

AL/CAD : a computer aided design system for log house construction
by Ray Spade Babcock

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science
Montana State University
© Copyright by Ray Spade Babcock (1987)

Abstract:
National Log Construction Company, a company created and run by Stephen D. Babcock in Thompson Falls, Montana encouraged a study to determine the feasibility of using a computer to design log house kits. This study was undertaken in two parts. A prototype system called Automatic Design System (ADS) was developed during the time period 1972-1978. A final system called Air-Lock Computer Aided Design (AL/CAD) was developed during the time period 1982-1987. The ADS system was developed for a Prime multiuser computer system using a Tektronix 4006-1 storage display screen. The AL/CAD system was developed for an IBM PC-AT clone machine running the MS-DOS operating system. The AL/CAD system extensively uses windowing techniques to enhance user interaction. Both of these systems are described. Examples of the earlier ADS command set and display techniques are provided. The AL/CAD system is shown with its expanded data structure and windowing design. Both systems are evaluated, and future growth is projected.

AL/CAD: A COMPUTER AIDED DESIGN SYSTEM FOR LOG HOUSE CONSTRUCTION

by

Ray Spade Babcock

A thesis submitted in partial fulfillment
of the requirements for the degree

of

Master of Science

in

Computer Science

MONTANA STATE UNIVERSITY
Bozeman, Montana

November 1987

APPROVAL

of a thesis submitted by

Ray Spade Babcock

This thesis has been read by each member of the thesis committee and has been found to be satisfactory regarding content, English usage, format, citations, bibliographic style, and consistency, and is ready for submission to the College of Graduate Studies.

11/24/87
Date

Rockford J. Ross
Chairperson, Graduate Committee

Approved for the Major Department

Nov 24th 1987
Date

J. Dunbigh Stanley
Head, Major Department

Approved for the College of Graduate Studies

12-7-87
Date

Graduate Dean

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a master's degree at Montana State University, I agree that the Library shall make it available to borrowers under the rules of the Library. Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgement of source is made.

Permission for extensive quotation from or reproduction of this thesis may be granted by my major professor, or in his/her absence, by the Dean of Libraries when, in the opinion of either, the proposed use of the material is for scholarly purposes. Any copying or use of the material in this thesis for financial gain shall not be allowed without my written permission.

Signature Ray S Babcock

Date 12/2/87

To my parents
Stephen and Daun
who inspired and encouraged me.

TABLE OF CONTENTS

Page

APPROVAL PAGE. ii

STATEMENT OF PERMISSION TO USE iii

DEDICATION. iv

TABLE OF CONTENTS v

LIST OF TABLES vi

LIST OF FIGURES. vii

ABSTRACT ix

1. INTRODUCTION 1

2. PROTOTYPE AUTOMATIC DESIGN SYSTEM 7

Automatic Design System User Interface. 7
Automatic Design System Data Structure. 18
Evaluation of Automatic Design System 20

3. AL/CAD SYSTEM. 22
AL/CAD User Interface 22
AL/CAD Data Structure 36
AL/CAD Boxing Algorithm. 46
AL/CAD Splicing Algorithm 48

4. CONCLUSIONS 52

Evaluation of Microsoft Windows 52
Evaluation of Microsoft Windows Development Tool Kit. 53
Evaluation of AL/CAD 54

BIBLIOGRAPHY. 56

LIST OF TABLES

Page

1. ADS User Commands 9
2. ADS Commands For Sample House Design 15
3. Job Record. 40
4. Plan Record 41
5. Miscellaneous Record 41
6. Wall Record 42
7. Opening Record 42
8. Intersection Record 43
9. Box Record. 43
10. Gable Record. 43
11. Row Record 44
12. Piece Record 44
13. Cut Record. 45
14. AL/CAD Data Structure Manipulation Algorithms 45

LIST OF FIGURES

Page

1. An Air-Lock Log House 2
2. Cross Section of an Air-Lock Log 2
3. ADS Screen Layout 8
4. ADS Screen After Four Commands 10
5. Sample ADS Design of Four Walls 13
6. Sample ADS House With Openings 14
7. Sample ADS House After LAYOUT Command 16
8. Sample ADS Elevation View 17
9. Sample ADS House With Gable 18
10. Sample ADS House After LAYUP Command 19
11. AL/CAD Screen Layout 23
12. System Menu Being Selected 24
13. Selecting The Close Option 25
14. AL/CAD Mode Menu 26
15. AL/CAD Grid Menu 26
16. AL/CAD Create Menu 27
17. AL/CAD Phase Menu 27
18. AL/CAD Move Menu 28

LIST OF FIGURES - Continued

Page

19. AL/CAD View Menu 28
20. AL/CAD Delete Menu. 28
21. AL/CAD Edit Menu. 29
22. AL/CAD File Menu. 29
23. AL/CAD Utility Menu 30
24. AL/CAD Debug Menu. 30
25. Selecting Create Wall Option. 31
26. Creating A Vertical Wall 32
27. Positioning To A Middle Wall 33
28. Adding The Middle Wall 34
29. Adding A Window To A Wall 35
30. AL/CAD Data Structure. 38
31. A Wall Record And Opening List 39
32. Four Boxes In An Elevation 47
33. Splicing One Box 51

ABSTRACT

National Log Construction Company, a company created and run by Stephen D. Babcock in Thompson Falls, Montana encouraged a study to determine the feasibility of using a computer to design log house kits. This study was undertaken in two parts. A prototype system called Automatic Design System (ADS) was developed during the time period 1972-1978. A final system called Air-Lock Computer Aided Design (AL/CAD) was developed during the time period 1982-1987. The ADS system was developed for a Prime multi-user computer system using a Tektronix 4006-1 storage display screen. The AL/CAD system was developed for an IBM PC-AT clone machine running the MS-DOS operating system. The AL/CAD system extensively uses windowing techniques to enhance user interaction. Both of these systems are described. Examples of the earlier ADS command set and display techniques are provided. The AL/CAD system is shown with its expanded data structure and windowing design. Both systems are evaluated, and future growth is projected.

CHAPTER 1

INTRODUCTION

Log house construction is a tradition in America. Everyone wants a cabin in the hills, by a lake, near a stream or just out in the cool fresh country away from town. Cutting your own logs in the forest and carving them into a house is a complicated job. In recent years, various log house manufacturing companies have been created to solve this problem for the majority of log house customers. I was privileged to be born into a family that owned and operated such a company.

On March 21, 1946, my father, Stephen D. Babcock, signed papers to purchase a patent for a new kind of log house construction. He with three other partners established a company called National Log Construction Company in Thompson Falls, Montana, to manufacture this product. The material used for the houses was called "Air-Lock" logs. An example of an Air-Lock log house is shown in Figure 1.

This new kind of construction method involved the drilling of a three inch diameter hole down the length of a log that had been previously turned to a constant diameter of 6, 7, or 8 inches. A cross section of an Air-Lock log is shown in Figure 2.

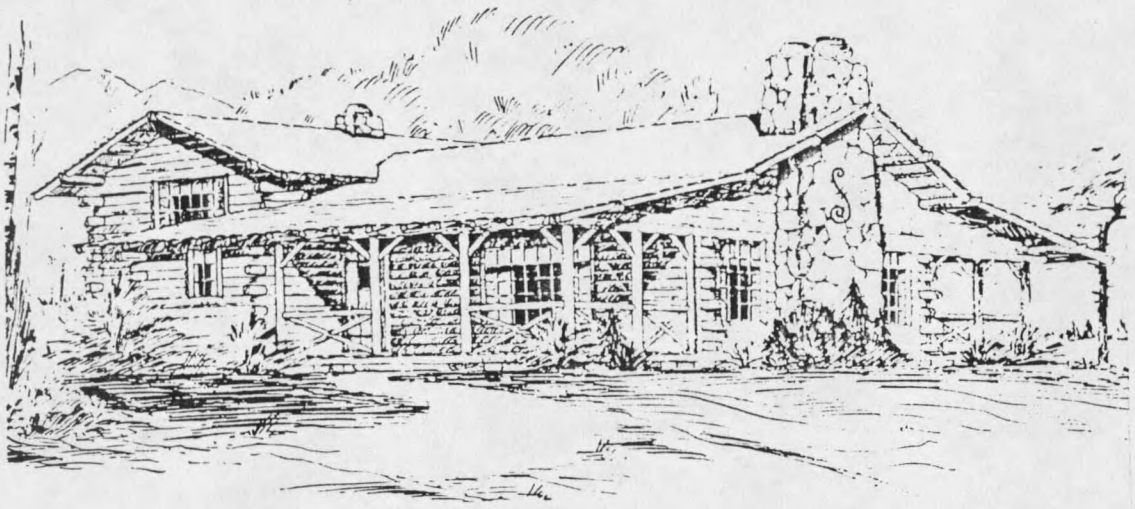

Figure 1. An Air-Lock Log House.

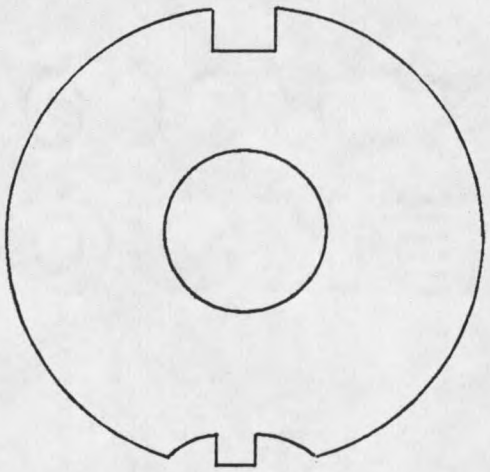

Figure 2. Cross Section of an Air-Lock Log

When a solid log dries it dries from the outside only. The outside shrinks more quickly than the inside and the log splits open in long longitudinal cracks. The three inch hole in the Air-Lock process allows the

log to dry from the inside as well as from the outside thus assuring a completely dry log and a reduction in "checking" or cracking. The "Air-Lock" process provides for a well seasoned log product of a high dimensional stability.

In designing the plans for an "Air-Lock" log house, a set of floor plans is drawn from customer supplied specifications. Wall elevations are then drawn and the various pieces that will be manufactured are designed. A list of material is used by the factory to manufacture the required pieces for a design. The pieces are marked and shipped to the customer site where they are assembled into a log house.

The design process is complicated and time consuming. It is desired to speed-up this process and reduce the number of errors that occurred. A computer system was proposed and developed. This development consisted of a prototype system developed during the time period 1972-1978 called Automatic Design System or ADS. This package was not successful and a newer system development project was begun during the time period 1982-1987. The newer system was called Air-Lock Computer Aided Design or AL/CAD.

AL/CAD is a highly graphical user interface to a set of design tools. It is more successful than the ADS system and has produced usable results. In preparing to develop the AL/CAD system, several system requirements were important. These were: portability, a user friendly interface, development in a programming language with dynamic memory allocation, use of a window environment, use of a pointing device, use of an output device, speed of design, and waste reduction. These are described below.

In considering portability, the system needed to be portable between various locations. The IBM PC-AT was chosen as the target machine. Any

IBM-AT or compatible clone having the following characteristics would run the package: an 80286 cpu, 640 K bytes of main memory with an EGA graphics board and a monochrome or color monitor, and a Microsoft Mouse.

Since the users of this package were unfamiliar with computers, the friendly nature of a windowing system was chosen. Due to the highly dynamic nature of the data structure chosen for AL/CAD, a programming language supporting dynamic memory allocation was needed. In addition, the language needed to be available on the IBM PC-AT or its clones. Microsoft's version 4.0 of the C programming language was chosen.

Only one window environment was available for the target machine. This was Microsoft Windows. In order to develop an application for Windows, the Microsoft Windows Development Tool-kit was obtained.

Due to the highly graphic nature of the Air-Lock Log house design system, a pointing device usable by a person not familiar with computers was needed. The mouse device was chosen. To provide the highest possible compatibility with Microsoft Windows, the Microsoft mouse was selected.

In the first release of the AL/CAD package, hard copy output was not needed. However, it was felt that some form of screen print utility was needed. A dot matrix printer with a graphics mode was chosen to allow the user to make a copy of the screen at any time. Future versions will produce output for a digital plotter. This will allow complete full sized plans to be drawn.

The normal manufacturing process for Air-Lock logs has the potential to produce a large amount of very expensive waste material in the form of raw material pieces of a length unused in the particular house being manufactured. A final goal of AL/CAD was to reduce this waste factor. Only a small

reduction would completely pay for the computer system in one year.

The AL/CAD system allows a user to generate a complete factory list of material and cut-out guides from a simple floor plan sketch. There are five main phases to AL/CAD: the floor plan phase, the layout phase, the elevation phase, the layup phase, and the list of material phase.

During the floor plan phase the floor plan is drawn to scale and the window and door openings are added. During the layout phase all walls are intersected and the intersection records are created in the data structure. In the elevation phase the height and width of each elevation is checked and/or modified. The window and door opening heights are verified and each intersection is checked. Gables, the sloped portion of a wall beneath the roof and above the wall, can be added to the walls during this phase and blank row records are added to the data structure.

The layup phase is the most complicated of all the phases. During this phase, the walls are boxed and spliced. Boxing is the process of dividing an elevation into separate boxes of wood material that contains no openings. Since the longest log manufactured is 8 feet 3 inches, each row longer than this must be spliced. A splice is the joining of two logs at their ends. As the walls are spliced, individual piece records are created and added to the data structure.

The last phase is the list of material phase. In this phase the complete factory cutting information and customer list of materials is generated. The list of material shows the number and sizes of all log pieces in the building.

The AL/CAD system makes numerous checks during its various phases to prevent missing piece errors introduced by the user. Waste reduction tables are maintained to allow the splicing algorithm to re-use odd shaped left-overs

from previous log allocations. The raw material is limited to 8 foot 3 inch lengths. All other log pieces must be less than or equal to this length. For example, if there is a 5 foot 0 inch space between two windows, the obvious solution is to provide a 5 foot 0 inch log to fill this space. However, for each 5 foot 0 inch piece cut, there is a 3 foot 3 inch piece left over. If another area in the building does not make use of this piece, then it may go to waste. Later, when splicing a 7 foot 0 inch length, the system may choose to use a 4 foot 0 inch length and a 3 foot 0 inch length rather than cutting a 7 foot 0 inch length from an 8 foot 3 inch length in order to conserve material. However, since every splice weakens the wall, the number of splices needs to be kept to a low. The AL/CAD program attempts to balance the need for splicing with the need for waste reduction.

CHAPTER 2

PROTOTYPE AUTOMATIC DESIGN SYSTEM

The first version of the Air-Lock Computer Aided Design System was called Automatic Design System or ADS. ADS was developed during the time period 1972-1974. It consisted of over one hundred subroutines written in the FORTRAN-66 programming language. The ADS package was designed for the following hardware: a Prime 300 cpu with 128 K Ram memory, a Tektronix 4006-1 storage tube graphics display, a Tektronix 4662 flat bed X-Y plotter, a Pertec 10Mb removable cartridge disk unit, and a Perkin Elmer 1200 crt terminal.

Automatic Design System User Interface

Since the main interaction with the user was to be via a storage tube graphics device, the user interface had to allow for the inability of the terminal to erase any one part of its screen. If the screen was to be changed, the entire screen had to be erased and completely redrawn. Figure 3 shows the three separate areas on the face of the terminal.

The main command input area was three lines at the bottom of the screen. The error message output area was one line at the top of the screen. The remainder of the screen was to be used to draw the graphics portion of the data. In order to provide for enough commands at the bottom of the screen a special input command parser was developed. The prompt for a new command was an open parenthesis "(". The cursor was then moved to the

space to the right of the parenthesis. As each character of a command was typed by the user, the character was evaluated by the command parser and assembled into a valid command.

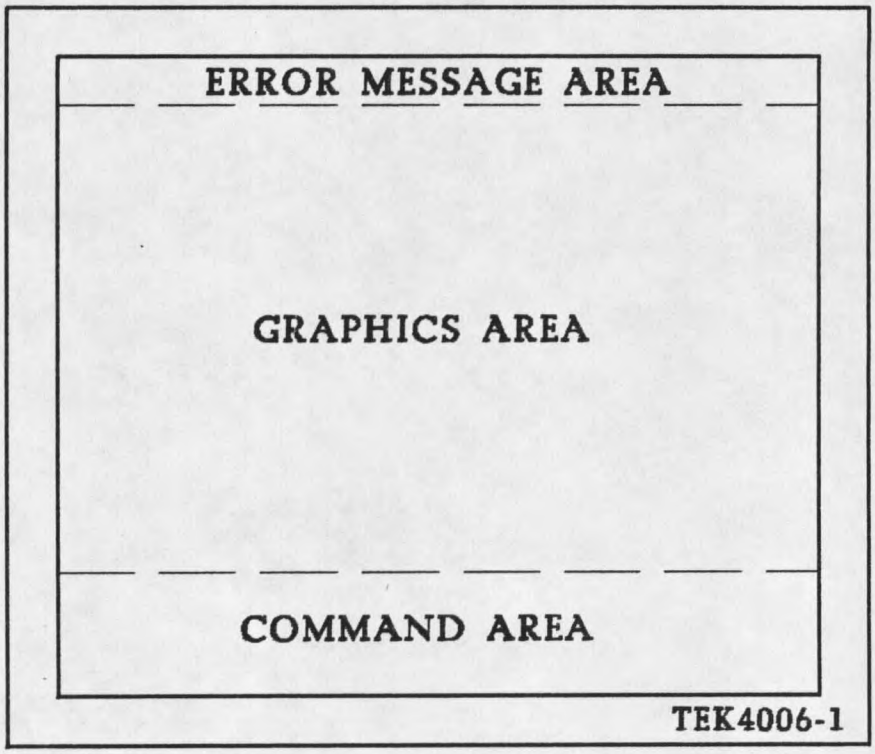

Figure 3. ADS Screen Layout.

If an incorrect character was typed, a bell sounded and no character was reflected to the screen. If a correct character was typed, it was reflected to the screen and the cursor was moved one space to the right. This allowed for the packing of many commands on a single display line. After a command was finished and the operator pressed the return key, a closing parenthesis ")" was reflected to the operator. The command was executed by other routines within

ADS and the parser was then called again. A new open parenthesis was sent to the screen to tell the operator that the system was ready for the next command. Table 1 gives a list of the ADS commands.

Table 1. ADS User Commands.

Command	Description
MOVE	Move Pen.
UP	Define A Wall In UP Direction.
DOWN	Define A Wall In DOWN Direction.
LEFT	Define A Wall In LEFT Direction.
RIGHT	Define A Wall In RIGHT Direction.
ANGLE	Define A Wall At An Angle.
DOOR	Define A Door Opening.
WINDOW	Define A Window Opening.
WALL	Define An Interior Wall.
OPEN	Define A Non-Standard Opening.
DELETE	Delete A Feature.
DRAW	Draw The Design On The Screen.
PLOT	Draw The Design On The Plotter.
QUIT	Exit The Design Process.
JOB	Load Data From An Existing Job File.
LIST	Generate The List Of Material.
LAYOUT	Intersect All The Walls.
LAYUP	Calculate Log Splices.
ELEVATE	Draw The Elevation View Of A Wall.
POV	Swap The Point Of View.

Figure 4 shows the screen after four commands defining a wall were typed. Note in the command input area, the four commands were packed together on one line.

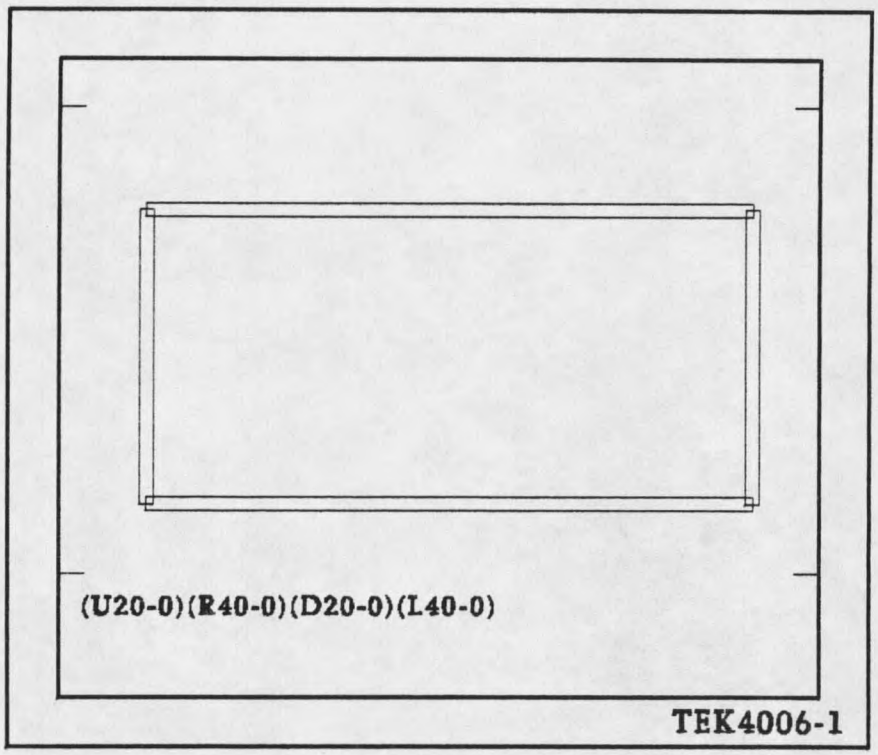

Figure 4. ADS Screen After Four Commands.

The command parser system allowed the user to see previous commands that had been typed and provided a more "friendly" interpretation of each keystroke as it was typed. If any error occurred, an error message was displayed at the top of the screen. When the operator took corrective action, this message and all the information on the screen was cleared by a screen

clear function. Then the correct graphics data was redrawn and a new command prompt was sent to the screen.

In using ADS a special length code was chosen that was the same as the length code used in manufacturing. Each length was represented by a three part code separated by dashes, F-I-E. The first number was an integer representing the number of feet, the second number was an integer from 0 to 11 representing the number of inches, and the third number was an integer from 0 to 7 representing the number of eighths of an inch. This measuring system was used through-out the design and manufacturing process. For example a log measuring 7 feet 6 and 1/2 inches, was denoted by a length code of 7-6-4. A log measuring 4 feet 11 and 1/8 inches was denoted by a length code of 4-11-1. The process for designing a house was as follows: 1. Draw the log walls of the floor plan with ADS in the plan mode., 2. Add window and door openings to the plan., 3. Intersect the walls at each corner with the LAYOUT command., 4. Build each elevation record with the ELEVATE command., 5. Add any gables needed to the elevation views., 6. Splice the walls with the LAYUP command., and 7. Generate the list of materials for the factory.

To define an individual wall, one of the wall commands was given. The individual wall command indicated the direction of the wall. Starting at the lower left of the screen the cursor was moved up or to the right by the operator. If, for example, the operator wished to draw a wall from the current location up to a new location, the UP command was used. The UP command, abbreviated U, was followed by a length code representing the desired length of the wall. For example, if the operator wished to define a

wall of length 24 feet 6 inches in the upward direction, any of the following commands would work:

U 24-6-0 or UP 24-6 or U 24-6

Note that trailing zeros in the length code could be omitted and that the command name could be abbreviated. After this command was parsed and evaluated, a rectangle defining the wall was drawn to scale on the screen. The cursor was moved to the opposite end of the wall from which the wall was defined. The operator could then proceed to define additional walls by entering additional wall commands. For example, to define another wall going to the right that was 34 feet 6 and 1/2 inches long, the following command could have been entered:

R 34-6-4

This command would have been packed on the command input line with the previous command as follows:

(U 24-6)(R 34-6-4)

To define the rest of a rectangular house, two additional walls would have required definition. The final graphics screen representing four walls would have been as shown in Figure 5. Note that the command line now reads:

(U 24-6)(R 34-6-4)(D 24-6)(L 34-6-4)

Figure 6 shows a similar illustration with a set of openings defined. The set of ADS commands required for the layout of Figure 6 with a short explanation of each command is given in Table 2.

Note that the commands required the operator to keep track of where he or she was on the diagram. This was one of the most limiting factors in the implementation of this package on the storage tube display terminal. The next step would have been to input the LAYOUT command. Figure 7 shows

the results of executing this command. Note that all the log walls were now shown as intersecting one another. This was the normal appearance of a log wall in plan view.

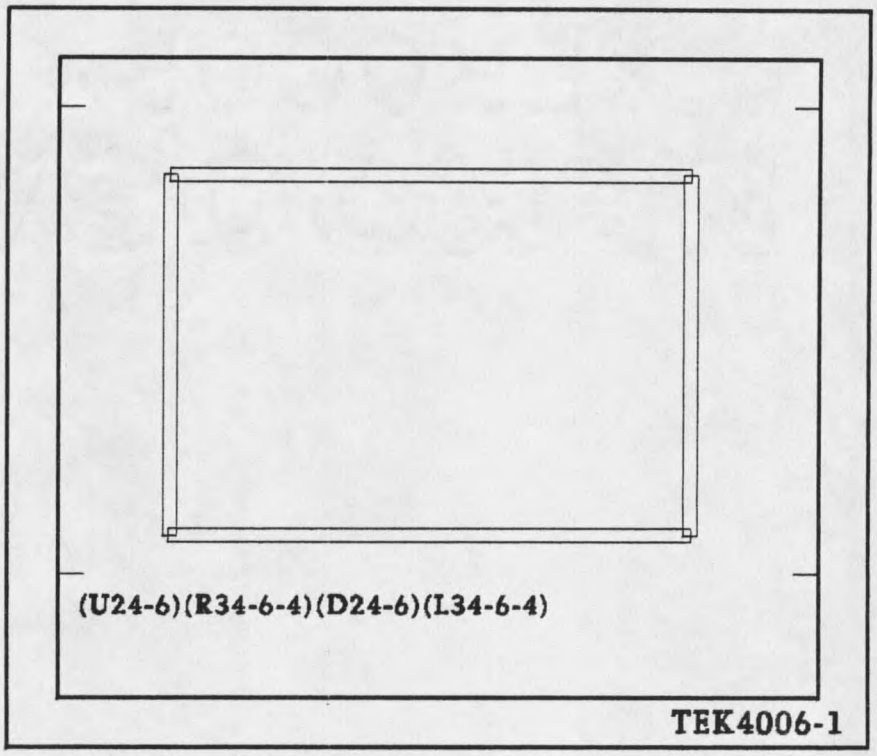

Figure 5. Sample ADS Design of Four Walls

The walls could then have been shown in "elevation" view by the use of the ELEVATE command. Each wall was numbered as it was originally defined. The elevate command referred to this number when defining the wall. For example to elevate the first wall defined the following command was used:

(E 1)

This would have caused the screen to clear and Figure 8 to be displayed.

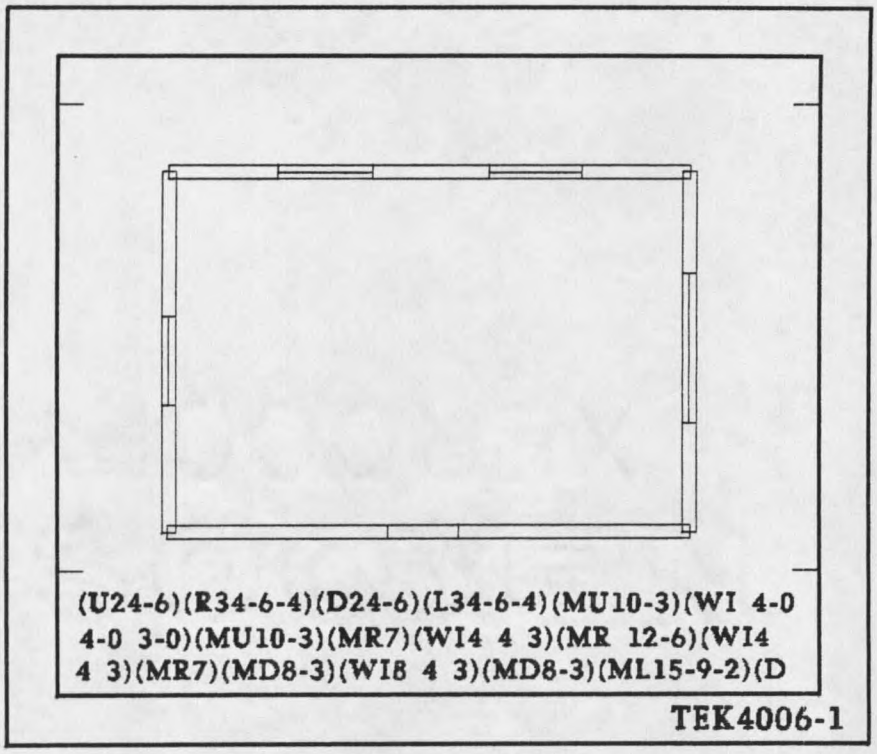

Figure 6. Sample ADS House With Openings.

Note that the opening was shown containing no horizontal lines. This indicated that no log material was to be used for this area, thereby creating an opening in the wall. The solid wall was indicated by the horizontal lines representing the individual log rows needed for of this particular elevation. This command did not produce a spectacular drawing, but it was extremely important to the design of the building on the computer. It was with this command that the row records of the original data structure were built.

Table 2. ADS Commands For Sample House Design.

Command	Action Taken
U 24-6	Draw a wall up 24 feet 6 inches.
R 34-6-4	Draw a wall right 40 feet 6 and 1/2 inches.
D 24-6	Draw a wall down 24 feet 6 inches.
L 34-6-4	Draw a wall left 34 feet 6 and 1/2 inches.
M U 10-3	Move up 10 feet 3 inches.
WI 4-0 4-0 3-0	Define a window that is 4 feet 0 inches wide by 4 feet 0 inches high and is 3 feet 0 inches from the floor.
M U 10-3	Move up 10 feet 3 inches.
M R 7	Move right 7 feet 0 inches.
WI 4 4 3	Define another window.
M R 12-6	Move right 12 feet 6 inches.
WI 4 4 3	Define another window.
M R 7	Move right 7 feet 0 inches.
M D 8-3	Move down 8 feet 3 inches.
WI 8 4 3	Define a window that is 8 feet 0 inches wide by 4 feet 0 inches high and 3 feet 0 inches from the floor
M D 8-3	Move down 8 feet 3 inches.
M L 15-9-2	Move left 15 feet 9 and 1/4 inches.
DOOR 3-0 6-8	Define a door 3 feet 0 inches wide by 6 feet 8 inches high.

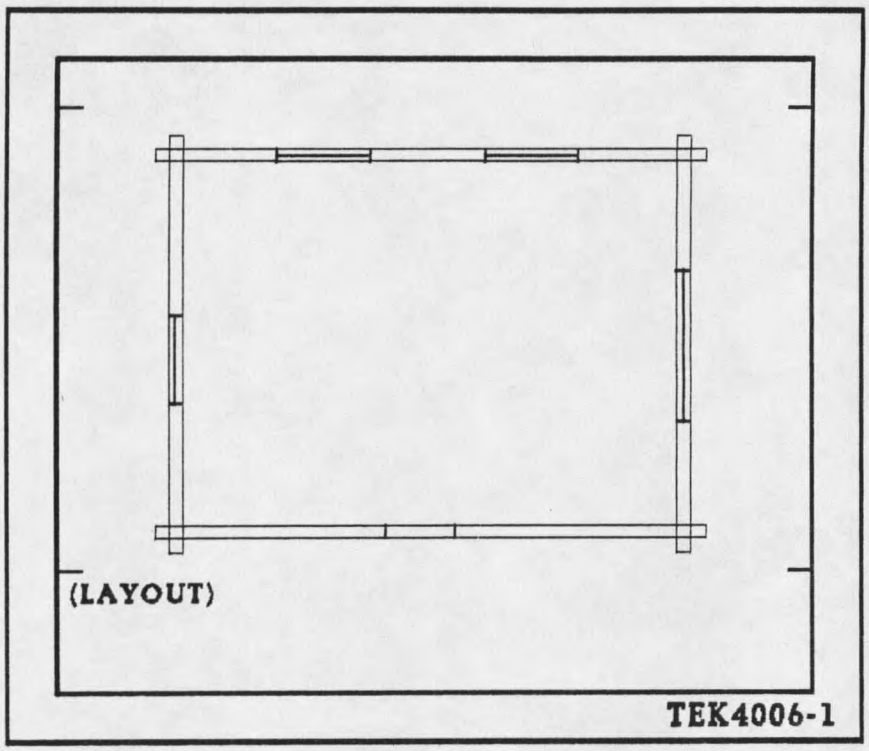

Figure 7. Sample ADS House After LAYOUT Command.

If a gable were to be added to this elevation, the GABLE command would have been executed as follows:

GABLE 4-12 12-3 -4-12 12-3

This command indicated that a gable would have been built from the left rising at a 4-12 pitch (4 feet high for each 12 feet in length) for a length of 12 feet 3 inches. Then the gable would have dropped at the same pitch for another 12 feet 3 inches, completing the wall. Figure 9 shows the results of this gable command.

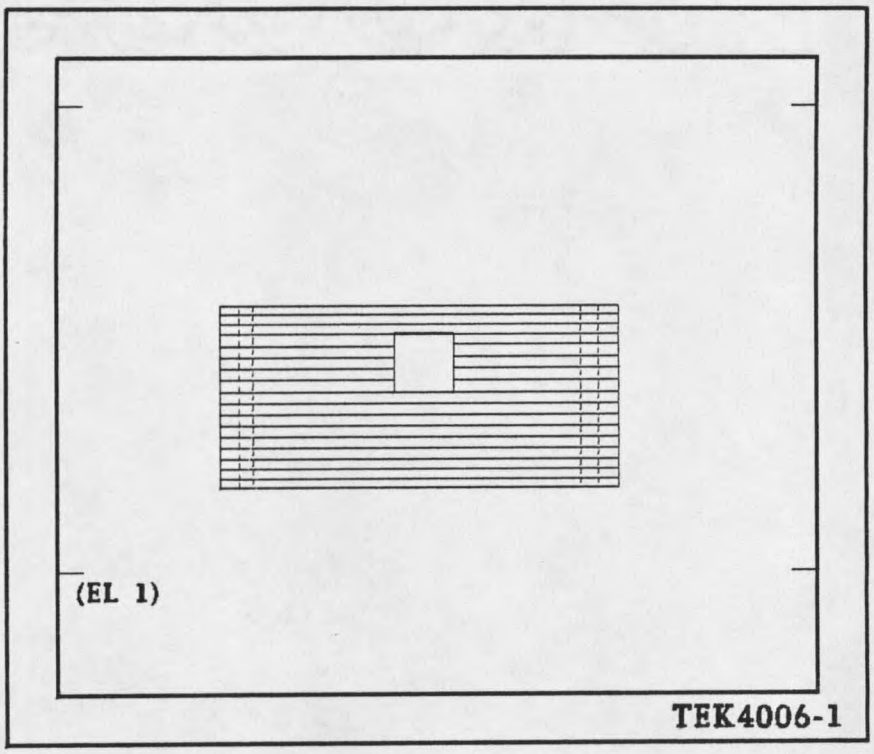

Figure 8. Sample ADS Elevation View.

After elevating all the walls, a LAYUP command was typed and the original attempt at splicing took place. Figure 10 shows wall one spliced as it would have been by the normal splicing algorithm. It was after the splicing was finished, that a list of individual logs could be created.

Generating the list of materials and drawing complete and dimensioned drawings of the individual walls was intended to provide the company with a tremendous aid in designing log houses.

This effort when done by hand for a nominal sized house would take a designer from two to seven days. ADS should have reduced this time to less than one day.

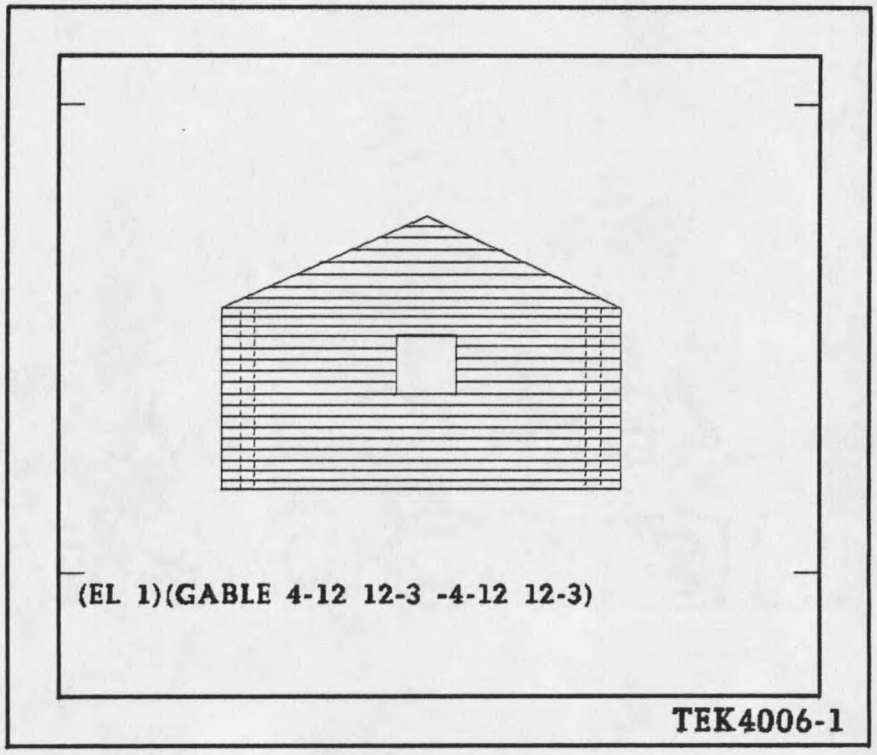

Figure 9. Sample ADS House With Gable.

Automatic Design System Data Structure

ADS used a proprietary file management system developed by Prime Computer Inc. called MIDAS. This stood for Multiple Indexed Data Access Scheme. The MIDAS system was extensive and provided high speed access to each of the parts needed by ADS.

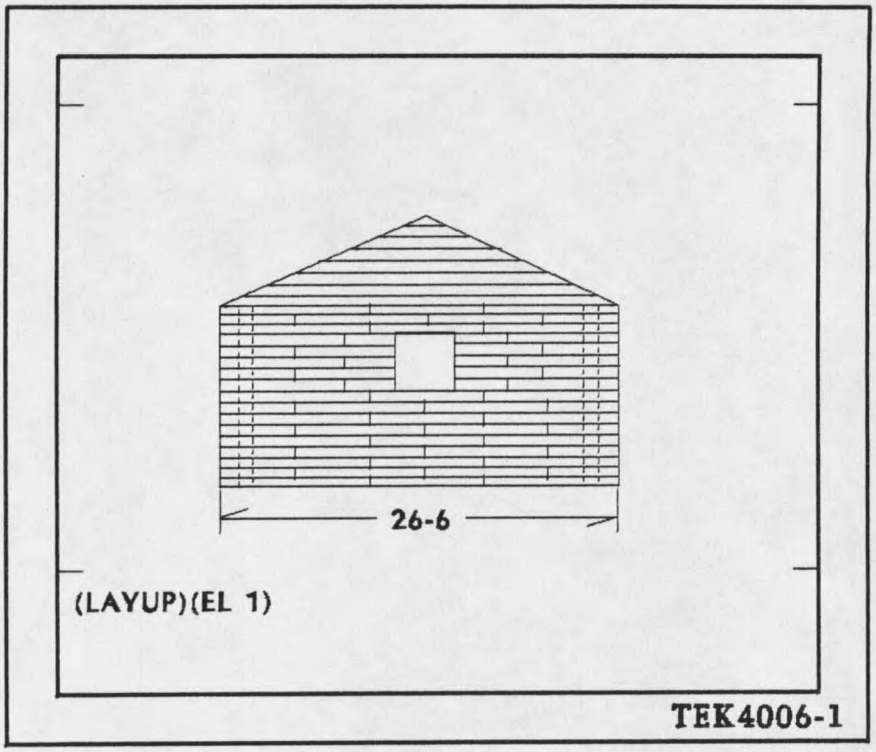

Figure 10. Sample ADS House After LAYUP Command.

It was not portable, however, and added quite a bit of overhead to the entire package. Also, during the design of a new building, MIDAS put most of the new records in overflow areas, and the indexing speed was not reduced until a reorganization of the file took place by executing a command called REMAKE for the file. After REMAKE was completed, the file was accessed very quickly and the individual records were stored in an efficient fashion. However, the user would forget to execute the REMAKE command and therefore the system continued to operate slowly.

MIDAS was a keyed file system. Each record had to have one primary key defined and allowed any number of secondary keys. These keys were used to build separate indices for the individual records allowing access in a number of different ways.

The following records were defined in the MIDAS implementation of ADS: Wall Records, Row Records, Piece Records, and Gable Records. Wall Records contained data about an individual wall including its length and graphics position. Row Records contained data about an individual row on and individual wall. All of the row records for a wall were connected to the Wall record for that wall. Piece Records were generated by the design system. This record contained data as to the length of the piece, which row it was in, and any special cuts that were to be made in the piece. Each gable in a house was treated as a separate entity in the MIDAS system and had its own Gable Record. Data concerning the pitch and length of the gable in addition to its graphics location were included.

The MIDAS system allowed an individual house to have lots of rows with a few walls or lots of walls with a few rows. In any combination, the space used was just the space needed for the data.

Evaluation of Automatic Design System

The ADS prototype version was never successful for two main reasons. The first was because it never spliced correctly. It was always making what could be interpreted by even an untrained human as "crazy" choices. Therefore, the human designer was always called upon to "clean up" after the

computer. The second reason that this system failed was its poor user interface. The commands were hard to remember, the results and the location of the current drawing position were hard to read.

CHAPTER 3

AL/CAD SYSTEM

The Air-Lock Computer Aided Design or AL/CAD was the final development effort in automating the Air-Lock log house design process. AL/CAD was developed during the time period 1982-1987. It consisted of a major new design for a linked list data structure, a new user interface utilizing a windowing environment, and a new set of computer equipment. The AL/CAD system was designed for the following hardware: a Multitec (IBM-AT clone) computer with 640 K bytes high speed memory and an Extended Graphics Adapter card, a monochrome display, a 30Mb secondary storage disk unit, a Microsoft Mouse pointing device, a dot-matrix printer with graphics capability, and the Microsoft versions of the C programming language, the Windows operating system, and the Windows Software Development Tool Kit. The AL/CAD system description that follows is divided into a discussion of the user interface, the data structure, the boxing algorithm, the splicing algorithm, and an evaluation of the development tools and the resulting software package.

AL/CAD User Interface

An important requirement for AL/CAD was that its user interface needed to be much more friendly to the user. In addition AL/CAD had to run on a widely available computer system. The IBM AT family of clone computers using the MS-DOS operating system was chosen as the computer and various windowing packages for this system were examined as a solution to the user

interface problem. Microsoft Windows in conjunction with the Microsoft Mouse was finally selected for this purpose. In addition, a Microsoft Windows Development Tool Kit was acquired to enable the development of a customized windows application.

How the AL/CAD package works on this system is now described. After initiating the AL/CAD package the screen shown in Figure 11 is displayed.

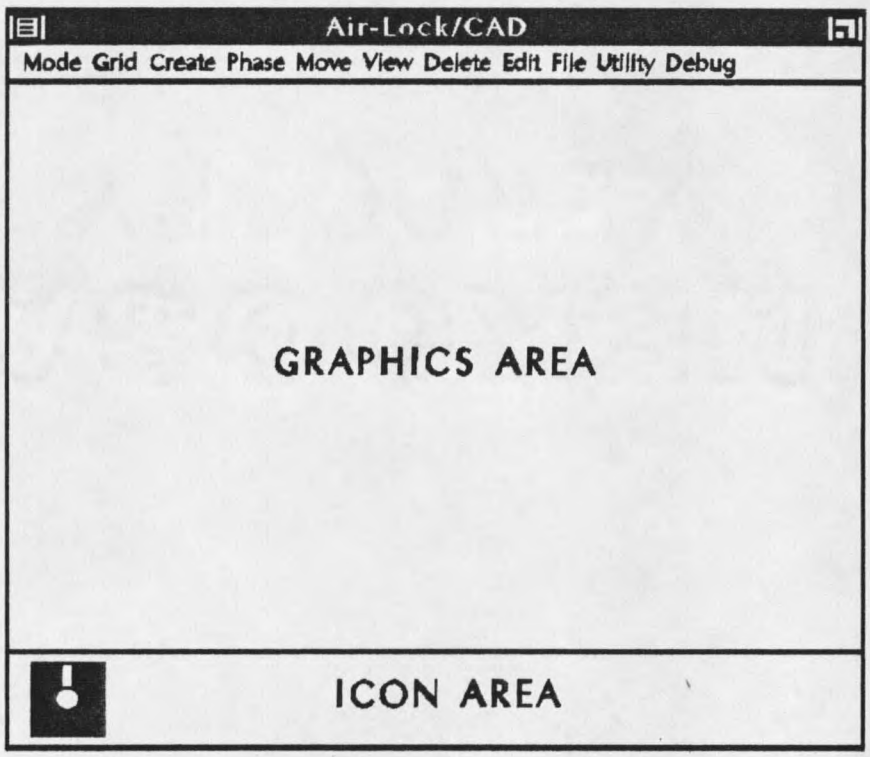

Figure 11. AL/CAD Screen Layout.

The title of the package is shown in the title bar area of the window. A "size" box is shown at the right hand top corner. This size box can be selected by the user (using the mouse) if it is desired to change the size of

the window. A system menu box is displayed at the upper left corner. The system menu is shown in Figure 12. (For details on the system menu box see the Microsoft Windows manual.) The CLOSE option will terminate the AL/CAD application and return the user to the control of the windows package. To return to MSDOS, a CLOSE selection is made using the system menu in the resulting MS-DOS window.

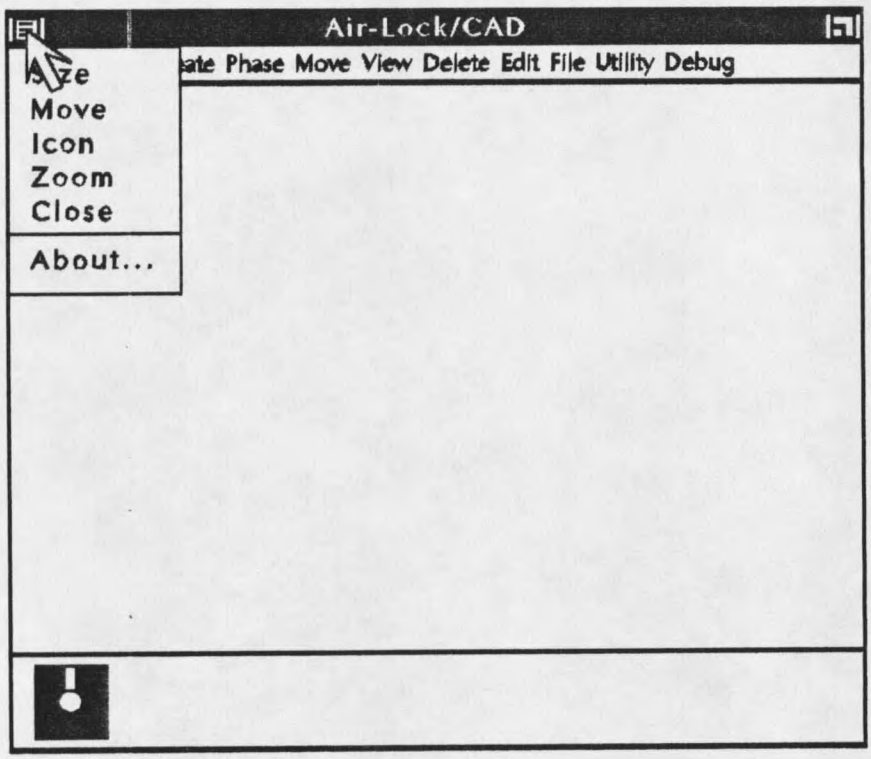

Figure 12. System Menu Being Selected.

Selections from the system menu are made by moving the mouse cursor to the iconic or picture representation of the system menu and pressing and holding the left mouse button. At this point a "pop-up" menu is displayed on

the screen. A pop-up menu is displayed temporarily over the information below it on the screen. After the selection is made, the window disappears and the information below the menu is displayed again. While holding the button and moving the mouse down through this menu, the different selections in the menu are highlighted. The highlighting is accomplished by displaying the entry in reverse video. The user makes a selection by releasing the left mouse button when the desired entry is highlighted. Figure 13 shows the user selecting the CLOSE option from the system menu.

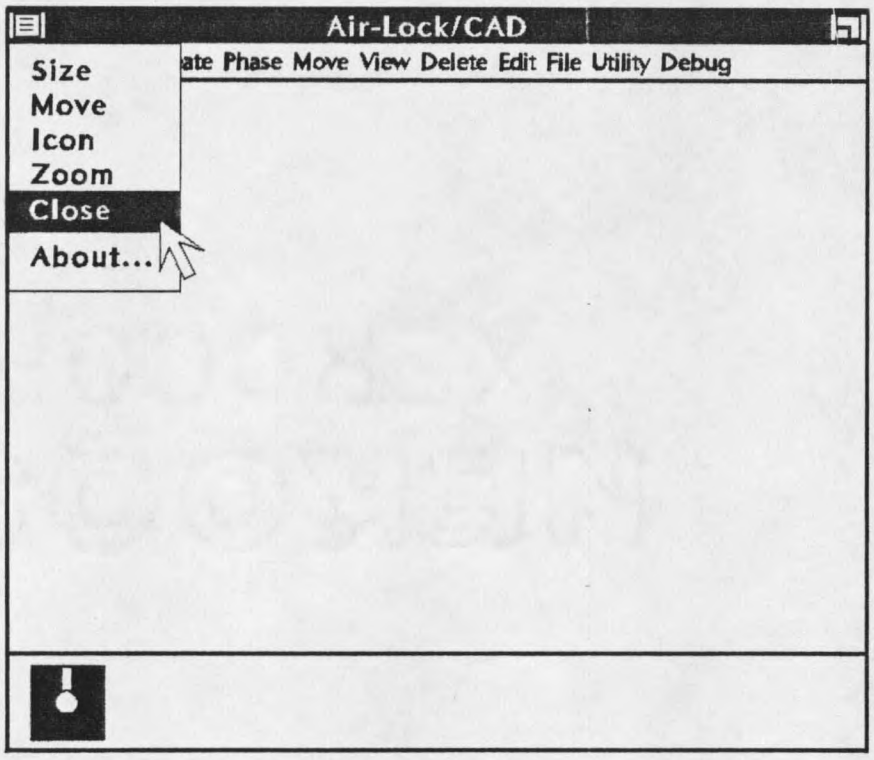

Figure 13. Selecting The Close Option.

The pop-up menu selection technique is used for all pop-up menus in the AL/CAD system. The AL/CAD high level menu consists of the eleven selections shown just below the title line on the screen. These selections are: Mode, Grid, Create, Phase, Move, View, Delete, Edit, File, Utility, and Debug. The pop-up menus that are displayed for each of these selections are shown in Figures 14 through 24.

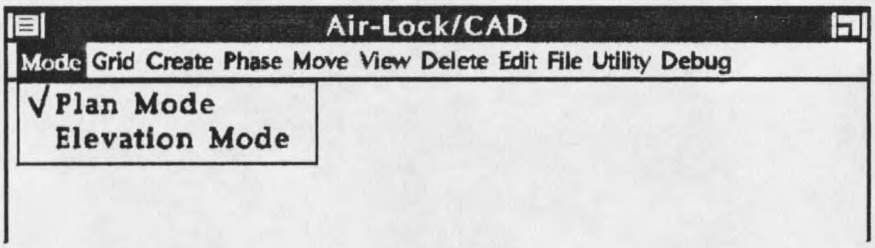

Figure 14. AL/CAD Mode Menu.

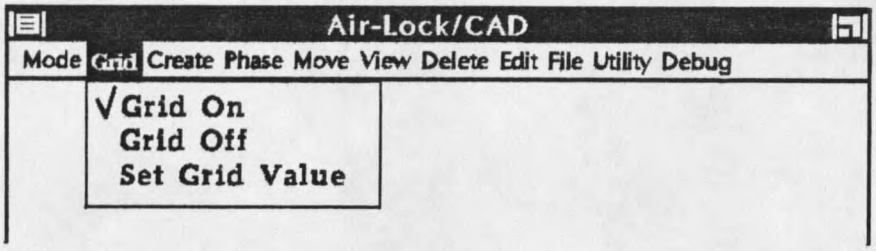

Figure 15. AL/CAD Grid Menu.

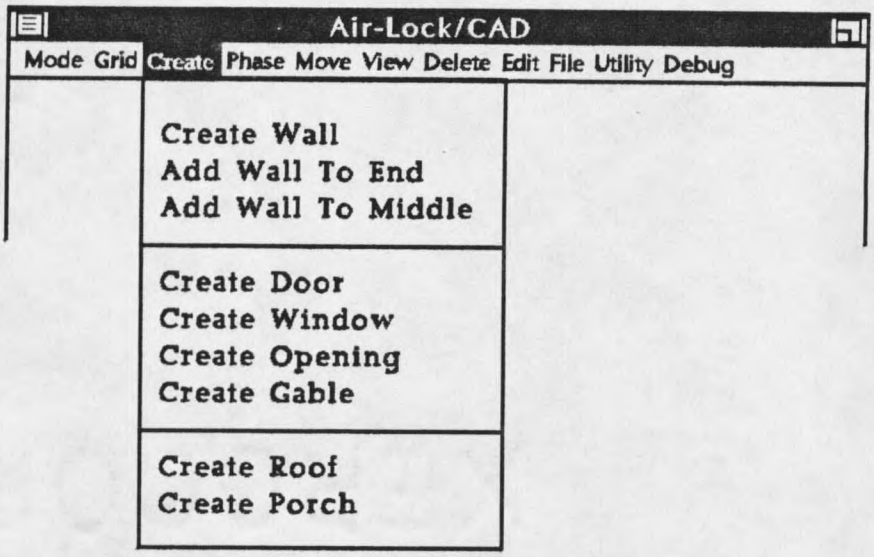

Figure 16. AL/CAD Create Menu.

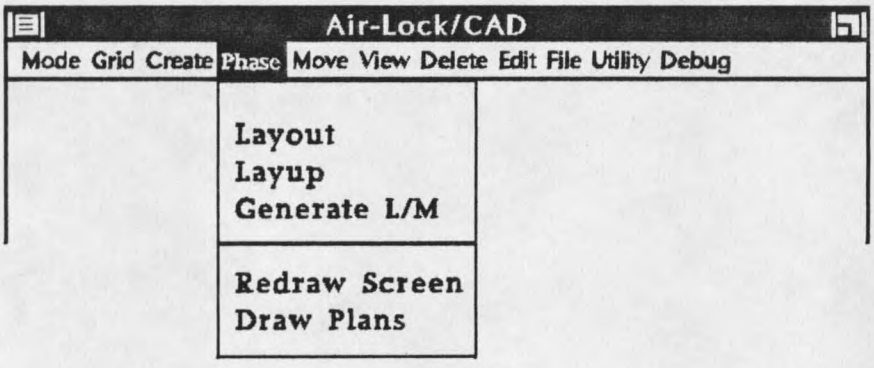

Figure 17. AL/CAD Phase Menu.

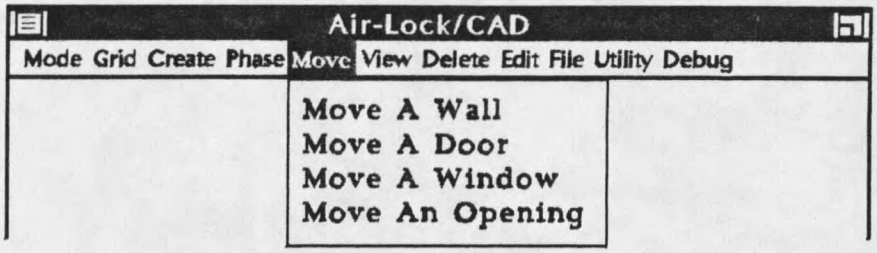

Figure 18. AL/CAD Move Menu.

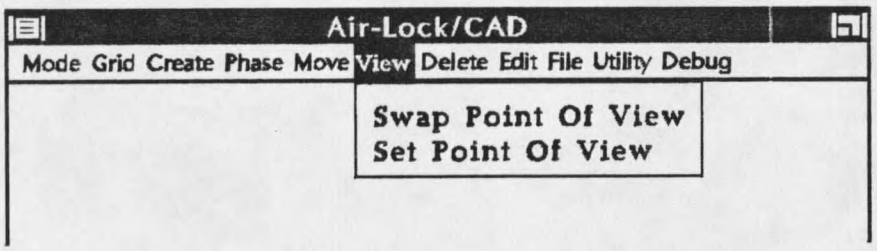

Figure 19. AL/CAD View Menu.

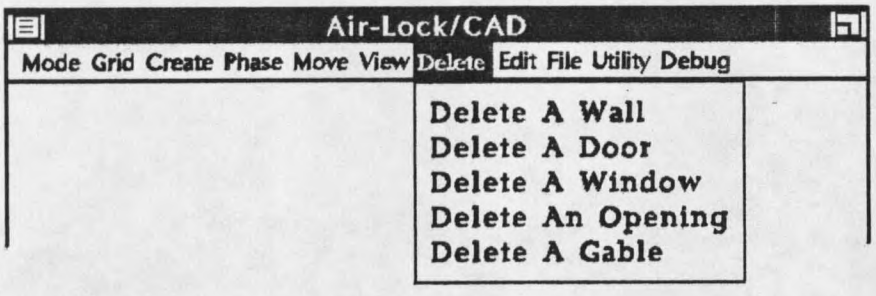

Figure 20. AL/CAD Delete Menu.

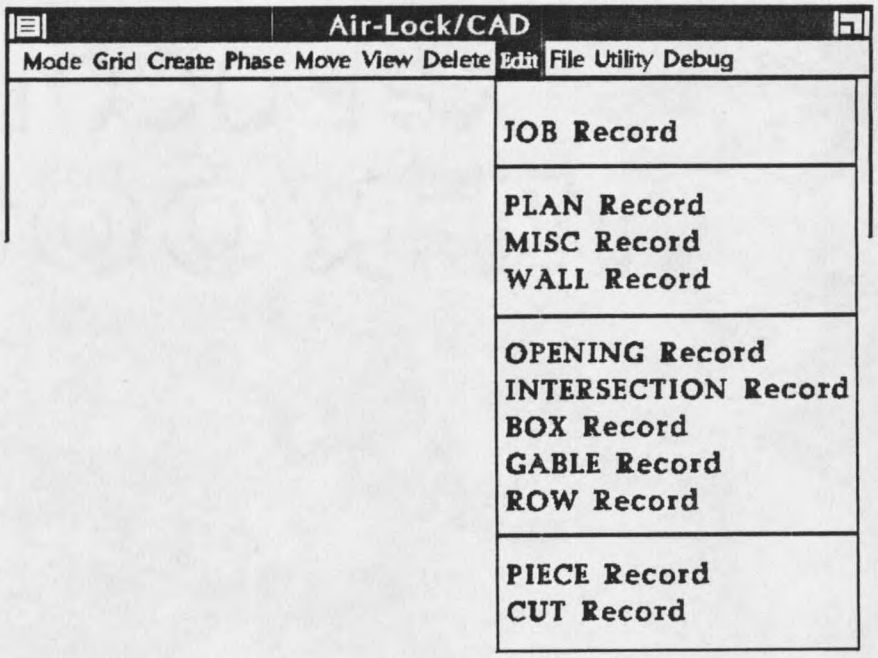

Figure 21. AL/CAD Edit Menu.

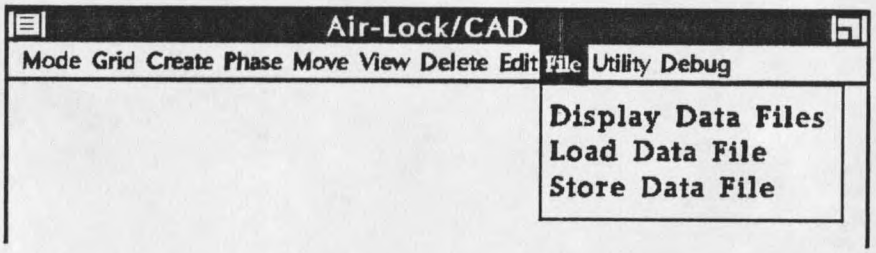

Figure 22. AL/CAD File Menu.

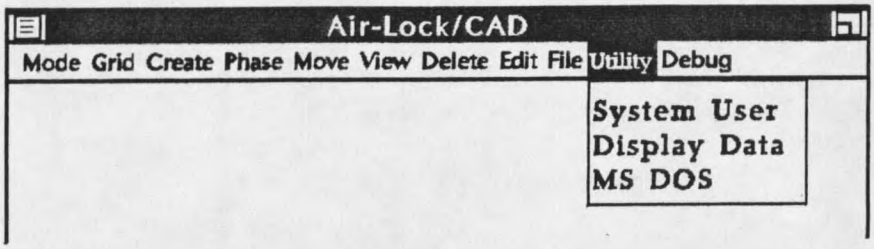

Figure 23. AL/CAD Utility Menu.

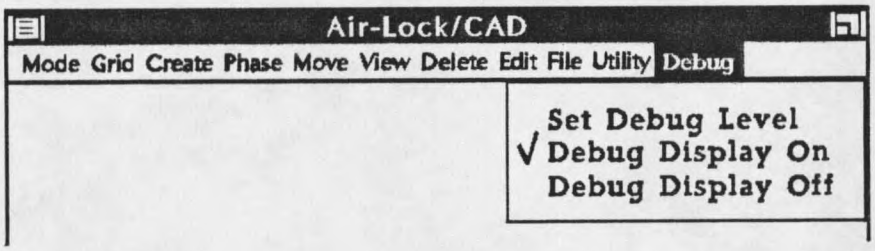

Figure 24. AL/CAD Debug Menu.

During the floor plan phase of AL/CAD, the user "draws" the various walls that make up a house. There are three methods that can be used: Create Wall, Add Wall To End, and Add Wall To Middle. Create Wall is used to draw a wall without reference to any existing wall. The user first selects the Create Wall option from the Create Menu. Figure 25 shows the user making this selection.

The user then selects the starting position by moving the mouse cursor to the desired location on the screen and clicking (pressing and releasing) the left mouse button. The system begins a wall at this point and begins to display the wall in "rubber band" mode. This means that as the user moves the mouse, the wall drawing expands and contracts to follow the cursor. At the upper left of the screen, another pop-up display shows the length of the

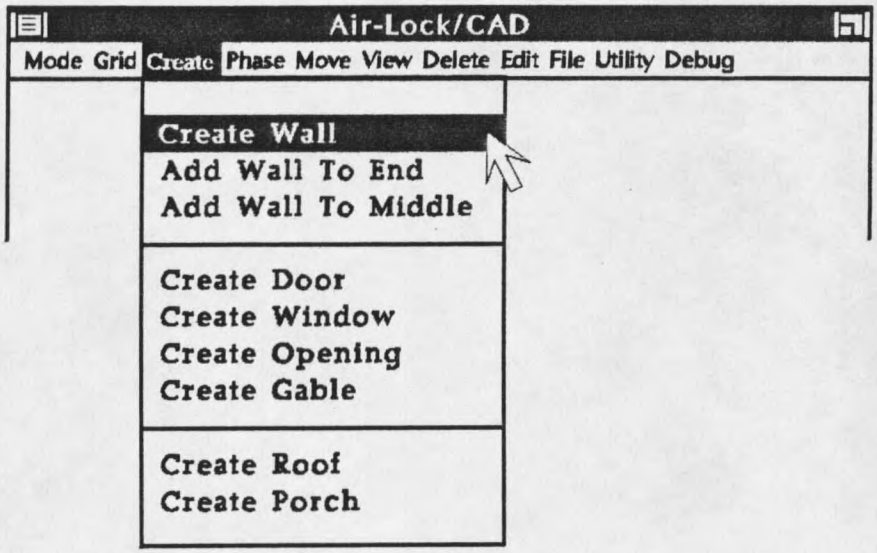

Figure 25. Selecting Create Wall Option.

wall in feet, inches, and eighths of inch units. As the user moves the mouse, and the wall expands and contracts, the current length of the wall is constantly displayed. Figure 26 shows the user creating a wall in this way. When the user has drawn the wall to the desired length, the left mouse button is again clicked to terminate the creation of the new wall.

After the wall has been drawn, the wall data is added to the data structure and the complete data structure is displayed. If a mistake is made, the user can select the Delete Wall option from the Delete menu to delete the wall just created. If a wall needs to be moved to a new location, the user can select the Move Wall option from the Move menu.

The Add Wall To End selection is used to add a new wall to the end of an existing wall. This is the normal location to add a new wall. After selecting Add Wall To End, the user positions the cursor near the end of an

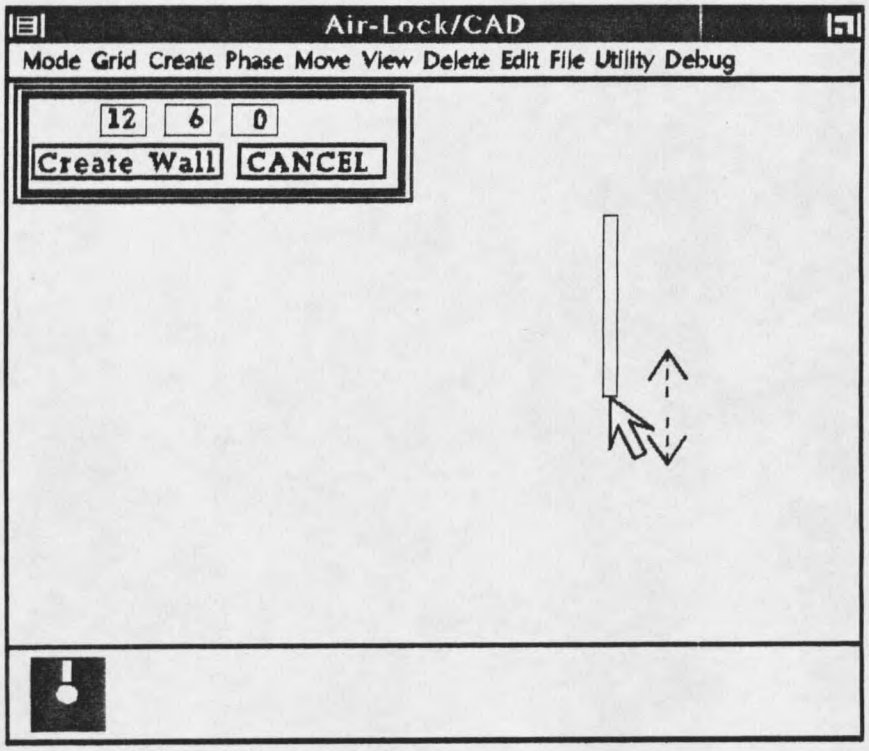

Figure 26. Creating A Vertical Wall.

existing wall. The user clicks the left mouse button and a gravity field algorithm draws the cursor to the nearest existing end of a wall. The new wall is then assumed to be at right angles to the existing wall. The user uses the rubber band display and the pop-up length display to help in selecting the length of the added wall.

The Add Wall To Middle selection is used to add a wall to the interior area of an existing wall. After moving the cursor near an existing wall, the user clicks the left mouse button. A gravity field algorithm then draws the cursor within the confines of the nearest existing wall. A position display

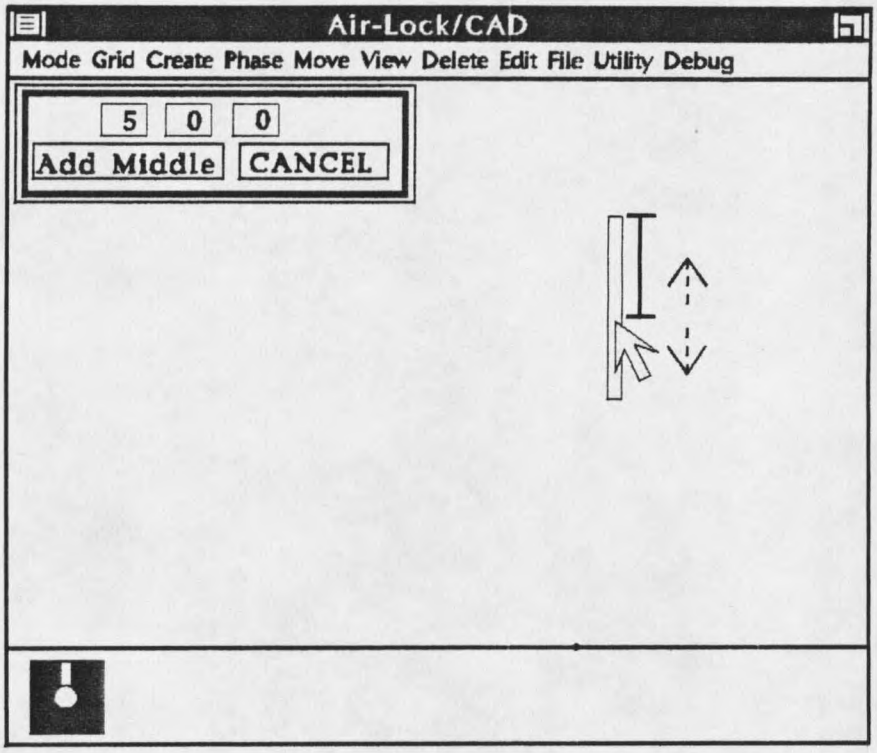

Figure 27. Positioning To A Middle Wall.

showing the length between the current location of the cursor and the end of the existing wall helps the user determine the proper location to add the wall (See Figure 27). The left mouse button is clicked again, and the new wall is designed in the same fashion as the create wall method described above. Figure 28 shows the added wall in rubber band mode. The user continues in this fashion until all the walls for the house have been drawn. The next step is to add the window and door openings to the building. Windows are chosen from a set of standard sizes made by various manufacturers. A specific manufacturer's set of windows is stored in a supporting disk file. The user

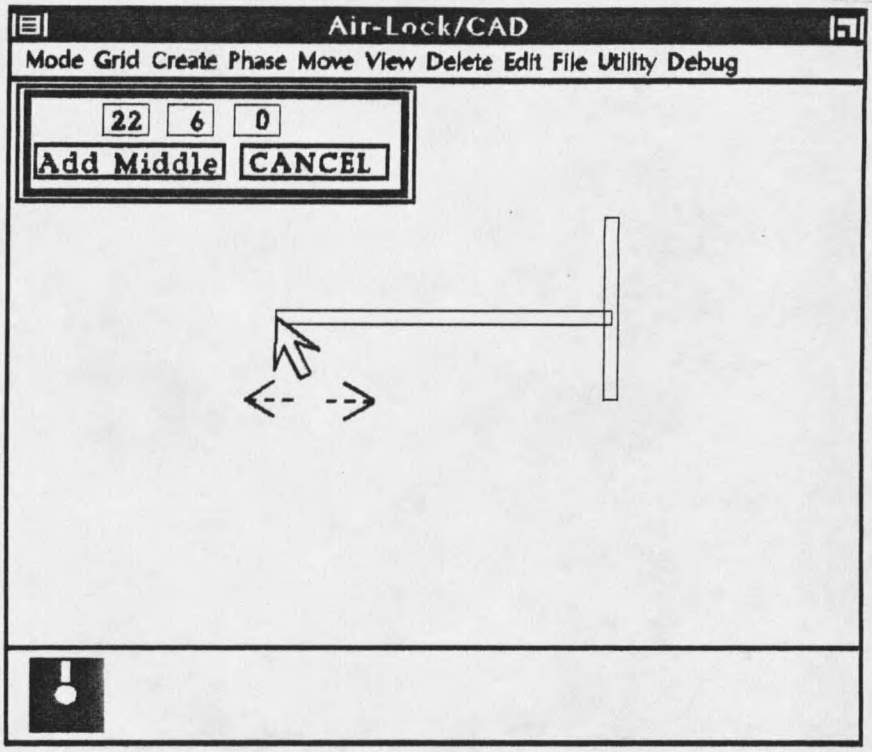

Figure 28. Adding The Middle Wall.

selects a manufacturer to get a display of possible window sizes. Upon selecting a specific window, the needed rough opening sizes are provided to the AL/CAD package. These sizes are used to draw the iconic representation of a window for the user. The window can be positioned near an appropriate wall. As the window nears a wall, a position display similar to the Add Wall To Middle system is provided. If the user moves the window away from one wall and near another wall, the position display follows. When the user is satisfied with the location of the window, the left mouse button is clicked. The window is drawn at the appropriate location in the floor plan and the window opening record is added to the AL/CAD data structure. Vertical positioning of the window is adjusted during the elevation mode of operation.

Figure 29 shows a window being added to an existing wall in plan view. A similar process is used to add any additional window or door openings.

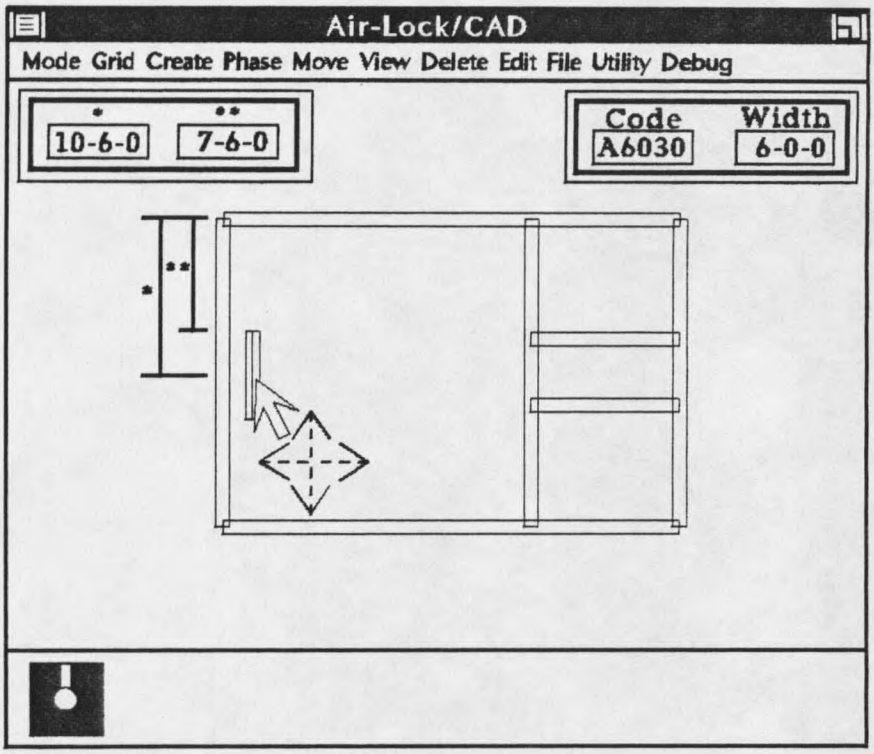

Figure 29. Adding A Window To A Wall.

At this point, the next phase is initiated by selecting the LAYOUT command. This command will cause all existing walls to be intersected and intersection records to be added to the data structure. The walls are extended to cover the normal corner extensions and preparations are made for the elevation mode. Elevation mode is selected to prepare for the final two phases of design. These are LAYUP and List of Material generation. Layup is the most complex phase of AL/CAD. To begin the layup phase the LAYUP

command is selected. The AL/CAD system begins by running what is called a "boxing" algorithm on each of the elevated walls. The boxing algorithm divides the wall into separate boxes of log material. These boxes are rectangular in shape and contain no openings. The boxes can be spliced separately allowing for intra-box constraints. A set of rules is used to splice each box. After splicing is complete, the box records, row records, piece records, and any necessary cut records are added to the data structure.

During the splicing algorithm, waste lists are maintained by AL/CAD to reduce the waste material generated. The preferred method is to choose a particular log length from the waste list. The List of Materials system is initiated by selecting the L/M command. The L/M algorithm gathers and sorts all the piece and cut records. Identical pieces are counted and a list of material is generated. The other commands provided by the user interface are used to enhance the design process.

AL/CAD Data Structure

The data structure used for AL/CAD needed to be extremely flexible in order to handle the wide range of house designs. Some houses have many walls with a few openings, and some houses have many openings with a few walls. In order to conserve high speed memory, a complex linked-list data structure is used. The general high level structure is shown in Figure 30.

The highest level structure is a simple linked list of job records. The job records contain data about the jobs currently being stored in the data structure. Each job record contains the pointers to three separate linked lists for Plan records, Miscellaneous records, and Wall records. Plan records are

included for future expansion into auxiliary processing of foundation plans, main floor plans, second floor plans, roof plans, plumbing plans, electrical and heating plans, and other plan view data. Miscellaneous records are used to store any data of a job specific nature but global in scope. The wall records contain the pertinent data for the design of walls within a specific job. Each wall record contains pointers to five linked lists. These lists contain opening records, intersection records, box records, gable records, and row records. Figure 31 shows an individual wall record with the head and tail pointers to an opening linked list. These pointers allow the system to add and delete records very quickly by positioning directly to the end of the list. Opening records contain data for door, window, fireplace and other openings. Intersection records contain data about every intersection between the current wall and other walls. Box records contain data about the boxes used by the splicing algorithm. Gable records contain data on any gables (normally just one) that exist on the current wall. Row records contain data on each row of logs in the current wall.

Each row record contains pointers to a linked list of piece records which contain data on each individual piece of log material. Each piece record contains pointers to a linked list of cut records which contain data on any special cuts within the current piece.

The purpose of this complex data structure is to provide an extremely adaptable structure. This means that the minimum high speed memory is consumed for each job. If a particular job has a lot of short walls with few rows and few openings or if it contains a few long walls with many rows and many openings, the structure will adapt. By storing all the data for a current

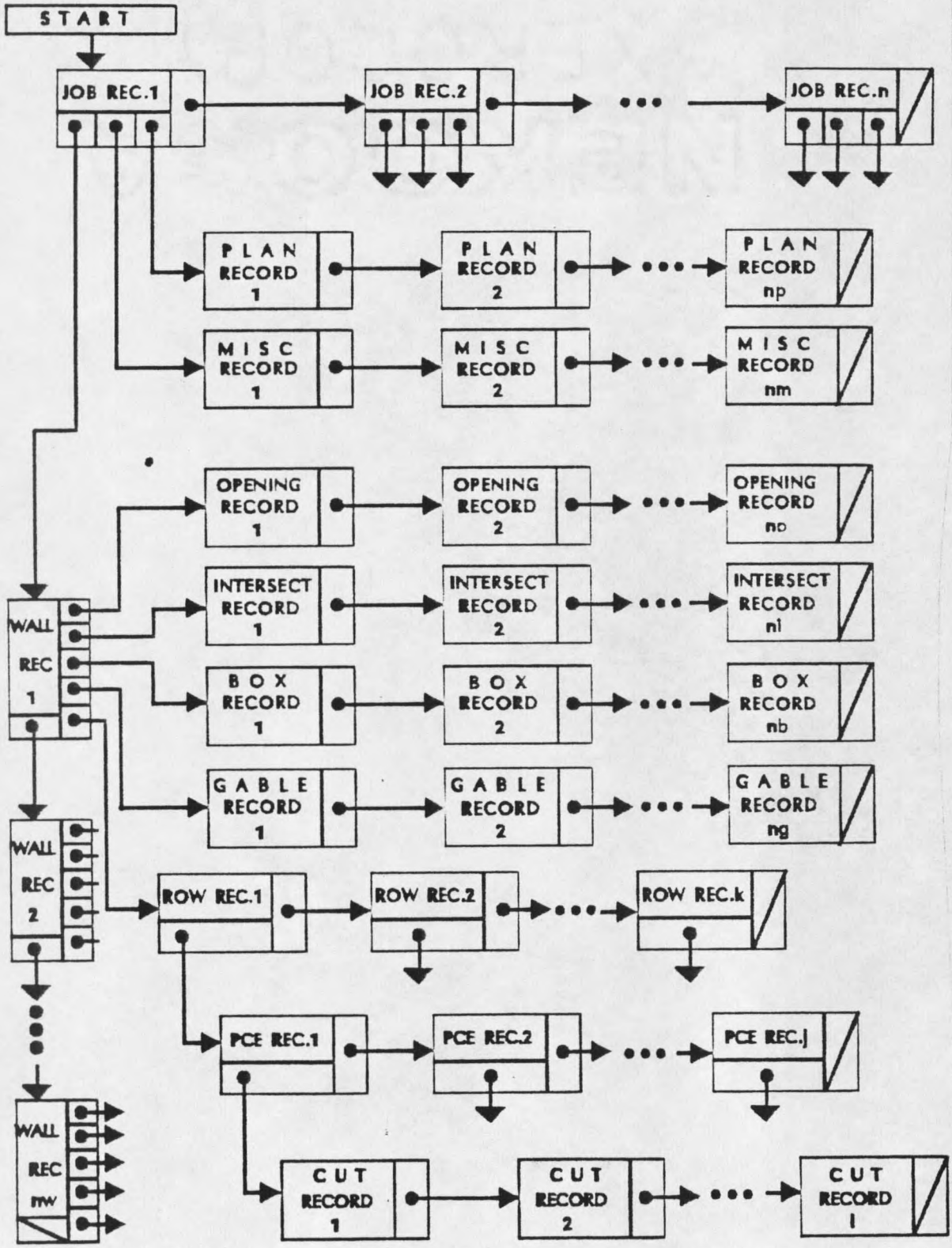

Figure 30. AL/CAD Data Structure.

job in high speed memory, the response time to the user during modifications can be minimized.

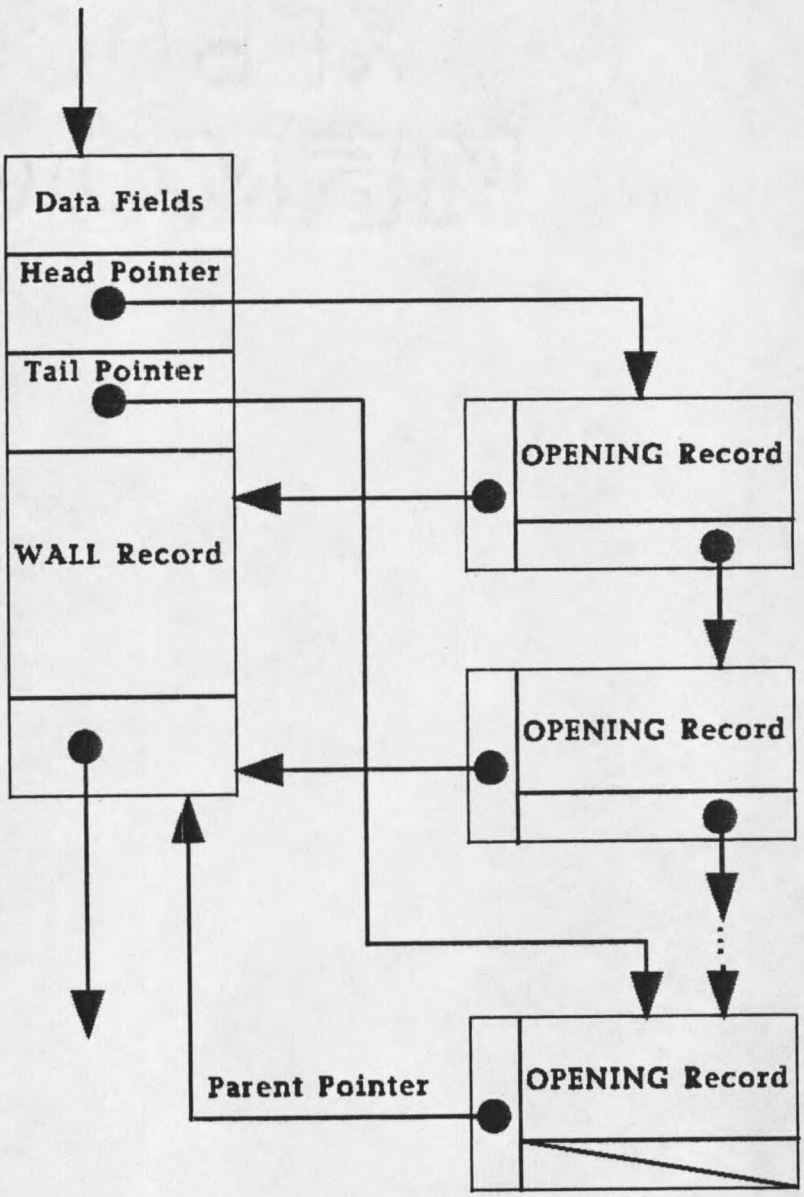

Figure 31. A Wall Record And Opening List.

At the highest level are job records. The internal fields of the job record are shown in Table 3. The algorithms that manipulate these data structure records are gathered into one place to allow easy modification without hazardous side effects.

Table 3. Job Record.

Name	Type	Description
id_number	integer	Job Record Identification Number
job_name	string	Job Name
job_date	3 integers	Date of the Current Job
next_job	pointer	Points to the next job record
wall_head	pointer	Points to the head of the wall list
wall_tail	pointer	Points to the tail of the wall list
plan_head	pointer	Points to the head of the plan list
plan_tail	pointer	Points to the tail of the plan list
misc_head	pointer	Points to the head of the misc. list
misc_tail	pointer	Points to the tail of the misc. list

A plan record is shown in Table 4 and a miscellaneous record is shown in Table 5. A wall record is shown in Table 6. The wall record is the most extensive record in the data structure. A wall record contains data concerning both the graphical location in screen coordinates and the actual location in logical coordinates for the separate features of the wall. The size (diameter) of the logs, the number of rows, and the height and width of the wall are stored. The wall record contains a pointer to its parent job record to facilitate gathering current job data while processing a wall record. The wall record contains the heads and tails of five other lists: openings, intersections, boxes, gables, and rows. An opening record is shown in Table 7. This record contains data about a specific log opening in the current wall. A log opening is used for a door, window, fireplace, pass-through, arch or for any other need in the house. The width and height of the opening and the distance from the

floor are provided. Opening data is used minimally in drawing floor plans. The opening data is used extensively in drawing elevations, and in generating row records, box records, and piece records.

Table 4. Plan Record.

Name	Type	Description
id_number	integer	Plan Record Identification Number
job_parent	pointer	Points to the Job parent record
next_plan	pointer	Points to the next Plan record

Table 5. Miscellaneous Record.

Name	Type	Description
id_number	integer	Misc Record Identification Number
job_parent	pointer	Points to the Job parent record
next_misc	pointer	Points to the next Misc record

An individual intersection record shown in Table 8 contains this wall's view of an intersection between this wall and another wall. The height in feet, inches, and eighths of inch as well as the top and bottom row numbers are stored.

A box record is shown in Table 9. This record contains information about the "box" generated by the boxing algorithm during the layup phase. The "box" is a rectangular piece of the wall that has no openings in it. The location of the box on this wall is also stored. A gable record is shown in Table 10.

Table 6. Wall Record.

Name	Type	Description
id_number	integer	Wall Record Identification Number
name	string	Wall name (e.g. "ab")
g_origin	2 integers	Origin of wall in screen coordinates
g_endwall	2 integers	End of wall in screen coordinates
direction	integer	Wall direction
g_scale	2 integers	Graphics scale
origin	3 integers	Origin of wall in logical coordinates
endwall	3 integers	End of wall in logical coordinates
width	integer	Width of wall in logical coordinates
height	integer	Height of wall in logical coordinates
numrow	integer	Number of rows in wall
logsize	integer	Diameter of logs
job_parent	pointer	Points to the Job parent record
next_wall	pointer	Points to the next Wall record
box_head	pointer	Points to the head of the box list
box_tail	pointer	Points to the tail of the box list
inter_head	pointer	Points to the head of the intersection list
inter_tail	pointer	Points to the tail of the intersection list
open_head	pointer	Points to the head of the opening list
open_tail	pointer	Points to the tail of the opening list
gable_head	pointer	Points to the head of the gable list
gable_tail	pointer	Points to the tail of the gable list
row_head	pointer	Points to the head of the row list
row_tail	pointer	Points to the tail of the row list

Table 7. Opening Record.

Name	Type	Description
id_number	integer	Opening Record Identification Number
g_origin	2 integers	Origin of opening in screen coordinates
g_width	integer	Width of opening in screen coordinates
g_height	integer	Height of opening in screen coordinates
g_to_floor	integer	Distance from opening to floor
origin	3 integers	Origin of opening in logical coordinates
width	integer	Width of opening in logical coordinates
height	integer	Height of opening in logical coordinates
to_floor	integer	Distance from opening to floor
wall_parent	pointer	Points to the Wall parent record
next_open	pointer	Points to the next Opening record

Table 8. Intersection Record.

Name	Type	Description
id_number	integer	Intersection Record Identification Number
g_center	2 integers	Location of center bottom of intersection
bot_row	integer	Bottom row number
top_row	integer	Top row number
height	integer	Height of intersection
center	integer	Location of center bottom
wall_parent	pointer	Points to the Wall parent record
next_inter	pointer	Points to the next Intersection record

Table 9. Box Record.

Name	Type	Description
id_number	integer	Box Record Identification Number
g_origin	2 integers	Origin of box in screen coordinates
g_width	integer	Width of box in screen coordinates
g_height	integer	Height of box in screen coordinates
width	integer	Width of box in logical coordinates
height	integer	Height of box in logical coordinates
box_status	integer	Box status
wall_parent	pointer	Points to the Wall parent record
next_box	pointer	Points to the next Box record

Table 10. Gable Record.

Name	Type	Description
id_number	integer	Gable Record Identification Number
g_origin	2 integers	Origin of gable in screen coordinates
g_width	integer	Width of gable in screen coordinates
g_height	integer	Height of gable in screen coordinates
origin	3 integers	Origin of gable in logical coordinates
pitch	integer	Pitch number (4 for 4-12, 3 for 3-12, etc.)
wall_parent	pointer	Points to the Wall parent record
next_gable	pointer	Points to the next Gable record

This record contains data for the gable on this wall if needed. In very special cases, there may be more than one gable associated with one wall. In

this case, there would be more than one gable record joined together into a linked list.

A row record is shown in Table 11. This record contains data that is specific to a particular row of pieces in this wall. The length of the row and the size of the row in graphical coordinates are stored. Row records contain the heads and tail for a linked list of piece records.

A piece record is shown in Table 12. This is where the actual data for a specific log piece is stored.

Table 11. Row Record.

Name	Type	Description
id_number	integer	Row Record Identification Number
g_width	integer	Width of row in screen coordinates
g_height	integer	Height of row in screen coordinates
g_origin	2 integers	Origin of row in screen coordinates
width	integer	Width of row in logical coordinates
height	integer	Height of row in logical coordinates
origin	3 integers	Origin of row in logical coordinates
wall_parent	pointer	Points to Wall parent record
next_row	pointer	Points to next Row record
piece_head	pointer	Points to the head of the Piece list
piece_tail	pointer	Points to the tail of the Piece list

Table 12. Piece Record.

Name	Type	Description
id_number	integer	Piece Record Identification Number
g_origin	2 integers	Origin of piece in screen coordinates
g_width	integer	Width of piece in screen coordinates
g_height	integer	Height of piece in screen coordinates
origin	3 integers	Origin of piece in logical coordinates
length	integer	Length of piece in logical coordinates
aux_length	integer	Length of auxiliary cut
style	string	Style code (e.g. "c" for corner)
row_parent	pointer	Points to the Row parent record
next_piece	pointer	Points to the next piece record
cut_head	pointer	Points to the head of the cut list
cut_tail	pointer	Points to the tail of the cut list

The list of materials portion of AL/CAD gathers the piece records and counts the number of pieces that are identical. The style of the piece and any normal cut lengths are stored in the piece record. If a piece contains more than one cut, these are stored in the linked list of cut records attached to this piece. These special cuts occur in almost every job but the number of these special logs is usually quite small. A cut record is shown in Table 13. This record contains data for a special cut in the current log piece.

Table 13. Cut Record.

Name	Type	Description
id_number	integer	Cut Record Identification Number
g_length	integer	Length of cut in screen coordinates
length	integer	Length of cut in logical coordinates
c_style	string	Style code for cut
piece_parent	pointer	Points to the Piece parent record
next_cut	pointer	Points to the next cut record

The data structure described above is complex but works very well. The results confirm the design goal of fast data access to a wide ranging combination of walls, openings, and rows. To manipulate the data structure, a set of algorithms were developed. For each type of record described above, the set of algorithms shown in Table 14 was written.

Table 14. AL/CAD Data Structure Manipulation Algorithms.

Algorithm	Description
MakeRecord	Create the record.
InitRecord	Initialize the fields of the record.
DisplayRecord	Display the record on the screen.
FileInRecord	Read the data for a record from a file.
FileOutRecord	Write a copy of the data from a record to a file.
AddRecord	Add the record to the data structure.
DeleteRecord	Delete the record from the data structure.
ChangeRecord	Edit a record in the data structure.

The MakeRecord algorithm requests a correctly sized memory segment from the memory manager. The InitRecord algorithm initializes all the integer data fields to zero. The head, tail, and parent pointer fields are initialized to NULL. Any string fields are initialized to blanks and then the end of string indicator (a null) is placed at the first location. The DisplayRecord algorithm generates a dialogue window type of display. (For information on the window system see the Microsoft Windows Development Tool Kit Manual.) The various fields in the record are identified and their current values are displayed. The FileInRecord and FileOutRecord algorithms permit the system to save and restore a data file in a text format that contains the data structure. This allows for the backup of a current data structure or for saving a structure to work on at a later time. The AddRecord algorithm adds a record to the tail of the correct linked list. If the list is empty, it starts a new linked list. The DeleteRecord algorithm deletes a record from the data structure. The ChangeRecord algorithm generates another dialogue window allowing the operator to change any fields within a record.

AL/CAD Boxing Algorithm

Each wall when shown in elevation view will contain from zero to any number of openings and two or more intersections with other walls. The boxing algorithm separates the wall into one or more boxes. A box is a part of the wall that is solid wood and contains either no intersections or a part of an intersection. If the wall contained no openings, the box would be the entire wall. If the wall contained one window, the algorithm would find four boxes. Figure 32 shows such a wall separated into the four boxes by the

boxing algorithm. The data for these boxes is stored in the box records of the data structure.

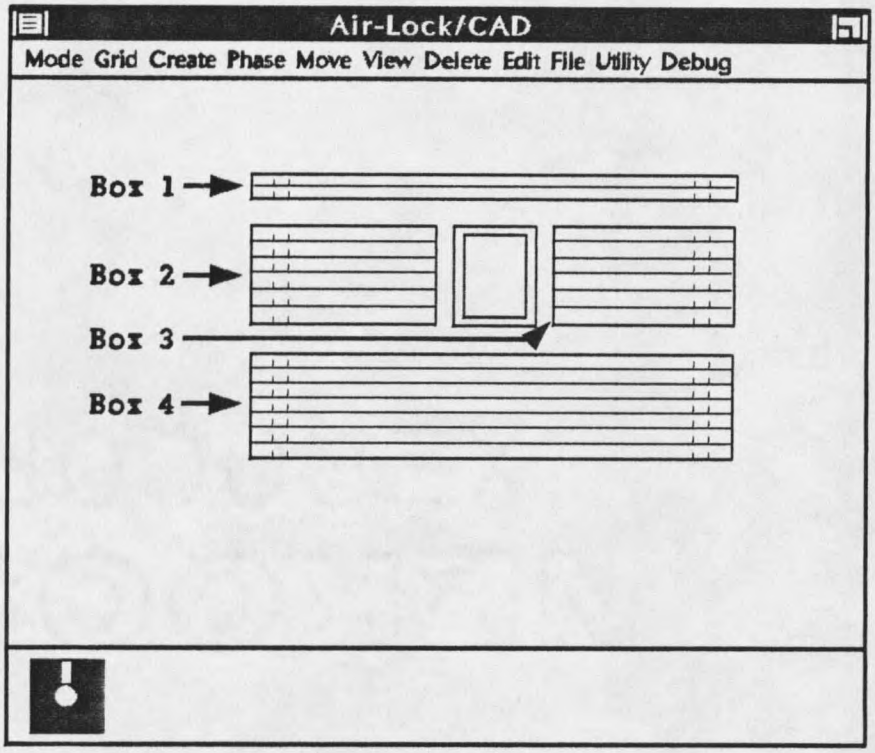

Figure 32. Four Boxes In An Elevation.

The boxing algorithm starts at the lower left of a wall and moves to the right until reaching either an end of wall or the left edge of an opening. This defines the horizontal dimension of the box. The horizontal dimension would be along the X or Y axis depending on the direction of the wall in plan view. Horizontal walls in plan view are placed along the X axis in elevation view. Vertical walls in plan view are placed along the Y axis in elevation view. Two lines are projected up (in the positive Z axis) until they reach the top of

the wall. The data on all openings contained completely or partially between these two lines are gathered into a list and sorted to find the lowest (in the Z axis) bottom of opening. This locates the top of the box. The values calculated are then stored in a new box record.

The algorithm then checks to see if the right hand edge of this box is an end of a wall or the left edge of an opening. If it is the left edge of an opening, the next box origin is started at the bottom right of that opening, and the operation described above is repeated. This process continues until the right edge of the wall is reached. The algorithm then moves to the top left of the first box on this wall and repeats its operation. This left to right scan is repeated until all the boxes are found. When the top of one or more boxes completely covers the length of the wall at its top, the algorithm terminates.

The splicing algorithm that is described in the next section depends on the boxing algorithm to divide the entire elevation into small boxes, an example of a divide and conquer technique.

AL/CAD Splicing Algorithm

There are numerous types of logs produced by the Air-Lock manufacturing process. Some of these are standard logs, corner logs, half cope logs, and standard crossover logs. Standard logs are hollow completely through their length and have a vertical groove cut in each end to facilitate splicing. Corner logs are solid on one end and have a cope cut near the solid end. This cope cut is a half round notch cut to allow the logs to interlock at the corner intersection between two walls. Half cope logs are similar to a corner

log that was cut off exactly half way through the cope. This cut allows for a blind intersection between one wall and another. Standard crossover logs are similar to standard logs but have a cope cut somewhere along their length to allow the corner log of an intersecting wall to pass over. These are only four of the twelve different styles produced by the Air-Lock factory. The AL/CAD splicing algorithm calculates the style and length of every log piece needed to fill an individual box.

Splices weaken the structure of a log wall. The tongue cut longitudinally in the bottom of a log and the groove cut longitudinally in the top of a log make up for the loss of structure by spanning the splice in a row both above and below the splice. Also, the interlocking corners provide an extremely strong construction. However, in splicing a wall various rules must be followed. For instance, no splices may be made over an opening. No splice may be made in one row that is directly beneath a splice in the row above it. No splice may be made within an intersection.

The box determined by the boxing algorithm is passed to the splicing algorithm. Since there can be an interface between two boxes, constraints concerning splices must be passed from one box to another. The splicing algorithm maintains a list of these constraints until all the boxes from a wall have been processed. Data on waste is also maintained. Waste is defined as that portion of an 8-3-0 raw material log left over after a piece is removed. For example, if the splicing algorithm needs a piece that is 6-3-0 in length, then the waste would be 2-0-0. The raw material processed by the final cutting machines in the Air-Lock factory is very expensive. Therefore, it is desired to reduce the final amount of waste.

As the splicing algorithm processes boxes, it uses a set of strategy tables, a set of rules, and the waste data to make the choices required. For example, if the box is less than 8-3-0 long and if there are no intersections in the wall and if the left and right edges of the box are constrained by openings, then standard logs are defined of a length equal to the box width. However, if the box is constrained on the left by the beginning of a wall and constrained on the right by an opening, and if there is only one intersection in the box, and if the intersection is 1-0-0 from the left edge of the box, and if the box is less than 8-3-0 in width, then simple corner logs are defined of a length equal to the box width.

For boxes longer than 8-3-0 the process is complicated by splices. A rule stated above requires that no splices be in adjoining rows at the same location. This produces what is sometimes called the zipper effect. Two sets of rows are defined that interleave one another. Figure 33 shows a box of this type after the splicing algorithm is finished.

The strategy tables and waste data are then used to make choices as to splice location. The strategy tables produce a set of rules to use for splicing. They contain various length values for different box lengths and box constraints. Each strategy has associated with it a priority value. All possible strategies are gathered and sorted by priority and the highest priority is selected. When equal priorities are found, the contention is resolved using the waste data.

The splicing algorithm continues with all the boxes of one wall until that wall is finished. It then continues with the boxes of all other walls until all the boxes in a house are finished. The splicing algorithm will always be under

further development since the constant need to reduce waste will prompt continual adjustments in the rules, strategy tables, and waste algorithms.

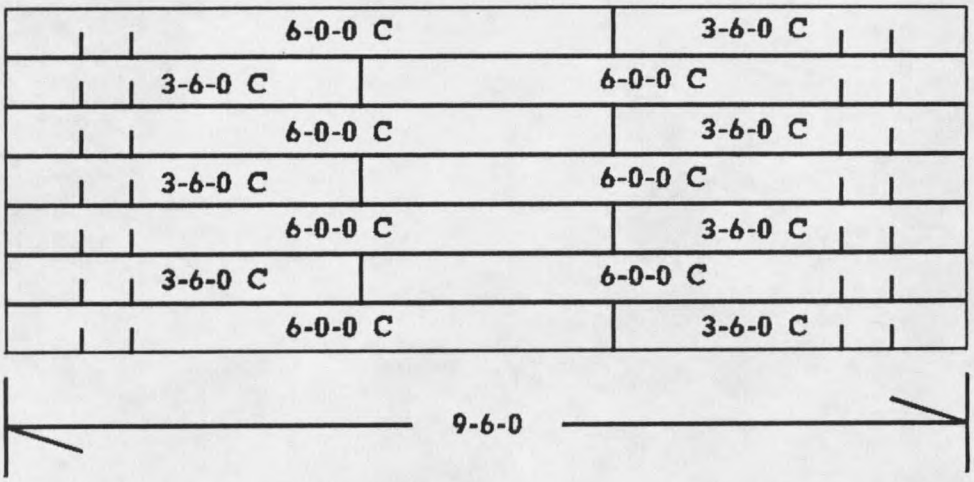

Figure 33. Splicing One Box.

CHAPTER 4

CONCLUSIONS

The major goal of this thesis project was to determine the possibility of using a computer to design Air-Lock log houses. This goal was achieved. A design package was developed which demonstrated that the interactive nature of a windowing system greatly enhances the friendliness of a computer program. The mouse pointing device allowed even an uninitiated user to quickly master the development process. The sections that follow describe individual conclusions about Windows, the Windows Development Tool Kit, and the AL/CAD development system.

Evaluation of Microsoft Windows

Microsoft Windows worked exceedingly well during my project. Windows is essentially a multi-tasking operating system with a highly graphics user interface. The screen representations were clear and easy to use. The response time was very fast and the mouse tracking was sure and positive. I was able to copy files between sub directories and manipulate run images with ease. The documentation supplied was clear and complete. The sample packages were easy to run and demonstrated the main features of Windows well. This package would be a very reasonably priced addition to any IBM-PC or clone.

Evaluation of Windows Development Tool Kit

The windows tool kit used was MicroSoft Windows Development Tool Kit Version 1.04. I found this package worked well as soon as I was able to learn it. After over 24 years of experience with computer hardware and software, I am able to understand most software or hardware documentation and make use of new products with a minimum of reference to the documentation. This tool kit package was different. It took over one hundred and twenty hours of study before I was able to begin to utilize this package effectively. If I had not been committed so strongly to IBM compatibility, I would have not used this package.

There were three complete manuals of documentation and six complete sample applications delivered with the tool kit. I was able to compile and run the examples without any problems. However, in trying to develop my own application, I began to run into many problems. The documentation was poorly organized and it was extremely difficult to find where the item you needed was located. There was no good tutorial section that helped you start from scratch. Limitations with using C began to appear. For example, you cannot use the C library function fscanf, the main input function, with MS Windows.

The amount of source code needed to do even a simple application window was extensive. I had problems with the various memory models in C needed by the 80286 cpu system. I was never able to get the large memory model version of AL/CAD running and have only been successful because the AL/CAD package currently fits in a small memory model.

Since MS Windows has control of the keyboard and screen, debugging was difficult. A separate terminal was connected to the auxiliary port on the

computer and a simple symbolic debugger was used. However, I was never able to arrange the configuration of the system so that I could enter commands from this separate terminal. The only data I had during the many initial crashes were the names of the functions that were on the run time stack at the time of the crash. Error messages produced were a four digit hexadecimal code. This code was then interpreted by reference to one of the manuals. However, a few of the codes presented on the terminal were not listed in the manual and I had to use other means to determine my error.

On the positive side, the documentation was mostly correct. I found that the functions worked as described and parameters were correctly listed. The large problem was finding the correct section of the manual. If the documentation were better organized, and a tutorial section were added, I could recommend this package. It did save me time at the end of the project when I was redesigning the screen display. However, the time saved has not yet made up for the time lost in learning the package.

Evaluation of AL/CAD

The prototype ADS system proved that a computer program could be written to do log house design. This was the major objective of the thesis project. However, the prototype ADS user interface was not very easy to use. The Tektronix 4006-1 storage screen terminal was dim and hard to read in normal office light. The user commands were understood well by a person familiar with computers, but the average office worker did not have the necessary background to utilize the package effectively. Since it was so hard to use the commands and so hard to read the screen, the office draftsman never spent enough time to adequately learn the package. The boxing

algorithm failed approximately ten per-cent of the time leaving large portions of the house design unspliced. This would leave many needed pieces of log material out of the final list of materials.

The final AL/CAD implementation alleviated all of the problems with the prototype. The screen was now bright and easy to read. The pop-up menus and the mouse made command selection very easy for the office staff. There is still a need to understand the Air-Lock log manufacturing process, but anyone who does know the Air-Lock system can learn to use the AL/CAD system in a few days.

The major disadvantage of the AL/CAD package is in handling the "special cases" that always occur in the design of a real log house. The AL/CAD package does not allow for easy modifications of specific splices. This feature is planned for a future release of the package.

The list of material generated by AL/CAD agreed favorably with what a human would generate. The waste reduction results were adequate, but further development in this area is indicated. AL/CAD proved the applicability of using computer technology in designing log homes.

SELECTED BIBLIOGRAPHY

Programming Reference Manuals

"CodeView and C Language Reference Manual." MICROSOFT C COMPILER. Microsoft Corporation. 1986.

"MS-DOS User's Reference." HP9000 SERIES300. Hewlett Packard Corporation. 1987.

"Programmer's Reference." MICROSOFT WINDOWS SOFTWARE DEVELOPMENT KIT. Microsoft Corporation. 1986.

"Quick Reference/Programming Guide/Application Style Guide." MICROSOFT WINDOWS SOFTWARE DEVELOPMENT KIT. Microsoft Corporation. 1986.

"Run-Time Library Reference Manual." MICROSOFT C COMPILER. Microsoft Corporation. 1986.

"Update/Programmer's Utility Guide." MICROSOFT WINDOWS SOFTWARE DEVELOPMENT KIT. Microsoft Corporation. 1986.

"User's Guide." MICROSOFT MACRO ASSEMBLER. Microsoft Corporation. 1986.

"User's Guide." MICROSOFT MOUSE. Microsoft Corporation. 1986.

"User's Guide." MICROSOFT WINDOWS, DESKTOP APPLICATIONS, WRITE, AND PAINT. Microsoft Corporation. 1987.

"Users Manual." MICROSOFT C COMPILER. Microsoft Corporation. 1986.

"Using the DOS Coprocessor." HP9000 SERIES300. Hewlett Packard Corporation. 1987.

Birns, Peter; Brown, Patrick; and Muster, John C. "Unix For People." Prentice-Hall, Inc., 1985.

Christian, Kaare. "The Unix Operating System." John Wiley & Sons, 1983.

Geary, Michael. "Microsoft Windows 2.0: Major Enhancements Offer More Control." MICROSOFT SYSTEMS JOURNAL. (July 1987) 19-36.

Houghton, Raymond C. Jr. "Online Help Systems: A Conspectus", COMMUNICATIONS OF THE ACM. (February 1984) 126-133.

Jamsa, Kris. "DOS The Complete Reference." Osborne McGraw-Hill. 1987.

Kernighan, Brian W. and Ritchie, Dennis M. "The C Programming Language." Prentice-Hall, Inc. 1978.

Kelley, Al. and Pohl, Ira. "A Book On C." Benjamin/Cummings Publishing Company, Inc. 1984.

Olsen, Dan R. Jr. "Pushdown Automata for User Interface Management." ACM TRANSACTIONS ON GRAPHICS. (July 1984) 177-203.

Schildt, Herbert. "Artificial Intelligence Using C." Osborne McGraw-Hill. 1987.

Waite, Mitchell; Martin, Donald; and Prata, Stephen. "Unix Primer Plus." Howard W. Sams & Co., Inc., 1983.

Wilson, Lynwood. "Rule-Based Programming in C." AI EXPERT 2 (August 1987) 15-21.

MONTANA STATE UNIVERSITY LIBRARIES
3 1762 10021532 4

AMERICAN BINDERIES