
Retrospective Theses and Dissertations

1988

Saw Draw: An Interactive Graphical Layout System for Surface Acoustic Wave Devices

Jeffrey Blair Abbott
University of Central Florida

 Part of the [Engineering Commons](#)

Find similar works at: <https://stars.library.ucf.edu/rtd>

University of Central Florida Libraries <http://library.ucf.edu>

This Masters Thesis (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Retrospective Theses and Dissertations by an authorized administrator of STARS. For more information, please contact STARS@ucf.edu.

STARS Citation

Abbott, Jeffrey Blair, "Saw Draw: An Interactive Graphical Layout System for Surface Acoustic Wave Devices" (1988). *Retrospective Theses and Dissertations*. 5146.
<https://stars.library.ucf.edu/rtd/5146>

SAW DRAW: AN INTERACTIVE GRAPHICAL LAYOUT SYSTEM
FOR SURFACE ACOUSTIC WAVE DEVICES

BY

JEFFREY BLAIR ABBOTT
B.S.E., University of Central Florida, 1985

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Engineering
in the Graduate Studies Program
of the College of Engineering
University of Central Florida
Orlando, Florida

Spring Term
1988

ABSTRACT

This thesis introduces a solution to the problem of time and memory space requirements associated with the manipulation/creation of solid state device layout. Through the use of a hierarchical organization of data and a tailored indexing technique, the software described here, referred to as Saw Draw, is capable of manipulating huge amounts of data in a short period of time. This program was written for surface acoustic wave (SAW) device layout but works for a broad range of devices to include semiconductors, microstrip and others.

Due to the large number of details which must be stored for each SAW structure, simply displaying a typical SAW device can become exceedingly tedious. When an entire mask of devices is organized, disk storage requirements can become prohibitive. This software has been designed to minimize both of these critical problems.

This work describes the capabilities, structure and special algorithms used in Saw Draw. Included is an example of how a SAW device might be created and a listing of the program code in the Appendix.

ACKNOWLEDGMENTS

I would like to acknowledge Don Malocha, Ben Abbott and Clinton Hartman for their assistance in defining the edit functions within Saw Draw. I would also like to acknowledge Amar Mukherjee without whose assistance I would not have become introduced to today's VLSI layout technology. Finally, I acknowledge the support I have received from my family, friends and members of the research group in the Solid State Devices and Systems Laboratory at the University of Central Florida, to which I belong.

TABLE OF CONTENTS

LIST OF FIGURES.	vi
LIST OF TABLES	ix
INTRODUCTION	1
BASIC THEORY	3
MODES OF OPERATION	12
GRAPHICAL INTERFACE.	21
Draw Function (D Key Stroke).	22
Scroll on Screen Function (O Key Stroke).	23
Zoom Functions (Z & Cntl+Z Keystroke)	24
View Function (V Keystroke)	28
EDITING ALGORITHMS AND COMMANDS.	30
Array Function (A Keystroke).	32
Box Function (B Keystroke).	34
Copy Function (C Keystroke)	34
Hard Expand Function (E Keystroke).	36
Soft Expand Function (Cntl+E Keystroke)	38
Get File Function (F Keystroke)	39
Include Cell Function (Cntl+F Keystroke).	41
Grid Function (G Keystroke)	43
Change Layer Function (H Keystroke)	45
Kill Function (K Keystroke)	45
Set Layer Function (L Keystroke).	45
Move/Rotate/Scale Function (M Keystroke).	45
Merge Cell Function (Alt+M Keystroke)	48
Paint/Add Function (P Keystroke).	48
Step and Repeat Function (R Keystroke).	49
Select Function (S Keystroke)	51
Edit Cell Function (Cntl+S Keystroke)	51
Unedit Cell Function (Alt+S Keystroke).	53
Text/Label Function (T Keystroke)	53
Unexpand Function (U Keystroke)	55
Wavelength Function (W Keystroke)	55
X Mirror Image Function (X Keystroke)	56
Y Mirror Image Function (Y Keystroke)	56
EXAMPLE.	57

CONCLUSIONS.	67
APPENDIX	68
REFERENCES	69

LIST OF FIGURES

1. Cell Hierarchy	7
2. File Menu.	12
3. Sample Open of the Cell: SAW	13
4. Sample Plot of the Cell: SAW	14
5. Output Menu.	14
6. Printer Selection Menu	16
7. Define Print Area.	17
8. Sample Printout of the Cell: SAW	18
9. Help Menu.	19
10. Edit/Mouse mode	20
11. Scroll on Screen Prompts.	23
12. Scroll on Screen Results.	24
13. Z Keystroke (Zoom) Example.	25
14. Z Keystroke Results	26
15. Cntl+Z (Zoom) Keystroke Example	27
16. Cntl+Z Keystroke Results.	28
17. V Keystroke (View) Example.	29
18. Array Selections Prompts.	33
19. Array Selections Results.	33
20. Set Edit Box Position	34
21. Copy Function	35

22.	Copy Results.	36
23.	Expand Selected Subcell Function.	37
24.	Expand Selected Subcell Results	37
25.	Expand Level Function	38
26.	Expand Level Results.	39
27.	Get File Function	40
28.	Get File Prompts.	40
29.	Get Rile Results.	41
30.	Include Cell Function	42
31.	Include Cell Function Prompts	42
32.	Include Cell Results.	43
33.	Grid Function	44
34.	Grid Function Results	44
35.	Move/Scale/Rotate Function.	46
36.	Move/Rotate/Scale Prompts	47
37.	Move/Rotate/Scale Results	47
38.	Merge Cell Function Prompts	48
39.	Step and Repeat Function.	50
40.	Step and Repeat Prompts	50
41.	Step and Repeat Results	51
42.	Edit Cell Function.	52
43.	Edit Cell Results	53

44.	Text/Label Function Prompts	54
45.	Text/Label Results.	55
46.	Example: Two Rectangles Added	58
47.	Example: Select Rectangles.	59
48.	Example: Array Prompts.	60
49.	Example: Array Results.	61
50.	Example: Bus Bar Added.	62
51.	Example: Copy and Y Mirror Image.	63
52.	Example: Complete Transducer.	64
53.	Example: Cell Included.	65
54.	Example: Complete Device.	66

LIST OF TABLES

1. Saw Draw Record Structure	3
2. Example Cell	6
3. Virtual Versus Actual Records Stored	10
4. Saw Cad Record Structure	15
5. Windowing Commands	22
6. Edit Commands.	31

INTRODUCTION

In the past, three classes of software packages have been used in surface acoustic wave, or SAW, device layout. These are VLSI tools, CAD drawing tools and custom SAW software [2]. All three of these types of software suffer short falls in SAW layout.

VLSI layout tools are expensive and are written for specific silicon fabrication technologies. Because these tools are rule based [4]. Many of these rules exclude specific patterns of layout used by the SAW technology. Also, some of the graphical layout algorithms needed for SAW layout are not supported.

CAD software suffers from many problems. First, these packages are not easily manipulated for SAW layout [2]. This is due to the fact that CAD systems are based on vector or bit-mapped graphics rather than rectangle formats. Second, this software rarely takes advantage of hierarchical structures. Third, as with VLSI tools, many graphical layout algorithms needed for SAW layout are not supported.

Custom software, although written by SAW engineers, manipulates files of data rather than individual elements. That is, edit capabilities at the rectangle level are

virtually nonexistent. The files are produced using SAW compilers with a desired time response and which produce a device structure. They do not support an interactive graphical interface. Their hierarchical structures are limited to the mask level and the file level. At the file level, one file is not capable of including other files as sources of data. Finally, custom SAW software is rarely written to accommodate multi-layered devices which require more than one mask level.

Saw Draw overcomes all of these shortfalls. Being hierarchally based, storage requirements for an entire mask can realistically be reduced two to six orders of magnitude. The hierarchy also increases plot time since if an entire file of data does not fall within the limits of the graphic view port, it is skipped. The graphic interface is fully integrated with a mouse to allow real time editing of both components and individual rectangles. Multi-layered devices are easily designed since Saw Draw supports six different masking levels.

BASIC THEORY

In order to conserve disk and memory space a special record format was designed. The record structure is rectangle based. All data is stored in a binary format on disk, eliminating any large memory requirements. The table below describes the record format in terms of its fields.

TABLE 1
SAW DRAW RECORD STRUCTURE

FIELD	TYPE	SIZE	UNITS
Record Type	Integer	2	none
Entries	Integer	2	none
X Coordinate	Long Integer	4	nanometers
Y Coordinate	Long Integer	4	nanometers
Width	Long Integer	4	nanometers
Height	Long Integer	4	nanometers
Angle	Real	4	radians
Cell Name	String	8	none
X Offset	Long Integer	4	nanometers
Y Offset	Long Integer	4	nanometers

The record type may take on values from 0 to 7, thereby defining how that particular record is to be used. A value of 0 specifies a deleted or indexing record. If the record is an index, then the value of entries will be non-zero. Values of 1 through 6 refer to different masking levels or layers of fabrication. A value of 7 flags this record as defining the position at which point another Saw Draw cell is to be included.

Entries are used in all record types. For the cell header, the absolute value of entries defines the number of records in the cell. In the case of an index, entries specify the number of records over which the index is active. For record types 1 through 6, this value determines that record's indexing record number.

Coordinates, x and y, define the lower left-hand corner of the rectangle or cell. For indexing records this defines the lowest left-hand corner of the rectangles being indexed. For subcells these coordinates define position of the cell for plotting purposes.

Width and height give the dimensions of the rectangle. In the case of an indexing record these values define the total dimensions of all records being indexed. For subcells these dimensions define the area of the subcell for plotting purposes.

The angle specifies the rotation of the rectangle in radians. This value is ignored for indexes but may be used to rotate an entire subcell.

The cell name, used only for record types of 7, describes the name of the cell to be included. The x and y coordinates, width, height and angle define the position, dimension and orientation of the cell respectively. An included cell represents a subcell and a level in the cell hierarchy. The base or root cell represents level 0.

Offsets, x and y, are used to offset data in a subcell when it is opened. Because the value of the lower left-hand corner of a cell may change, the x and y coordinates of an included cell record will not sufficiently define a cell's position. When the lower left-hand coordinate of a subcell is changed, the x, y coordinates, width and height of its parent cell are updated using the subcell header and the offsets specified in the parent.

The angle specifies the rotation of the rectangle in radians. This value is ignored for indexes but may be used to rotate an entire subcell.

The cell name, used only for record types of 7, describes the name of the cell to be included. The x and y coordinates, width, height and angle define the position, dimension and orientation of the cell respectively. An included cell represents a subcell and a level in the cell hierarchy. The base or root cell represents level 0.

Offsets, x and y, are used to offset data in a subcell when it is opened. Because the value of the lower left-hand corner of a cell may change, the x and y coordinates of an included cell record will not sufficiently define a cell's position. When the lower left-hand coordinate of a subcell is changed, the x, y coordinates, width and height of its parent cell are updated using the subcell header and the offsets specified in the parent.

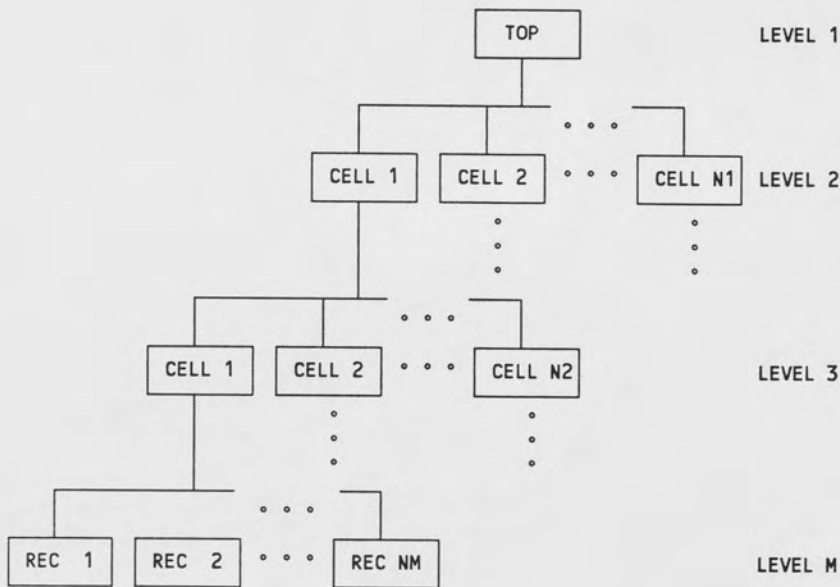


Figure 1. Cell Hierarchy.

If all cells on a single level of hierarchy contain only other cells except for the bottom or Mth level, which contains only rectangles then the total number of virtual rectangles which exist are:

$$total = \prod_{m=1}^M N_m$$

However the total number of records actually stored on disk are:

$$actual = \sum_{m=1}^M total_m$$

Rewriting,

Where $total_m$ refers to the total number of records which exist on the m_{th} level of hierarchy.

$$actual = \sum_{m=1}^M \prod_{j=1}^m N_j$$

If K_m variations of cells exist on each level of hierarchy, then,

$$actual = \sum_{m=1}^M N_m \prod_{j=1}^{m-1} K_j$$

in the limit as K approaches 1 over all j ,

$$actual = \lim_{K_j \rightarrow 1} \sum_{m=1}^M N_m \prod_{j=1}^{m-1} K_j$$

$$actual = \sum_{m=1}^M N_m$$

then taking N_m as a constant for all m ,

$$actual = N \cdot M$$

$$virtual = N^M$$

combining,

$$actual = N \cdot \log_N(virtual)$$

Therefore, the actual number of records stored on disk is proportional to the LOG of the number of virtual records used. These virtual records represent total number of records which would have to be stored in a cell which was not hierarchical in nature.

The number of records which must be accessed in order to plot a single cell at the bottom of the hierarchy may be compared to that of a single level cell in the same manner. The results are:

$$virtualrecords = \prod_{m=1}^M N_m$$

$$recordsread = \sum_{m=1}^M N_m$$

As before, the number of records read is proportional to the log of the total number of virtual records used.

As a more realistic example, The following table depicts the number of records actually stored in a cell under the conditions that all cells on each level are identical but that the number of records in each cell increase by a power of 4 on each level.

TABLE 3
VIRTUAL VERSUS ACTUAL RECORDS STORED

RECORDS PER CELL	LEVEL	LOG BASE 4 VIRTUAL	LOG BASE 4 ACTUAL
0004	1	01.00	1.00
0016	2	03.00	2.16
0064	3	06.00	3.20
0256	4	10.00	4.20
1024	5	15.00	5.21

The equations describing the total actual and virtual records are:

$$virtual = \prod_{m=1}^M 4^m = (4 \cdot 16 \cdot 32 \cdots 4^M)$$

$$actual = \sum_{m=1}^M 4^m = (4 + 16 + 32 \cdots 4^M)$$

To further improve response time of record access, every 100 records have been indexed, where the indexing record defines the area over which following 100 rectangles reside [1]. Indexing permits Saw Draw to make judgements as to whether or not to access the the indexed records which follow. For example, during plotting, if the dimensions of

the index do not fall on or within the graphic window's view port then the records described by that index may be skipped. The position and dimensions of subcells are used in the same way. A cell is only plotted when it falls within the view port. This reduces plot time as well as the time required to search for any records which are to be edited.

Due to the regular pattern of SAW devices, indexed groups of rectangles are typically confined to a small area. The editing capabilities and commands of Saw Draw were designed to enhance this characteristic thereby extending the effectiveness of the indexes. Any time a modification is made to a cell's contents, some reindexing must be performed. Saw Draw has been written to remember the range of records affected by each edit operation. Therefore, only those records affected are reindexed. Deleted records are removed only upon the exiting from or saving of an edited cell.

MODES OF OPERATION

Saw Draw has been divided into two modes of operation. The first is the menu mode, which consists of three pull down menus, the file, output and help menus. The second is the mouse or edit mode.

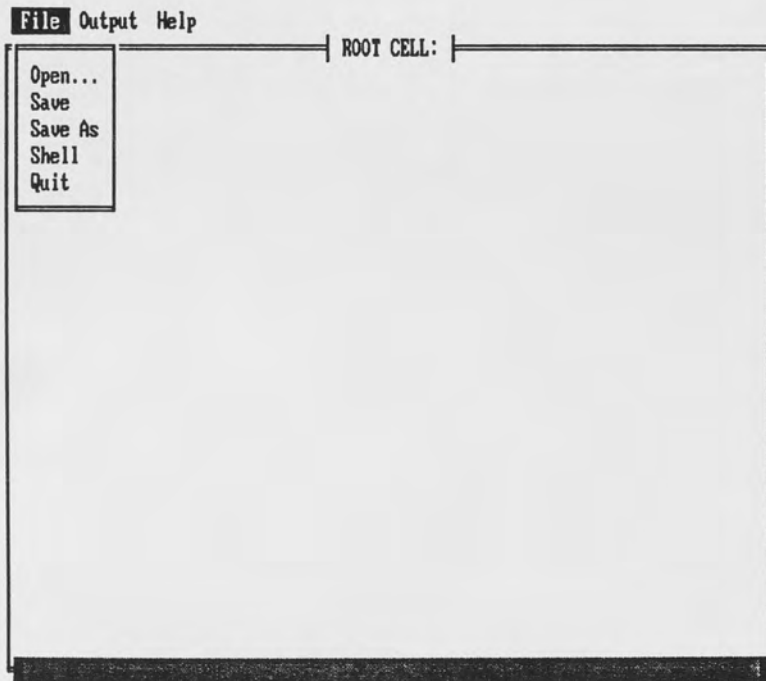


Figure 2. File Menu.

The file menu permits the opening and saving of files. In addition, the shell option permits the user to execute DOS commands. Note that the "Save as" option permits saving the root cell under a new name but does not change the name

of any subcells which exist in the cell hierarchy. When a cell is opened using the "Open ..." option, Saw Draw first prompts for the cell name and then plots the cell in the graphic view port. The two figures which follow use the "SAW" cell as an example.

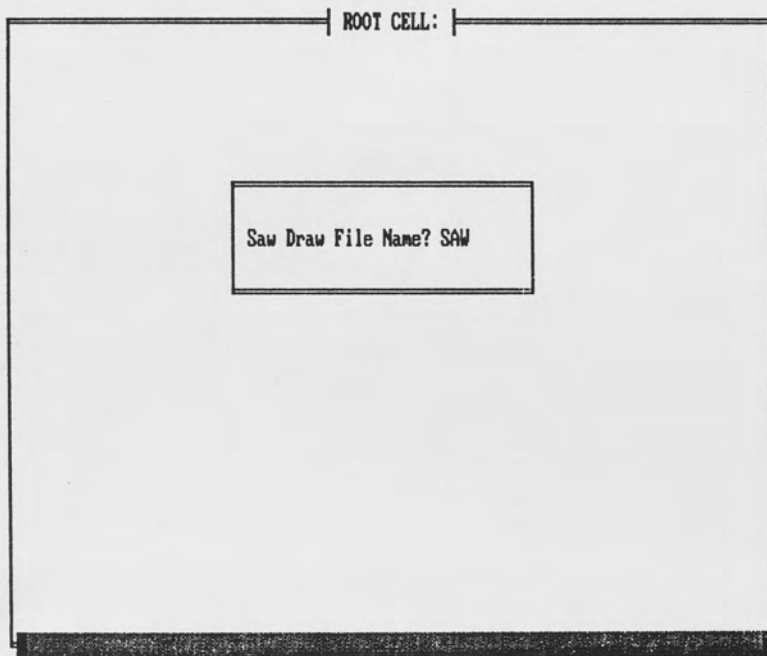


Figure 3. Sample Open of the Cell: SAW.

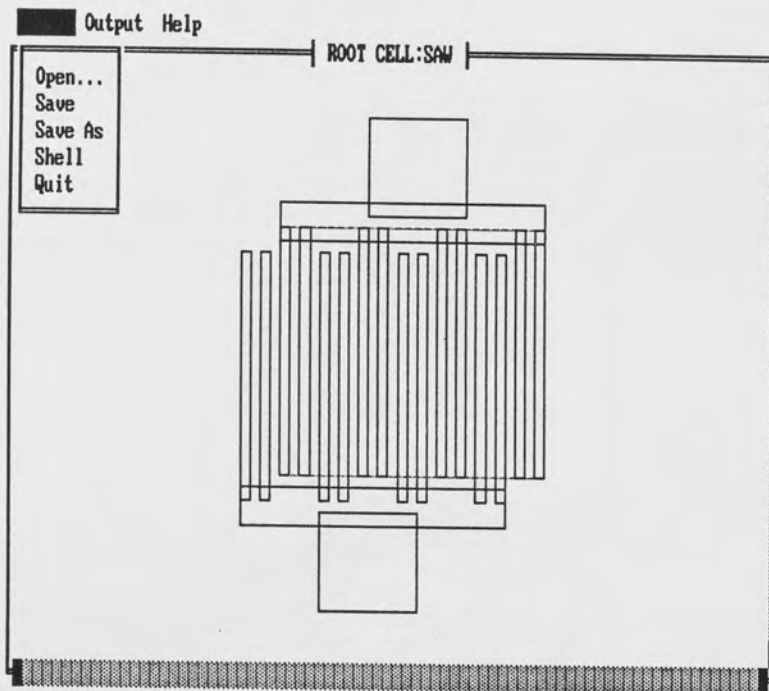


Figure 4. Sample plot of the cell: SAW.

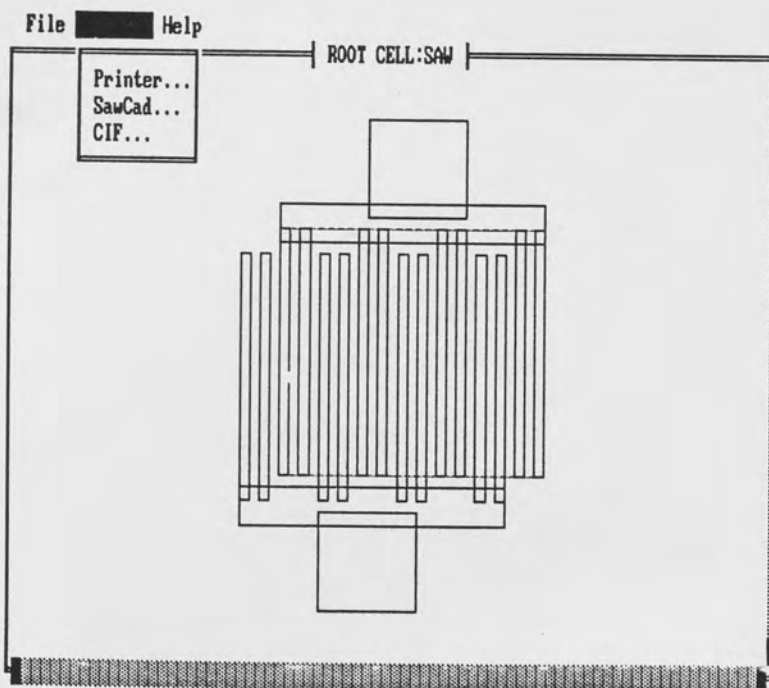


Figure 5. Output Menu.

The output menu permits output of data in Saw Cad structure format, CIF and printer graphics. Saw Cad structure format is an ASCII based format used by the package, Saw Cad, developed at the University of Central Florida. [2]

TABLE 4
SAW CAD RECORD STRUCTURE

FIELD	TYPE
X lower left corner coordinate	Real
Y lower left corner coordinate	Real
Width	Real
Height	Real
Angle in degrees	Real
Repeats (zero is deleted)	Integer
X step	Real
Y step	Real

CIF or California Tech Intermediate Format is an ASCII based format developed for use in VLSI design [5]. This format like Saw Draw will support hierarchical data structures through the definition of cells.

Saw Draw outputs graphic commands in two formats. Either IBM dot matrix compatible format or Hewlett Packard raster graphics for use with HP or compatible laser jets. Graphics may be output directly to the printer or to any other device or disk file. The following menus are used to define printer output.

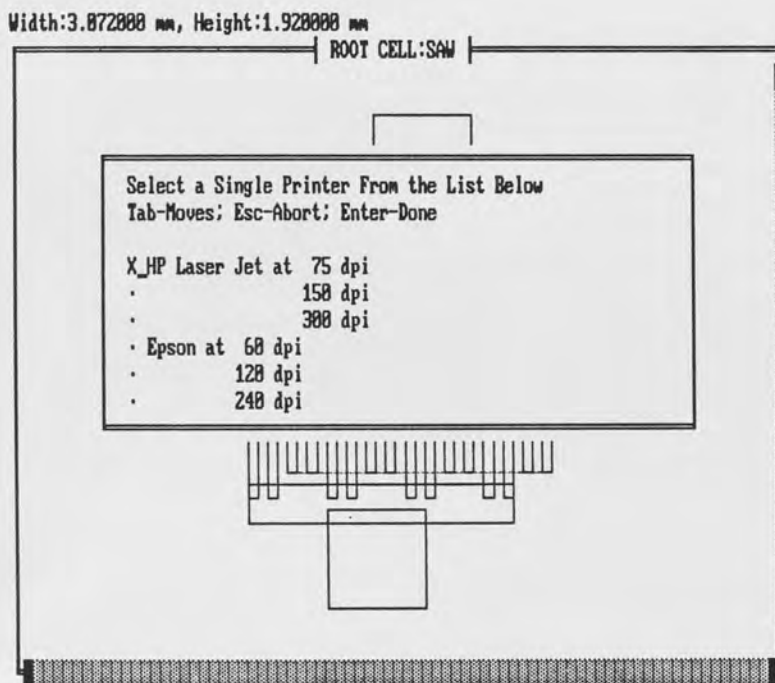


Figure 6. Printer Selection Menu.

Although the resolution of Epson compatible graphics varies in the horizontal direction, the resolution in the vertical direction is fixed at 72 dots per inch. The resolution of the HP laser jets may be varied equally in both directions.

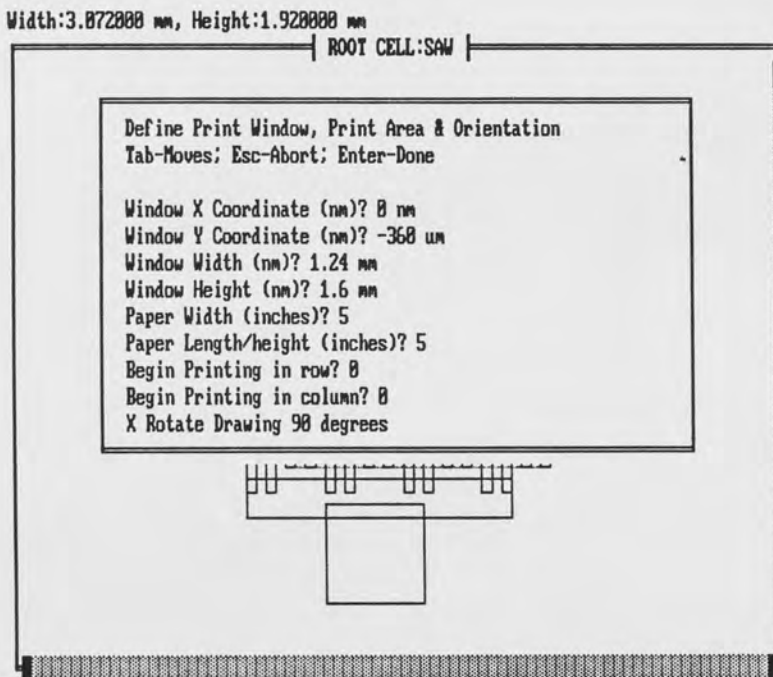


Figure 7. Define Print Area.

Once a printer has been selected, the print window, print area and orientation must be specified. The print window refers to the area of the cell which is to be printed. This window defaults to the x,y position, width and height dimensions of the edit box. The print area represents the printable area on the paper. Note that by specifying a beginning row and column, the print may be positioned anywhere on a page.

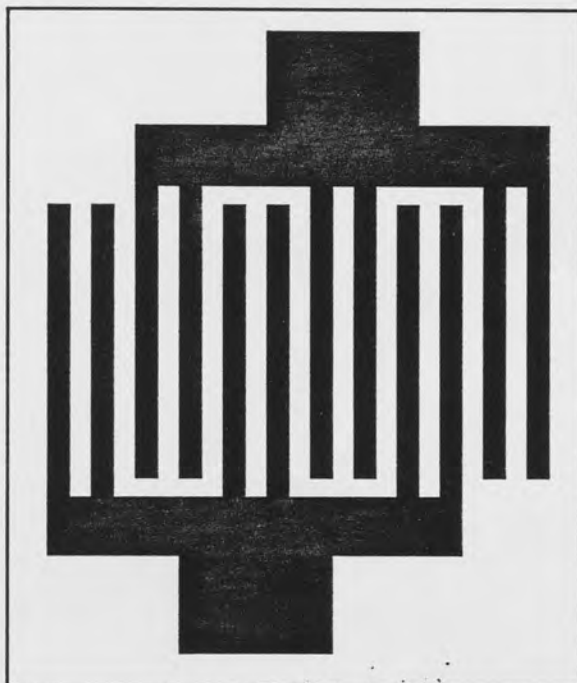


Figure 8. Sample printout of the cell: SAW.

Saw Draw bit maps the cell data so that each bit corresponds to a single dot of output on the printer. The following equation defines the dimensions of a single dot:

$$width = (windowwidth) / (dpi(printwidth))$$

$$height = (windowheight) / (dpi(printheight))$$

The help menu supplies help in five subject areas of general help, menus, keystrokes, input/output and printing. The information for each subject is contained in the files `sd0.hlp`, `sd1.hlp`, `sd2.hlp`, `sd3.hlp` and `sd4.hlp`. The files are ASCII based and may be modified or updated with a text editor.

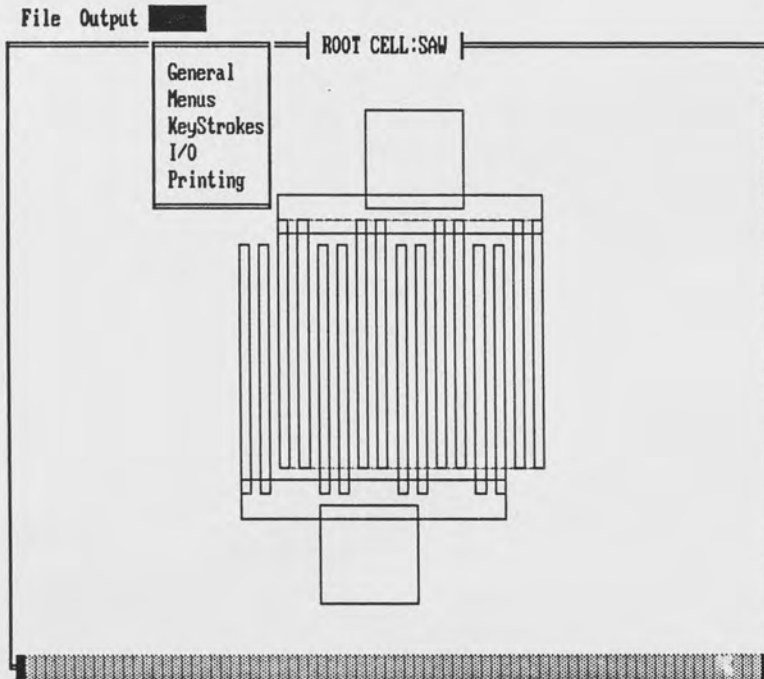


Figure 9. Help Menu.

The edit mode is entered by pressing the escape (Esc) key. In this mode all data in the opened, root, cell and any subcells to that root cell may be modified. When the the edit mode is entered, a dynamic rectangle, referred to as the edit box, surrounds the limits of the root cell. See the following figure.

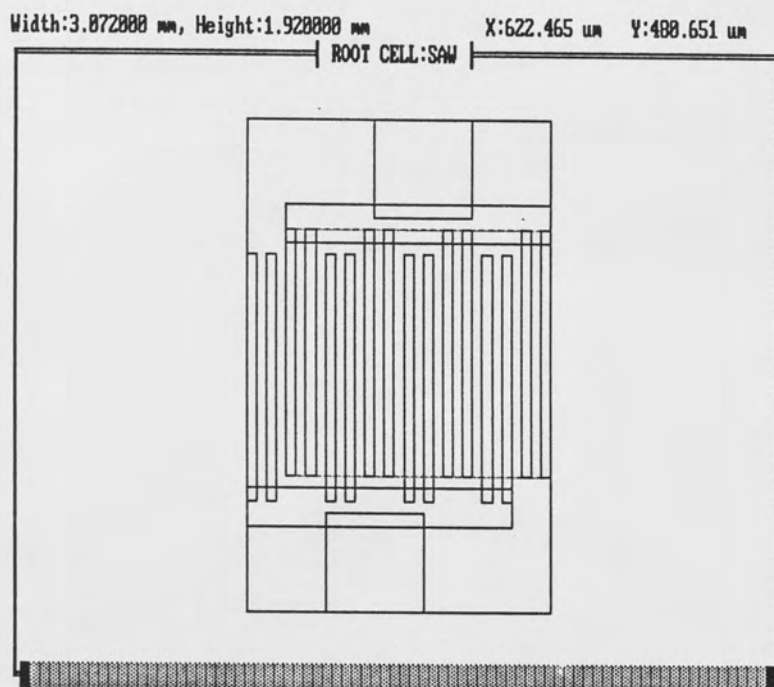


Figure 10. Edit/Mouse mode.

This edit box is manipulated using the mouse. See the next chapter for further information concerning the mouse and its graphical interface.

GRAPHICAL INTERFACE

The graphical interface of Saw Draw and many of the layout algorithms used have been tailored after existing VLSI tools [6], [4]. The graphical interface consists of a single mouse-driven window. The window is manipulated in two ways. First, the mouse may be used to move about the cell being edited by pointing and clicking along the horizontal and vertical scroll bars [6]. For example, if the mouse is pointed at the bottom of the vertical bar on the right-hand side of the screen, and the first button is pressed the window will then shift down to the bottom of the cell.

The second method of manipulating the window is through a set of keystroke commands. These commands may be used to zoom and scroll. See Table 3 for a command summary.

The edit box is the primary tool used in the graphics window. The lower left corner of the edit box may be set by pointing the mouse and clicking the left button. The right button may be used in the same way to set the upper right corner of the edit box. When a cell is first opened, the edit box defaults to the dimensions and coordinates of the cell being edited.

TABLE 5
WINDOWING COMMANDS

KEYSTROKE	NAME	DESCRIPTION
D	Draw	Clears and plots all data within the current window.
O	On Screen	Permits scrolling around the screen in user specified directions and distances.
V	View	View the entire edit cell.
Z	Zoom	Zoom to the dimensions and
Cntl+Z	Zoom Factor	coordinates of the edit box. Zoom by a user specified factor.

Draw Function (D Key Stroke)

The D keystroke is used to redraw the screen. Under certain conditions, the screen may not be fully displayed.

For instance, after a SHELL from the file menu, all screen contents are lost, and it is convenient to redraw the screen using the current view port.

Scroll on Screen Function (O Key Stroke)

In order to shift the view port to a new position, the O keystroke may be used to scroll on the screen. After selecting a direction and the number of screens to move, Saw Draw will adjust the view port. The number of screens may take on fractional values.

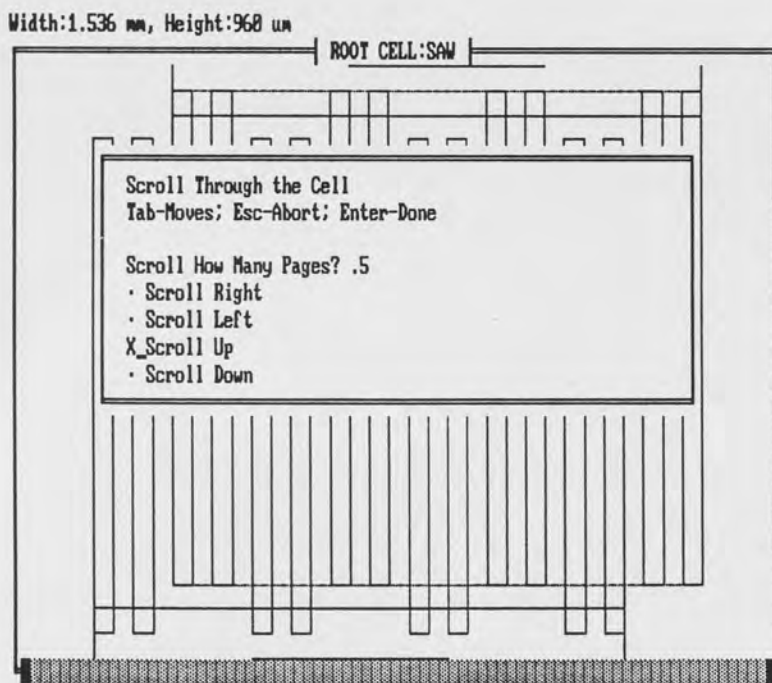


Figure 11. Scroll on Screen Prompts.

The following figure demonstrates the resulting action taken by Saw Draw for the inputs supplied in the previous figure.

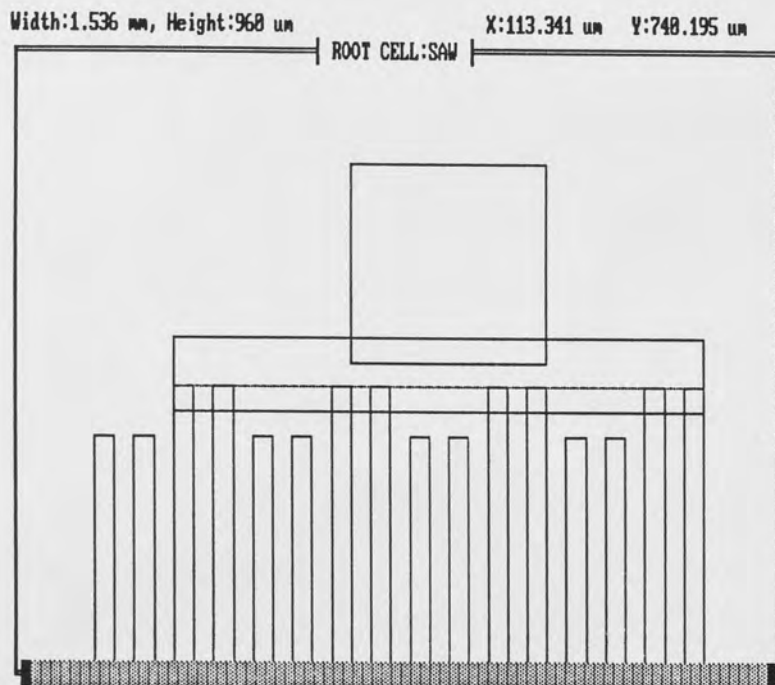


Figure 12. Scroll on Screen Results.

Zoom Functions (Z & Cntl+Z Keystroke)

Two commands exist to permit zooming which adjusts the view port's resolution. The Z keystroke adopts the dimensions and coordinates of the edit box as those of the new view port. In order to zoom to the middle of the view port by a particular factor, the Cntl+Z keystroke is used. The following figure demonstrates how the Z keystroke works. Once the edit box has been placed around the upper right corner of the SAW structure, the Z key is pressed.

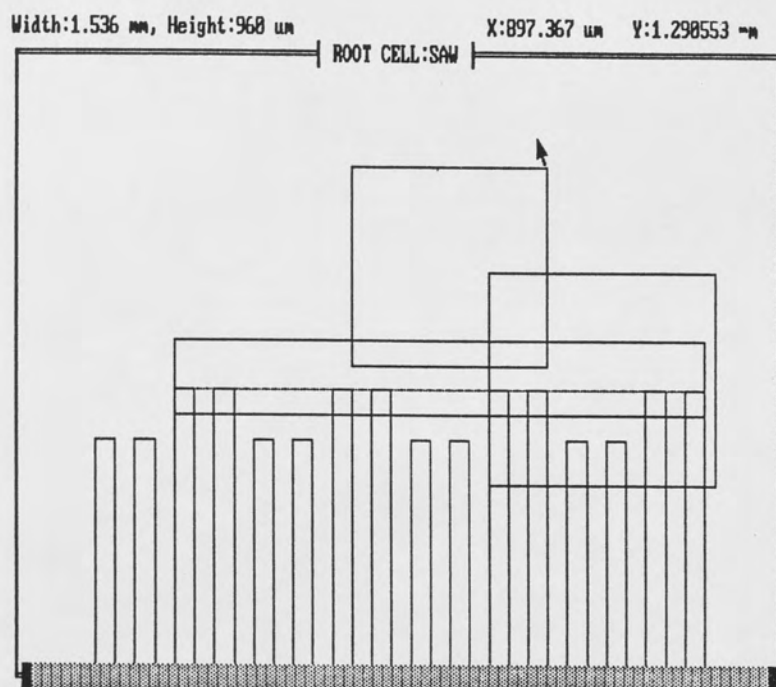


Figure 13. Z Keystroke (Zoom) Example.

The following figure displays the result of the zoom operation.

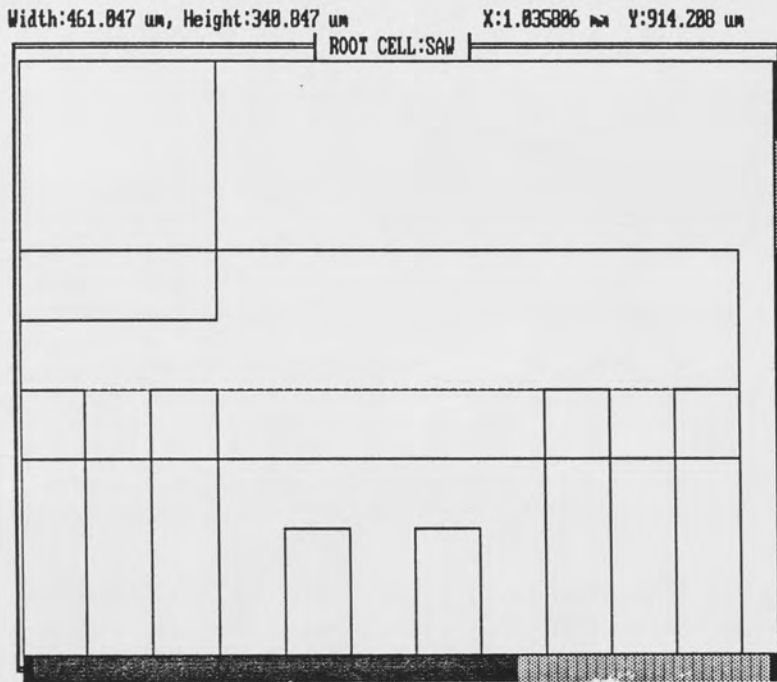


Figure 14. Z Keystroke Results.

In order to zoom to the center of the screen and maintain the correct aspect ratio, the Cntl+Z keystroke may be used. Pressing Cntl+Z results in Saw Draw's prompting for a zoom factor. In the following figure a factor of 2 has been entered.

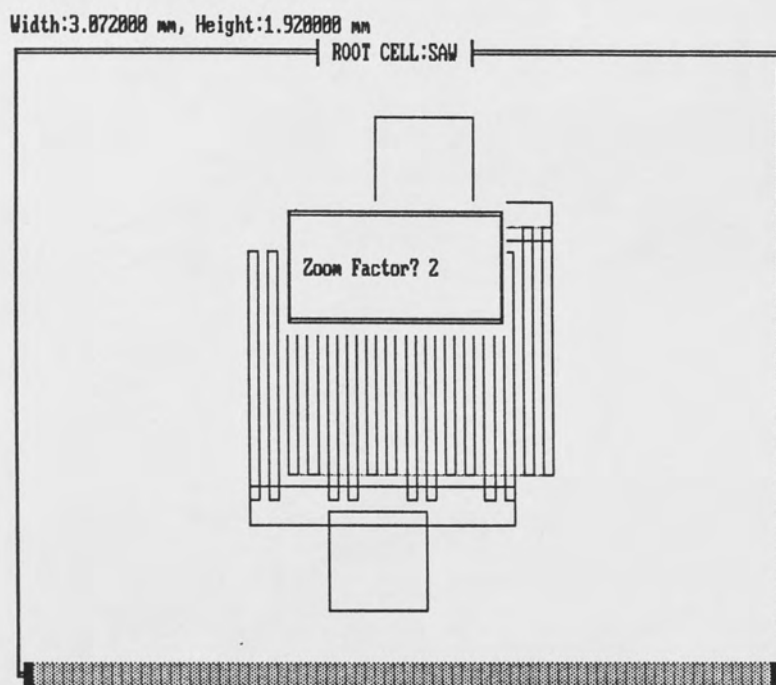


Figure 15. Cntl+Z (Zoom) Keystroke Example.

After entering the zoom factor, Saw Draw adjusts the view port accordingly as shown in the next figure.

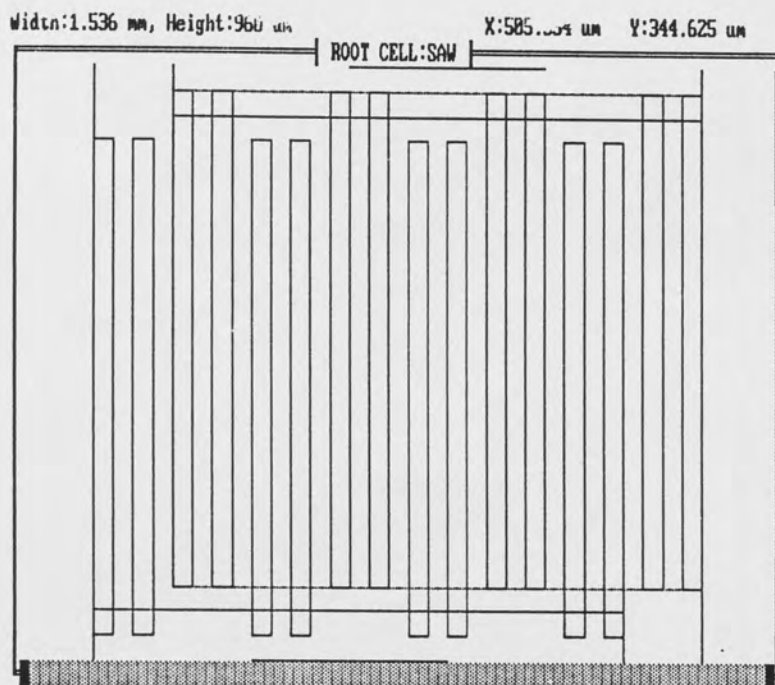


Figure 16. Cntl+Z Keystroke Results.

View Function (V Keystroke)

The V keystroke may be used in order to view the entire cell again. By pressing the V key, Saw Draw responds by returning to the default view port and centers the cell as shown in the following figure.

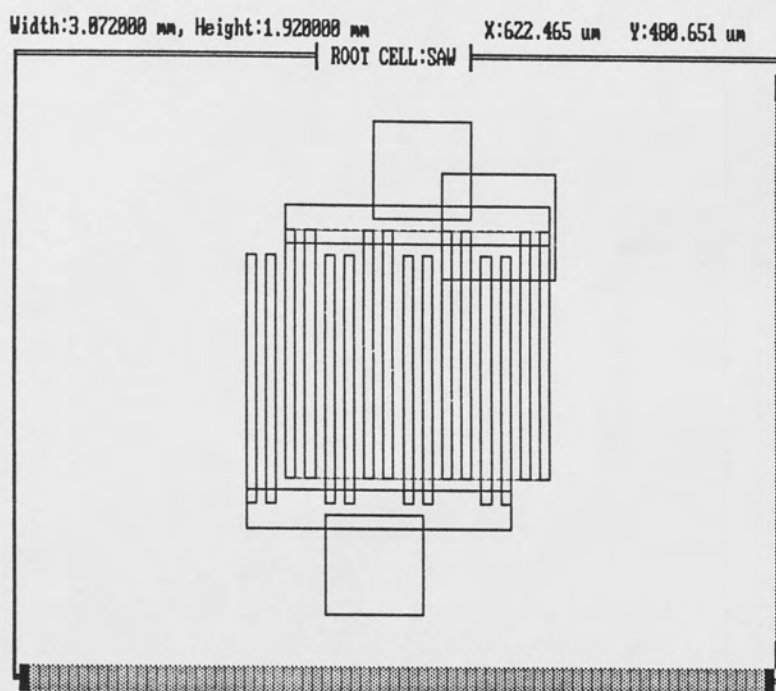


Figure 17. V Keystroke (View) Example.

EDITING ALGORITHMS AND COMMANDS

Editing is performed through the combined use of the edit box and keystroke commands. The major function of the edit box is to select rectangles and subcells to be edited. By placing the edit box over the desired rectangles and pressing "s", those rectangles may be selected and highlighted. See the discussion on the S keystroke for more information. Once a selection has been made, a keystroke command may be issued. Many of these commands will respond with a menu so that more information may be supplied.

These menus consist of three types of prompts: switches, numeric with units [3] and unitless numeric. Switches provide a convenient way of answering yes (X) or no (.) to a question by simply pressing the space bar. The other two prompts are identical except that one will accept units of centimeters, millimeters, micrometers, nanometers or wavelengths. The unitless prompts are used for scales and input in units other than meters such as inches and degrees.

TABLE 6
EDIT COMMANDS

KEY STROKE	NAME	DESCRIPTION
A	array	array selections
B	box	set edit box position
C	copy	copy selections
E	hard expand	expand selected subcells
Cntl+E	soft expand	expand to a specified level
F	get file	get a Saw Cad file
Cntl+F	get cell	include contents of a cell
G	grid	create a grid
H	change layer	change the selection's layer
K	kill	kill/delete selections
L	set layer	set the default layer
M	move	move selections
Alt+M	merge cell	merge the contents of a cell
P	paint	paint/add the edit box
R	step & repeat	step and repeat selections
S	select	select rectangles & subcells
Cntl+S	edit	edit a selected subcell
Alt+S	unedit	back out of the edit path
T	text	make a text label
U	unexpand	unexpand a subcell's contents
W	wave lengths	set the wavelength magnitude
X	x mirror	mirror image of selections
Y	y mirror	mirror image of selections

Array Function (A Keystroke)

The A keystroke, array function permits a group of selected records to be copied into an arrayed or checker-board pattern. Inputs to this function are x, y coordinates, arrays in the y direction and arrays in the x direction. A switch is also provided to allow the x and y coordinates to be used as offsets from the edit box. The x and y coordinates default to the lower left-hand corner of the edit box. If the offset switch has been previously set, then the x and y coordinates take on the offset values between the lower left-hand corners of the edit box and the selections. The array is treated as base 0, that is, if the x and y arrays are both entered as 1, then the actual array will be a 2 by 2.

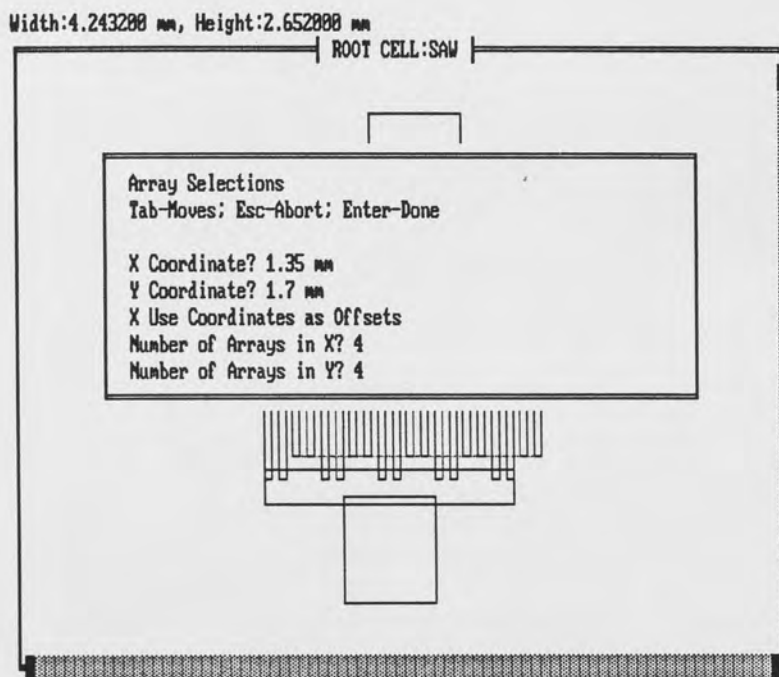


Figure 18. Array Selections Prompts.

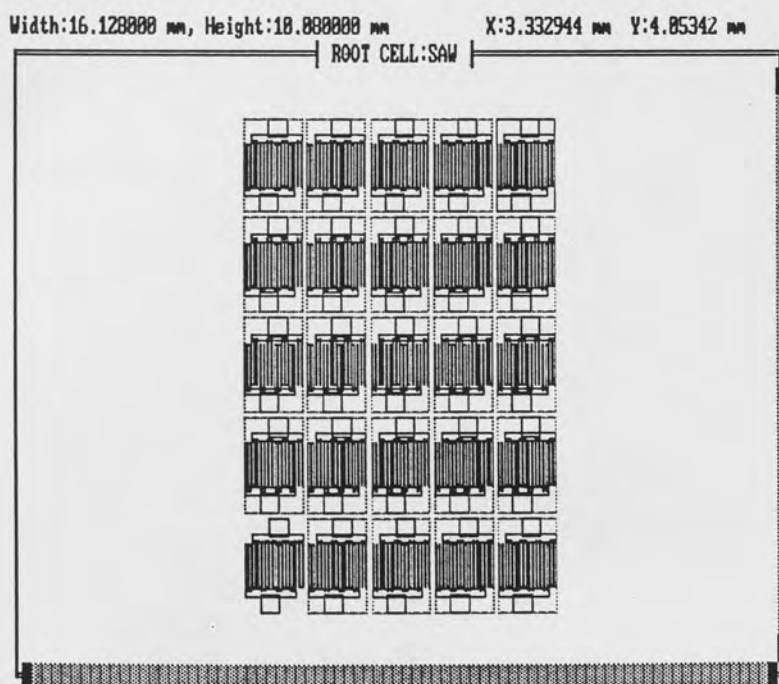


Figure 19. Array Selections Results.

Box Function (B Keystroke)

The B keystroke, box function, allows changes to the edit box's position, dimension, and orientation. This function becomes very useful when running Saw Draw without a mouse. When a grid value has been specified, this function allows positioning of the edit box between grid marks.



Figure 20. Set Edit Box Position.

Copy Function (C Keystroke)

The C keystroke, copy function, copies the selected section of data. Inputs to this function are x, y coordinates and rotation angle. As with the array keystroke, a switch exists to permit entry of offsets rather than input in absolute coordinates. The x and y coordinates

default to the lower left-hand corner of the edit box. If the offset switch has been previously set, then the x and y coordinates take on the offset values between the lower left-hand corners of the edit box and the selections.

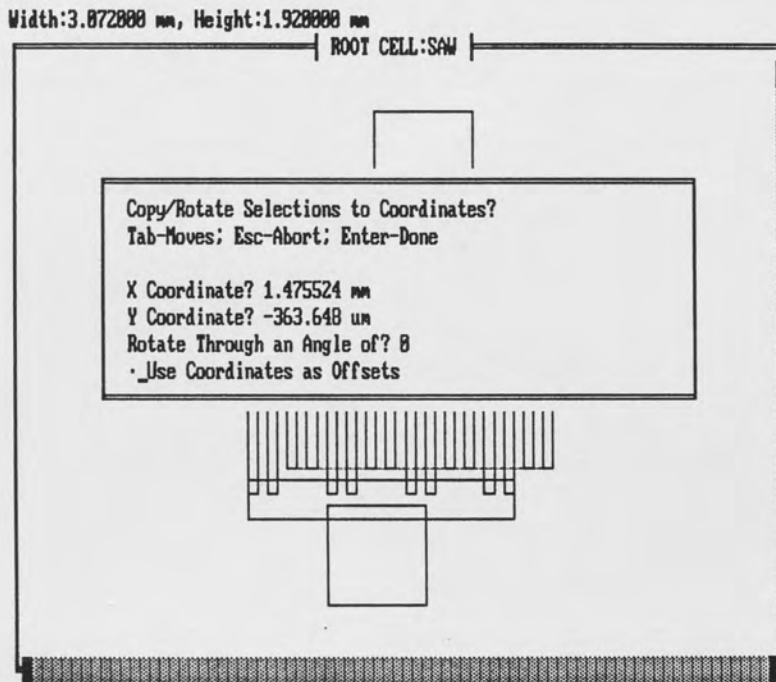


Figure 21. Copy Function.

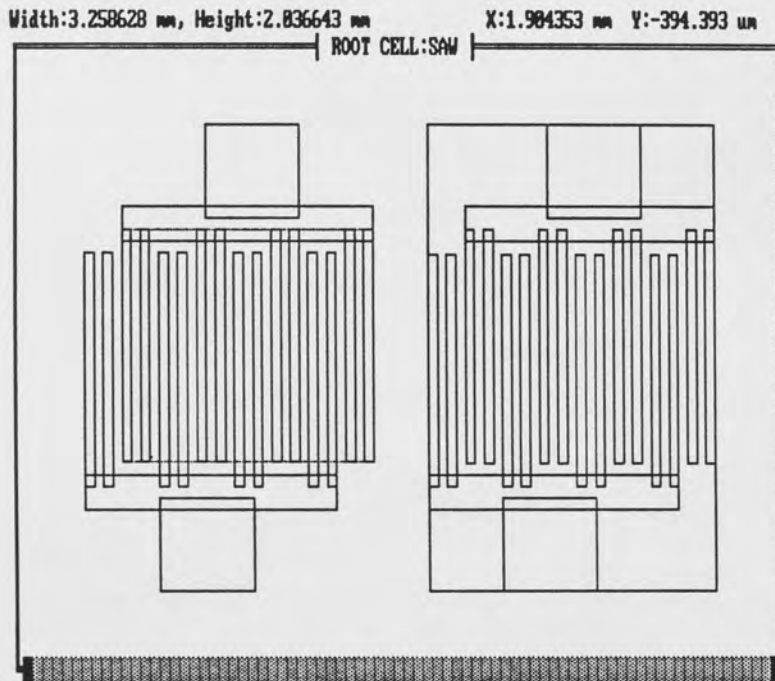


Figure 22. Copy Results.

Hard Expand Function (E Keystroke)

The E keystroke, hard expand function, sets the expand flag on all selected subcells and displays their contents. Any time these cells fall within the view port, their contents will be displayed. Any rectangles displayed which are not records of the cell being edited are plotted as dashed lines. There are no inputs to this function.

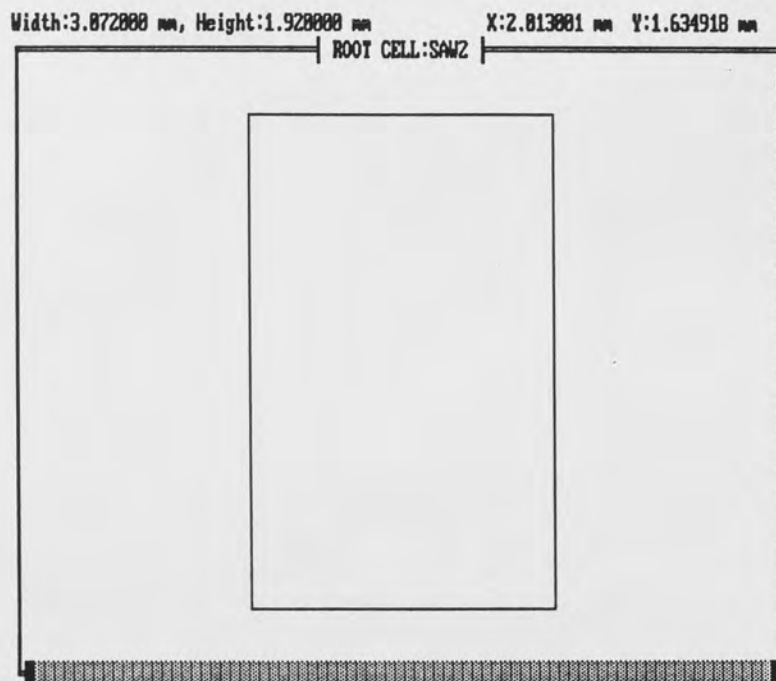


Figure 23. Expand Selected Subcell Function.

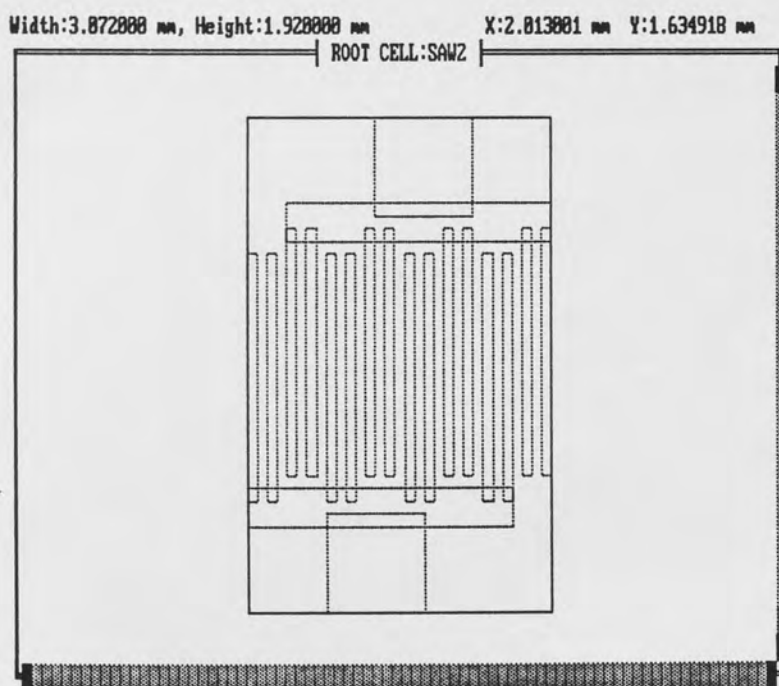


Figure 24. Expand Selected Subcell Results.

Soft Expand Function (Cntl+E Keystroke)

The Cntl+E keystroke, soft expand function, sets the hierarchical level to which all subcells falling within the view port will be expanded. For example, if the expand level is set to 1 then all subcells in the root or bottom cell will be expanded, but any subcells within these cells will be expanded only if their hard expand flag has been set.

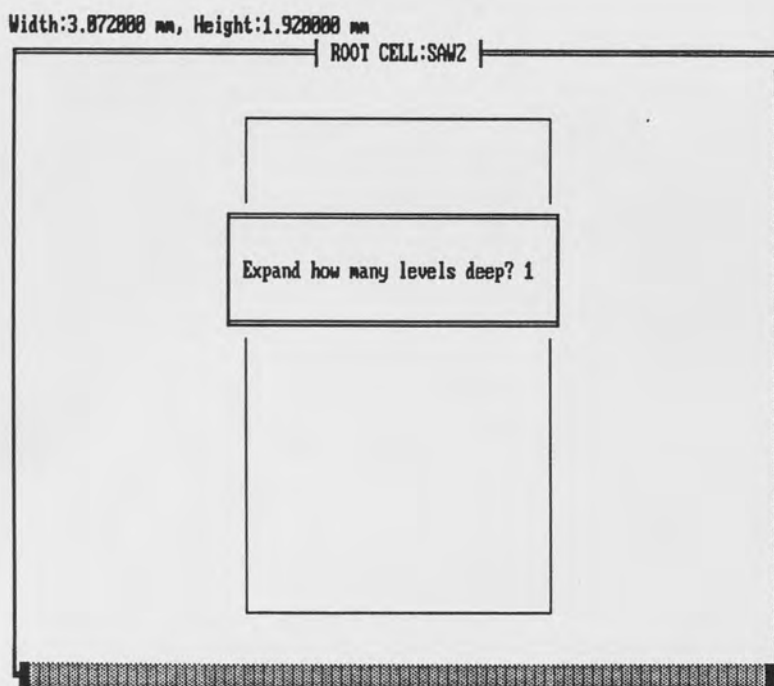


Figure 25. Expand Level Function.

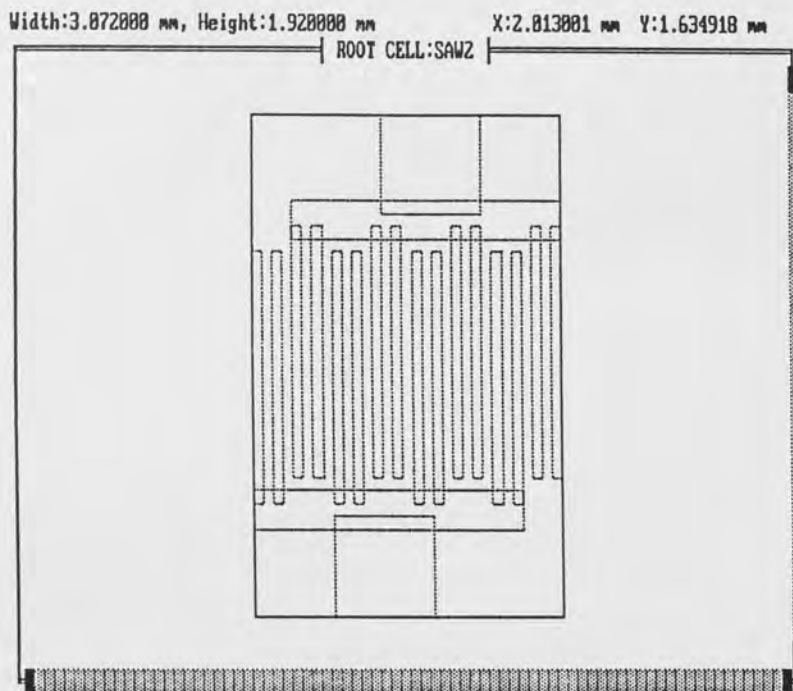


Figure 26. Expand Level Results.

Get File Function (F Keystroke)

The F keystroke, get file function, imports a Saw Cad structure file. Inputs to this function consist of two scales, layer number and a file name. The first scale is used to scale the incoming data to dimensions of nanometers. The second is used to scale incoming angles to degrees. No positioning is performed, that is the incoming data are offset in no way. Once the import is complete, all of the new rectangles are selected so that they may be easily moved, copied or rescaled. The following three figures demonstrate the steps taken to get a Saw Cad file. Note that the cell is empty prior to the F keystroke.

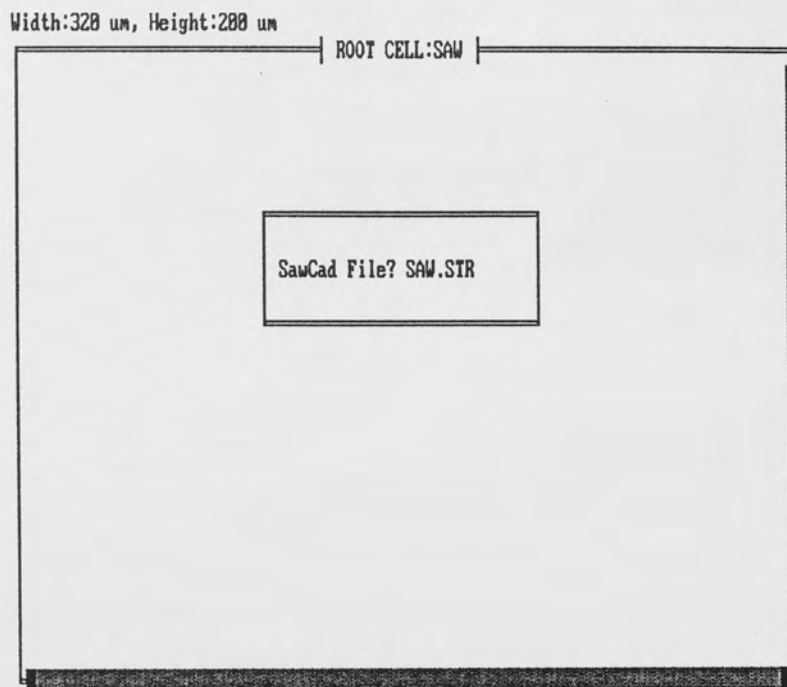


Figure 27. Get File Function.

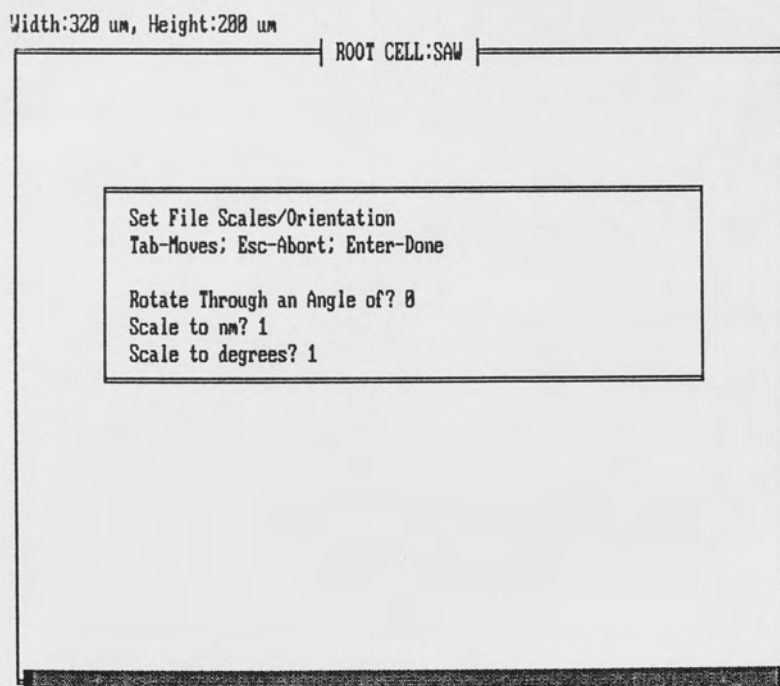


Figure 28. Get File Prompts.

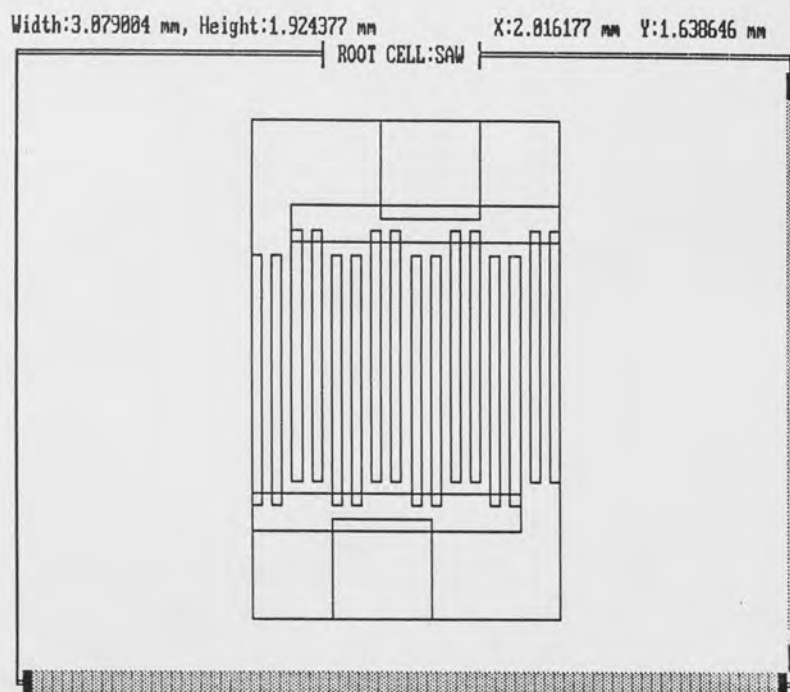


Figure 29. Get File Results.

Include Cell Function (Cntl+F Keystroke)

The Cntl+F keystroke, get cell function, includes a Saw Draw cell in the current edit cell. Inputs to this function are cell name, position and orientation. The position defaults to the current position of the edit box. The new subcell's contents are displayed only if the expand level is greater than or equal to the level of the new subcell. Caution should be observed to prevent recursion in the cell hierarchy. That is, one cell must not be allowed to call itself through any path in the hierarchy.

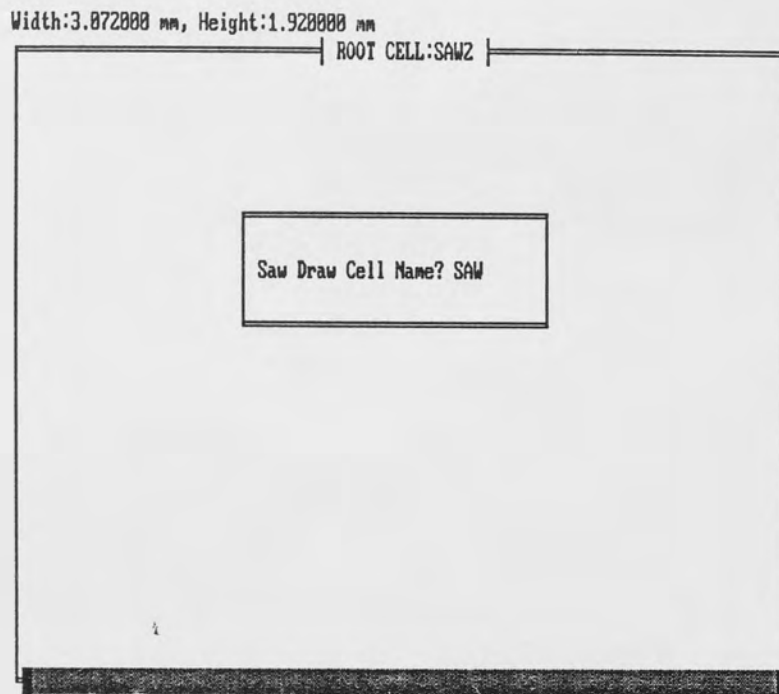


Figure 30. Include Cell Function.

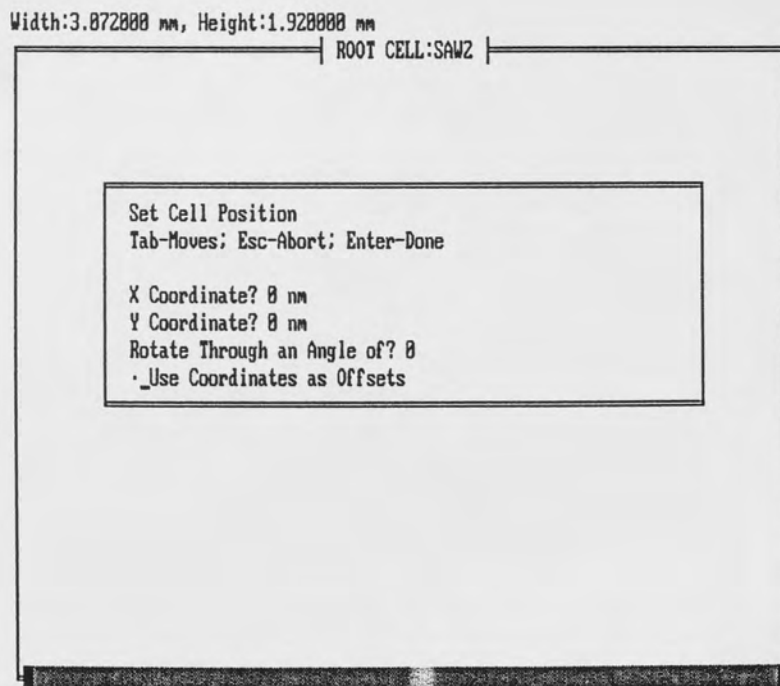


Figure 31. Include Cell Function Prompts.

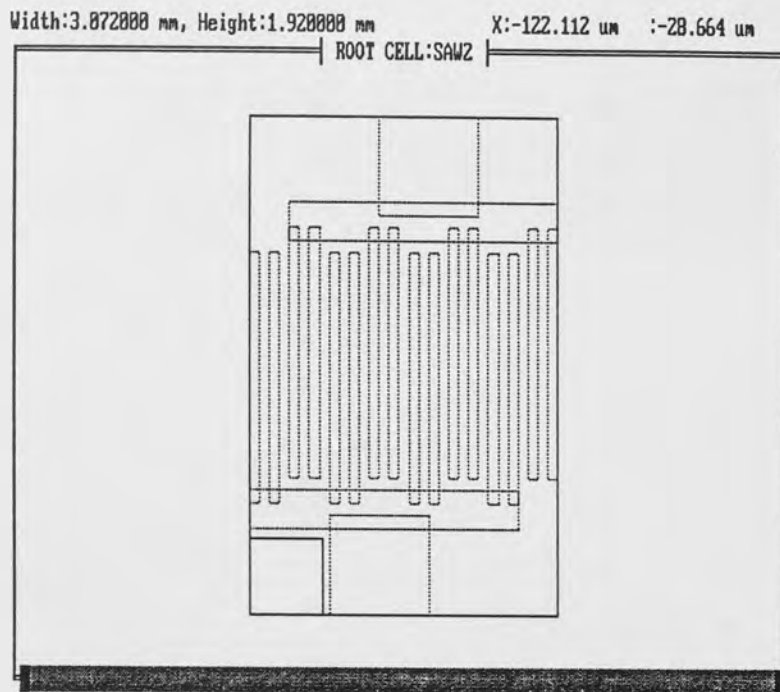


Figure 32. Include Cell Results.

Grid Function (G Keystroke)

The G keystroke, grid function, permits the specification of a grid. The inputs consist of the grid origin and steps in the x and y directions. Specifying grid steps of 0 switches the grid off. Grid axes are drawn through the grid origin and points are plotted at each x and y step. After the grid has been specified, the mouse will position the edit box exactly on the grid marks. In order to place the edit box between grid marks, the B keystroke may be used.

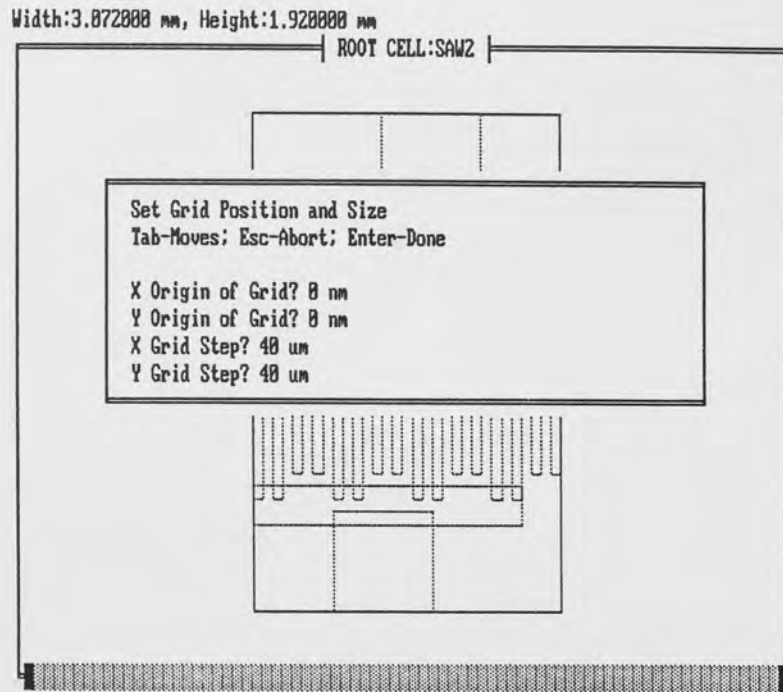


Figure 33. Grid Function.

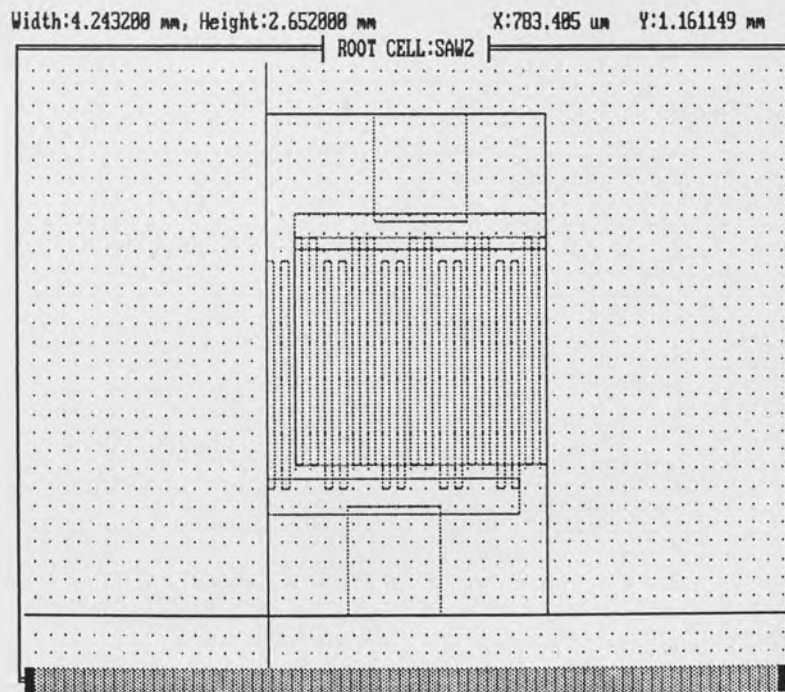


Figure 34. Grid Function Results.

Change Layer Function (H Keystroke)

The H keystroke, change layer function, is used to change the layer number of the selected rectangles. Selected subcells are not affected. The new layer may take on a value of 1 through 6.

Kill Function (K Keystroke)

The K keystroke, kill function, is used to delete all selections from the edit cell. Once selections have been deleted, they are lost and cannot be recovered. There are no inputs to this function.

Set Layer Function (L Keystroke)

The L keystroke, set layer function, is used to change the default layer of Saw Draw. Legal layers are 1 through 6. The layer is used to create new rectangles as well as for output of Saw Cad structure data.

Move/Rotate/Scale Function (M Keystroke)

The M keystroke, move/rotate/scale function, is used to move, rotate and scale the selections. This function permits independent scaling in x and y directions. Inputs to the function are x and y destination coordinates, rotation angle, x scale and y scale. The x and y scale prompts display the scale values required to scale the selections so that they exactly fill the dimensions of the edit box [3]. The x and y coordinates default to the lower

left-hand corner of the edit box. If the offset switch has been previously set, then the x and y coordinates take on the offset values between the lower left-hand corners of the edit box and the selections. The rotation angle defaults to the rotation of the edit box. The scales have no effect on subcell definitions.

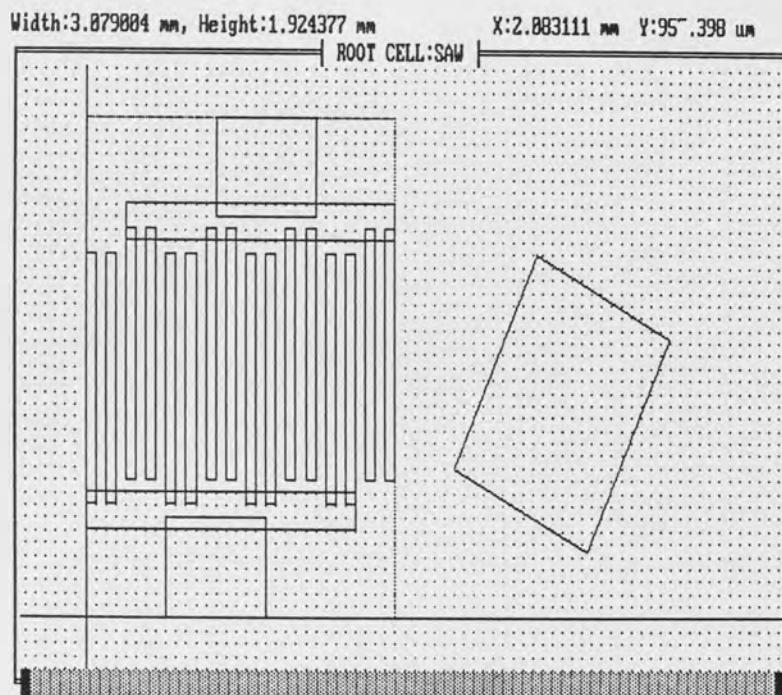


Figure 35. Move/Scale/Rotate function.

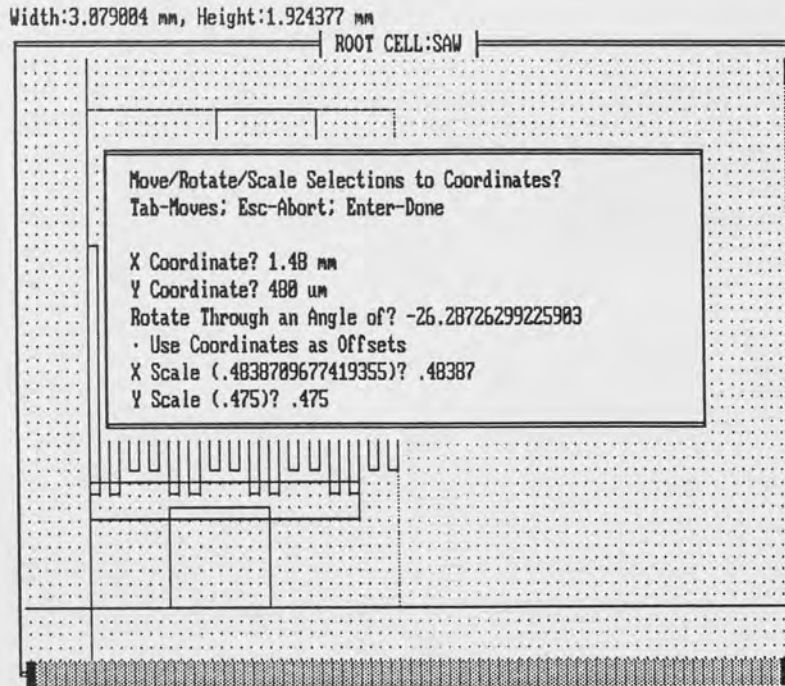


Figure 36. Move/Rotate/Scale Prompts.

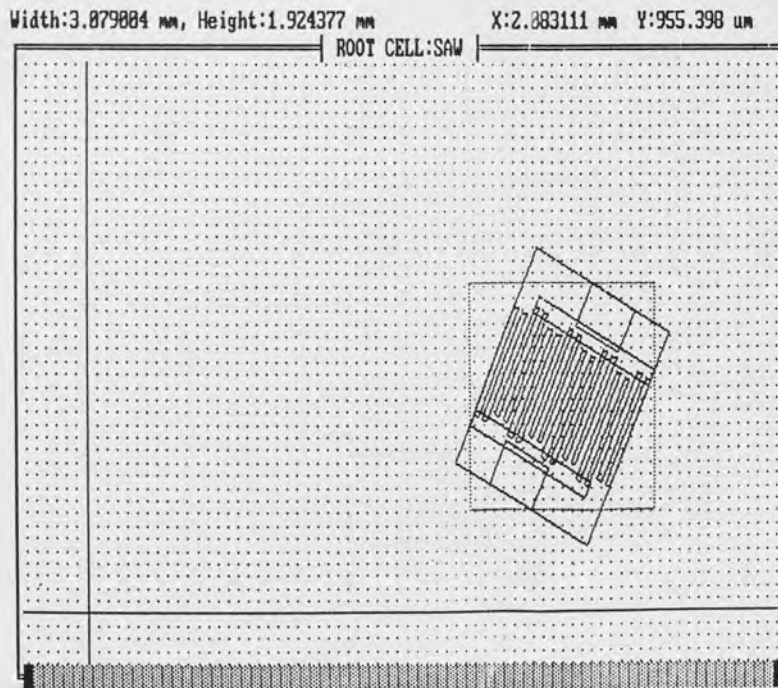


Figure 37. Move/Rotate/Scale Results.

Merge Cell Function (Alt+M Keystroke)

The Cntl+M keystroke, merge function, is used to merge data from one cell into the edit cell. Caution should be observed to prevent recursion in the cell hierarchy. That is one cell must not be allowed to call itself through any path in the hierarchy. The inputs include source cell name, position and orientation.

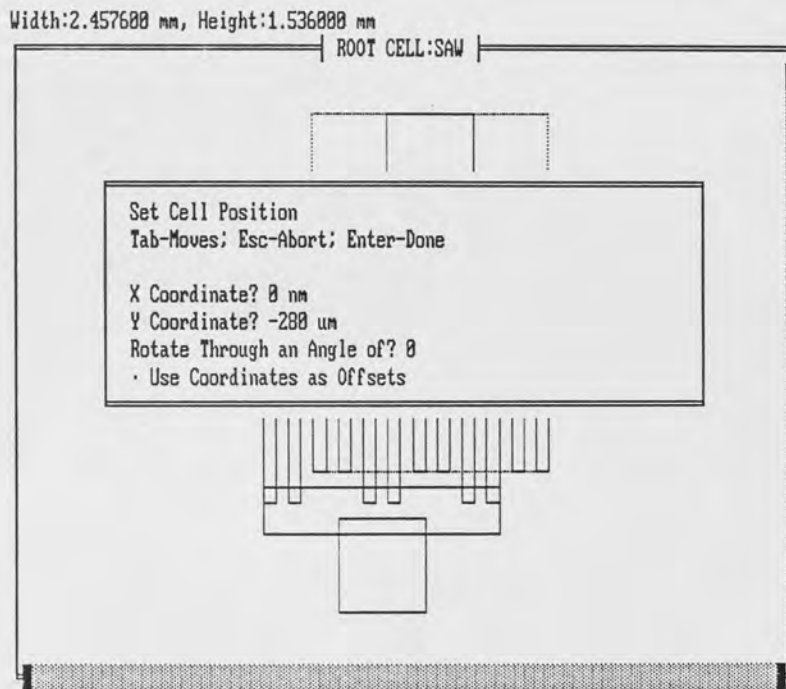


Figure 38. Merge Cell Function Prompts.

Paint/Add Function (P Keystroke)

The P keystroke, paint/add function, is used to create a new rectangle at the position of and dimensions of the edit box. The layer of the new rectangle corresponds to the default layer number as set using the L and H keystrokes.

Rectangles created using this function are not indexed. The B and G keystrokes may be used to gain accuracy in placement of the edit box.

Step and Repeat Function (R Keystroke)

The R keystroke, step and repeat function, is used to step and repeat all selections [2]. Inputs to this function are x, y steps, delta width, delta height, delta theta and repetitions. The x and y steps correspond to the offset between the lower left-hand corner of the edit box and the lower left-hand corner of the selections. The delta width and height correspond to the difference of dimensions between the edit box and selections. If the x and y steps are non-zero then delta theta defaults to 0. Otherwise, delta theta defaults to the value in degrees which will produce a continuous sweep of the data with the origin of rotation corresponding to the lower left-hand corner of the selections. This sweeping property allows Saw Draw to create smooth arcs which may be used to replace sharp corners.

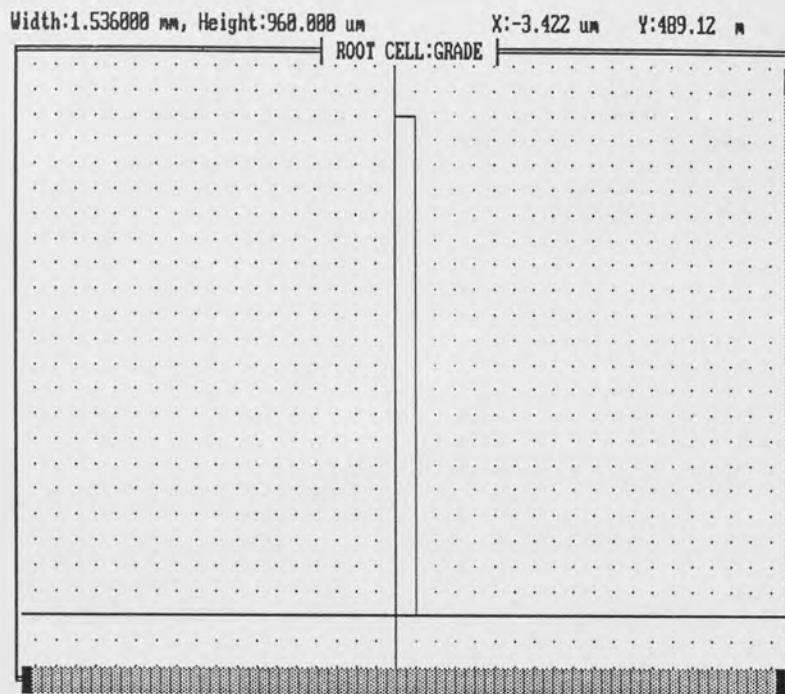


Figure 39. Step and Repeat Function.

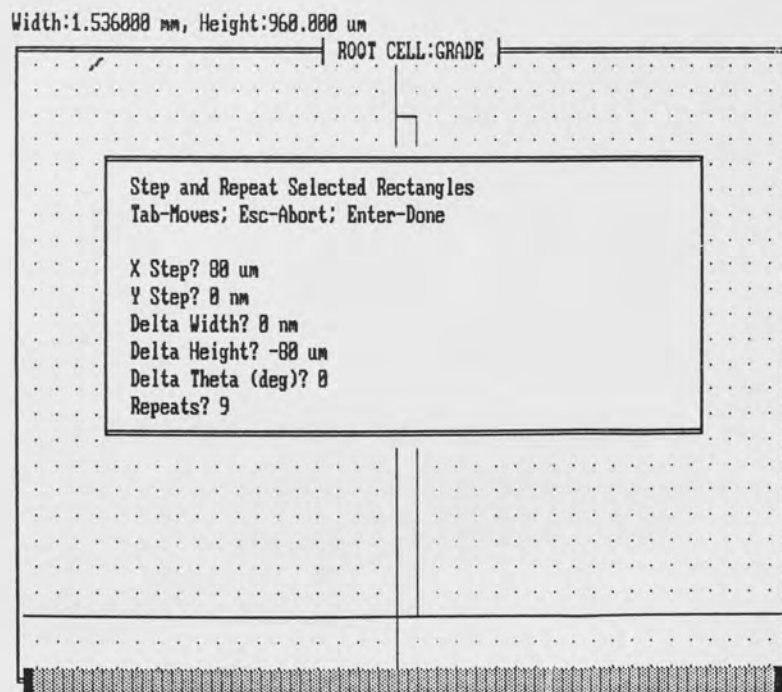


Figure 40. Step and Repeat Prompts.

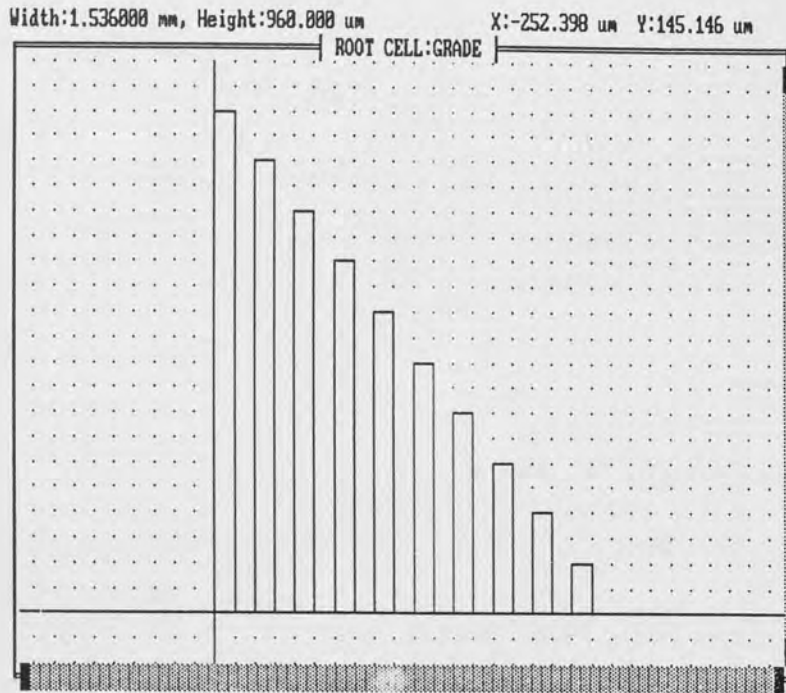


Figure 41. Step and Repeat Results.

Select Function (S Keystroke)

The S keystroke, select function, is used to select subcells and rectangles for editing. All subcells and rectangles which are elements of the edit cell and fall under the edit box will be selected. The selected records are then highlighted on the screen. Using the + keystroke permits new selections to be added to the current set of selections. There are no inputs to this function.

Edit Cell Function (Cntl+S Keystroke)

The Cntl+S keystroke, edit cell function, is used to edit a selected cell. If more than one cell has been selected, then Saw Draw prompts as to whether or not to edit

each cell one at a time. After selecting an edit cell, Saw Draw redraws the screen plotting the new edit cell with solid lines and all other cells with dashed lines. The "Save As" option in the main menu has no effect on a subcell which is edited in this manner.

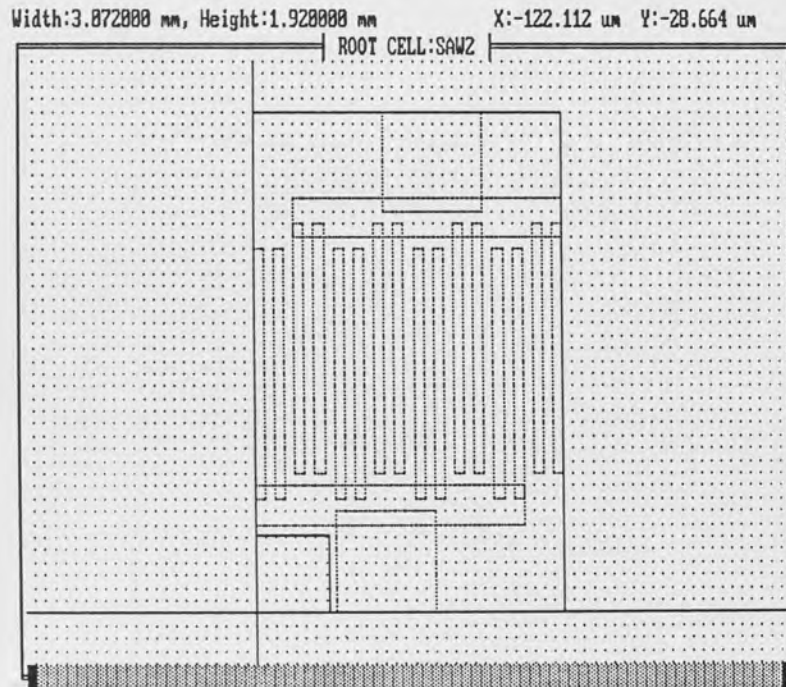


Figure 42. Edit Cell Function.

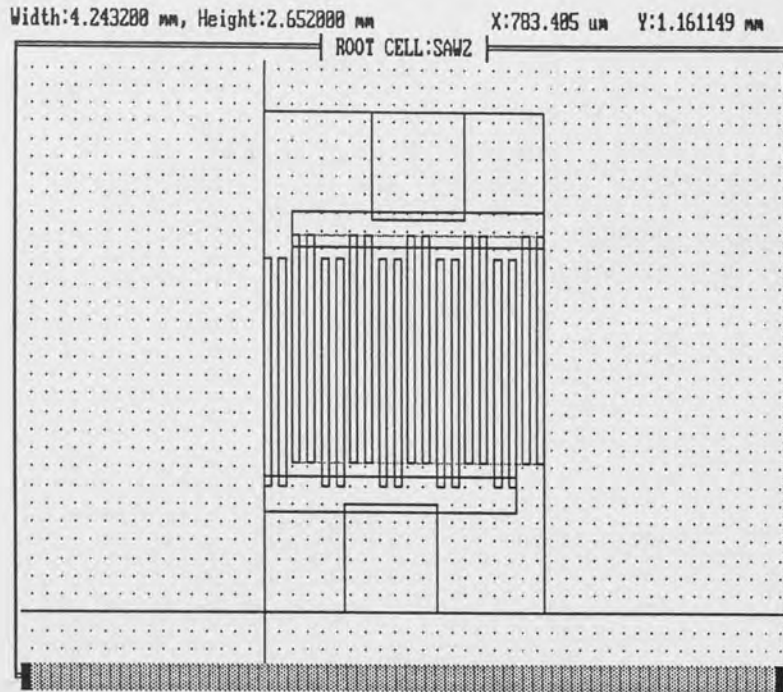


Figure 43. Edit Cell Results.

Unedit Cell Function (Alt+S Keystroke)

The Alt+S keystroke, unedit cell function, is used to back out of the edit path. Before returning to the parent cell, Saw Draw reindexes and removes any deleted records. There are no inputs to the function.

Text/Label Function (T Keystroke)

The T keystroke, text/label function, is used to create labels in the form of text. Each letter, number or symbol is made up of an 8 by 8 pixel pattern. All ASCII symbols of decimal value 32 to 128 are supported. Inputs to this

function are position, orientation and pixel size in micrometers. The position and orientation default to those of the edit box.

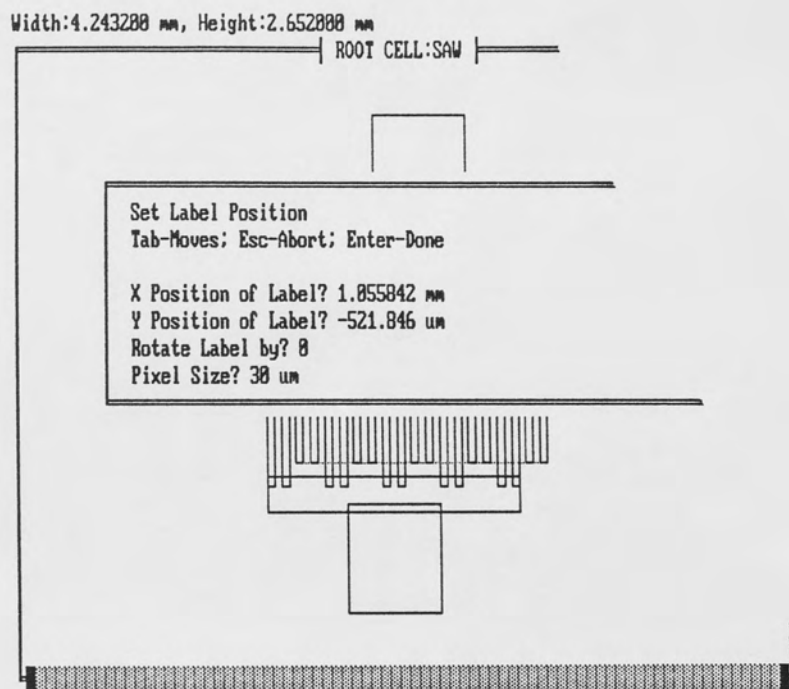


Figure 44. Text/Label Function Prompts.

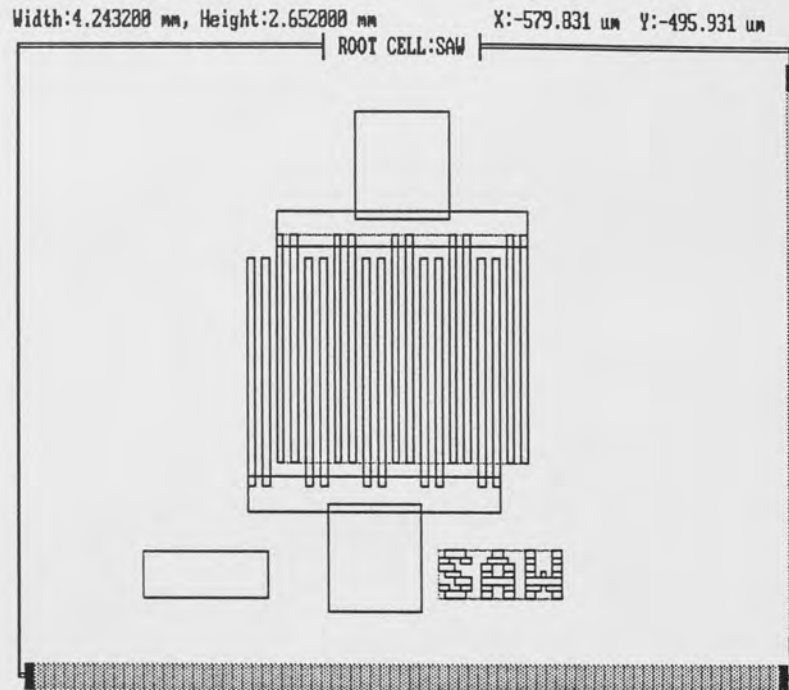


Figure 45. Text/Label Results.

Unexpand Function (U Keystroke)

The U keystroke, unexpand function, is used to reverse a hard expand. The expand flag of all selected cells is reset and the screen is redrawn. There are no inputs to this function.

Wavelength Function (W Keystroke)

The W keystroke, wavelength function, is used to set the value of wavelengths creating a new unit of 'wl' [3]. This unit may be used for any input of distance which normally uses units of meters.

X Mirror Image Function (X Keystroke)

The X keystroke, x mirror image function, is used to take the mirror image of the selections about an x axis. The mirror image x axis is that of the selection's center x value. There are no inputs to this function.

Y Mirror Image Function (Y Keystroke)

The Y keystroke, y mirror image function, is used to take the mirror image of the selections about a y axis. The mirror image y axis is that of the selection's center y value. There are no inputs to this function.

EXAMPLE

This example demonstrates how an unweighted SAW device might be designed using the hierarchical structure of Saw Draw. First the unweighted transducer is designed, then two matching networks which are used to reduce second order effects. For this example, the actual electrical characteristics will be ignored so that the editing capabilities of Saw Draw may be more easily demonstrated.

To begin, a cell is opened under the name 'unweight' and Saw Draw is toggled to the edit mode with the Esc key. Next a grid is set to 10 by 10 micrometers. Then two rectangles are created as shown in the following figure. These rectangles represent a single split tap electrode. Each rectangle is 10 μm by 160 μm , where 80 μm corresponds to a wave length.

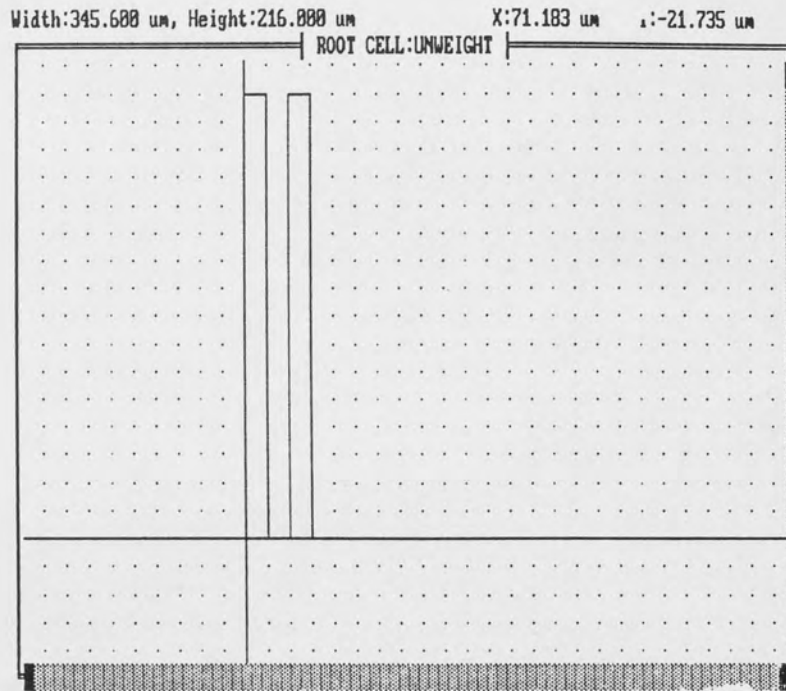


Figure 46. Example: Two Rectangles Added.

Both rectangles are then selected in preparation for an array operation. This is done by placing the edit box across both taps and pressing the S key.

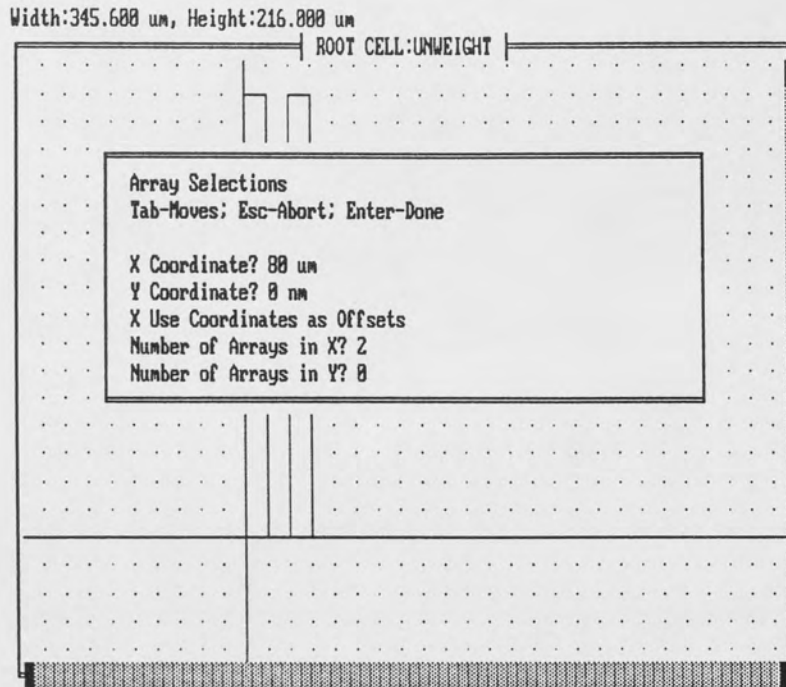


Figure 48. Example: Array Prompts.

After entry of all data to the prompts has been completed, the taps are arrayed as shown.

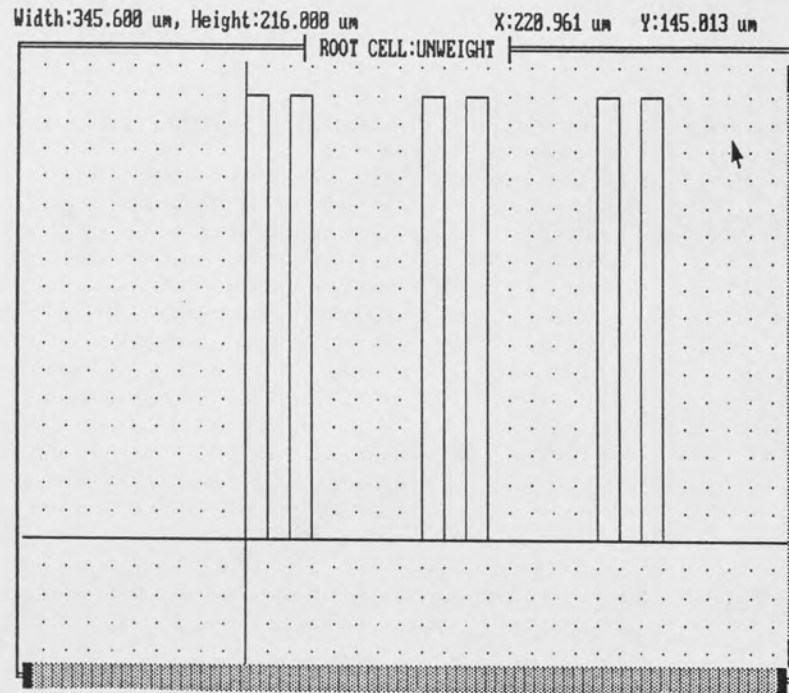


Figure 49. Example: Array Results.

At this point we have created one half of the fingers required. By pressing V to view the entire cell and adding the bus bar and bond pad, the transducer is half complete.

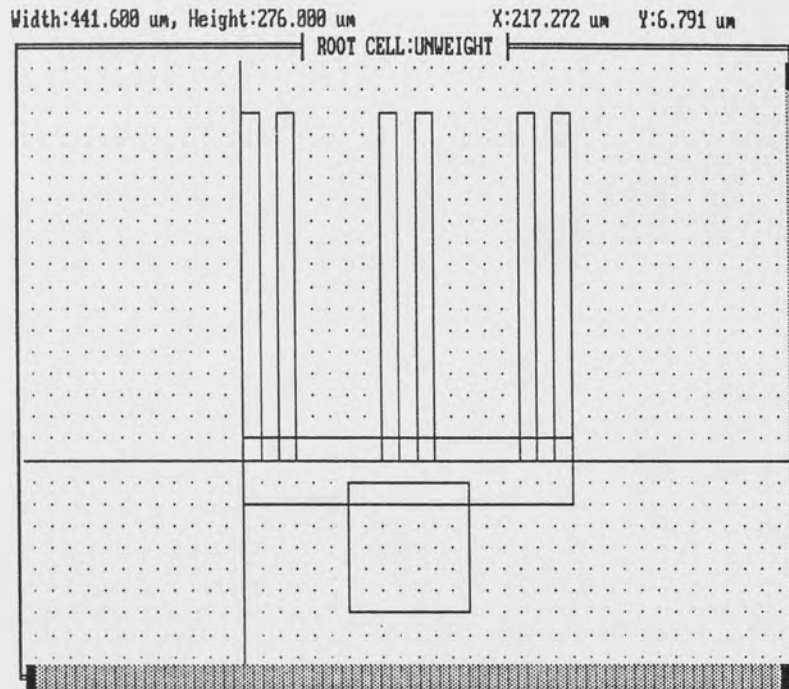


Figure 50. Example: Bus Bar Added.

Next all rectangles are selected by placing the edit box around all of them and pressing the S key. Then the edit box is positioned as shown in the next figure. The position of the edit box will enable a copy operation followed by a y mirror image operation to complete the transducer.

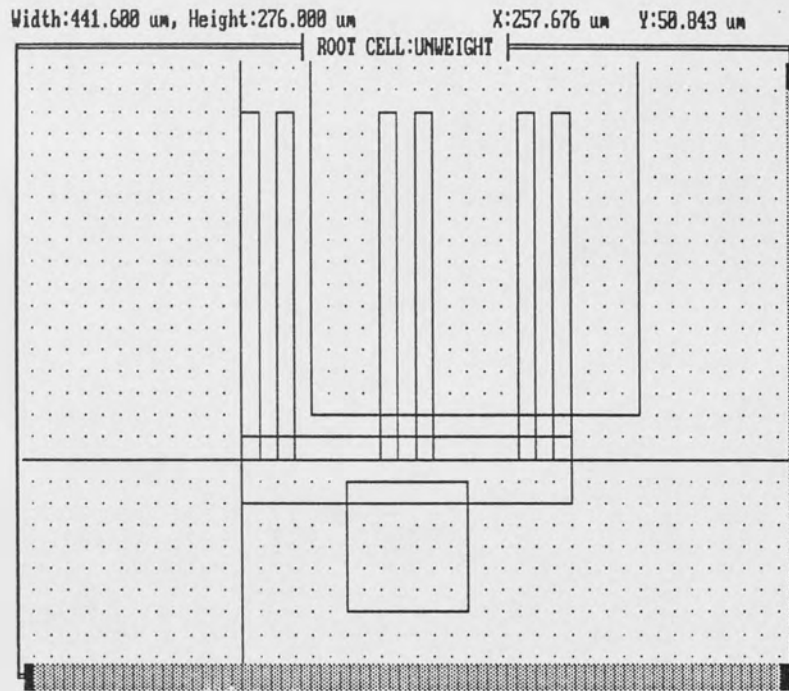


Figure 51. Example: Copy and Y Mirror Image.

The results of the copy and y mirror operations are shown in the following figure.

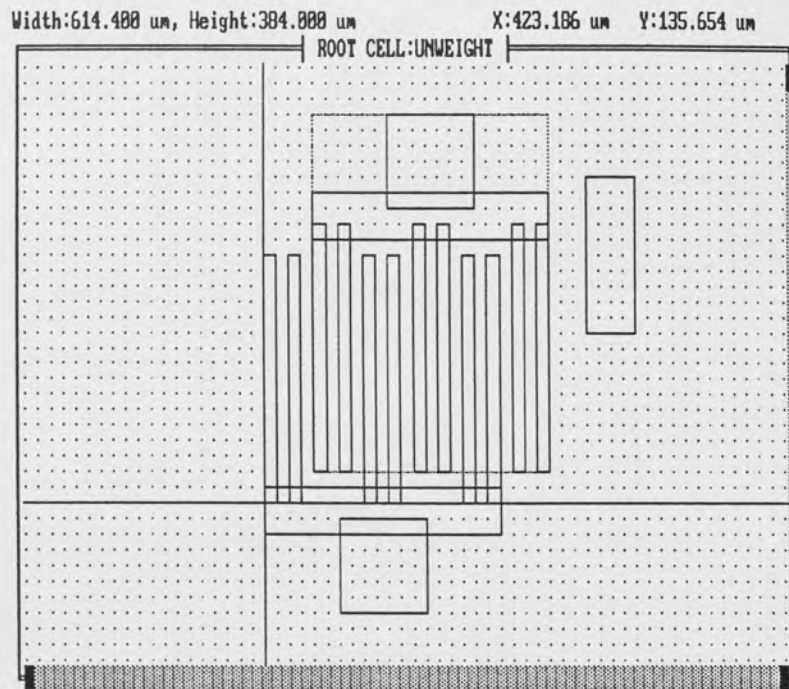


Figure 52. Example: Complete Transducer.

The cell "unweight" is then saved by toggling back to the menu mode. Next the cell "example" is opened and edited. Using the include cell function, Cntl+F key stroke, the previous cell is included as shown.

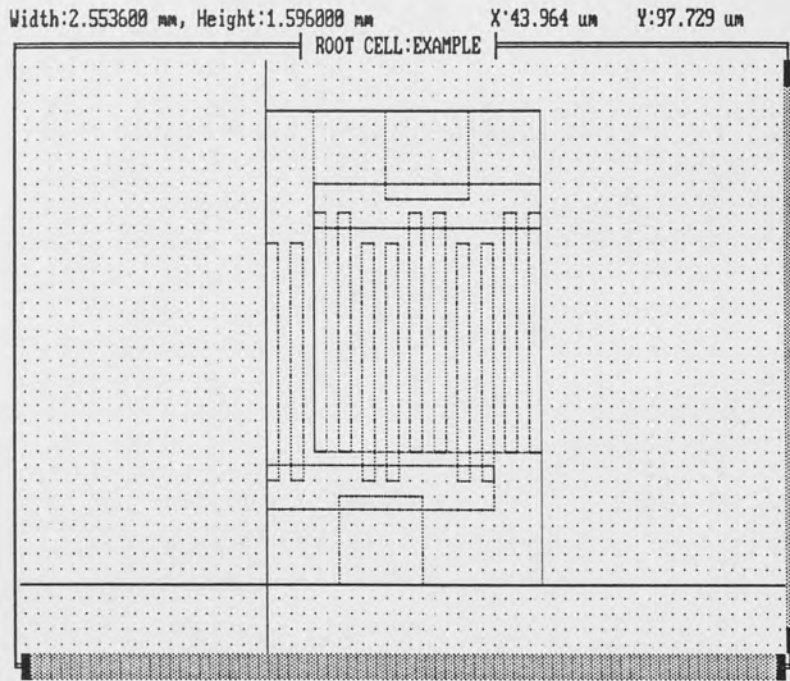


Figure 53. Example: Cell Included.

By including this cell again to the left of the first subcell, the device is completed.

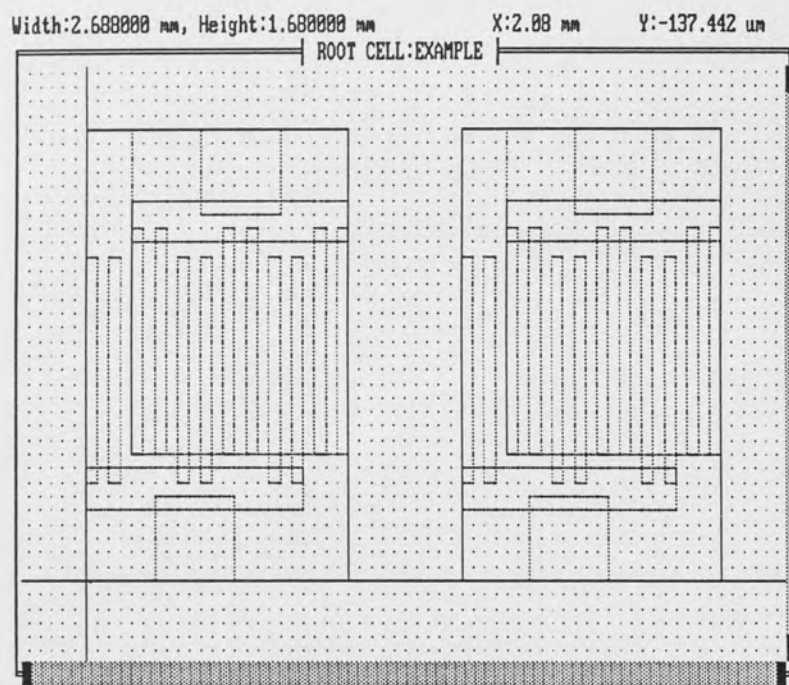


Figure 54. Example: Complete Device.

CONCLUSIONS

A software package solving problems associated with SAW layout has been presented. This package was compared to past work in the area of VLSI tools, CAD packages and custom SAW layout packages and has been shown to have several superior characteristics.

The data structure of Saw Draw was designed around the requirements of SAW layout, permitting multi-level masking, indexing and cell hierarchy. The data structure was then defined in terms of functionality. This data structure was shown to significantly reduce storage requirements, access and plot time.

Finally, a QuickBASIC program was written and functionally described in terms of its edit capabilities so as to encompass the needs of SAW layout.

APPENDIX

```

DECLARE FUNCTION insidebox% (x1#, x2#, y1#, y2#, x%, y%)
DECLARE SUB reccopy (title$, sel#(), c%, test%, sx!, st!)
DECLARE SUB filesearch (filter$, cellname$, ext$, c%)
DECLARE SUB aspectratio (x#, y#, w#, h%)
DECLARE SUB mouserereset ()
DECLARE SUB saveas (rec#())
DECLARE SUB mergecell (rec#(), sel#())
DECLARE SUB recfile (title$, sel#(), c%, sx!, st!)
DECLARE FUNCTION units# (v$)
DECLARE FUNCTION lobyte% (i%)
DECLARE FUNCTION directory$ ()
DECLARE FUNCTION timmer# ()
DECLARE FUNCTION setrec% (s#, i%)
DECLARE FUNCTION getrec# (i%)
DECLARE FUNCTION short$ (s$)
DECLARE FUNCTION packstr$ (s$)
DECLARE FUNCTION arctan# (x#, y%)
DECLARE FUNCTION mki% (h%, l%)
DECLARE FUNCTION hbyte% (i%)
DECLARE SUB checkwraparound (rec#())
DECLARE SUB savecell (rec#(), c%)
DECLARE SUB show (rec#())
DECLARE SUB changelayer (rec#())
DECLARE SUB recarray (sel#(), c%, x%, y%)
DECLARE SUB recstep (dx#, dy#, dw#, dh#, dt#, repeats%, c%)
DECLARE SUB recmove (sel#(), c%, sx!, sy!)
DECLARE SUB steprepeat (sel#(), rec#())
DECLARE SUB recquery (title$, sel#(), c%, test%, sx!, st!)
DECLARE SUB selstep (dx#, dy#, dw#, dh#, dt#, repeats%, c%)
DECLARE SUB printcell (sel#())
DECLARE FUNCTION value$ (v#)
DECLARE FUNCTION degrees# (d#)
DECLARE FUNCTION error$ (i%)
DECLARE SUB dump (format%)
DECLARE SUB findfile (file$, ax%, dir$)
DECLARE SUB grid (iflag%)
DECLARE SUB setgrid ()
DECLARE SUB filestatus (file$, ax%, dir$)
DECLARE SUB mouse (ax%, bx%, cx%, dx%)
DECLARE SUB openwindow (row%, col%, num%, length%)
DECLARE SUB main (mainmenu$, mnu$(), mnu%(), imnu%, ians%)
DECLARE SUB help (i%)
DECLARE SUB restart (irec#, rec#())
DECLARE SUB loadfile (irec#, rec#())
DECLARE SUB plotall (expand%)
DECLARE SUB plottedit ()
DECLARE SUB pack (rec#(), del%)
DECLARE SUB mc (sel#())
DECLARE SUB errors (a%)
DECLARE SUB within (rec#(), lmt#(), test%)
DECLARE SUB rotate (rec#(), x#, y#, t#)
DECLARE SUB corners (rec#(), xbl#, ybl#, xtr#, ytr#, xtl#, ytl#, xbr#, ybr#)
DECLARE SUB winon ()
DECLARE SUB readrec (irec#, rec#())
DECLARE SUB plotrec (irec#, rec#(), i%)
DECLARE SUB onscreen (rec#(), test%)
DECLARE SUB opencell (irec#, rec#())
DECLARE SUB closecell (irec#, rec#())
DECLARE SUB plotcell (rec#(), flag%)
DECLARE SUB editcell (rec#())
DECLARE SUB selplot (bright%)
DECLARE SUB prompt (p$, v$, i%, c%)
DECLARE SUB openfile (cellname$, rec#())
DECLARE SUB header ()
DECLARE SUB viewport ()
DECLARE SUB checkdims (rec#(), xmin#, ymin#, w#, h#)
DECLARE SUB writerec (irec#, rec#(), check%)
DECLARE SUB checkheader (irec#, rec#())

```

```

DECLARE SUB checkmins (rec#(), xmin#, ymin#, w#, h#)
DECLARE SUB addoff (rec#())
DECLARE SUB suboff (rec#())
DECLARE SUB addrec (irec#, rec#(), check%)
DECLARE SUB checktheta (rec#())
DECLARE SUB menu (mnu$, mnu%, length%, row%, col%, ians%, c%)
DECLARE SUB inverse (row%, col%, length%)
DECLARE SUB graphwindow (row%, col%, num%, length%, action%)
DECLARE SUB box (row%, col%, num%, length%)
DECLARE SUB clearwindow (row%, col%, num%, length%)
DECLARE SUB winoff ()
DECLARE SUB query (title$, opts%, c%)
DECLARE SUB getline (v$, lv%, c%)
DECLARE SUB switch (f$, c%)
DECLARE SUB recscale (title$, sx!, sy!, c%)
DECLARE SUB editrec (title$, sel#(), c%)
DECLARE SUB cursor (c$, insert%)
DECLARE SUB validate (v&, hi&, lo&)
DECLARE SUB zoom (rec#())
DECLARE SUB systemstatus (rec#())
DECLARE SUB setbox (button%, sel#(), rec#(), cursorx%, cursory%, c%)
DECLARE SUB evaluate (c$, sel#(), rec#(), irec#, cursorx%, cursory%)
DECLARE SUB array (sel#(), rec#())
DECLARE SUB selcopy (sel#(), rec#(), flag%)
DECLARE SUB expandall (irec#, rec#(), test%)
DECLARE SUB getfile (rec#(), sel#())
DECLARE SUB selerase (sel#(), rec#(), allplot%)
DECLARE SUB layers ()
DECLARE SUB selmove (sel#(), rec#(), allplot%)
DECLARE SUB scroll (test%)
DECLARE SUB editset (action%, irec#, sel#(), rec#(), sx!, sy!)
DECLARE SUB selscale (sel#(), rec#())
DECLARE SUB slct (sel#(), rec#(), j%)
DECLARE SUB putlabel (irec#, sel#(), rec#(), c%)
DECLARE SUB flip (sel#(), rec#(), iflag%)
DECLARE SUB setwindow (sel#())
DECLARE SUB getcell (rec#(), sel#())
DECLARE SUB label (sel#(), s&, l$)
DECLARE SUB selrecs (jrec#, rec#(), sel#())
DECLARE SUB seteditpath (rec#(), sel#())
DECLARE SUB unexpand (irec#, rec#())
DECLARE SUB expand (irec#, rec#())
DECLARE SUB mirror (rec#(), iflag%)
DECLARE SUB unedit (rec#())
DECLARE SUB scale (rec#(), sx!, sy!)
DECLARE SUB sawcad (rec#(), sel#(), sx!, st!)
'=====
'PROGRAM MODULE SD - SawDraw version 1.00
'=====
' Copyright Jeff Abbott 1987 (c), all rights reserved
'
TYPE regtype
    ax AS INTEGER
    bx AS INTEGER
    cx AS INTEGER
    dx AS INTEGER
    bp AS INTEGER
    si AS INTEGER
    di AS INTEGER
    flags AS INTEGER
    ds AS INTEGER
    es AS INTEGER
END TYPE

TYPE sdrecord
    l AS INTEGER
    r AS INTEGER
    x AS LONG

```

```

y AS LONG
w AS LONG
h AS LONG
t AS SINGLE
f AS DOUBLE
xo AS LONG
yo AS LONG
END TYPE

COMMON current#(), parent#(), child#(), explevel%, level%, layer%
COMMON p$(), v#(), l%(), t%(), ifor1%, ifor2%, ifor3%, fileopen%
COMMON xoff#, yoff#, toff#, blank$, sminx#, sminy#, sw#, sh#
COMMON xstep&, ystep&, xorigin&, yorigin&, gridon%, saved%, editing%
COMMON mouseon%, xpixel%, ypixel%, wavelength&, cells%, recs%
COMMON sx1#, sx2#, sy1#, sy2#, editlevel%, plotted%, zfactor&

'$DYNAMIC
DIM rec$(10), mnu$(10), mnu%(10), editpath$(50), sel$(32000)
DIM current$(50), parent$(50, 10), child$(50, 10)
DIM p$(10), v$(10), l$(10), t$(10)
DIM f$(32), sel$(10)
'$STATIC

CONST fifteen% = 15
CONST pi# = 3.1415926535#

'ON ERROR GOTO errorline

'-----
DEF FNrow% (y%)
'-----
FNrow% = INT(y% / ypixel%) + 1
END DEF

'-----
DEF FNcol% (x%)
'-----
FNcol% = INT(x% / xpixel%) + 1
END DEF

'=====
' Main program: sd.bas
'=====

SCREEN 9
blank$ = SPACE$(79)

'setup - screens, colors, plotter type, printer type
'      paper size of plotter/printer, pens/ribbon colors
'      resolution (dpi/pen thick)
'
ifor1% = 14: ifor2% = 7: ifor3% = 2: ibak% = 0: editing% = 1
xpixel% = 8: ypixel% = 14: xmax% = INT(xpixel% * 80 - 1): ymax% = INT(ypixel% * 25 - 1)
speed% = 1: plotfile% = 1: layer% = 2
sx1# = -320000!: sy1# = 2000000!: sx2# = 320000!: sy2# = 200000!

COLOR ifor3%, ibak%

imnu% = 1: ians% = 22
irow% = 25: irec# = 0
saved% = 0

IF INSTR(1, UCASE$(COMMAND$), "OFF") THEN mouseon% = 0 ELSE mouseon% = 1

mainmenu$ = "  File  Output  Help  "

mnu$(0) = "11  Open...   Save...   Copy As... DOS Shell  Quit      "
mnu$(1) = "11  Printer... SawCad... CIF...   "
mnu$(2) = "11  General  Menus    KeyStrokes I/O    Printing  "

```

```

imnu% = 0
ians% = 1
saved% = 1

CALL mouse(0, mouseon%, cx%, dx%)           ' check mouse status
CALL mouse(1, bx%, cx%, dx%)                 ' show mouse
CALL mouse(2, bx%, cx%, dx%)                 ' hide mouse
CALL mouse(7, 0, xpixel%, INT(80 * xpixel% - 1)) ' set x mouse limits
CALL mouse(8, 0, 2 * ypixel%, INT(25 * ypixel% - 1)) ' set y mouse limits

start1:

CALL openwindow(3, 2, 22, 78)

WHILE 1
  CALL main(mainmenu$, mnu$(), mnu%(), imnu%, ians%)
  SELECT CASE imnu%
  CASE 0: SELECT CASE ians%
    CASE 0
      idum% = fileopen%
      fileopen% = 0
      CALL savecell(rec#(), c%)
      fileopen% = idum%
      SELECT CASE c%
      CASE 27
      CASE ELSE
        CALL loadfile(irec#, rec#())
        IF plotted% = 0 AND fileopen% = 1 THEN
          CALL plotall(0)
          saved% = 1
        END IF
      END SELECT
    CASE 1: SELECT CASE saved%
      CASE 0: SELECT CASE needspacking%
        CASE 1: CALL pack(rec#(), 1)
        CASE ELSE
          END SELECT
        CALL restart(irec#, rec#())
        CLOSE
        SHELL "copy *.rnd *.cel/b > nul "
        cellname$ = packstr$(MKD$(child#(0, 8)) + ".rnd")
        CALL openfile(cellname$, rec#())
        saved% = 1
      CASE ELSE
        END SELECT
    CASE 2: CALL saveas(rec#())
    CASE 3
      PCOPY 0, 1
      SCREEN 0
      PRINT "Type 'EXIT' to Return to SawDraw"
      SHELL
      SCREEN 9
      PCOPY 1, 0
      CALL openwindow(3, 2, 22, 78)
    CASE 4
      CALL savecell(rec#(), c%)
      SELECT CASE c%
      CASE 27
      CASE ELSE
        SCREEN 0
        IF fileopen% THEN CLOSE : KILL "*.rnd"
        SYSTEM
      END SELECT
    CASE ELSE
      END SELECT
  CASE 1: SELECT CASE ians%
    CASE 0
      FOR i% = 3 TO 6

```

```

                                sel#(i%) = child#(level%, i%)
                                NEXT
                                CALL printcell(sel#())
CASE 1: CALL dump(1)
CASE 2: CALL dump(2)
CASE ELSE
END SELECT
CASE 2: SELECT CASE ians%
CASE 0, 1, 2, 3, 4: CALL help(ians%)
CASE ELSE
END SELECT
CASE ELSE
END SELECT
IF (ians% = -1) THEN
    CALL mc(sel#())
END IF
WEND

errorline:
IF ERR = 6 THEN RESUME NEXT
CALL errors(a%)
IF a% = ASC("a") THEN
    CLS
    PRINT "SYNTAX: SD [/mouse off]"
    SYSTEM
ELSEIF a% = ASC("i") THEN
    RESUME NEXT
ELSEIF a% = ASC("r") THEN
    RESUME start1:
ELSEIF a% = ASC("h") THEN
    CALL help(0)
    RESUME start1:
END IF
'END SUB

```

```
-----  
SUB addoff (rec#()) STATIC  
-----  
SHARED xoff#, yoff#, toff#, child#(), level%  
STATIC dx#, dy#, r#  
IF toff# THEN  
    CALL rotate(rec#(), child#(level%, 3), child#(level%, 4), (toff#))  
END IF  
rec#(3) = rec#(3) + xoff#  
rec#(4) = rec#(4) + yoff#  
END SUB
```



```

'-----
SUB addrec (irec#, rec#(), check%) STATIC
'-----
SHARED child#(), level%

IF check% THEN
    CALL checktheta(rec#())
    IF child#(level%, 2) = 0 THEN
        FOR i% = 3 TO 6: child#(level%, i%) = rec#(i%): NEXT
    ELSE
        CALL checkheader(irec#, rec#())
    END IF
END IF

IF (FIX(rec#(5)) < 1) THEN rec#(5) = 1
IF (FIX(rec#(6)) < 1) THEN rec#(6) = 1

child#(level%, 2) = child#(level%, 2) + 1
irec# = child#(level%, 2)
CALL writerec(irec#, rec#(), check%)

END SUB

```

```
!-----  
FUNCTION arctan# (x#, y#) STATIC  
!-----  
IF x# > 0 THEN  
    arctan# = ATN(y# / (x# + .00001))  
ELSE  
    arctan# = ATN(y# / (x# - .00001)) + pi#  
END IF  
END FUNCTION
```

```

-----
SUB array (sel#(), rec#()) STATIC
-----
SHARED sminx#, sminy#, sw#, sh#
STATIC j%, i%, dum%, ix%, iy%, dx#, dy#, sx#, sy#
IF getrec#(0) > 0 THEN
    dum% = 3
    i% = 1
    c% = 0
    sx# = sminx#
    sy# = sminy#
    dum1# = sel#(3)
    dum2# = sel#(4)
    sel#(3) = sminx# + sw#
    sel#(4) = sminy# + sh#
    CALL recarray(sel#(), c%, ix%, iy%)
    LOCATE 1, 1: PRINT sel#(7);
    IF ix% >= 0 AND iy% >= 0 AND c% <> 27 THEN
        dy# = sel#(4) - sy#
        dx# = sel#(3) - sx#
        sel#(7) = 0
        CALL selplot(0)
        i% = 0
        WHILE i% <= ix%
            sel#(3) = sx# + dx# * i%
            j% = 0
            WHILE j% <= iy%
                sel#(4) = sy# + dy# * j%
                IF (i% > 0 OR j% > 0) THEN
                    CALL selcopy(sel#(), rec#(), 0)
                END IF
                j% = j% + 1
            WEND
            i% = i% + 1
        WEND
    ELSE
        sel#(3) = dum1#
        sel#(4) = dum2#
    END IF
END IF
END SUB

```

```

SUB changelayer (rec#()) STATIC
  SHARED layer%
  IF setrec%(s#, -1) THEN
    i% = 1
    c% = 0
    l$ = STR$(layer%)
    DO
      CALL prompt("Change to Layer Number(1-6)? ", l$, 2, c%)
      layer% = VAL(l$)
    LOOP UNTIL c% = 27 OR (layer% > 0 AND layer% < 7)
    IF c% <> 27 THEN
      WHILE i% <= setrec%(s#, -1)
        irec# = getrec%(i%)
        CALL readrec(irec#, rec#())
        SELECT CASE rec%(1)
          CASE 1, 2, 3, 4, 5, 6: rec%(1) = layer%
            CALL writerec(irec#, rec#(), 1)
          CASE ELSE
            END SELECT
        i% = i% + 1
      WEND
    END IF
  END IF
END SUB

```

```
-----  
SUB checkdims (rec#(), xmin#, ymin#, w#, h#) STATIC  
-----  
CALL corners(rec#(), x1#, y1#, x3#, y3#, x2#, y2#, x4#, y4#)  
IF x3# > w# + xmin# THEN w# = x3# - xmin#  
IF x4# > w# + xmin# THEN w# = x4# - xmin#  
IF y3# > h# + ymin# THEN h# = y3# - ymin#  
IF y2# > h# + ymin# THEN h# = y2# - ymin#  
END SUB
```

```
!-----  
SUB checkheader (irec#, rec#()) STATIC  
!-----  
SHARED fileopen%, child#(), level%, parent#(), xoff#, yoff#  
  
CALL checkmins(rec#(), child#(level%, 3), child#(level%, 4), child#(level%, 5), child#(level%, 6))  
CALL checkdims(rec#(), child#(level%, 3), child#(level%, 4), child#(level%, 5), child#(level%, 6))  
  
parent#(level%, 3) = child#(level%, 3) - parent#(level%, 9)  
parent#(level%, 4) = child#(level%, 4) - parent#(level%, 10)  
parent#(level%, 5) = child#(level%, 5)  
parent#(level%, 6) = child#(level%, 6)  
  
END SUB
```

```
!-----  
SUB checkmins (rec#(), xmin#, ymin#, w#, h#) STATIC  
!-----  
CALL corners(rec#(), x1#, y1#, x3#, y3#, x2#, y2#, x4#, y4#)  
IF x1# < xmin# THEN w# = w# + xmin# - x1#: xmin# = x1#  
IF x2# < xmin# THEN w# = w# + xmin# - x2#: xmin# = x2#  
IF y1# < ymin# THEN h# = h# + ymin# - y1#: ymin# = y1#  
IF y4# < ymin# THEN h# = h# + ymin# - y4#: ymin# = y4#  
END SUB
```



```

-----
SUB checktheta (rec#()) STATIC
-----
WHILE ABS(rec#(7)) > (pi# / 2!)
    CALL corners(rec#(), x1#, y1#, x3#, y3#, x2#, y2#, x4#, y4#)

    IF rec#(7) > (pi# / 2!) THEN
        rec#(7) = rec#(7) - pi#
    ELSEIF rec#(7) < (-pi# / 2!) THEN
        rec#(7) = rec#(7) + pi#
    END IF
    rec#(3) = x3#
    rec#(4) = y3#
WEND

IF ABS(rec#(7)) > (pi# / 4!) THEN
    CALL corners(rec#(), x1#, y1#, x3#, y3#, x2#, y2#, x4#, y4#)
    w# = rec#(5)
    rec#(5) = rec#(6)
    rec#(6) = w#
    IF (rec#(7) > 0) THEN
        rec#(3) = x2#
        rec#(4) = y2#
        rec#(7) = rec#(7) - (pi# / 2!)
    ELSE
        rec#(3) = x4#
        rec#(4) = y4#
        rec#(7) = rec#(7) + (pi# / 2!)
    END IF
END IF
END SUB

```

```

'-----
SUB closecell (irec#, rec#()) STATIC
'-----
SHARED level%, current#(), parent#(), editing%, child#()
'
' update calling (subcell) record
'
CALL header ' update child header
IF level% THEN
    level% = level% - 1
    FOR i% = 1 TO 10
        rec#(i%) = child#(level%, i%) ' find parent cell name
    NEXT
    irec# = current#(level%)
    cellname$ = packstr$(MKD$(rec#(8))) + ".rnd"
    CALL openfile(cellname$, rec#()) ' open old parent
    CALL readrec(irec#, rec#())
    FOR i% = 1 TO 10
        rec#(i%) = parent#(level% + 1, i%)
    NEXT
    rec#(3) = child#(level% + 1, 3) + parent#(level% + 1, 9) ' reset ll
    rec#(4) = child#(level% + 1, 4) + parent#(level% + 1, 10) '
    IF editing% THEN
        CALL writerec(irec#, rec#(), 1)
        CALL checkheader(irec#, rec#()) ' check dims and ll of current header
    END IF
END IF
END SUB

```

```
!-----  
SUB corners (rec#(), xbl#, ybl#, xtr#, ytr#, xtl#, ytl#, xbr#, ybr#) STATIC  
!-----  
xang = -rec#(7)  
xbl# = rec#(3)  
ybl# = rec#(4)  
xtl# = xbl# + rec#(6) * SIN(xang)  
ytl# = ybl# + rec#(6) * COS(xang)  
xtr# = xtl# + rec#(5) * COS(xang)  
ytr# = ytl# - rec#(5) * SIN(xang)  
xbr# = xtr# - rec#(6) * SIN(xang)  
ybr# = ytr# - rec#(6) * COS(xang)  
END SUB
```

```
-----  
SUB editcell (rec#()) STATIC  
-----  
SHARED editing%, editlevel%, editpath#(), level%  
STATIC irec#  
CALL restart(irec#, rec#())  
WHILE level% < editlevel%  
    irec# = editpath#(level%)  
    CALL readrec(irec#, rec#())  
    rec#(2) = 1  
    CALL writerec(irec#, rec#(), 1)  
    CALL opencell(irec#, rec#())  
WEND  
editing% = 1  
END SUB
```

```

-----
SUB editsel (action%, irec#, sel#(), rec#(), sx, sy) STATIC
-----
DIM a%(3)
SHARED sminx#, sminy#, sw#, sh#, selbot&, seltop&, saved%, needspacking%
STATIC dx#, dy#, dt#

saved% = 0

FOR i% = 1 TO 3
    a%(i%) = action% - INT(FIX(action% / 10!)) * 10
    action% = INT(FIX(action% / 10!))
NEXT

SELECT CASE a%(1)
CASE 0
CASE 1 ' rotate
    CALL rotate(rec#(), sel#(3), sel#(4), sel#(7))
CASE 2 ' flipx
    CALL mirror(rec#(), 1)
CASE 3 ' flipy
    CALL mirror(rec#(), 0)
CASE 4 ' move & rotate
    CALL scale(rec#(), sx, sy)
    dx# = FIX(sel#(3) - sminx#)
    dy# = FIX(sel#(4) - sminy#)
    dt# = FIX(sel#(7))
    rec#(3) = rec#(3) + dx#
    rec#(4) = rec#(4) + dy#
    rec#(9) = rec#(9) + dx#
    rec#(10) = rec#(10) + dy#
    CALL rotate(rec#(), sel#(3), sel#(4), sel#(7))
END SELECT

SELECT CASE a%(2)
CASE 0
CASE 1 ' add
    CALL addrec(irec#, rec#(), 1)
CASE 2 ' update
    CALL writerec(irec#, rec#(), 1)
CASE 3 ' delete
    needspacking% = 1
    jrec# = irec#
    SELECT CASE rec#(1)
    CASE 0, 7
    CASE ELSE
        SELECT CASE rec#(2)
        CASE 0
        CASE ELSE: irec# = rec#(2)
        END SELECT
    END SELECT
    rec#(1) = 0: rec#(2) = 0
    CALL writerec(jrec#, rec#(), 1)
END SELECT

SELECT CASE a%(3)
CASE 0
CASE 1 ' expand
    CALL expand(irec#, rec#())
CASE 2 ' unexpand
    CALL unexpand(irec#, rec#())
CASE 3 ' plot
    CALL plotrec(irec#, rec#(), 1)
CASE 4 ' edit
    CALL seteditpath(rec#(), sel#())
CASE 5 ' unedit
    CALL unedit(rec#())

```

END SELECT

IF (selbot& > irec#) OR (selbot& = 0) THEN selbot& = irec#
IF seltop& < irec# THEN seltop& = irec#

END SUB

```

-----
SUB evaluate (c%, sel#(), rec#(), irec#, cursorx%, cursory%) STATIC
-----
SHARED speed%, editlevel%, xpixel%, ypixel%, layer%, explevel%, wavelength%
SHARED seltop&, selbot&, child#(), blank$, sx1#, sy1#, sx2#, sy2#
DIM rec2$(10)
packall% = 0
allplot% = 0
plotsel% = 0
plotflag% = 0
del% = 0
c$ = LCASE$(CHR$(c%))
SELECT CASE c$
  CASE "a" ' array
    CALL array(sel#(), rec#())
    plotsel% = 1
    packall% = 1
  CASE "b" ' box
    CALL checktheta(sel#())
    CALL editrec("Set Edit Box Position", sel#(), c%)
  CASE "c" ' copy
    CALL selcopy(sel#(), rec#(), 1)
    plotsel% = 1
    packall% = 1
  CASE "d" ' draw screen
    allplot% = 1
  CASE "e" ' expand selected cells
    CALL expandall(irec#, rec#(), 1)
    allplot% = 1
  CASE "f"
    CALL getfile(rec#(), sel#())
    plotsel% = 1
    packall% = 1
  CASE "g"
    CALL setgrid
  CASE "h" ' change layers
    CALL changelayer(rec#())
    plotsel% = 1
  CASE "k" ' kill
    CALL selerase(sel#(), rec#(), allplot%)
    packall% = 1
  CASE "l"
    CALL layers
  CASE "m" ' move selections
    CALL selmove(sel#(), rec#(), allplot%)
    plotsel% = 1
    packall% = 1
  CASE "o" ' on screen
    CALL scroll(allplot%)
  CASE "p" ' paint
    sel#(1) = layer%
    CALL checktheta(sel#())
    CALL editssel(310, irec#, rec#(), sel#(), 1!, 1!)
  CASE "q"
    CALL systemstatus(rec#())
  CASE "r" ' step and repeat
    CALL steprepeat(sel#(), rec#())
  CASE "s" ' select
    CALL slct(sel#(), rec#(), 0)
    plotsel% = 1
  CASE "t" ' text
    CALL putlabel(irec#, sel#(), rec#(), c%)
    plotsel% = 1
  CASE "u" ' unexpand
    CALL expandall(irec#, rec#(), 0)
    allplot% = 1
  CASE "v" ' view/center

```



```

        CALL viewport
        allplot% = 1
CASE "w" ' set wavelength value
    w# = wavelength&
    w$ = value$(w#)
    CALL prompt("Value of Wave Lengths? ", w$, 20, c%)
    IF c% <> 27 THEN wavelength& = ABS(VAL(w$)) * units$(w$)
CASE "x" ' flip x
    CALL flip(sel#(), rec#(), 1)
    plotsel% = 1
CASE "y" ' flip y
    CALL flip(sel#(), rec#(), 0)
    plotsel% = 1
CASE "z" ' zoom
    CALL zoom(sel#())
    allplot% = 1
CASE "?" ' help
    CALL help(0)
CASE "+" ' select
    CALL slct(sel#(), rec#(), 1)
    plotsel% = 1
CASE ELSE
    SELECT CASE c%
    CASE 4 ' cntl+d show dta
        CALL show(rec#())
    CASE 5 'cntl+e set soft expand level
        CALL prompt("Expand how many levels deep? ", l$, 2, c%)
        explevel% = INT(ABS(VAL(l$)))
        allplot% = 1
    CASE 6 'cntl+f - add a subcell
        CALL getcell(rec#(), sel#())
        CALL plotrec(irec#, rec#(), 1)
        plotsel% = 1
    CASE 16 ' cntl+p - print
        CALL printcell(sel#())
    CASE 18 ' cntl+r - restart the mouse
        CALL mouserreset
    CASE 19 ' cntl+s - edit a selected cell
        CALL editcell(400, irec#, sel#(), rec#(), 1!, 1!)
        allplot% = 1
        plotflag% = 1
    CASE 26 ' cntl+z zoom with input mag at box center
        CALL setwindow(sel#())
        allplot% = 1
    CASE 31 ' alt+s - back out of edit path
        CALL editcell(500, irec#, sel#(), rec#(), 1!, 1!)
        FOR i% = 3 TO 7
            sel#(i%) = child#(editlevel%, i%)
        NEXT
        allplot% = 1
        plotflag% = 1
    CASE 44 ' alt+z - correct aspect ratio after zoom
        w# = sx2# - sx1#
        h# = sy2# - sy1#
        CALL aspectratio((sx1# - xoff#), (sy1# - yoff#), w#, h#)
        allplot% = 1
    CASE 50 'alt+m
        CALL mergecell(rec#(), sel#())
        packall% = 1
        plotsel% = 1
    CASE ELSE
        LOCATE 1, 1: PRINT " Key Pressed:"; c%;
        'a$ = INPUT$(1)
    END SELECT
END SELECT
IF allplot% = 1 THEN
    CALL plotall(plotflag%)
END IF

```

```
IF plotse1% = 1 OR allplot% = 1 THEN
  CALL header
  IF getrec#(0) > 0 THEN
    CALL selplot(1)
  END IF
END IF
IF packall% = 1 THEN CALL pack(rec#(), 0)
END SUB
```

```
-----  
SUB expand (irec#, rec#()) STATIC  
-----  
|   rec#(2) set for expand - test% set to 1  
|   rec#(2) unchanged      - test% unchanged  
|  
CALL readrec(irec#, rec#())  
IF rec#(1) = 7 THEN  
    SELECT CASE rec#(2)  
    CASE 0: rec#(2) = 1  
        CALL writerec(irec#, rec#(), 0)  
    CASE ELSE  
    END SELECT  
END IF  
END SUB
```

```

'-----
SUB expandall (irec#, rec#(), test%) STATIC
'-----
SHARED level%, selbot&, seltop&, saved%
irec# = 0
i% = 0
WHILE i% < getrec#(0)
    i% = i% + 1
    irec# = getrec#(i%)
    SELECT CASE test%
    CASE 0: CALL unexpand(irec#, rec#())
    CASE 1: CALL expand(irec#, rec#())
    END SELECT
    saved% = 0
    IF (selbot& > irec#) OR (selbot& = 0) THEN selbot& = irec#
    IF (seltop& < irec#) THEN seltop& = irec#
WEND
END SUB

```

```
!-----  
SUB flip (sel#(), rec#(), iflag%) STATIC  
!-----  
CALL selplot(-1)  
i% = 1  
WHILE i% <= setrec%(s#, -1)  
    irec# = getrec%(i%)  
    CALL readrec(irec#, rec#())  
    SELECT CASE iflag%  
    CASE 0: CALL editssel(23, irec#, sel#(), rec#(), 1!, 1!)  
    CASE 1: CALL editssel(22, irec#, sel#(), rec#(), 1!, 1!)  
    END SELECT  
    i% = i% + 1  
WEND  
CALL selplot(1)  
END SUB
```

```

-----
SUB getfile (rec#(), sel#()) STATIC
-----
SHARED format%, level%, layer%, child#(), sminx#, sminy#, sw#, sh#, blank$
irtn% = 1
format$ = "SawCad"
p$ = format$
CALL winoff
c% = 0
total# = child#(level%, 2)
WHILE irtn% AND c% <> 27
    CALL prompt(p$ + " File? ", file$, 12, c%)
    IF short$(file$) = "" THEN file$ = "**.*"
    IF INSTR(1, file$, "**") > 0 OR INSTR(1, file$, "?") > 0 THEN
        SELECT CASE INSTR(1, file$, ".")
            CASE 0: ext$ = ""
            CASE ELSE
                ext$ = MID$(file$, INSTR(1, file$ + " ", ".") + 1)
        END SELECT
        file$ = LEFT$(file$, INSTR(1, file$ + ".", ".") - 1)
        CALL filesearch(short$(file$), file$, ext$, c%)
    END IF
    CALL filestatus(file$, irtn%, dir$)
    p$ = " File Not Found: " + dir$ + short$(file$) + ", "
WEND
IF c% <> 27 THEN
    CLOSE 4
    OPEN file$ FOR INPUT AS 4
    CALL recfile("Set File Scales/Orientation", sel#(), c%, sx, st)
    IF c% <> 27 THEN
        LOCATE 1, 1: PRINT LEFT$("Loading File..." + blank$, 78);
        SELECT CASE format%
            CASE 0: CALL sawcad(rec#(), sel#(), sx, st)
                    w# = sel$(5)
                    h# = sel$(6)
                    recs% = sel$(2)
            CASE ELSE
                END SELECT
        LOCATE 1, 1: PRINT LEFT$("Selecting Rectangles..." + blank$, 78)
        CALL selplot(0)
        CALL selrecs(jrec#, rec#(), sel#())
        CALL checkheader(jrec#, sel#())
        CALL header
    END IF
    CLOSE 4
END IF
END SUB

```

```
!-----  
FUNCTION getrec# (i%) STATIC  
!-----  
SHARED sel%()  
getrec# = sel%(i%)  
END FUNCTION
```



```

-----
SUB grid (flag%) STATIC
-----
SHARED xstep&, ystep&, xorigin&, yorigin&, gridon%
SHARED sx1#, sx2#, sy1#, sy2#
IF gridon% THEN
    CALL winon

    LINE (sx1#, yorigin&)-(sx2#, yorigin&), 3 * flag%, , &HAAAA
    LINE (sx1#, yorigin&)-(sx2#, yorigin&), 6 * flag%, , &H5555
    LINE (xorigin&, sy1#)-(xorigin&, sy2#), 3 * flag%, , &HAAAA
    LINE (xorigin&, sy1#)-(xorigin&, sy2#), 6 * flag%, , &H5555

    x1& = FIX((sx1# - xorigin&) / xstep&) * xstep& + xorigin&
    x2& = FIX((sx2# - xorigin&) / xstep&) * xstep& + xorigin&
    y1& = FIX((sy1# - yorigin&) / ystep&) * ystep& + yorigin&
    y2& = FIX((sy2# - yorigin&) / ystep&) * ystep& + yorigin&

    stepsx# = (x2& - x1&) / xstep&
    stepsy# = (y2& - y1&) / ystep&

    IF stepsy# < 70 AND stepsx# < 128 THEN
        FOR i& = x1& TO x2& STEP xstep&
            FOR j& = y1& TO y2& STEP ystep&
                LINE (i&, j&)-(i&, j&), 7 * flag%
            NEXT
            IF INKEY$ = CHR$(27) THEN EXIT FOR
        NEXT
    END IF
END IF
END SUB

```

```
!-----  
SUB header STATIC  
!-----  
SHARED child#(), level%, editing%, fileopen%  
DIM rec#(10)  
FOR i% = 1 TO 8  
    rec#(i%) = child#(level%, i%)  
NEXT  
irec# = 0!  
CALL writerec(irec#, rec#(), 0)  
END SUB
```

```

-----
SUB label (sel#(), scl&, l$) STATIC
-----
SHARED xpixel%, ypixel%, layer%
DIM rec#(10)
CLOSE 4
CALL findfile("sd.fnt", irtn%, dir$)
IF irtn% THEN
    LOCATE 1, 1
    PRINT error$(irtn%); " - "; dir$; "sd.fnt"; CHR$(6);
ELSE
    OPEN "R", 4, dir$ + "sd.fnt", 8
    l$ = short$(l$)
    sel#(1) = 0
    sel#(2) = 0
    sel#(5) = 8 * LEN(l$) * scl&
    sel#(6) = 8 * scl&
    sel#(8) = 1
    n% = 0
    CALL editset(10, jrec#, rec#(), sel#(), 1!, 1!)
    CALL winon
    x# = sel#(3)
    y# = sel#(4)
    rec#(5) = 0
    rec#(6) = scl&
    FOR i% = 1 TO LEN(l$)
        c% = ASC(MID$(l$, i%, 1))
        FIELD #4, 8 AS letter$
        GET #4, c% - 31
        FOR y% = 0 TO 7
            c% = ASC(MID$(letter$, y% + 1, 1))
            FOR x% = 0 TO 7
                bit% = (-1) ^ FIX(c% / (2 ^ x%))
                SELECT CASE bit%
                    CASE -1
                        IF rec#(5) = 0 THEN
                            rec#(1) = layer%
                            rec#(2) = 0
                            rec#(3) = x# + x% * scl&
                            rec#(4) = y# + y% * scl&
                        END IF
                        rec#(5) = scl& + rec#(5)
                    CASE ELSE
                        IF rec#(5) THEN
                            n% = n% + 1
                            rec#(7) = 0
                            CALL editset(11, irec#, sel#(), rec#(), 1!, 1!)
                        END IF
                        rec#(5) = 0
                    END SELECT
            NEXT
        NEXT
        x# = x# + 8 * scl&
    NEXT
    sel#(2) = n%
    sel#(4) = sel#(4) + scl&
    sel#(5) = sel#(5) - scl&
    sel#(6) = sel#(6) - scl&
    CALL editset(20, jrec#, rec#(), sel#(), 1!, 1!)
    LOCATE 1, 1: PRINT "Text Consists of "; n%; " Rectangles";
END IF
END SUB

```

```
!-----  
SUB layers STATIC  
!-----  
STATIC l$  
SHARED layer%  
DO  
    CALL prompt("Set Default Layer to (1-6)? ", l$, 1, c%)  
    IF c% <> 27 THEN layer% = INT(VAL(l$))  
LOOP UNTIL INSTR(1, "123456", CHR$(layer% + 48)) OR c% = 27  
END SUB
```

```

-----
SUB loadfile (irec#, rec#()) STATIC
-----
SHARED level%, fileopen%, plotted%, child#(), editlevel%, editing%, needspacking%
c$ = "N"
needspacking% = 0
irtn% = 1
jrtn% = 1
c% = 0
WHILE jrtn% AND UCASE$(c$) <> "Y" AND c% <> 27
    CALL prompt("Saw Draw Cell Name? ", file$, 8, c%)
    SELECT CASE c%
    CASE 27
    CASE ELSE
        IF short$(file$) = "" THEN file$ = " "
        IF INSTR(1, file$, "**") > 0 OR INSTR(1, file$, "?") > 0 THEN
            CALL filesearch(short$(file$), file$, "CEL", c%)
            file$ = LEFT$(file$, INSTR(1, file$ + ".", ".") - 1)
        END IF
        IF c% <> 27 THEN
            CALL filestatus(packstr$(file$ + ".cel"), jrtn%, d$)
            IF jrtn% THEN
                c$ = "y"
                p$ = "Cell Not Found: " + d$ + short$(file$) + ", Create it? "
                CALL prompt(p$, c$, 1, c%)
                c$ = UCASE$(c$)
            END IF
        END IF
    END SELECT
WEND
IF (c% <> 27) THEN
    CLOSE 3
    IF fileopen% = 1 THEN CLOSE : KILL "*.rnd"
    cellname$ = packstr$(file$ + ".rnd")
    CALL filestatus(cellname$, irtn%, d$)
    SELECT CASE irtn%
    CASE 0: LOCATE 1, 1
        saved% = 0
        PRINT "Recovered edits on cell: "; file$; " Press any key to continue..."
        BEEP
        a$ = INPUT$(1)
    CASE ELSE
        SHELL "copy " + packstr$(file$ + ".cel") + " " + packstr$(file$ + ".rnd") + "/b > n"
    END SELECT
    irec# = setrec%(irec#, -1)
    level% = 0
    plotted% = 0
    fileopen% = 1
    editing% = 1
    editlevel% = 0
    CALL openfile(cellname$, rec#())
    SELECT CASE LCASE$(c$)
    CASE "y"
        FOR i% = 1 TO 7
            child#(level%, i%) = 0
        NEXT
    CASE ELSE
    END SELECT
    child#(level%, 8) = CVD(LEFT$(UCASE$(file$) + SPACE$(8), 8))
    CALL header
    CALL viewport
END IF
END SUB

```

```

-----
SUB mergecell (rec#(), sel#()) STATIC
-----
SHARED child#(), sminx#, sminy#, level%, editlevel%
DIM inrec AS sdrecord
c% = 0
irtn% = 1
p$ = "Merge Saw Draw Cell? "

WHILE c% <> 27 AND irtn% <> 0
    CALL prompt(p$, file$, 8, c%)
    SELECT CASE c%
    CASE 27
    CASE ELSE
        IF short$(file$) = "" THEN file$ = "*"
        IF INSTR(1, file$, "***") > 0 OR INSTR(1, file$, "?") > 0 THEN
            CALL filesearch(short$(file$), cellname$, "CEL", c%)
        END IF
        CALL filestatus(cellname$, irtn%, dir$)
    END SELECT
    p$ = error$(irtn%) + " Merge Saw Draw Cell? "
WEND

SELECT CASE c%
CASE 27
CASE ELSE
    CALL selplot(0)
    sminx# = 0
    sminy# = 0
    total# = setrec%(0#, 0)
    CALL recquery("Set Cell Position", sel#(), c%, 0, 1!, 1!)
    CLOSE 4
    OPEN "R", 4, cellname$, 40
END SELECT

irec# = 0
notdone% = 1
total% = 1

WHILE irec# <= total% AND c% <> 27
    GET #4, irec# + 1, inrec
    SELECT CASE irec#
    CASE 0: total% = inrec.r
        sminx# = inrec.x
        sminy# = inrec.y
        rec#(1) = 0
    CASE 1: krec# = jrec#
        rec#(1) = inrec.l
    CASE ELSE
        rec#(1) = inrec.l
    END SELECT
    rec#(2) = inrec.r
    rec#(3) = inrec.x
    rec#(4) = inrec.y
    rec#(5) = inrec.w
    rec#(6) = inrec.h
    rec#(7) = inrec.t
    rec#(8) = inrec.f
    rec#(9) = inrec.xo
    rec#(10) = inrec.yo
    CALL editssel(314, jrec#, sel#(), rec#(), 1!, 1!)
    irec# = irec# + 1
WEND

SELECT CASE c%
CASE 27
CASE ELSE

```

```
        CLOSE 4  
        CALL selrecs(krec#, rec#(), sel#())  
END SELECT  
  
END SUB
```



```

-----
SUB mirror (rec#(), iflag%) STATIC
-----
' mirror image of select rectangles in x or y
'
SHARED sminx#, sminy#, sw#, sh#
CALL corners(rec#(), x1#, y1#, x3#, y3#, x2#, y2#, x4#, y4#)
SELECT CASE iflag%
CASE 1      ' flip x
    xaxis# = sminx# + sw# / 2
    dx# = FIX((2 * xaxis# - x4#) - rec#(3))
    dy# = FIX(y4# - rec#(4))
CASE ELSE  ' flip y
    yaxis# = sminy# + sh# / 2
    dy# = FIX((2 * yaxis# - y2#) - rec#(4))
    dx# = FIX(x2# - rec#(3))
END SELECT

rec#(3) = rec#(3) + dx#
rec#(4) = rec#(4) + dy#

SELECT CASE rec#(1)
CASE 7: rec#(9) = rec#(9) + dx#
      rec#(10) = rec#(10) + dy#
CASE ELSE
END SELECT

rec#(7) = -rec#(7)

END SUB

```

```
'-----  
SUB moureset STATIC  
'-----  
SHARED xpixel%, ypixel%  
CALL mouse(0, mouseon%, cx%, dx%)           ' check mouse status  
CALL mouse(1, bx%, cx%, dx%)                 ' show mouse  
CALL mouse(2, bx%, cx%, dx%)                 ' hide mouse  
CALL mouse(7, 0, xpixel%, INT(80 * xpixel% - 1)) ' set x mouse limits  
CALL mouse(8, 0, 2 * ypixel%, INT(25 * ypixel% - 1)) ' set y mouse limits  
END SUB
```

```
!-----  
SUB onscreen (rec#(), test%) STATIC  
!-----  
SHARED sx1#, sx2#, sy1#, sy2#, xoff#, yoff#, toff#  
STATIC i%  
DIM lmts#(10)  
lmts#(3) = sx1# - xoff#  
lmts#(4) = sy1# - yoff#  
lmts#(5) = sx2# - sx1#  
lmts#(6) = sy2# - sy1#  
lmts#(7) = -toff#  
CALL within(rec#(), lmts#(), test%)  
END SUB
```

```

'-----
SUB opencell (irec#, rec#()) STATIC
'-----
SHARED current#(), level%, parent#(), path$
DIM dum#(10)
'
' Need to check for recursive entries
'
file$ = MKD$(rec#(8))
cellname$ = packstr$(file$ + ".rnd")
CALL filestatus(cellname$, irtn%, d$)
SELECT CASE irtn%
CASE 0
CASE ELSE
    CALL filestatus(packstr$(file$ + ".cel"), irtn%, d$)
    SELECT CASE irtn%
    CASE 0: SHELL "copy " + packstr$(file$ + ".cel") + " " + packstr$(file$ + ".rnd") + "/b > n
    CASE ELSE
        LOCATE 1, 1
        PRINT " Cell Not Found: "; file$; ", Creating as New ..."
        CLOSE 3
        OPEN "R", 3, cellname$, 40
        FOR i% = 1 TO 10
            SELECT CASE i%
            CASE IS < 3, IS > 6: dum#(i%) = 0
            CASE ELSE: dum#(i%) = rec#(i%)
            END SELECT
        NEXT
        CALL writerec(0#, dum#(), 1)
    END SELECT
END SELECT

CLOSE 3
current#(level%) = irec#
level% = level% + 1
FOR i% = 1 TO 10
    parent#(level%, i%) = rec#(i%)
NEXT
CALL openfile(cellname$, rec#())
irec# = 1

END SUB

```

```

'-----
SUB openfile (cellname$, rec#()) STATIC
'-----
  SHARED toff#, xoff#, yoff#, child#(), fileopen%, parent#(), editing%, level%
  CLOSE 3
  a$ = cellname$
  OPEN "R", 3, cellname$, 40
  CALL readrec(0#, rec#())
  i% = INSTR(1, cellname$, ".") - 1
  rec#(8) = CVD(LEFT$(cellname$, i%) + SPACE$(8 - i%))
  FOR i% = 1 TO 10
    child#(level%, i%) = rec#(i%)
  NEXT
  '
  ' solve for internal offsets for this level
  '
  xoff# = 0
  yoff# = 0
  toff# = 0
  i% = 0
  WHILE i% < level%
    i% = i% + 1
    xoff# = xoff# + FIX(parent#(i%, 9))
    yoff# = yoff# + FIX(parent#(i%, 10))
    toff# = toff# + parent#(i%, 7)
  WEND
END SUB

```

```

-----
SUB openwindow (row%, col%, num%, length%) STATIC
-----
SHARED level%, child#(), ifor2%, sx1#, sx2#, sy1#, sy2#, xoff#, yoff#
SHARED editlevel%
VIEW PRINT 1 TO 25

IF child#(level%, 2) AND child#(level%, 5) > 0 AND child#(level%, 6) > 0 THEN
    dc& = INT(length% * (sx2# - sx1#) / child#(level%, 5))
    c1& = INT(length% * (sx1# - child#(level%, 3) - xoff#) / child#(level%, 5)) - 0
    dr& = INT(num% * (sy2# - sy1#) / child#(level%, 6))
    r1& = INT(num% * (child#(level%, 4) + child#(level%, 6) - sy2# + yoff#) / child#(level%, 6))

    CALL validate(r1&, num% - 3&, 0&)
    CALL validate(c1&, length% - 3&, 0&)
    CALL validate(dr&, num% - 2 - r1&, 1&)
    CALL validate(dc&, length% - 2 - c1&, 1&)
ELSE
    c1& = 0: dc& = 0
    r1& = 0: dr& = 0
END IF

COLOR ifor2%
CALL clearwindow(row%, col%, num%, length%)
LOCATE row% - 1, col% - 1, 0: PRINT CHR$(213); STRING$(length%, 205); CHR$(184);
PRINT CHR$(31); CHR$(29); CHR$(219);

FOR i% = 1 TO num% - 2: PRINT CHR$(31); CHR$(29); CHR$(177); : NEXT

PRINT CHR$(31); CHR$(29); CHR$(219);
LOCATE row% + num%, col% - 1, 0
PRINT CHR$(212); CHR$(219); STRING$(length% - 2, 177); CHR$(219); CHR$(190);

LOCATE row% + num%, col%
FOR i% = 1 TO num%: PRINT CHR$(30); CHR$(29); CHR$(179); : NEXT

c$ = ""
FOR i% = 0 TO editlevel%
    c$ = c$ + packstr$(MKD$(child#(i%, 8))) + "\"
NEXT
c$ = RIGHT$(c$, 70)
c$ = LEFT$(c$, LEN(c$) - 1)
p$ = UCASE$( "\" Edit Path:" + c$ + " |")
LOCATE row% - 1, col% + (length% / 2) - (LEN(p$) / 2)
PRINT p$

LOCATE row% + num%, col% + c1& + 1, 0
PRINT STRING$(dc&, 176);

LOCATE row% + r1&, col% + length%, 0
FOR i% = 1 TO dr&: PRINT CHR$(31); CHR$(176); CHR$(29); : NEXT

PCOPY 0, 1
END SUB

```

```

-----
SUB pack (rec#(), del%) STATIC
-----
SHARED child#(), level%, seltop%, selbot%, blank$, deleted%
STATIC indexing%, index#, i%, irec#, jrec#, incr%, count%
DIM ind#(10)
IF child#(level%, 2) AND (del% = 1 OR seltop% > 0) THEN ' reindex file
irec# = 1
jrec# = 1
indexing% = 0
LOCATE 1, 1: PRINT blank$
SELECT CASE del%
CASE 1: CALL winon
      CALL selplot(0)
      PCOPY 0, 1
      dum% = setrec%(0#, 0)
      CALL winoff
      seltop% = child#(level%, 2)
      selbot% = 1
      notset% = 1
      LOCATE 1, 1: PRINT "Packing & Indexing File..."
      needspacking% = 0
CASE 0: deleted% = 0
      notset% = 0
      irec# = selbot%
      CALL readrec(irec#, rec#())
      SELECT CASE rec#(2)
      CASE IS > 0
        SELECT CASE rec#(1)
        CASE 0, 7
        CASE ELSE
          selbot% = rec#(2)
          irec# = selbot%
          CALL readrec(irec#, rec#())
          IF irec# + rec#(2) > seltop% THEN seltop% = irec# + rec#(2)
        END SELECT
      CASE ELSE
      END SELECT
      LOCATE 1, 1: PRINT "Indexing File...("; selbot%; "to"; seltop%; ")";
      notset% = 0
END SELECT

col% = POS(0)
irec# = selbot%
jrec# = selbot%

WHILE irec# <= seltop%
  CALL readrec(irec#, rec#())
  incr% = 0
  SELECT CASE indexing%
  CASE 1
    IF ind#(2) = count% OR (rec#(1) = 0 AND rec#(2) > 0) OR rec#(1) = 7 THEN
      ind#(2) = count%
      IF ind#(2) < 5 THEN ind#(2) = 0
      CALL writerec(index#, ind#(), 0)
      indexing% = 0
    END IF
  CASE 0
  END SELECT
  SELECT CASE rec#(1)
  CASE 0 ' deleted or index record (test repeats, skip or +1)
    SELECT CASE FIX(rec#(2))
    CASE 0
      SELECT CASE del%
      CASE 1: ind#(2) = ind#(2) - 1
              incr% = 0
      CASE 0: incr% = 1

```



```

        deleted% = deleted% + 1
    END SELECT
CASE ELSE
    incr% = 1
    indexing% = 1
    index# = jrec#
    count% = 0
    FOR i% = 1 TO 8: ind#(i%) = 0: NEXT
    ind#(2) = ABS(FIX(rec#(2)))
END SELECT
CASE 1, 2, 3, 4, 5, 6, 7
    SELECT CASE notset%
    CASE 1: notset% = 0
        FOR i% = 3 TO 6
            child#(level%, i%) = rec#(i%)
        NEXT
    CASE ELSE
    END SELECT
    CALL checkheader(irec#, rec#())
    deleted% = 0
    incr% = 1
    SELECT CASE indexing%
    CASE 1: SELECT CASE rec#(1)
        CASE 0, 7
        CASE ELSE: rec#(2) = index#
        END SELECT
        count% = count% + 1
        IF count% = 1 THEN ind#(3) = rec#(3): ind#(4) = rec#(4)
        CALL checkmins(rec#(), ind#(3), ind#(4), ind#(5), ind#(6))
        CALL checkdims(rec#(), ind#(3), ind#(4), ind#(5), ind#(6))
    CASE 0: SELECT CASE rec#(1)
        CASE 0, 7
        CASE ELSE: rec#(2) = 0
        END SELECT
    END SELECT
CASE ELSE
    SELECT CASE del%
    CASE 1: ind#(2) = ind#(2) - 1
        incr% = 0
    CASE 0: incr% = 1
    END SELECT
END SELECT
IF incr% THEN CALL writerec(jrec#, rec#(), 1)
irec# = irec# + 1
jrec# = jrec# + incr%
WEND
SELECT CASE indexing%
CASE 1: ind#(2) = count%
    IF ind#(2) < 5 THEN ind#(2) = 0
    CALL writerec(index#, ind#(), 0)
    indexing% = 0
CASE 0
END SELECT
LOCATE 1, 1: PRINT SPACE$(79);
SELECT CASE del%
CASE 1: child#(level%, 2) = jrec# - 1
CASE 0 ' if last records deleted about eof then remove them
    IF seltop% = child#(level%, 2) THEN
        child#(level%, 2) = child#(level%, 2) - deleted%
    END IF
END SELECT
CALL header
END IF

seltop% = 0
selbot% = 0

END SUB

```

```

-----
SUB plotall (k%) STATIC
-----
SHARED child#(), level%, editlevel%, editpath#(), explevel%, plotted%
SHARED sy1#, sy2#, sx1#, sx2#

DIM rec#(10), plotlev%(50)

CALL openwindow(3, 2, 22, 78)
CALL winon
CALL restart(irec#, rec#())
CALL editcell(rec#())
ecell# = child#(editlevel%, 8)
CALL restart(irec#, rec#())

CLS
CALL grid(1)
irec# = 1
bottom% = level%
notdone% = 1
plotted% = 1
plotlev%(0) = 1

SELECT CASE editlevel%
CASE 0: bright% = 1
CASE ELSE: bright% = 0
END SELECT

WHILE notdone% AND INKEY$ <> CHR$(27) AND child#(level%, 2) > 0
  CALL readrec(irec#, rec#())
  SELECT CASE rec#(1)
    CASE 7
      CALL plotrec(irec#, rec#(), bright%)
      IF level% < explevel% OR rec#(2) THEN
        CALL onscreen(rec#(), test%)
        SELECT CASE test%
          CASE 1: plotlev%(level%) = irec#
            CALL readrec(irec#, rec#())
            CALL opencell(irec#, rec#())
            IF rec#(8) = ecell# THEN
              bright% = 1
            ELSE
              bright% = 0
            END IF
            irec# = 0
          CASE ELSE
            END SELECT
        END IF
      CASE 0 'deleted or index record (test repeats, skip or +1)
        IF rec#(2) >= 1 THEN
          CALL onscreen(rec#(), test%)
          IF test% = 0 THEN
            irec# = irec# + rec#(2)
          ELSE
            rec#(1) = 8
            CALL plotrec(irec#, rec#(), 0)
          END IF
        END IF
      CASE ELSE
        CALL plotrec(irec#, rec#(), bright%)
      END SELECT
  END SELECT
  IF irec# >= child#(level%, 2) THEN
    notdone% = level% - bottom%
    IF level% > bottom% THEN
      CALL closecell(irec#, rec#())
      CALL readrec(0#, rec#())
      IF rec#(8) = ecell# THEN

```

```
                bright% = 1
            ELSE
                bright% = 0
            END IF
        END IF
    END IF
    irec# = irec# + 1
WEND
PCOPY 0, 1
CALL restart(irec#, rec#())
CALL editcell(rec#())
END SUB
```

```

'-----
SUB plotrec (irec#, rec#(), i%) STATIC
'-----
STATIC xbl#, ybl#, xtr#, ytr#, xtl#, ytl#, xbr#, ybr#
SHARED zfactor&
SELECT CASE zfactor&
CASE IS > 200
    CALL onscreen(rec#(), test%)
    SELECT CASE test%
    CASE 1: CALL addoff(rec#())
            CALL checkwraparound(rec#())
    CASE ELSE
    END SELECT
CASE ELSE
    CALL addoff(rec#())
    test% = 1
END SELECT
IF test% = 1 THEN
    CALL corners(rec#(), xbl#, ybl#, xtr#, ytr#, xtl#, ytl#, xbr#, ybr#)
    f% = FIX(rec#(1))
    SELECT CASE i%
    CASE 1: style% = &HFFFF
    CASE 0: SELECT CASE f%
            CASE 8: style% = &HAAAA
            CASE ELSE: style% = &H5555
        END SELECT
    END SELECT
    ON ERROR GOTO errorline
    LINE (xbl#, ybl#)-(xtl#, ytl#), f%, , style%
    LINE (xtl#, ytl#)-(xtr#, ytr#), f%, , style%
    LINE (xtr#, ytr#)-(xbr#, ybr#), f%, , style%
    LINE (xbr#, ybr#)-(xbl#, ybl#), f%, , style%
    ON ERROR GOTO 0
END IF
CALL suboff(rec#())
END SUB

```

```
!-----  
SUB readrec (irec#, rec#()) STATIC  
!-----  
SHARED fileopen%  
DIM inrec AS sdrecord  
SELECT CASE fileopen%  
CASE 1: GET #3, irec# + 1, inrec  
    rec#(1) = inrec.l  
    rec#(2) = inrec.r  
    rec#(3) = inrec.x  
    rec#(4) = inrec.y  
    rec#(5) = inrec.w  
    rec#(6) = inrec.h  
    rec#(7) = inrec.t  
    rec#(8) = inrec.f  
    rec#(9) = inrec.xo  
    rec#(10) = inrec.yo  
CASE ELSE  
END SELECT  
END SUB
```

```
!-----  
SUB restart (irec#, rec#()) STATIC  
!-----  
SHARED level%  
DO  
    CALL closecell(irec#, rec#())  
LOOP UNTIL level% = 0  
irec# = 0  
CALL readrec(irec#, rec#())  
END SUB
```

```

!-----
SUB rotate (rec#(), x#, y#, t#) STATIC
!-----
STATIC dx#, dy#, r#, dt#
IF t# THEN
    CALL checktheta(rec#())
    dx# = rec#(3) - x#
    dy# = rec#(4) - y#
    r# = SQR(dx# ^ 2 + dy# ^ 2)
    SELECT CASE rec#(1)
    CASE 7: SELECT CASE (rec#(7) + t#)
        CASE IS > (pi# / 4!): t2# = 0
        CASE ELSE: t2# = t#
    END SELECT
    CASE ELSE: t2# = t#
    END SELECT
    dt# = arctan#(dx#, dy#) + t2#
    rec#(3) = r# * COS(dt#) + x#
    rec#(4) = r# * SIN(dt#) + y#
    rec#(7) = rec#(7) + t2#
    CALL checktheta(rec#())
END IF
END SUB

```

```

-----
SUB savecell (rec#(), c%) STATIC
-----
SHARED saved%, fileopen%, child#()
c% = 0
SELECT CASE saved%
CASE 0: a$ = "Y"
    WHILE a$ <> "y" AND a$ <> "n" AND c% <> 27
        CALL prompt("All Cells Not Saved, Save them now? ", a$, 1, c%)
        a$ = LCASE$(a$)
    WEND
    SELECT CASE c%
    CASE 27
    CASE ELSE
        SELECT CASE LCASE$(a$)
        CASE "y"
            CALL restart(irec#, rec#())
            CALL pack(rec#(), 1)
            CLOSE
            SHELL "copy *.rnd *.cel/b > nul "
            cellname$ = RTRIM$(MKD$(child#(0, 8))) + ".rnd"
            CALL openfile(cellname$, rec#())
        CASE "n"
        END SELECT
    END SELECT
CASE ELSE
END SELECT
END SUB

```



```

'-----
SUB sawcad (rec#(), sel#(), sx, st) STATIC
'-----
SHARED level%, child#(), layer%, format$
STATIC dx#, dy#
LINE INPUT #4, l$
WHILE INSTR(1, LCASE$(l$), "isnum") = 0 AND EOF(4) <> -1
    LINE INPUT #4, l$
WEND
IF EOF(4) <> -1 THEN LINE INPUT #4, l$
rec#(1) = layer%
rec#(2) = 0
sel#(1) = 0
sel#(2) = 0
sel#(5) = 0
sel#(6) = 0
notset% = 1
count% = 0
WHILE EOF(4) <> -1
    FOR i% = 1 TO 6
        SELECT CASE i%
            CASE 1, 2, 3, 4: INPUT #4, s#: rec#(i% + 2) = FIX(s# * sx)
            CASE 5: INPUT #4, rec#(7): rec#(7) = (rec#(7) * st * pi#) / 180!
            CASE 6
                INPUT #4, s#
                INPUT #4, dx#: dx# = dx# * sx
                INPUT #4, dy#: dy# = dy# * sx
                WHILE (s# > 0)
                    sel#(2) = sel#(2) + 1
                    rec#(1) = layer%
                    rec#(2) = 0
                    IF notset% THEN
                        xmin# = rec#(3)
                        ymin# = rec#(4)
                        notset% = 0
                    END IF
                    SELECT CASE count% ' divide into sets of 100 indexed records
                        CASE IS > 0
                            count% = count% - 1
                        CASE ELSE
                            count% = 99
                            sel#(1) = 0
                            sel#(2) = 100
                            CALL editssel(10, irec#, rec#(), sel#(), 1!, 1!)
                    END SELECT
                    IF rec#(5) > 0 AND rec#(6) > 0 THEN
                        CALL editssel(10, irec#, sel#(), rec#(), 1!, 1!)
                    END IF
                    rec#(3) = rec#(3) + dx#
                    rec#(4) = rec#(4) + dy#
                    s# = INT(s# - 1)
                WEND
            CASE ELSE
                WEND
        END SELECT
    NEXT
WEND
END SUB

```

```

|-----
SUB scale (rec#(), sx, sy) STATIC
|-----
SHARED sminx#, sminy#
CALL corners(rec#(), x1#, y1#, x3#, y3#, x2#, y2#, x4#, y4#)
dt# = rec#(7)
rec#(3) = (rec#(3) - sminx#) * sx + sminx#
rec#(4) = (rec#(4) - sminy#) * sy + sminy#
SELECT CASE rec#(1)
CASE 7
CASE ELSE
    x# = (x3# - x1#) * sx
    y# = (y3# - y1#) * sy
    rec#(5) = rec#(5) * SQR((SIN(dt#) * sy) ^ 2 + (COS(dt#) * sx) ^ 2)
    rec#(6) = rec#(6) * SQR((COS(dt#) * sy) ^ 2 + (SIN(dt#) * sx) ^ 2)
    rec#(7) = arctan#(x#, y#) - arctan#(rec#(5), rec#(6))
END SELECT
END SUB

```

```

-----
SUB selcopy (sel#(), rec#(), flag%) STATIC
-----
STATIC i%
SHARED level%, child#(), sminx#, sminy#, sw#, sh#

c% = 0
IF setrec%(s#, -1) THEN
    krec# = child#(level%, 2) + 1
    IF flag% THEN CALL reccopy("Copy/Rotate Selections to Coordinates?", sel#(), c%, 0, 1!, 1!)
    IF c% <> 27 THEN
        IF flag% THEN CALL selplot(0)
        i% = 1
        count% = 100
        WHILE i% <= setrec%(s#, -1)
            IF count% > 99 THEN
                rec#(1) = 0
                rec#(2) = 100
                CALL editssel(10, jrec#, sel#(), rec#(), 1!, 1!)
                count% = 0
            END IF
            irec# = getrec%(i%)
            CALL readrec(irec#, rec#())
            SELECT CASE flag%
            CASE 0: action% = 314
            CASE ELSE: action% = 14
            END SELECT
            count% = count% + 1
            CALL editssel(action%, irec#, sel#(), rec#(), 1!, 1!)
            i% = i% + 1
        WEND
        IF flag% THEN CALL selrecs(krec#, rec#(), sel#())
    END IF
END IF
END SUB

```

```

-----
SUB selerase (sel#(), rec#(), allplot%) STATIC
-----
STATIC i%, j%, s#
SHARED level%, child#(), sminx#, sminy#, sw#, sh#, explevel%
CALL selplot(-1)
i% = 1
j% = 0
WHILE i% <= setrec%(s#, -1)
    j% = j% + 1
    irec# = getrec%(i%)
    CALL readrec(irec#, rec#())
    SELECT CASE rec%(1)
    CASE 7: IF rec%(2) > 0 OR explevel% > level% THEN allplot% = 1
    CASE ELSE
    END SELECT
    CALL editssel(330, irec#, sel#(), rec#(), 1!, 1!)
    i% = i% + 1
WEND
LOCATE 1, 1: PRINT j%; " Records Deleted";
total% = setrec%(s#, 0)
END SUB

```

```

'-----
SUB selmove (sel#(), rec#(), allplot%) STATIC
'-----
STATIC t#, r#, dx#, dy#, i%, sx, sy
SHARED level%, child#(), sminx#, sminy#, sw#, sh#, editlevel%, explevel%
IF getrec#(0) THEN
    i% = 1
    c% = 0
    t# = sel#(7)
    CALL recmove(sel#(), c%, sx, sy)
    IF c% <> 27 THEN
        CALL selplot(-1)
        WHILE i% <= getrec#(0)
            irec# = getrec#(i%)
            CALL readrec(irec#, rec#())
            SELECT CASE rec#(1)
            CASE 7
                IF rec#(2) > 0 OR explevel% > edotlevel% THEN
                    allplot% = 1
                END IF
            CASE ELSE
            END SELECT
            CALL editssel(24, irec#, sel#(), rec#(), sx, sy)
            SELECT CASE i%
            CASE 1
                w# = 0
                h# = 0
                minx# = rec#(3)
                miny# = rec#(4)
            CASE ELSE
            END SELECT
            CALL checkmins(rec#(), minx#, miny#, w#, h#)
            CALL checkdims(rec#(), minx#, miny#, w#, h#)
            i% = i% + 1
        WEND

        sminx# = minx#
        sminy# = miny#
        sw# = w#
        sh# = h#
        sel#(7) = 0
    END IF
END IF
END SUB

```

```

!-----
SUB selplot (bright%) STATIC
!-----
SHARED child#(), sminx#, sminy#, sw#, sh#
DIM rec#(10)
rec#(3) = sminx#
rec#(4) = sminy#
rec#(5) = sw#
rec#(6) = sh#
CALL onscreen(rec#(), test%)
idum% = getrec#(0)
WHILE idum% > 0 AND test% AND INKEY$ = ""
    irec# = getrec#(idum%)
    CALL readrec(irec#, rec#())
    IF (rec#(1) > 0) THEN
        IF bright% >= 0 THEN
            rec#(1) = rec#(1) + bright% * 8
        ELSE
            rec#(1) = 0
        END IF
        CALL plotrec(irec#, rec#(), 1)
    END IF
    idum% = idum% - 1
WEND
PCOPY 0, 1
END SUB

```

```

-----
SUB selrecs (jrec#, rec#(), sel#()) STATIC
-----
SHARED child#(), level%, sminx#, sminy#, sw#, sh#, toff#, recs%, cells%
STATIC w%, total% ', recs%, cells%
total% = setrec%(total#, 0)
cells% = 0
recs% = 0
FOR irec# = jrec# TO child#(level%, 2)
  CALL readrec(irec#, rec#())
  SELECT CASE rec#(1)
    CASE 1, 2, 3, 4, 5, 6, 7
      total% = setrec%(irec#, 1)
      SELECT CASE total%
        CASE 1
          sminx# = rec#(3)
          sminy# = rec#(4)
          sw# = 0
          sh# = 0
        CASE ELSE
          END SELECT
      CALL checkmins(rec#(), sminx#, sminy#, sw#, sh#)
      CALL checkdims(rec#(), sminx#, sminy#, sw#, sh#)
      SELECT CASE rec#(1)
        CASE 7: cells% = cells% + 1
          rec#(2) = 0
          CALL writerec(irec#, rec#(), 0)
        CASE ELSE
          recs% = recs% + 1
        CASE ELSE
          END SELECT
      END SELECT
  CASE ELSE
    END SELECT
NEXT
LOCATE 1, 1
PRINT cells%; " Cells Selected; "; recs%; " Rectangles Selected ";
sel#(3) = sminx#
sel#(4) = sminy#
sel#(5) = sw#
sel#(6) = sh#
sel#(7) = toff#
END SUB

```

```

-----
SUB seteditpath (rec#(), sel#()) STATIC
-----
SHARED level%, editlevel%, editpath#(), blank$
i% = 0
irec# = 0
c$ = ""
c% = 0
SELECT CASE getrec#(0)
CASE 1: irec# = getrec#(1): c$ = "Y"
      CALL readrec(irec#, rec#())
CASE ELSE
      WHILE (i% < getrec#(0)) AND (c$ <> "Y") AND (c% <> 27)
        i% = i% + 1
        irec# = getrec#(i%)
        CALL readrec(irec#, rec#())
        IF rec#(1) = 7 THEN
          CALL prompt("More than one cell selected. Edit " + UCASE$(packstr$(MKD$(rec
            c$ = UCASE$(c$)
          END IF
        WEND
      END SELECT
IF c$ = "Y" AND (rec#(1) = 7) AND (c% <> 27) THEN
  CALL addoff(sel#())
  LOCATE 1, 1: PRINT LEFT$(" Current Edit Cell: " + MKD$(rec#(8)) + blank$, 78);
  editpath#(editlevel%) = irec#
  editlevel% = editlevel% + 1
  i% = setrec%(irec#, 0)
  CALL editcell(rec#())
  CALL suboff(sel#())
END IF
END SUB

```



```
-----  
FUNCTION setrec% (s#, i%) STATIC  
-----  
SHARED sel%()  
SELECT CASE i%  
CASE 0: sel%(0) = 0  
CASE 1: sel%(0) = sel%(0) + 1  
        sel%(sel%(0)) = INT(s#)  
CASE ELSE  
END SELECT  
setrec% = sel%(0)  
END FUNCTION
```

```

-----
SUB setwindow (sel#()) STATIC
-----
STATIC x#, y#, z$, dx#, dy#
SHARED sx1#, sx2#, sy1#, sy2#
CALL addoff(sel#())
x# = FIX((sx1# + sx2#) / 2)
y# = FIX((sy1# + sy2#) / 2)
z$ = "1"
CALL prompt("Zoom Factor? ", z$, 6, c%)
CALL suboff(sel#())
IF c% <> 27 THEN
    dx# = FIX((sx2# - sx1#) / (VAL(z$) * 2))
    dy# = FIX((sy2# - sy1#) / (VAL(z$) * 2))
    sx1# = x# - dx#
    sx2# = x# + dx#
    sy1# = y# - dy#
    sy2# = y# + dy#
END IF
END SUB

```

```

-----
SUB slct (sel#(), rec#(), add%) STATIC
-----
SHARED child#(), level%, sminx#, sminy#, sw#, sh#, toff#, cells%, recs%
STATIC w%, total% ' , recs%, cells%
SELECT CASE add%
CASE 0: CALL selplot(0)
        total% = setrec%(total#, 0)
        cells% = 0
        recs% = 0
CASE 1
END SELECT
irec# = 1
WHILE irec# <= child#(level%, 2)
    CALL readrec(irec#, rec#())
    SELECT CASE INT(rec#(1))
    CASE 0
        SELECT CASE INT(rec#(2))
        CASE 0: incr% = 1
        CASE ELSE: CALL within(rec#(), sel#(), w%)
                    IF w% = 0 THEN incr% = INT(rec#(2)) + 1 ELSE incr% = 1
        END SELECT
    CASE 1, 2, 3, 4, 5, 6, 7
        CALL within(rec#(), sel#(), w%)
        SELECT CASE add%
        CASE 1: SELECT CASE w%
                CASE 1: FOR i% = 1 TO getrec#(0)
                        IF irec# = getrec#(i%) THEN
                            w% = 0
                            EXIT FOR
                        END IF
                NEXT
                CASE 0
                END SELECT
        CASE 0
        END SELECT
        IF w% THEN
            total% = setrec%(irec#, 1)
            SELECT CASE total%
            CASE 1: sminx# = rec#(3): sminy# = rec#(4)
                    sw# = 0: sh# = 0
            CASE ELSE
            END SELECT
            CALL checkmins(rec#(), sminx#, sminy#, sw#, sh#)
            CALL checkdims(rec#(), sminx#, sminy#, sw#, sh#)
            SELECT CASE rec#(1)
            CASE 7: cells% = cells% + 1
            CASE ELSE: recs% = recs% + 1
            END SELECT
        END IF
        incr% = 1
    CASE ELSE
        incr% = 1
    END SELECT
    irec# = irec# + incr%
WEND
LOCATE 1, 1
PRINT cells%; " Cells Selected; "; recs%; " Rectangles Selected ";
sel#(3) = sminx#
sel#(4) = sminy#
sel#(5) = sw#
sel#(6) = sh#
sel#(7) = 0
END SUB

```

```

-----
SUB steprepeat (sel#(), rec#()) STATIC
-----
'step and repeat rectangles with:
'
'   delta x
'   delta y
'   delta theta
'   delta width
'   delta height
'
'has no effect on cells' width or height
'

STATIC dt#, dw#, dh#, dx#, dy#, i%
SHARED level%, child#(), sminx#, sminy#, sw#, sh#
x# = sminx#
y# = sminy#
IF getrec#(0) THEN
  c% = 0
  dx# = sel#(3) - sminx#
  dy# = sel#(4) - sminy#
  dw# = sel#(5) - sw#
  dh# = sel#(6) - sh#
  SELECT CASE (dx# + dy# + dw# + dh#)
  CASE 0: dt# = arctan#(sh#, sw#)
  CASE ELSE: dt# = 0
  END SELECT
  CALL recstep(dx#, dy#, dw#, dh#, dt#, repeats%, c%)
  CALL winon
  s% = 0
  IF getrec#(0) <= 10 THEN
    sel#(1) = 0
    CALL editssel(10, jrec#, rec#(), sel#(), sx, sy)
  END IF
  IF c% <> 27 THEN
    FOR r% = 1 TO repeats%
      i% = 0
      IF getrec#(0) > 10 THEN
        s% = 0
        sel#(1) = 0
        CALL editssel(10, jrec#, rec#(), sel#(), sx, sy)
      END IF
      WHILE i% < getrec#(0)
        i% = i% + 1
        irec# = getrec#(i%)
        CALL readrec(irec#, rec#())
        t# = dt# * r%
        CALL rotate(rec#(), x#, y#, t#)
        rec#(3) = rec#(3) + r% * dx#
        rec#(4) = rec#(4) + r% * dy#
        SELECT CASE rec#(1)
        CASE 7
          rec#(9) = rec#(9) + r% * dx#
          rec#(10) = rec#(10) + r% * dy#
        CASE ELSE
          rec#(5) = ABS(rec#(5) + r% * dw#)
          rec#(6) = ABS(rec#(6) + r% * dh#)
        END SELECT
        IF rec#(5) > 0 AND rec#(6) > 0 THEN
          CALL editssel(310, irec#, sel#(), rec#(), sx, sy)
          s% = s% + 1
        END IF
      WEND
      IF getrec#(0) > 10 THEN
        sel#(1) = 0
        sel#(2) = s%
        sel#(3) = sminx# + r% * dx#
        sel#(4) = sminy# + r% * dy#
      END IF
    NEXT r%
  END IF
END IF

```

```

        sel#(5) = sw# + r% * dw#
        sel#(6) = sh# + r% * dh#
        CALL editssel(20, jrec#, rec#(), sel#(), sx, sy)
    END IF
NEXT
IF getrec#(0) > 10 THEN
    sel#(1) = 0
    sel#(2) = s%
    sel#(3) = sminx#
    sel#(4) = sminy#
    sel#(5) = sw# + r% * dw#
    sel#(6) = sh# + r% * dh#
    CALL editssel(20, jrec#, rec#(), sel#(), sx, sy)
END IF
sel#(5) = sw#
sel#(6) = sh#
sel#(7) = 0
    END IF
END IF
END SUB

```

```
!-----  
SUB suboff (rec#()) STATIC  
!-----  
SHARED xoff#, yoff#, toff#, child#(), level%  
STATIC dx#, dy#, r#  
rec#(3) = rec#(3) - xoff#  
rec#(4) = rec#(4) - yoff#  
IF toff# THEN  
    CALL rotate(rec#(), child#(level%, 3), child#(level%, 4), -toff#)  
END IF  
END SUB
```

```
-----  
SUB unedit (rec#()) STATIC  
-----  
SHARED editlevel%  
SELECT CASE editlevel%  
CASE 0  
CASE ELSE  
    CALL editcell(rec#())  
    CALL pack(rec#(), 1)  
    CALL restart(irec#, rec#())  
    i% = setrec%(0#, 0)  
    editlevel% = editlevel% - 1  
    CALL editcell(rec#())  
END SELECT  
END SUB
```

```

-----
SUB unexpand (irec#, rec#()) STATIC
-----
'   rec#(2) set for expand - test% set to 1
'   rec#(2) unchanged      - test% unchanged
'
SHARED explevel%, editlevel%
CALL readrec(irec#, rec#())
IF rec#(1) = 7 THEN
    SELECT CASE rec#(2)
    CASE 1: rec#(2) = 0
        CALL writerec(irec#, rec#(), 0)
    CASE ELSE
    END SELECT
END IF
END SUB

```



```
-----  
SUB validate (v&, hi&, lo&) STATIC  
-----  
IF v& > hi& THEN  
    v& = hi&  
ELSEIF v& < lo& THEN  
    v& = lo&  
END IF  
END SUB
```

```

!-----
SUB viewport STATIC
!-----
SHARED child#(), sx1#, sx2#, sy1#, sy2#, level%, xoff#, yoff#, toff#
SHARED editlevel%
DIM rec#(10)
IF child#(editlevel%, 5) > 0 AND child#(editlevel%, 6) > 0 THEN
    FOR i% = 1 TO 8
        rec#(i%) = child#(editlevel%, i%)
    NEXT
    w# = 0
    h# = 0
    CALL checkdims(rec#(), rec#(3), rec#(4), w#, h#)
    dx# = (w# * .2)
    dy# = (h# * .2)
    sx1# = rec#(3) - dx#
    sy1# = rec#(4) - dy#
    sx2# = sx1# + dx# + w#
    sy2# = sy1# + dy# + h#
    CALL aspectratio(rec#(3), rec#(4), w#, h#)
ELSE
    sx1# = -160000
    sx2# = 160000
    sy1# = -100000
    sy2# = 100000
END IF
END SUB

```

```
!-----  
SUB winoff STATIC  
!-----  
WINDOW  
VIEW  
END SUB
```

```
!-----  
SUB winon STATIC  
!-----  
SHARED sx1#, sx2#, sy1#, sy2#, ypixel%, xpixel%  
IF (sx1# - sx2#) >= 0 OR (sy1# - sy2#) >= 0 THEN CALL viewport  
VIEW (xpixel%, ypixel% * 2)-(xpixel% * 79 - 1, ypixel% * 24 - 1)  
WINDOW (sx1#, sy2#)-(sx2#, sy1#)  
END SUB
```

```

SUB within (rec#(), lmt#(), test%) STATIC
DIM x1#(4), y1#(4), x2#(4), y2#(4)
STATIC x%, y%, i%, testx1#, testx2#, testy1#, testy2#, t#

SELECT CASE lmt#(7)
CASE 0
    t# = 0
CASE ELSE
    CALL checktheta(lmt#())
    t# = -lmt#(7)
    CALL rotate(rec#(), lmt#(3), lmt#(4), t#)
    lmt#(7) = 0
END SELECT
CALL corners(rec#(), x1#(1), y1#(1), x1#(3), y1#(3), x1#(2), y1#(2), x1#(4), y1#(4))
CALL corners(lmt#(), x2#(1), y2#(1), x2#(3), y2#(3), x2#(2), y2#(2), x2#(4), y2#(4))
x% = 0: y% = 0
FOR i% = 1 TO 4
    SELECT CASE rec#(7)
    CASE IS < 0
        SELECT CASE y%
        CASE 0
            IF (x1#(i%) >= x2#(1)) AND (x1#(i%) <= x2#(3)) THEN x% = 1
            IF (x2#(i%) <= x1#(1)) AND (x2#(i%) >= x1#(3)) THEN x% = 1
            IF (x2#(i%) >= x1#(1)) AND (x2#(i%) <= x1#(3)) THEN x% = 1
            IF (x1#(i%) <= x2#(1)) AND (x1#(i%) >= x2#(3)) THEN x% = 1
        CASE 1
        END SELECT
        SELECT CASE y%
        CASE 0
            IF (y1#(i%) >= y2#(2)) AND (y1#(i%) <= y2#(4)) THEN y% = 1
            IF (y2#(i%) >= y1#(2)) AND (y2#(i%) <= y1#(4)) THEN y% = 1
            IF (y1#(i%) <= y2#(2)) AND (y1#(i%) >= y2#(4)) THEN y% = 1
            IF (y2#(i%) <= y1#(2)) AND (y2#(i%) >= y1#(4)) THEN y% = 1
        CASE 1
        END SELECT
    CASE ELSE
        SELECT CASE x%
        CASE 0
            IF (x1#(i%) >= x2#(2)) AND (x1#(i%) <= x2#(4)) THEN x% = 1
            IF (x2#(i%) >= x1#(1)) AND (x2#(i%) <= x1#(3)) THEN x% = 1
            IF (x1#(i%) <= x2#(2)) AND (x1#(i%) >= x2#(4)) THEN x% = 1
            IF (x2#(i%) <= x1#(1)) AND (x2#(i%) >= x1#(3)) THEN x% = 1
        CASE 1
        END SELECT
        SELECT CASE y%
        CASE 0
            IF (y1#(i%) >= y2#(1)) AND (y1#(i%) <= y2#(3)) THEN y% = 1
            IF (y2#(i%) >= y1#(2)) AND (y2#(i%) <= y1#(4)) THEN y% = 1
            IF (y1#(i%) <= y2#(1)) AND (y1#(i%) >= y2#(3)) THEN y% = 1
            IF (y2#(i%) <= y1#(2)) AND (y2#(i%) >= y1#(4)) THEN y% = 1
        CASE 1
        END SELECT
    END SELECT
NEXT
SELECT CASE t#
CASE 0
CASE ELSE
    t = -t#
    CALL rotate(rec#(), lmt#(3), lmt#(4), t#)
    lmt#(7) = t#
END SELECT
test% = x% * y%
END SUB

```

```

!-----
SUB writerec (irec#, rec#(), check%) STATIC
!-----
SHARED fileopen%, child#(), level%
DIM outrec AS sdrecord
SELECT CASE fileopen%
CASE 1: outrec.l = INT(rec#(1))
        outrec.r = INT(rec#(2))
        outrec.x = FIX(rec#(3))
        outrec.y = FIX(rec#(4))
        outrec.w = FIX(rec#(5))
        outrec.h = FIX(rec#(6))
        IF outrec.w < 1 THEN outrec.w = 1
        IF outrec.h < 1 THEN outrec.h = 1
        outrec.t = rec#(7)
        outrec.f = rec#(8)
        outrec.xo = FIX(rec#(9))
        outrec.yo = FIX(rec#(10))
        PUT #3, irec# + 1, outrec
CASE ELSE
END SELECT
END SUB

```

```
!-----  
SUB zoom (rec#()) STATIC  
!-----  
SHARED sx1#, sx2#, sy1#, sy2#  
CALL winoff  
CALL addoff(rec#())  
sx1# = rec#(3): sx2# = rec#(3) + rec#(5)  
sy1# = rec#(4): sy2# = rec#(4) + rec#(6)  
CALL suboff(rec#())  
CALL winon  
END SUB
```

```

DECLARE FUNCTION insidebox% (x1#, x2#, y1#, y2#, x%, y%)
DECLARE FUNCTION arctan# (x#, y#)
DECLARE SUB maxof (v%, v1%, v2%)
DECLARE SUB plotall (k%)
DECLARE SUB header ()
DECLARE SUB checkheader (irec#, rec#())
DECLARE SUB editset (action%, irec#, sel#(), rec#(), sx!, sy!)
DECLARE SUB filesearch (filter$, cellname$, ext$, c%)
DECLARE SUB validate (v%, hi%, lo%)
DECLARE SUB pack (rec#(), del%)
DECLARE SUB openfile (cellname$, rec#())
DECLARE SUB selectfile (dir#(), ext#(), total%, file$, c%)
DECLARE SUB reclabel (title$, sel#(), pixel%, c%)
DECLARE SUB clearwindow (row%, col%, num%, length%)
DECLARE FUNCTION getrec# (i%)
DECLARE FUNCTION short$ (s$)
DECLARE FUNCTION packstr$ (s$)
DECLARE FUNCTION mki% (h%, l%)
DECLARE FUNCTION hbyte% (i%)
DECLARE FUNCTION lbyte% (i%)
DECLARE FUNCTION directory$ ()
DECLARE FUNCTION timmer# ()
DECLARE SUB cursor (c$, insert%)
DECLARE SUB label (sel#(), s$, l$)
DECLARE SUB selrecs (jrec#, rec#(), sel#())
DECLARE SUB recquery (title$, sel#(), c%, test%, sx!, st!)
DECLARE SUB checktheta (rec#())
DECLARE SUB readrec (irec#, rec#())
DECLARE SUB setbox (button%, sel#(), rec#(), cursorx%, cursory%, c%)
DECLARE SUB evaluate (c%, sel#(), rec#(), irec#, cursorx%, cursory%)
DECLARE SUB mouse (ax%, bx%, cx%, dx%)
DECLARE SUB fillbox (l$, bit1%, bit2%)
DECLARE SUB outlinebox (l$, bit1%, bit2%)
DECLARE SUB inverse (row%, col%, length%)
DECLARE SUB graphwindow (row%, col%, num%, length%, action%)
DECLARE SUB menu (mnu$, mnu%, length%, row%, col%, ians%, c%)
DECLARE SUB plotrec (irec#, rec#(), i%)
DECLARE SUB writerec (irec#, rec#(), check%)
DECLARE SUB addrec (irec#, rec#(), check%)
DECLARE SUB seteditpath (rec#(), sel#())
DECLARE SUB unexpand (irec#, rec#())
DECLARE SUB expand (irec#, rec#())
DECLARE SUB unedit (rec#())
DECLARE SUB printsetup (sel#(), r%, printer%, dpix%, dpiy%, dx%, dy%, l$, c%, row%, col%)
DECLARE SUB laser (sel#(), dy%, l$, dpix%, col%)
DECLARE SUB epson (sel#(), dy%, l$, dpix%, col%)
DECLARE SUB printers (printer%, dpix%, dpiy%, c%)
DECLARE FUNCTION units# (v$)
DECLARE FUNCTION value$ (v#)
DECLARE SUB addoff (rec#())
DECLARE SUB rotate (rec#(), x#, y#, t#)
DECLARE SUB setbit (byte%, bit%, l$)
DECLARE SUB setbyte (byte%, l$)
DECLARE SUB setlines (rec#(), lineseg&(), m!(), b&())
DECLARE SUB within (rec#(), lmt#(), test%)
DECLARE SUB suboff (rec#())
DECLARE SUB box (row%, col%, num%, length%)
DECLARE SUB getline (v$, lv%, c%)
DECLARE SUB switch (f%, c%)
DECLARE SUB help (i%)
DECLARE SUB errors (a%)
DECLARE FUNCTION radians# (d#)
DECLARE FUNCTION degrees# (r#)
DECLARE FUNCTION error$ (ierr%)
DECLARE SUB winoff ()
DECLARE SUB filestatus (file$, ax%, dir$)
DECLARE SUB prompt (p$, v$, i%, c%)
DECLARE SUB qdump (title$, dx#, dy#, sx, st, c%)

```



```

DECLARE SUB query (title$, opts%, c%)
DECLARE SUB offset (rec#())
DECLARE SUB findfile (file$, ax%, d$)
DECLARE SUB corners (rec#(), xbl#, ybl#, xtr#, ytr#, xtl#, ytl#, xbr#, ybr#)
DECLARE SUB rotate (rec#(), x#, y#, t#)
DECLARE SUB checktheta (rec#())
DECLARE SUB readrec (irec#, rec#())
DECLARE SUB grid (iflag%)
DECLARE SUB openwindow (row%, col%, num%, length%)
DECLARE SUB restart (irec#, rec#())
DECLARE SUB winon ()
DECLARE SUB opencell (irec#, rec#())
DECLARE SUB closecell (irec#, rec#())

```

```

'=====
'PROGRAM MODULE SDIO - SawDraw version 1.00, User I/O Routines
'=====

```

```
TYPE regtype
```

```

    ax AS INTEGER
    bx AS INTEGER
    cx AS INTEGER
    dx AS INTEGER
    bp AS INTEGER
    si AS INTEGER
    di AS INTEGER
    flags AS INTEGER
    ds AS INTEGER
    es AS INTEGER

```

```
END TYPE
```

```
TYPE sdrecord
```

```

    l AS INTEGER
    r AS INTEGER
    x AS LONG
    y AS LONG
    w AS LONG
    h AS LONG
    t AS SINGLE
    f AS DOUBLE
    xo AS LONG
    yo AS LONG

```

```
END TYPE
```

```

COMMON SHARED current#(), parent#(), child#(), explevel%, level%, layer%
COMMON SHARED p$(), v#(), l%(), t%(), ifor1%, ifor2%, ifor3%, fileopen%
COMMON SHARED xoff#, yoff#, toff#, blank$, sminx#, sminy#, sw#, sh#
COMMON SHARED xstep&, ystep&, xorigin&, yorigin&, gridon%, saved%, editing%
COMMON SHARED mouseon%, xpixel%, ypixel%, wavelength&, cells%, recs%
COMMON SHARED sx1#, sx2#, sy1#, sy2#, editlevel%, plotted%, zfactor&

```

```
CONST pi# = 3.1415926535#
```

```
DEF FNrow% (y%)
```

```

FNrow% = INT(y% / ypixel%) + 1
END DEF

```

```
DEF FNcol% (x%)
```

```

FNcol% = INT(x% / xpixel%) + 1
END DEF

```

```
'END SUB
```

```

!-----
SUB aspectratio (x#, y#, w#, h#) STATIC
!-----
DIM rec$(10)
FOR i% = 1 TO 8
    rec$(i%) = child$(editlevel%, i%)
NEXT
centerx# = xoff# + x# + INT(w# / 2) * COS(toff#) + INT(h# / 2) * SIN(toff#)
centery# = yoff# + y# + INT(h# / 2) * COS(toff#) + INT(w# / 2) * SIN(toff#)
dx# = ABS(sx2# - sx1#) / 2
dy# = ABS(sy2# - sy1#) / 2
ratio# = 400! / 640!
IF (dy# / dx#) > ratio# THEN
    dx# = (dy# / ratio#)
ELSE
    dy# = (dx# * ratio#)
END IF
sx1# = centerx# - dx#
sx2# = centerx# + dx#
sy1# = centery# - dy#
sy2# = centery# + dy#
END SUB

```

```
'-----  
SUB box (row%, col%, num%, length%) STATIC  
'-----  
SHARED ifor3%  
COLOR ifor3%  
CALL clearwindow(row%, col%, num%, length%)  
LOCATE row% - 1, col% - 1, 0: PRINT CHR$(213); STRING$(length%, 205); CHR$(184);  
FOR i% = 1 TO num%: PRINT CHR$(31); CHR$(29); CHR$(179); : NEXT  
LOCATE row% + num%, col% - 1, 0: PRINT CHR$(212); STRING$(length%, 205); CHR$(190);  
LOCATE row% + num%, col%  
FOR i% = 1 TO num%: PRINT CHR$(30); CHR$(29); CHR$(179); : NEXT  
END SUB
```

```

-----
SUB checkwraparound (rec#()) STATIC
-----
IF rec#(7) = 0 THEN
  IF rec#(4) < sy1# THEN
    dy# = sy1# - rec#(4)
    rec#(4) = sy1#
    rec#(6) = rec#(6) - dy#
  END IF
  IF rec#(4) + rec#(6) > sy2# THEN
    rec#(6) = sy2# - rec#(4)
  END IF
  IF rec#(3) < sx1# THEN
    dx# = sx1# - rec#(3)
    rec#(3) = sx1#
    rec#(5) = rec#(5) - dx#
  END IF
  IF rec#(3) + rec#(5) > sx2# THEN
    rec#(5) = sx2# - rec#(3)
  END IF
END IF
END SUB

```

```
-----  
SUB clearwindow (row%, col%, num%, length%) STATIC  
-----  
STATIC x1%, y1%, x2%, y2%  
CALL winoff  
x1% = (col% - 1) * xpixel%: x2% = (col% + length% - 1) * xpixel% - 1  
y1% = (row% - 1) * ypixel%: y2% = (row% + num% - 1) * ypixel% - 1  
VIEW (x1%, y1%)-(x2%, y2%)  
CLS 1  
VIEW  
END SUB
```

```

'-----
SUB cursor (c$, insert%) STATIC
'-----
'SHARED xpixel%, ypixel%
STATIC d%, d#
DIM csr%(200)
row% = CSRLIN
col% = POS(1) + 1
y1% = row% * ypixel% - (3 + 4 * insert%)
x1% = (col% - 2) * xpixel%
y2% = row% * ypixel% - 2
x2% = (col% - 1) * xpixel% - 1
WHILE c$ = ""
    GET (x1%, y1%)-(x2%, y2%), csr%
    LINE (x1%, y1%)-(x2%, y2%), 15, BF
    d% = 0: d# = timer#
    WHILE d% < 15 AND c$ = ""
        d% = INT(timer# - d#)
        c$ = INKEY$
    WEND
    PUT (x1%, y1%), csr%, PSET
    d% = 0: d# = timer#
    WHILE d% < 15 AND c$ = ""
        d% = INT(timer# - d#)
        c$ = INKEY$
    WEND
WEND
END SUB

```

```
'-----  
FUNCTION degrees# (r#) STATIC  
'-----  
degrees# = r# * 180# / pi#  
END FUNCTION
```

```

'-----
FUNCTION directory$ STATIC
'-----
STATIC i%, dir$
DIM inreg AS regtype, outreg AS regtype, f%(32)
FOR i% = 1 TO 31
    f%(i%) = 0
NEXT
inreg.ax = mki%(&H19, 0)
inreg.bx = 0
CALL interruptx(&H21, inreg, outreg)
dir$ = CHR$(lobyte%(outreg.ax) + 65) + ":\\"
inreg.ax = mki%(&H47, 0)
inreg.bx = 0
inreg.ds = VARSEG(f%(0))
inreg.si = VARPTR(f%(0))
CALL interruptx(&H21, inreg, outreg)
FOR i% = 0 TO 31
    dir$ = dir$ + MKI$(f%(i%))
NEXT
i% = 0
dir$ = packstr$(dir$) + "\\"
directory$ = dir$
END FUNCTION

```



```

'-----
SUB dump (format%) STATIC
'-----
DIM str32 AS STRING * 32
DIM rec#(10)
CALL restart(irec#, rec#())

c% = 0
irtn% = 64
CALL qdump("Specify Dump Information", dx#, dy#, sx, st, c%)

SELECT CASE format%
CASE 1: file$ = RTRIM$(MKD$(child#(0, 8))) + "." + LTRIM$(STR$(layer%))
CASE 2: file$ = RTRIM$(MKD$(child#(0, 8))) + ".CIF"
END SELECT

p$ = "Dump to File Named? "

irec# = 1
notdone% = 1
WHILE c% <> 27 AND irtn% = 64
    CALL prompt(p$, file$, 20, c%)
    CALL filestatus(file$, irtn%, dir$)
    SELECT CASE irtn%
    CASE 76: p$ = "Path Not Found: " + dir$ + ", File? "
    CASE 64: p$ = "Illegal File Name: " + file$ + ", File? "
    CASE ELSE
    END SELECT
WEND
IF c% <> 27 THEN
    LOCATE 1, 1: PRINT SPACE$(70);
    IF c% <> 27 THEN
        CLOSE 4
        isnum& = 0
        OPEN file$ FOR OUTPUT AS 4
        SELECT CASE format%
        CASE 1: 'sawcad
            PRINT #4, "ISNUM = "
            PRINT #4, "IREF = 1"
        CASE 2: 'cif
            PRINT #4, "L CM"
        END SELECT
        WHILE (irec# <= child#(level%, 2) OR level%) AND child#(level%, 2) > 0
            CALL readrec(irec#, rec#())
            LOCATE 1, 1: PRINT "Level: "; level%; " Total: "; isnum&;
            SELECT CASE rec#(1)
            CASE 7
                IF level% < expllevel% OR rec#(2) THEN
                    CALL opencell(irec#, rec#())
                    irec# = 0
                END IF
            CASE 1, 2, 3, 4, 5, 6
                IF rec#(1) = layer% THEN
                    CALL addoff(rec#())
                    FOR i% = 3 TO 6
                        rec#(i%) = rec#(i%) * sx
                    NEXT
                    rec#(3) = rec#(3) + dx#
                    rec#(4) = rec#(4) + dy#
                    SELECT CASE format%
                    CASE 1: ' sawcad
                        rec#(7) = degrees#(rec#(7)) * st
                        FOR i% = 3 TO 7
                            PRINT #4, rec#(i%);
                        NEXT
                        PRINT #4, 1; 0; 0
                    CASE 2: ' CIF

```

```

                                PRINT #4, "B "; rec#(5); rec#(6); rec#(3);
                                END SELECT
                                isnum& = isnum& + 1
                                END IF
                                CASE ELSE
                                END SELECT
                                IF irec# >= child#(level%, 2) AND level% THEN
                                    CALL closecell(irec#, rec#())
                                    irec# = irec# + 1
                                ELSE
                                    irec# = irec# + 1
                                END IF
                                WEND
                                CLOSE 4
                                SELECT CASE format%
                                CASE 1 ' SawCad
                                    OPEN "R", 4, file$, 32
                                    GET #4, 1, str32
                                    MID$(str32, 8, LEN(STR$(isnum&))) = STR$(isnum&)
                                    PUT #4, 1, str32
                                    CLOSE 4
                                CASE ELSE
                                END SELECT
                                END IF
                                END SUB

```

```

-----
SUB editrec (title$, sel#(), c%) STATIC
-----
CALL addoff(sel#())
t%(1) = 0: l%(1) = 20: v#(1) = FIX(sel#(3)): p$(1) = "Lower Left X Coordinate? "
t%(2) = 0: l%(2) = 20: v#(2) = FIX(sel#(4)): p$(2) = "Lower Left Y Coordinate? "
t%(3) = 0: l%(3) = 20: v#(3) = FIX(sel#(5)): p$(3) = "Width? "
t%(4) = 0: l%(4) = 20: v#(4) = FIX(sel#(6)): p$(4) = "Height? "
t%(5) = 2: l%(5) = 20: v#(5) = sel#(7) * 180! / pi#: p$(5) = "Rotate Through an Angle of? "

CALL query(title$, 5, c%)

IF c% <> 27 THEN
    sel#(3) = FIX(v#(1))
    sel#(4) = FIX(v#(2))
    sel#(5) = ABS(FIX(v#(3)))
    sel#(6) = ABS(FIX(v#(4)))
    IF (v#(3) < 0) THEN sel#(3) = sel#(3) + v#(3)
    IF (v#(4) < 0) THEN sel#(4) = sel#(4) + v#(4)
    sel#(7) = v#(5) * pi# / 180!
END IF
CALL suboff(sel#())
END SUB

```

```

SUB Epson (sel#(), dy&, l$, dpix%, col%) STATIC

DIM map$(7), b%(7), gbyte%(7)
r& = 1
n2% = FIX((8! * LEN(l$)) / 256!)
n1% = (8 * LEN(l$)) - 256 * n2%
record& = INT(sel#(6) / dy&) + 2
esc$ = CHR$(27)
l% = LEN(l$) / 8
SELECT CASE dpix%
CASE 60: g$ = "K"
CASE 120: g$ = "L"
CASE 240: g$ = "Z"
END SELECT
WHILE (r& < record&)
  PRINT #5, SPACE$(col%); esc$; "3"; CHR$(24); esc$; g$; CHR$(n1%); CHR$(n2%);
  FOR i% = 7 TO 0 STEP -1
    r& = r& + 1
    IF r& < record& THEN
      GET #4, r&, l$
    ELSE
      l$ = STRING$(LEN(l$), 0)
    END IF
    map$(i%) = l$
  NEXT
  data$ = LEFT$("total:" + STR$(record&) + " current:" + STR$(r&) + SPACE$(79), 79)
  LOCATE 1, 1: PRINT data$
  FOR byte% = 1 TO LEN(l$)
    FOR i% = 0 TO 7
      b%(i%) = ASC(MID$(map$(i%), byte%, 1))
    NEXT
    FOR graphic% = 0 TO 7 ' for 7 graphic characters
      gbyte%(graphic%) = 0
      FOR i% = 0 TO 7
        a% = b%(i%) - 2 * FIX(b%(i%) / 2!)
        b%(i%) = FIX(b%(i%) / 2!)
        gbyte%(graphic%) = gbyte%(graphic%) + a% * (2 ^ i%)
      NEXT
    NEXT
    FOR graphic% = 7 TO 0 STEP -1
      PRINT #5, CHR$(gbyte%(graphic%));
    NEXT
  NEXT
  PRINT #5,
WEND
PRINT #5, CHR$(12);
END SUB

```

```
FUNCTION error$ (ierr%) STATIC
STATIC l$
CALL findfile("sd.err", irtn%, dir$)
IF irtn% = 0 THEN
    CLOSE 5
    OPEN dir$ + "sd.err" FOR INPUT AS 5
    l$ = ""
    WHILE ierr% <> VAL(l$) AND (EOF(5) <> -1)
        LINE INPUT #5, l$
    WEND
    CLOSE 5
ELSE
    l$ = STR$(ierr%) + " Error Codes Missing: SD.ERR"
END IF
error$ = l$
END FUNCTION
```

```
!-----  
SUB errors (a%) STATIC  
!-----  
ierr% = ERR  
l$ = error$(ierr%)  
LOCATE 1, 1, 0: PRINT " "; MID$(l$, 3); " (A)bort (R)estart (I)gnore (H)elp"; CHR$(7);  
a$ = " "  
WHILE INSTR(1, "airh", a$) = 0  
    a$ = INPUT$(1)  
    a$ = LCASE$(a$)  
WEND  
a% = ASC(a$)  
LOCATE 1, 1: PRINT blank$;  
END SUB
```

```

'-----
SUB filesearch (filter$, cellname$, ext$, c%) STATIC
'-----
DIM inreg AS regtype, outreg AS regtype, dta%(64), f%(32), dir$(80), ext$(80)

filter$ = RTRIM$(filter$) + "."
filter$ = MID$(filter$, 1, INSTR(1, filter$, ".")) + ext$

FOR i% = 0 TO FIX((LEN(filter$) + 2) / 2) - 1
    f%(i%) = CVI(MID$(filter$ + CHR$(0) + CHR$(0), i% * 2 + 1, 2))
NEXT

' set DTA location

inreg.ax = CVI(CHR$(0) + CHR$(&H1A))
inreg.bx = 0
inreg.ds = VARSEG(dta%(0))
inreg.dx = VARPTR(dta%(0))
CALL interruptx(&H21, inreg, outreg)

' find first matching file

inreg.ax = CVI(CHR$(0) + CHR$(&H4E))
inreg.bx = 0
inreg.ds = VARSEG(f%(0))
inreg.dx = VARPTR(f%(0))
CALL interruptx(&H21, inreg, outreg)

total% = 0

WHILE outreg.ax = 0

    f$ = ""
    FOR i% = 15 TO 15 + 13
        f$ = f$ + MKI$(dta%(i%))
    NEXT

    f$ = LEFT$(MID$(f$, 1, INSTR(1, f$ + CHR$(0), CHR$(0)) - 1) + SPACE$(13), 13)

    SELECT CASE total%
    CASE IS < 80
        total% = total% + 1
        n$ = LEFT$(LEFT$(f$, INSTR(1, f$ + ".", ".") - 1) + SPACE$(8), 8)
        e$ = MID$(f$ + SPACE$(4), INSTR(1, f$ + ".", ".") + 1, 4)
        dir$(total%) = CVD(n$)
        ext$(total%) = CVL(e$)
    CASE ELSE
    END SELECT

    inreg.ax = CVI(CHR$(0) + CHR$(&H4F))
    inreg.bx = 0
    inreg.ds = VARSEG(f%(0))
    inreg.dx = VARPTR(f%(0))

    ' find next matching file
    CALL interruptx(&H21, inreg, outreg)

WEND

SELECT CASE total%
CASE 0: file$ = ""
CASE ELSE: CALL selectfile(dir#(), ext&(), total%, cellname$, c%)
END SELECT

END SUB

```

```

'-----
SUB filestatus (file$, ax%, dir$) STATIC
'-----
'
' s%=1 for file found
'
' ax% 2 - invalid file name (mode)
'      3 - invalid path name
'      18 - file not found
'
DIM reg AS regtype, outreg AS regtype, f%(32)
SELECT CASE LEN(dir$)
CASE 0: dir$ = packstr$(directory$)
CASE ELSE: dir$ = packstr$(dir$)
END SELECT

file$ = packstr$(file$)

FOR i% = 0 TO FIX((LEN(dir$ + file$) + 2) / 2) - 1
    f%(i%) = CVI(MID$(dir$ + file$ + CHR$(0) + CHR$(0), i% * 2 + 1, 2))
NEXT

reg.ax = mki%(&H4E, 0)
reg.bx = 0
reg.ds = VARSEG(f%(0))
reg.dx = VARPTR(f%(0))

CALL interruptx(&H21, reg, outreg)

SELECT CASE outreg.ax
CASE 18: ax% = 53      ' file not found
CASE 3: ax% = 76      ' bad path (file spec)
CASE 2: ax% = 64      ' bad file name (file spec)
CASE ELSE: ax% = 0
END SELECT

FOR i% = 0 TO 32
    path$ = MKI$(f%(i%))
NEXT

END SUB

```



```

SUB fillbox (l$, bit1&, bit2&) STATIC

byte1% = FIX(bit1& / 8!)
byte2% = FIX((bit2&) / 8!)
bita% = bit1& - (8 * byte1%)
WHILE bita% <= 7 AND (byte1% < byte2%)
    CALL setbit(byte1%, bita%, l$)
    bita% = bita% + 1
WEND
bitb% = bit2& - (8 * byte2%)
WHILE bitb% >= 0 AND (byte1% < byte2%)
    CALL setbit(byte2%, bitb%, l$)
    bitb% = bitb% - 1
WEND
IF byte1% = byte2% THEN
    FOR bit% = bita% TO bitb%
        CALL setbit(byte2%, bit%, l$)
    NEXT
ELSE
    bytes% = (byte2% - byte1%) - 1
    IF bytes% > 0 THEN
        LOCATE 1, 40: PRINT "bytes: "; bytes%; " in map    ";
        MID$(l$, byte1% + 2) = STRING$(bytes%, 255)
    END IF
END IF

END SUB

```

```

-----
SUB findfile (file$, ax%, d$) STATIC
-----
d$ = ""
CALL filestatus(file$, ax%, d$)
IF ax% <> 0 THEN
    n% = 1
    i% = 0
    WHILE ENVIRON$(n%) <> "" AND i% = 0
        IF LEFT$(UCASE$(ENVIRON$(n%)), 4) = "PATH" THEN i% = n%
        n% = n% + 1
    WEND
    i1% = 5
    path$ = UCASE$(ENVIRON$(i%))
    i2% = INSTR(i1% + 1, path$, ";")
    WHILE i2% > 0 AND (ax% <> 0) AND i% > 0
        d$ = RTRIM$(LTRIM$(MID$(path$, i1% + 1, i2% - i1% - 1)))
        IF RIGHT$(d$, 1) <> "\" THEN d$ = d$ + "\"
        CALL filestatus(file$, ax%, d$)
        i1% = i2%
        i2% = INSTR(i1% + 1, path$, ";")
    WEND
END IF
END SUB

```

```

-----
SUB getcell (rec#(), sel#()) STATIC
-----
'SHARED level%, child#(), parent#()
STATIC x1#, y1#, x$, y$, file$, dir$, irtn%, p$, c%
x$ = STR$(sel#(3))
y$ = STR$(sel#(4))
t$ = "0"
irtn% = 1
c% = 0
WHILE irtn% AND c% <> 27
    p$ = "Saw Draw Cell Name? "
    CALL prompt(p$, file$, 8, c%)
    SELECT CASE c%
    CASE 27
    CASE ELSE
        IF short$(file$) = "" THEN file$ = "*"
        IF INSTR(1, file$, "***") > 0 OR INSTR(1, file$, "?") > 0 THEN
            CALL filesearch(short$(file$), file$, "CEL", c%)
            file$ = LEFT$(file$, INSTR(1, file$ + ".", ".") - 1)
        END IF
        CALL filestatus(packstr$(file$ + ".cel"), irtn%, dir$)
        a$ = "Y"
    END SELECT
    WHILE irtn% AND a$ <> "y" AND a$ <> "n" AND c% <> 27
        p$ = "Cell Not Found: " + short$(file$) + ", Create it? "
        CALL prompt(p$, a$, 1, c%)
        a$ = LCASE$(a$)
        SELECT CASE a$
        CASE "y": irtn% = 0
        CASE ELSE
        END SELECT
    WEND
WEND
IF c% <> 27 THEN
    CALL recquary("Set Cell Position", sel#(), c%, 0, 1!, 1!)
    IF c% <> 27 THEN
        rec#(1) = 7
        rec#(2) = 0
        rec#(3) = sel#(3)
        rec#(4) = sel#(4)
        rec#(5) = sel#(5)
        rec#(6) = sel#(6)
        rec#(7) = sel#(7)
        rec#(8) = CVD(LEFT$(UCASE$(file$) + SPACES$(8), 8))
        CALL editset(10, irec#, sel#(), rec#(), 1!, 1!)
        CALL header
        CALL opencell(irec#, rec#())
        parent#(level%, 5) = child#(level%, 5)
        parent#(level%, 6) = child#(level%, 6)
        parent#(level%, 9) = sel#(3) - child#(level%, 3)
        parent#(level%, 10) = sel#(4) - child#(level%, 4)
        CALL closecell(irec#, rec#())
        CALL readrec(irec#, rec#())
        CALL checkheader(irec#, rec#())
        CALL winon
    END IF
END IF
END SUB

```

```

-----
SUB getline (v$, lv%, c%) STATIC
-----
'SHARED ifor2%, xpixel%, ypixel%
CALL winoff
COLOR ifor2% + 8
i% = 1
row% = CSRLIN
col% = POS(1)
notdone% = 1
v$ = LEFT$(LTRIM$(v$), lv%)
WHILE notdone%
  c% = 0
  j% = 0
  LOCATE row%, col%, 0
  PRINT v$;
  LOCATE row%, col% + i% - 1, 1
  WHILE c% = 0
    c$ = INKEY$
    CALL cursor(c$, insert%)
    c% = ASC(RIGHT$(CHR$(0) + c$, 1))
  WEND
  l% = LEN(c$)
  SELECT CASE l%
    CASE 1: SELECT CASE c%
      CASE 8: del% = 1: i% = i% - 1
      CASE 9, 11, 13, 27: notdone% = 0
      CASE ELSE: j% = c%: move% = 1
    END SELECT
    CASE 2: SELECT CASE c%
      CASE 71: i% = 1
      CASE 72, 80, 15: notdone% = 0
      CASE 75: i% = i% - 1
      CASE 77: i% = i% + 1
      CASE 79: i% = LEN(v$)
        WHILE MID$(v$, i%, 1) = " "
          i% = i% - 1
        WEND
        i% = i% + 1
      CASE 82: insert% = 1 - insert%
      CASE 83: del% = 1
      CASE 117: del% = LEN(v$) - i% + 1
      CASE ELSE
    END SELECT
  END SELECT
  IF i% < 1 THEN i% = 1: del% = 0
  IF i% > lv% THEN i% = lv%
  SELECT CASE j%
    CASE 0
      v$ = LEFT$(v$, i% - 1) + MID$(v$ + SPACE$(lv%), i% + del%, lv% - i% + 1)
    CASE ELSE
      v$ = LEFT$(v$, i% - 1) + CHR$(j%) + MID$(v$ + SPACE$(lv%), i% + 1 - insert%, lv% -
    END SELECT
    i% = i% + move%
    move% = 0
    del% = 0
  WEND
END SUB

```

```

'-----
SUB help (i%) STATIC
'-----
COLOR 2
CALL winoff
CLOSE 4
file$ = "sd" + CHR$(48 + i%) + ".hlp"
CALL findfile(file$, irtn%, dir$)
c% = 0
SELECT CASE irtn%
CASE 0
    OPEN dir$ + file$ FOR INPUT AS 4
    WHILE (c% <> 27) AND (EOF(4) <> -1)
        i% = 3
        CALL winon
        CLS
        WHILE (i% < 25) AND (EOF(4) <> -1) AND (c% <> 27)
            c% = ASC(INKEY$ + " ")
            LINE INPUT #4, l$
            LOCATE i%, 2, 0
            PRINT LEFT$(l$ + blank$, 77);
            i% = i% + 1
        WEND
        c$ = INPUT$(1): c% = ASC(c$)
    WEND
    CLOSE 4
CASE ELSE
    LOCATE 1, 1
    PRINT error$(irtn%); " - "; file$; CHR$(6);
    a$ = INPUT$(1)
END SELECT
PCOPY 1, 0
END SUB

```

```
!-----  
FUNCTION hbyte% (i%) STATIC  
!-----  
hbyte% = CVI(MID$(MKIS$(i%), 2, 1) + CHR$(0))  
END FUNCTION
```

```
-----  
FUNCTION insidebox% (x1#, x2#, y1#, y2#, x&, y&) STATIC  
-----  
test% = 0  
IF ((x1# < x&) AND (x& < x2#)) OR ((x1# > x&) AND (x2# < x&)) THEN  
    IF ((y1# < y&) AND (y& < y2#)) OR ((y1# > y&) AND (y2# < y&)) THEN  
        test% = 1  
    END IF  
END IF  
insidebox% = test%  
END FUNCTION
```

```

!-----
SUB intersection (x1#, x2#, y1#, y2#, xy#, segment%, passed%) STATIC
!-----
m! = (y2# - y1#) / (x2# - x1#)
b# = FIX(((y1# + y2#) - m! * (x2# + x1#)) / 2)
passed% = 0
SELECT CASE segment%
CASE 1, 4 ' x is constant so find y
    xy# = m! * xy# + b#
    IF ((xy# > y1#) AND (xy# < y2#)) OR ((xy# < y1#) AND (xy# < y2#)) THEN passed% = 1
CASE 2, 3 ' y is constant so find x
    xy# = (xy# - b#) / m!
    IF ((xy# > x1#) AND (xy# < x2#)) OR ((xy# < x1#) AND (xy# < x2#)) THEN passed% = 1
END SELECT
END SUB

```



```
|-----  
SUB inverse (row%, col%, length%) STATIC  
|-----  
STATIC x1%, y1%, x2%, y2%  
DIM crt%(1000)  
CALL winoff  
x1% = (col% - 1) * xpixel%  
x2% = (col% + length%) * xpixel% - 1  
y1% = (row% - 1) * ypixel%  
y2% = row% * ypixel% - 1  
GET (x1%, y1%)-(x2%, y2%), crt%  
PUT (x1%, y1%), crt%, PRESET  
END SUB
```

```

SUB laser (sel#(), dy&, l$, dpix%, col%) STATIC
r& = 1
record& = INT(sel#(6) / dy&) + 1
esc$ = CHR$(27)
l% = LEN(l$)
PRINT #5, SPACE$(col%); esc$; "*" + LTRIM$(STR$(dpix%)) + "R"; esc$; "**r1A";
WHILE (r& < record&)
  r& = r& + 1
  GET #4, r&, l$
  PRINT #5, esc$; "**b"; LTRIM$(STR$(l%)); "W"; l$;
WEND
PRINT #5, esc$; "**rB"; CHR$(12);
END SUB

```

```
!-----  
FUNCTION lobyte% (i%) STATIC  
!-----  
lobyte% = CVI(MID$(MKI$(i%), 1, 1) + CHR$(0))  
END FUNCTION
```

```

-----
SUB main (mainmenu$, mnu$( ), mnu$( ), imnu%, ians%) STATIC
-----
DIM mpos%(25)
COLOR ifor2%, ibak%

optns% = 0
last% = 1

WHILE INSTR(last%, mainmenu$, " ")
    mpos%(optns%) = INSTR(last%, mainmenu$, " ") + 1
    last% = mpos%(optns%)
    optns% = optns% + 1
WEND

LOCATE 1, 1: PRINT LEFT$(mainmenu$ + blank$, 80);
row% = 3
ians% = 0
c% = 0
optns% = optns% - 2

WHILE (c% <> 13 AND c% <> 27)
    CALL inverse(1, mpos%(imnu%), mpos%(imnu% + 1) - mpos%(imnu%) - 1)

    m$ = MID$(mnu$(imnu%), 4)
    col% = mpos%(imnu%)
    length% = VAL(MID$(mnu$(imnu%), 1, 3))
    num% = LEN(m$) / length%
    CALL menu(m$, length%, num%, row%, col%, ians%, c%)

    move% = 0

    WHILE c% = 0
        c$ = ""
        WHILE c$ = "": c$ = INKEY$: WEND
        c% = ASC(LCASE$(RIGHT$(c$, 1)))

        SELECT CASE c%
            CASE 13: c% = 80
            CASE 72, 80, 77, 75, 27, 82
            CASE ELSE
                c% = 0
                FOR i% = 0 TO optns%
                    a% = ASC(LTRIM$(LCASE$(MID$(mainmenu$, mpos%(imnu% + 2), 1))))
                    IF c% = a% THEN move% = i% - imnu%: c% = 13
                NEXT
            END SELECT
        END SELECT

        SELECT CASE c%
            CASE 72, 80, 13
                menuon% = 1
            CASE 77: move% = 1
            CASE 75: move% = -1
            CASE 27, 82: ians% = -1
            CASE ELSE: c% = 0
        END SELECT

        CALL inverse(1, mpos%(imnu%), mpos%(imnu% + 1) - mpos%(imnu%) - 1)
        imnu% = imnu% + move%
        IF (imnu% < 0) THEN imnu% = optns%
        IF (imnu% > optns%) THEN imnu% = 0
    WEND
WEND SUB

```

```
!-----  
SUB maxof (v&, v1&, v2&) STATIC  
!-----  
IF ABS(v1&) > ABS(v2&) THEN  
    v& = ABS(v1&)  
ELSE  
    v& = ABS(v2&)  
END IF  
END SUB
```

```

'-----
SUB mc (sel#()) STATIC
'-----
DIM rec#(10)
STATIC c%, button%, i%, c$, v$, xpos$, ypos$, x#, y#
c% = 0
button% = 0
sel#(1) = 15

IF plotted% = 1 THEN
    FOR i% = 3 TO 6
        sel#(i%) = child#(0, i%)
    NEXT
    plotted% = 2
END IF

WHILE c% <> 27

    strspace& = FRE("")

    CALL winon
    PCOPY 1, 0
    COLOR ifor2% + 8
    LOCATE 1, 1
    CALL setbox(button%, sel#(), rec#(), cursorx%, cursory%, c%)
    CALL evaluate(c%, sel#(), rec#(), irec#, cursorx%, cursory%)

    x& = FIX((sx1# + sx2#) / 2!)
    y& = FIX((sy1# + sy2#) / 2!)

    w1& = FIX(x& - child#(0, 5))
    w2& = FIX(child#(0, 5) - w1&)
    w3& = FIX(x& - sel#(5))
    w4& = FIX(sel#(5) - w3&)

    h1& = FIX(y& - child#(0, 6))
    h2& = FIX(child#(0, 6) - h1&)
    h3& = FIX(y& - sel#(6))
    h4& = FIX(sel#(6) - h3&)

    CALL maxof(x1&, w1&, w2&) ' find max of w
    CALL maxof(x2&, w3&, w4&) ' find max of w
    CALL maxof(y1&, h1&, h2&) ' find max of h
    CALL maxof(y2&, h3&, h4&) ' find max of h
    CALL maxof(x&, x1&, x2&) ' find max of w
    CALL maxof(y&, y1&, y2&) ' find max of h

    zfx& = 2 * INT(x& / (sx2# - sx1#))
    zfy& = 2 * INT(y& / (sy2# - sy1#))

    CALL maxof(zfactor&, zfx&, zfy&)

    CALL winon
    v$ = LEFT$("Width:" + value$(sx2# - sx1#) + ", Height:" + value$(sy2# - sy1#) + blank$, 79)
    LOCATE 1, 1: PRINT v$;
    PCOPY 0, 1
    sel#(1) = 15
    CALL plotrec(irec#, sel#(), 1)
    CALL winoff
    CALL mouse(1, 0, 0, 0)
    c% = 0
    button% = 0
    WHILE c% = 0 AND button% = 0
        CALL mouse(3, button%, cursorx%, cursory%)
        CALL winon
        x# = FIX(PMAP(cursorx% - xpixel%, 2))
        y# = FIX(PMAP(cursory% - 2 * ypixel%, 3))
    
```

```
CALL winoff
xpos$ = LEFT$("X:" + value$(x#) + SPACE$(15), 15)
ypos$ = LEFT$("Y:" + value$(y#) + SPACE$(15), 15)
LOCATE 1, 50: PRINT xpos$; ypos$;
c$ = INKEY$: c% = ASC(RIGHT$(CHR$(0) + c$, 1))
WEND
CALL mouse(2, 0, 0, 0)
WEND
PCOPY 1, 0
END SUB
```

```

'-----
SUB menu (mnu$, length%, num%, row%, col%, ians%, c%) STATIC
'-----
PCOPY 0, 1

c% = 0
ians% = 0
COLOR ifor3%
CALL box(row%, col%, num%, length%)
COLOR ifor2% + 8

FOR i% = 0 TO num% - 1
    LOCATE row% + i%, col%
    PRINT MID$(mnu$, i% * length% + 1, length%)
NEXT

WHILE (c% <> 27) AND (c% <> 13) AND (c% <> 75) AND (c% <> 77)
    CALL inverse(row% + ians%, col%, length% - 1)
    WHILE c% = 0
        c$ = "": WHILE c$ = "": c$ = INKEY$: WEND
        c% = ASC(RIGHT$(c$, 1))
        move% = 0
        SELECT CASE c%
            CASE 80: move% = 1
            CASE 72: move% = -1
            CASE 13:
            CASE 27, 77, 75: move% = -ians%
            CASE ELSE:
                LOCATE 1, 40
                FOR i% = 0 TO num% - 1
                    a$ = LCASE$(MID$(mnu$, i% * length% + 2, 1))
                    IF CHR$(c%) = a$ THEN move% = i% - ians%
                NEXT
                c% = move%
            END SELECT
        END SELECT
    WEND

    CALL inverse(row% + ians%, col%, length% - 1)
    ians% = ians% + move%
    IF ians% < 0 THEN ians% = num% - 1
    IF ians% > num% - 1 THEN ians% = 0

    SELECT CASE c%
        CASE 13, 27, 77, 75
        CASE ELSE: c% = 0
    END SELECT
WEND
PCOPY 1, 0
END SUB

```



```
'-----  
FUNCTION mki% (h%, l%) STATIC  
'-----  
mki% = CVI(CHR$(l%) + CHR$(h%))  
END FUNCTION
```

```
!-----  
SUB mouse (ax%, bx%, cx%, dx%) STATIC  
!-----  
IF mouseon% THEN  
    DEF SEG = 0  
    mseg% = INT(256 * PEEK(51 * 4 + 3) + PEEK(51 * 4 + 2))  
    mouseon% = INT(256 * PEEK(51 * 4 + 1) + PEEK(51 * 4) + 2)  
    DEF SEG = mseg%  
    CALL ABSOLUTE(ax%, bx%, cx%, dx%, mouseon%)  
END IF  
END SUB
```

```
SUB outlinebox (l$, bit1&, bit2&) STATIC
byte1% = FIX(bit1& / 8!)
byte2% = FIX((bit2&) / 8!)
bita% = bit1& - (8 * byte1%)
CALL setbit(byte1%, bita%, l$)
bitb% = bit2& - (8 * byte2%)
CALL setbit(byte2%, bitb%, l$)
END SUB
```

```
'-----  
FUNCTION packstr$ (s$) STATIC  
'-----  
STATIC i%, c%, j%  
i% = 0  
j% = 0  
WHILE i% < LEN(s$)  
    i% = i% + 1  
    c% = ASC(MID$(s$, i%, 1))  
    IF (c% > 32) THEN  
        j% = j% + 1  
        MID$(s$, j%, 1) = CHR$(c%)  
    END IF  
WEND  
packstr$ = LEFT$(s$, j%)  
END FUNCTION
```

```

!-----
SUB printcell (sel#()) STATIC
!-----
STATIC dx1&, dy1&, dx2&, dy2&
DIM lineseg&(4, 4), m!(4), b&(4), rec#(10), b%(7)
editing% = 0
esc$ = CHR$(27)
CALL restart(irec#, rec#())
dy& = 1000
dx& = 1000
irec# = 1
notdone% = 1
xorigin# = sel#(3)
yorigin# = sel#(4)
torigin# = -(pi# / 2!)
c% = 0

CALL printers(printer%, dpix%, dpiy%, c%)
CALL printsetup(sel#(), r%, printer%, dpix%, dpiy%, dx&, dy&, l$, c%, row%, col%)

prt$ = RTRIM$(MKD$(child#(level%, 8))) + ".PRT"
irtn% = 64
WHILE (irtn% = 76 OR irtn% = 64) AND c% <> 27
    CALL prompt("Write Printer Commands to? ", prt$, 20, c%)
    CALL filestatus(prt$, irtn%, d$)
WEND

LOCATE 1, 1
PRINT LEFT$(blank$, 79);

WHILE (irec# <= child#(level%, 2) OR level%) AND child#(level%, 2) > 0 AND c% <> 27
    CALL readrec(irec#, rec#())
    CALL addoff(rec#())
    IF r% THEN CALL rotate(rec#(), xorigin#, yorigin#, torigin#)
    SELECT CASE rec#(1)
    CASE 7
        IF level% < explevel% OR rec#(2) THEN
            IF r% THEN CALL rotate(rec#(), xorigin#, yorigin#, -torigin#)
            CALL opencell(irec#, rec#())
            IF r% THEN CALL rotate(rec#(), xorigin#, yorigin#, torigin#)
            irec# = 0
        ELSE
            fill% = 0
        END IF
    CASE 1, 2, 3, 4, 5, 6
        fill% = 1
    CASE ELSE
        fill% = -1
    END SELECT
    IF irec# <> 0 AND rec#(1) <> 0 THEN ' evaluate record
        CALL within(rec#(), sel#(), inwindow%)
        data$ = "cell rec:" + STR$(irec#) + "      "
        LOCATE 1, 1: PRINT data$;
        IF inwindow% THEN ' print record
            CALL setlines(rec#(), lineseg&(), m!(), b&())

            topy% = 4
            topseg% = 2
            endy% = 4
            botseg% = 4

            IF lineseg&(topseg%, 3) > lineseg&(topseg%, 4) THEN
                topy% = 3
            END IF
            IF lineseg&(botseg%, 3) < lineseg&(botseg%, 4) THEN
                endy% = 3
            END IF

```

```

endy& = lineseg&(topseg%, topy%)
starty& = lineseg&(botseg%, endy%)
IF lineseg&(topseg%, topy%) > sel#(4) + sel#(6) THEN
    endy& = (sel#(4) + sel#(6))
END IF
IF lineseg&(botseg%, endy%) < sel#(4) THEN
    starty& = sel#(4)
END IF
y& = starty& + 1
WHILE (y& >= starty& + 1) AND (y& <= endy&)
    seg2% = 3
    seg1% = 1
    FOR i% = 2 TO 4 STEP 2 ' find segs for y
        y# = y&
        testy# = (lineseg&(i%, 3) - y#) * (lineseg&(i%, 4) - y#)
        IF testy# < 0 THEN
            SELECT CASE rec#(7)
                CASE IS > 0
                    SELECT CASE i%
                        CASE 2: seg1% = i%
                        CASE 4: seg2% = i%
                    END SELECT
                CASE IS < 0
                    SELECT CASE i%
                        CASE 2: seg2% = i%
                        CASE 4: seg1% = i%
                    END SELECT
                CASE ELSE
                    END SELECT
            END IF
        NEXT
        SELECT CASE m!(seg1%)
            CASE 0
                startx& = INT(rec#(3))
                endx& = INT(rec#(3) + rec#(5))
            CASE ELSE
                startx& = (y& - b&(seg1%)) / m!(seg1%)
                endx& = (y& - b&(seg2%)) / m!(seg2%)
            END SELECT
            IF startx& < sel#(3) THEN startx& = sel#(3)
            IF endx& > sel#(3) + sel#(5) THEN endx& = sel#(3) + sel#(5)
            record& = FIX((sel#(4) + sel#(6) - y&) / (dy& * 1!)) + 2

            GET #4, record&, l$

            bit1& = FIX((startx& - sel#(3)) / dx&)
            bit2& = FIX((endx& - sel#(3)) / dx&)

            SELECT CASE fill%
                CASE 1: CALL fillbox(l$, bit1&, bit2&)
                CASE 0
                    IF (y& <= (starty& + dy&)) OR (y& >= endy& - dy&) THEN
                        CALL fillbox(l$, bit1&, bit2&)
                    ELSE
                        CALL outlinebox(l$, bit1&, bit2&)
                    END IF
                CASE ELSE
                    END SELECT

            PUT #4, record&, l$

            y& = y& + dy&
        WEND
    END IF
END IF
IF irec# >= child#(level%, 2) AND level% THEN
    IF r% THEN CALL rotate(rec#(), xorigin#, yorigin#, -torigin#)
    CALL closecell(irec#, rec#())

```

```
END IF
irec# = irec# + 1
c% = ASC(INKEY$ + CHR$(0))
WEND
IF c% <> 27 THEN
    WIDTH "lpt1:", 255
    WIDTH "lpt2:", 255
    WIDTH "lpt3:", 255
    OPEN prt$ FOR OUTPUT AS 5
    FOR i% = 1 TO row%
        PRINT #5,
    NEXT
    SELECT CASE printer%
    CASE 4, 5, 6: CALL epson(sel#(), dy&, l$, dpix%, col%)
    CASE 1, 2, 3: CALL laser(sel#(), dy&, l$, dpix%, col%)
    END SELECT
    CLOSE 4, 5
END IF
IF r% THEN CALL rotate(sel#(), xorigin#, yorigin#, -torigin#)
editing% = 1
END SUB
```

```

-----
SUB printers (printer%, dpix%, dpiy%, c%) STATIC
-----

```

```

t%(1) = 1: l%(1) = 1: v#(1) = 0: p$(1) = "HP Laser Jet at 75 dpi "
t%(2) = 1: l%(2) = 1: v#(2) = 0: p$(2) = "                  150 dpi "
t%(3) = 1: l%(3) = 1: v#(3) = 0: p$(3) = "                  300 dpi "
t%(4) = 1: l%(4) = 1: v#(4) = 0: p$(4) = "Epson at 60 dpi"
t%(5) = 1: l%(5) = 1: v#(5) = 0: p$(5) = "                  120 dpi"
t%(6) = 1: l%(6) = 1: v#(6) = 0: p$(6) = "                  240 dpi"

```

```

v#(printer%) = 1

```

```

title$ = "Select a Single Printer From the List Below"

```

```

c% = 0

```

```

printer% = 0

```

```

WHILE printer% = 0 AND c% <> 27

```

```

    CALL query(title$, 6, c%)

```

```

    printer% = 0

```

```

    i% = 0

```

```

    WHILE (printer% <= 0 OR printer% > 6) AND c% <> 27 AND i% < 6

```

```

        i% = i% + 1

```

```

        printer% = i% * v#(i%)

```

```

    WEND

```

```

    SELECT CASE printer%

```

```

        CASE 1: dpix% = 75

```

```

            dpiy% = 75

```

```

        CASE 2: dpix% = 150

```

```

            dpiy% = 150

```

```

        CASE 3: dpix% = 300

```

```

            dpiy% = 300

```

```

        CASE 4: dpix% = 60

```

```

            dpiy% = 72

```

```

        CASE 5: dpix% = 120

```

```

            dpiy% = 72

```

```

        CASE 6: dpix% = 240

```

```

            dpiy% = 72

```

```

        CASE ELSE

```

```

    END SELECT

```

```

WEND

```

```

END SUB

```



```

!-----
SUB printsetup (sel#(), r%, printer%, dpix%, dpiy%, dx&, dy&, l$, c%, row%, col%) STATIC
!-----
STATIC i%

IF pw! <= 0 THEN pw! = 8
IF ph! <= 0 THEN ph! = 10

t%(1) = 0: l%(1) = 20: v#(1) = FIX(sel#(3)): p$(1) = "Window X Coordinate (nm)? "
t%(2) = 0: l%(2) = 20: v#(2) = FIX(sel#(4)): p$(2) = "Window Y Coordinate (nm)? "
t%(3) = 0: l%(3) = 20: v#(3) = FIX(sel#(5)): p$(3) = "Window Width (nm)? "
t%(4) = 0: l%(4) = 20: v#(4) = FIX(sel#(6)): p$(4) = "Window Height (nm)? "
t%(5) = 2: l%(5) = 4: v#(5) = pw!: p$(5) = "Paper Width (inches)? "
t%(6) = 2: l%(6) = 4: v#(6) = ph!: p$(6) = "Paper Length/height (inches)? "
t%(7) = 2: l%(7) = 3: v#(7) = row%: p$(7) = "Begin Printing in row? "
t%(8) = 2: l%(8) = 3: v#(8) = col%: p$(8) = "Begin Printing in column? "
t%(9) = 1: l%(9) = 20: v#(9) = r%: p$(9) = "Rotate Drawing 90 degrees"

title$ = "Define Print Window, Print Area & Orientation"

DO
    CALL query(title$, 9, c%)
LOOP UNTIL c% = 27 OR (INT(v#(5)) > 0 AND INT(v#(6)) > 0 AND INT(v#(3)) > 0 AND INT(v#(4)) > 0)

IF c% <> 27 THEN
    row% = v#(7)
    col% = v#(8)
    xorigin# = sel#(3)
    yorigin# = sel#(4)
    torigin# = -(pi# / 2!)
    sel#(3) = v#(1)
    sel#(4) = v#(2)
    sel#(5) = v#(3)
    sel#(6) = v#(4)
    r% = v#(9)
    IF r% THEN
        CALL rotate(sel#(), xorigin#, yorigin#, torigin#)
    END IF
    pw! = v#(5)
    ph! = v#(6)
    dx& = INT(sel#(5) / (dpix% * pw!))
    dy& = INT(sel#(6) / (dpiy% * ph!))
    recs% = dpiy% * ph! + 1
    bit% = FIX(sel#(5) / dx&)
    lrecl% = FIX((bit% + 8!) / 8!)
    l$ = CHR$(128) + STRING$(lrecl% - 2, 0) + CHR$(1)
    CLOSE 4
    OPEN "sd.map" FOR RANDOM ACCESS READ WRITE AS 4 LEN = lrecl%
    LOCATE 1, 1
    PRINT LEFT$(" Initializing the Bit Map" + blank$, 79);
    FOR i% = 1 TO recs%
        PUT #4, i%, l$
    NEXT
    l$ = STRING$(lrecl%, 255)
    PUT #4, 2, l$
    PUT #4, recs%, l$
    a$ = MKI$(recs%) + MKI$(lrecl%)
    PUT #4, 1, a$
END IF

END SUB

```

```
-----  
SUB prompt (p$, v$, i%, c%) STATIC  
-----  
COLOR ifor3%  
p$ = LTRIM$(p$)  
num% = 3: length% = i% + LEN(p$) + 2  
row% = 10  
col% = 40 - INT(length% / 2)  
CALL box(row% - 1, col%, num%, length%)  
LOCATE row%, col% + 1, 0  
COLOR ifor2%  
PRINT p$;  
c = 0  
CALL getline(v$, i%, c%)  
PCOPY 1, 0  
END SUB
```

```

-----
SUB putlabel (irec#, sel#(), rec#(), c%) STATIC
-----
c% = 0
COLOR ifor2%
WHILE c% <> 13 AND c% <> 27
    CALL prompt("Text? ", l$, 40, c%)
WEND
IF c% <> 27 THEN
    jrec# = child#(level%, 2) + 1
    CALL reclabel("Set Label Position", sel#(), scle&, c%)
    CALL label(sel#(), scle&, l$)
    CALL selrecs(jrec#, rec#(), sel#())
END IF
END SUB

```

```

-----
SUB qdump (title$, dx#, dy#, sx, st, c%) STATIC
-----

t%(1) = 0: l%(1) = 20: v#(1) = -FIX(child#(0, 3)): p$(1) = "X Offset? "
t%(2) = 0: l%(2) = 20: v#(2) = -FIX(child#(0, 4)): p$(2) = "Y Offset? "
t%(3) = 2: l%(3) = 20: v#(3) = 1: p$(3) = "Scale data by? "
t%(4) = 2: l%(4) = 20: v#(4) = 1: p$(4) = "Scale degrees by? "
t%(5) = 2: l%(5) = 2: v#(5) = layer%: p$(5) = "Dump Layer Number (Saw Cad only)? "
t%(6) = 2: l%(6) = 2: v#(6) = explevel%: p$(6) = "Expand How Many Levels? "

CALL query(title$, 6, c%)

IF c% <> 27 THEN
    sx = v#(3)
    st = v#(4)
    dx# = FIX(v#(1)) * sx
    dy# = FIX(v#(2)) * sx
    layer% = INT(v#(5))
    explevel% = INT(v#(6))
END IF

END SUB

```

```

!-----
SUB query (title$, opts%, c%) STATIC
!-----
length% = 60
num% = opts% + 3
col% = 11
row% = 12 - INT(num% / 2)
CALL winoff
CALL box(row% - 1, col%, num%, length%)
COLOR ifor1%
LOCATE row% - 1, col% + 2: PRINT title$;
COLOR ifor2% + 8
LOCATE row% - 0, col% + 2: PRINT "Tab-Moves; Esc-Abort; Enter-Done"
COLOR ifor2%
FOR i% = 1 TO opts%
    SELECT CASE t%(i%)
    CASE 0: v$ = value$(v#(i%))
        LOCATE row% + 1 + i%, col% + 2: PRINT p$(i%); LEFT$(v$, l%(i%));
    CASE 2: v$ = LTRIM$(STR$(v#(i%)))
        LOCATE row% + 1 + i%, col% + 2: PRINT p$(i%); LEFT$(v$, l%(i%));
    CASE 1: LOCATE row% + 1 + i%, col% + 2: COLOR ifor1%
        PRINT CHR$(v#(i%) * 88 + (1 - v#(i%)) * 249); " ";
        COLOR ifor2%
        PRINT p$(i%);
    CASE ELSE
    END SELECT
NEXT

p% = 1
c% = 0

WHILE c% <> 27 AND (c% <> 13 OR p% <> 1)

    LOCATE row% + 1 + p%, col% + 2
    COLOR ifor2%
    SELECT CASE t%(p%)
    CASE 0, 2: PRINT p%(p%);
        SELECT CASE t%(p%)
        CASE 0: d$ = value$(v#(p%))
        CASE 2: d$ = LTRIM$(STR$(v#(p%)))
        END SELECT
        CALL getline(d$, l%(p%), c%)
        SELECT CASE t%(p%)
        CASE 0: v#(p%) = VAL(d$) * units#(d$)
        CASE 2: v#(p%) = VAL(d$)
        END SELECT
    CASE 1: f% = FIX(v#(p%))
        CALL switch(f%, c%)
        v#(p%) = f%
    CASE ELSE
    END SELECT

    SELECT CASE c%
    CASE 15, 72: p% = p% - 1
    CASE 9, 80, 13: p% = p% + 1
    CASE 71: p% = 1
    CASE ELSE
    END SELECT

    IF p% > opts% THEN p% = 1
    IF p% < 1 THEN p% = opts%

WEND

PCOPY 1, 0
CALL winon

END SUB

```

```
!-----  
FUNCTION radians# (d#) STATIC  
!-----  
radians# = d# * pi# / 180#  
END FUNCTION
```

```

!-----
SUB recarray (sel#(), c%, x%, y%) STATIC
!-----
STATIC f%
IF f% = 0 THEN CALL addoff(sel#())
t%(1) = 0: l%(1) = 20: v%(1) = FIX(sel#(3) - sminx# * f%): p$(1) = "X Coordinate? "
t%(2) = 0: l%(2) = 20: v%(2) = FIX(sel#(4) - sminy# * f%): p$(2) = "Y Coordinate? "
t%(3) = 1: l%(3) = 0: v%(3) = f%: p$(3) = "Use Coordinates as Offsets "
t%(4) = 2: l%(4) = 4: v%(4) = 1: p$(4) = "Number of Arrays in X? "
t%(5) = 2: l%(5) = 4: v%(5) = 1: p$(5) = "Number of Arrays in Y? "

CALL query("Array Selections", 5, c%)

IF c% <> 27 THEN
    sel#(3) = FIX(v%(1) + sminx# * v%(3))
    sel#(4) = FIX(v%(2) + sminy# * v%(3))
    x% = INT(v%(4))
    y% = INT(v%(5))
END IF
f% = v%(3)
IF f% = 0 THEN CALL suboff(sel#())

END SUB

```

```

!-----
SUB reccopy (title$, sel#(), c%, test%, sx, st) STATIC
!-----
STATIC f%
IF f% = 0 THEN CALL addoff(sel#())

t%(1) = 0: l%(1) = 20: v%(1) = FIX(sel#(3) - sminx# * f%): p$(1) = "X Coordinate? "
t%(2) = 0: l%(2) = 20: v%(2) = FIX(sel#(4) - sminy# * f%): p$(2) = "Y Coordinate? "
t%(3) = 2: l%(3) = 20: v%(3) = degrees#(sel#(7)): p$(3) = "Rotate Through an Angle of? "
t%(4) = 1: l%(4) = 0: v%(4) = f%: p$(4) = "Use Coordinates as Offsets "

CALL query(title$, 4, c%)

IF c% <> 27 THEN
    sel#(3) = FIX(v%(1) + sminx# * v%(4))
    sel#(4) = FIX(v%(2) + sminy# * v%(4))
    sel#(7) = radians#(v%(3))
    sx = v%(5)
    st = v%(6)
END IF
f% = v%(4)

IF f% = 0 THEN CALL suboff(sel#())
END SUB

```



```

!-----
SUB recfile (title$, sel#(), c%, sx, st) STATIC
!-----
t%(1) = 2: l%(1) = 20: v#(1) = degrees#(sel#(7)): p$(1) = "Rotate Through an Angle of? "
t%(2) = 2: l%(2) = 15: v#(2) = 1: p$(2) = "Scale to nm? "
t%(3) = 2: l%(3) = 15: v#(3) = 1: p$(3) = "Scale to degrees? "

CALL query(title$, 3, c%)

IF c% <> 27 THEN
    sel#(7) = radians#(v#(1))
    sx = v#(2)
    st = v#(3)
END IF
END SUB

```

```

SUB reclabel (title$, sel#(), pixel&, c%) STATIC
!-----
CALL addoff(sel#())
t%(1) = 0: l%(1) = 15: v%(1) = sel#(3): p$(1) = "X Position of Label? "
t%(2) = 0: l%(2) = 15: v%(2) = sel#(4): p$(2) = "Y Position of Label? "
t%(3) = 2: l%(3) = 15: v%(3) = degrees#(sel#(7)): p$(3) = "Rotate Label by? "
t%(4) = 0: l%(4) = 15: v%(4) = 1000: p$(4) = "Pixel Size? "

CALL query(title$, 4, c%)

IF c% <> 27 THEN
    sel#(3) = v%(1)
    sel#(4) = v%(2)
    sel#(7) = radians#(v%(3))
    pixel& = FIX(v%(4))
END IF
CALL suboff(sel#())
END SUB

```

```

!-----
SUB recmove (sel#(), c%, sx, sy) STATIC
!-----
STATIC f%
IF f% = 0 THEN CALL addoff(sel#())
t%(1) = 0: l%(1) = 20: v%(1) = FIX(sel#(3) - sminx# * f%): p$(1) = "X Coordinate? "
t%(2) = 0: l%(2) = 20: v%(2) = FIX(sel#(4) - sminy# * f%): p$(2) = "Y Coordinate? "
t%(3) = 2: l%(3) = 20: v%(3) = degrees#(sel#(7)): p$(3) = "Rotate Through an Angle of? "
t%(4) = 1: l%(4) = 0: v%(4) = f%: p$(4) = "Use Coordinates as Offsets "
IF sw# < 1 OR sh# < 1 THEN
    sw# = sel#(5)
    sh# = sel#(6)
END IF
t%(5) = 2: l%(5) = 12: v%(5) = 1: p$(5) = "X Scale (" + LEFT$(LTRIM$(STR$(sel#(5) / sw#)), 8) + ")?
t%(6) = 2: l%(6) = 12: v%(6) = 1: p$(6) = "Y Scale (" + LEFT$(LTRIM$(STR$(sel#(6) / sh#)), 8) + ")?

CALL query("Move/Rotate/Scale Selections", 6, c%)

IF c% <> 27 THEN
    sel#(3) = FIX(v%(1) + sminx# * v%(4))
    sel#(4) = FIX(v%(2) + sminy# * v%(4))
    sel#(7) = radians#(v%(3))
    sx = v%(5)
    sy = v%(6)
END IF
f% = v%(4)

IF f% = 0 THEN CALL suboff(sel#())
END SUB

```

```

|-----
SUB recquery (title$, sel#(), c%, test%, sx, st) STATIC
|-----
STATIC f1%, f2%
IF f1% = 0 THEN CALL addoff(sel#())

t%(1) = 0: l%(1) = 20: v%(1) = FIX(sel#(3) * (1 - f1%)): p$(1) = "X Coordinate? "
t%(2) = 0: l%(2) = 20: v%(2) = FIX(sel#(4) * (1 - f1%)): p$(2) = "Y Coordinate? "
t%(3) = 2: l%(3) = 20: v%(3) = degrees#(sel#(7)): p$(3) = "Rotate Through an Angle of? "
t%(4) = 1: l%(4) = 0: v%(4) = f1%: p$(4) = "Use Coordinates as Offsets "
t%(5) = 1: l%(5) = 0: v%(5) = f2%: p$(5) = "Position Cell Relative to its Origin"

CALL query(title$, 4, c%)

IF c% <> 27 THEN
    sel#(3) = FIX(v%(1) + sel#(3) * v%(4))
    sel#(4) = FIX(v%(2) + sel#(4) * v%(4))
    sel#(7) = radians#(v%(3))
    sx = v%(5)
    st = v%(6)
END IF
f1% = v%(4)

IF f1% = 0 THEN CALL suboff(sel#())
END SUB

```

```

|-----
SUB recstep (dx#, dy#, dw#, dh#, dt#, repeats%, c%) STATIC
|-----
t%(1) = 0: l%(1) = 15: v#(1) = dx#: p$(1) = "X Step? "
t%(2) = 0: l%(2) = 15: v#(2) = dy#: p$(2) = "Y Step? "
t%(3) = 0: l%(3) = 15: v#(3) = dw#: p$(3) = "Delta Width? "
t%(4) = 0: l%(4) = 15: v#(4) = dh#: p$(4) = "Delta Height? "
t%(5) = 2: l%(5) = 8: v#(5) = degrees#(dt#): p$(5) = "Delta Theta (deg)? "
t%(6) = 2: l%(6) = 5: v#(6) = repeats%: p$(6) = "Repeats? "

title$ = "Step and Repeat Selected Rectangles"
CALL query(title$, 6, c%)

IF c% <> 27 THEN
    dx# = v#(1)
    dy# = v#(2)
    dw# = v#(3)
    dh# = v#(4)
    dt# = radians(v#(5))
    repeats% = INT(v#(6))
END IF
END SUB

```

```
!-----  
SUB rotatepoint (x0&, y0&, x&, y&, dt#) STATIC  
!-----  
dx# = x& - x0&  
dy# = y& - y0&  
r& = FIX(SQR(dx# ^ 2 + dy# ^ 2))  
t# = arctan#(dx#, dy#) + dt#  
x& = r& * COS(t#) + x0&  
y& = r& * SIN(t#) + y0&  
END SUB
```

```

!-----
SUB saveas (rec#()) STATIC
!-----
SELECT CASE fileopen%
CASE 1: SELECT CASE saved%
    CASE 0: CALL pack(rec#(), 1)
    CASE 1
    END SELECT

    CALL restart(irec#, rec#())
    file$ = MKD$(child#(0, 8))
    cellname1$ = short$(file$) + ".rnd"
    CALL prompt("Save Cell As? ", file$, 8, c%)

    SELECT CASE c%
    CASE 27
    CASE ELSE
        CLOSE
        child#(0, 8) = CVD(file$)
        cellname2$ = short$(file$) + ".rnd"
        CALL filestatus(cellname2$, irtn%, dir$)
        a$ = "y"
        SELECT CASE irtn%
        CASE 0: CALL prompt(" Cell Already Exists, Save Anyway? ", a$, 1, c%)
            a$ = LCASE$(a$)
        CASE IS <> 53: a$ = "n"
            BEEP
            LOCATE 1, 1: PRINT "Bad Cell Name, Not Saved: "; cellname2$
        CASE ELSE
        END SELECT
        SELECT CASE a$
        CASE "y": SHELL "rename " + cellname1$ + " " + cellname2$
            SHELL "copy *.rnd *.cel/b > nul "
            CALL openfile(cellname2$, rec#())
            CALL openwindow(3, 2, 22, 78)
            saved% = 1
        CASE ELSE
        END SELECT
    END SELECT
CASE ELSE
    LOCATE 1, 1
    PRINT LEFT$("No Cells Open" + blank$, 79);
    BEEP
END SELECT
END SUB

```

```

!-----
SUB scroll (test%) STATIC
!-----
STATIC u%, d%, r%, l%, pages

IF pages <= 0 THEN pages = 1

t%(1) = 2: l%(1) = 3: v%(1) = pages: p$(1) = "Scroll How Many Pages? "
t%(2) = 1: l%(2) = 0: v%(2) = sr%: p$(2) = "Scroll Right "
t%(3) = 1: l%(3) = 0: v%(3) = sl%: p$(3) = "Scroll Left "
t%(4) = 1: l%(4) = 0: v%(4) = su%: p$(4) = "Scroll Up "
t%(5) = 1: l%(5) = 0: v%(5) = sd%: p$(5) = "Scroll Down "

CALL query("Scroll Through the Cell", 5, c%)

pages = v%(1)
sr% = v%(2)
sl% = v%(3)
su% = v%(4)
sd% = v%(5)

IF c% <> 27 THEN
    test% = 1
    CALL winoff
    pagex# = (sx2# - sx1#) * pages
    pagey# = (sy2# - sy1#) * pages
    sx1# = sx1# + sr% * pagex# - sl% * pagex#
    sy1# = sy1# + su% * pagey# - sd% * pagey#
    sx2# = sx2# + sr% * pagex# - sl% * pagex#
    sy2# = sy2# + su% * pagey# - sd% * pagey#
    CALL winon
ELSE
    test% = 0
END IF
END SUB

```



```

!-----
SUB selectfile (dir#(), ext&(), total%, file$, c%) STATIC
!-----
PCOPY 0, 1

cols% = 5
rows% = INT(FIX((total% + cols% - 1!) / cols%))

CALL box(5, 3, 16, 75)
LOCATE 4, 29
PRINT "| Select Desired Cell |";

COLOR ifor2% + 8

FOR i% = 1 TO total%
    row% = INT(FIX((i% + cols% - 1!) / cols%))
    col% = i% - ((row% - 1) * cols%)
    LOCATE row% + 4, col% * 15 - 11, 0
    PRINT packstr$(MKD$(dir#(i%)) + "." + MKL$(ext&(i%)))
NEXT

i% = 1
c% = 0

WHILE (c% <> 13) AND (c% <> 27)
    row% = INT(FIX((i% + cols% - 1!) / cols%))
    col% = i% - ((row% - 1) * cols%)
    CALL inverse(row% + 4, col% * 15 - 11, 13)
    c% = 0
    lasti% = i%
    WHILE c% = 0
        c$ = ""
        WHILE LEN(c$) = 0: c$ = INKEY$: WEND
        c% = ASC(RIGHT$(c$, 1))
        SELECT CASE c%
            CASE 77: i% = i% + 1
            CASE 75: i% = i% - 1
            CASE 72: i% = i% - cols%
            CASE 80: i% = i% + cols%
            CASE 79: i% = 1
            CASE 27, 13: done
            CASE ELSE: c% = 0
        END SELECT
    WEND

    CALL inverse(row% + 4, col% * 15 - 11, 13)

    IF i% < 1 THEN i% = total%
    IF i% > total% THEN i% = 1

    row% = INT(FIX((i% + cols% - 1!) / cols%))
    col% = i% - ((row% - 1) * cols%)
WEND
file$ = packstr$(MKD$(dir#(i%)) + "." + MKL$(ext&(i%)))
PCOPY 1, 0
END SUB

```

```

!-----
SUB setbit (byte%, bit%, l$) STATIC
!-----
n% = 7 - bit%
i% = ASC(MID$(l$, byte% + 1, 1))
b% = FIX(i% / (2! ^ n%))
test% = b% - 2 * FIX(b% / 2!)
SELECT CASE test%
CASE 0: j% = (i% + (2 ^ n%))
        c$ = CHR$(j%)
        MID$(l$, byte% + 1, 1) = c$
CASE ELSE
END SELECT
END SUB

```

```

-----
SUB setbox (button%, sel#(), rec#(), cursorx%, cursory%, c%) STATIC
-----

CALL addoff(sel#())
IF cursorx% < xpixel% * 79 AND cursory% < ypixel% * 24 THEN
  x& = FIX(PMAP(cursorx% - xpixel%, 2))
  y& = FIX(PMAP(cursory% - 2 * ypixel%, 3))
  SELECT CASE gridon%
    CASE 1
      gx& = INT((x& - xorigin& + (xstep& / 2)) / xstep&) * xstep& + xorigin&
      gy& = INT((y& - yorigin& + (ystep& / 2)) / ystep&) * ystep& + yorigin&
    CASE 0
      gx& = x&
      gy& = y&
  END SELECT
  SELECT CASE button%
    CASE 1 ' set x,y
      sel#(2) = 0
      sel#(3) = gx&
      sel#(4) = gy&
    CASE 2 ' set w,h
      w# = gx& - sel#(3)
      h# = gy& - sel#(4)
      IF (w# < 0) THEN sel#(3) = sel#(3) + w#
      IF (h# < 0) THEN sel#(4) = sel#(4) + h#
      sel#(5) = ABS(w#)
      sel#(6) = ABS(h#)
      sel#(7) = 0
    CASE 4 ' set angle
      x# = x& - sel#(3)
      y# = y& - sel#(4)
      IF ABS(y#) > 1 THEN
        sel#(7) = arctan#(x#, y#) - (pi# / 2!)
        sel#(6) = SQR(x# ^ 2 + y# ^ 2)
      ELSE
        sel#(7) = 0
      END IF
    CASE ELSE
  END SELECT
ELSEIF FNcol%(cursorx%) > 79 AND button% = 1 THEN ' scroll y
  IF FNrow%(cursory%) > 3 AND FNrow%(cursory%) < 24 THEN
    ratio# = ((cursory% - 2 * ypixel%) / (22! * ypixel%))
    center# = (child#(level%, 4) + yoff# + child#(level%, 6)) - (ratio# * child#(level%
    ds# = (sy2# - sy1#) / 2
    sy1# = center# - ds#
    sy2# = center# + ds#
    CALL plotall(0)
  END IF
ELSEIF FNrow%(cursory%) > 24 AND button% = 1 THEN ' scroll x
  IF FNcol%(cursorx%) > 2 AND FNcol%(cursory%) < 79 THEN
    ratio# = (cursorx% - xpixel%) / (79! * xpixel%)
    center# = (child#(level%, 3) + xoff#) + (ratio# * child#(level%, 5))
    ds# = (sx2# - sx1#) / 2
    sx1# = center# - ds#
    sx2# = center# + ds#
    CALL plotall(0)
  END IF
END IF
CALL suboff(sel#())
END SUB

```

```
'-----  
SUB setbyte (byte%, l$) STATIC  
'-----  
MID$(l$, byte% + 1, 1) = CHR$(255)  
END SUB
```

```

-----
SUB setgrid STATIC
-----
IF xstep& = 0 OR ystep& = 0 THEN
    ystep& = 10000
    xstep& = 10000
END IF

t%(1) = 0: l%(1) = 20: v#(1) = xorigin&: p$(1) = "X Origin of Grid? "
t%(2) = 0: l%(2) = 20: v#(2) = yorigin&: p$(2) = "Y Origin of Grid? "
t%(3) = 0: l%(3) = 20: v#(3) = xstep&: p$(3) = "X Grid Step? "
t%(4) = 0: l%(4) = 20: v#(4) = ystep&: p$(4) = "Y Grid Step? "

c% = 0
CALL query("Set Grid Position and Size", 4, c%)

SELECT CASE c%
CASE 27
CASE ELSE
    SELECT CASE gridon%
    CASE 1: CALL grid(0)
    CASE ELSE
    END SELECT

    xorigin& = FIX(v#(1))
    yorigin& = FIX(v#(2))
    xstep& = INT(v#(3))
    ystep& = INT(v#(4))

    IF (xstep& > 0) AND (ystep& > 0) THEN
        gridon% = 1
        CALL grid(1)
    ELSE
        gridon% = 0
    END IF
END SELECT

END SUB

```

```

SUB setlines (rec#(), lineseg&(), m!(), b&()) STATIC

CALL corners(rec#(), cx1#, cy1#, cx3#, cy3#, cx2#, cy2#, cx4#, cy4#)
lineseg&(1, 1) = cx1#: lineseg&(1, 2) = cx2#
lineseg&(1, 3) = cy1#: lineseg&(1, 4) = cy2#
lineseg&(2, 1) = cx2#: lineseg&(2, 2) = cx3#
lineseg&(2, 3) = cy2#: lineseg&(2, 4) = cy3#
lineseg&(3, 1) = cx3#: lineseg&(3, 2) = cx4#
lineseg&(3, 3) = cy3#: lineseg&(3, 4) = cy4#
lineseg&(4, 1) = cx4#: lineseg&(4, 2) = cx1#
lineseg&(4, 3) = cy4#: lineseg&(4, 4) = cy1#
CALL winon
FOR s% = 1 TO 4
    x1& = lineseg&(s%, 1)
    x2& = lineseg&(s%, 2)
    y1& = lineseg&(s%, 3)
    y2& = lineseg&(s%, 4)
    SELECT CASE rec#(7)
    CASE 0: m!(s%) = 0
    CASE ELSE: m!(s%) = ((y2& - y1&) * 1!) / ((x2& - x1&) * 1!)
    END SELECT
    b&(s%) = FIX(y1& - m!(s%) * x1&)
NEXT
END SUB

```

```
!-----  
FUNCTION short$ (s$) STATIC  
!-----  
short$ = RTRIM$(LTRIM$(s$))  
END FUNCTION
```

```

-----
SUB show (rec#()) STATIC
-----
todo% = 0
drw% = 1
irec# = 1
WHILE todo% <> 27
    SELECT CASE drw%
    CASE 1: FOR i% = 3 TO 24
        LOCATE i%, 2
        r# = i% + irec# - 3
        PRINT LEFT$(STR$(r#) + blank$, 78);
        IF r# < child#(level%, 2) THEN
            CALL readrec(r#, rec#())
            FOR j% = 1 TO 7
                SELECT CASE j%
                CASE 1, 2
                    LOCATE i%, j% * 6 + 2
                    v$ = STR$(rec#(j%))
                CASE 3, 4, 5, 6
                    LOCATE i%, (j% - 3) * 10 + 22
                    v$ = value$(rec#(j%))
                CASE 7
                    LOCATE i%, (j% - 3) * 10 + 22
                    v$ = STR$(degrees$(rec#(7)))
                END SELECT
                PRINT v$;
            NEXT
        END IF
    NEXT
END SELECT

DO
    todo% = ASC(RIGHT$(CHR$(0) + INKEY$, 1))
LOOP UNTIL todo% <> 0

SELECT CASE todo%
CASE 71: irec# = 1
CASE 81: irec# = irec# + 22: drw% = 1
CASE 73: irec# = irec# - 22: drw% = 1
CASE ELSE
END SELECT

IF irec# <= 0 THEN irec# = 1

WEND
PCOPY 1, 0
END SUB

```



```

!-----
SUB switch (f%, c%) STATIC
!-----
CALL winoff
notdone% = 1
row% = CSRLIN
col% = POS(0)
COLOR ifor1%
WHILE notdone%
    c% = 0
    LOCATE row%, col%
    PRINT CHR$(f% * 88 + (1 - f%) * 249);
    WHILE c% = 0
        c$ = INKEY$
        CALL cursor(c$, insert%)
        c% = ASC(RIGHT$(CHR$(0) + c$, 1))
    WEND
    SELECT CASE LEN(c$)
    CASE 1
        SELECT CASE c%
        CASE 13, 9, 27: notdone% = 0
        CASE ELSE: f% = 1 - f%
        END SELECT
    CASE 2
        SELECT CASE c%
        CASE 15, 72, 80, 81, 71: notdone% = 0
        CASE ELSE: f% = 1 - f%
        END SELECT
    END SELECT
WEND
END SUB

```

```

-----
SUB systemstatus (rec#()) STATIC
-----
CALL winon
CLS 1
LOCATE 4, 4: PRINT "Root Cell: "; MKD$(child#(0, 8)), FIX(child#(0, 5)); "by"; FIX(child#(0, 6)); "
LOCATE 5, 4: PRINT "Edit Cell: "; MKD$(child#(level%, 8)), FIX(child#(level%, 5)); "by"; FIX(child#
LOCATE 7, 4: PRINT "Current Level: "; level%; " Expand Level: "; explevel%
LOCATE 8, 4: PRINT "Default Layer: "; layer%
LOCATE 10, 4: PRINT "Total Offsets of Edit Cell"
LOCATE 11, 4: PRINT "    x axis: "; xoff#
LOCATE 12, 4: PRINT "    y axis: "; yoff#
LOCATE 13, 4: PRINT "    theta: "; toff# * 180! / pi#
LOCATE 15, 4: PRINT "Screen Coordinates:"
LOCATE 16, 4: PRINT "    lower left: "; FIX(sx1#); FIX(sy1#);
LOCATE 17, 4: PRINT "    upper right: "; FIX(sx2#); FIX(sy2#);
LOCATE 19, 4: PRINT "File Coordinates:"
LOCATE 20, 4: PRINT "    lower left: "; FIX(child#(level%, 3)); FIX(child#(level%, 4));
LOCATE 21, 4: PRINT "    dimensions: "; FIX(child#(level%, 5)); FIX(child#(level%, 6));
LOCATE 23, 4: PRINT "Selected Rectangles: "; recs%; " Cells: "; cells%
a$ = INPUT$(1)
PCOPY 1, 0
END SUB

```

```
-----  
FUNCTION timer# STATIC  
-----  
DIM inreg AS regtype, outreg AS regtype  
STATIC hr#, mn#, sc#, hd#  
inreg.ax = mki%(&H2C, 0)  
inreg.bx = 0  
CALL interruptx(&H21, inreg, outreg)  
hr# = hiByte%(outreg.cx)  
mn# = loByte%(outreg.cx)  
sc# = hiByte%(outreg.dx)  
hd# = loByte%(outreg.dx)  
timer# = ((hr# * 60 + mn#) * 60 + sc#) * 100 + hd#  
END FUNCTION
```

```

'-----
FUNCTION units# (v$)
'-----
u$ = LCASE$(RIGHT$(RTRIM$("nm" + v$), 2))
SELECT CASE u$
CASE "cm": u# = 1E+07
CASE "mm": u# = 1000000!
CASE "um": u# = 1000!
CASE "nm": u# = 1!
CASE "wl"
    SELECT CASE wavelength&
    CASE IS <= 0
        BEEP
        LOCATE 1, 1: PRINT "No Value for Wavelengths Has Been Specified";
        u& = 1000
    CASE ELSE
        u# = wavelength&
    END SELECT
CASE ELSE
    SELECT CASE wavelength&
    CASE IS <= 0: u# = wavelength&
    CASE ELSE: u# = 1
    END SELECT
END SELECT
units# = u#
END FUNCTION

```

```

-----
FUNCTION value$ (v#) STATIC
-----
power% = FIX(LOG(ABS(v# + .1)) / LOG(10))
l% = LEN(LTRIM$(STR$(FIX(v#)))) + 1
SELECT CASE power%
CASE IS < 3: p# = 1: u$ = " nm"
CASE 3 TO 5: p# = 1000: u$ = " um"
CASE 6 TO 7: p# = 1000000: u$ = " mm"
CASE IS > 7: p# = 10000000: u$ = " cm"
END SELECT
value$ = LEFT$(LTRIM$(STR$(v# / p#)), l%) + u$
END FUNCTION

```

REFERENCES

1. Hartman, Clinton S. Private communication. Hartman Research Inc., Dallas, Texas, 1987, 1988.
2. Malocha, Donald C. Private communication. College of Engineering, University of Central Florida, Orlando, 1987.
3. Abbott, Benjamin P. Private communication. College of Engineering, University of Central Florida, Orlando, 1987.
4. Mukherjee, Amar. "Architecture and Design of VLSI" Lecture notes, College of Computer Science, University of Central Florida, Orlando, 1987.
5. Mead, Caver; Conway, Lynn. Introduction to VLSI Systems. Philippines: Addison-Wesley Publishing Company Inc., 1980.
6. Scott, Walter S.; Hamachi, Gordon; Ousterhout, John; Mayo, John N. "1985 VLSI Tools." Unpublished Technical Report, Computer Science Division (EECS), University of California, Berkeley, 1985.