SOFTWARE MANUAL

AlphaMENU

USER'S MANUAL

DSS-10044-00

REV. A00

alpha micro

ALPHA MICROSYSTEMS
17881 Sky Park North
Irvine, CA. 92714

SOFTWARE MANUAL

AlphaMENU

USER'S MANUAL

DSS-10044-00

REV. A00

alpha micro

FIRST EDITION

April 1, 1983

REVISIONS INCORPORATED	
REVISION	DATE

This manual applies to AMOS/L versions 1.0B and later.

ALPHA MICROSYSTEMS
17881 Sky Park North
Irvine, CA. 92714

Table of Contents

LIST OF FIGURES

A QUICK REFERENCE GUIDE

TO USING AlphaMENU

CALLING THE MENU

To bring a menu to the terminal screen, log into the account containing the menu, and enter the name of the menu you want to use. For example:

.SHELL DOC (RET)

if your menu is named DOC.CMN. The sample menu supplied with AlphaMENU is AMOSL.CMN. To invoke AMOSL.CMN, enter:

.SHELL (RET)

AMOSL.CMN is the default, so simply typing SHELL will bring you this menu.

WHAT YOU SEE

You will see six "boxes," some or all containing numbered choices. You may (depending on what kind of terminal you have) see an area highlighted in reverse-video display-- if so, this is called the "selection marker." You will see a question displayed at the bottom of the screen, followed by a number.

MAKING A SELECTION

There are two different ways that you can select a menu option. The first way is to enter the number of the selection you want. The six major options are numbers 10 to 60. Type the number of the selection you wish, and press RETURN, and the next level of menu appears, or that function is performed.

The second is to use the cursor control (arrow) keys to move the selection marker onto the function you wish to select. Once the selection marker is on that selection, press RETURN, and the next level of menu appears, or that function is performed.

MOVING THE SELECTION MARKER

Use the cursor control keys (the ones with arrows on them) to move the position of the selection marker. If your terminal doesn't have such keys, you can move the selection marker by pressing and holding down the control key (labeled CTRL on some terminals) while pressing the K (up), J (down), H (left) or L (right) keys.

The menus are set up so that the selection marker "wraps" around the screen. If you are at any edge of the menu screen, and you press a key that would move the selection marker in a direction that is off the screen, you will see the selection marker appear on the opposite side of the menu.

This features makes it easy to reach any place on the screen with just a few keystrokes. With a little imagination, you can find the quickest way to position the selection marker to any place on the menu.

GETTING HELP

To get a display of helpful information, place the selection marker on the function you want information on (or type the number of it), and type a question mark.

PERFORMING A FUNCTION

To perform a function, you make selections until you are at the level where the function you want to perform is displayed. Enter the number of the selection, or place the selection marker on the function, and press RETURN.

EXITING A MENU DISPLAY

To go back to the previous menu, press the escape key (often labeled ESC or ALT MODE) or hold down the control key (often labeled CTRL) and press "C". You will then see the next highest level of menu.

CHAPTER 1

PREFACE

1.1 THE CONTENTS OF THIS BOOK

CHAPTER 2 - Introduction

We begin with an introduction to AlphaMENU. In this chapter we explain some of the features and uses of the AlphaMENU system, and give you an example of what the AlphaMENU display looks like. We also discuss the files that make up AlphaMENU.

CHAPTER 3 - Installing AlphaMENU

Chapter 3 instructs you in the software installation instructions for AlphaMENU. Following that are instructions on how to invoke and use the menu, including how to perform a function and how to get help.

CHAPTER 4 - Defining a Menu

This chapter contains information on how to define a menu. It shows you how to create a menu file, how to define the levels inside a menu, and how to define the DO file elements that make the menu perform its functions.

CHAPTER 5 - The Help File

Chapter 5 contains complete instructions on how to create and invoke AlphaMENU help files.

CHAPTER 6 - Query file and DO file Elements

This chapter discusses the actual elements that make the menu perform the selected functions. All the special AlphaMENU commands are discussed, along with examples of use.

Appendix A is a brief reference sheet containing all the special AlphaMENU commands. Appendix B contains a list of the screen-handling TCRT calls.

1.2 READER'S COMMENTS FORM

Please note the Reader's Comments Form at the back of this manual. Your suggestions are important to us, and we will use them to expand and enhance later versions of this manual.

1.3 GRAPHICS CONVENTIONS USED IN THIS BOOK

To make our examples clear and easy to use, this manual conforms to the standard Alpha Micro documentation graphics conventions:

PPN — A Project-programmer number. This number identifies a user account (e.g., [100,2]). We also represent an account number as [P,pn].

Filespec — A file specification. Such a specification identifies a file. It usually has these elements:

Devn:Filename.Extension[P,pn]

where Devn: specifies the logical device the file resides on, Filename.Extension gives the name and file extension of the file, and [P,pn] specifies the disk account in which the file resides.

___ — Underlined characters indicate those characters that AMOS/L prints on your terminal display. For example, throughout this document you see an underlined dot, .., which indicates the prompt symbol that the operating system displays on your terminal when you are at monitor command level.

(RET) — Carriage return symbol. The RET symbol marks the place in your keyboard entry to press the RETURN key. For example, ".LOGOFF (RET) " tells you "After a monitor prompt, type LOGOFF and press RETURN."

^ — Indicates a Control-character. As you enter characters from the keyboard directly to the monitor, the system usually displays these characters on your terminal. If you type a Control-C, you see a ^C on your terminal display. (Refer to the AMOS User's Guide, DWM-00100-35, for more information on Control-characters.)

CHAPTER 2

INTRODUCTION

This manual is aimed at the experienced user of an Alpha Micro computer system who wishes to make use of the AlphaMENU system to create his or her own screen-oriented menus. The discussions that follow assume that you are quite familiar with DO files-- if not, please see the chapter "Command and DO files" in the AMOS User's Guide, DWM-00100-35.

One of the most impressive characteristics of an Alpha Micro computer system is the huge number of commands and programs available. Add to this the fact that the Alpha Micro computer system is designed to make adding new user defined commands very simple, and you can see that the number of functions you can perform on an Alpha Micro computer is potentially vast. Although such a large number of commands and functions confirms the power and versatility of the Alpha Micro computer, it can also prove somewhat bewildering to the novice Alpha Micro computer user.

A user-friendly way of providing access to a defined set of commands is the screen-oriented menu. The AlphaMENU utility allows you to define such a screen-oriented menu of commands and functions. This attractive screen display lists a number of options from which the user may choose by using the cursor control keys to move to the appropriate option and pressing the RETURN key. (The user may also enter the number of the option as it appears on the screen, and press RETURN.) At this point, depending on how you have defined the menu options, the user may see another menu display from which to choose an option, or the user may reach the level at which the chosen function is performed. (See Figure 2-1, for a sample AlphaMENU screen menu.)

At any point, the user may enter a ? followed by a RETURN to get help. The user then sees the contents of a text Help file you have defined as being appropriate to that location in the menu.

2.1 FEATURES

1. Makes using the system easier for inexperienced users.

2. Allows you to define what system functions and commands the users on your system will be able to perform (thus allowing you to prohibit users from performing certain functions).

3. Provides an attractive and simple to use front end to your applications software that ties the separate programs into a unified structure.

4. Automatically makes use of special features of the terminal, such as reverse video, reduced intensity, and graphics characters. (See Section 3.1.2, "Terminal Features," for information on what terminal features AlphaMENU can make use of.) Also performs all cursor positioning and handling automatically.

5. Allows an unlimited number of menu levels.

6. Allows your main menu to chain to an unlimited number of separate menus.

7. Allows you to designate certain menu options as "special" (for example, not installed or reserved for System Operator use) by having those options displayed in reduced intensity.

8. Allows the user to interact with the selected options by asking the user for specifications and information. AlphaMENU does this by building a type of "DO file" in memory into which parameters supplied by the user are inserted.

9. Allows you to lock the user into the menu so that they can only perform those functions that are applicable.

Defining an AlphaMENU menu is a simple process of creating a text file using the AlphaMENU definition language, and then using the AlphaMENU compiler to build a menu from the file. You define the structure of the menu, define what information will be displayed to the user, and what questions will be asked of the user. You do not have to worry about how the menu will be displayed on the screen-- AlphaMENU automatically takes care of arranging the contents of the menu on the screen for you.

2.2 THE MENU STRUCTURE

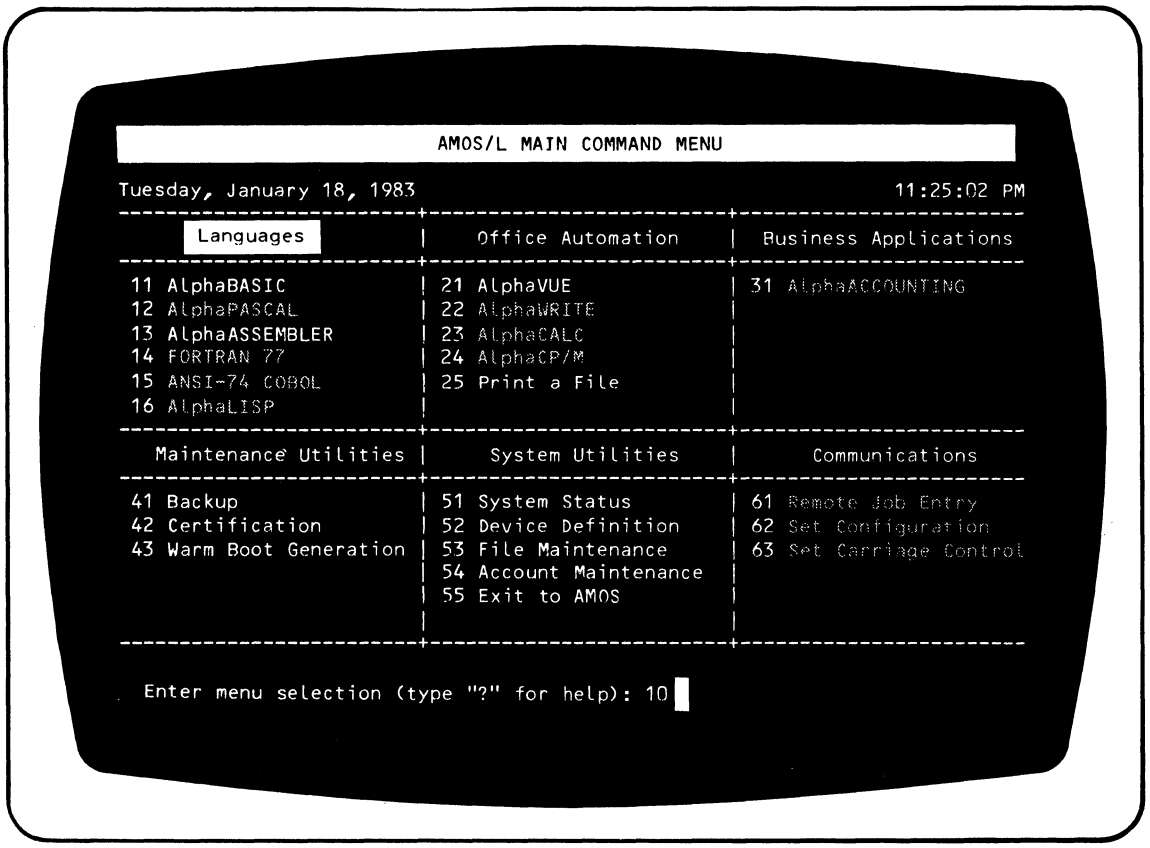

Figure 2-1: Sample Menu Display

NOTE: You will notice that a "10" is already displayed after "Enter menu selection" at the bottom of the screen. If your terminal supports reverse-video display, the title of the menu will be highlighted using reverse-video display. The first major option ("Languages") will also be highlighted using reverse-video. Throughout this manual, we will call this second highlighted area the "Selection Marker."

If your terminal does not support reverse-video, you will not be able to see the selection marker. So, we have provided another method of making the selection-- you can enter the number of the selection you wish and then press RETURN.

In the example above, the number 10 corresponds to the "Languages" section. If you enter another number, or if you use the cursor movement controls to move the selection marker, the number will change.

The six main divisions in the menu above are "Languages," "Office Automation," "Business Applications," "Maintenance Utilities," "System Utilities," and "Communications." Each main selection can have up to six sub-sections (the second level of the menu). When you select a main option (let us say, "System Utilities"), you are then brought to the second level menu, which looks like this:

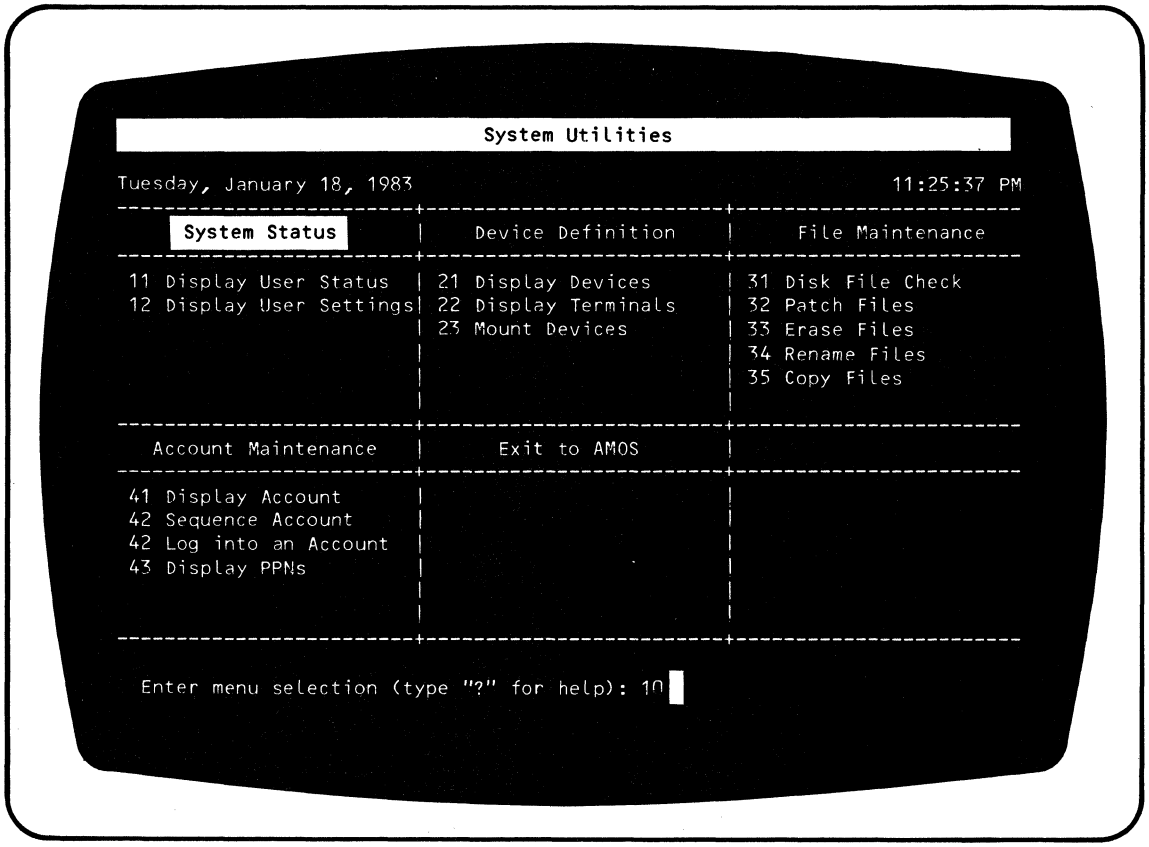

Figure 2-2: Sample Secondary Menu Display

Notice that for most of the secondary level selections, there is a third level (and perhaps more levels) below that. The "Exit to AMOS" section, however, is at the bottom level-- if you select it, it will return you to AMOS/L command level. Here is a diagram that shows what a menu structure might look like:

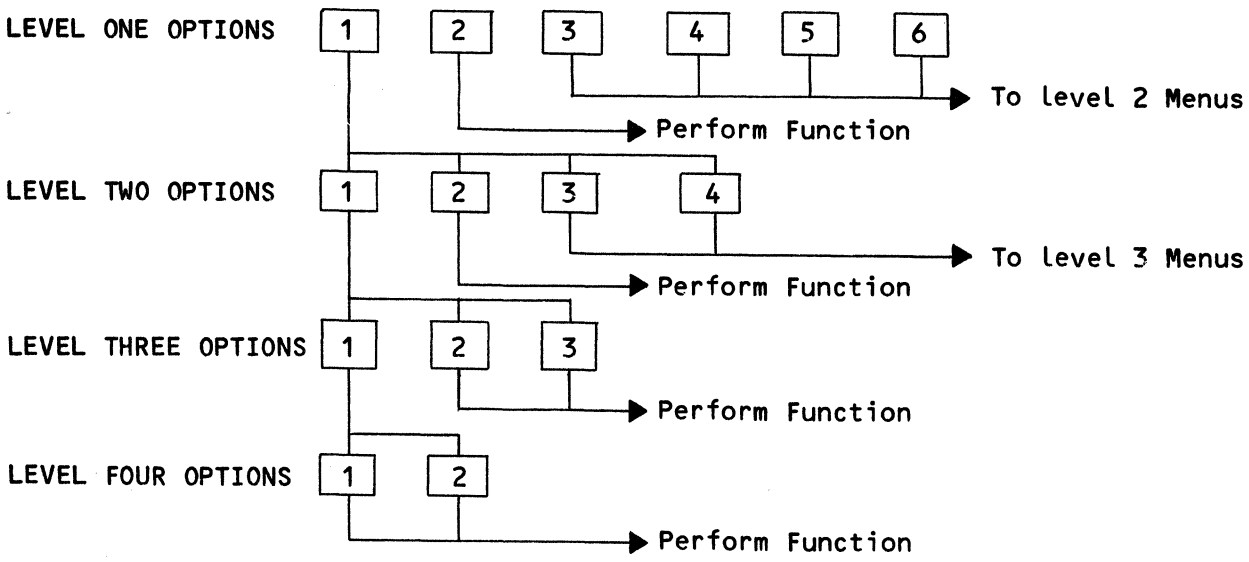

Figure 2-3: Menu Structure Diagram

You may want to think of the menu as a pyramid of different screen displays. The top levels of the pyramid have few but very general options, and the bottom of the pyramid has many specific functions to be performed.

The number of menu levels is defined by the creator of the menu system, and is limited only by the practical limitations of disk and memory space. To return back up through the menu structure, the user presses the Escape key or uses Control-C. (For more information, see Section 3.3, "Using the Menu.")

2.3 ALPHAMENU PROGRAMS

The AlphaMENU programs are:

SHELL.LIT	Displays specified menu
MENU.LIT	Compiles a .MNU file to create a menu.
AMOS.LIT	Returns menu user to the menu after running an AMOS/L program.
AMOSL.MNU	Sample Menu
AMOSL.CMN	Compiled Sample Menu

2.4 MENU FILES (.MNU)

You use AlphaVUE to create a new Menu file or to modify an existing Menu file. By using the AlphaMENU menu definition language, you tell AlphaMENU what options to display in the menu, how many levels the menu will contain, and what actions to perform in response to the user's responses. You also tell AlphaMENU what Help and Query files to use for each selection. Help and Query files are explained below.

2.5 COMPILED MENU FILES (.CMN)

The MENU program compiles the text .MNU menu file into a .CMN file. It is the .CMN file that the SHELL program uses to display the menu.

2.6 HELP FILES (.HLP)

At any level of any branch of the menu, you can display information to the user that is applicable to the option selected. Your menu definition tells AlphaMENU when to display such a Help file. You create a Help file by using AlphaVUE, creating text in any format you choose. The operating system will look for .HLP files first in the account the user is in, next in your library account [P,0] (for example, if you are in [122,3], your library account would be [122,0]) and lastly in the menu library in account DSK0:[7,11].

2.7 QUERY FILES (.QRY)

At the lowest level of any branch of the menu, you can ask the user for information or specifications. This information is fed back to the .CMN file which makes use of the information in invoking the appropriate DO file or AMOS/L commands. Your menu definition tells AlphaMENU when to use a specific Query file. You create a Query file by using an Alpha Micro text editor, using normal text editing procedures. The monitor follows the same search pattern as the Help files above when looking for Query files.

CHAPTER 3

INSTALLING AlphaMENU

3.1 PREREQUISITES

The programs AMOS.LIT, SHELL.LIT, and MENU.LIT must be in account DSK0:[1,4]. This is the default account, which allows these programs to be run from any other account on the system. The Menu Library account is DSK0:[7,11]. This account is reserved for the menus, help and query files that you create for your AlphaMENU system. If a menu, help file, or query file is in this account, it can be run from any other account.

3.1.1 Memory Required

You must have at least 32K of memory in your memory partition in order to run the SHELL.LIT and MENU.LIT programs.

3.1.2 Terminal Features

Although the AlphaMENU menu will display adequately on any CRT terminal compatible with an Alpha Micro computer system, it will display to its fullest advantage on a terminal that contains the features that it automatically makes use of. These features are:

1. A line drawing graphics character set.
2. The ability to underline characters on the screen.
3. The ability to display characters in either high or low intensity.
4. Reverse video display capability.
5. The ability to display blinking characters.

3.2 INVOKING A MENU

To bring a menu to the terminal screen, log into the account containing the menu, and enter:

.SHELL filename (RET)

where "filename" is the name of the menu you want to use. For example:

.SHELL DOC (RET)

if your menu is named DOC.CMN. The sample menu supplied with AlphaMENU is AMOSL.CMN. To invoke it, enter:

.SHELL (RET)

AMOSL.CMN is the default, so simply typing SHELL will bring you this menu.

If you wish users to automatically use a specific menu when they use their system, you might want to include the SHELL command in the job setup commands for each job in your system initialization command file (AMOSL.INI). You should make sure that the job is logged in, and has memory assigned to it, and then use a FORCE command to force the job to run the menu. The series of commands might look like this:

```
ATTACH USER1,RALPH
KILL RALPH
FORCE RALPH MEMORY 76700
WAIT RALPH
FORCE RALPH LOG 100,5
FORCE RALPH SHELL
```

Or, you might want to include a SHELL command in the START.CMD files in the accounts into which the users log.

3.3 USING THE MENU

AlphaMENU has been designed to make the use of the Menu as simple as possible. The sections below show you how to perform the various functions of AlphaMENU.

3.3.1 Making a Selection

There are two different ways that you can select a menu option. The first way is to enter the number of the selection you want. The second is to use the cursor control keys to move the selection marker onto the function you wish to select. Once the selection marker is on that selection, press RETURN, and the next level of menu appears, or that function is selected.

3.3.2 Moving the Selection Marker

The menus are set up so that the selection marker "wraps" around the screen. If you are at any edge of the menu screen, and you press a key that would move the selection marker in a direction that is off the screen, you will see the selection marker appear on the opposite side of the menu.

For example, if the selection marker is placed on "Languages" in our sample menu, and you press the left arrow key, the selection marker will jump to "Business Applications." This features makes it easy to reach any place on the screen with just a few keystrokes. With a little imagination, you can find the quickest way to position the selection marker to any place on the menu.

3.3.3 Getting Help

To get a display of helpful information, type a question mark followed by a RETURN. AlphaMENU looks to see if a help file has been defined for the function on which the selection marker rests. If it finds the file, it displays the contents of the file on the screen. You can customize these help files to explain both normal AMOS/L functions and your own special functions.

Alpha Micro provides a set of help files to go along with the AMOSL.CMN menu. If you make your own menus, you can access these help files, or create your own.

3.3.4 Performing a Function

To perform a function, you make selections until you are at the level where the function you want to perform is displayed. In some cases, you will only have to press RETURN to run that function, in others, you will select it by number (or selection marker position). For example, if you select the option "AlphaBASIC", you will see:

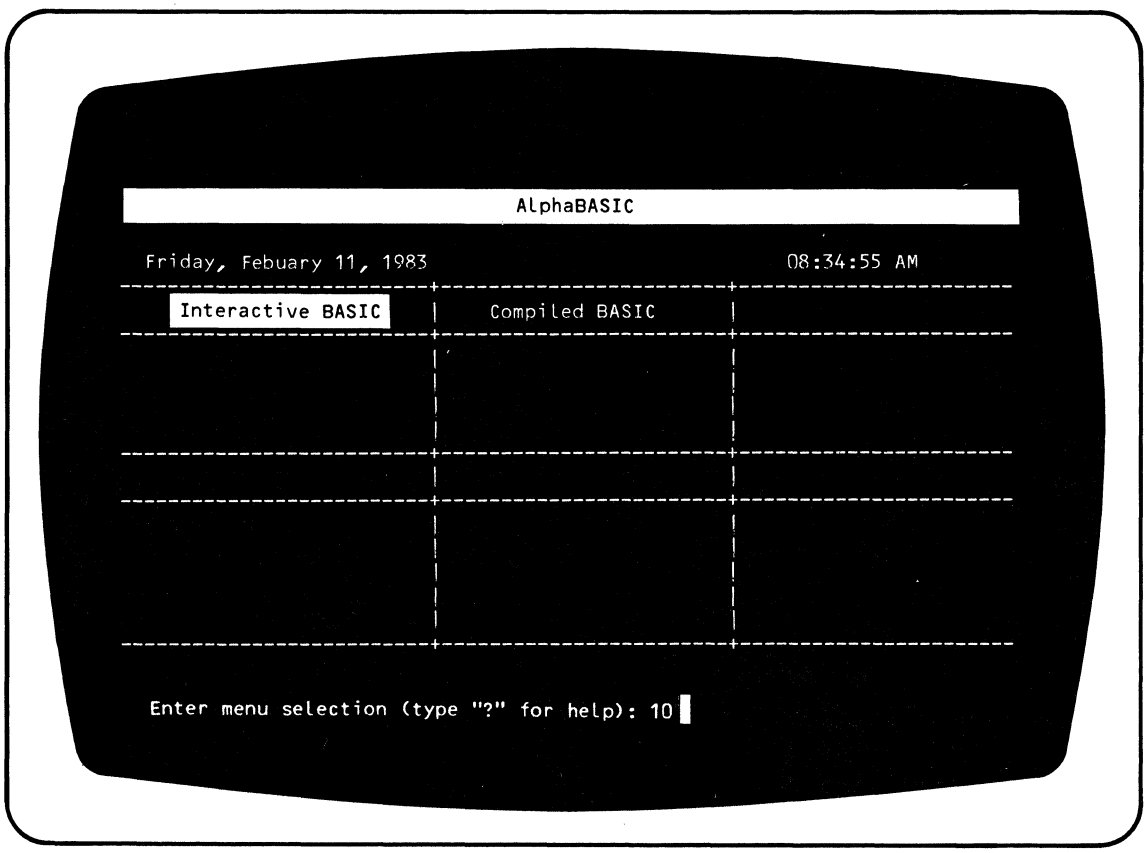

Figure 3-1 AlphaBASIC Menu Screen Example

appears on your screen. The selection marker will be on "Interactive BASIC", so if you now press RETURN, Interactive BASIC will be selected. If you move the selection marker to Compiled BASIC and press RETURN, you will get the compiled mode of BASIC.

3.3.5 Exiting a Menu Display

To go back to the previous menu, press the escape key (often labelled ESC) or use Control-C.

CHAPTER 4

DEFINING A MENU

This chapter discusses how to create a menu and how to define the physical appearance of the menu on the terminal screen. Each menu screen display is divided into six rectangles. A main option for that menu level appears at the top of each rectangle, and any suboptions that may be available are listed under the main option.

You can define up to 6 entries in each rectangle, and each entry name can be up to 16 characters in length. When the user reaches a bottom level of a menu branch, the function is performed.

4.1 CREATING A .MNU FILE

Use the AlphaVUE text editor to create an empty .MNU file. Then enter the components discussed in the sections below.

However, before you actually begin to create the .MNU file, you will find it is a good idea to sit down for a few minutes and design your menu system. Make a list of the actual functions you want to perform. You might begin by making a list of the very general categories of functions you want the user of your menu to perform (e.g., "Mailing Labels," "Office Schedules," "Document Preparation," "System Maintenance," etc.).

These categories will be the main options that will appear on the first, top level menu display. Now, list the options that should appear under each of those categories.

For example, under "Document Preparation" you might list such functions as "Edit Document," "Erase Document," "Display Document," etc. For each of these options, list any suboptions that should appear under it.

The number of levels in your menu is entirely your choice. You can have as many or as few levels as you want. For example, a main option on the top level menu display might have no options listed under it and

would therefore be the beginning and end of that particular branch of the menu. Or, a complicated function might require four or five levels.

After you have this basic structure defined, you will then decide what combination of DO file elements and Query files you will need to accomplish the functions at the bottom level of the menu branches. (See Chapter 5.)

Creating a menu takes some testing and experimentation. When you test your first menu, you will probably find that you have forgotten some functions, or you may find that you want to put a function in a different area, etc.

4.1.1 Level Indicator

In your empty .MNU file, begin to build your menu by entering the various options you have already decided should appear in the menu. You will tell AlphaMENU what menu level a specific option will appear at by entering a level indicator (a number followed by a > or < symbol) followed by the option name. Each option name can be up to 16 characters in length.

The level indicator number tells AlphaMENU what level display the option should appear in. You must always begin a menu with a 0> or a 0< level indicator, which indicates the top level of the menu. Any text you supply after the Level 0 Indicator will be the title of the entire menu, and will appear at the top of the first level menu (which is called the Main or Home level).

The > symbol tells AlphaMENU to display the option name in regular intensity on the screen; a < symbol tells AlphaMENU to display the option name in reduced intensity. (You may define for yourself the meaning of displaying an option in reduced intensity. On menus generated by Alpha Micro, we use the reduced intensity to indicate an option that is not installed on this system but may be available from an Alpha Micro dealer).

Each Level 1 Indicator designates those options that will appear as main options on the top level menu display. Since there are only six rectangles per screen, there can only be six level 1 indicators in an .MNU file (if you specify more, the extra ones will be ignored). Here is an example:

```
1>Document Maintenance
```

Each Level 2 Indicator designates those options that will appear as suboptions on the top level menu display, but which will appear as main options on the second level menu display. There may be up to 36 different level 2 options. Level 3 options will not appear on the main menu. You can have as many levels as your system memory will

allow you to create. Here is an example of what your .MNU file might look like at this stage:

```
0>DOCUMENTATION SYSTEM #1
        1>Log to another Account
        1>Document Maintenance
                2>Edit Document
                2>Rename Document
                2>Copy Document
                2>Erase Document
                2>Display Document
                        3>Terminal Display
                        3>Print
        1>Account Maintenance
                2>See File Directory
                2>Erase *.BAK Files
                2>Resequence Files
        1>System Status
                2>Dynamic Status
                2>Device Status
```

Note that in our sample we indented each level so that the .MNU file is easy to read. This is not necessary, but you might find it helpful when you design your own menu.

Let us follow the structure of one branch of the simple menu defined above. The option "Document Maintenance" appears as a main option on the top level menu display. Each of the Level 2 Indicators listed below that option appear as suboptions on the top level menu display. If the user of the menu chooses the "Document Maintenance" option, a new screen display appears, in which each of the "Document Maintenance" suboptions ("Edit Document," "Create Document," etc.) appears as the main options. Any Level 3 Indicators subordinate to "Document Maintenance" appear as suboptions on this display.

For example, under the main option "Display Document," appear the suboptions "Terminal Display" and "Print". Here is what the second level menu looks like:

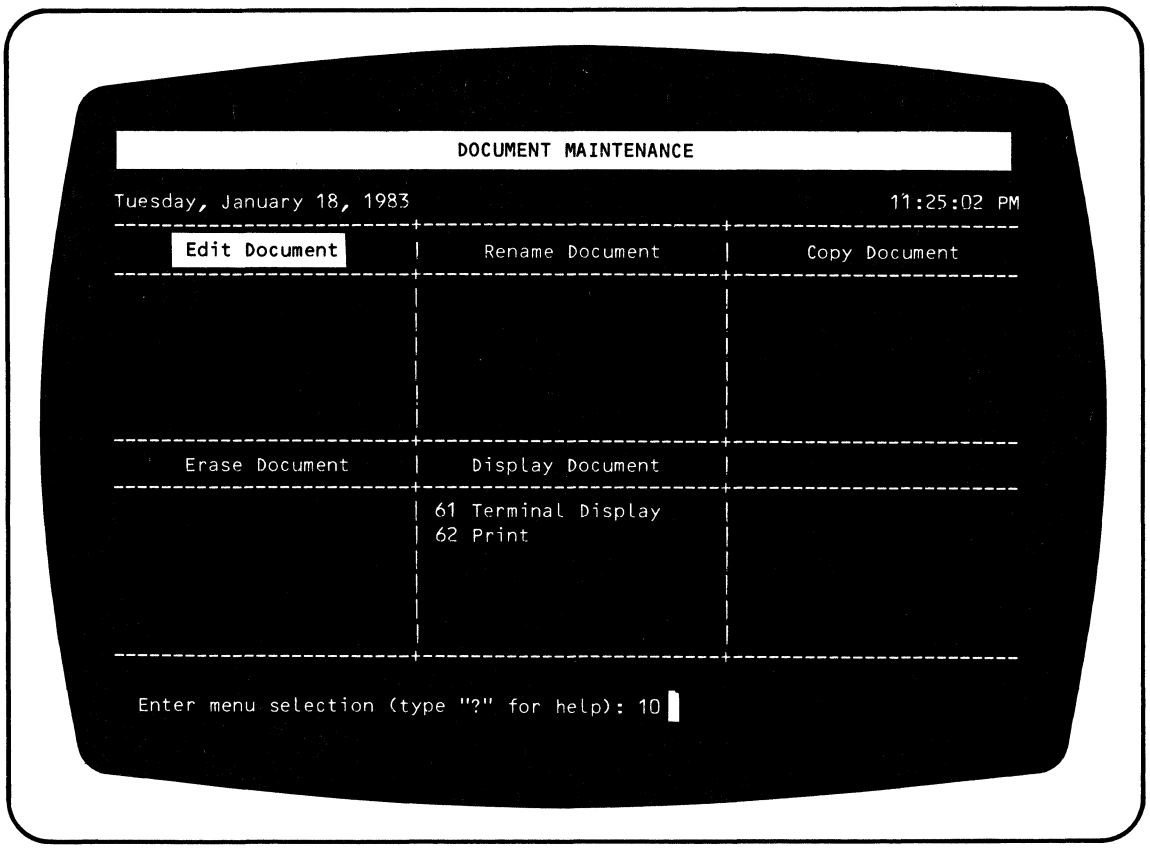

Figure 4-1: Document Maintenance Sample Menu

When a user selects an option that is at the bottom level (that is, no level indicators appear beneath it), he or she has reached the point at which the selected function will actually be performed. In the above example, every option except "Display Document" is at the bottom level.

The sample .MNU file above is a very simple one. In fact, some essential elements are missing from it, preventing it from performing any functions at all except for listing the options for the user of the menu. The next section discusses these other fundamental components.

4.1.2 Query Files and DO File Elements

In order to make the sample menu in the section above perform any of the functions listed as options, we need to define Query files and/or DO file elements. The DO file elements are what actually perform the operations that are listed on the menu. Sometimes the menu file needs input from the user in order to perform a function. For instance, what good is a VUE command unless the user tells it which file he or she wants to edit? The query file allows the user to input information into the menu. We will discuss these elements in greater detail in Chapter 6. For now, we will explain how to specify them in the menu file.

To tell AlphaMENU to use a Query file, enter the name of the Query file preceded by a @ symbol. For example, to tell AlphaMENU to use the BASIC.QRY file, place:

```
@BASIC
```

in your .MNU file after the level selection:

```
1>AlphaBASIC
   @BASIC
```

(AlphaMENU assumes a default extension of .QRY). You may enter any combination of upper and lower case when specifying Query files or DO file elements; AlphaMENU doesn't see any difference between "@BASIC", "@Basic", or "@basic". The @ symbol indicates a query file.

NOTE: A query file must be run only at the bottom level, where a function is going to be performed. In the example above, for instance, you could not have a level 2 (or lower) indicator below @BASIC. It would have to be followed by another level 1 indicator. If you have a lower level specified below a query file, that level would never appear on the menu.

DO file elements allow your menu file to build a DO file in memory. You may use any of the usual DO file symbols. Enter a DO file element after a period. As with regular DO files, you may specify any command line that is legal at AMOS/L command level. For example, to tell AlphaMENU to run AlphaBASIC, enter:

```
.BASIC
```

You can also specify data or commands understood by the program currently running.

You may want to have the DO file elements perform their tasks without displaying commands on the screen. In a regular DO file, you can do this with the :R and :S commands. You can use these commands in the .MNU file also. For example:

```
.:S
.COMPIL MYPROG.BAS
.:R
.RUN MYPROG
```

In the above example, you would not see any of the displays from the COMPIL program, but you would see the results of the RUN.

Query files can return information entered by the user, which can be substituted into DO file parameters in your menu file (designated by the symbols $0 - $9). The $n symbols are used just like their equivalents in regular DO files-- the arguments to be returned are passed into the symbols defined for them (for ease of understanding, the symbols are usually defined in order-- the first argument into $0, the second into $1, etc.)

For example, if the GETSPC.QRY Query file returns three arguments, "/D", "MYFILE.LST", and "HEADER.LST", the DO file element that follows the @GETSPC in the menu might contain ".RENAME$0 $1=$2", and thus generate the command "RENAME/D MYFILE.LST=HEADER.LST":

```
1>Document Maintenance
        2>Rename Document
          @GETSPC
          .RENAME$0 $1=$2
```

You may also specify the names of command or DO files that exist on your own system. They will be executed just as they would at AMOS/L command level.

4.1.3 Chaining to other Menus

If you want a menu to run another menu, use a DO file command to invoke it:

```
1>Document Maintenance
      .SHELL NEWMNU.CMN
```

4.1.4 Help Files

A Help file (discussed in the next chapter) contains text concerning an appropriate subject that you want the user to see when he or she asks for help by entering a question mark followed by a RETURN. Unlike a Query file, a Help file does not have to appear at the bottom level of a menu branch. To specify a Help file in .MNU file, enter a ? symbol followed by the name of the Help file. For example:

```
?BASIC
```

in the menu file, it would look like this:

```
1>AlphaBASIC
   ?BASIC
      @BASIC
      .BASIC
```

Thus, if the user has the cursor on "AlphaBASIC" and enters a "?" and a RETURN, he or she will see the contents of the file BASIC.HLP displayed on the terminal screen. You may enter the file name in any combination of upper and lower case. The default extension is .HLP.

4.2 SAMPLE MENUS

Now that we have discussed some of the elements of a menu file, here is our earlier example which now contains specifications of Query files, Help files, and DO file elements:

```
0>DOCUMENTATION SYSTEM #1
      1>Log to another Account
        ?LOG
           @LOG
           .LOG $0
      1>Document Maintenance
        ?DOC
           2>Edit Document
             ?EDIT
                @EDIT
                .VUE $0
           2>Rename Document
             ?RENAME
                @GETSPC
                .RENAME$0 $1=$2
           2>Copy Document
             ?COPY
                @GETSPC
                .COPY$0 $1=$2
           2>Erase Document
             ?ERASE
                @ERASE
                .ERASE $0
           2>Display Document
             ?DISPLY
                3>Terminal Display
                    @TERM
                    .TYPE $0
                3>Print
                  ?PRINT
                    @PRINT
                    .PRINT $0$1
 1>Account Maintenance
           2>See File Directory
                @DIR
                .DIR$0
           2>Erase *.BAK Files
                .ERASE *.BAK
           2>Resequence Files
                .DIRSEQ
1>System Status
           2>Dynamic Status
                .STAT
           2>Device Status
                .SYSTAT
```

Note again how we indent the command lines so that the various levels can be easily seen. If you have to later correct or add something to the menu file, you can find the appropriate section quickly.

4.2.1 The : Symbol

A colon, :, indicates that whatever follows it on the line is a special AlphaMENU command. At present, the only two commands are :P and :EXIT. The :EXIT commands ends the SHELL.LIT program and returns the user to AMOS/L command level. You may want to have an "Exit to AMOS" option on your menu, or you may want the users of the menu to be "locked" into the menu. It depends upon your needs. You may even define different menus for different departments or for different users.

When a function on the menu has been executed, you may see a "Type CR to return to Menu" message displayed at the bottom of the screen. Whether or not you see this message depends on how you set up your menu file. Depending on what type of function was just performed, you may want to add a :P command after the function's DO statements. The :P command causes the "Type CR to return to Menu" question to be skipped over. When :P is inserted into a function, the ending of that function returns the user immediately to the menu, rather than displaying the prompt question.

A good example of a place where you do want to see the "Type CR to return to Menu" prompt is after executing SYSTAT. If you went immediately back to the menu, the user would have no time to look at the SYSTAT display! With the prompt, the user is returned to the menu only when he or she is ready to return.

In many cases, though, the function is such that the "Type CR to return to Menu" prompt is just a waste of the user's time. An example would be after exiting AlphaVUE. There is no display to be seen, and thus no reason to wait before re-displaying the menu.

4.3 COMPILING A MENU

Once you have finished your menu file, you must have it compiled before it can be run. To compile a menu, be sure that you are logged into the account containing the .MNU file and enter:

```
.MENU filename.MNU (RET)
```

where "filename" is the name of your Menu file. (The compiler assumes a default extension of .MNU.) When MENU finishes compiling the Menu file, you see the message:

```
Pass 1: 0 errors detected
Pass 2: Menu file completed, menu code size is xxx bytes
.
```

If an error is detected in your .MNU file, you will see an error message:

```
Pass 1: ?Syntax error - 1 error detected
```

If you see such a message, check your .MNU file for errors. A compiled menu file has now been created in the account you are logged into that has the filename of your menu and the extension .CMN. To invoke the menu, use the command:

```
.SHELL filename (RET)
```

where "filename" is the name of your .CMN file.

NOTE: MENU only compiles the .MNU file-- any Query or Help files specified inside the .MNU file are not incorporated into the .CMN file. Instead, when the user invokes your menu, SHELL looks for them in the accounts mentioned in Section 2.6. Query and Help files do not have to be compiled.

CHAPTER 5

THE HELP FILE

To create a help file, use AlphaVUE and TXTFMT to create a file containing the helpful text you would like to have displayed, and then use the features of AlphaVUE and TXTFMT to arrange the text the way you want it to look on the terminal screen.

Format the file, and then rename or copy the .LST file to give it a .HLP extension:

```
.RENAME MYMENU.HLP=MYMENU.LST (RET)
```

You may add special screen displays to your help file if you have terminals that support them, and if you want them. Section 5.3 below discusses how to create special screen displays.

5.1 SCREEN POSITIONING

The menu begins by clearing the screen. Then it displays the help file text until it reaches the end, or until a full screen is displayed. The message:

```
Type CR to continue -
```

is displayed at the bottom of the screen if there is still more of the help file to be seen.

5.2 SAMPLE HELP FILES

Here are two examples of what a help file will look like when displayed on the terminal screen:

SAMPLE #1:

DISPLAY ACCOUNT

There are times that you will find it necessary to find out what files are currently in your account. To accomplish this you may use the option DISPLAY ACCOUNT.

This function allows you to get either of two types of listings of the files in your account.

Minimal Statistics, displays to the terminal, in four columns, a listing of the files that are in your account, along with the size, in blocks, of each of the files.

Full Statistics, displays to the terminal, one entry per line, a listing of the files that are in your account. Besides the information given above, this option displays each file's hash total (i.e., the file's fingerprint), the base disk address of each file, and a "C" next to those files that are contiguous files (random data files).

In order to do this, simply select one of the two functions and press return.

SAMPLE #2:

LOGGING INTO AN ACCOUNT

The function LOGGING INTO AN ACCOUNT has two options. First, you may find out which account you are currently logged into by selecting Determine current account. Second, you may move from your current account to another account by selecting Move to another account.

If you select the option Determine current account, the system displays the account to the terminal.

Current login is DSK0:[30,30]

If you select the option Move to another account, the terminal displays:

Enter account number:

Here you enter the account number to which you wish to log.

You may be required to supply a password before the system will log you into an account. If so, the terminal displays:

Password:

Enter the appropriate password and press RETURN. The system does not display your password to the screen as you type it; this prevents other users from seeing your password.

For further information regarding the log function you may refer to the AMOS/L System Commands Reference Manual, DSS-10004-00.

5.3 USING SPECIAL SCREEN DISPLAYS IN HELP FILES

Depending on your terminal and its capabilities, there are many different special displays you can use with a help file. Reverse video highlighting, low-intensity display, underlining, blinking characters-- these are just a few of the graphics you may be able to use to make parts of a help file stand out.

These features of screen control are contained in assembly language subroutines. You can "call" these subroutines by using TCRT calls. A TCRT call is a number that corresponds to a subroutine that will perform a special screen display. Appendix B contains a complete list of the standard TCRT calls.

If you want to add some of these features to your help file, use AlphaVUE again to edit the .HLP file. Go to AlphaVUE command mode and enter "CONTROL TRUE" and press RETURN.

Now return to the text by using ESC. Find the places in the text where you wish to have special displays occur (such as reverse video display, blinking characters, underlining, etc.). Move the cursor to the place within the text where you want the special display to begin. DO NOT place the cursor on a text character-- place it on a blank space. Otherwise, when you insert the TCRT number, you will write over your text.

When the cursor is properly positioned, use Control-G and press the escape key. You will see a ^[displayed at that point in your text. Now enter the number of the TCRT call you wish performed (see Appendix B for the applicable TCRT calls).

Now you have enabled one of the video display modes-- remember to repeat this process to turn "off" the mode you just enabled somewhere later in your text (when you don't want that type of display anymore).

For example, say that you are creating a help file to explain the use of the DIR command. You would like the majority of the text to be displayed in reduced intensity, and the examples to be in normal intensity (so that they stand out). You enter the text:

The DIR command allows you to display a directory listing of the files that are in your account. You can also display listings of other accounts. The format is:

.DIR {Listfilespec=}{Filespec1{,Filespec2,...FilespecN}}{/Switches}

To see the account you are in:

.DIR/W

To see account DSK1:[100,5]:

.DIR DSK1:[100,5]

Now exit AlphaVUE, format the file and rename it (as described above). When you are again inside the file (and CONTROL=TRUE is set), place the cursor at the beginning of the first sentence. Use Control-G, and then type in the number "11". Move the cursor onto the colon at the end of the third sentence, use Control-G, and type in "12" (this sets the display back to normal intensity).

Repeat this process at the beginning and the end of the seventh and eleventh lines. Then Finish out of the file.

NOTE: These video displays work only when accessed through AlphaMENU. For example, typing:

.HELP DIR

will not display the special effect. You will only see the ^[and the numbers you entered. What this means is that help files with normal text can be used by AlphaMENU, but help files with special video displays designed for AlphaMENU cannot be used as regular help files.

CHAPTER 6

QUERY FILE AND DO FILE ELEMENTS

The reason we have included discussions on both Query files and DO file elements in this one chapter is that they interact with one another quite a bit, and full understanding of either requires that you be familiar with both. Our examples below show some types of interactions. Although the examples we show are very simple, with a little imagination you can create very powerful and interesting combinations of Query files and DO file elements.

6.1 QUERY FILES

Query files are files that the menu uses to ask the user for information. They contain explanatory text and special query file commands which allow the user to input text and which place that text into DO file variables. Query file commands have more power and dimension than ordinary DO file user input commands.

For example, how could the menu know which file you wish to edit with AlphaVUE unless you tell it the filename? The query file allows you to input variables such as this to the menu.

Note that one query file may be used for more than one function. For example, a query file that contains only the question:

```
Enter filename:
```

could be used to provide the filename for many different commands, such as VUE, TYPE, ERASE, COMPIL, etc.

6.2 QUERY FILE COMPONENTS

The purpose of a query file is to get specific information from the user and return it to the menu in the form of DO file variable $0-$9. In this form, they can be used by DO file elements to perfrom specific tasks. Query files are much like the :K (keyboard input) symbol in normal DO files. But have more features, so that you can control what your questions look like, and where and how they will appear on the screen.

6.3 TEXT

You may include any type of text you wish inside a query file. It is often a good idea to give a brief description or set of instructions with a query file. For example, the query file:

```
                    ERASING GROUPS OF FILES

   This section allows you to erase all files with certain extensions.

   Place the selection marker on the option you would like to perform,
   and press RETURN:


     $S,12,20,Erase all BAK files,*.BAK
     $S,14,20,Erase all LST files,*.LST
     $S,16,20,Erase all OLD files,*.OLD
     $G,0
     $P
```

would produce a screen display like this:

```
                    ERASING GROUPS OF FILES

 This section allows you to erase all files with certain extensions.

 Place the selection marker on the option you would like to perform,
 and press RETURN:



                    Erase all BAK files

                    Erase all LST files

                    Erase all OLD files
```

Anything that does not begin with a dollar sign is printed on the screen as text. As you can see, this feature allows you to design your screen displays any way you like.

Be careful, however, to confine your text to one screen page. Don't run your lines too long, or put more lines than the screen can hold. And remember to leave room for the prompt text.

6.3.1 The $S Command

The $S command defines a prompt display to be placed on the terminal, and what text value will be transferred to the DO file command if it is selected. The $S command is useful when you wish to place a number of options on the screen and have the user choose from the list (rather than asking the user for an answer to a specific question). The format is:

```
$S,Line#,Column#,Prompt text,text value
```

where the line and column numbers specify the position that the prompt text will begin printing on the terminal screen, the prompt text defines what will be displayed on the terminal as the user prompt (the name of the option, an input question, etc.), and the text value is what will be transferred to the DO file argument by the $G command (explained below).

You can specify any number of $S commands (the limit being how many can fit on the screen). The user then places the selection marker on the one he or she wishes, and presses RETURN. The $G command works as the "input" for two or more $S commands. Depending upon where the selection marker is when the RETURN key is pressed, the text value of that $S statement will be placed into the DO file argument (say, $0) by the $G command.

The top left-hand corner of the terminal screen is defined as line 1, column 1. If you want an option to appear 15 lines down, and indented 20 spaces from the left margin of the screen, specify:

```
$S,15,20,Erase *.BAK,*.BAK
```

Be careful to limit your prompts to the size of your terminal screen-- if you specify a line number or column number that is off the screen, your query file will not work properly.

NOTE: If your terminal does not support reverse-video, $S commands may be hard to use. Because $S displays must be selected using the selection marker (which is a reverse-video display), the user may have trouble making selections. You can use $I commands, or you can modify your terminal driver to simulate the selection marker (see Appendix B).

6.3.2 The $G Command

Gets the text value from the selected $S command and puts it into a DO file argument. The format is:

```
$G,argument#
```

where argument# is the DO file variable into which the text value from the $S statement is to be placed.

Let us consider an example to clarify matters. Say that we have an option for a "Directory display". When that option is selected, we want to have a display like this appear:

```
Display full account in wide format

Display only .TXT files (/W)

Display .CMD files
```

The user could then select which of the three types of DIR displays he or she wishes by placing the selection marker on that option and pressing RETURN. The query file elements needed to produce the above display are:

```
$S,12,25,Display full account in wide format,*.*/W
$S,14,25,Display only .TXT files (/W),*.TXT/W
$S,16,25,Display .CMD files,*.CMD
$G,0
```

You can see how the prompt texts are placed on lines 12,14,16, all beginning in column 25. Notice that the argument number for the $G command is 0. Now, the DO command in the menu file would look like this:

```
.DIR $0
```

The $G command will take the text value of whichever $S line is selected and send it to the menu file as $0. If the user places the selection marker on "Display full account in wide format" and presses RETURN, $0 will become *.*/W, and the DO command in the menu will become DIR *.*/W. If the second line is selected, $0 will become *.TXT/W, etc.

6.3.3 The $I Command

The $I command prints a specific question on the screen. The answer that the user inputs is then placed into a DO file command argument (for instance, into $0). The format is:

$I,Argument#,Line#,Column#,Prompt

where the argument number is the DO file argument that the text will go to (such as $0), the line and column numbers are the position on the terminal screen where the prompt is placed, and the prompt is the text displayed on the screen that prompts the input. $I is used to place a straight question on the screen for the user to answer. For instance, if you are in the AlphaVUE section of your menu, you might have a DO command like this:

.VUE $0

Then, in the query file, you might have:

$I,0,20,10,Enter the name of the file you wish to Vue:

Whatever the user enters after this prompt is displayed will become $0. For example, if the user enters:

Enter the name of the file you wish to Vue: MYTXT (RET)

then the menu file will execute the DO command: VUE MYTXT.

Note that because you are able to specify where on the screen the prompt text appears, you can control exactly how your screens are displayed. You can put all of your text first in the query file, and then position the questions at the end. Or, you could put part of the text first, then a question, then more text. In this case, the user wouldn't see the second part of the text until after the question was answered (or the selection was made). You can also have the questions placed in the middle of text.

6.3.4 The $P Command

Causes a page eject. This command is used at the end of a query file if you do not want a "Type CR to return to main menu" message to appear after the query file is finished executing. The format is:

$P

6.3.5 The $@ Command

Causes another query file to be executed. The format is:

```
$@filename
```

where filename is the name of a query file. The default extension is .QRY. See section 6.3.7 below for an example.

6.3.6 The $= Command

Sets the DO file variable to text. The format is:

```
$=argument#,text
```

Where argument# is the DO file variable (such as $0), and text is what is assigned to it. For example, the statement:

```
$=1,FISH
```

causes the $1 argument to be "FISH". The next section gives further examples.

6.3.7 Query Files that Branch

The most powerful feature of the query file is its ability to perform a limited type of IF-THEN processing. You are able to conditionally branch to other query files based upon user input. For example, if the user selects from three options, the query file can branch to three different query files depending on which choice is made. The "$G" command moves the text at the end of a selected "$S" entry into an argument, and if that text is the name of another .QRY file, then a "$@" command executes the new .QRY file.

If secondary query files do not need, or only require, a single argument, then the "$=" command allows you to force a text value.

For example: You select an option called "Mount Devices". The screen presents you with the choices:

```
Select from the following:

          Mount a device
        Unmount a device
    Display Mounted Devices
```

Here is what the MOUNT.QRY file that handles these choices looks like:

```
                    Select from the following:

$S,15,20,     Mount a device       ,MOUNT.QRY
$S,16,20,   Unmount a device       ,UNMNT.QRY
$S,17,20,Display Mounted Devices ,DISPLY.QRY
$G,1
$@$1
$P
```

In the menu file itself is the command:

```
.MOUNT $0$2
```

In other words, the user must supply an argument-- a name must be specified for the device that is to be mounted or unmounted. To unmount a disk, a "/U" must be appended to the end of the DO file command so that it looks like this:

```
.MOUNT Devn:/U
```

In the third case (Display Mounted Devices), you do not need to specify any arguments to the DO file-- it will merely perform:

```
.MOUNT
```

The way to accomplish these three options is to create three secondary level query files-- one for each $S option. The $G command will place into the $1 variable the name of a query file. Then, the $@ command will cause that query file to be executed.

The first secondary query file (MOUNT.QRY), which corresponds to "Mount a device", looks like this:

```
                    ** The MOUNT Command **


$I,0,10,23,Enter device:
$=2,
$P
```

The $I statement places whatever the user enters after the question "Enter device:" into $0. The $= command makes $2 a blank. For example, if the user enters DSK2:, the DO file command will be:

```
.MOUNT DSK2:
```

The second query file (UNMNT.QRY), which corresponds to "Unmount a disk", looks like this:

** The UNMOUNT Command **

```
$I,0,10,23,Enter device:
$=2,/U
$P
```

Here, the $I command is the same, but the "$=" command specifies that $2 in the DO file command line will equal "/U". This then makes the DO file command line in the menu file look like this (using, as above, an example of DSK2: specified by the user):

```
.MOUNT DSK2:/U
```

which unmounts DSK2:. The third query file (DISPLY.QRY) contains:

```
$=0,
$=2,
$P
```

The "$=0," and "$=2," lines assign blank text strings to $0 and $2, so that the DO file command in the menu file becomes:

```
.MOUNT
```

which displays all currently mounted devices.

It is most important that you realize that you are doing text processing inside query files. You are moving text from a selection to an argument or forcing text into an argument in order to create a text string to be executed by the DO processor of AMOS.

6.3.8 Sample Query Files

** The COPY Command **

Use the COPY command to copy one or more files: within accounts, between accounts, and between disks.

Select from the following:

```
$S,10,23, Copy/query    ,/Q
$S,11,23, Copy/no query ,
$S,12,23, Copy/no delete,/NOD
$G,2
$I,0,13,12,Enter New File Specification:
$I,1,14,12,Enter Old File Specification:
$P
```

In the example above, the lines of text at the top are displayed on the terminal screen. Then the three $S statements cause this:

```
Copy/query
Copy/no query
Copy/no delete
```

to be displayed below the text. At this point, because of the $G, the query file halts and waits for an input. The user places the selection marker on one of the three options above, and presses RETURN. If the first option is selected, the "/Q" will be taken from the $S command, and placed (by the $G command) into $2 for later DO file processing.

Now the message "Enter New File Specification:" appears, and the file stops for an input. The user inputs a file specification, and the $I command places that text string into $0. Then the next $I command is executed, asking "Enter Old File Specification:". When this text string is entered, and $I has placed it into $1, the query file ends (using $P so no "Type CR to return to main menu" message is output).

In the menu file, the DO file command:

```
.COPY$2 $0=$1
```

is executed, using the information input. Here is another example:

```
                ** The DISPLAY ACCOUNT command **

    Use the DISPLAY ACCOUNT command to display the contents of
    your account to the terminal. This function lists only the
    the contents of your current account. The following options
    are available:

                    Minimal statistics
                    Full statistics


    $S,10,23,Minimal statistics,/W
    $S,11,23,Full statistics   ,/F
    $G,0
    $P
```

Here the query file is working with a DO file element in the menu file that looks like this:

```
.DIR$0
```

and will make it either .DIR/W or .DIR/F.

APPENDIX A

SUMMARY OF SPECIAL ALPHAMENU SYMBOLS

A.1 > HIGHLIGHTED ENTRY

The entry will appear on the menu in normal (highlighted) intensity.

A.2 < - REDUCED INTENSITY ENTRY

The entry will appear on the menu in reduced (dim) intensity.

A.3 . - DO COMMAND

Causes the text following the period to be executed as a DO file command.

A.4 @ - QUERY COMMAND

Causes the query file specified after the @ to be executed.

A.5 ? - HELP COMMAND

Causes the help file specified after the ? to be executed.

A.6 $G - GET SELECTED TEXT FROM VALUE

Causes an input choice between prior $S generated prompts. The text value from the selected $S statement is passed into the DO file argument number. The format is:

$G,argument#

A.7 $I - INPUT LINE OF TEXT INTO ARGUMENT

Inputs a line of text into a DO file argument. The format is:

$I, Argument#,Line#,Column#,Prompt

A.8 $P - PAGE EJECT

Suppresses "Type CR to continue" message that is normally output at the end of a query file.

A.9 $S - SELECTION ITEM DEFINITION

Causes the prompt text to be displayed at position x,x on the terminal screen. The text value is passed on to the DO file argument by a $G command at time of input. The format is:

$S,Line#,Column#,Prompt text,text value

A.10 $0 - $9 - ARGUMENT NUMBERS

These are the argument numbers that will be passed to the DO file to be executed.

A.11 $= - SET DO FILE VARIABLE TO TEXT

Sets the DO file variable to text. For example, $=1, FISH causes the $1 argument to be "FISH".

A.12 $@ - GO TO ANOTHER QUERY FILE

$@ will execute the contents of a query file. The format is:

```
$@,filename
```

A.13 :EXIT

Used within the menu file. The menu ends and the user is returned to AMOS command level.

A.14 :P

Used within the menu file, :P causes the output of following DO file commands to be silenced-- that is, they will not appear on the screen.

APPENDIX B

TCRT Functions

The following codes can be used within query files and help files to create special effects on the terminal screen. See Chapter 5 for instructions.

Code	Function
0	Clear screen and set normal intensity
1	Cursor home (move to 1,1 - upper left corner)
2	Cursor return (move to column 1 without line-feed)
3	Cursor up one row
4	Cursor down one row
5	Cursor left one column
6	Cursor right one column
7	Lock keyboard
8	Unlock keyboard
9	Erase to end of line
10	Erase to end of screen
11	Enter background display mode (reduced intensity)
12	Enter foreground display mode (normal intensity)
13	Enable protected fields
14	Disable protected fields
15	Delete line
16	Insert line
17	Delete character
18	Insert character
19	Read cursor address
20	Read chracter at current cursor address
21	Start blinking field
22	End blinking field
23	Start line drawing mode
24	End line drawing mode
25	Set horizontal position
26	Set vertical position
27	Set terminal attributes
28	Cursor On
29	Cursor Off
30	Start underscore
31	End underscore
32	Start reverse video

33	End reverse video
34	Start reverse blink
35	End reverse blink
36	Turn off screen display
37	Turn on screen display
38	Top left corner symbol
39	Top right corner symbol
40	Bottom left corner symbol
41	Bottom right corner symbol
42	Top intersect symbol
43	Right intersect symbol
44	Left intersect symbol
45	Bottom intersect symbol
46	Horizontal line
47	Vertical line
48	Intersection
49	Solid block
50	Slant block
51	Cross-hatch block
52	Double line horizontal
53	Double line vertical
54	Send message to function key line
55	Send message to shifted function key line
56	Set normal display format
57	Set horizontal split (follow with row code)
58	Set vertical split (39 char columns)
59	Set vertical split (40 char columns)
60	Set vertical split column to next char
61	Activate split segment 0
62	Activate split segment 1
63	Send message to host message field
64	Up arrow symbol
65	Down arrow symbol
66	Raised dot
67	End of line marker
68	Horizontal tab symbol
69	Paragraph symbol
70	Dagger symbol
71	Section symbol
72	Cent sign
73	One-quarter symbol
74	One-half symbol
75	Degree symbol
76	Trademark symbol
77	Copyright symbol
78	Registered symbol

Not all terminal drivers have all of the functions above simply because not all terminals are able to perform all of these functions. If your terminal has additional features, Alpha Micro recommends starting at 128 (decimal) when you assign function codes in your terminal driver.

B.1 IF YOUR TERMINAL DOES NOT SUPPORT SPECIAL FUNCTIONS

If your terminal does not support some of the features used in AlphaMENU, especially the reverse-video used for the selector marker, you may consider modifying your terminal driver to substitute some type of effect your terminal screen can support that will serve to show where the selector marker is on the screen. Be certain that you understand the procedure for modifying a terminal driver, or you could cause a problem. Contact your Alpha Micro Dealer for advice.

Index

AlphaMENU User's Manual

Document History

Revision A00 - (Printed March, 1983) - New Document

TECHNICAL PUBLICATIONS READERS COMMENTS

We appreciate your help in evaluating our documentation efforts. Please feel free to attach additional comments.
If you require a written response, check here: ☐

NOTE: This form is for comments on documentation only. To submit reports on software problems, use Software Performance Reports (SPRs), available from Alpha Micro.

Please comment on the usefulness, organization, and clarity of this manual:

Did you find errors in this manual? If so, please specify the error and the number of the page on which it occurred.

What kinds of manuals would you like to see in the future?

Please indicate the type of reader that you represent (check all that apply):

- ☐ Alpha Micro Dealer or OEM
- ☐ Non-programmer, using Alpha Micro computer for:
  - ☐ Business applications
  - ☐ Education applications
  - ☐ Scientific applications
  - ☐ Other (please specify):

    _____
- ☐ Programmer:
  - ☐ Assembly language
  - ☐ Higher-level language
  - ☐ Experienced programmer
  - ☐ Little programming experience
  - ☐ Student
  - ☐ Other (please specify):

    _____

NAME: _____ DATE: _____

TITLE: _____ PHONE NUMBER: _____

ORGANIZATION: _____

ADDRESS: _____

CITY: _____ STATE: _____ ZIP OR COUNTRY: _____

STAPLE

STAPLE

FOLD

PLACE
STAMP
HERE

alpha micro

17332 Von Karman
P.O. Box 18347
Irvine, California 92714

ATTN: TECHNICAL PUBLICATIONS

FOLD

FOLD