            PRINT 'DO YOU WANT TO START OVER(Y-N)?';:\
            B$ = INCH$(0):\
            PRINT:\
            IF B$ <> 'Y' AND B$ <> 'y' THEN RETURN
                GOTO NAMR
NAMR_40     PRINT:\
            IF LEN(A$) = 0 THEN CR_SW% = 1
            RETURN
NUMBR       C$ = '':\
            CR_SW% = 0
NUMB_90     B$ = INCH$(0):\
            IF B$ = '+' OR B$ ='-' OR B$ = '.'   \
              OR B$ > CHR$(47) AND B$ < CHR$(58) GOTO NUMB_100
                IF B$ = CHR$(13) GOTO NUMB_200
                IF B$ = CHR$(08) THEN PRINT CHR$(12);CHR$(12);
                IF B$ = CHR$(11) THEN PRINT CHR$(10);
                IF B$ = CHR$(10) THEN PRINT CHR$(11);
                     PRINT CHR$(8); ' '; CHR$(8);:\
                     GOTO NUMBR
NUMB_100    C$ = C$ + B$
NUMB_110     B$ = INCH$(0):\
            IF B$ = CHR$(13) GOTO NUMB_150
            IF B$ > CHR$(47) AND B$ < CHR$(58) GOTO NUMB_100
            IF B$ = "."   GOTO NUMB_100
                IF B$ = CHR$(8) GOTO NUMB_115
                     PRINT CHR$(8); ' '; CHR$(8);:\
                     IF LEN(C$) = 2 GOTO NUMBR
                     C$ = LEFT$(C$,LEN(C$)-2):\
                     GOTO NUMB_110
NUMB_115            PRINT ' ';CHR$(8);:\
```

```
                IF LEN(C$) = 1 GOTO NUMBR
                C$ = LEFT$(C$,LEN(C$)-1):\
                GOTO NUMB_100
NUMB_150        A      = VAL(C$):\
                RETURN
NUMB_200        CR_SW% = 1:\
                RETURN
ENVIRN          SIZ = HEX("9750"):\
                POINTR = DPEEK(52267) - 2:\
                PLOT = HEX("9C60"):\
                ERASE = HEX("9758"):\
                X_LOC = HEX("9700"):\
                Y_LOC = HEX("9702"):\
                G_X1 = HEX("970C")
                G_X2 = HEX("9710"):\
                G_Y1 = HEX("970E"):\
                G_Y2 = HEX("9712"):\
                X1 = HEX("9704"):\
                Y1 = HEX("9706"):\
                X2 = HEX("9708"):\
                Y2 = HEX("970A")
                X_LO = HEX("9746"):\
                Y_LO = HEX("9748"):\
                X_HI = HEX("974A"):\
                Y_HI = HEX("974C"):\
                XCUR = HEX("E008"):\
                YCUR = HEX("E00A"):\
                BUTTON = HEX("E009")
                WINDO = HEX("9E0D"):\
                CODE = HEX("9773"):\
                CIR    = HEX("A210"):\
                Q_CIR = HEX("A37B")
                H_CIR = HEX("A3A8"):\
                RADIUS = HEX("976D"):\
                PRNTR = HEX("A3EB"):\
                CHPTR = HEX("9781"):\
                X0 = HEX("9784"):\
                Y0 = HEX("9786"):\
                CRT_X = HEX("976F"):\
                CRT_Y = HEX("9771")
                WIPE = HEX("A72C"):\
                CRT_X% = 224:\
                CRT_Y% = 160:\
                WIN_SW = HEX("9753")
                OPEN OLD "PARA" AS 1:\
                INPUT #1, TITL$:\
                CLOSE 1
                OPEN TITL$ AS 1:\
                FOR I% = 0 TO 13:\
                     FIELD #1, 18*I% AS Z$,\
                         10 AS ELEM$(I%),\
```

```
                    2 AS P$(0,I%),\
                    2 AS P$(1,I%),\
                    2 AS P$(2,I\68),\
                    2 AS P$(3,I%):\
          NEXT I%
          OPEN 'MAC_ELEM' AS 2:\
          FOR I% = 0 TO 13:\
               FIELD #2, 18*I% AS Z$,\
                   10 AS M_ELEM$(I%),\
                    2 AS MP$(0,I%),\
                    2 AS MP$(1,I%),\
                    2 AS MP$(2,I%),\
                    2 AS MP$(3,I%):\
          NEXT I%
          OPEN 'MAC_DIR' AS 3:\
          FOR I% = 0 TO 13:\
               FIELD #3, 18*I% AS Z$,\
                    10 AS  NAM$(I%),\
                   2 AS    REC$(I%),\
                   2 AS EL_PTR$(I%),\
                   2 AS  NO_EL$(I%),\
                   2 AS  ROT_CN$(I%):\
        NEXT I%
        GET #3, RECORD 1:\
        LAST_M_DIR_REC% = CVT$%(REC$(0)):\
         NO_MAC% = CVT$%(NO_EL$(0))
         GET #1, RECORD 1:\
         SCALE% = CVT$%(P$(2,0)):\
         D_PTR% = CVT$%(P$(1,0)):\
         N% = CVT$%(P$(0,0))
         X_MAX% = 3584*SCALE%:\
         Y_MAX% = 2560*SCALE%:\
         GOTO INIT_0
```

Preplot

```
         DIM P%(3), ELEM$(13), P$(3,13), MP$(3,13)
         DIM NAM$(13), REC$(13), EL_PTR$(13), NO_EL$(13)
         DIM M_ELEM$(13), ROT_CN$(13)
         ON ERROR GOTO END_OF_FILE
         GOTO ENVIRN
END_OF_FILE    IF ERR <> 7 THEN ON ERROR GOTO
               PRINT 'THIS DISKETTE IS FULL'
               RESUME QUIT_1
PL_10    DN% = 0
         PRINT CHR$(26);
         FOR I% = 1 TO 10: PRINT: NEXT I%
         PRINT TAB(10); 'WHAT DO YOU WANT TO DO?'
         PRINT TAB(10); 'POSITION THE CURSOR AND PRESS ENTER'
```

```
          PRINT
          PRINT TAB(10); '( ) QUIT'
          PRINT TAB(10); '( ) PLOT THE ENTIRE DRAWING'
          PRINT TAB(10);\
     '( ) PLOT THE DRAWING IN THE LOWER LEFT 1/4 OF THE PLOT'
          PRINT TAB(10);\
     '( ) PLOT THE DRAWING IN THE LOWER RIGHT 1/4 OF THE PLOT'
          PRINT TAB(10);\
     '( ) PLOT THE DRAWING IN THE UPPER LEFT 1/4 OF THE PLOT'
          PRINT TAB(10);\
     '( ) PLOT THE DRAWING IN THE LOWER RIGHT 1/4 OF THE PLOT'
          PRINT TAB(10); '( ) SPECIAL HANDLING';
          FOR I% = 1 TO 6: PRINT CHR$(11); CHR$(8);: NEXT I%
          FOR I% = 1 TO 13: PRINT CHR$(8);: NEXT I%
PL_20     B$ = INCH$(0)
          IF B$ < CHR$(20) GOTO PL_30
               PRINT CHR$(8);
               GOTO PL_20
PL_30     IF B$ = CHR$(13) GOTO PL_40
          IF B$ = CHR$(8) THEN PRINT CHR$(12);
          IF B$ = CHR$(9) GOTO PL_10
          IF B$ = CHR$(10) THEN DN% = DN% + 1
          IF B$ = CHR$(11) THEN DN% = DN% - 1
          IF B$ = CHR$(12) THEN PRINT CHR$(8);
          GOTO PL_20
PL_40     IF DN% < 0 OR DN% > 6 GOTO PL_10
          IF DN% = 0 GOTO QUIT_1
          IF DN% <> 1 GOTO PL_50
               XP_LOW = 0
               YP_LOW = 0
               XP_HIGH = 3556
               YP_HIGH = 2540
               GOTO PL_190
PL_50     IF DN% <> 2 GOTO PL_60
               XP_LOW = 0
               YP_LOW = 0
               XP_HIGH = 1778
               YP_HIGH = 1270
               GOTO PL_190
PL_60     IF DN% <> 3 GOTO PL_70
               XP_LOW = 1778
               YP_LOW = 0
               XP_HIGH = 3556
               YP_HIGH = 1270
               GOTO PL_190
PL_70     IF DN% <> 4 GOTO PL_80
               XP_LOW = 0
               YP_LOW = 1270
               XP_HIGH = 1778
               YP_HIGH = 2540
               GOTO PL_190
```

```
PL_80       IF DN% <> 5 GOTO PL_90
                 XP_LOW = 1778
                 YP_LOW = 1270
                 XP_HIGH = 3556
                 YP_HIGH = 2540
                 GOTO PL_190
PL_90       PRINT CHR$(26)
            PRINT\
             'POSITION THE GRAPHICS CURSOR ON THE LOWER LEFT'
            PRINT\
             'CORNER OF THE AREA TO BE PLOTTED.(RED BUTTON)';
            GOSUB POSITION
            X1% = X%: Y1% = Y%
            X_LOW = X%: Y_LOW = Y%
            FOR I% = 0 TO 100: S% = S% + 1: NEXT I%
            PRINT
            PRINT 'CURSOR ON UPPER RIGHT CORNER.(RED BUTTON)';
            GOSUB POSITION
            PRINT
            X2% = X%: Y2% = Y%
            X_HIGH = X%: Y_HIGH = Y%
            P%(0) = X1%
            P%(1) = Y1%
            P%(2) = ABS(X2% - X1%)
            P%(3) = ABS(Y2% - Y1%)
            ERASE% = 0
            GOSUB REC_1
            PRINT 'CORRECT (Y-N)?';
            B$ = INCH$(0)
            PRINT
            IF B$ = 'Y' OR B$ = 'y' GOTO PL_100
                 ERASE% = 255
                 GOSUB REC_1
                 GOTO PL_90
PL_100      DPOKE POINTR, WIPE
            DUM% = USR (0) :\
            DPOKE X_LO,X_LOW% :\
            DPOKE Y_LO,Y_LOW% :\
            DPOKE X_HI,X_HIGH% :\
            DPOKE Y_HI,Y_HIGH%
            P%(0) = X_LOW% :\
            P%(1) = Y_LOW% :\
            P%(2) = X_HIGH% - X_LOW% :\
            P%(3) = Y_HIGH% - Y_LOW% :\
            GOSUB REC_1
            DPOKE CRT_X, CRT_X% :\
            DPOKE CRT_Y, CRT_Y% :\
            DPOKE POINTR, PLOT :\
            FOR X1% = 0 TO X_HIGH% STEP 256 :\
                 FOR Y1% = 0 TO Y_HIGH% STEP 256 :\
                      IF X1% < X_LOW% OR Y1% < Y_LOW% \
```

```
                    GOTO PL_105
                    DPOKE X_LOC,(X1% - X_LOW%)/SIZE%:\
                    DPOKE Y_LOC,(Y1% - Y_LOW%)/SIZE%:\
                    DUM% = USR(0)
PL_105          NEXT Y1% :\
              NEXT X1%
PL_110        PRINT CHR$(26);\
               'POSITION THE CURSOR ON THE LOWER LEFT CORNER'
              PRINT 'OF THE AREA OF THE PLOT BED WHERE YOU'
              PRINT 'WANT THE PLOT TO APPEAR.(RED BUTTON)';
              GOSUB POSITION
              PRINT
              X1% = X%
              Y1% = Y%
              X1 = X%/SIZE%
              Y1 = Y%/SIZE%
              FOR I% = 0 TO 100: S% = S% + 1: NEXT I%
              PRINT 'POSITION THE CURSOR ON THE RIGHT SIDE'
              PRINT 'OF THE AREA OF THE PLOT BED WHERE YOU'
              PRINT 'WANT THE PLOT TO APPEAR.(RED BUTTON)';
              GOSUB POSITION
              PRINT
              X2% = X%
              Y2% = Y%
              X2 = X%/SIZE%
              Y2 = Y%/SIZE%
              SIZE = 15.875
              XP_LOW=X1*SIZE: YP_LOW=Y1*SIZE: XP_HIGH=X2*SIZE
              YP_HIGH=ABS(XP_HIGH - XP_LOW)*ABS(Y_HIGH - Y_LOW)\
                        /ABS(X_HIGH - X_LOW) + YP_LOW
              IF YP_HIGH > 2540 THEN YP_HIGH = 2540
              Y2% = YP_HIGH/SIZE
              Y2% = Y2%*SIZE%
              P%(0) = X1%
              P%(1) = Y1%
              P%(2) = ABS(X2% - X1%)
              P%(3) = ABS(Y2% - Y1%)
              ERASE% = 0
              GOSUB REC_1
              PRINT 'CORRECT (Y-N)?';
              B$ = INCH$(0)
              PRINT
              IF B$ = 'Y' OR B$ = 'y' GOTO PL_200
                   ERASE% = 255
                   GOSUB REC_1
                   GOTO PL_110
PL_190        X_HIGH = 3584*SCALE
              Y_HIGH = 2560*SCALE
              X_LOW = 0
              Y_LOW = 0
PL_200        OPEN NEW 'PARA' AS 2
```

```
          PRINT #2, TITL$;',';X_LOW;',';Y_LOW;',';\
                    X_HIGH;',';Y_HIGH;',';XP_LOW;',';\
                    YP_LOW;',';XP_HIGH;',';YP_HIGH
          CLOSE 1
          CHAIN '0.PLOTTER'
*
POSITION  GOSUB RED_B
          PRINT
          X% = PEEK(XCUR)*SIZE%
          Y% = (191 - PEEK(YCUR))*SIZE%
          IF X% <= X_HIGH% AND X% >= X_LOW%\
               AND Y% <= Y_HIGH% AND Y% >= Y_LOW% THEN RETURN
          PRINT 'NOT A VALID COORDINATE, TRY AGAIN'
          GOTO POSITION
*
RED_B     POKE BUTTON, 0
RED_10    IF PEEK(BUTTON) = HEX("80") THEN RETURN
          GOTO RED_10
*
QUIT_1    CLOSE 1 :\
          EXEC,'TTYSET,PS=Y'
          CHAIN "0.MADS-3"
*
INIT_100  DPOKE POINTR, CLEAR :\
          DUM% = USR (0) :\
          DPOKE X_LO,X_LOW% :\
          DPOKE Y_LO,Y_LOW% :\
          DPOKE X_HI,X_HIGH% :\
          DPOKE Y_HI,Y_HIGH%
          P%(0) = X_LOW% :\
          P%(1) = Y_LOW% :\
          P%(2) = X_HIGH% - X_LOW% :\
          P%(3) = Y_HIGH% - Y_LOW% :\
          GOSUB REC_1
          DPOKE CRT_X, CRT_X% :\
          DPOKE CRT_Y, CRT_Y% :\
          DPOKE POINTR, PLOT :\
          FOR X1% = 0 TO X_HIGH% STEP 256*SCALE% :\
               FOR Y1% = 0 TO Y_HIGH% STEP 256*SCALE% :\
                    IF X1% < X_LOW% OR Y1% < Y_LOW% \
                                   GOTO INIT_105
                    DPOKE X_LOC,(X1% - X_LOW%)/SIZE%:\
                    DPOKE Y_LOC,(Y1% - Y_LOW%)/SIZE%:\
                    DUM% = USR(0)
INIT_105            NEXT Y1% :\
          NEXT X1% :\
          IF D_PTR% = 0  AND  N% = 1  GOTO QUIT_1
          FOR J_REC% = 1 TO  N% :\
               GET #1, RECORD J_REC% :\
               IF J_REC% = 1  THEN  M% = 1\
                           ELSE M% = 0
```

```
                    FOR I_ELEM% = M% TO 13 :\
                         A$ = ELEM$(I_ELEM%) :\
                         IF A$ = '0000000000' OR \
                            A$ = 'ZZZZZZZZZZ'  GOTO INIT_300
                         FOR K% = 0 TO 3 :\
                              P%(K%) = CVT$%(P$(K%,I_ELEM%)) :\
                         NEXT K% :\
                         IF  A$ = 'LINE'  GOTO L_1
                         IF  A$ = "RECT"  GOTO R_1
                         IF  A$ = 'N6' THEN A$ = 'CIRCLE'
                         IF  A$ = "CIRCLE" GOTO A_1
                         IF  A$ = 'H_CIR'  GOTO A_1
                         IF  A$ = "Q_CIR"  GOTO A_1
                         IF LEFT$(A$,1) = 'P'       GOTO P_1
                              GOTO INIT_300
L_1                      GOSUB  LIN_1
                              GOTO INIT_300
R_1                      GOSUB  REC_1
                              GOTO INIT_300
A_1                      GOSUB  AR_1
                              GOTO INIT_300
P_1                      GOSUB  PRNTER
INIT_300            NEXT I_ELEM% :\
               NEXT J_REC% :\
               GOTO PL_10
*
*
*
LIN_1          X1%=P%(0) :\
               Y1%=P%(1) :\
               X2%=P%(2) :\
               Y2%=P%(3) :\
               GOSUB WIN_CON
               RETURN
*
REC_1          IF P%(3) <= SIZE% THEN REC_5
                    IF P%(2) <= SIZE% THEN REC_2
                         BASE%= P%(2) :\
                         HEIGHT%= P%(3) :\
                         X1%=P%(0) :\
                         Y1%=P%(1) :\
                         X2%=X1% + BASE% :\
                         Y2%=Y1%
REC_1_4                  GOSUB WIN_CON
                         X1%=X2% :\
                         Y2%=Y1% + HEIGHT%
REC_1_6                  GOSUB WIN_CON
                         Y1%=Y2% :\
                         X2%=X1% - BASE%
REC_1_7                  GOSUB WIN_CON
                         X1%=X2% :\
```

```
                    Y1%=Y2% :\
                    Y2%=Y1% - HEIGHT%
REC_1_8             GOSUB WIN_CON
                    RETURN
REC_2          IF P%(3) <= SIZE% THEN REC_4
                    X1%=P%(0) :\
                    Y1%=P%(1) :\
                    X2% = P%(0) :\
                    Y2% = P%(1) + HEIGHT
REC_3               GOSUB WIN_CON
                    RETURN
REC_4          X1%=P%(0) :\
               Y1%=P%(1) :\
               X2%=P%(0) :\
               Y2%=P%(1) :\
               GOSUB WIN_CON
               RETURN
REC_5       IF P%(2) <= SIZE%  GOTO REC_4
            X1%=P%(0) :\
            Y1%=P%(1) :\
            X2%=P%(0) + BASE :\
            Y2%=P%(1) :\
            GOSUB WIN_CON
            RETURN
*
WIN_CON     IF X1% > X_MAX%  OR  X1% < 0  GOTO BND_ERR
            IF X2% > X_MAX%  OR  X2% < 0  GOTO BND_ERR
            IF Y1% > Y_MAX%  OR  Y1% < 0  GOTO BND_ERR
            IF Y2% > Y_MAX%  OR  Y2% < 0  GOTO BND_ERR
            DPOKE G_X1, X1% :\
            DPOKE G_Y1, Y1% :\
            DPOKE G_X2, X2% :\
            DPOKE G_Y2, Y2% :\
            POKE ERASE, ERASE%
            DPOKE POINTR,WINDO :\
            DUM% = USR(0) :\
            RETURN
BND_ERR     PRINT 'Cannot draw this line--out of bounds.' :\
            PRINT'X1=';X1%;'Y1=';Y1%;'X2=';X2%;'Y2=';Y2%:\
            BND_SW% = 1 :\
            RETURN
*
AR_1        X1% = P%(0) :\
            Y1% = P%(1) :\
            RAD% = P%(2) :\
            CODE% = P%(3) :\
            S1% = ABS(X_HIGH% - X1%) :\
            S2% = ABS(X_LOW% - X1%)
            S3% = ABS(Y_HIGH% - Y1%) :\
            S4% = ABS(Y_LOW% - Y1%) :\
            IF X1% < X_LOW%  GOTO AR_16
```

```
                IF X1% > X_HIGH%  GOTO AR_10
                     IF Y1% < Y_LOW%  GOTO AR_8
                          IF Y1% > Y_HIGH%  GOTO AR_6
AR_4                           IF   RAD% > SQR(S1%^2 + S3%^2)\
                               AND  RAD% > SQR(S1%^2 + S4%^2)\
                               AND  RAD% > SQR(S2%^2 + S3%^2)\
                               AND  RAD% > SQR(S2%^2 + S4%^2)\
                                         THEN AR_120 \
                                         ELSE  AR_98
AR_6                           IF RAD% < S3% THEN AR_120
                               IF   RAD% > S3% + 230*SIZE%   \
                               OR CODE% = 0  THEN  RETURN
                                 GOTO AR_98
AR_8                     IF  RAD% < S4%  THEN AR_120
                         IF  RAD% > S3% + 230*SIZE%  THEN AR_120
                         IF CODE% = 2 GOTO AR_120
                              GOTO AR_98
AR_10               IF Y1% > Y_HIGH%  GOTO AR_14
                         IF Y1% < Y_LOW%  GOTO AR_12
                            IF RAD% <S1% THEN AR_120
                            IF RAD% > S1% + 273*SIZE% THEN AR_120
                            IF CODE% = 3 GOTO AR_120
                               GOTO AR_98
AR_12                    SCRATCH%= SQR(S1%^2 + S4%^2):\
                         IF RAD% < SCRATCH%            THEN AR_120
                         IF RAD% > SCRATCH%+320*SIZE% THEN AR_120
                          IF CODE% >= 2  GOTO AR_120
                              GOTO AR_98
AR_14                     SCRATCH%= SQR(S1%^2 + S3%^2):\
                          IF RAD% < SCRATCH%              THEN  AR_120
                          IF RAD% > SCRATCH% + SIZE%*320 THEN AR_120
                          IF CODE% = 3  GOTO AR_120
                              GOTO AR_98
AR_16               IF Y1% >Y_HIGH%  GOTO AR_20
                         IF Y1% < Y_LOW%  GOTO AR_18
                              IF RAD% < S2% THEN AR_120
                              IF  RAD% > S2%+273*SIZE%     THEN AR_120
                              IF  CODE% = 1  GOTO AR_120
                                 GOTO AR_98
AR_18                    SCRATCH%= SQR(S2%^2 + S4%^2):\
                         IF RAD% < SCRATCH%            THEN AR_120
                         IF RAD% > SCRATCH% + 320*SIZE% THEN AR_120
                          IF CODE% = 1 OR CODE% =2  GOTO AR_120
                              GOTO AR_98
AR_20               SCRATCH%= SQR(S2%^2 + S3%^2):\
                    IF RAD% < SCRATCH%            THEN AR_120
                    IF RAD% > SCRATCH% + 320*SIZE% THEN AR_120
                     IF  CODE% =1  GOTO AR_120
AR_98               DPOKE RADIUS, RAD% :\
                    DPOKE X1, X1% :\
                    DPOKE Y1, Y1% :\
```

```
            POKE WIN_SW, 1 :\
            IF A$ <> 'CIRCLE'  AND A$ <> 'N6' GOTO AR_110
                 DPOKE POINTR, CIR :\
                 DUM% = USR(0):\
                 RETURN
AR_110      IF A$ = 'H_CIR'  GOTO AR_115
                 DPOKE POINTR, Q_CIR :\
                 DPOKE CODE, P%(3) :\
                 DUM% = USR(0):\
                 RETURN
AR_115      DPOKE POINTR, H_CIR:\
            DPOKE CODE, P%(3):\
            DUM% = USR(0):\
            RETURN
AR_120      POKE WIN_SW, 0:\
            RETURN
*
PRNTER      CODE% = P%(3) :\
            FOR I% = 1 TO 9 :\
                 IF RIGHT$(A$,1) <> ' ' GOTO PRN_5
                      A$ = LEFT$(A$,LEN(A$)-1) :\
            NEXT I%
PRN_5       IF CODE% > 16      GOTO PRN_10
                 P%(2) = (LEN(A$)-1)*8*CODE% :\
                 P%(3) = 8*CODE% :\
                 GOSUB REC_1
                 P%(2) = 0 :\
                 P%(3) = CODE% :\
                 IF 8*CODE%/SIZE% < 9 THEN RETURN
                 GOTO PRN_20
PRN_10      CODE% = CODE% - 256 :\
            P%(2) = 8*CODE% :\
            P%(0) = P%(0) - P%(2) :\
            P%(3) = 8*(LEN(A$)-1)*CODE% :\
            GOSUB REC_1
            P%(0) = P%(0) + P%(2) :\
            P%(2) = 0 :\
            P%(3) = CODE% + 256 :\
            IF 8*CODE%/SIZE% < 9 THEN RETURN
PRN_20      DPOKE POINTR, PRNTR :\
            X1% = (P%(0)-X_LOW%)/SIZE% + 1 :\
            Y1% = (P%(1)-Y_LOW%)/SIZE% + 1 :\
            IF X1%  <  0 \
               OR Y1%  <  0 \
               OR X1%  > 218 \
               OR Y1%  > 153  THEN  RETURN
                DPOKE X0, X1% :\
                DPOKE Y0, Y1%
PRN_30      DPOKE CODE, P%(3) :\
            DPOKE CHPTR, PTR(A$) :\
            DUM% = USR(0) :\
```

```
            PRT_SW% = 1 :\
            RETURN
*
*
ENVIRN      OPEN "0.PRINT" AS 0 :\
            EXEC,'TTYSET,PS=N'
            SIZ = HEX("9750") :\
            POINTR = DPEEK(52267) - 2 :\
            CLEAR = HEX("9C7D") :\
            GRAPH = HEX("9AE5")
            PLOT = HEX("9C60") :\
            ERASE = HEX("9758") :\
            X_LOC = HEX("9700") :\
            Y_LOC = HEX("9702") :\
            G_X1 = HEX("970C")
            G_X2 = HEX("9710") :\
            G_Y1 = HEX("970E") :\
            G_Y2 = HEX("9712") :\
            X1 = HEX("9704") :\
            Y1 = HEX("9706") :\
            X2 = HEX("9708") :\
            Y2 = HEX("970A")
            X_LO = HEX("9746") :\
            Y_LO = HEX("9748") :\
            X_HI = HEX("974A") :\
            Y_HI = HEX("974C") :\
            XCUR = HEX("E008") :\
            YCUR = HEX("E00A") :\
            BUTTON = HEX("E009")
            WINDO = HEX("9E0D") :\
            CODE = HEX("9773") :\
            CIR   = HEX("A210") :\
            Q_CIR = HEX("A37B")
            H_CIR = HEX("A3A8") :\
            RADIUS = HEX("976D") :\
            PRNTR = HEX("A3EB") :\
            LI    = HEX("977D")
            LII   = HEX("977F") :\
            CHPTR = HEX("9781") :\
            X0 = HEX("9784") :\
            Y0 = HEX("9786") :\
            CRT_X = HEX("976F") :\
            CRT_Y = HEX("9771")
            WIPE = HEX("A72C") :\
            CRT_X% = 224 :\
            CRT_Y% = 160 :\
            WIN_SW = HEX("9753")
*
            OPEN OLD "PARA" AS 1 :\
            INPUT #1, TITL$ :\
            CLOSE 1
```

```
*
          OPEN TITL$ AS 1 :\
          FOR I% = 0 TO 13 :\
               FIELD #1, 18*I% AS Z$,\
                    10 AS ELEM$(I%),\
                     2 AS P$(0,I%),\
                     2 AS P$(1,I%),\
                     2 AS P$(2,I%),\
                     2 AS P$(3,I%) :\
          NEXT I%
*
          GET #1, RECORD 1 :\
          N% = CVT$%(P$(0,0))
          D_PTR% = CVT$%(P$(1,0)) :\
          SCALE% = CVT$%(P$(2,0)) :\
          SCALE = SCALE%
          SIZE% = 16*SCALE% :\
          SIZE = SIZE% :\
          SIZE = SIZE*3584/3556
          DPOKE SIZ, SIZE% :\
          X_LOW% = 0 :\
          Y_LOW% = 0 :\
          X_HIGH% = CRT_X% * SIZE% :\
          Y_HIGH% = CRT_Y%*SIZE%
          ERASE% = 0
          CONV = 256
          X_MAX% = 3584*SCALE% :\
          Y_MAX% = 2560*SCALE% :\
          PRINT CHR$(26);\
             'SHALL WE DO THE DRAWING ON THE CRT(Y-N)? ';
          B$ = INCH$(0)
          PRINT
          IF B$ = 'Y' OR B$ = 'y' THEN INIT_100 ELSE PL_10
```

Plotter

```
          DIM P%(3), ELEM$(13), P$(3,13), P(3)
          OPEN "0.PRINT" AS 0
          EXEC,'TTYSET,PS=N'
*
*
*
          OPEN OLD "PARA" AS 1
          INPUT #1, TITL$,X_LOW,Y_LOW,X_HIGH,Y_HIGH,XP_LOW,\
             YP_LOW,XP_HIGH,YP_HIGH
          CLOSE 1
*
          OPEN TITL$ AS 1
          FOR I% = 0 TO 13
```

```
            FIELD #1, 18*I% AS Z$,\
                10 AS ELEM$(I%),\
                 2 AS P$(0,I%),\
                 2 AS P$(1,I%),\
                 2 AS P$(2,I%),\
                 2 AS P$(3,I%)
        NEXT I%
*
        GET #1, RECORD 1
        D_PTR% = CVT$%(P$(1,0))
        N% = CVT$%(P$(0,0))
        SCALE% = CVT$%(P$(2,0))
        SIZE = (X_HIGH - X_LOW)/(XP_HIGH - XP_LOW)
        PRINT CHR$(26);\
              'PREPARE THE PLOTTER THEN PRESS ANY KEY ';
        B$ = INCH$(0)
        PRINT
        PRINT #0, 'H';CHR$(13);'B24'
        IF D_PTR% = 0  AND  N% = 1  GOTO QUIT
        PRT_SW% =0:\
        FOR REC_PTR% = 1 TO  N%
            GET #1, RECORD REC_PTR%
            IF REC_PTR% = 1  THEN  M% = 1\
                          ELSE  M% = 0
            FOR ELEM_PTR% = M% TO 13
                A$ = ELEM$(ELEM_PTR%)
                IF A$ <> '0000000000' \
                AND A$ <> 'ZZZZZZZZZZ' \
                GOTO FRST_45
                    PRINT #0, 'R'
                    GOTO FRST_99
FRST_45             FOR K% = 0 TO 3
                    P%(K%) = CVT$%(P$(K%,ELEM_PTR%))
                    P(K%) = P%(K%)
                NEXT K%
                IF  A$ = 'LINE'  GOTO L_1
                IF  A$ = "RECT"  GOTO R_1
                IF  A$ = "CIRCLE" GOTO A_1
                IF A$ = 'H_CIR' GOTO A_1
                IF  A$ = "Q_CIR"  GOTO A_1
                IF LEFT$(A$,1) = 'P' GOTO P_1
                IF A$ = 'DASH_ON' THEN PRINT #0, 'L1'
                IF A$ = 'DASH_OFF' THEN PRINT #0, 'L0'
                IF A$ <> 'N6' GOTO FRST_99
                  XP1% = (P(0) - X_LOW)/SIZE + XP_LOW
                  YP1% = (P(1) - Y_LOW)/SIZE + YP_LOW
                  IF XP1% < 0 OR XP1% > 3600 \
                  OR YP1% < 0 OR YP1% > 2600 \
                  GOTO FRST_99
                  PRINT #0,'M';STR$(XP1%);',';STR$(YP1%)
                     S1% = 4/SIZE
```

```
                              IF S1% > 15 THEN S1% = 15
                              PRINT#0,'S';STR$(S1%);CHR$(13);'N6'
                              PRINT #0,  'R'
                              GOTO FRST_99
*
L_1                     GOSUB  LIN_1
                        PRT_SW% = 0:\
                        GOTO FRST_99
*
R_1                     GOSUB  REC_1
                        PRT_SW% = 0:\
                        GOTO FRST_99
*
A_1                     GOSUB  AR_1
                        PRT_SW% = 0:\
                        GOTO FRST_99
*
P_1                     GOSUB  PRNTER
FRST_99           NEXT ELEM_PTR%
             NEXT REC_PTR%
QUIT         PRINT'PLOT COMPLETE--'
             PRINT #0, 'H'
             CLOSE 1
             EXEC, "TTYSET,PS=Y"
             CHAIN "0.PREPLOT"
*
*
*
PRNTER       CODE% = P%(3) :\
             FOR I% = 1 TO 9 :\
                  IF RIGHT$(A$,1) <> ' ' GOTO PRN_5
                       A$ = LEFT$(A$,LEN(A$)-1) :\
             NEXT I%
PRN_5        XP1 = (P(0) - X_LOW)/SIZE + XP_LOW
             YP1 = (P(1) - Y_LOW)/SIZE + YP_LOW
             IF XP1 < XP_LOW OR XP1 > XP_HIGH \
             OR YP1 < YP_LOW OR YP1 > YP_HIGH THEN RETURN
             XP1% = XP1
             YP1% = YP1
             XP3 = XP1
             YP3 = YP1
             PRINT #0, 'M'; STR$(XP1%); ','; STR$(YP1%)
             IF CODE% > 16      GOTO PRN_50
                  CODE = CODE%
                  CODE% = CODE/SIZE
                  IF CODE% > 15 THEN CODE% = 15
                  PRINT #0, 'Q0': PRINT #0, 'S'; STR$(CODE%)
                  PRINT #0, A$
                  RETURN
PRN_50       CODE% = CODE% - 256 :\
                  CODE = CODE%
```

```
                CODE% = CODE/SIZE
                IF CODE% > 15 THEN CODE% = 15
                PRINT #0, 'Q1': PRINT #0, 'S'; STR$(CODE%)
                PRINT #0, A$
                RETURN
*
*
LIN_1       X1=P(0)
            Y1=P(1)
            X2=P(2)
            Y2=P(3)
LIN_2       GOSUB WINDOW
            RETURN
*
*
REC_1       IF P(3) <= SIZE THEN REC_5
                IF P(2) <= SIZE THEN REC_2
                    BASE= P(2)
                    HEIGHT= P(3)
                    X1=P(0)
                    Y1=P(1)
                    X2=X1 + BASE
                    Y2=Y1
REC_1_4             GOSUB WINDOW
                    X1=X2
                    Y2=Y1 + HEIGHT
REC_1_6             GOSUB WINDOW
                    Y1=Y2
                    X2=X1 - BASE
REC_1_7             GOSUB WINDOW
                    X1=X2
                    Y1=Y2
                    Y2=Y1 - HEIGHT
REC_1_8             GOSUB WINDOW
                    RETURN
REC_2           IF P(3) <= SIZE THEN REC_4
                    X1=P(0)
                    Y1=P(1)
                    X2 = P(0)
                    Y2 = P(1) + HEIGHT
REC_3               GOSUB WINDOW
                    RETURN
REC_4           X1=P(0)
                Y1=P(1)
                X2=P(0)
                Y2=P(1)
                GOSUB WINDOW
                RETURN
REC_5       IF P(2) <= SIZE  GOTO REC_4
            X1=P(0)
            Y1=P(1)
```

```
            X2=P(0) + BASE
            Y2=P(1)
            GOSUB WINDOW
            RETURN
*
AR_1        RAD = P(2)
            IF RAD/SIZE >= 20   GOTO AR_1_2
                 INCR = PI/4
                 GOTO AR_2
AR_1_2      IF RAD/SIZE >= 100  GOTO AR_1_3
                 INCR = PI/6
                 GOTO AR_2
AR_1_3      IF RAD/SIZE >= 1000 GOTO AR_1_4
                 INCR = PI/12
                 GOTO AR_2
AR_1_4      IF RAD/SIZE >= 3000 GOTO AR_1_5
                 INCR = PI/18
                 GOTO AR_2
AR_1_5      INCR = PI/36
AR_2        ANGLE = 0
            ANGLE2 = 2*PI
            CODE% = P%(3)
            S1 = ABS(X_HIGH - P(0))
            S2 = ABS(X_LOW - P(0))
            S3 = ABS(Y_HIGH - P(1))
            S4 = ABS(Y_LOW - P(1))
            X1 = P(0)
            Y1 = P(1)
            IF X1 < X_LOW  GOTO AR_16
              IF XI > X_HIGH  GOTO AR_10
                IF Y1 < Y_LOW  GOTO AR_8
                  IF Y1 > Y_HIGH  GOTO AR_6
                       IF  RAD > S1 \
                       OR RAD > S2 \
                       OR RAD > S3 \
                       OR RAD > S4 GOTO AR_4
                          IF A$ <> 'CIRCLE' GOTO AR_50
                          XP2%=((P(0)+RAD)-X_LOW)/SIZE+XP_LOW
                          YP2%=(P(1)-Y_LOW)/SIZE+YP_LOW
                          PRINT #0,\
                             'M'; STR$(XP2%); ','; STR$(YP2%)
AR_3                      IF ANGLE + INCR < ANGLE2 \
                          THEN ANGLE = ANGLE + INCR \
                          ELSE ANGLE = ANGLE2
                     XP2%=((P(0)+RAD*COS(ANGLE))-X_LOW)/SIZE+XP_LOW
                     YP2%=((P(1)+RAD*SIN(ANGLE))-Y_LOW)/SIZE+YP_LOW
                          PRINT #0,\
                            'D'; STR$(XP2%); ','; STR$(YP2%)
                          IF  ANGLE < ANGLE2  GOTO AR_3
                          PRINT #0, 'R'
                          XP3 = XP2
```

```
                        YP3 = YP2
                        RETURN
AR_4                  IF   RAD > SQR(S1^2 + S3^2) \
                      AND  RAD > SQR(S1^2 + S4^2) \
                      AND  RAD > SQR(S2^2 + S3^2) \
                      AND  RAD > SQR(S2^2 + S4^2) \
                      GOTO AR_120
                           IF A$ = 'CIRCLE' GOTO AR_90
                              GOTO AR_50
AR_6              IF RAD < S3 THEN AR_120
                  IF   RAD > SQR(S1^2 + S4^2) \
                  AND  RAD > SQR(S4^2 + S2^2) THEN AR_120
                  IF A$<>'CIRCLE' GOTO AR_50
                       ANGLE = -PI
                       ANGLE2 =  0
                       GOTO AR_90
AR_8            IF  RAD < S4  THEN AR_120
                IF  RAD > SQR(S3^2 + S1^2) \
                AND RAD > SQR(S3^2 + S2^2) THEN AR_120
                     IF A$<>'CIRCLE' GOTO AR_50
                        ANGLE = 0
                        ANGLE2 = PI
                        GOTO AR_90
AR_10         IF Y1 > Y_HIGH  GOTO AR_14
                    IF Y1 < Y_LOW  GOTO AR_12
                         IF RAD <S1 THEN AR_120
                         IF  RAD > SQR(S2^2 + S3^2) \
                         AND RAD > SQR(S2^2 + S4^2) THEN AR_120
                         IF A$<>'CIRCLE' GOTO AR_50
                              ANGLE = PI/2
                              ANGLE2 = 3*PI/2
                              GOTO AR_90
AR_12               IF RAD < SQR(S1^2 + S4^2) THEN AR_120
                    IF RAD > SQR(S2^2 + S3^2)  THEN AR_120
                         IF A$<>'CIRCLE' GOTO AR_50
                              ANGLE = PI/2
                              ANGLE2 = PI
                              GOTO AR_90
AR_14       IF RAD < SQR(S1^2 + S3^2)  THEN AR_120
            IF RAD > SQR(S2^2 + S4^2)  THEN AR_120
                 IF A$<>'CIRCLE' GOTO AR_50
                 ANGLE = PI
                 ANGLE2 = 3*PI/2
                 GOTO AR_90
AR_16      IF Y1 >Y_HIGH  GOTO AR_20
                IF Y1 < Y_LOW  GOTO AR_18
                     IF RAD < S2 THEN AR_120
                     IF  RAD > SQR(S1^2 + S3^2) \
                     AND RAD > SQR(S1^2 + S4^2) THEN AR_120
                          IF A$<>'CIRCLE' GOTO AR_50
                          ANGLE = -PI/2
```

```
                        ANGLE2 = PI/2
                        GOTO AR_90
AR_18           IF RAD < SQR(S2^2 + S4^2) THEN AR_120
                IF RAD > SQR(S1^2 + S3^2) THEN AR_120
                        IF A$<>'CIRCLE' GOTO AR_50
                        ANGLE = 0
                        ANGLE2 = PI/2
                        GOTO AR_90
AR_20       IF RAD < SQR(S2^2 + S3^2)  THEN AR_120
            IF RAD > SQR(S1^2 + S4^2)  THEN AR_120
                IF A$<>'CIRCLE' GOTO AR_50
                    ANGLE = 3*PI/2
                    ANGLE2 = 2*PI
                    GOTO AR_90
AR_50       IF A$ = 'H_CIR' GOTO AR_70
                IF CODE%  > 0 GOTO AR_55
                    ANGLE = 0
                    ANGLE2 = PI/2
                    GOTO AR_90
AR_55           IF CODE%  > 1 GOTO AR_60
                    ANGLE = PI/2
                    ANGLE2 = PI
                    GOTO AR_90
AR_60           IF CODE% > 3 GOTO AR_65
                    ANGLE = -PI
                    ANGLE2 = -PI/2
                    GOTO AR_90
AR_65           ANGLE = -PI/2
                ANGLE2 = 0
                GOTO AR_90
AR_70       IF CODE% > 0 GOTO AR_75
                ANGLE = 0
                ANGLE2 = PI
                GOTO AR_90
AR_75       IF CODE% > 1 GOTO AR_80
                ANGLE = PI/2
                ANGLE2 = 3*PI/2
                GOTO AR_90
AR_80       IF CODE% > 2 GOTO AR_85
                ANGLE = PI
                ANGLE2 = 2*PI
                GOTO AR_90
AR_85       ANGLE = -PI/2
            ANGLE2 = PI/2
AR_90       X2 = P(0) + RAD*COS(ANGLE) + .5
            Y2 = P(1) + RAD*SIN(ANGLE) + .5
AR_100      X1 = X2
            Y1 = Y2
            IF ANGLE+INCR < ANGLE2 THEN ANGLE = ANGLE+INCR\
                                   ELSE ANGLE = ANGLE2
            X2 = P(0) + RAD*COS(ANGLE) + .5
```

```
          Y2 = P(1) + RAD*SIN(ANGLE) + .5
          GOSUB  WINDOW
AR_110    IF  ANGLE < ANGLE2  GOTO AR_100
          WIN_SW% = 0
AR_120    RETURN
*
WINDOW    IF  X2 >= X1  GOTO  W_0_3
              SWAP X1, X2
              SWAP Y1, Y2
              SWAP_SW% = 1
W_0_3     IF X2 = X1 THEN W_0_5
              NUM = Y2 - Y1
              DEN = X2 - X1
              SLOPE = NUM/DEN
              GOTO W_0_6
W_0_5     SLOPE = 32767
W_0_6     IF WIN_SW% = 0  GOTO W_0_8
              XP1 = X1
              XP2 = X2
              YP1 = Y1
              YP2 = Y2
              GOTO DRAW_PREP
W_0_8     IF X1 > X_HIGH THEN OUT
          IF X1 < X_LOW THEN W_SIX
              IF Y1 > Y_HIGH THEN W_FOUR
                    IF Y1 < Y_LOW THEN W_TWO
*                        P1 ON SCREEN
                         XP1 = X1
                         YP1 = Y1
                         IF X2 > X_HIGH THEN W_ONE
*                             P2 IN COL 2
                              IF Y2 > Y_HIGH THEN TOP_BND
                                   IF Y2 < Y_LOW THEN BOT_BND
*                                       P2 ON SCREEN
                                        XP2 = X2
                                        YP2 = Y2
                                        GOTO DRAW_PREP
*                        P2 IN COL 3
W_ONE                    SCRATCH = X_HIGH - X1
                         YTEST = SLOPE*SCRATCH
                         YTEST = YTEST + Y1
                         IF YTEST > Y_HIGH THEN TOP_BND
                              IF YTEST < Y_LOW THEN BOT_BND
                                   XP2 = X_HIGH
                                   YP2 = YTEST
                                   GOTO DRAW_PREP
W_TWO               IF X2 > X_HIGH THEN W_THREE
                         IF Y2 < Y_LOW THEN OUT
                         XP1 = (YLOW-Y1)/SLOPE + X1
                         IF X1 = X2 THEN XP1 = X1
                         YP1 = Y_LOW
```

```
                              IF Y2 > Y_HIGH THEN TOP_BND
                                   XP2 = X2
                                   YP2 = Y2
                                   GOTO DRAW_PREP
W_THREE                  SCRATCH = X_HIGH - X1
                         YTEST = SLOPE*SCRATCH
                         YTEST = YTEST + Y1
                         IF YTEST < Y_LOW THEN OUT
                         SCRATCH = Y_LOW - Y1
                         XP1 = SCRATCH/SLOPE + X1
                         IF X1 = X2 THEN XP1 = X1
                         YP1 = Y_LOW
                         IF YTEST > Y_HIGH THEN TOP_BND
                              YP2 = YTEST
                              XP2 = X_HIGH
                              GOTO DRAW_PREP
W_FOUR              IF X2 > X_HIGH THEN W_FIVE
                         IF Y2 > Y_HIGH THEN OUT
                         SCRATCH = Y_HIGH - Y1
                         XP1 = SCRATCH/SLOPE + X1
                         IF X1 = X2 THEN XP1 = X1
                         YP1 = Y_HIGH
                         IF Y2 < Y_LOW THEN BOT_BND
                              XP2 = X2
                              YP2 = Y2
                              GOTO DRAW_PREP
W_FIVE              SCRATCH = X_HIGH - X1
                    YTEST = SLOPE*SCRATCH
                    YTEST = YTEST + Y1
                    IF YTEST > Y_HIGH THEN OUT
                    SCRATCH = Y_HIGH - Y1
                    XP1 = SCRATCH/SLOPE + X1
                    YP1 = Y_HIGH
                    IF YTEST < Y_LOW THEN BOT_BND
                         XP2 = X_HIGH
                         YP2 = YTEST
                         GOTO DRAW_PREP
W_SIX          IF X2 <= X_LOW THEN OUT
               SCRATCH = X_LOW  - X1
               YTEST = SLOPE*SCRATCH
               YTEST = YTEST + Y1
               IF Y1 > Y_HIGH THEN W_14_5
                    IF Y1 < Y_LOW THEN W_EIGHT
                         IF YTEST >= Y_HIGH THEN OUT
                         IF YTEST <= Y_LOW THEN OUT
                         YP1 = YTEST
                         XP1 = X_LOW
                         IF X2 > X_HIGH THEN W_SEVEN
                              IF Y2 > Y_HIGH THEN TOP_BND
                                   IF Y2 < Y_LOW THEN BOT_BND
                                        YP2 = Y2
```

```
                                        XP2 = X2
                                        GOTO DRAW_PREP
W_SEVEN                 SCRATCH = X_HIGH - X1
                        YTEST = SLOPE*SCRATCH
                        YTEST = YTEST + Y1
                        IF YTEST > Y_HIGH THEN TOP_BND
                             IF YTEST < Y_LOW THEN BOT_BND
                                  YP2 = YTEST
                                  XP2 = X_HIGH
                                  GOTO DRAW_PREP
W_EIGHT             IF YTEST > Y_HIGH THEN OUT
                    IF Y2 < Y_LOW THEN OUT
                    IF X2 > X_HIGH THEN W_TWELVE
                         IF Y2 <= Y_HIGH THEN W_TEN
                              IF YTEST >= Y_LOW THEN W_NINE
                                   SCRATCH = Y_LOW - Y1
                                   XP1 = SCRATCH/SLOPE + X1
                                   YP1 = Y_LOW
                                   GOTO TOP_BND
W_NINE                             XP1 = X_LOW
                                   YP1 = YTEST
                                   GOTO TOP_BND
W_TEN                   XP2 = X2
                        YP2 = Y2
                        IF YTEST >= Y_LOW THEN W_ELEVEN
                             SCRATCH = Y_LOW - Y1
                             XP1 = SCRATCH/SLOPE + X1
                             YP1 = Y_LOW
                             GOTO DRAW_PREP
W_ELEVEN                XP1 = X_LOW
                        YP1 = YTEST
                        GOTO DRAW_PREP
W_TWELVE            IF YTEST < Y_LOW THEN W_13
                             YP1 = YTEST
                             XP1 = X_LOW
                             GOTO W_14
W_13                SCRATCH = Y_LOW - Y1
                    XP1 = SCRATCH/SLOPE + X1
                    YP1 = Y_LOW
W_14                SCRATCH = X_HIGH - X1
                    YTEST = SLOPE*SCRATCH
                    YTEST = YTEST + Y1
                    IF YTEST > Y_HIGH THEN TOP_BND
                         YP2 = YTEST
                         XP2 = X_HIGH
                         GOTO DRAW_PREP
TOP_BND             SCRATCH = Y_HIGH - Y1
                    XP2 = SCRATCH/SLOPE + X1
                    YP2 = Y_HIGH
                    GOTO DRAW_PREP
W_14_5          IF YTEST < Y_LOW THEN OUT
```

```
            IF Y2 > Y_HIGH THEN OUT
            IF X2 > X_HIGH THEN W_18
                IF Y2 >= Y_LOW THEN W_16
                     IF YTEST >= Y_HIGH THEN W_15
                          YP1 = YTEST
                          XP1 = X_LOW
                          GOTO BOT_BND
W_15                 SCRATCH = Y_HIGH - Y1
                     XP1 = SCRATCH/SLOPE + X1
                     YP1 = Y_HIGH
                     GOTO BOT_BND
W_16            XP2 = X2
                YP2 = Y2
                IF YTEST < Y_HIGH THEN W_17
                     SCRATCH = Y_HIGH - Y1
                     XP1 = SCRATCH/SLOPE + X1
                     YP1 = Y_HIGH
                     GOTO DRAW_PREP
W_17            XP1 = X_LOW
                YP1 = YTEST
                GOTO DRAW_PREP
W_18        IF YTEST > Y_HIGH THEN W_19
                YP1 = YTEST
                XP1 = X_LOW
                GOTO W_20
W_19        SCRATCH = X_HIGH - X1
            YTEST = SLOPE*SCRATCH
            YTEST = YTEST + Y1
            IF YTEST >= Y_HIGH  GOTO  OUT
            SCRATCH = Y_HIGH - Y1
            XP1 = SCRATCH/SLOPE + X1
            YP1 = Y_HIGH
W_20        SCRATCH = X_HIGH - X1
            YTEST = SLOPE*SCRATCH
            YTEST = YTEST + Y1
            IF YTEST < Y_LOW THEN BOT_BND
                YP2 = YTEST
                XP2 = X_HIGH
                GOTO DRAW_PREP
BOT_BND     SCRATCH = Y_LOW - Y1
            XP2 = SCRATCH/SLOPE + X1
            YP2 = Y_LOW
DRAW_PREP   IF  SWAP_SW% = 0  GOTO PREP_2
                SWAP XP1,XP2
                SWAP YP1,YP2
                SWAP X1, X2
                SWAP Y1, Y2
                SWAP_SW% = 0
PREP_2      XP1% = (XP1 - X_LOW)/SIZE + XP_LOW
            YP1% = (YP1 - Y_LOW)/SIZE + YP_LOW
            XP2% = (XP2 - X_LOW)/SIZE + XP_LOW
```

```
            YP2% = (YP2 - Y_LOW)/SIZE + YP_LOW
DIS_1       IF XP1 = XP3 AND YP1 = YP3  GOTO DIS_2
                 PRINT #0, 'M'; STR$(XP1%); ','; STR$(YP1%)
DIS_2       PRINT #0, 'D'; STR$(XP2%); ','; STR$(YP2%)
            XP3 = XP2
            YP3 = YP2
            RETURN
OUT         IF SWAP_SW% = 0 THEN RETURN
                SWAP X1, X2
                SWAP Y1, Y2
                SWAP_SW% = 0
                RETURN
*
            END
```

Macro

```
            DIM P%(3), ELEM$(13), P$(3,13), X_1$(13)
            DIM NAM$(13), REC$(13), EL_PTR$(13), NO_EL$(13)
            DIM       ROT_CN$(13)
*
            SIZ = HEX("9750")
            POINTR = DPEEK(52267) - 2
            CLEAR = HEX("9C7D")
            GRAPH = HEX("9AE5")
            PLOT = HEX("9C60")
            ERASE = HEX("9758")
            X_LOC = HEX("9700")
            Y_LOC = HEX("9702")
            G_X1 = HEX("970C")
            G_X2 = HEX("9710")
            G_Y1 = HEX("970E")
            G_Y2 = HEX("9712")
            X1 = HEX("9704")
            Y1 = HEX("9706")
            X2 = HEX("9708")
            Y2 = HEX("970A")
            X_LO = HEX("9746")
            Y_LO = HEX("9748")
            X_HI = HEX("974A")
            Y_HI = HEX("974C")
            XCUR = HEX("E008")
            YCUR = HEX("E00A")
            BUTTON = HEX("E009")
            WINDO = HEX("9E0D")
            CODE = HEX("9773")
            CIR    = HEX("A210")
            Q_CIR = HEX("A37B")
            H_CIR = HEX("A3A8")
```

```
            RADIUS = HEX("976D")
            WIN_SW = HEX("9753")
            PRNTR = HEX("A3EB")
            CHPTR = HEX("9781")
            X0 = HEX("9784")
            Y0 = HEX("9786")
            CRT_X% = 255
            CRT_Y% = 191
            EXEC,'TTYSET,PS=N'
*
*
            PRINT CHR$(26); TAB(30); 'DEFINE AN ELEMENT'
*
*
            OPEN 'MAC_ELEM' AS 2
            FOR I% = 0 TO 13
                 FIELD #2, 18*I% AS Z$,\
                       10 AS ELEM$(I%),\
                       2 AS P$(0,I%),\
                       2 AS P$(1,I%),\
                       2 AS P$(2,I%),\
                       2 AS P$(3,I%)
            NEXT I%
*
            OPEN 'MAC_DIR' AS 1
            FOR I% = 0 TO 13
                 FIELD #1, 18*I% AS Z$,\
                       10 AS  NAM$(I%),\
                      2 AS    REC$(I%),\
                      2 AS EL_PTR$(I%),\
                      2 AS  NO_EL$(I%),\
                      2 AS ROT_CN$(I%)
           NEXT I%
INIT_0     GET #1, RECORD 1
           M_DIR_PTR% = CVT$%(EL_PTR$(0))
           LAST_M_DIR_REC% = CVT$%(REC$(0))
            NO_MAC% = CVT$%(NO_EL$(0))
            DPOKE SIZ, 1
           X_MAX% = 255
           Y_MAX% = 191
           X_LOW% = 0
           Y_LOW% = 0
           X_HIGH% = 255
            Y_HIGH% = 191
INIT_2     GOSUB SCRN_WIP
           DPOKE X_LO,X_LOW%
           DPOKE Y_LO,Y_LOW%
           DPOKE X_HI,X_HIGH%
           DPOKE Y_HI,Y_HIGH%
            P%(0) = X_LOW%
            P%(1) = Y_LOW%
```

```
          P%(2) = X_HIGH% - X_LOW%
          P%(3) = Y_HIGH% - Y_LOW%
         DPOKE POINTR, GRAPH
          DPOKE X1, 0
          DPOKE X2, 255
          DPOKE Y1, 0
          DPOKE Y2, 0
         DUM% = USR(0)
          DPOKE X1,255
          DPOKE Y2, 191
         DUM% = USR(0)
          DPOKE X1, 0
          DPOKE Y1, 191
         DUM% = USR(0)
          DPOKE X2, 0
          DPOKE Y2, 0
         DUM% = USR(0)
          DPOKE X1,  64
          DPOKE Y1,  96
          DPOKE X2, 192
          DPOKE Y2,  96
         DUM% = USR(0)
          DPOKE X1, 128
          DPOKE Y1,  32
          DPOKE X2, 128
          DPOKE Y2, 160
         DUM% = USR(0)
*
INIT_3    PRINT
          PRINT
          PRINT TAB(33);'NAME?'
          PRINT
          PRINT TAB(32); ' ';
          LENGTH% = 9
          GOSUB NAMR
          IF CR_SW% = 1 GOTO EXIT
          A$ = 'M' + A$
          IF M_DIR_PTR% = 0  AND  LAST_M_DIR_REC% = 1  \
                                        GOTO INIT_5
          FOR J% = 1 TO  LAST_M_DIR_REC%
               GET #1, RECORD J%
               FOR I% =  0 TO 13
                    IF A$ = NAM$(I%) GOTO INIT_20
               NEXT I%
          NEXT J%
          PRINT'NEW ELEMENT.'
INIT_5    LAST_M_ELEM_REC% = CVT$%(REC$(M_DIR_PTR%))
          M_ELEM_PTR% = CVT$%(EL_PTR$(M_DIR_PTR%))
          NO_ELEM% = CVT$%(NO_EL$(M_DIR_PTR%))
          A = M_ELEM_PTR% + NO_ELEM% + 14*LAST_M_ELEM_REC%
          M_ELEM_PTR% = INT(A)
```

```
INIT_6     IF M_ELEM_PTR% < 14 GOTO INIT_7
           M_ELEM_PTR% = M_ELEM_PTR% - 14
           GOTO INIT_6
INIT_7     M_ELEM_PTR% = M_ELEM_PTR% - 1
           A = A/14
           LAST_M_ELEM_REC% = INT(A)
INIT_8     M_DIR_PTR% = M_DIR_PTR% + 1
            IF M_DIR_PTR% <= 13  GOTO INIT_10
                 M_DIR_PTR% = 0
                 LAST_M_DIR_REC% = LAST_M_DIR_REC% + 1
INIT_10    LSET NAM$(M_DIR_PTR%) = A$
           LSET REC$(M_DIR_PTR%) = CVT%$(LAST_M_ELEM_REC%)
           LSET EL_PTR$(M_DIR_PTR%) = CVT%$(M_ELEM_PTR% + 1)
           NO_ELEM% = 0
           IF M_ELEM_PTR% = -1 GOTO INIT_12
                GET #2, RECORD LAST_M_ELEM_REC%
INIT_12    FOR I% = 1 TO 10: PRINT : NEXT I%
           PRINT TAB(24); 'WHICH DRAWING SPACE DO YOU WANT?'
           PRINT TAB(33); '( ) 1"W BY .75"H'
           PRINT TAB(33); '( ) 2"W BY 1.5"H'
           PRINT TAB(22);\
                   'POSITION THE CURSOR AND PRESS RETURN'
           PRINT CHR$(11); CHR$(11);
           FOR I% = 1 TO 46: PRINT CHR$(8);: NEXT I%
           B% = 1
INIT_14    B$ = INCH$(0)
           IF B$ = CHR$(13) GOTO INIT_16
           IF B$ <> CHR$(11) GOTO INIT_15
                B% = B% - 1
                GOTO INIT_14
INIT_15    IF B$ <> CHR$(10) GOTO INIT_14
                B% = B% + 1
                GOTO INIT_14
INIT_16    IF B% > 0 AND B% < 3 GOTO INIT_50
                PRINT CHR$(26); "LET'S TRY AGAIN.";
                GOTO INIT_12
*
INIT_20    PRINT   A$;' ALREADY EXISTS,'
           PRINT    'HERE ARE YOUR CHOICES.'
           PRINT    '   (1) DELETE THE ORIGINAL'
           PRINT    '   (2) ENTER A NEW NAME '
           PRINT    '   (3) EXIT THIS PROCEDURE'
           PRINT
           PRINT'WHAT DO YOU WANT TO DO?';
           B$ = INCH$(0)
           IF B$ = '1'  GOTO  INIT_30
           IF B$ = '2'  GOTO  INIT_3
           IF B$ = '3'  GOTO EXIT
           GOTO  INIT_20
INIT_30    LSET NAM$(I%) = 'ZZZZZZZZZZ'
           IF NO_MAC% >= 1 THEN NO_MAC% = NO_MAC% - 1
```

```
            PUT #1, RECORD J%
            GOTO INIT_3
INIT_50     FOR I% = 1 TO 10: PRINT: NEXT I%:\
            ROT_CN% = 0
            PRINT 'DO YOU WANT EXTERNAL CONNECTIONS?';
            B$ = INCH$(0)
            PRINT
            IF B$ = 'Y' GOTO INIT_60
                 ROT_CN% = 256
                 GOTO DRAW_1
INIT_60     A$ = 'CONNECTOR'
            FOR I% = 0 TO 3: P%(I%) = 0: NEXT I%:\
            GOSUB UPDATE
            GOTO DRAW_1
EXIT        CLOSE 1
            CLOSE 2
            CHAIN "0.MADS-3"
*
*
*
SCRN_WIP    DPOKE POINTR,CLEAR
            DUM%=USR(0)
            RETURN
*
*
DRAW_1      DR_SW% = 1
            CONV = 256
            SIZE% = B%
            SIZE = SIZE%
            DPOKE SIZ, SIZE%
            X_HIGH% = CRT_X%*SIZE%
            Y_HIGH% = CRT_Y%*SIZE%
            X_MAX% = X_HIGH%
            Y_MAX%  = Y_HIGH%
            POKE ERASE,0
           DPOKE X_LO,X_LOW%
           DPOKE Y_LO,Y_LOW%
           DPOKE X_HI,X_HIGH%
           DPOKE Y_HI,Y_HIGH%
            DPOKE POINTR, PLOT
            FOR X1% = 0 TO CRT_X% STEP 16/SIZE%
                 FOR Y1% = 0 TO CRT_Y% STEP 16/SIZE%
                      DPOKE X_LOC, X1%
                      DPOKE Y_LOC, Y1%
                      DUM% = USR(0)
                 NEXT Y1%
            NEXT X1%
            DPOKE POINTR, WINDO
DRAW_OUT    PRINT CHR$(26)
DRAW_3      PRINT CHR$(30); TAB(32); 'ELEMENT MENU'
            DN% = 1
```

```
          RIGHT% = 1
          PRINT TAB(21); CHR$(27);'1';TAB(43);CHR$(27);'1'
          PRINT TAB(20); '( ) REFERENCE DOT';
          PRINT TAB(41); '( ) TOP    HALF CIRCLE'
          PRINT TAB(20); '( ) CURSOR POSITION';
          PRINT TAB(41); '( ) LEFT   HALF CIRCLE'
          PRINT TAB(20); '( ) LINE';
          PRINT TAB(41); '( ) BOTTOM HALF CIRCLE'
          PRINT TAB(20); '( ) RECTANGLE';
          PRINT TAB(41); '( ) RIGHT  HALF CIRCLE'
          PRINT TAB(20); '( ) CIRCLE';
          PRINT TAB(41); '( ) UPPER RIGHT 1/4 CIRCLE'
          PRINT TAB(20); '( ) HORIZONTAL TEXT';
          PRINT TAB(41); '( ) UPPER LEFT  1/4 CIRCLE'
          PRINT TAB(20); '( ) VERTICAL   TEXT';
          PRINT TAB(41); '( ) LOWER LEFT  1/4 CIRCLE'
          PRINT TAB(20); '( ) ERASE LAST ITEM';
          PRINT TAB(41); '( ) UPPER LEFT  1/4 CIRCLE'
          PRINT TAB(20); '( ) QUIT'
          PRINT
          PRINT TAB(20); 'POSITION CURSOR AND PRESS RETURN';
          FOR I% = 1 TO 10
               PRINT CHR$(11); CHR$(8);
          NEXT I%
          FOR I% = 1 TO 21
               PRINT CHR$(8);
          NEXT I%
DRAW_5    B$ = INCH$(0)
          IF B$ >= CHR$(08) AND B$ <= CHR$(13)  GOTO DRAW_10
               PRINT CHR$(8)
               GOTO DRAW_5
DRAW_10   IF B$ = CHR$(13) GOTO DRAW_20
          IF B$ = CHR$(12) THEN RIGHT% = RIGHT% + 1
          IF B$ = CHR$(11) THEN DN% = DN% - 1
          IF B$ = CHR$(10) THEN DN% = DN% + 1
          IF B$ = CHR$(09) THEN RIGHT% = 21
          IF B$ = CHR$(08) THEN RIGHT% = RIGHT% - 1
          GOTO DRAW_5
DRAW_20   IF RIGHT% > 60 OR RIGHT% < 0 \
               OR DN% > 9  OR  DN% < 1 GOTO DRAW_OUT
          FOR I% = 1 TO 11-DN%
               PRINT
          NEXT I%
          PRINT CHR$(27); CHR$(89)
          IF RIGHT% > 20 GOTO DRAW_30
          IF DN% = 1 GOTO DOT
          IF DN% = 2 GOTO CURSOR
          IF DN% = 3 GOTO LINE_R
          IF DN% = 4 GOTO RECTANGL
          IF DN% = 5 GOTO CIRCLE
          IF DN% = 6 GOTO PRNT_H
```

```
            IF DN% = 7 GOTO PRNT_V
            IF DN% = 8 GOTO ER_LAST
            IF DN% = 9 GOTO QUIT_1
            GOTO DRAW_OUT
DRAW_30     IF DN% >= 5  GOTO DRAW_40
                 P%(3) = DN% - 1
                 A$ = 'H_CIR'
                 GOTO CRCL_10
DRAW_40     P%(3) = DN% - 5
                 A$ = 'Q_CIR'
                 GOTO CRCL_10
SW_TEST     POKE ERASE, 0
            BND_SW% = 0
                GOTO DRAW_OUT
*
CONST       DR_SW% = 0
            POKE ERASE,0
            GOTO DRAW_OUT
*
ERASE_I     DR_SW% = -1
            GOTO INIT_2
*
DRAW_2      DR_SW% = 1
            POKE ERASE,0
            GOTO DRAW_OUT
*
DOT         POKE BUTTON, 0
            PRINT 'PRESS THE RED BUTTON TO PLACE ';
            PRINT 'A DOT AT THE CURSOR LOCATION'
            GOSUB RED_B
            DPOKE X_LOC, PEEK(XCUR)
            DPOKE Y_LOC, (191 - PEEK(YCUR))
            DPOKE POINTR, PLOT
            DUM% = USR(0)
            GOTO SW_TEST
*
CURSOR      POKE BUTTON,0 :\
            PRINT ' PRESS THE RED BUTTON FOR THE MENU'
CUR_20      GOSUB POSITION
            A = (X% - 128*SIZE%)/CONV :\
            B = (Y% - 96*SIZE%)/CONV
CUR_25      PRINT CHR$(13); :\
            PRINT "X="; A ;
            PRINT "  Y="; B ; CHR$(27); CHR$(84);
            IF PEEK(BUTTON) = HEX("80") \
                     THEN DRAW_3  ELSE CUR_20
800        REM
ER_LAST     IF LAST_M_DIR_REC% > 1 GOTO ER_L_10
            IF M_ELEM_PTR% = 0 GOTO SW_TEST
ER_L_10     POKE ERASE, 255 :\
            A$ = ELEM$(M_ELEM_PTR%) :\
```

```
            FOR I% = 0 TO 3 :\
                 P%(I%) = CVT$%(P$(I%,M_ELEM_PTR%)) :\
            NEXT I% :\
            IF A$ = 'LINE'     GOTO L_1
            IF A$ = 'RECT'     GOTO R_1
            IF A$ = 'H_CIR'    GOTO A_1
            IF A$ = 'Q_CIR'    GOTO A_1
            IF A$ = 'N6' THEN A$ = 'CIRCLE'
            IF A$ = 'CIRCLE'   GOTO A_1
            IF LEFT$(A$,1) = 'P' GOTO P_1
            GOTO ER_L_15
L_1         GOSUB LIN_1
            GOTO ER_L_15
R_1         GOSUB REC_1
            GOTO ER_L_15
A_1         GOSUB AR_1
            GOTO ER_L_15
P_1         GOSUB PRNTER
ER_L_15     POKE ERASE, 0 :\
            NO_ELEM% = NO_ELEM% - 1 :\
            IF M_ELEM_PTR% = 0  GOTO ER_L_20
                 M_ELEM_PTR% = M_ELEM_PTR% - 1 :\
                 GOTO SW_TEST
ER_L_20     M_ELEM_PTR% = 13 :\
            LAST_M_ELEM_REC% = LAST_M_ELEM_REC% - 1 :\
            GET #2, RECORD LAST_M_ELEM_REC%
            GOTO SW_TEST
900         REM
LINE_R      PRINT"IS THE CURSOR AT X1,Y1(RED BUTTON)?" :\
            GOSUB RED_B
            GOSUB POSITION
            P%(0) = X% :\
            P%(1) = Y% :\
            DPOKE POINTR,PLOT :\
            DPOKE X_LOC,(P%(0)-X_LOW%)/SIZE%
            DPOKE Y_LOC,(P%(1)-Y_LOW%)/SIZE% :\
            DUM% = USR(0) :\
            DPOKE POINTR,WINDO :\
            A% = 0 :\
            FOR I% = 1 TO 100: A% = A% + 1: NEXT I%:\
            A% = 0 :\
            PRINT'CURSOR AT X2,Y2(RED BUTTON)?' :\
            GOSUB RED_B
LINE_R_0    GOSUB POSITION
            P%(2) = X% :\
            P%(3) = Y% :\
            IF P%(0) = P%(2) AND P%(1) = P%(3)  GOTO SW_TEST
LINE_R_2    GOSUB LIN_1
            IF  DR_SW% <= 0  OR  BND_SW% = 1  GOTO SW_TEST
            A$ = 'LINE' :\
            GOSUB UPDATE
```

```
            P%(0) = P%(2) :\
            P%(1) = P%(3) :\
            IF A% = 1 GOTO LINE_R_10
            A% = 1 :\
            PRINT 'PRESS THE RED BUTTON' :\
            PRINT ' AT A NEW LOCATION FOR ANOTHER LINE' :\
            PRINT ' AT THE SAME LOCATION FOR THE MENU';
LINE_R_10   GOSUB RED_B
            GOTO LINE_R_0
1000        REM
RECTANGL    PRINT"CURSOR ON LOWER LEFT CORNER(RED BUTTON)" :\
            GOSUB RED_B
            GOSUB POSITION
            P%(0) = X% :\
            P%(1) = Y% :\
            DPOKE POINTR,PLOT :\
            DPOKE X_LOC,(P%(0)-X_LOW%)/SIZE%
            DPOKE Y_LOC,(P%(1)-Y_LOW%)/SIZE% :\
            DUM% = USR(0) :\
            DPOKE POINTR,WINDO :\
            PRINT'X CURSOR ON RIGHT SIDE, Y CURSOR ON TOP' :\
            GOSUB RED_B
            GOSUB POSITION
            P%(2) = X% :\
            P%(3) = Y% :\
            P%(2) = ABS(P%(0) - P%(2)) :\
            P%(3) = ABS(P%(1) - P%(3)) :\
            GOSUB REC_1
            IF  DR_SW% <= 0  OR  BND_SW% = 1  GOTO SW_TEST
                A$ = 'RECT' :\
                GOSUB UPDATE
                GOTO SW_TEST
1100        REM
CIRCLE      P%(3) = 0 :\
            A$ = 'CIRCLE'
CRCL_10     PRINT"CURSOR AT CENTER(RED BUTTON)" :\
            GOSUB RED_B
            GOSUB POSITION
            P%(0) = X% :\
            P%(1) = Y% :\
            DPOKE POINTR,PLOT
            DPOKE X_LOC,(P%(0)-X_LOW%)/SIZE% :\
            DPOKE Y_LOC,(P%(1)-Y_LOW%)/SIZE%:\
            DUM% = USR(0) :\
            DPOKE POINTR,WINDO
CRCL_20     PRINT"WHAT IS THE RADIUS?"; :\
            GOSUB NUMBR
            IF CR_SW% = 0 GOTO CRCL_30
                A$ = 'N6'
                P%(2) = 12 :\
                GOTO CRCL_40
```

```
CRCL_30    IF A <= 255 AND A > 0 GOTO CRCL_35
               PRINT 'THE RADIUS MUST BE <= 255' :\
               GOTO CRCL_20
CRCL_35    P%(2) = A*CONV
CRCL_40    GOSUB AR_1
           IF  DR_SW% <= 0  GOTO SW_TEST
               GOSUB UPDATE
              GOTO SW_TEST
1200      REM
PRNT_H     CODE% = 0
           GOTO PRNT_10
1300      REM
PRNT_V     CODE% = 256
PRNT_10    PRINT\
       'CURSOR ON LOWER LEFT OF 1st CHARACTER(RED BUTTON)' :\
           GOSUB RED_B
           P%(0) =  PEEK(XCUR)*SIZE% + X_LOW% :\
           P%(1) =  (191-PEEK(YCUR))*SIZE% + Y_LOW%
           CODE% = CODE% + 4  :\
           P%(2) = 0 :\
           P%(3) = CODE% :\
           CODE% = 0 :\
           PRINT'TEXT( <= 9  CHAR ) '; :\
           LENGTH% = 9 :\
           GOSUB NAMR
           IF CR_SW% = 1 GOTO SW_TEST
           B$ = A$ :\
           A$ = 'P' + B$
PRNT_40    GOSUB PRNTER
           IF DR_SW% <= 0 GOTO SW_TEST
           GOSUB UPDATE
           GOTO SW_TEST
*
QUIT_1     GOSUB CLEAN_UP
           LSET NO_EL$(M_DIR_PTR%) = CVT%$(NO_ELEM%)
           LSET ROT_CN$(M_DIR_PTR%) = CVT%$(ROT_CN%)
           M_ELEM_REC% = CVT$%(REC$(M_DIR_PTR%))
           M_ELEM_PTR% = CVT$%(EL_PTR$(M_DIR_PTR%))
           GET #2, RECORD M_ELEM_REC%
           IF ROT_CN% <> 0 GOTO QUIT_8
            IF ELEM$(M_ELEM_PTR%) = 'CONNECTOR' GOTO QUIT_5
                GOTO QUIT_8
QUIT_5      PRINT TAB(27); 'EXTERNAL CONNECTION POINTS':\
            PRINT 'CURSOR AT X1,Y1(LEFT OR TOP--RED BUTTON)';
            GOSUB RED_B
            PRINT
   LSET P$(0,M_ELEM_PTR%)=CVT%$((PEEK(XCUR)-128)*SIZE%)
   LSET P$(1,M_ELEM_PTR%)=CVT%$(((191-PEEK(YCUR))-96)*SIZE%)
           FOR I% = 1 TO 100: A% = A% + 1: NEXT I%
           PRINT \
             'CURSOR AT X2,Y2(RIGHT OR BOTTOM--RED BUTTON)';
```

```
            GOSUB RED_B
            PRINT
   LSET P$(2,M_ELEM_PTR%)=CVT%$((PEEK(XCUR)-128)*SIZE%)
   LSET P$(3,M_ELEM_PTR%)=CVT%$(((191-PEEK(YCUR))-96)*SIZE%)
            I% = 0: NO_ELEM% = NO_ELEM% - 1:\
            GOSUB PTR_UP
QUIT_8     FOR I% = 1 TO NO_ELEM%
              IF ELEM$(M_ELEM_PTR%) = 'ZZZZZZZZZZ' \
                           GOTO QUIT_15
              X1%=CVT$%(P$(0,M_ELEM_PTR%))
              LSET P$(0,M_ELEM_PTR%)=CVT%$(X1% - 128*SIZE%)
              Y1%=CVT$%(P$(1,M_ELEM_PTR%))
              LSET P$(1,M_ELEM_PTR%)=CVT%$(Y1% - 96*SIZE%)
              IF ELEM$(M_ELEM_PTR%) <> 'LINE' GOTO  QUIT_10
                X2%=CVT$%(P$(2,M_ELEM_PTR%))
                LSET P$(2,M_ELEM_PTR%)=CVT%$(X2% - 128*SIZE%)
                Y2%=CVT$%(P$(3,M_ELEM_PTR%))
                LSET P$(3,M_ELEM_PTR%)=CVT%$(Y2% - 96*SIZE%)
QUIT_10         GOSUB PTR_UP
QUIT_15    NEXT I%
           PUT #2, RECORD M_ELEM_REC%
           A$ = '0000000000'
           IF M_DIR_PTR% = 13 GOTO QUIT_30
           FOR J% = M_DIR_PTR%+1 TO 13
                LSET NAM$(J%) = A$
           NEXT J%
QUIT_30    PUT #1, RECORD LAST_M_DIR_REC%
           GET #1, RECORD 1
           LSET REC$(0) = CVT%$(LAST_M_DIR_REC%)
           LSET EL_PTR$(0) = CVT%$(M_DIR_PTR%)
           LSET NO_EL$(0) = CVT%$(NO_MAC% + 1)
           PUT #1, RECORD 1
           PRINT CHR$(26);\
                 'DO YOU WANT TO DO ANOTHER ELEMENT(Y-N)?';
           B$ = INCH$(0)
           IF B$ = 'Y' OR B$ = 'y' GOTO INIT_0
           CLOSE 1
           CLOSE 2
           EXEC,'TTYSET,PS=Y'
           CHAIN "0.MENU"
*
PTR_UP           M_ELEM_PTR% = M_ELEM_PTR% + 1
                 IF M_ELEM_PTR% <= 13  THEN RETURN
                 IF I% = NO_ELEM% THEN RETURN
                      M_ELEM_PTR% = 0
                      PUT #2, RECORD M_ELEM_REC%
                      M_ELEM_REC% = M_ELEM_REC% + 1
                      GET #2, RECORD M_ELEM_REC%
                      RETURN
*
CLEAN_UP   A$ = '0000000000'
```

```
            IF M_ELEM_PTR% = 13  GOTO  CLN_5
            FOR J% = M_ELEM_PTR%+1 TO 13
                 LSET ELEM$(J%) = A$
                 FOR I% = 0 TO 3
                      LSET P$(I%,J%) = CVT%$(0)
                 NEXT I%
            NEXT J%
CLN_5       PUT #2, RECORD LAST_M_ELEM_REC%
            RETURN
*
*
*
LIN_1       X1%=P%(0)
            Y1%=P%(1)
            X2%=P%(2)
            Y2%=P%(3)
LIN_2       GOSUB WIN_CON
            RETURN
*
*
*
REC_1       IF P%(3) <= 1 THEN REC_5
                 IF P%(2) <= 1 THEN REC_2
                      BASE%= P%(2)
                      HEIGHT%= P%(3)
                      X1%=P%(0)
                      Y1%=P%(1)
                      X2%=X1% + BASE%
                      Y2%=Y1%
REC_1_4               GOSUB WIN_CON
                      X1%=X2%
                      Y2%=Y1% + HEIGHT%
REC_1_6               GOSUB WIN_CON
                      Y1%=Y2%
                      X2%=X1% - BASE%
REC_1_7               GOSUB WIN_CON
                      X1%=X2%
                      Y1%=Y2%
                      Y2%=Y1% - HEIGHT%
REC_1_8               GOSUB WIN_CON
                      RETURN
REC_2            IF P%(3) <= 1 THEN REC_4
                      X1%=P%(0)
                      Y1%=P%(1)
                      X2% = P%(0)
                      Y2% = P%(1) + HEIGHT
REC_3                 GOSUB WIN_CON
                      RETURN
REC_4            X1%=P%(0)
                 Y1%=P%(1)
                 X2%=P%(0)
```

```
                Y2%=P%(1)
                GOSUB WIN_CON
                RETURN
REC_5       IF P%(2) <= 1  GOTO REC_4
            X1%=P%(0)
            Y1%=P%(1)
            X2%=P%(0) + BASE
            Y2%=P%(1)
            GOSUB WIN_CON
            RETURN
*
*
*
WIN_CON     IF X1% > X_MAX%  OR  X1% < 0  GOTO BND_ERR
            IF X2% > X_MAX%  OR  X2% < 0  GOTO BND_ERR
            IF Y1% > Y_MAX%  OR  Y1% < 0  GOTO BND_ERR
            IF Y2% > Y_MAX%  OR  Y2% < 0  GOTO BND_ERR
            DPOKE G_X1, X1%
            DPOKE G_Y1, Y1%
            DPOKE G_X2, X2%
            DPOKE G_Y2, Y2%
            DPOKE POINTR, WINDO
            DUM% = USR(0)
            RETURN
BND_ERR     PRINT 'Cannot draw this line--out of bounds.'
            PRINT'X1=';X1%;'Y1=';Y1%;'X2=';X2%;'Y2=';Y2%
            BND_SW% = 1
            RETURN
*
*
AR_1        X1% = P%(0) :\
            Y1% = P%(1) :\
            RAD% = P%(2) :\
            CODE% = P%(3) :\
            S1% = ABS(X_HIGH% - X1%) :\
            S2% = ABS(X_LOW% - X1%)
            S3% = ABS(Y_HIGH% - Y1%) :\
            S4% = ABS(Y_LOW% - Y1%) :\
            IF X1% < X_LOW%  GOTO AR_16
                 IF X1% > X_HIGH%  GOTO AR_10
                      IF Y1% < Y_LOW%  GOTO AR_8
                           IF Y1% > Y_HIGH%  GOTO AR_6
AR_4                            IF   RAD% > SQR(S1%^2 + S3%^2)\
                                AND  RAD% > SQR(S1%^2 + S4%^2)\
                                AND  RAD% > SQR(S2%^2 + S3%^2)\
                                AND  RAD% > SQR(S2%^2 + S4%^2)\
                                          THEN AR_120 \
                                          ELSE  AR_98
AR_6                       IF RAD% < S3% THEN AR_120
                           IF   RAD% > S3% + 230*SIZE%   \
                           OR CODE% = 0  THEN  RETURN
```

```
                              GOTO AR_98
AR_8                  IF  RAD% < S4%  THEN AR_120
                      IF  RAD% > S3% + 230*SIZE%  THEN AR_120
                      IF CODE% = 2 GOTO AR_120
                           GOTO AR_98
AR_10            IF Y1% > Y_HIGH%  GOTO AR_14
                      IF Y1% < Y_LOW%  GOTO AR_12
                           IF RAD% <S1% THEN AR_120
                           IF  RAD% > S1%+273*SIZE% THEN AR_120
                           IF CODE% = 3 GOTO AR_120
                              GOTO AR_98
AR_12                 SCRATCH%= SQR(S1%^2 + S4%^2):\
                      IF RAD% < SCRATCH%            THEN AR_120
                      IF RAD% > SCRATCH%+320*SIZE% THEN AR_120
                       IF CODE% >= 2  GOTO AR_120
                           GOTO AR_98
AR_14                  SCRATCH%= SQR(S1%^2 + S3%^2):\
                       IF RAD% < SCRATCH%             THEN  AR_120
                       IF RAD% > SCRATCH% + SIZE%*320 THEN AR_120
                       IF CODE% = 3  GOTO AR_120
                           GOTO AR_98
AR_16          IF Y1% >Y_HIGH%  GOTO AR_20
                      IF Y1% < Y_LOW%  GOTO AR_18
                           IF RAD% < S2% THEN AR_120
                           IF  RAD% > S2% + 273*SIZE%  THEN AR_120
                            IF  CODE% = 1  GOTO AR_120
                              GOTO AR_98
AR_18                 SCRATCH%= SQR(S2%^2 + S4%^2):\
                      IF RAD% < SCRATCH%            THEN AR_120
                      IF RAD% > SCRATCH% + 320*SIZE% THEN AR_120
                       IF CODE% = 1 OR CODE% =2  GOTO AR_120
                           GOTO AR_98
AR_20          SCRATCH%= SQR(S2%^2 + S3%^2):\
               IF RAD% < SCRATCH%             THEN AR_120
               IF RAD% > SCRATCH% + 320*SIZE% THEN AR_120
                IF  CODE% =1  GOTO AR_120
AR_98          DPOKE RADIUS, RAD% :\
               DPOKE X1, X1% :\
               DPOKE Y1, Y1% :\
               POKE WIN_SW, 1 :\
               IF A$ <> 'CIRCLE'  AND A$ <> 'N6' GOTO AR_110
                    DPOKE POINTR, CIR :\
                    DUM% = USR(0):\
                    RETURN
AR_110         IF A$ = 'H_CIR'  GOTO AR_115
                    DPOKE POINTR, Q_CIR :\
                    DPOKE CODE, P%(3) :\
                    DUM% = USR(0):\
                    RETURN
AR_115         DPOKE POINTR, H_CIR:\
               DPOKE CODE, P%(3):\
```

```
            DUM% = USR(0):\
            RETURN
AR_120      POKE WIN_SW, 0:\
            RETURN
*
*
POSITION    X% = PEEK(XCUR)*SIZE%+X_LOW%:\
            Y% = (191-PEEK(YCUR))*SIZE%+Y_LOW%:\
            IF Y% >= Y_LOW% AND Y% <= Y_HIGH% THEN RETURN
                  PRINT 'INVALID COORDINATE--TRY AGAIN';
                  GOSUB RED_B
                  PRINT
                  GOTO POSITION
*
RED_B       POKE BUTTON, 0
RED_10      IF PEEK(BUTTON) = HEX("80") THEN RETURN
                  GOTO RED_10
*
UPDATE      M_ELEM_PTR% = M_ELEM_PTR% + 1
            NO_ELEM% = NO_ELEM% + 1
            IF M_ELEM_PTR% <= 13  GOTO UP_5
                  PUT #2, RECORD LAST_M_ELEM_REC%
                  LAST_M_ELEM_REC% = LAST_M_ELEM_REC% + 1
                  M_ELEM_PTR% = 0
UP_5        LSET ELEM$(M_ELEM_PTR%) = A$
            FOR I% = 0 TO 3
                  LSET P$(I%,M_ELEM_PTR%) = CVT%$(P%(I%))
            NEXT I%
            RETURN
*
*
PRNTER      CODE% = P%(3) :\
            FOR I% = 1 TO 9 :\
                  IF RIGHT$(A$,1) <> ' ' GOTO PRN_5
                        A$ = LEFT$(A$,LEN(A$)-1) :\
            NEXT I%
PRN_5       IF CODE% > 16      GOTO PRN_10
                  P%(2) = (LEN(A$)-1)*8*CODE% :\
                  P%(3) = 8*CODE% :\
                  GOSUB REC_1
                  P%(2) = 0 :\
                  P%(3) = CODE% :\
                  IF 8*CODE%/SIZE% < 9 THEN RETURN
                  GOTO PRN_20
PRN_10      CODE% = CODE% - 256 :\
            P%(2) = 8*CODE% :\
            P%(0) = P%(0) - P%(2) :\
            P%(3) = 8*(LEN(A$)-1)*CODE% :\
            GOSUB REC_1
            P%(0) = P%(0) + P%(2) :\
            P%(2) = 0 :\
```

```
            P%(3) = CODE% + 256 :\
            IF 8*CODE%/SIZE% < 9 THEN RETURN
PRN_20      DPOKE POINTR, PRNTR :\
            X1% = (P%(0)-X_LOW%)/SIZE% + 1 :\
            Y1% = (P%(1)-Y_LOW%)/SIZE% + 1 :\
            IF X1%  <  0 \
               OR Y1%  <  0 \
               OR X1%  > 217 \
               OR Y1%  > 152  THEN  RETURN
                DPOKE X0, X1% :\
                DPOKE Y0, Y1%
PRN_30      DPOKE CODE, P%(3) :\
            DPOKE CHPTR, PTR(A$) :\
            DUM% = USR(0) :\
            PRT_SW% = 1 :\
            RETURN
*
NAMR        A$ = ''
            CR_SW% = 0 :\
            I% = 1
NAMR_12     B$ = INCH$(0)
            IF B$ = CHR$(13) GOTO NAMR_40
            IF B$ <> CHR$(8)  GOTO NAMR_15
                 IF LEN(A$) <= 1  GOTO NAMR
                 A$ = LEFT$(A$,LEN(A$)-1)
                 PRINT  ' ';CHR$(8);
                 I% = I% - 1
                 GOTO NAMR_12
NAMR_15     IF B$ > CHR$(31) AND B$ < CHR$(127) GOTO NAMR_20
                 PRINT CHR$(10);CHR$(10);
                 PRINT 'PRINTABLE CHARACTERS ONLY  ';
                 PRINT CHR$(11);CHR$(11);
                 FOR J% = 1 TO 28
                      PRINT CHR$(8);
                 NEXT J%
                 PRINT ' ';CHR$(8);
                 GOTO NAMR_12
NAMR_20      A$ = A$ + B$
            I% = I% + 1
            IF I% <= LENGTH% GOTO NAMR_12
            B$ = INCH$(0)
            IF B$ = CHR$(13) GOTO NAMR_40
            IF B$ <> CHR$(8) GOTO NAMR_30
                 PRINT ' '; CHR$(8);
                 I% = I% - 1
                 GOTO NAMR_12
NAMR_30     PRINT CHR$(10); CHR$(7);
            PRINT 'ONLY'; LENGTH%; '  CHARACTERS ARE ALLOWED '
            A$ = LEFT$(A$,LENGTH%)
            PRINT 'THE TEXT IS '; A$
            PRINT 'DO YOU WANT TO START OVER(Y-N)?';
```

```
            B$ = INCH$(0)
            PRINT
            IF B$ <> 'Y' AND B$ <> 'y' THEN RETURN
                 GOTO NAMR
NAMR_40     IF LEN(A$) < 1 THEN CR_SW% = 1
            PRINT :\
            RETURN
*
NUMBR       C$ = ''
            CR_SW% = 0
NUMB_90     B$ = INCH$(0)
            IF B$ = '+' OR B$ ='-' OR B$ = '.'  \
              OR B$ > CHR$(47) AND B$ < CHR$(58) GOTO NUMB_100
                 IF B$ =CHR$(13) GOTO NUMB_200
                 IF B$ = CHR$(08) THEN PRINT CHR$(12);CHR$(12);
                 IF B$ = CHR$(11) THEN PRINT CHR$(10);
                 IF B$ = CHR$(10) THEN PRINT CHR$(11);
                      PRINT CHR$(8); ' '; CHR$(8)
                      GOTO NUMB_90
NUMB_100    C$ = C$ + B$
NUMB_110     B$ = INCH$(0)
            IF B$ = CHR$(13) GOTO NUMB_150
            IF B$ > CHR$(47) AND B$ < CHR$(58) GOTO NUMB_120
            IF B$ = "."   GOTO NUMB_120
                 IF B$ = CHR$(8) GOTO NUMB_115
                      IF B$ < CHR$(32) GOTO NUMB_112
                      PRINT CHR$(8); ' '; CHR$(8);
NUMB_112              IF LEN(C$) = 1 GOTO NUMBR
                      C$ = LEFT$(C$,LEN(C$)-1)
                      GOTO NUMB_110
NUMB_115         PRINT ' ';CHR$(8);
                 IF LEN(C$) = 1 GOTO NUMBR
                 C$ = LEFT$(C$,LEN(C$)-1)
                 GOTO NUMB_110
NUMB_120    C$ = C$ + B$
            GOTO NUMB_110
NUMB_150    A      = VAL(C$)
            RETURN
NUMB_200    CR_SW% = 1
            RETURN
```

Menu

```
$SCALE  6
            DIM P%(3), ELEM$(13), P$(3,13), MP$(3,13):\
            DIM NAM$(13), REC$(13), EL_PTR$(13), NO_EL$(13)
            DIM    ROT_CN$(13), M_ELEM$(13):\
            SIZ = HEX("9750") :\
            POINTR = DPEEK(52267) - 2 :\
```

```
          CLEAR = HEX("9C7D") :\
          GRAPH = HEX("9AE5")
          PLOT = HEX("9C60") :\
          ERASE = HEX("9758") :\
          X_LOC = HEX("9700") :\
          Y_LOC = HEX("9702") :\
          G_X1 = HEX("970C")
          G_X2 = HEX("9710") :\
          G_Y1 = HEX("970E") :\
          G_Y2 = HEX("9712") :\
          X1 = HEX("9704") :\
          Y1 = HEX("9706") :\
          X2 = HEX("9708") :\
          Y2 = HEX("970A")
          X_LO = HEX("9746") :\
          Y_LO = HEX("9748") :\
          X_HI = HEX("974A") :\
          Y_HI = HEX("974C") :\
          XCUR = HEX("E008") :\
          YCUR = HEX("E00A") :\
          BUTTON = HEX("E009")
          WINDO = HEX("9E0D") :\
          CODE = HEX("9773") :\
          CIR   = HEX("A210") :\
          Q_CIR = HEX("A37B")
          H_CIR = HEX("A3A8") :\
          RADIUS = HEX("976D") :\
          PRNTR = HEX("A3EB") :\
          LI    = HEX("977D")
          LII   = HEX("977F") :\
          CHPTR = HEX("9781") :\
          X0 = HEX("9784") :\
          Y0 = HEX("9786") :\
          CRT_X = HEX("976F") :\
          CRT_Y = HEX("9771")
          WIPE = HEX("A72C") :\
          CRT_X% = 255 :\
          CRT_Y% = 191 :\
          WIN_SW = HEX("9753")
          PRINT CHR$(26); 'MENU START'
*
          ON ERROR GOTO ER_10
          GOTO START_10
ER_10     IF ERR <> 4 GOTO ER_20
          PRINT \
           'THERE IS NO MACRO ELEMENT FILE ON THIS DISKETTE'
          PRINT 'EXIT MENU ABEND'
          CHAIN '0.MADS-3'
ER_20     IF ERR <> 16 THEN ON ERROR GOTO
          PRINT \
          'INSERT A DRAWING DISKETTE IN THE RIGHT-HAND DRIVE'
```

```
            PRINT 'AND PRESS ENTER'
            B$ = INCH$(0)
            RESUME START_20
*
START_10    OPEN OLD 'MAC_ELEM' AS 2 :\
            FOR I% = 0 TO 13 :\
                 FIELD #2, 18*I% AS Z$,\
                      10 AS M_ELEM$(I%),\
                       2 AS MP$(0,I%),\
                       2 AS MP$(1,I%),\
                       2 AS MP$(2,I%),\
                       2 AS MP$(3,I%) :\
            NEXT I%
*
            OPEN OLD 'MAC_DIR' AS 3 :\
            FOR I% = 0 TO 13 :\
                 FIELD #3, 18*I% AS Z$,\
                      10 AS  NAM$(I%),\
                      2 AS    REC$(I%),\
                      2 AS EL_PTR$(I%),\
                      2 AS  NO_EL$(I%),\
                      2 AS ROT_CN$(I%) :\
            NEXT I%
START_20    GET #3, RECORD 1 :\
            LAST_M_DIR_REC% = CVT$%(REC$(0))
            LAST_M_ELEM_PTR% = CVT$%(EL_PTR$(0))
INIT_0      SIZE% = 16  :\
            X_MAX% = 4080
            Y_MAX% = 3056
            CRT_X% = 255
            CRT_Y% = 191
            SIZE = SIZE% :\
            DPOKE SIZ, SIZE% :\
            DPOKE CRT_X, 255 :\
            DPOKE CRT_Y, 191 :\
            DPOKE POINTR,CLEAR :\
            DUM%=USR(0)
            DPOKE POINTR, GRAPH :\
            DPOKE X1, 255 :\
            DPOKE Y1, 0 :\
            DPOKE X2, 255 :\
            DPOKE Y2, 191 :\
            DUM% = USR(0)
            DPOKE Y1, 191 :\
            DPOKE X1, 0 :\
            DUM% = USR(0) :\
            DPOKE Y1, 161 :\
            FOR I% = 0 TO 7 :\
                 DPOKE X1, 32*I% :\
                 DPOKE X2, 32*I% :\
                 DUM% = USR(0) :\
```

```
            NEXT I%
            DPOKE X1, 225 :\
            DPOKE X2, 254 :\
            FOR I% = 0 TO 5 :\
                  DPOKE Y1, 32*I% :\
                  DPOKE Y2, 32*I% :\
                  DUM% = USR(0) :\
            NEXT I%
            DPOKE CODE, 0 :\
            DPOKE POINTR, PRNTR :\
            DPOKE Y0, 161 :\
            FOR I% = 0 TO 7 :\
                  DPOKE X0, (32*I%+1) :\
                  A$ = 'P' + MID$(STR$(I%),2,1) :\
                  DPOKE CHPTR, PTR(A$) :\
                  DUM% = USR(0) :\
            NEXT I%
            FOR I% = 8 TO 12 :\
                  DPOKE X0, 225 :\
                  DPOKE Y0, 161 - (I%-7)*32 :\
                  IF I% < 10 THEN A$ = 'P'+MID$(STR$(I%),2,1) \
                              ELSE A$='P1'+MID$(STR$(I%-10),2,1)
                  DPOKE CHPTR, PTR(A$) :\
                  DUM% = USR(0) :\
            NEXT I% :\
            X_LOW% = 0 :\
            Y_LOW% = 0 :\
            X_HIGH% = CRT_X% * SIZE% :\
            Y_HIGH% = CRT_Y%*SIZE%
INIT_100    DPOKE POINTR, WIPE :\
            DUM% = USR (0) :\
            DPOKE X_LO,X_LOW% :\
            DPOKE Y_LO,Y_LOW% :\
            DPOKE X_HI,X_HIGH% :\
            DPOKE Y_HI,Y_HIGH%
            P%(0) = X_LOW% :\
            P%(1) = Y_LOW% :\
            P%(2) = 3584  :\
            P%(3) = 2560  :\
            GOSUB REC_1
            DPOKE CRT_X, CRT_X% :\
            DPOKE CRT_Y, CRT_Y%

            NO_MAC% = 0
            FOR M_DIR_REC% = 1 TO LAST_M_DIR_REC%
             GET #3, RECORD M_DIR_REC%
             IF M_DIR_REC% <> LAST_M_DIR_REC%  THEN K% = 13 \
                                    ELSE K% = LAST_M_ELEM_PTR%
             FOR M_DIR_PTR% = 0 TO K%
                   IF NAM$(M_DIR_PTR%) = '0000000000' GOTO DONE
                   IF NAM$(M_DIR_PTR%) = 'ZZZZZZZZZZ' \
```

```
                                        GOTO INIT_500
                M_ELEM_REC% = CVT$%(REC$(M_DIR_PTR%))
                M_ELEM_PTR% = CVT$%(EL_PTR$(M_DIR_PTR%))
                NO_ELEM% = CVT$%(NO_EL$(M_DIR_PTR%))
                IF CVT$%(ROT_CN$(M_DIR_PTR%)) <> 0 \
                                        GOTO INIT_180
                     M_ELEM_PTR% = M_ELEM_PTR% + 1
                     IF M_ELEM_PTR% <= 13  GOTO  INIT_170
                          M_ELEM_PTR% = 0
                          M_ELEM_REC% = M_ELEM_REC% + 1
INIT_170             NO_ELEM% = NO_ELEM% - 1
INIT_180        GET #2, RECORD  M_ELEM_REC%
                X_0% = 3840
                Y_0% = 2816
                IF NO_MAC% < 8      \
                  THEN  X_0% = (16 + 32*NO_MAC%)*SIZE% \
                  ELSE Y_0% = (144 - 32*(NO_MAC% - 8))*SIZE%
                CTR% = 1
INIT_200        A$ = M_ELEM$(M_ELEM_PTR%)
                FOR J% = 0 TO 3
                     P%(J%) = CVT$%(MP$(J%,M_ELEM_PTR%))
                NEXT J%
                P%(0) = P%(0)*2 + X_0%
                P%(1) = P%(1)*2 + Y_0%
                IF A$ <> 'LINE'  GOTO  INIT_240
                     P%(2) = P%(2)*2 + X_0%
                     P%(3) = P%(3)*2 + Y_0%
                     GOTO INIT_300
INIT_240        IF A$ <> 'RECT' GOTO INIT_250
                     P%(2) = P%(2)*2
                     P%(3) = P%(3)*2
                     GOTO INIT_300
INIT_250        P%(2) = P%(2)*2
INIT_300        GOSUB FILE_OUT
                M_ELEM_PTR% = M_ELEM_PTR% + 1
                IF M_ELEM_PTR% <= 13  GOTO  INIT_400
                IF CTR% = NO_ELEM%  GOTO  INIT_400
                     M_ELEM_PTR% = 0
                     M_ELEM_REC% = M_ELEM_REC% + 1
                     GET #2, RECORD M_ELEM_REC%
INIT_400        CTR% = CTR% + 1
                IF CTR% <= NO_ELEM%  GOTO  INIT_200
                NO_MAC% = NO_MAC% + 1
                IF NO_MAC% >= 13 GOTO DONE
INIT_500    NEXT M_DIR_PTR%
           NEXT M_DIR_REC%
DONE       PRINT'EXIT MENU NORMAL'
           CHAIN "0.MADS-3"
*
PRNTER     CODE% = P%(3) :\
           IF CODE% > 16 GOTO PRN_10
```

```
                P%(2) = (LEN(A$)-1)*8*CODE% :\
                P%(3) = 8*CODE% :\
                GOSUB REC_1
                P%(2) = 0 :\
                P%(3) = CODE% :\
                IF 8*CODE%/SIZE% < 7 THEN RETURN
                GOTO PRN_20
PRN_10      CODE% = CODE% - 256 :\
            P%(2) = 8*CODE% :\
            P%(0) = P%(0) - P%(2) :\
            P%(3) = 8*(LEN(A$)-1)*CODE% :\
            GOSUB REC_1
            P%(0) = P%(0) + P%(2) :\
            P%(2) = 0 :\
            P%(3) = CODE% + 256 :\
            IF 8*CODE%/SIZE% < 7 THEN RETURN
PRN_20      DPOKE POINTR, PRNTR :\
            X1% = (P%(0)-X_LOW%)/SIZE% :\
            Y1% = (P%(1)-Y_LOW%)/SIZE% :\
            IF X1%  <  0 \
               OR Y1%  <  0 \
               OR X1%  > 218 \
               OR Y1%  > 153  THEN  RETURN
                DPOKE X0, X1% :\
                DPOKE Y0, Y1%
PRN_30      DPOKE CODE, P%(3) :\
            DPOKE CHPTR, PTR(A$) :\
            DUM% = USR(0) :\
            PRT_SW% = 1 :\
            RETURN
*
LIN_1       X1%=P%(0) :\
            Y1%=P%(1) :\
            X2%=P%(2) :\
            Y2%=P%(3) :\
            GOSUB WIN_CON
            RETURN
*
REC_1       IF P%(3) <= SIZE% THEN REC_5
                IF P%(2) <= SIZE% THEN REC_2
                     BASE%= P%(2) :\
                     HEIGHT%= P%(3) :\
                     X1%=P%(0) :\
                     Y1%=P%(1) :\
                     X2%=X1% + BASE% :\
                     Y2%=Y1%
REC_1_4              GOSUB WIN_CON
                     X1%=X2% :\
                     Y2%=Y1% + HEIGHT%
REC_1_6              GOSUB WIN_CON
                     Y1%=Y2% :\
```

```
                    X2%=X1% - BASE%
REC_1_7             GOSUB WIN_CON
                    X1%=X2% :\
                    Y1%=Y2% :\
                    Y2%=Y1% - HEIGHT%
REC_1_8             GOSUB WIN_CON
                    RETURN
REC_2          IF P%(3) <= SIZE% THEN REC_4
                    X1%=P%(0) :\
                    Y1%=P%(1) :\
                    X2% = P%(0) :\
                    Y2% = P%(1) + HEIGHT
REC_3               GOSUB WIN_CON
                    RETURN
REC_4          X1%=P%(0) :\
               Y1%=P%(1) :\
               X2%=P%(0) :\
               Y2%=P%(1) :\
               GOSUB WIN_CON
               RETURN
REC_5       IF P%(2) <= SIZE%  GOTO REC_4
            X1%=P%(0) :\
            Y1%=P%(1) :\
            X2%=P%(0) + BASE :\
            Y2%=P%(1) :\
            GOSUB WIN_CON
            RETURN
*
WIN_CON     IF X1% > X_MAX%  OR  X1% < 0  GOTO BND_ERR
            IF X2% > X_MAX%  OR  X2% < 0  GOTO BND_ERR
            IF Y1% > Y_MAX%  OR  Y1% < 0  GOTO BND_ERR
            IF Y2% > Y_MAX%  OR  Y2% < 0  GOTO BND_ERR
            DPOKE G_X1, X1% :\
            DPOKE G_Y1, Y1% :\
            DPOKE G_X2, X2% :\
            DPOKE G_Y2, Y2% :\
            DPOKE POINTR,WINDO :\
            DUM% = USR(0) :\
            RETURN
BND_ERR     PRINT 'Cannot draw this line--out of bounds.' :\
            PRINT'X1=';X1%;'Y1=';Y1%;'X2=';X2%;'Y2=';Y2%:\
            BND_SW% = 1 :\
            RETURN
*
AR_1        X1% = P%(0) :\
            Y1% = P%(1) :\
            RAD% = P%(2) :\
            CODE% = P%(3) :\
            S1% = ABS(X_HIGH% - X1%) :\
            S2% = ABS(X_LOW% - X1%)
            S3% = ABS(Y_HIGH% - Y1%) :\
```

```
          S4% = ABS(Y_LOW% - Y1%) :\
          IF X1% < X_LOW%  GOTO AR_16
               IF X1% > X_HIGH%  GOTO AR_10
                    IF Y1% < Y_LOW%  GOTO AR_8
                         IF Y1% > Y_HIGH%  GOTO AR_6
AR_4                          IF   RAD% > SQR(S1%^2+S3%^2)\
                              AND  RAD% > SQR(S1%^2+S4%^2)\
                              AND  RAD% > SQR(S2%^2+S3%^2)\
                              AND  RAD% > SQR(S2%^2+S4%^2)\
                                        THEN AR_120 \
                                        ELSE  AR_98
AR_6                     IF RAD% < S3% THEN AR_120
                         IF   RAD% > S3% + 230*SIZE%   \
                         OR CODE% = 0  THEN  RETURN
                              GOTO AR_98
AR_8                IF  RAD% < S4%  THEN AR_120
                    IF  RAD% > S3% + 230*SIZE%  THEN AR_120
                    IF CODE% = 2 GOTO AR_120
                         GOTO AR_98
AR_10          IF Y1% > Y_HIGH%  GOTO AR_14
                    IF Y1% < Y_LOW%  GOTO AR_12
                         IF RAD% <S1% THEN AR_120
                         IF RAD% > S1%+273*SIZE% THEN AR_120
                          IF CODE% = 3 GOTO AR_120
                              GOTO AR_98
AR_12               SCRATCH%= SQR(S1%^2 + S4%^2):\
                    IF RAD% < SCRATCH%            THEN AR_120
                    IF RAD% > SCRATCH%+320*SIZE% THEN AR_120
                     IF CODE% >= 2  GOTO AR_120
                         GOTO AR_98
AR_14                SCRATCH%= SQR(S1%^2 + S3%^2):\
                     IF RAD% < SCRATCH%              THEN  AR_120
                     IF RAD% > SCRATCH% + SIZE%*320 THEN AR_120
                     IF CODE% = 3  GOTO AR_120
                         GOTO AR_98
AR_16          IF Y1% >Y_HIGH%  GOTO AR_20
                    IF Y1% < Y_LOW%  GOTO AR_18
                         IF RAD% < S2% THEN AR_120
                         IF  RAD% > S2% + 273*SIZE%  THEN AR_120
                          IF  CODE% = 1  GOTO AR_120
                              GOTO AR_98
AR_18               SCRATCH%= SQR(S2%^2 + S4%^2):\
                    IF RAD% < SCRATCH%            THEN AR_120
                    IF RAD% > SCRATCH% + 320*SIZE% THEN AR_120
                     IF CODE% = 1 OR CODE% =2  GOTO AR_120
                         GOTO AR_98
AR_20          SCRATCH%= SQR(S2%^2 + S3%^2):\
               IF RAD% < SCRATCH%            THEN AR_120
               IF RAD% > SCRATCH% + 320*SIZE% THEN AR_120
                IF  CODE% =1  GOTO AR_120
AR_98          DPOKE RADIUS, RAD% :\
```

```
            DPOKE X1, X1% :\
            DPOKE Y1, Y1% :\
            POKE WIN_SW, 1 :\
            IF A$ <> 'CIRCLE'  AND A$ <> 'N6' GOTO AR_110
                 DPOKE POINTR, CIR :\
                 DUM% = USR(0):\
                 RETURN
AR_110      IF A$ = 'H_CIR'  GOTO AR_115
                 DPOKE POINTR, Q_CIR :\
                 DPOKE CODE, P%(3) :\
                 DUM% = USR(0):\
                 RETURN
AR_115      DPOKE POINTR, H_CIR:\
            DPOKE CODE, P%(3):\
            DUM% = USR(0):\
            RETURN
AR_120      POKE WIN_SW, 0:\
            RETURN
*
FILE_OUT    IF  A$ = 'LINE'  GOTO L_1
            IF  A$ = "RECT"  GOTO R_1
            IF  A$ = "CIRCLE" GOTO A_1
            IF  A$ = 'H_CIR'  GOTO A_1
            IF  A$ = "Q_CIR"  GOTO A_1
            IF  A$ = 'N6'     GOTO A_1
            IF LEFT$(A$,1) = 'P' GOTO P_1
            RETURN
*
L_1         GOSUB  LIN_1
            PRT_SW% = 0 :\
            RETURN
*
R_1         GOSUB  REC_1
            PRT_SW% = 0 :\
            RETURN
*
A_1         GOSUB  AR_1
            PRT_SW% = 0 :\
            RETURN
*
P_1         GOSUB  PRNTER
            RETURN
```

Programs in Assembler

```
*****************************************
        OPT       PAG
        TTL       MAIN
        PAG
```

```
RLIST   REG         D,X,U,Y,DP,CC
MEMEND  EQU         $96FF
BASE    EQU         $A800
FLEX    EQU         $CD03
DIRECT  EQU         $9700
DIR    EQU         $97
PSTRING EQU       $CD1E
         SETDP      DIR
*
*MACROS*************************************
*
NEGD     MACRO
         COMA
         COMB
         ADDB       #1
         ADCA       #0
         ENDM
*
ASLD      MACRO
          ASLB
          ROLA
          ENDM
*
*****************************************************
*    macros to drive the math pack subroutines
*
*                    ZERO U,BYTES
ZERO    MACRO
         LDX  #&1
         LDD  #&2
         JSR  LOAD
         ENDM
*
*                    TRNSFR FROM,TO,BYTES
TRNSFR   MACRO
         LDX  #&1
         LDY  #&2
         LDB  #&3
         JSR  TRANS
         ENDM
*
*                    SHFT_R U,BYTES
SHFT_R   MACRO
         LDX  #&1
         LDB  #&2
         JSR  R_SHFT
         ENDM
*
*                    SHFT_L U,BYTES
SHFT_L   MACRO
         LDX  #&1
```

```
          LDB   #&2
          JSR   L_SHFT
          ENDM
*
*                    SUBT U,V,BYTES     V-U => V
SUBT      MACRO
          LDY   #&1
          LDX   #&2
          LDB   #&3
          JSR   L_SUBT
          ENDM
*
*                    ADD U,V,BYTES    U+V => V
ADD       MACRO
          LDY   #&1
          LDX   #&2
          LDB   #&3
          JSR L_ADD
          ENDM
*
NEGD      MACRO
          COMA
          COMB
          ADDB        #1
          ADCA        #0
          ENDM
*
*
*
           PAG
           STTL
*
*         Variables On The Direct Page
          ORG         DIRECT
X_LOCAL   FDB         0
Y_LOCAL   FDB         0
X1        FDB         0
Y1        FDB         0
X2        FDB         0
Y2        FDB         0
G_X1      FDB         0
G_Y1      FDB         0
G_X2      FDB         0
G_Y2      FDB         0
DEL_X     FDB         0
DEL_Y     FDB         0
DIVND     FDB         0
DIVSR     FDB         0
BYTE1     FDB         0
BYTE2     FDB         0
A_0        FDB 0000
```

```
A_2       FDB 0000
B_0       FDB 0000
B_2       FDB 0000
C_0       FDB 0000
C_2       FDB 0000
C_4       FDB 0000
C_6       FDB 0000
SCR_0     FDB  0000
SCR_2     FDB  0000
SCR_4     FDB  0000
SCR_6     FDB  0000
SCR_8     FDB  0000
SCR_10    FDB  0000
SCR_12    FDB  0000
SCR_14    FDB  0000
SCR_16    FDB  0000
SLOPE      RMB  4
X_LO       FDB  0
Y_LO       FDB  0
X_HI       FDB  0
Y_HI       FDB  0
YTEST      FDB  0
SIZE       FDB  0
SWP_SW     FCB  0
WIN_SW     FCB  0
EXPO      FCB 00
N_BYTE    FCB 00
SWITCH   FCB        0
ERR_SW   FCB        0
ERASE    FCB        0
TEMP     FCB        0
QUOT     FCB        0
TEST     FCB        0
DR_SW      FCB  0
SIGN       FCB  0
SLP_SI     FCB  0
P_IN       FDB  0
P_OUT      FDB  0
DONE       FDB  0
R_SQ       RMB  8
RADIUS     FDB  0
CRT_X      FDB  255
CRT_Y      FDB  191
CODE       FDB  0
ADDR_T     FDB  QUAD_1,QUAD_2,QUAD_3,QUAD_4
L1         FDB  0
L2         FDB  0
CHPTR      FDB  0
EX_SW      FCB  0
X0         FDB  0
Y0         FDB  0
```

```
         STTL        PACK- Math Package
          PAG
*
*
*MATH PACK*****************************************
*This routine does A = C/B where
* C is 96 bits and A and B are 48.
* division overflow (large number
* divided by a small one) is
* flagged by returning A filled
* with 1's. e.g. divide by 0.
* USES EXPO
*       C_0
*       A_0
*       B_0
* CALLS L_SHFT
*       R_SHFT
*       L_SUBT
*
*
            ORG DIRECT+256
L_DIV       PSHS      #RLIST
*                           initialize the exponent
            CLRA
            STA EXPO
*       adjust the dividend to the left
*       so that the high order byte is
*       of the form; 01xxxxxx
            LDU  #0
L_D_2       TST C_0
            BMI L_D_4
            LDX #C_0
            LDB #8
            JSR  L_SHFT
            INC EXPO
            LEAU 1,U
            CMPU #64
            BNE  L_D_2
*                           dividend = 0; clear A
            CLRA
            CLRB
            STD  A_0
            STD  A_2
            LBRA L_D_30
*
L_D_4       LDX #C_0
            LDB #8
            JSR  R_SHFT
            DEC EXPO
*               adjust the divisor
            LDU  #0
```

```
L_D_5       TST B_0
            BMI L_D_6
            LDX #B_0
            LDB #4
            JSR  L_SHFT
            DEC EXPO
            LEAU 1,U
            CMPU #32
            BNE  L_D_5
*               divide by 0; fill A with 1's
            BRA L_D_26
*
L_D_6       LDX #B_0
            LDB #4
            BSR R_SHFT
*               clear A
            CLRA
            CLRB
            STD  A_0
            STD  A_2
            LDU  #0
*               left shift A
L_D_8       LDX #A_0
            LDB #4
            BSR L_SHFT
*               start of the compare loop
            LDX #C_0
            LDY #B_0
L_D_10      LDD ,X++
            CMPD ,Y++
            BHI L_D_18
            BLO L_D_22
            CMPX #C_0+3
            BLS L_D_10
*               end of the compare loop
*               clear the MS six bytes of C
*               (C0 - C5 = B)
            CLRA
            CLRB
            LDX #C_0
L_D_16      STD ,X++
            CMPX #C_0+3
            BLS L_D_16
            BRA L_D_20
*               subtract B from C
L_D_18      LDX #C_0
            LDY #B_0
            LDB #4
            BSR L_SUBT
*               increment A
L_D_20      INC A_0+3
```

```
*              left shift C
L_D_22     LDX #C_0
           LDB #8
           BSR L_SHFT
           LEAU 1,U
           CMPU #32
           BLO L_D_8
*              end of main routine
*              adjust quotient to the right
L_D_24     TST  EXPO
           BMI  L_D_26
           BEQ  L_D_30
           LDX  #A_0
           LDB  #4
           BSR  R_SHFT
           DEC  EXPO
           BRA  L_D_24
*
*              fill quotient with 1's
L_D_26     LDD  #$FFFF
           STD  A_0
           STD  A_2
L_D_30     PULS #RLIST
           RTS
*
*              right shift
*                   (X) => high order byte
*                   (B) => # bytes
*                    uses N_BYTE
R_SHFT     STB N_BYTE
           CLC
R_S_2      ROR  ,X+
           DEC N_BYTE
           TST
            N_BYTE
           BNE R_S_2
           RTS
*
*              left shift
*                   (X) => high order byte
*                   (B) => # bytes
*                   no variables
L_SHFT     DECB
           BMI L_S_4
           ASL B,X
L_S_2      DECB
           BMI L_S_4
           ROL B,X
           BRA L_S_2
L_S_4      RTS
*
*              long subtract
```

```
*                   B is subtracted from C
*                   (X) => C
*                   (Y) => B
*                   no variables
L_SUBT      CLC
L_SU_2      DECB
            BMI L_SU_4
            LDA B,X
            SBCA B,Y
            STA B,X
            BRA L_SU_2
L_SU_4      RTS
*               This routine does A = sqrt(C)
*                   load C and JSR SQ_RT
*                    calls TRANS
*                          L_DIV
*                          L_ADD
*                          R_SHFT
*                    uses C_0
*                          A_0
*                          SCR_12
*                          SCR_0
*
SQ_RT       PSHS #RLIST
*               move C to scratch
            LDX  #C_0
            LDY  #SCR_0
            LDB  #8
            BSR  TRANS
*               first guess
            LDX  #C_0
            TFR  X,U
            LDY  #A_0
            LDB  #4
*               test C_0
            TST  ,U+
            BNE  SQ_15
*               test C_0+1
            TFR  U,X
            TST  ,U+
            BNE  SQ_15
*               test C_0+2
            TST  ,U+
            BNE  SQ_15
*               test C_0+3
            LEAX 1,X
            TST  ,U+
            BNE  SQ_15
*               test C_0+4
            TST  ,U+
            BNE  SQ_15
```

```
*               test C_0+5
              LEAX 1,X
              TST  ,U+
              BNE  SQ_15
*               test C_0+6
              LEAX 1,X
SQ_15         BSR TRANS
*               iterative loop
*               A => scratch
SQ_20         LDX  #A_0
              LDY  #SCR_12
              LDB  #4
              BSR TRANS
*               get the original number
*               put it in  C
              LDX  #SCR_0
              LDY  #C_0
              LDB  #8
              BSR TRANS
*               get ready to divide
*               by the present guess
              LDX  #SCR_12
              LDY  #B_0
              LDB  #4
              BSR TRANS
*               divide
              JSR  L_DIV
*               check for overflow
              TST  A_0
              BMI  SQ_30
*               add present guess
              LDX  #A_0
              LDY  #SCR_12
              LDB  #4
              BSR L_ADD
*               divide by 2
              LDB  #4
              JSR  R_SHFT
*               new guess in A
*               check for convergence
              LDX  #SCR_12
              LDY  #A_0
SQ_25         LDA  ,X+
              CMPA ,Y+
              BNE  SQ_20
              CMPX #SCR_12+2
              BNE  SQ_25
*               we have a value
SQ_30         PULS #RLIST
              RTS
*
```

```
*              This routine transfers multibyte
*              variables - least significant
*              byte first.
*                   (X) => from (MSB)
*                   (Y) => to (MSB)
*                    no variables
TRANS     DECB
          BMI   TR_RT
          LDA   B,X
          STA   B,Y
          BRA   TRANS
TR_RT     RTS
*
*              This routine adds multi-byte
*              variables.
*                   (X) => accumulator
*                   (Y) => operand
*                    no variables
L_ADD     CLC
L_A_2     DECB
          BMI   L_A_4
          LDA   B,X
          ADCA  B,Y
          STA   B,X
          BRA   L_A_2
L_A_4     RTS
*    This routine does C = A*B,
*    where A and B are 32 bits each
*        and C is 64 bits.
*    A and B are not altered.
*    calls MUL_CY
*    uses  TEMP
*
*
L_MULT    PSHS #RLIST
          CLRA
          STA  TEMP
*         clear C_0
          LDX  #0
          CLRA
          CLRB
L_M_2     STD  C_0,X
          LEAX 2,X
          CMPX #8
          BNE  L_M_2
*         initialize
          LDX  #0
          LDY  #0
          LDU  #C_0
*         do it
*              form the partial product
```

```
L_M_4       LDA   A_0,X
            LDB   B_0,Y
            MUL
            ADDD        ,U
            STD         ,U
*               deal with carry out
            BCC   L_M_6
            BSR   MUL_CY
L_M_6       LEAY  1,Y
            LEAU  1,U
            CMPY  #4
            BNE   L_M_4
*               end of inner loop
            LDY   #0
            LEAU  -3,U
            LEAX  1,X
            CMPX  #4
            BNE   L_M_4
            PULS  #RLIST
            TST   TEMP
            BEQ   L_M_8
            SEC
L_M_8       RTS
*           recursivc carry routine
MUL_CY      PSHS  U
            LEAU  -1,U
            CMPU  #C_0
            BHS   M_CY_2
            LDA   #$FF
            STA   TEMP
            BRA   M_CY_R
M_CY_2      LDA   #1
            ADDA  ,U
            STA   ,U
            BCC   M_CY_R
            BSR   MUL_CY
M_CY_R      PULS  U
            RTS
*
*     load the contents of A into successive bytes
*
LOAD        STA   ,X+
            DECB
            BNE   LOAD
            RTS
*
*     ytest = (xp - x1)*slope + y1
*
CALC_Y      PSHS  D
            CLR   SIGN
            ZERO  A_0,16
```

```
            PULS D
            SUBD G_X1
            BEQ  C_Y_6
            BPL  C_Y_2
            COM  SIGN
            NEGD
C_Y_2       STD  A_0
            TRNSFR SLOPE,B_0,4
            JSR  L_MULT
            LDA  SIGN
            EORA SLP_SI
            BMI  C_Y_10
            LDD  C_0
            BNE  C_Y_8
            LDD  C_2
            BMI  C_Y_8
C_Y_6       ADDD G_Y1
            BVS  C_Y_8
            STD  YTEST
            RTS
C_Y_8       LDD  #$7FFF
            STD  YTEST
            RTS
C_Y_10      LDD  C_0
            BNE  C_Y_14
            LDD  C_2
            BMI  C_Y_14
            NEGD
            ADDD G_Y1
            STD  YTEST
            RTS
C_Y_14      LDD  #$8001
            STD  YTEST
            RTS
*
*     x = (yp - y1)/slope + x1
*
CALC_X      PSHS D
            CLR  SIGN
            ZERO A_0,16
            PULS D
            SUBD G_Y1
            BPL  C_X_2
            COM  SIGN
            NEGD
C_X_2       STD  C_2
            TRNSFR SLOPE,B_0,4
            JSR  L_DIV
            LDD  A_0
            BMI  C_X_10
            PSHS D
```

```
            LDA   SIGN
            EORA  SLP_SI
            BMI   C_X_8
            PULS  D
            ADDD  G_X1
            RTS
*(yp-y1)/slope negative and smaller than 32767
C_X_8       PULS  D
            NEGD
            ADDD  G_X1
            BPL   C_X_9
            LDD   X_LO
C_X_9       RTS
*|(yp-y1)/slope| > 32767
C_X_10      LDA   SIGN
            EORA  SLP_SI
            BMI   C_X_12
            LDD   #$7FFF
            RTS
C_X_12      LDD   X_LO
            RTS
*
*
SCALE       ZERO  C_0,8
            PULU  X
            PULU  D
            NEGD
            LEAX  D,X
            STX   C_2
            ZERO  B_0,4
            LDD   SIZE
            STD   B_0
            JSR   L_DIV
            LDD   A_0
            TST   A_2
            BPL   SCAL_2
            ADDD  #1
            STD   A_0
SCAL_2      RTS
*
*
           STTL       GRAPH-Plot points and lines.
           PAG
*
*
*GRAPH*********************************************
*
*
*
*          This routine will plot;
*          a point if x1=x2 and y1=y2
```

```
*         a straight line, x1<>x2 or y1<>y2
*         It calls;
*         DIVIDE
*         PLOT
*         HORIZ
*         VERT
*         It references;
*         X1,Y1,X2,Y2,TEST,ERASE,
*         DEL_X,DEL_Y,QUOT
*
*
GRAPH     PSHS      #RLIST
*
*         Set the direct page.
          LDA       #DIR
          TFR       A,DP
*
*         Initialize TEST (sign of slope)
*         and SWITCH (negative for 45 deg. slope).
          CLRA
          STA       TEST
          STA       SWITCH
          STA       ERR_SW
*
*         Form delta-x
          LDD       X2
          SUBD      X1
*
*         If delta-x is 0, go check
*         for a vertical line.
          LBEQ      V_TEST
          STD       DEL_X
*
*         Form delta-y
          LDD       Y2
          SUBD      Y1
*         If delta-y is 0, do a horizontal line.
          LBEQ      H_JMP
          STD       DEL_Y
*
*         Form ABS(DEL_Y), put on the stack.
          BPL       GRPH_1
          NEGD
GRPH_1    PSHS      D
*
*         Form ABS(DEL_X)
          LDD       DEL_X
          BPL       GRPH_2
          NEGD
GRPH_2    CMPD      ,S++
*
```

```
*         If |delta-x|>|delta-y|, do increment x.
*         Enter at SLP_1 if slope is one.
          LBEQ      SLP_1
          LBGT      INCR_X
INCR_Y    LDD       DEL_Y
          BPL       INY_1
*
*         Delta-y must be > 0.
          NEGD
          STD       DEL_Y
          LDD       Y1
          LDX       Y2
          STD       Y2
          STX       Y1
          LDD       X1
          LDX       X2
          STD       X2
          STX       X1
          LDD       DEL_X
          NEGD
          STD       DEL_X
*
*         If delta-x is < 0, set test switch
*         and make it positive.
INY_1     LDD       DEL_X
          BPL       INY_2
          COM       TEST
          NEGD
*
*         Get ready to divide.
INY_2     STD       DIVND
          LDD       DEL_Y
          STD       DIVSR
          JSR       DIVIDE
*
*         Plot the first point.
          LDY       X1
          STY       X_LOCAL
          LDY       Y1
          STY       Y_LOCAL
          JSR       PLOT
          TST       ERR_SW
          LBMI      RETURN
*
*         Initialize the counter.
          LDY       #0
          LEAY      1,Y
*
*         Draw the line.
Y_LOOP    TFR       Y,D
          LDA       QUOT
```

```
          MUL
          ADDD      #$0080
          TFR       A,B
          CLRA
          TST       TEST
          BPL       X_CAL
          NEGD
X_CAL     ADDD      X1
          STD       X_LOCAL
          TFR       Y,D
          ADDD      Y1
          STD       Y_LOCAL
          JSR       PLOT
          TST       ERR_SW
          LBMI      RETURN
          LEAY      1,Y
          CMPY      DEL_Y
          BLE       Y_LOOP
*
*         The line is done.
          LBRA       RETURN
*
*         This is a connection
*         to the horizontal routine.
H_JMP     JSR       HORIZ
          LBRA       RETURN
*
*         Increment x.
SLP_1     COM       SWITCH
INCR_X    LDD       DEL_X
          BPL       INX_1
*
*         Delta-x must be > 0.
          NEGD
          STD       DEL_X
          LDD       Y1
          LDX       Y2
          STD       Y2
          STX       Y1
          LDX       X2
          LDD       X1
          STD       X2
          STX       X1
          LDD       DEL_Y
          NEGD
          STD       DEL_Y
*
*         If delta-y is < 0  set test switch
*         and make it positive.
INX_1     LDD       DEL_Y
          BPL       INX_2
```

```
          COM       TEST
          NEGD
INX_2     TST       SWITCH
          BMI       INX_3
*
*         Get ready to divide.
          STD       DIVND
          LDD       DEL_X
          STD       DIVSR
          JSR       DIVIDE
*
*         Plot the first point.
INX_3     LDY       X1
          STY       X_LOCAL
          LDY       Y1
          STY       Y_LOCAL
          BSR       PLOT
          TST       ERR_SW
          BMI       RETURN
*
*         Initialize the counter.
          LDY       #1
*
*         Draw the line.
X_LOOP    TFR       Y,D
          TST       SWITCH
          BMI       INX_4
          LDA       QUOT
          MUL
          ADDD      #$0080
          TFR       A,B
          CLRA
INX_4     TST       TEST
          BPL       Y_CAL
          NEGD
Y_CAL     ADDD      Y1
          STD       Y_LOCAL
          TFR       Y,D
          ADDD      X1
          STD       X_LOCAL
          BSR       PLOT
          TST       ERR_SW
          BMI       RETURN
          LEAY      1,Y
          CMPY      DEL_X
          BLT       X_LOOP
*
*         Plot the last point.
          LDD       X2
          STD       X_LOCAL
          LDD       Y2
```

```
          STD       Y_LOCAL
          BSR       PLOT
*
*         The line is done.
RETURN    CLR       ERASE
          PULS      #RLIST
          RTS
*
*         See if this is a vertical line.
V_TEST    LDD       Y2
          SUBD      Y1
          BNE       V_JMP
*
*         It is a single point.
          LDD       Y1
          STD       Y_LOCAL
          LDD       X1
          STD       X_LOCAL
          BSR       PLOT
          BRA       RETURN
*
*         It is a vertical line.
V_JMP     JSR       VERT
          BRA       RETURN
          STTL      PLOT-Plot a point
          PAG
*
*
*         This routine will plot one point.
*         It calls;
*         FIND
*
PLOT      PSHS      #RLIST              ;Store the registers
*
          LDA      #DIR
          TFR       A,DP
          JSR       FIND
          TST       ERR_SW
          BMI       PLOT_3
          TST       ERASE
          BPL       PLOT_1
          COMA
          ANDA      ,X
          BRA       PLOT_2
PLOT_1    ORA       ,X
PLOT_2    STA       ,X
PLOT_3    PULS      #RLIST
          RTS
          STTL     CLEAR-Erase the graphic memory.
          PAG
*
```

```
*
*         This routine clears the graphic memory.
*
CLEAR   PSHS        D,X,Y,U,DP,CC
          LDX         #BASE
          CLRA
CLR_LOOP  STA         ,X+
          CMPX        #BASE+6144
          BNE         CLR_LOOP
          PULS        D,X,Y,U,DP,CC
          RTS
          STTL        HORIZ-Plot a horizontal line.
          PAG
*
*
*         Plot a horizontal line
*         Calls       FIND
*         PLOT
*         References;
*         X1,X2,Y1,Y2,ERASE,
*         BYTE1,BYTE2
*
HORIZ     PSHS        D,X,Y,U,DP,CC
          LDA        #DIR
          TFR         A,DP
*
*         If x1 > x2, swap them.
          LDD         X2
          CMPD        X1
          BGT         HOR_1
          LDX         X1
          STD         X1
          STX         X2
*
*         Is the line too short?
HOR_1     LDD         X2
          SUBD        X1
          CMPD        #8
          BLE         H_SHRT
*
*         Find the first byte.
          LDD         X1
          STD         X_LOCAL
          LDD         Y1
          STD         Y_LOCAL
          BSR         FIND
          TST         ERR_SW
          BMI         H_RET
*
*         X points to left byte,
*         A has partial content.
```

```
          NEGA
          COMA
          ASLA
          INCA
          TST       ERASE
          BPL       HOR_2
          COMA
          ANDA      ,X
          BRA       HOR_3
HOR_2     ORA       ,X
HOR_3     STA       ,X+
          STX       BYTE1
*
*         Find byte x2.
          LDD       X2
          STD       X_LOCAL
          BSR       FIND
          TST       ERR_SW
          BMI       H_RET
*
*         X points at right byte,
*         A contains partial content.
          NEGA
          TST       ERASE
          BPL       HOR_4
          COMA
          ANDA      ,X
          BRA       HOR_5
HOR_4     ORA       ,X
HOR_5     STA       ,X
*
*         Is there at least one byte between x1 & x2.
          CMPX      BYTE1
          BEQ       H_RET
*
*         At least one byte.
*         Fill in the middle.
          STX       BYTE2
          LDX       BYTE1
          CLRA
          TST       ERASE
          BMI       HOR_LOOP
          COMA
HOR_LOOP  STA       ,X+
          CMPX      BYTE2
          BNE       HOR_LOOP
          PULS      #RLIST
          RTS
*
*         Plot short lines here.
H_SHRT    LDD       Y1
```

```
          STD       Y_LOCAL
          LDD       X1
H_LO_2    STD       X_LOCAL
          JSR       PLOT
          TST       ERR_SW
          BMI       H_RET
          INCB
          CMPD      X2
          BLE       H_LO_2
H_RET     PULS      #RLIST
          RTS
          STTL      FIND-Find the address and partial content.
          PAG
*
*
*         This routine returns the address of the byte
*         corresponding to X_LOCAL,Y_LOCAL in X and
*         the partial content in A.
*         References
*         X_LOCAL,Y_LOCAL,BASE
*
FIND      LDD       X_LOCAL
          BMI       ERROR
          CMPD      #255
          BGT       ERROR
          CLRA
          LSRB
          LSRB
          LSRB
          PSHS      D
*
*         Multiply y by 32 to get the vertical row
*         of the object byte.
          LDD       Y_LOCAL
          BMI       ERROR
          CMPD      #191
          BGT       ERR_1
          ASLD
          ASLD
          ASLD
          ASLD
          ASLD
*
*         Add the horizontal and vertical positions.
          ADDD      ,S++
*
*         Get into the right block of memory.
          ADDD      #BASE
*
*         Point at the byte with the index reg. X.
          TFR       D,X
```

```
*
*         Form the contents of the object byte.
*
          LDD         X_LOCAL
          CLRA
          ANDB        #$07
          LDY         #TABLE
          LDA         B,Y
          RTS
ERR_1      PULS D
ERROR     LDB         #$FF
          STB         ERR_SW
          LDX         #MESAGE
          JSR         PSTRING
          RTS
TABLE     FCB         $80,$40,$20,$10,8,4,2,1
MESAGE    FCC         "Cannot plot this point.",$04
          STTL        VERT-Plot a vertical line.
          PAG
*
*
*         This routine plots a vertical line.
*         Calls       FIND
*         References
*         X1,X2,Y1,Y2,ERASE,TEMP
*
*
VERT      PSHS        #RLIST
          LDA        #DIR
          TFR         A,DP
*
*         If Y1 > Y2, swap them.
          LDD         Y2
          CMPD        Y1
          BGT         VERT_1
          LDX         Y1
          STD         Y1
          STX         Y2
VERT_1    LDD         X1
          STD         X_LOCAL
          LDD         Y1
          STD         Y_LOCAL
          JSR         FIND
          TST         ERR_SW
          BMI         VERT_4
          STA         TEMP
          LDB         Y1+1
V_LOOP    LDA         TEMP
          TST         ERASE
          BPL         VERT_2
          COMA
```

```
          ANDA      ,X
          BRA       VERT_3
VERT_2    ORA       ,X
VERT_3    STA       ,X
          LEAX      32,X
          INCB
          CMPB      Y2+1
          BLS       V_LOOP
VERT_4    PULS      #RLIST
          RTS
*
          STTL      DIVIDE
          PAG
*
*
*
*         This routine divides DIVND by DIVSR
*         where both are positive integers
*         and DIVSR > DIVND.
*         QUOT is an eight bit unsigned fraction
*         with the binary point to the left of
*         the MSB.
*
DIVIDE    LDX       #0
*
*         Adjust the dividend to the left.
          LDD       DIVND
DIV_1     ASLB
           ROLA
          BMI       DIV_2
          LEAX      1,X
          BRA       DIV_1
DIV_2     LSRA
          RORB
          LEAX      -2,X
          STD       DIVND
*
*         Adjust the divisor to the left.
          LDD       DIVSR
DIV_3     ASLB
           ROLA
          BMI       DIV_4
          LEAX      -1,X
          BRA       DIV_3
DIV_4     LSRA
          RORB
          LEAX      1,X
          CMPD      DIVND
          BGT       DIV_4
          STD       DIVSR
          LDY       #0
```

```
          CLR       QUOT
          LDD       DIVND
*
*         Divide those digits.
DV_TEST   CMPD      DIVSR
          BGE       DIV_5
          LEAY      1,Y
          ASL       QUOT
          BRA       D_END
DIV_5     SUBD      DIVSR
          LEAY      1,Y
          ASL       QUOT
          INC       QUOT
D_END     ASLB
           ROLA
          CMPY      #8
          BLT       DV_TEST
*
*Adjust the quotient.
Q_ADJ     CMPX      #0
          BEQ       D_RTS
          LSR       QUOT
          LEAX           -1,X
          BRA       Q_ADJ
*
*         Return
D_RTS     RTS
           STTL WINDOW
          PAG
*
*WINDOW**********************************
WINDOW     PSHS #RLIST
           LDU  #BASE
          LDA #DIR
          TFR  A,DP
           CLR  SLP_SI
          LDD  G_X1
          CMPD G_X2
          BLT  W__5
           BGT  W__3
*      G_X1=G_X2  =>  VERTICAL LINE
W__1        LDD  #$7FFF
            STD  SLOPE
            ORA  #$FF
            STD  SLOPE+2
           LDD  G_Y2
           CMPD G_Y1
           BPL  W__8
           COM  SLP_SI
            BRA  W__8
W__3       PSHS D
```

```
            LDD  G_X2
            STD  G_X1
            PULS D
            STD  G_X2
            LDD  G_Y1
            PSHS D
            LDD  G_Y2
            STD  G_Y1
            PULS D
            STD  G_Y2
            COM  SWP_SW
*          calculate slope
W__5       ZERO A_0,16
           LDD  G_Y2
           SUBD G_Y1
            BPL  W__6
            COM SLP_SI
            NEGD
W__6       STD  C_2
           LDD  G_X2
           SUBD G_X1
           STD  B_0
           JSR  L_DIV
            LDD  A_0
            STD  SLOPE
            LDD  A_2
            STD  SLOPE+2
*
*          start the search
W__8       LDD G_X1
*          if x1 > x-hi, exit
           CMPD X_HI
           LBGT OUT
*          if x1 < x-lo, w-6
           CMPD X_LO
           LBLT W_6
*           if  y1 > y-hi, w-4
            LDD  G_Y1
            CMPD Y_HI
            LBGT W_4
*       if y1 < y-lo, w-2
             CMPD Y_LO
             BLT W_2
*        p1 on screen
                   LDD  G_X1
                   STD  X1
                   LDD  G_Y1
                   STD  Y1
               LDD  X_HI
*        is p2 on the right, w-1
             CMPD G_X2
```

```
            BLT  W_1
*       p2 is in the middle column
             LDD  G_Y2
*  if p2 above the screen, top boundary routine
             CMPD Y_HI
             LBGT TOP_BD
*   if p2 below the screen, bottom boundary routine
              CMPD Y_LO
              LBLT BOT_BD
*    p2 on the screen, draw the line.
               LDD  G_X2
               STD  X2
               LDD  G_Y2
               STD  Y2
               LBRA DR_PRP
*
*      p1 on screen,
*      p2 in right-center
W_1          JSR CALC_Y X_HI    ytest
*      if ytest > y-hi, top boundary
             LDD  YTEST
             CMPD Y_HI
             LBGT  TOP_BD
*      if ytest < y-lo, bottom boundary
             CMPD Y_LO
             LBLT  BOT_BD
*      p2 off the right side
*      of the screen
               LDD  X_HI
               STD  X2
               LDD  YTEST
               STD  Y2
               LBRA DR_PRP
*     p1 in bottom center
*     if x2 > x-hi, w-3
W_2          LDD  X_HI
             CMPD G_X2
             BLE  W_3
*      p2 in the center
*      if y2 < y-lo, exit
             LDD  G_Y2
             CMPD Y_LO
             LBLT  OUT
*      if x1 = x2, w-2-2
             LDD  G_X1
             CMPD G_X2
             BNE  W_2_2
              STD  X1
              LDD  Y_LO
              STD  Y1
              BRA  W_2_4
```

```
*        calculate x1=(y-lo - y1)/slope + x1
W_2_2      LDD  Y_LO
             JSR  CALC_X  Y_LO
             STD  X1
*       y1 = y-lo
             LDD  Y_LO
             STD  Y1
*       if y2 > y-hi, top
*       if y2 <= y-hi, draw
W_2_4        LDD  G_Y2
             CMPD Y_HI
             LBGT TOP_BD
                LDD  G_X2
                STD  X2
                LDD  G_Y2
                STD  Y2
             LBRA DR_PRP
*       p1 in bottom center
*       p2 on right
W_3          JSR CALC_Y  X_HI   ytest
             CMPD Y_LO
             LBLT OUT
           LDD  Y_LO
             JSR  CALC_X  Y_LO   x1
             STD  X1
*       y-lo ==> y1
             LDD  Y_LO
             STD  Y1
             LDD  YTEST
             CMPD Y_HI
             LBGT TOP_BD
*        ytest ==> y2
*        x-hi ==> x2
              STD  Y2
              LDD  X_HI
              STD  X2
              LBRA DR_PRP
*          p1 in upper center
*          if x2 > x-hi, w-5
W_4        LDD  X_HI
           CMPD G_X2
           BLE  W_5
*       if y2 > y-hi, exit
            LDD  Y_HI
            CMPD G_Y2
            LBLE OUT
             JSR  CALC_X  Y_HI    x1
             STD  X1
*        y-hi ==> y1
             LDD  Y_HI
             STD  Y1
```

```
*       if y2 < y-lo, bottom
             LDD  G_Y2
             CMPD Y_LO
             LBLT BOT_BD
*           p2 on screen
              LDD  G_X2
              STD  X2
              LDD  G_Y2
              STD  Y2
              LBRA DR_PRP
*           p1 in upper center
*           p2 on right
W_5         JSR CALC_Y  X_HI    ytest
            CMPD Y_HI
            LBGT OUT
            LDD  Y_HI
            JSR  CALC_X  Y_HI
            STD  X1
            LDD  Y_HI
            STD  Y1
*           if ytest < y-lo , bottom
            LDD  YTEST
            CMPD Y_LO
            LBLT BOT_BD
*       ytest ==> y2
*       x-hi ==> x2
             STD  Y2
             LDD  X_HI
             STD  X2
             LBRA DR_PRP
*           p1 on left
*           if x2 <= x-lo, exit
W_6         LDD  X_LO
            CMPD G_X2
            LBGT OUT
*           calc  ytest
*           at left bound
            JSR CALC_Y  X_LO
*           if y1 > y-hi, w-14-5
            LDD  G_Y1
            CMPD Y_HI
            LBGT W_14_5
*       if y1 < y-lo, w-8
             CMPD Y_LO
             LBLT W_8
*          if ytest >= y-hi, exit
*          if ytest <= y-lo, exit
               LDD  YTEST
               CMPD Y_HI
               LBGE OUT
               CMPD Y_LO
```

```
            LBLE OUT
*         ytest ==> y1
*         x-lo ==> x1
               STD  Y1
               LDD  X_LO
               STD  X1
*         if x2 > x-hi, w-7
               LDD  X_HI
               CMPD G_X2
               BLE W_7
*          if y2 > y-hi, top
*          if y2 < y-lo, bottom
                 LDD  G_Y2
                 CMPD Y_HI
                 LBGT TOP_BD
                  CMPD Y_LO
                  LBLT BOT_BD
*         p2 on the screen
                   LDD  G_X2
                   STD  X2
                   LDD  G_Y2
                   STD  Y2
                   LBRA  DR_PRP
*        p1 on left
*        p2 on right
W_7           JSR  CALC_Y  X_HI
              CMPD Y_HI
              LBGT TOP_BD
                CMPD Y_LO
                LBLT BOT_BD
                STD  Y2
                 LDD X_HI
                STD X2
                 LBRA DR_PRP
*      p1 in leftlower
*      if ytest > a-hi, exit
W_8      -   LDD  YTEST
             CMPD Y_HI
             LBGT OUT
*      if y2 < y-lo, exit
             LDD  G_Y2
             CMPD Y_LO
             LBLT OUT
*      if x2 > x-hi, w-12
             LDD  G_X2
             CMPD X_HI
             BGT  W_12
*       if y2 <= y-hi, w-10
               LDD  G_Y2
               CMPD Y_HI
               BLE  W_10
```

```
*       if ytest >= y-lo, w-9
            LDD  YTEST
            CMPD Y_LO
            BGE  W_9
*        line crosses bottom
*        and top boundary
         LDD  Y_LO
              JSR  CALC_X  Y_LO
              STD  X1
              LDD  Y_LO
              STD  Y1
              BRA  TOP_BD
*       line crosses the left
*       and top boundary
W_9           LDD  X_LO
              STD  X1
              LDD  YTEST
              STD  Y1
              BRA  TOP_BD
*      p2 on the screen
*      if ytest >= y-lo, w-11
W_10        LDD  Y_LO
            CMPD YTEST
            BLT  W_11
*       line crosses bottom
*       p2 on screen
              JSR  CALC_X  Y_LO
              STD  X1
              LDD  Y_LO
              STD  Y1
              LDD  G_X2
              STD  X2
              LDD  G_Y2
              STD  Y2
              LBRA DR_PRP
*      line crosses left
*      p2 on screen
W_11        LDD  X_LO
            STD  X1
            LDD  YTEST
            STD  Y1
            LDD  G_X2
            STD  X2
            LDD  G_Y2
            STD  Y2
            LBRA DR_PRP
*     p1 in lower left
*     p2 on right top
*      or right center
W_12       LDD  YTEST
           CMPD Y_LO
```

```
            BLT  W_13
*       line crosses left
            STD  Y1
            LDD  X_LO
            STD  X1
            BRA  W_14
*      line crosses bottom
W_13       LDD  Y_LO
            JSR  CALC_X  Y_LO
            STD  X1
            LDD  Y_LO
           STD  Y1
*      does line cross
*      top or right
W_14       LDD  X_HI
            JSR  CALC_Y  X_HI
            CMPD Y_HI
            BGT  TOP_BD
*       line crosses right
            STD  Y2
            LDD  X_HI
            STD  X2
            LBRA  DR_PRP
*      line crosses top
TOP_BD     LDD  Y_HI
            JSR  CALC_X  Y_HI
            STD  X2
            LDD  Y_HI
            STD  Y2
            LBRA  DR_PRP
*          p1 in upper left
*          if ytest < y-lo, exit
*          if y2 > y-hi, exit
*          if x2 > x-hi, w-18
W_14_5     LDD  YTEST
           CMPD Y_LO
           LBLT  OUT
           LDD  G_Y2
           CMPD Y_HI
           LBGT  OUT
           LDD  G_X2
           CMPD X_HI
           BGT  W_18
*      if y2 >= y-lo, w-16
            LDD  G_Y2
            CMPD Y_LO
            BGE  W_16
*       if ytest >= y-hi, w-15
             LDD  YTEST
             CMPD Y_HI
             BGE  W_15
```

```
*       line crosses left
             STD  Y1
             LDD  X_LO
             STD  X1
             BRA  BOT_BD
*      line crosses top
W_15      LDD  Y_HI
            JSR  CALC_X  Y_HI
            STD  X1
            LDD  Y_HI
            STD  Y1
            BRA  BOT_BD
*     p2 on screen
W_16           LDD  G_X2
               STD  X2
               LDD  G_Y2
               STD  Y2
            LDD  YTEST
            CMPD Y_HI
            BLT  W_17
*      line crosses top
           LDD  Y_HI
             JSR  CALC_X  Y_HI
             STD  X1
             LDD Y_HI
             STD  Y1
             BRA  DR_PRP
*     line crosses left
W_17       STD  Y1
           LDD  X_LO
           STD  X1
           BRA  DR_PRP
*         p1 in upper left
*         p2 on right
W_18      LDD  YTEST
          CMPD Y_HI
          BGT  W_19
*     line crosses left
           STD  Y1
           LDD  X_LO
           STD  X1
           BRA  W_20
*         does line cross top?
W_19      LDD  X_HI
          JSR  CALC_Y  X_HI
          LDD  Y_HI
          CMPD YTEST
          LBLT  OUT
*         line does cross top
          JSR  CALC_X  Y_HI
          STD  X1
```

```
          LDD  Y_HI
          STD  Y1
*         does the line cross
*         on right side
*         or on the bottom
W_20      LDD X_HI
          JSR  CALC_Y  X_HI
          CMPD Y_LO
          BLT  BOT_BD
*      line crosses right
           STD  Y2
           LDD  X_HI
           STD  X2
           BRA  DR_PRP
BOT_BD    LDD  Y_LO
          JSR  CALC_X  Y_LO
          STD  X2
          LDD  Y_LO
          STD  Y2
DR_PRP    TST  SWP_SW
          BEQ  PRP_2
*      swap x1,x2 & y1,y2
           LDD  G_X1
           PSHS D
           LDD  G_X2
           STD  G_X1
           PULS D
           STD  G_X2
           LDD  G_Y1
           PSHS D
           LDD  G_Y2
           STD  G_Y1
           PULS D
           STD  G_Y2
           CLR  SWP_SW
*
PRP_2     LDD  X_LO
          PSHU D
          LDD  X1
          PSHU D
          JSR  SCALE
          STD    X1
*
          LDD  Y_LO
          PSHU D
          LDD  Y1
          PSHU D
          JSR  SCALE
          STD    Y1
*
          LDD  X_LO
```

```
          PSHU D
          LDD  X2
          PSHU D
          JSR  SCALE
          STD    X2
*
          LDD  Y_LO
          PSHU D
          LDD  Y2
          PSHU D
          JSR  SCALE
          STD    Y2
*
          CLRA
          ADDA #1
          STA WIN_SW
          JSR  GRAPH
          PULS #RLIST
          RTS
OUT       TST  SWP_SW
          BEQ  PRP_8
*      swap x1,x2 & y1,y2
           LDD  G_X1
           PSHS D
           LDD  G_X2
           STD  G_X1
           PULS D
           STD  G_X2
           LDD  G_Y1
           PSHS D
           LDD  G_Y2
           STD  G_Y1
           PULS D
           STD  G_Y2
PRP_8      CLRA
          STA  WIN_SW
          NOP
          PULS #RLIST
          RTS
*
*
*                    p-out = sqrt( rad^2 - p-in^2 )
SID_OP    PSHS #RLIST
          ZERO  A_0,8
          LDD  P_IN
          STD  A_0
          STD  B_0
          JSR  L_MULT
          TRNSFR  C_0,SCR_0,8
          TRNSFR  R_SQ,C_0,8
          SUBT SCR_0,C_0,8
```

```
            JSR   SQ_RT
            LDD   A_0
            STD   P_OUT
            PULS  #RLIST
            RTS
*
*
CIR         PSHS  #RLIST
            JSR   C_INIT
            CLRA
            CLRB
*
CIR_10      STD   P_IN
            JSR   SID_OP
            JSR    QUAD_1
            JSR    QUAD_2
            JSR    QUAD_3
            JSR    QUAD_4
            LDD   P_IN
            ADDD  #1
            CMPD  DONE
            BLE   CIR_10
CIR_RT      PULS  #RLIST
            RTS
*
*
QUAD_1      LDD   X1
            ADDD  P_IN
            BMI   Q1_10
            CMPD  CRT_X
            BGT   Q1_RT
            STD   X_LOCA
            LDD   Y1
            ADDD  P_OUT
            BMI   Q1_RT
            CMPD  CRT_Y
            BGT   Q1_10
            STD   Y_LOCA
            JSR   PLOT
Q1_10       LDD   X1
            ADDD  P_OUT
            BMI   Q1_RT
            CMPD  CRT_X
            BGT   Q1_RT
            STD   X_LOCA
            LDD   Y1
            ADDD  P_IN
            BMI   Q1_RT
            CMPD  CRT_Y
            BGT   Q1_RT
            STD   Y_LOCA
```

```
            JSR   PLOT
Q1_RT       RTS
*
QUAD_2      LDD   X1
            SUBD  P_IN
            BMI   Q2_RT
            CMPD  CRT_X
            BGT   Q2_10
            STD   X_LOCA
            LDD   Y1
            ADDD  P_OUT
            BMI   Q2_RT
            CMPD  CRT_Y
            BGT   Q2_10
            STD   Y_LOCA
            JSR   PLOT
Q2_10       LDD   X1
            SUBD  P_OUT
            BMI   Q2_RT
            CMPD  CRT_X
            BGT   Q2_RT
            STD   X_LOCA
            LDD   Y1
            ADDD  P_IN
            BMI   Q2_RT
            CMPD  CRT_Y
            BGT   Q2_RT
            STD   Y_LOCA
            JSR   PLOT
Q2_RT       RTS
*
QUAD_3      LDD   X1
            SUBD  P_IN
            BMI   Q3_RT
            CMPD  CRT_X
            BGT   Q3_10
            STD   X_LOCA
            LDD   Y1
            SUBD  P_OUT
            BMI   Q3_10
            CMPD  CRT_Y
            BGT   Q3_RT
            STD   Y_LOCA
            JSR   PLOT
Q3_10       LDD   X1
            SUBD  P_OUT
            BMI   Q3_RT
            CMPD  CRT_X
            BGT   Q3_RT
            STD   X_LOCA
            LDD   Y1
```

```
            SUBD  P_IN
            BMI   Q3_RT
            CMPD  CRT_Y
            BGT   Q3_RT
            STD   Y_LOCA
            JSR   PLOT
Q3_RT       RTS
*
QUAD_4      LDD   X1
            ADDD  P_IN
            BMI   Q4_10
            CMPD  CRT_X
            BGT   Q4_RT
            STD   X_LOCA
            LDD   Y1
            SUBD  P_OUT
            BMI   Q4_10
            CMPD  CRT_Y
            BGT   Q4_RT
            STD   Y_LOCA
            JSR   PLOT
Q4_10       LDD   X1
            ADDD  P_OUT
            BMI   Q4_RT
            CMPD  CRT_X
            BGT   Q4_RT
            STD   X_LOCA
            LDD   Y1
            SUBD  P_IN
            BMI   Q4_RT
            CMPD  CRT_Y
            BGT   Q4_RT
            STD   Y_LOCA
            JSR   PLOT
Q4_RT       RTS
*
*
C_INIT      LDU   #BASE
            LDA   #DIR
            TFR   A,DP
*
*                 scale radius
            LDD   #0
            PSHU  D
            LDD   RADIUS
            PSHU  D
            JSR   SCALE
            STD   RADIUS
*
*                 scale cnter point
            LDD   X_LO
```

```
            PSHU D
            LDD  X1
            PSHU D
            JSR  SCALE
            STD  X1
*
            LDD  Y_LO
            PSHU D
            LDD  Y1
            PSHU D
            JSR  SCALE
            STD  Y1
*
*                find terminal value
            LDD  RADIUS
            ASRA
            RORB
            STD  DONE
            ASRA
            RORB
            ADDD DONE
            STD  DONE
*
*                calculate r^2
            ZERO A_0,8
            LDD  RADIUS
            STD  A_0
            STD  B_0
            JSR  L_MULT
            TRNSFR C_0,R_SQ,8
            RTS
*
*
Q_CIR       PSHS #RLIST
            JSR  C_INIT
*                2*code + addr_t => adr-1
            LDD  CODE
            ASLB
            ROLA
            ANDB #7
            CLRA
            LDX  #ADDR_T
            LEAX D,X
            LDD  ,X
            STD  ADR_1+1
            CLRA
            CLRB
Q_C_10      STD  P_IN
            JSR  SID_OP
ADR_1       JSR  QUAD_4
            LDD  P_IN
```

```
            ADDD #1
            CMPD DONE
            BLE  Q_C_10
            PULS #RLIST
            RTS
*
*
H_CIR       PSHS #RLIST
            JSR  C_INIT
            LDD  CODE
            ASLB
            ROLA
            ANDB #7
            CLRA
            STD CODE
            LDX  #ADDR_T
            LEAX D,X
            LDD  ,X
            STD  ADR_2+1
            LDD  CODE
            ADDD #2
            ANDB #$07
            LDX  #ADDR_T
            LEAX D,X
            LDD  ,X
            STD  ADR_3+1
            CLRA
            CLRB
H_C_10      STD  P_IN
            JSR  SID_OP
ADR_2       JSR  QUAD_3
ADR_3       JSR  QUAD_4
            LDD  P_IN
            ADDD #1
            CMPD DONE
            BLE  H_C_10
            PULS #RLIST
            RTS
*
            STTL PRNTR  hor and vert labels
            PAG
*
*
*
PRNTR       PSHS #RLIST
            LDA  #DIR
            TFR  A,DP
            LDX  [CHPTR]
            LDY  CHPTR
            LEAY 2,Y
            LDD  ,Y
```

```
          TFR   D,Y
          LDB   ,X+
          CLR   EX_SW
PR_10     LDB   ,X+
          LEAY  -1,Y
          BEQ   PR_EX
          BSR   CH_0
          TST   EX_SW
          BNE   PR_EX
          BRA   PR_10
*
PR_EX     PULS  #RLIST
          RTS
* form the address
CH_0       PSHS #RLIST
          ANDB  #$7F
          CMPB  #$20
          LBLT   P_EXIT
          CMPB  #$5F
          BGT   CH_10
          SUBB  #$20
          BRA   CH_20
CH_10      SUBB #$40
*  0 < (B) < 63 -- capital letter or special
CH_20      CMPB #$20
          BLT       CH_30
          SUBB  #$20
          LDX   #HTBL
          LDA   B,X
          LDX   #FQUT
          BRA   CH_40
CH_30     LDX   #LTBL
          LDA   B,X
          LDX   #SP
CH_40     CLRB
          EXG   A,B
          LEAX  D,X
          LDY   #7
CH_45     LEAY  -1,Y
          LBEQ   CH_100
           CLRA
          LDB   ,X
          LBEQ   NEXTCH
          ASRB
          ASRB
          ASRB
          ASRB
          STD   G_X1
          CLRA
          LDB   ,X+
          ANDB  #$0F
```

```
            CMPB #8
            BLT  CH_47
            ORB  #$F0
            SEX
CH_47        STD  G_Y1
            CLRA
            LDB  ,X
            ASRB
            ASRB
            ASRB
            ASRB
            STD  G_X2
            CLRA
            LDB  ,X+
            ANDB #$0F
            CMPB #8
            BLT  CH_48
            ORB  #$F0
            SEX
CH_48        STD  G_Y2
            TST  CODE
            BEQ  CH_50
* vertical print
            LDD  G_X1
            ADDD Y0
            CMPD #191
            LBGT  P_EXIT
            STD  Y1
            LDD  G_Y1
            NEGD
            ADDD X0
            BLT P_EXIT
            STD  X1
            LDD  G_X2
            ADDD Y0
            CMPD #191
            BGT  P_EXIT
            STD  Y2
            LDD  G_Y2
            NEGD
            ADDD X0
            BLT  P_EXIT
            STD  X2
            JSR  GRAPH
            BRA  CH_45
* horizontal print
CH_50        LDD  G_X1
            ADDD X0
            CMPD #255
            BGT P_EXIT
            STD  X1
```

```
            LDD   G_Y1
            ADDD  Y0
            CMPD  #191
            BGT   P_EXIT
            STD   Y1
            LDD   G_X2
            ADDD  X0
            CMPD  #255
            BGT P_EXIT
            STD   X2
            LDD   G_Y2
            ADDD  Y0
            CMPD  #191
            BGT   P_EXIT
            STD   Y2
            JSR   GRAPH
            LBRA   CH_45
NEXTCH      TST   CODE
            BEQ   CH_100
            LDD   Y0
            ADDD  #6
            CMPD  #187
            BGT   P_EXIT
            STD   Y0
            BRA   CH_110
CH_100      LDD   X0
            ADDD  #6
            CMPD  #252
            BGE   P_EXIT
            STD   X0
            BRA   CH_110
P_EXIT      COM   EX_SW
CH_110      PULS  #RLIST
            RTS
*
LTBL        FCB    0
            FCB   1
            FCB   6
            FCB   11
            FCB   20
            FCB   29
            FCB   36
            FCB   47
            FCB   50
            FCB   57
            FCB   64
            FCB   71
            FCB   76
            FCB   79
            FCB   82
            FCB   85
```

```
                FCB   88
                FCB   99
                FCB   106
                FCB   113
                FCB   124
                FCB   131
                FCB   142
                FCB   153
                FCB   158
                FCB   171
                FCB   182
                FCB   187
                FCB   192
                FCB   197
                FCB   202
                FCB   207
HTBL            FCB    0
                FCB   3
                FCB   12
                FCB   25
                FCB   36
                FCB   45
                FCB   54
                FCB   61
                FCB   72
                FCB   79
                FCB   86
                FCB   93
                FCB   100
                FCB   105
                FCB   114
                FCB   121
                FCB   130
                FCB   139
                FCB   150
                FCB   161
                FCB   172
                FCB   177
                FCB   184
                FCB   189
                FCB   198
                FCB   203
                FCB   210
                FCB   217
                FCB   224
                FCB   227
                FCB   234
                FCB   243
*
*
STRK_T
```

```
SP      FCB  0
        FCB  $20,$20,$27,$22,0
        FCB  $27,$25,$47,$45,0
        FCB  $16,$11,$36,$31,$02,$42,$04,$44,0
        FCB  $01,$41,$06,$46,$05,$42,$27,$20,0
        FCB  $07,$16,$31,$40,$47,$00,0
        FCB  $40,$15,$27,$27,$36,$03,$02,$01
        FCB  $10,$30,0
        FCB  $25,$47,0
        FCB  $47,$26,$25,$22,$21,$40,0
        FCB  $07,$17,$26,$21,$10,$00,0
        FCB  $02,$45,$05,$42,$26,$21,0
        FCB  $03,$43,$25,$21,0
        FCB  $21,$1F,0
        FCB  $03,$43,0
        FCB  $20,$20,0
        FCB  $47,$00,0
        FCB  $01,$06,$17,$37,$46,$41,$10,$30,0,0,0
        FCB  $16,$26,$27,$20,$10,$30,0
        FCB  $07,$37,$46,$00,$40,$00,0
        FCB  $07,$37,$47,$14,$24,$34,$43,$41
        FCB  $30,0,0
        FCB  $27,$02,$02,$42,$37,$30,0
        FCB  $07,$47,$06,$04,$14,$34,$43,$41
        FCB  $30,0,0
        FCB  $47,$17,$06,$01,$14,$34,$10,$30
        FCB  $41,$43,0
        FCB  $07,$47,$47,$00,0
        FCB  $03,$46,$06,$43,$02,$01,$10,$30
        FCB  $17,$37,$42,$41,0
        FCB  $30,$00,$41,$46,$37,$17,$06,$04
        FCB  $13,$43,0
        FCB  $21,$21,$25,$25,0
        FCB  $25,$25,$21,$1F,0
        FCB  $03,$45,$03,$41,0
        FCB  $02,$42,$04,$44,0
        FCB  $01,$43,$43,$05,0
        FCB  $05,$06,$17,$37,$46,$22,$20,$20,0
FQUT    FCB  $07,$25,0
        FCB  $06,$00,$17,$37,$46,$40,$13,$33,0
        FCB  $07,$00,$17,$37,$14,$34,$10,$30
        FCB  $46,$45,$43,$41,0
        FCB  $41,$41,$10,$30,$06,$01,$17,$37
        FCB  $46,$46,0
        FCB  $07,$00,$17,$37,$10,$30,$46,$41,0
        FCB  $07,$00,$17,$47,$14,$34,$10,$40,0
        FCB  $07,$00,$17,$47,$14,$34,0
        FCB  $06,$01,$17,$47,$10,$40,$22,$43
        FCB  $42,$41,0
        FCB  $07,$00,$47,$40,$14,$34,0
        FCB  $27,$20,$17,$37,$10,$30,0
```

```
FCB   $47,$41,$10,$30,$02,$01,0
FCB   $07,$00,$03,$47,$24,$40,0
FCB   $07,$00,$10,$40,0
FCB   $07,$00,$07,$23,$23,$47,$47,$40,0
FCB   $07,$00,$07,$40,$47,$40,0
FCB   $06,$01,$17,$37,$10,$30,$46,$41,0
FCB   $07,$00,$17,$37,$13,$33,$46,$44,0
FCB   $06,$01,$17,$37,$10,$30,$46,$41
FCB   $22,$40,0
FCB   $07,$00,$17,$37,$13,$33,$46,$44
FCB   $42,$40,0
FCB   $01,$01,$10,$30,$06,$41,$17,$37
FCB   $46,$46,0
FCB   $26,$20,$07,$47,0
FCB   $07,$01,$10,$30,$47,$41,0
FCB   $07,$20,$20,$47,0
FCB   $07,$10,$10,$25,$25,$30,$30,$47,0
FCB   $17,$40,$37,$00,0
FCB   $07,$24,$24,$47,$23,$20,0
FCB   $07,$37,$47,$00,$10,$40,0
FCB   $17,$10,$27,$37,$20,$30,0
FCB   $07,$40,0
FCB   $20,$00,$07,$27,$37,$30,0
FCB   $04,$27,$27,$44,0,0,0,0,0
FCB   $04,$00,0,0,0
END   FLEX
```