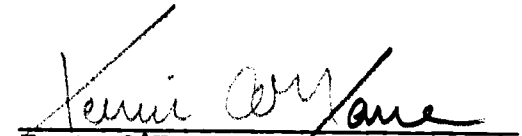
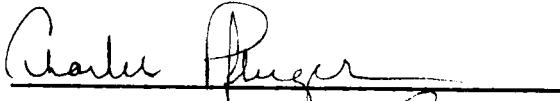


To the Graduate Council:

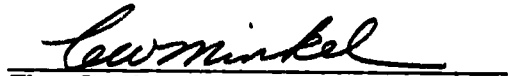
I am submitting herewith a thesis written by Jimmy Elaine Jansen entitled "OATS: An Office Automation Tool System For The Digital Equipment Corporation VAX Computer." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.


Kevin O'Kane, Major Professor

We have read this thesis
and recommend its acceptance:




Accepted for the Council:


The Graduate School

STATEMENT OF PERMISSION TO USE

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

Permission for extensive quotation from or reproduction of this thesis may be granted by my major professor, or in his absence, by the Head of Interlibrary Services 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 Jimmy Elaine Jensen
Date May 7, 1985

OATS: AN OFFICE AUTOMATION TOOL SYSTEM FOR THE
DIGITAL EQUIPMENT CORPORATION
VAX COMPUTER

A Thesis
Presented for the
Master of Science
Degree
The University of Tennessee, Knoxville

Jimmy Elaine Jansen

June 1985

DEDICATION

I would like to dedicate this thesis to my daughter Rebecca Ann Jansen, in appreciation for her prayers and in the hope that it may encourage her in the pursuit of excellence in whatever she attempts in this life.

ACKNOWLEDGMENTS

I am indebted to Dr. Kevin O'Kane, Dr. Charles Phleegeer, and Dr. Michael Thomason, who served as my chairman and committee respectively, for their consultation, criticism, and encouragement.

ABSTRACT

The need for an easy-to-learn, simple to use office automation system for the Digital Equipment Corporation VAX Computer which did not require the addition of expensive software formed the basis for the design of the system presented in this thesis. Although there are two office automation systems currently available for the VAX, they are not appropriate for many situations.

All-IN-1, from Digital Equipment Corporation, is expensive and requires many terminals and personal computers to be cost-effective and efficient. It is not suitable for a small or single-user application.

VAX/VMS Professional Workstation, available from the Digital Equipment Corporation Users' Society, is certainly more cost-effective, regardless of the size of the system or number of users, but it is not well thought out and as a result is complicated to use and cumbersome.

Office Automation Tool System (OATS) is an attempt to overcome the difficulties mentioned above and has resulted in a system that is both easy to learn and use. The doubly-linked list structure of the menu system and the fact that it is written almost exclusively in Digital Command Language (DCL) makes OATS easy to customize and enhance.

TABLE OF CONTENTS

CHAPTER	PAGE
I. INTRODUCTION	1
General Concepts.....	2
Design Considerations.....	3
Overview of Three Systems	7
OATS -- Office Automation Tool System.....	10
II. DESCRIPTION OF OATS	13
Capabilities.....	13
Architecture	15
Major Command Files.....	21
File Storage System.....	36
Utilities.....	41
III. CUSTOMIZATION AND PERSONALIZATION OF OATS	46
Addition of a Program.....	46
Addition of a Menu.....	47
Additional Information on Customizing.....	48
An Example.....	50
IV. OATS TUTORIAL	53
Invoking OATS.....	53
Menu Selection.....	54
Creation and Filing of Documents.....	62
Control Y.....	65
V. SUMMARY AND FUTURE ENHANCEMENTS	66
BIBLIOGRAPHY	68
APPENDIXES	71
APPENDIX A. Digital Command Language Programs	72
APPENDIX B. Macro Assembly Language Program	127
VITA	130

LIST OF TABLES

TABLE	PAGE
1. Types of Office Automation Systems	4
2. Functions and the Methods that Have Been Used to Achieve those Functions in OATS.....	13
3. Format for DCL Assignment Statement for a Menu Option.....	18
4. Format for DCL Assignment Statement for a Program Option.....	18
5. Global Symbols Used in OATS and Available to Command Files Added to OATS.....	24
6. Utility Command Files	42

LIST OF FIGURES

FIGURE	PAGE
1. Example of User-Friendly Help Menu	5
2. Functions Available with VAX/VMS Professional Workstation.....	9
3. Flow of Control Through the Menus of OATS	16
4. Flow of Control Through OATS Command Files	21
5. Global Symbol Values Extracted by the Command File PICKMENU.....	26
6. Correspondence of the File Storage System of OATS to the VAX/VMS Directory System	37
7. Example of Modification to CHOICELST.COM Command File to Add a Menu to OATS.....	52
8. Example of Command File Added as a Menu to OATS.....	52
9. Main Menu Screen	54
10. Mail Menu Screen	56
11. Reminder Menu Screen	57
12. Reminder Program Screen Example	57
13. Word Processor Menu Screen	60
14. File Menu Screen	60
15. Calculator Program Screen Example	61
16. Document Selection Screen Example.....	63
17. Format Document Menu Screen	64

CHAPTER I

INTRODUCTION

Office automation has a variety of definitions. It can be a single product, such as the Lisa Computer by Apple Computer. It can be a group of products consisting of hardware and software that can be customized to meet particular requirements, such as the Digital Equipment Corporation's VAX Computer and Professional Computers and the All-IN-1 Office Automation System that comprise their office automation system.¹ It can be a capability that allows the tying together of many different systems and programs into an easy-to-use system that is accessible from a single terminal, as is attempted by the VAX Professional Workstation program from the Digital Equipment Corporation Users' Society (DECUS).²

A computer dedicated solely to office automation is out of the realm of possibility for many, if not most, businesses or offices. A personal computer may not provide the user with the flexibility desired, such as communication with other users and integration of office automation functions with the other management and technical functions desired. The wide range of capabilities and lower cost of

¹ "ALL-IN-1 Handbook," EB-25810-52, Digital Equipment Corporation, Maynard, Mass., 1983.

² James G. Downward, "The VAX Professional Workstation," The DEC Professional, Vol. 3, No. 5, Sep. 1984, pp. 11-21.

minicomputers make this last concept of office automation a more realistic one for most users.

The office automation system presented in this paper is one which meets many of the needs mentioned above. It acts like a personal office automation computer, but because it runs on a Digital Equipment Corporation VAX Computer, it also allows integration with and access to other information and programs. It provides many of the advantages of the DEC ALL-IN-1 System but none of the additional cost. It provides the capabilities of the VAX Professional Workstation without the complexity.

General Concepts

The types of office automation systems available on today's market include electronic typewriters which can do limited word-processing and, when connected to a computer system, can serve as workstations and terminals. A step above the electronic typewriter is the word processor which can also perform the same functions as an electronic typewriter and also process records and files and perform some of the functions found in a personal computer. Other functions that can be performed by a personal computer are decision support, database and file management, administrative support, and voice telephone terminal. Executive workstations add capability by allowing communication with each other via electronic mail. At the top of the spectrum is the combined office automation/data processing system which can provide the majority of the options available in all the

other types of systems plus local networking, voice mail and electronic filing. The various types of office automation systems and functions which they usually include are shown in Table 1.

Office automation is no longer limited to use by clerk/typists. In addition to secretaries and office managers, professionals and executives are aided by efficient and useful automation because of the information-extensive nature of their jobs.

The first office automation systems handled routine office operations such as simple calculations, filing, searching and text production. The information search and retrieval then became automated. Most modern state-of-the-art office automation systems offer such functions as calendars for scheduling appointments, conferences and daily work schedules and are designed to allow sharing of documents, databases and files.

Design Considerations

There are several design considerations of importance in developing an office automation system. User-friendliness is in most cases more important than the capability of the system to do complex automation tasks. Ideally, easy-to-use software is that which gets the job done with no fuss because it stays out of the user's way by not drawing attention to itself and by offering the user only the necessary information to accomplish the desired goal. Even help messages can be too helpful. An example of a useful help menu is illustrated in Figure 1.

Table 1. Types of office automation systems.

	Electronic Typewriters	Word Processor	Personal Computers	Executive Workstations	PBX Systems	OA/Data Processing
Word Processing	limited	full	varies	limited	limited	full
Office Automation	limited	yes	yes	yes	no	yes
System Workstation	yes	yes	yes	yes	no	no
Data terminal	yes	yes	yes	yes	no	no
Records/file processing	no	yes	no	no	no	yes
Personal Computer	no	yes	yes	no	no	no
Decision support	no	no	yes	yes	no	yes
Database & file management	no	no	yes	no	no	yes
Administrative support	no	no	yes	yes	yes	yes
Electronic mail	no	no	no	yes	yes	yes
Local networking	no	no	no	no	yes	yes
Electronic filing	no	no	no	no	no	yes

Source: John Murphy, "Office Automation Systems", Mini-Micro Systems, v.XVI, n.14:p227, (December, 1983) Cahners Publishing Company.

- What kind of help do you want?
1. Where am I anyway?
 2. How do I enter data into this field?
 3. How can I make changes?
 4. I'm done here. What do I do now?

Figure 1. Example of User-friendly Help Menu

Consistency is an important part of user-friendliness. The functions of keys (such as "?" for HELP) should be the same for as much of the system as possible. The system should also use metaphors for objects with which the user is familiar (eg.- picture of a file with drawers labeled and the appropriate one opened and the file sticking up).

Customizing by adding the features desired by the user is another design consideration. The ease with which this addition (and corresponding deletion) can take place is important.

A staged user-interface architecture consists of two or more levels. Then it is possible to turn off functional layers so that the basic applications software can be run by the user by relatively simple commands, but the advanced functions can be easily engaged when requested. Physically, the stages would look alike, but the training user might get a message like "XXX not available on training system".

Hardware is a major cost factor in any system. As can be deduced from Table 1 there is quite a range between the simplest,

least expensive system and the most complex system. One choice could be a stand-alone office automation computer or network of computers such as the Apple Computer LISA system provides. Perhaps the decision would be to add to an already existing system which is used for other computing purposes, such as is possible with the addition of one or more personal computers to the Digital Equipment Corporation's VAX computer.

Training the users or potential users on the system is important, both in terms of classroom experience and tutorial training programs that are part of the system itself. This "on the job" training should be closely integrated with the "help" aspect of the system.

Software availability can be a major problem in designing a system. There is much being said today about integrated systems with regard to office automation and how integration should be handled. The debate seems to be centered around the dilemma between starting with the software desired and designing a compatible system or writing new programs that will run with existing office automation systems. At any rate, the goal should be a system with a high level of functionality that is easy to use, letting the user accomplish complicated tasks over a short period of time with a minimum of interaction with the computer.

Overview of Three Systems

Lisa by Apple Computer

Lisa, represented as a "total office system" or "executive workstation", is a dedicated office automation system designed to be an integral part of one's work. Because local networking is not available yet, it is essentially a stand-alone system. The software available for Lisa is, for the most part, outstanding, especially the spreadsheet calculator, graphics packages, and project manager. On the other hand, the database manager is not a true database management system, but simply a list manager with editing capabilities. There are also shortcomings in the word processing package such as no spelling checker, footnoting or indexing capabilities, and complex and difficult editing and scrolling. Apple's attempt to maintain consistency between all of Lisa's tools has created difficulty in using some of the tools. Other disadvantages are the slow speed of the machine and the limited transportability of documents or parts of documents between tools.

ALL-IN-1 by Digital Equipment Corporation

ALL-IN-1 integrates office functions with the power of the VAX/VMS computer system, utilizing a selection of workstations -- video terminals, printing terminals, and the three Digital personal computers. ALL-IN-1 offers all the standard office automation software as well as the capability of adding other applications to the

menu. The software is offered in three packages: Office Menu, Basic Software Package, and the Extended Software Package. However, in order for the Office Menu package to give the capability to add customized applications, the Forms Managment System (FMS) is a prerequisite. VAX/VMS command file parameter entry also requires FMS. However, FMS is included in the Extended package. Many features of the ALL-IN-1 system are already available on most VAX systems. Additional software products are available for graphics.

The VAX Professional Workstation

The VAX Professional Workstation is available from Digital Equipment Corporation User's Society (DECUS). It was written at KMS Fusion, Inc., Ann Arbor, Michigan. It is an outgrowth of an effort to provide both staff and support personnel easy menu-driven access to the growing number of VAX productivity tools. The Program is written in Digital Command Language (DCL) and was designed to work on VT100 terminals with advanced video option but can also be used from non-DEC word-processing terminals. Figure 2 shows the menus and functions available to the user. Because of the technical environment under which the system was written, the main emphasis is toward technical productivity tools and data analysis facilities. However, the existence of document preparation, filing, mail, calculator and reminder functions classifies it as an office automation system.

The VAX Professional Workstation evolved over the course of a year rather than having a solid design before any programming was done. It is a very powerful tool, but also tends to become

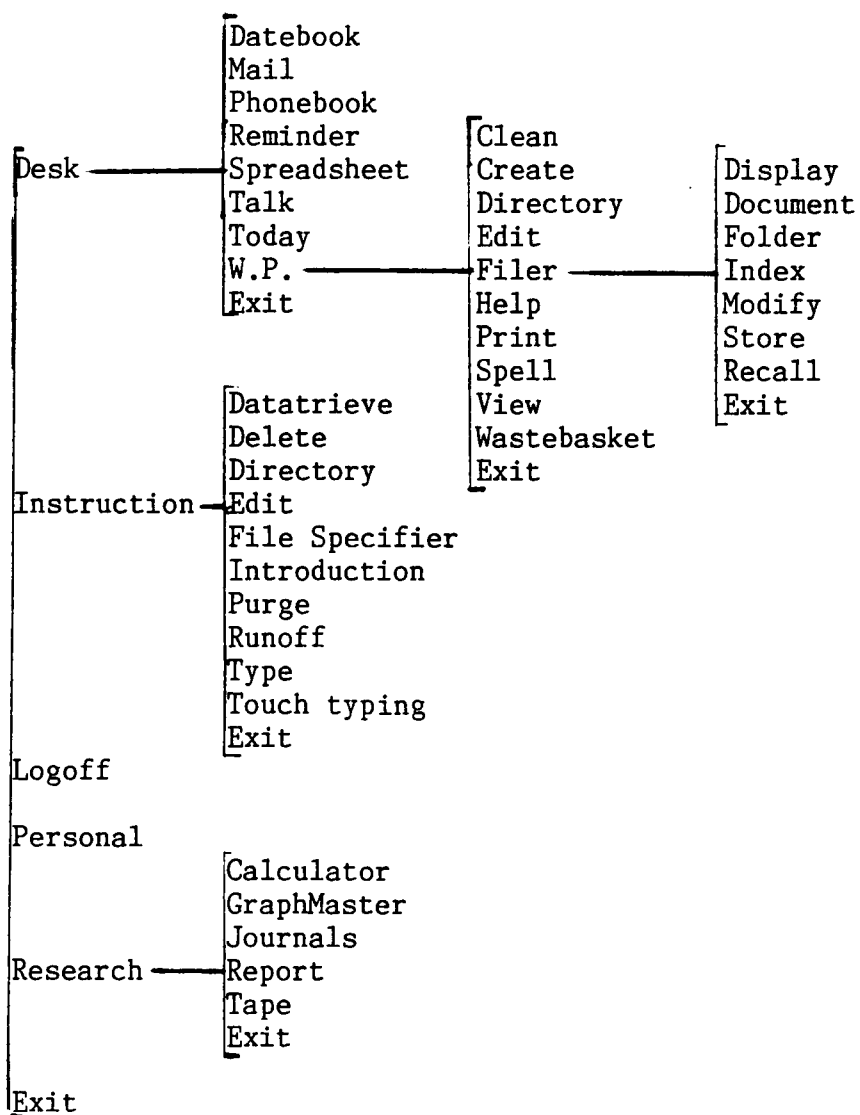


Figure 2. Functions available with VAX/VMS Professional Workstation.

complicated and overwhelming to a novice user. There are several menu choices which appear to do the same thing but may or may not be different. Some menus offer as many as eleven choices. The manual on using the desk calculator is twenty pages long. The manual for the entire system is over one hundred pages. This does not conform with the desirable criterion of ease of use.

OATS -- OFFICE AUTOMATION TOOL SYSTEM

Office Automation Tool System (OATS), the system presented here is an alternative that affords many of the advantages of the three other systems described with few of the drawbacks. For example, OATS provides the personal computer feel of the Lisa computer, the menu-driven architecture of the other two systems for the VAX computer, and it does not require additional software or hardware beyond the basic VAX computer and a VT100-compatible terminal.

Like Digital's ALL-IN-1, OATS is designed to run on the VAX/VMS computer. The only other requirement is a VT100 compatible terminal. Like VAX Professional Workstation, OATS is written entirely in DCL (Digital Command Language) with the exception of a Macro Assembly Language program to receive the input commands for menu selection. One of its chief advantages is the ease by which any software available for use with VAX/VMS can be integrated into OATS. Because OATS is written in DCL, any program can be added to the OATS menu by simply creating a command file (written in DCL) to set up parameters for the running of the program, begin execution of the

program, and return control to the desired place when execution is completed. OATS makes use of standard programs such as DECmail, EDT (text editor), RUNOFF (document formatter) and the hierarchical directory system of VAX/VMS without resorting to expensive additional software to accomplish the same purposes.

Speed is another advantage with OATS and it shares this advantage with the VAX Professional Workstation. Although command procedures are less efficient than executable images at some things such as putting menus on the screen, the VAX can handle the minimal delay. The overhead incurred is low, memory use is low, and start-up time (the time it takes to get a command started) is very rapid.

Because the capability of the system to do complex automation tasks is less important than easy-to-use software that gets the job done, one of the main goals of OATS is user-friendliness. Menus are relatively short (five to seven items), the menu screens are uncluttered, and documentation is not unnecessarily extensive. Essentially, the tutorial for OATS is an expansion of the help messages displayed on the screen for the user when he asks for assistance.

OATS features a two-level staged user-interface architecture. When an OATS user authorized to use VAX/VMS chooses the QUIT option from the OATS main menu, control is passed to the command level from which OATS was invoked. A user authorized to use the VAX only in OATS is logged off the system when he chooses the QUIT option from the main menu.

Adding software is an easy task in OATS. All that is required is modification of one line in the command file of menu symbols which is kept in the user's directory and creation of a command file to run the program. Entire menus can be added as well and almost as easily as programs.

While the choice of software for the VAX may not be greater than the choice for Lisa it is certainly as good. The transportability of data files depends on the software selected. With careful evaluation of program packages, this should not be a problem.

CHAPTER II

DESCRIPTION OF OATS

Capabilities of OATS

OATS has the functions and capabilities of most office automation systems as well as the normal functions of the VAX/VMS data processing system. Table 2 shows these functions.

Table 2. Functions and the methods that have been used to achieve those functions in OATS.

FUNCTION	METHOD USED
1. Word Processing	EDT, RUNOFF
2. File Managment	DCL
3. Calculator	C program
4. Mail	DECmail, DCL
5. Reminder messages	DCL
6. Additional support	DCL, other languages

OATS is designed to run on Digital Equipment's VAX/VMS Computer. The only other requirement is a VT100 compatible terminal. Because OATS is written in DCL (Digital Command Language), any program that runs on the VAX can be added to the OATS menu simply by creating a DCL command file to set up parameters for the running of the program, begin execution of the program, and return control to the desired place when execution is completed.

Detailed information on each of these functions and how they are implemented is given in Chapter IV.

Invoking OATS

Three commands need to be carried out to put the user in OATS. For convenience, they should be put in a file called LOGIN.COM in the user's primary directory. The commands are:

```
OATS_DIRECTORY := "[****]"
VMS_PRIV := "NO"
SET DEFAULT [user.OATS]
@'OATS_DIRECTORY'OFFICE
```

where "user" is the user's name and "[****]" is the directory where the OATS files reside. These commands will automatically cause the user to begin OATS when he logs into the VAX. When the user exits from the main menu of OATS, he will be logged off the VAX also.

If the user wants to be able to use VAX/VMS outside of OATS, the above commands could be put into a command file in the user's directory and executed by him when he wanted to use OATS. However, the value of VMS_PRIV should be "YES" instead of "NO". Also useful would be a command to put the user at the same directory level every time OATS is run.

Menu Selection

OATS utilizes programs and utilities normally found on a VAX/VMS Computer to accomplish the some of the fundamental tasks of office automation with set-up done by command files to make the running of the individual programs transparent to the user. To the user, it appears to simply be a part of the overall system. The

selection of these appropriate command files to accomplish the user's purposes is done through a command file called CHOICELST.COM which defines the content of each menu.

Architecture

This command file is part of OATS and resides in the same directory as the rest of the OATS files. However, if the user has a file of the same name in his directory, that one is the one used by OATS. This allows for customization of OATS by the user for his particular needs. This file has a numbered entry for each menu and program invoked by OATS.

How Menu Selection Is Done

The command file CHOICELST.COM is a collection of symbols, each with the name in the form of PICKnn, where "nn" is a two digit number. The set of PICKnn symbols shown in Figure 3 actually forms a doubly-linked list which determines the menu and program selections. Each PICKnn symbol is a node in the list with "nn" being the node number. The parent node of each symbol definition is the node number of the menu to be returned to on exit from the current menu. The children of each node are the node numbers of the menu or program options to be displayed. All non-terminal nodes are menus; all terminal nodes are programs. The first character of each symbol indicates if it is a terminal ("P") or non-terminal ("M") node. Tables 3 and 4 show how the strings for each type of node are defined and used.

```

PICK01 := "M04MAIN000703020906052549"

PICK03 := "M04MAIL0106081407202549"

PICK08 := "P14READ/SEND MAIL0308MAILPROG"
PICK14 := "P16OFFICE MAIL LIST0308MAILPROG"
PICK07 := "P18EXTERNAL MAIL LIST0308MAILPROG"
PICK20 := "P09SEND MEM00308MAILPROG"

PICK02 := "M09REMINDERS01052204212549"

PICK22 := "P04LOOK0208REMINDER"
PICK04 := "P03ADD0208REMINDER"
PICK21 := "P05ERASE0208REMINDER"

PICK09 := "M04FILE010713151719232549"

PICK13 := "P04VIEW0907FILEDOC"
PICK15 := "P05PRINT0907FILEDOC"
PICK17 := "P04MOVE0907FILEDOC"
PICK19 := "P09DUPLICATE0907FILEDOC"
PICK23 := "P10THROW AWAY0907FILEDOC"

PICK06 := "P10CALCULATOR0104CALC"

PICK05 := "M14WORD PROCESSOR010424262549"

PICK24 := "P12NEW DOCUMENT0508WORDPROC"
PICK26 := "P12OLD DOCUMENT0508WORDPROC"

PICK10 := "M16FORMATTING STYLE0506111216182549"

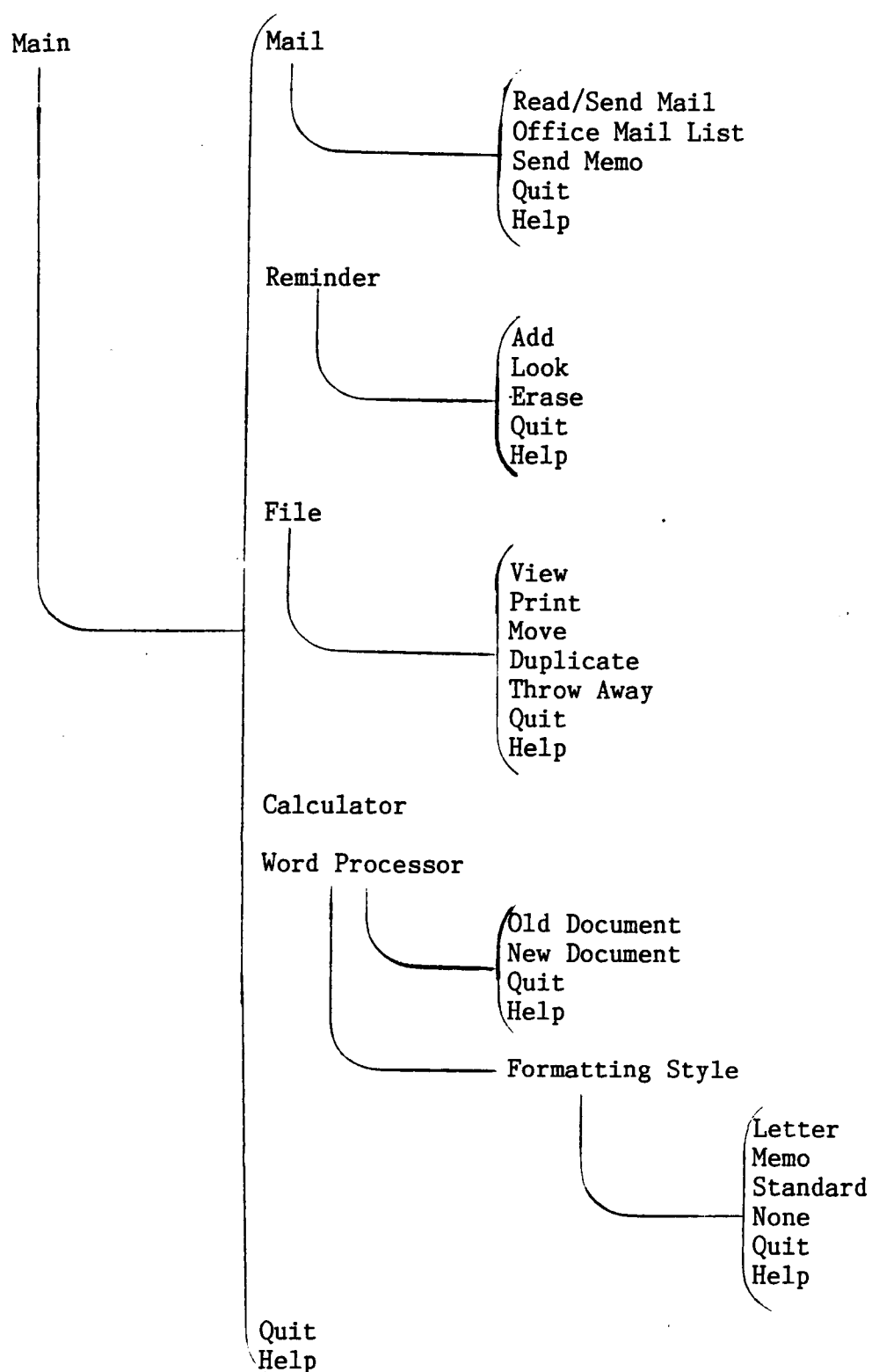
PICK11 := "P06LETTER0508WORDPROC"
PICK12 := "P04MEMO0508WORDPROC"
PICK16 := "P08STANDARD0508WORDPROC"
PICK18 := "P04NONE0508WORDPROC"

PICK25 := "P04QUIT0004QUIT"
PICK49 := "P04HELPO007HELPMMSG"

```

(A) Symbol definitions for PICKnn as defined in the command file CHOICELST.COM. These definitions determine the flow of control.

Figure 3. Flow of control through the menus of OATS.



(B) Menus of OATS as defined by the command file CHOICELST.COM.

Figure 3. (continued)

Table 3. Format for DCL assignment statement for a menu option.

PICK_{nn} ::= "MmmNAMExxppaabbcc...zz"
 where

nn is the node number of the menu
mm is the number of characters in the menu name
NAME is the name of the menu shown on the screen
xx is the exit node number
pp is the node number of options in the menu
aa,bb,cc,...,zz are the menu or program numbers of the
 options.

Table 4. Format for DCL assignment statement for a program option.

PICK_{nn} ::= "PmmNAMExxqqCNAME"
 where

nn is the node number of the program option
mm is the number of characters in the option name
NAME is the name of the program option
xx is the exit node number
qq is the number of characters in the command file name
CNAME is the command-file name.

CHOICE_NUMBER is the symbol that determines what is shown on the screen in OATS. Initially, the value of CHOICE_NUMBER is "01", which is the node number of the main menu, corresponding to the symbol PICK01 which is defined as (spaces added for clarity):

"M 04 MAIN 00 07 03 02 09 06 05 25 99"

The word "MAIN" taken from the symbol definition is displayed on the screen as the menu title. The "00" does not actually point to a defined mode, but will cause control to be returned to the next highest level of DCL and terminate execution of OATS if the user types Control-Y or chooses "Quit" from the main menu. A user unauthorized to run under anything other than OATS will be logged off the system. There are seven child nodes of which "03" is the first. Therefore, PICK03 is the first of the seven options to be listed in the menu. The names of these options are part of the PICK_{nn} symbol definition of each option or node. If it is selected, the MAIL menu will be displayed, because the node PICK03 is assigned the value

"M 04 MAIL 01 06 08 14 20 07 25 99".

The MAIL menu, as can be seen, has six options which will be listed on its menu screen. Each menu in OATS has the options of HELP and QUIT. Choosing HELP displays a brief description of each option in the menu. Control is then returned to that menu. If QUIT is chosen, control is returned to the next highest menu level through the parent node. This hierarchy is illustrated in Figure 3.

One of the options of the MAIN menu is defined in the symbol PICK06. Instead of a menu, this PICK_{nn} symbol names a DCL command file to which control will be given if that option is chosen. PICK02 is assigned the value

"P 12 CALCULATOR 01 04 CALC".

As in a menu option, "CALCULATOR" will be what is listed in the menu as an option. A DCL command file called CALC.COM will be given control if that option is chosen. That command file causes a program that performs the function of a calculator to be run and, as specified by the "01" parent node in the definition, control will return to OATS by bringing the menu named in the symbol PICK01, or the MAIN menu.

In several cases, a given command file is included in the assignment statement of several symbols. For example, PICK04, PICK21, and PICK22 all specify passing control to "REMINDER.COM". When the command file is called up, the P1 parameter passed to it is the symbol CHOICE_NUMBER, in this case, "04", "21", or "22". This causes a branch to the appropriate commands in the command file.

Major Command Files

OATS is constructed of more than twenty separate command files. The flow of control through these various command files is shown in Figure 4.

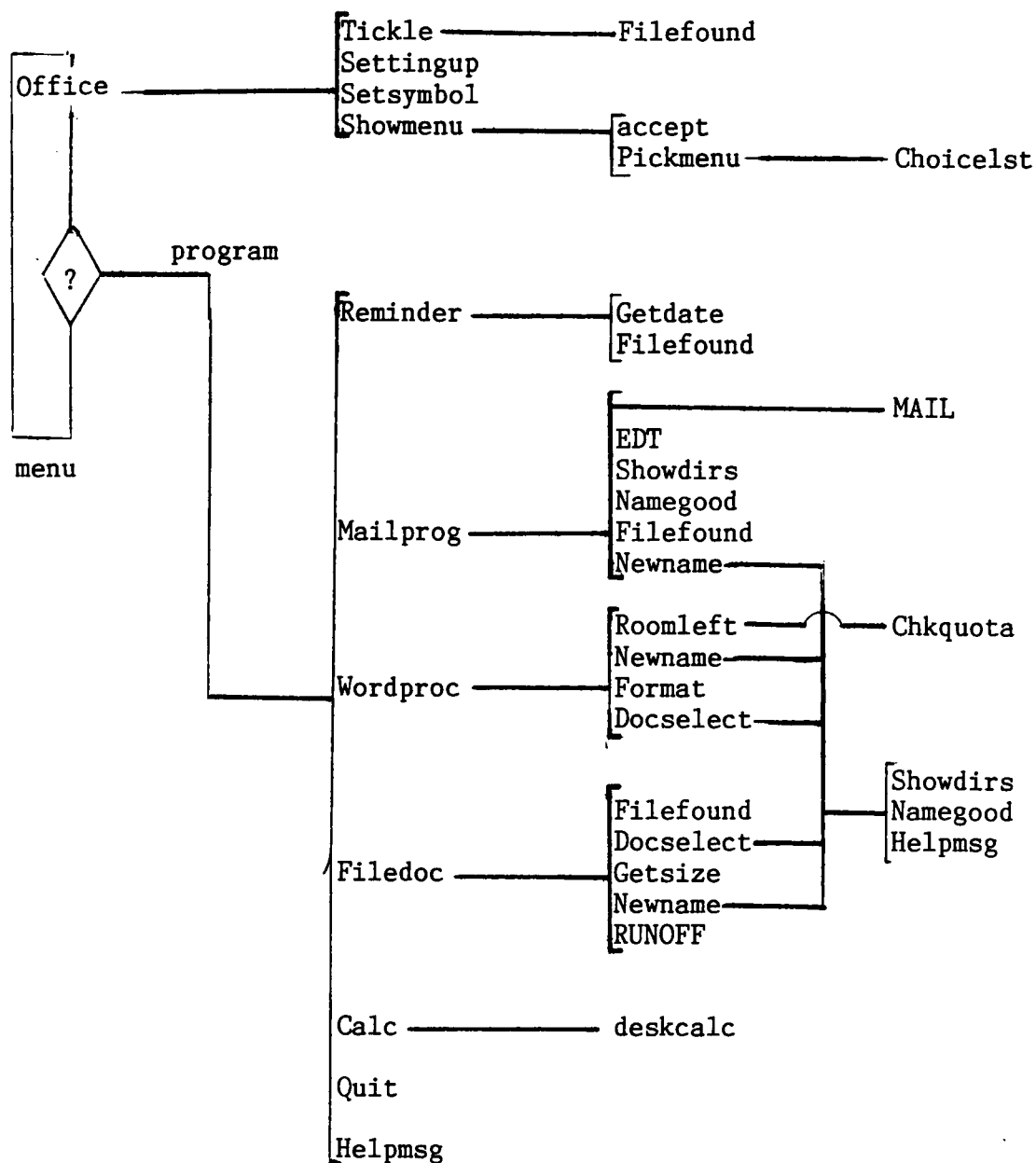


Figure 4. Flow of control through OATS command files.

OFFICE - the Main Program of OATS

OFFICE.COM is the command file that is the basic control for OATS. It controls the flow of the system by determining what the user has requested -- a program or another menu -- and responding accordingly. It calls SETTINGUP, the command file that initializes symbols for escape sequences used with the VT100 terminal. Some useful global symbols for executing DCL commands are then assigned values in the command file SETSYMBOL. Prior to showing any menu, OFFICE checks the file REMINDER.DTA in the user's directory and displays any reminders that exist for the current day. If there are reminders, the user clears the screen by pressing the RETURN key and the command file continues. Then global symbols for document selection and menu selection are defined and initialized. An OATS utility command file called FILEFOUND checks to see if the user has his own CHOICELST.COM file and executes it to define the PICKnn symbols for the user. If the user has no such file, the OATS CHOICELST.COM command file is run to initialize the menu and program choices. Initially, PICKNN is defined as the main menu (PICK01) and the command file SHOWMENU is called, returning a new value for PICKNN as chosen by the user. OFFICE decides if it is a menu or program by testing the value of the symbol M_OR_P. If it is a menu, then SHOWMENU is called again. If it is a program (another command file), the title of the command file is extracted from the PICKNN string and the command file is run via the "@" command, with the P1 argument being the symbol CHOICE_NUMBER (the

"nn" value that was chosen by the user). On return to OFFICE, the EXIT_MENU_NUMBER is checked and the command file loops back to display the appropriate menu (or leave OATS if EXIT_MENU_NUMBER is "00").

TICKLE. This command file is run when OATS is initially started. It checks for the existence of the file REMINDER.DTA in the user's directory. If the file does not exist, TICKLE exits. If the file does exist, it is read, and a search is made for the current date. If the date is found, it displays on the screen the contents of the message(s) for the current date. The user must then press the RETURN key to clear the screen and exit from TICKLE. The format of the file REMINDER.DTA is found under the description of the command file REMINDER.

SETTINGUP. Various symbol values for the VT100 terminal are defined in this command file. Escape sequences that handle modes to set and reset cursor control, auxillary keypad control, screen erasing, line size, character attributes, cursor movement are the most frequently used and make programming much easier.

SETSYMBOL. There are global symbols used in OATS that need to be defined prior to any use. Also, a symbol table for the months of the year and the number of days in each month is in this command file. A few other useful symbols are defined in this command file and are listed in Table 5.

SHOWMENU. This command file first checks to see if a menu is already displayed on the screen. If not, it erases the entire screen and writes the OATS logo at the top of the screen. If a menu was

Table 5. Global symbols used in OATS and available to command files added to OATS.

Symbol	Function
SAY	Same as "WRITE SYS\$OUTPUT" (writes to screen)
Character attributes (controls the appearance of characters sent to the screen):	
ATTRIBUTES_OFF	Turns off all character attributes.
BOLD	Turns on bolding of characters.
UNDERSCORE	Characters are underscored.
BLINK	Characters blink.
REVERSE	Reverse video.
Line size commands (affects entire line):	
DOUBLE_HEIGHT_TOP	Top half of line of text. 40 characters/line.
DOUBLE_HEIGHT_BOTTOM	Bottom half of line of text. 40 characters/line.
SINGLE	Returns to normal size. 80 characters/line.
DOUBLE_WIDTH	Double width (40 columns). 40 characters/line.
Cursor movement:	
MOVE," <u>l</u> ; <u>c</u> ",DIRECT	Positions cursor at line <u>l</u> , column <u>c</u> .
MOVE,UP	Move cursor up 1 line, same column.
MOVE,DOWN	Move cursor down 1 line, same column.
MOVE,RIGHT	Move cursor right 1 column, same line.
MOVE,LEFT	Move cursor left 1 column, same line.
INDEX_UP	Same as MOVE,UP but will scroll if at top.
INDEX_DOWN	Same as MOVE,DOWN but will scroll if at bottom.
SEE_CURSOR	Cursor is visible.
HIDE_CURSOR	Cursor not visible.
SAVE_CURSOR	Save the cursor position.
RESTORE_CURSOR	Restore previously save cursor position.
MOVE," <u>t</u> ; <u>b</u> "SCROLL	Set top (<u>t</u>) and bottom (<u>b</u>) of scrolling region. (<u>t</u> - <u>b</u> >3)

Table 5 (continued)

Symbol	Function
Symbols for clearing all or part of the screen:	
CLEAR_SCREEN	Clears entire screen. Cursor Home (line 1, column 1).
CLEAR_TO_CURSOR	Clear screen from Home to cursor.
CLEAR_TO_EOL	Clear line from cursor to end of current line.
CLEAR_LINE_TO_CURSOR	Clear line from column 1 to cursor.
CLEAR_LINE	Clear entire current line.
CLEAR_TO_EOSCREEN	Clear screen from cursor to end of the screen.
Symbols used in document selection:	
WHICHLAST	Last document accessed was "OLD", "NEW", or "" (none).
NAMELINE	Complete specification for old file selected.
NEWLINE	Complete specification for new file selected.
Symbols used in menu and program selection:	
TITLE	Name of menu or program selected from menu.
EXIT_MENU_NUMBER	Name of menu to return to on EXIT or Control-Y.
CHOICE_NUMBER	Two-digit "nn" number for PICKnn symbol currently using.
PICKNN	Actual PICKnn symbol (e.g. PICK01 - main menu).
OLD_TITLE	Name of current menu - used by HELPMMSG chose message.

already on the screen, only the menu portion of the screen is cleared. This saves some time, as output to the screen is a slow task in DCL.

The command file PICKMENU is then called with the P1 parameter of CHOICE_NUMBER (the node number of the symbol that defines the menu). PICKMENU returns the title of the menu (TITLE), the length of the title (TITLE_LENGTH), the number of options in the menu (NUM_ENTRIES), and the appropriate values for each option in the symbols CHOICE_n_NUM and CHOICE_n where "n" is the number 1 through the number of options. The symbol CHOICE_n_NUM is the two digit node number identifying the PICKnn symbol. This process is illustrated in Figure 5.

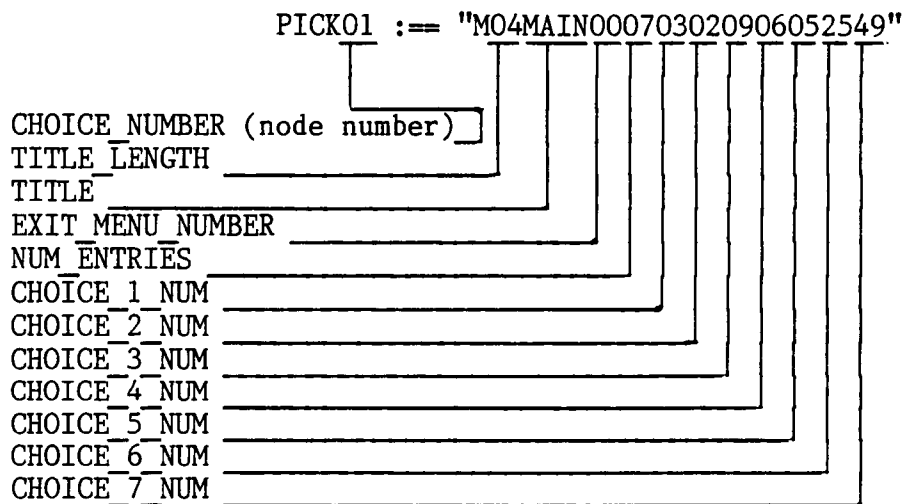


Figure 5. Global symbol values extracted by the command file PICKMENU.

Then PICKMENU goes through the list of menu options and evaluates symbols for each option in the following manner:

```
SYM := PICK'CHOICE_1_NUM'
SYM1 := '$SYM'
LEN = 'F$EXTRACT(1,2,SYM1)'
CHOICE_1 := 'F$EXTRACT(3,LEN,SYM1)'
```

which will result in the following values:

```
SYM := PICK03
SYM1 := "MO4MAIL0106081407202549"
LEN = 4
CHOICE_1 := "MAIL"
```

These commands are performed for each option in the menu and the values of the global symbols determined. The value of the CHOICE_n and CHOICE_n_NUM symbols for n greater than the number of options in the menu is the null string.

On return from PICKMENU, the value of each non-null CHOICE_n is displayed on the screen with the first two characters bolded.

The assembly language program ACCEPT is run to accept two alphabetic characters of input. The characters input are returned using the \$EXIT utility which returns one longword in the System Symbol \$STATUS. The error handler portion of this command file checks for an error code being returned. If it was (the global System Symbol \$STATUS less than or equal to 10), the user is prompted again for his choice. Three erroneous type-ins are allowed, after which the help message appears giving a brief description of the the options on the menu. A Control-Y is treated the same as if the user had typed "QU" for the "QUIT" option.

If an error code is not returned, a scan through the appropriate responses is made. If the response was not one of the valid choices for this menu, the situation is handled as the error return described above. If a match is found for the two characters typed in, the symbol CHOICE_NUMBER is set to the correct value, the symbol M_OR_P (the first character of the PICKnn string) is set, PICKNN is set equivalent to the new PICKnn symbol where "nn" is the node number of the menu or program chosen. The command file SHOWMENU then exits.

REMINDER

This command file allows the user to maintain a file of messages to himself. These messages (up to 70 characters in length) are stored in the file REMINDER.DTA in the user's directory. The messages are stored chronologically according to date. The date in the format dd-mmm-yyyy is stored at the beginning of each message with the text stored in the same line after the date. A carriage return is at the end of the message text. Reminders cannot be entered for days already passes. Likewise, all messages prior to the current day are automacally deleted whenever the user selects "Add" or "Erase" from the menu. Because "Look" does not write the file back out, no editing is done.

Each of the three menu choices have to check for the existence of the file, read the file until the date is found or passed, and display the message(s) (if there are any) for the date requested on the screen. Several "IF ... THEN" statements are used when the

functions of the three commands are different but the rest of the code is shared.

First, the screen is cleared and the existence of REMINDER.DTA is checked and it is created if it does not exist. The current date is obtained and put in the format "yyymmdd" where "yy" is the last two digits of the year, "mm" is the month of the year (numerically), and "dd" is the date. This gives a format that can be compared with other numbers of the same format to determine which date occurs first. The three symbols that use this format are TODAY_NUMBER (the current date), NEWDATE_NUMBER (the date the user is inquiring about), and READ_NUMBER (the date of the message just read from the file).

The symbol WHY_HERE is either "ADD", "ERASE", or "LOOK" and is determined by the value of the P1 parameter. The REMINDER.DTA is then read one line at a time, the dates compared, and the line rewritten in the file if it is not obsolete. As stated before, "Look" does not rewrite the file as it only reads it. If the correct date is found, the message(s) are displayed on the screen until the date read from the file is past the date requested. At this point, "Add" asks for the new message, writes in in the output file, copies the remainder of the file and exits. "Look" simply exits. "Erase" prompts the user for the number of the message to delete (they are numbered as they are displayed on the screen), and writes all the other remaining messages to the output file. "Erase" does not write the messages for the day selected in the output file, but writes them in a temporary file. After it is determined which message to leave out, the messages for

the day are read in from the temporary file and are written out to the output file except for the deleted message. Two useful utilities used by REMINDER are GETDATE and FILEFOUND. These are described under "Utilities".

MAILPROG

This command file attempts to assist in the use of the MAIL program on the VAX. When the user requests the "Read/Send Mail" option, a list of the most frequently used MAIL commands is displayed at the bottom of the screen and the MAIL program is run. As soon as the user does something in MAIL the list at the bottom of the screen is erased. All the options of MAIL are available at that point even though they are not listed at the bottom of the screen. When the user leaves MAIL with either the "exit" or "quit" command, the OATS mail menu is then put back up on the screen.

One of the features of the MAIL program is the ability to send mail to a list of users by typing in a command such as "@listname" in response to the prompt "TO:". This list can be created using any one of the text editors on the VAX. But OATS makes it easier for the user with the "Office Mail List" selection in the mail menu. When this is chosen, a command file called SHOWDIRS displays the lists he has already created (files in the user's directory with the file-type identifier "DIS") and asks the user to select one. If he chooses one that does not exist, the user is so informed and asked if he wishes to create a new one. If the response is "no", he is again prompted for the name. If it is "yes" or if an old name was chosen, EDT is entered

and the file may be edited. Upon exit, the mail menu is again displayed.

"Send Memo" uses that feature of MAIL which allows the user to send a file as mail instead of typing in the message in mail. This allows for correction of mistakes and in OATS allows the user to file memos using the OATS filing system. The user is prompted "Memo to:" and the response can either be a name, list of names, or file which contains a list of names (the filename is preceded by "@"). The user is then prompted "Subject:". These two responses are written into the memo, EDT is run, and the user can edit the message. It is written to a temporary file. The message is then sent using MAIL by using the response to "TO:" and "Subject:" previously mentioned to send the message. If the user wishes to file the memo (answers yes to prompt), he is prompted to select the place to file the document and the temporary file is so renamed. The name of the command file that allows selection of the name of a new document is "NEWNAME". If the file is not to be saved, the temporary file is deleted. Exiting returns control to the mail menu.

WORDPROC

This portion of OATS serves as a word processor by utilizing the file structure of OATS, the EDT editor, and the DSR/RUNOFF text-formatting program. This command file is unique in OATS in that it is entered twice in the course of accomplishing its task. The first time is for the purpose of selecting the program to be edited or

created, then, after exiting to the formatting menu, for the actual editing of the file.

In order to assure that there is sufficient room left in the user's quota of allotted disk space for the task ahead, a check is made to see that 100 blocks are available. If there is not, a message is displayed and the user is returned to the main menu. If there is enough space, a check is made of the P1 parameter. On this first entrance to the command file, the choice is either an old document or a new document. If new, a new document name is selected by the user (NEWNAME command file). If old, DOCSELECT is the command file used. Exit is made forcing the formatting style menu to be displayed by equating the global symbol EXIT_MENU_NUMBER to "10", the PICKnn symbol defining the formatting menu.

When WORDPROC is re-entered, the possible choices are letter-style, memo-style, standard-style, or no formatting. Memo- and letter-style prompt for "To Whom:", "From Whom", "Subject", and "Salutation name". Then the FORMAT command file is called which writes these values into the file name specified, using RUNOFF commands to create the appropriate formatting style. The user can customize these forms by copying the OATS file FORMAT.COM and substituting the appropriate address and name of the sender. WORDPROC checks for the existence of the user's file first. The current date is inserted automatically.

The file is then closed and set up to run with EDT, at which time any part of the document may be changed. Before EDT is called to run, the following assignment is made:

ASSIGN/USER TT: SYS\$INPUT

to allow proper running of EDT. Control-Y is also inhibited.

At the end of the edit, the file is purged with only two copies being kept and control is returned to the Word Processor menu.

FILEDOC

The file maintenance, typing files on the screen, and printing files for OATS is done by the command file FILEDOC. The function performed is dependent on the calling P1 parameter as set up by the menu choices. The value of P1 determines the branch taken in the beginning of the command file. In all branches of FILEDOC, the word "FILE" is displayed on the bottom right corner of the screen in reverse video, and type of operation being performed as well as the document name(s) displayed on the lower left corner of the screen.

The Control-Y exit routine from this command file must first clean up any partially finished document selection that may have been done. The global symbol WHICH_LAST which might be used elsewhere in referring to the last file accessed is assigned the null string.

View a Document. DOCSELECT is called and the name of the document displayed on the bottom of the screen. Next the first line of the file is read to determine if it is an unformatted RUNOFF file. If it is, the first character will be a period, the indication that it is a RUNOFF command. If it is, RUNOFF is run, outputting the

formatted document into a temporary file. Non-RUNOFF files are displayed as is. Only the top twenty lines of the screen are used to display the document. The user must press RETURN to see the next twenty lines of the document as well as when the viewing is done. This method was used rather than the TYPE command of VAX/VMS or using the terminal as the output for RUNOFF so the user would not have to be concerned with the Scroll/No Scroll key.

Print a Document. This works identically to viewing the document for document selection and formatting. The file is then sent to the printer using the VAX/VMS PRINT command. For both print and view in the case of a RUNOFF file, a message is displayed at the bottom of the screen while the formatting is being done so the user does not become impatient.

Move a Document. First the old document name is asked for, then the new document name. The VAX/VMS RENAME command is used on these document names. Then a check is made to determine if this operation has left any empty directories. Starting with the lowest subdirectory, FILEFOUND is used to see if any files exist. If none do, then the protection of the directory is changed and it is deleted. If at any point up the chain of directories files are found to exist, the deletion process is stopped. During this process the reporting of error messages is suppressed.

Duplicating a Document. After selection of the document to be duplicated, GETSIZE and ROOMLEFT are executed to get the size of the file and make sure there is enough space left to copy it. If there is

not enough room, the user is notified and the command file exits. If there is room, the new document specification is taken and both names are displayed on the bottom of the screen. The VAX/VMS COPY is run and the new document name becomes the last file accessed. The user is notified that the Document is copied and the command file exits.

Throw a Document Away. After displaying the name of the file being deleted, the user is asked "ARE YOU SURE?". If the response is positive, the document is deleted and clean up of any empty directories proceeds as in the "Move" option. The user is informed that the document is thrown away. If the user does not reply "Y" to the prompt, the command file exits without doing anything to the file.

CALC

The screen is divided horizontally at line 15 and below the line the directions for using the calculator program are displayed. The first fifteen lines of the screen are scrolled and that is where the user's calculations will appear. This way the last seven calculations he performed will be displayed for him. The calculator program is a very simple one allowing for whole number operations only. When the user presses RETURN twice in succession, the program ends, the screen is cleared, and the command file exits.

QUIT

This command file is called whenever "Quit" is chosen from any menu. Because it is always called from a menu, it sets the symbol HEADER to "on" so a new header will not be displayed. The symbol

PICKNN is reassigned the value of the symbol identified by EXIT_MENU_NUMBER and the value of EXIT_MENU_NUMBER is extracted from PICKnn (the parent node). If the exit is to be taken from the main menu, a check is made of the the symbol VMS_PRIV and the user is logged off the VAX if no such priveleges are granted. Users authorized to use VAX/VMS simply exit the command file OFFICE.COM and are at the level of VMS they were in before accessing OATS.

HELPMMSG

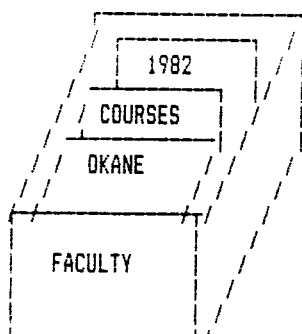
This command file displays the help message needed according to the value of the symbol OLD_TITLE. The text of the messages is part of the command file and is put on the screen via the command

TYPE SYS\$INPUT

followed by the text to be displayed. Following the displaying of the help message, the prompt "Press RETURN to get back to menu" is given in bold characters and then the program exits.

File Storage System

The OATS file storage system makes use of the directory hierarchy in VAX/VMS. Whereas subdirectories in VMS can contain data files as well as directories of other subdirectories below it in the hierarchy, OATS subdirectories only contain directories until the level containing all the data files is reached. Figure 6 illustrates the correspondence between the directories, subdirectories, and data files in VMS and the drawers, files, folders, and documents in OATS.



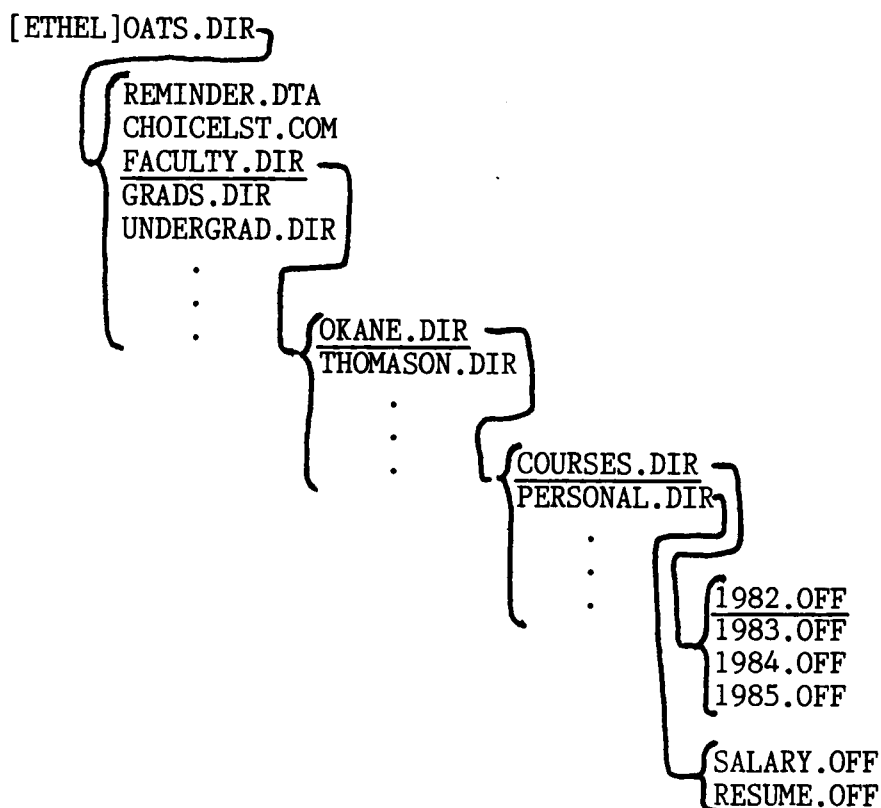
Default directory [ETHEL.OATS]

Drawer name - FACULTY
 File name - OKANE
 Folder name - COURSES
 Contents name - 1982

(A) Representation of document name in OATS.

[ETHEL.OATS.FACULTY.OKANE.COURSES]1982.OFF

(B) Actual VAX/VMS file name.



(C) Representation of VAX/VMS directory hierarchy.

Figure 6. Correspondence of the file storage system of OATS to the VAX/VMS directory system.

Drawers, files, and folders are subdirectories that contain only subdirectory files. The contents are the lowest level and contain only data files. To the user, this appears like the familiar file system of file cabinets containing drawers. Each drawer has its own label, and within that drawer, subdivisions occur (files). Each file might have several folders. Each folder would contain one or more documents.

There are two command files in OATS which are used to select documents to be processed. They are called from the FILEDOC or the WORDPROC command files. DOCSELECT is the command file that asks the user to specify the name (directory path) of a document that already exists. The NEWNAME command file asks for the user to name a place to file a document that he is creating either by WORDPROC (new document) or FILEDOC (move or duplicate). One other aspect of the document selection process that needs to be mentioned before a detailed description of the command files involved is the concept of last file accessed. WHICHLAST is a global symbol whose value is either the null string, "last" or "new". It indicates whether or not there is a file last accessed that still exists. The initial condition upon entering OATS, when the last file accessed has been deleted, or when a file specification has been aborted mid-stream is no last file accessed, i.e. the null string. If WHICHLAST is "last" then the symbol NAMELINE is equivalent to the directory path filename of the document. If WHICHLAST is "new" then the symbol NEWLINE is equivalent to the directory path filename.

DOCSELECT

When DOCSELECT is called, a picture appears on the left half of the screen that illustrates the drawer, file, folder, document concept. This picture uses the special graphics character set of the VT100 terminal. Then a list of all drawers (i.e. all subdirectories existing at this current directory level) is put on the right half of the screen by using the DIRECTORY command to list all the subdirectories as this level in a file looked at by the command file SHOWDIRS with the P1 argument ".DIR."

If there was a previous document accessed, the directory path names are displayed in reverse video in the appropriate place in the picture on the left half of the screen. If there was no previous document, the places for the name are left blank, but still reverse video so that the place for the name is obvious. The user is then prompted for the drawer name. If the user presses RETURN and does not type in any name, the previously accessed file is the one selected and the command file is exited after WHICHLAST is made equivalent to "last". If "quit" was typed in, the command file exits with an error return of "3" to indicate an abort. If "help" was typed in, the HELPMMSG command file is called after OLD_TITLE is equivalenced to "document". If something else was entered, a check is made for the validity of the name by the command file NAMEGOOD. If the name is not good, the user is prompted for it again. If it is good, the drawer name is filled in in the picture and the symbol NAMELINE is

constructed using the current directory and the name of the subdirectory (drawer) selected. The same operation is performed for file and folder levels, each time concatenating the proper information for the directory path onto the symbol NAMELINE. At the document level, the filename chosen is concatenated to NAMELINE followed by the file type identifier ".OFF". The command file exits with WHICHLAST equivalent to "last" and the document specification available in the symbols NAMELINE (the directory path), LAST_DRAWER (the drawer name), LAST_FILE (the file name), LAST_FOLDER (the folder name), and LAST_CONTENTS (the actual file name).

NEWNAME

This command file works very similarly to DOCSELECT with the directory path being made equivalent to the symbol NEWLINE and the drawer, file, folder, and contents names being the symbols "NEW_DRAWER", etc. The symbol WHICHLAST is then made equivalent to the string "new".

Last document accessed. When the user responds to the prompt for the drawer name with the RETURN key, instead of using the last document accessed, it only selects the last drawer accessed and goes on to prompt for the file selection. The same is true for the file and folder specifications. It would have been possible to have selected all three (drawer, file and folder) with the RETURN response as DOCSELECT does, but because of the circumstances under which this routine will be used, it seemed more efficient to do it this way. A user would probably not want to duplicate a document or copy a

document into the same folder, but into a different one, perhaps in the same drawer or file. Since this would be the most normal occurrence, this method was decided upon.

New names. When a new name for a drawer, file, folder, or contents is entered, the length of the name is checked and each character is checked to make sure it is alphanumeric.

Contents name. The contents name in NEWNAME is verified as not already existing in that folder. The name is checked for validity as are the new drawer, file, and folder names.

Utilities

There are several operations which are used several times in command files in OATS and might be useful for additional command files. Checking the existence of files, disk quotas, and finding out the size of a document are some that are available. Table 6 shows these various command files, what they do, what their parameters are, and how results are returned to the calling command file. The VAX/VMS System Symbol \$STATUS is frequently used to return values in these programs. The value can be tested with the IF...THEN command.

FILEFOUND

This command file has as the P1 parameter the name of a file. A DIRECTORY command is executed using this file name as the argument with the output going to a temporary file. The file is then read and

Table 6. Utility command files.

Command File	Parameters	Action Performed
DOCSELECT	none	Display Document Selection Screen and get specification for old document (Drawer, file, folder, and contents). NAMELINE is equivalent to file specification. WHICHLAST := "OLD".
FILEFOUND	P1 (filename)	\$STATUS equivalent to 1 if file exists, 3 if not.
GETDATE	none	NEWDATE equivalent to date entered.
GETSIZE	P1 (filename)	SIZE equivalent to the size of the file (blocks).
NAMEGOOD	P1 (filename)	Checks P1 against list of filenames in VALID_NAMES created by SHOWDIRS. \$STATUS equivalent to 1 if found, # if not.
NEWNAME	none	Display Document Selection Screen and get specification for new document (Drawer, file, folder, contents) NEWLINE equivalent to file specification. WHICHLAST := "NEW".
ROOMLEFT	P1 (number)	Checks disk quota remaining against P1. If P1 < remaining, \$STATUS equivalent to 1; if P1 > remaining, \$STATUS equivalent to 3.
SHOWDIRS	P1 (filetype)	Clear right half of screen unless filetype = .MAI or .DIS. Display filenames in file DIRECTORY.TMP (created with /NOHEADER/NOTRAIL switches) on right half of screen.

if the last line indicates that the file was not found, then the exit code returned is "3". If the file was found, it is "1".

GETDATE

This command file prompts for a date, displaying the current date in the place where the prompt is to be entered. All or part of the date specification may be typed over. The date must be in the proper format "dd-mmm-yyyy". The new date is arrived at by checking the length of the string entered from the terminal and appending to it the remaining characters of the current date. A check is made to assure that dashes are in the proper place and that the century is correct. Then the month is checked against a concatenated string of the first three letters of each month to see that it is valid. A set of twelve symbols was defined at startup in the command file SETSYMBOL with the names "monSYM" where "mon" is each month of the year. These symbols are equivalenced to a string of the form "xxyy" where "xx" is the number of days in the month and "yy" is the number of the month (e.g. April is equivalent to the string "3004"). A check is made to see if it is February in a leap year, and if so, the day is checked for less than or equal to "29". Other months are checked for a valid date. The screen is cleared and the symbol NEWDATE is equivalenced to the date received from the user. If there is an error anywhere in the date specification, the user is prompted again for the date.

GETSIZE

The P1 parameter to this command file is a file name. A DIRECTORY command is executed with the switches "/size" and "/versions=1" and is output to a temporary file. The temporary file is then read and the appropriate place in the last line of the temporary file is extracted to obtain the number of blocks in the file. Then the temporary file is closed and deleted.

NAMEGOOD

This command file searches through the symbol VALID_NAMES defined by the command file SHOWDIRS and searches for a match with the P1 parameter passed to it. The symbol VALID_NAMES has the format

"[nfilename][nfilename]...[nfilename]"

where "n" is the number of characters in the "filename". If a match is not found, an exit code of "3" is returned as the \$STATUS symbol. If it is found, the exit code is "1".

ROOMLEFT

This command file has as its P1 parameter a number. A command file called CHKQUOTA is called which contains the single command "SHOW QUOTA". The call to this command file is "@CHKQUOTA/output=chkquota.tmp" which will put any output from the execution of the command file that would normally be displayed on the screen into the file chkquota.tmp. This is then read by ROOMLEFT, the number of blocks left extracted from its location in the temporary

file and the number of blocks left compared with the P1 parameter (which is the number of blocks required or requested by the calling command file) passed to the command file. If there is room, the exit return in \$STATUS is "1". If not, the return is "3".

SHOWDIRS

Several times in OATS it is very useful to display on the screen a list of files from which the user may be asked to chose a name. The file DIRECTORY.LST is read and is to be created prior to a call to SHOWDIRS by the execution of the VAX Directory command with the switches "/output=directory.lst/noheader/notrail". The P1 argument may or not be included in the call to SHOWDIRS. If it is ".MAI" (mail message storage files) or ".DIS" (mail distribution lists), the screen is not erased. If the P1 argument is not included, the right half of the screen is erased. This is used in the document selection procedure to leave the drawer picture on the screen. The directory file is then read and the file names (without directory names or file type identifiers) are listed vertically in the right half of the screen in more than one column if necessary. At the same time, the length of the file names and the file names themselves are concatenated to an initially null string called VALID_NAMES which is used by the command file NAMEGOOD to verify that the filename is one that is valid. "QUIT" and "HELP" are also listed near the bottom of the screen to remind the user that he can select those options as well as giving a file name. Obviously, it is not possible to have a file name of "QUIT" or "HELP" .

CHAPTER III

CUSTOMIZATION AND PERSONALIZATION OF OATS

Addition of a Program

The menu-oriented structure of OATS makes adding options to it very easy. After the new program has been decided upon, it is necessary to determine if any parameters are necessary for the running of the program. For example, the name of a file for input or output might be required. Then a DCL command file is needed to set up the parameters and run the program. After this is written it must be added to the appropriate menu as an option.

Adding a program with no parameters which need to be passed requires only two steps. This first is to create a DCL command file to run the program. The file would contain the command needed to execute the program, "\$ RUN program" where "program" is the program name that resides in the user's directory. If it is a VAX system program, just the program name is sufficient.

The second step is to add the program to a menu in OATS by modifying the doubly-linked list that determines the menu selection process. First, copy the OATS CHOICELST.COM command file into the user's directory if the user does not already have one. Then select a PICKnn number greater than "49" that is unused as the symbol or node name. Using the format described in Chapter II, construct a string to which to equate the symbol node, using "P" for program as the first

character, the name you want to appear on the menu, and the name of the command file.

One other change must be made to the CHOICELST.COM command file. The new node must be inserted into the list. The PICKnn number of your program must be inserted into the list of PICKnn numbers for the symbol of the menu in which your program will be listed. It is necessary to increment the count of menu items also.

It may be that the function to be added to OATS can be written entirely as a command file. If so, it is added to the menu just as a program is, with the exception that the command file does everything and no program is called.

Addition of a Menu

It may be that the function to be performed is complex enough (i.e. choices are involved or decisions must be made) that it would lend itself to the menu-driven structure of OATS. Creating a menu actually requires only one more step than just adding a command file.

First the command file should be written so that the P1 parameter passed to it determines the action which will be taken within the command file. This parameter will be in the form of two digit node numbers which correspond to the PICKnn numbers in CHOICELST.COM. That is, instead of one PICKnn node symbol definition being added to CHOICELST, one is added for each action the command file can take, or each item the menu will list. After these node numbers have been assigned, the PICKnn node symbol for the menu must

be defined. "M" is selected, the title of the menu, the menu to be displayed on exit from this menu, and finally, the list of node numbers for each of the menu items. The node number for "quit" should also be included. Otherwise, the only way to exit from the menu would be with Control-Y. Lastly, the node number of the menu symbol must be added to some existing menu node in OATS. This might be the main menu, or if the command file is an enhancement of an existing function, it might be added to that menu. The maximum number of options that can be included in one menu is nine.

An example of how this might be useful is the addition of some specialized software that is used often by the user.

Additional Information on Customizing

Terminal Input to a Program Run by a Command File

Consideration also needs to be given to logical assignment of the user's terminal necessary for programs expecting terminal input. The command

ASSIGN/USER TT: SYS\$INPUT

is necessary before running the program. This is not necessary for input requested by a command file.

Input to the Command File

Two types of input may be required of the command file. One type is a file specification as part of the command. The other is input from the terminal to the program.

File specification. If the file specification is one that is an OATS document, a call to the appropriate OATS document selection command file must be part of the command file. If the OATS document already exists, "\$ @'OATS_DIRECTORY'DOCSELECT" is the command to use. The directory path and file name are equated to the global symbol NAMELINE and the subdirectory names are LAST_DRAWER, LAST_DRAWER, and LAST_FOLDER. The name of the document itself is equivalenced to the symbol LAST_CONTENTS. For a new file, "\$ @'OATS_DIRECTORY'NEWNAME" is used. The valid symbols in this case are NEWLINE for the directory path and file name and the symbols NEW_DRAWER, NEW_FILE, and NEW_FOLDER. If something is done in the program or command file to delete the file specified, then the symbol WHICHLAST must be made equivalent to the null string.

Other input. The INQUIRE command is useful in inputting values to command files. Using symbols for VT100 cursor control, the prompt for the command may be placed on any line on the screen. Output to the screen is also simplified by these symbols. Useful symbols already defined in OATS that may be utilized are listed in Table 5, page 24.

Global Symbols and Utility Command Files

There are numerous global symbols defined in OATS that would be very useful in constructing other command files. These symbols are shown in Table 5, page 24. The utility command files used by several OATS programs may also prove useful to the OATS user. These are found in Table 6, page 42.

An Example

The Objective

Suppose there is a need for the user to regularly use several programs resident on the VAX. It would be useful to the user to be able to access these programs from one menu in OATS. Assume the programs or functions used most often are DIFFERENCES (compares two files and reports their differences), and APPEND. Suppose also that the user would like to be able to run other programs whose executable images reside in his directory and that use two existing files as input by simply typing the program name.

The Analysis

There are three functions that are to be performed. Each one requires the specification of two existing file names. The numbers "50", "51", and "52" can be chosen as the node numbers for the functions and "53" as the node number for the menu. The best place to add the new menu would be at the bottom of the main menu. However, it is necessary to keep in mind that only nine menu choices are allowed in each menu.

The Solution

There would be very little programming required to construct the command file. Since all three functions require the specification of two already existing files, that could be done first, followed by three IF...THEN statements checking the value of P1 and branching to

the appropriate section of the command file. A fourth test and branch could be included if the user wished to add his own help message to the command file. Or all help messages could be included in one command file as they are in OATS. This requires another node symbol to be defined, but it can be used for all menus the user defines. Branching in the help command file is determined by the value of the symbol `OLD_TITLE`, which is the title of the last menu name. There is no help message included in the example. The first two branches would issue the appropriate command (`DIFFERENCES` and `APPEND`) with the symbols for the file names specified. The third could prompt for the name of the program to be run, and then using the symbols defined to run that program. Other refinements could be added as desired, such as clearing the screen, placing the cursor for prompts, etc. The modifications to the command file `CHOICELST.COM` are in Figure 7 and a simple command file to accomplish what is desired is in Figure 8.

```

PICK01 := "MO4MAIN00080302090605532549"
PICK50 := "P11DIFFERENCES4307SPECIAL"
PICK51 := "PO6APPEND5307SPECIAL"
PICK52 := "P14OTHER PROGRAMS53107SPECIAL"
PICK53 := "MO8PERSONAL010450515225"

```

Figure 7. Example of modification to CHOICELST.COM command file to add a menu to OATS.

```

$ ON CONTROL Y THEN EXIT
$ SAY CLEAR_SCREEN,MOVE,"23;1",DIRECT,"FILE 1:"
$ @'OATS_DIRECTORY'DOCSELECT
$ IF $STATUS .EQ. 3 THEN EXIT
$ FILE1 = NAMELINE
$ SAY MOVE,"24;1",DIRECT,"FILE 2:"
$ @'OATS_DIRECTORY'DOCSELECT
$ IF $STATUS .EQ. 3 THEN EXIT
$ FILE2 = NAMELINE
$ IF P1 .EQS. "40" THEN GOTO DIFFERENCE
$ IF P1 .EQS. "41" THEN GOTO APPEND_IT
$ IF P1 .EQS. "42" THEN GOTO OTHER
$DIFFERENCE:
$ DIFFERENCE 'FILE1' 'FILE2'
$ EXIT
$APPEND_IT:
$ APPEND 'FILE1' 'FILE2'
$ EXIT
$OTHER:
$ SAY MOVE,"21;1",DIRECT,CLEAR_TO_CURSOR
$ INQUIRE WHAT "PROGRAM"
$ 'WHAT' 'FILE1' 'FILE2'
$ EXIT

```

Figure 8. Example of command file added as a menu to OATS.

CHAPTER IV

OATS TUTORIAL

Invoking OATS

Three commands need to be carried out to put the user in OATS. For convenience, they should be put in a file called LOGIN.COM in the user's primary directory. The commands are:

```
OATS_DIRECTORY := "[****]"
VMS_PRIV := "NO"
SET_DEFAULT [user.OATS]
@'OATS_DIRECTORY'OFFICE
```

where "user" is the user's name and "[****]" is the directory where the OATS files reside. These commands will automatically cause the user to begin OATS when he is logged into the VAX. When the user exits from the main menu of OATS, he will be logged off the VAX also.

If the user wants to be able to use VAX/VMS outside of OATS, the above commands could be put into a command file in the user's directory and executed by him when he wanted to use OATS. However, the value of VMS_PRIV should be "YES" instead of "NO".

As soon as OATS is begun, if there are any messages created by the REMINDER option for the current date, they will be displayed on the screen. If there are no reminders for the day, then OATS will skip this part.

Menu Selection

The main menu of OATS lists the options available (Figure 9.). Selection is made by typing in the first two letters of the options desired. These two letters are bolded in the list of options as well as listed at the bottom of the screen. Depending on the option selected, OATS then displays another menu or passes control to the program that will accomplish the objective chosen by the user.

Main Menu. (Figure 9)

MAIL. This choice allows use of the MAIL program to send or read mail, create and edit office mail lists for use with MAIL, and create and edit MAIL messages that can be saved for future reference.

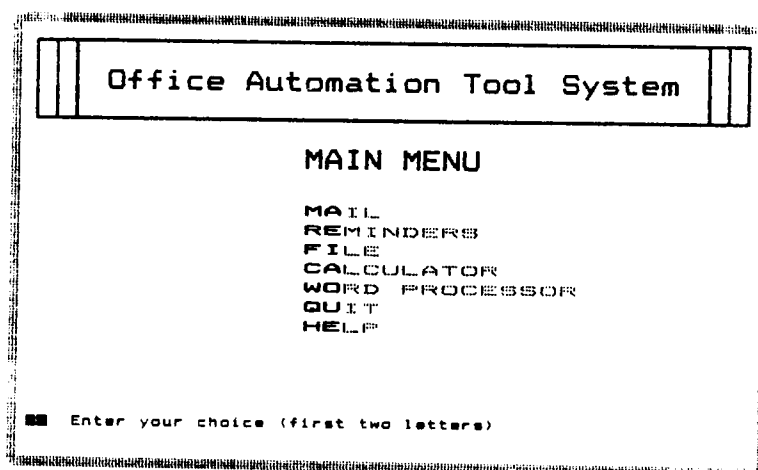


Figure 9. Main Menu screen.

REMINDERS. This will allow the user to create messages to himself for specific days that are displayed on the screen when OATS is begun, or when REMINDERS is selected from this menu.

FILE. The user can access documents previously filed. These documents may be moved, thrown away, viewed, printed, or duplicated.

WORD PROCESSOR. This gives the user access to the EDT editor to create and edit documents. They may be automatically formatted for letter, memo, or standard styles (using RUNOFF commands) or may contain no formatting. The formatted (completed) documents may be viewed or printed using the FILE selection.

CALCULATOR. This gives the user access to a calculator program that will display the calculations on the screen.

HELP. This message will be displayed on the screen.

QUIT. The user's session on OATS will be ended.

Mail Menu (Figure 10)

READ/SEND MAIL. This will put the user in the MAIL program on the VAX.

OFFICE MAIL LIST. EDT is used to create and edit lists of user names on the VAX to whom MAIL messages might be sent. One of these lists can be specified as the recipient of any MAIL message rather than any single user by using the response "@listname" to the prompt "TO:".

EXTERNAL MAIL LIST. This will create and maintain a list of names, addresses, and phone numbers. However, it is not implemented at this time.

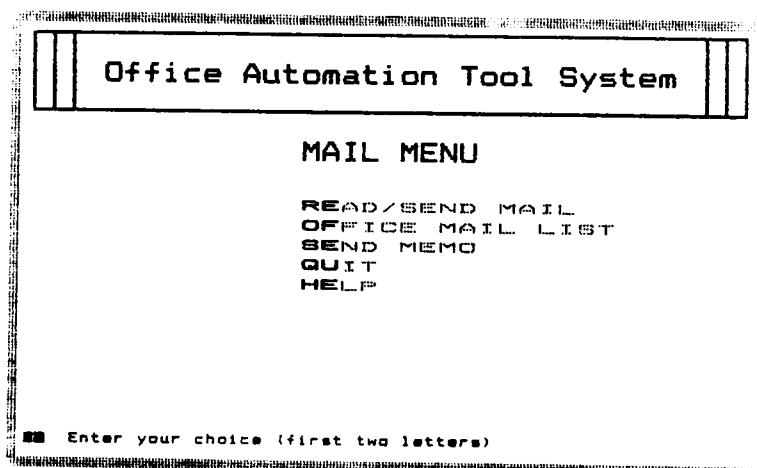


Figure 10. Mail Menu screen.

SEND MEMO. Using EDT, edit a memo to be sent to a user or group of users via MAIL.

HELP. Display a brief explanation of the menu options on the screen.

QUIT. Return to the OATS main menu.

Reminders Menu (Figures 11 and 12)

The user can maintain a message center for himself. Each option will prompt for a date, displaying the current date which may be partially or completely typed over with the date desired. If RETURN is pressed with no changes to the date displayed, the current date will be chosen.

Office Automation Tool System

REMINDERS MENU

LOOK
ADD
ERASE
QUIT
HELP

Enter your choice (first two letters)

Figure 11. Reminder Menu screen.

Reminders for
12-MAR-1985

1. Meeting with staff at 3:30.
2. Reports due.

DATE: 12-MAR-1985

REMINDER

Figure 12. Reminder program screen example.

ADD. A message will be added to the list for the date chosen. The other messages for that date will be displayed and a prompt will be given to type in the new message.

LOOK. Messages for the date chosen will be shown.

ERASE. This lists and numbers the messages for the date and asks for the number of the message to be erased from the list.

HELP. - This will display a brief explanation of the menu options on the screen.

QUIT. This will return to the OATS main menu.

Word Processor

There are three steps to using the OATS Word Processor. The first is to select the document to be used. This may be either an old document (one that already exists) or a new document.

Next is the selection of the format style. Three formats have already been set up that include the necessary RUNOFF commands for a letter, a memo, and a standard format (autoparagraph only). It is also possible to select NONE (No formatting). If the format selected is a letter or a memo, the user will be asked to supply the name of the addressee, the name to be used in the salutation, the subject of the memo or letter, and the signature at the bottom of the letter. All the characters typed in in answer to these prompts must be alphabetic characters. If you wish and of the characters to be in lower case or there is any non-alphabetic character (such as a period, space, or apostrophe) to be entered, enclose the whole response in double quotation marks. For example,

"Mr. Wilson O'Rourke"

would appear in the document as just "MR" without the quotation marks.

Once the format has been selected, the formatting commands may be changed by the user during the edit phase of the Word Processing

procedure. Editing (using EDT) is next. The formatting commands for RUNOFF are already inserted in the document. All that is necessary is that the text be inserted in the appropriate place which is indicated in the document itself. The document is then filed as specified in document selection. It can then be viewed on the screen, printed, etc., using the FILE selection from the MAIN menu. It will be automatically formatted for you if it contained formatting commands. (Figure 13)

OLD DOCUMENT. A previously existing document is to be edited by EDT. The user will be asked for the drawer name, file name, folder name, and contents name.

NEW DOCUMENT. A new document is to be created for editing by EDT.

HELP. Display a brief message of the menu options on the screen.

QUIT. Return to the MAIN menu of OATS.

File Document (Figure 14)

The documents created can be looked at, moved around, etc. in many different ways.

VIEW. View a document on the screen. The document will be displayed twenty lines at a time. Press RETURN to see the next twenty lines. Press RETURN again to return to the file menu. Documents containing RUNOFF formatting commands will be formatted before being displayed on the screen.

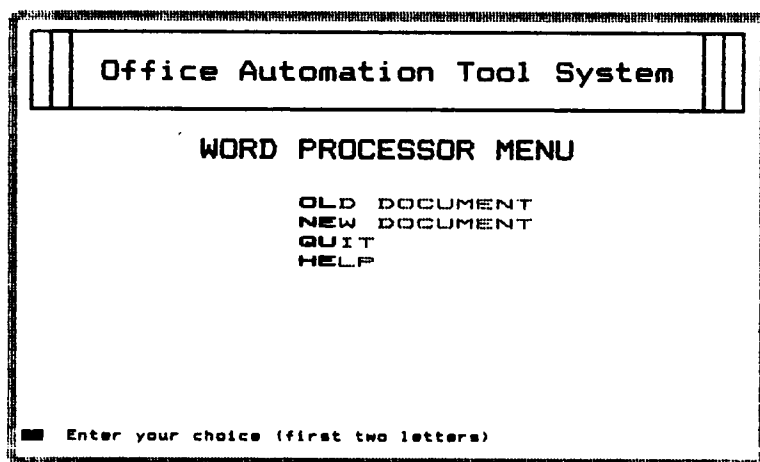


Figure 13. Word Processor Menu screen.

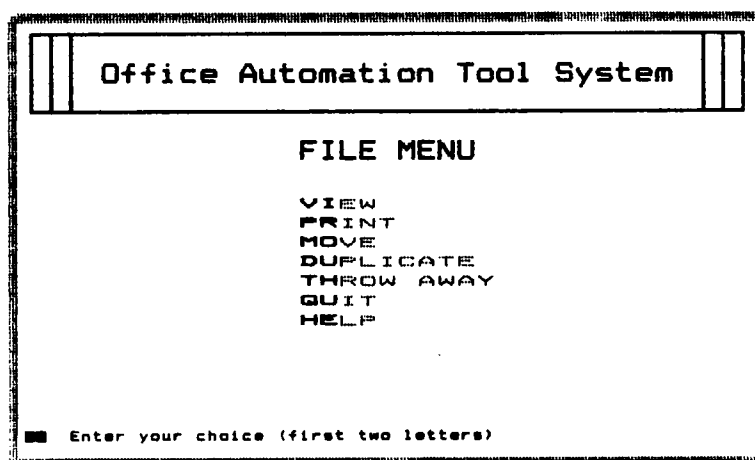


Figure 14. File Menu screen.

PRINT. Print a copy of the document (formatting done automatically).

MOVE. Move a document from one place (drawer, file, folder, contents) and place it in another. There is still only one copy of the document.

there is no precedence among the operators. Type the calculation desired on one line and use the RETURN or ENTER on the keypad as you would an equal sign on a hand-held calculator. Either the regular number keys may be used or the keypad numbers. The minus sign on the keypad also can be used. Press RETURN or ENTER twice to return to the Main Menu of OATS.

Creation and Filing of Documents

Documents are created using the editor EDT in one of several available formats. They are then automatically stored or filed for you in the place that you specify using the OATS filing system.

Document Selection (Figure 16)

The OATS filing system can be pictured like any standard filing system. There are four levels. When a document is to be selected, a picture of a file cabinet drawer is shown along with a list of the DRAWERS the user has defined and a prompt is given for the DRAWER selection. Refer to Figure 10 (page 56). If a new DRAWER is to be created (as might be done in WORD PROCESSING a new document or DUPLICATING or MOVING a document) the new name is typed in rather than one of the existing DRAWER names.

If a document was accessed or used previously during this session and it was not thrown away, the DRAWER, FILE, FOLDER, and CONTENTS names of that document are displayed on the picture of the file drawer on the screen. If you want to select that document again, just press RETURN. The exception to this is when creating a new

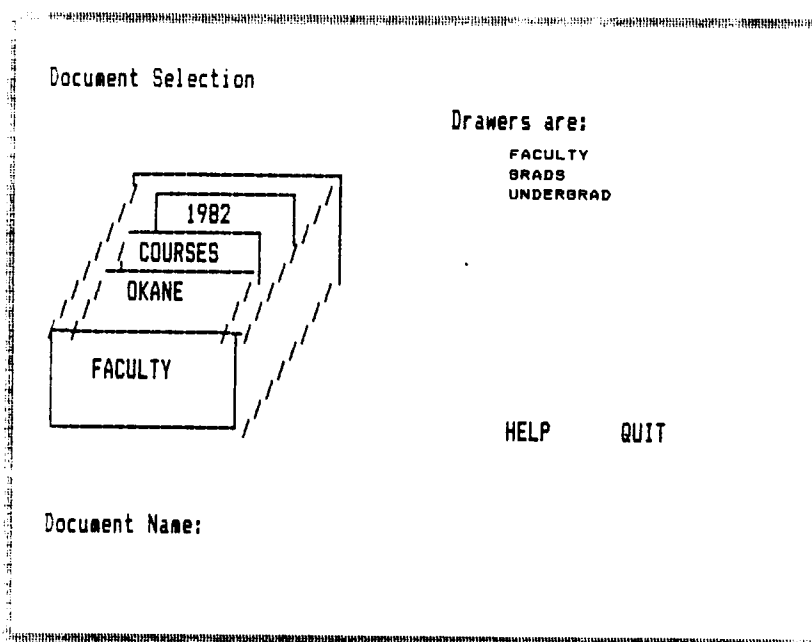


Figure 16. Document Selection screen example.

document. Then the DRAWER, FILE, and FOLDER (not CONTENTS) name of the last accessed file will be displayed. Pressing RETURN in that case will select only what was asked for (DRAWER, FILE, or FOLDER). In other words, to select the same DRAWER, FILE, and FOLDER as the previously accessed file when you are creating a new document, you will have to press RETURN three times before you are prompted for the NEW CONTENTS name.

Formatting a Document (Figure 17)

The menu for formatting a document gives the following options:

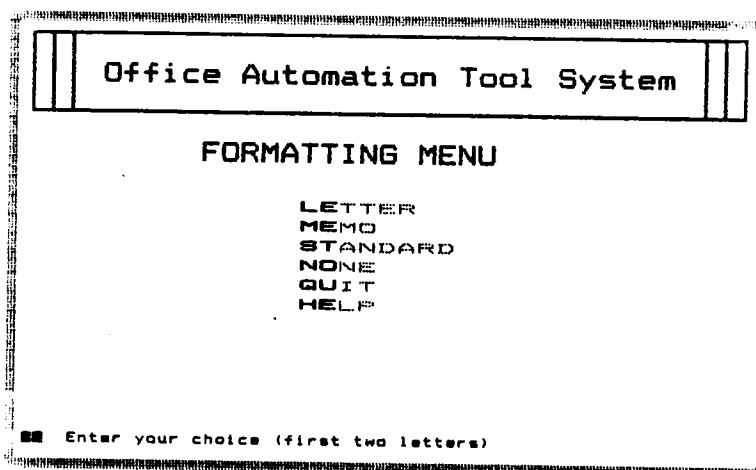


Figure 17. Format Document Menu screen.

LETTER. Creates a document with a letter format. You will be asked "To Whom:" (addressee of up to twenty characters in length), "From Whom:" (signature of up to twenty characters in length), and "Salutation Name:" (to be used in the greeting). The date is filled in automatically, as is the return address.

MEMO. Creates a document with a letter format. You will be asked TO WHOM, FROM WHOM, and SUBJECT (up to seventy characters).

STANDARD. Creates a document with auto paragraphing (Multiple spaces or a tab indicate new paragraph) and standard placement of left and right margins.

NONE. No formatting done. Document appears as is.

HELP. Displays a brief explanation of the menu options on the screen.

QUIT. Return to the OATS main menu.

Control Y

Pressing the "Y" key while holding down the "CONTROL" key at any time in OATS will accomplish the same purpose as typing QUIT at the Menu level. If not at the menu level, the current task you are performing will be aborted and you will be returned to where you were before you started the last task. The exception to this are when you are in EDT, where Control Y has no effect. To abort the edit, leave EDT with QUIT rather than the EXIT command.

If Control_Y is pressed during the document selection process, the name of the last document accessed may be destroyed. Therefore, it is best to use the "QUIT" option.

CHAPTER V

SUMMARY AND FUTURE ENHANCEMENTS

OATS provides the essential functions of other office automation systems available for the VAX without the addition of other software packages. The basic capabilities of OATS can be customized and expanded without the addition of expensive software, giving it an advantage over ALL-IN-1. The ease with which OATS is expanded because of its list-oriented menu structure and its reliance on DCL command files give it advantages over the VAX Professional Workstation's complexity.

Since OATS has virtually unlimited expandability there are a few additions for immediate expansion which would be quite advantageous. One of these is the inclusion of an address list manager to be part of the MAILPROG command file. Another is the APPEND cited in the example in Chapter IV.

The ability to keep a log file that records all transactions on OATS files would provide the user with helpful information. Included in this could be an index file with the date a file was last modified and a very brief description of the contents of the file. These additions could be extended to the concept of "browsing" through the files, searching for a particular document when unaware of its exact location.

Another aspect of making additions to OATS is modifying the current functions. For example, if a more useful program for a

particular function is available, the name of the newer program can be substituted for the old in the command file.

Although the addition of items to the menus is relatively simple, a command file written to accomplish that purpose would make the task even easier. The user could be prompted for the menu to add it to, command file name, menu name, etc. The modification program could even be a menu choice itself so that the entire process could take place within OATS.

As modifications are made to OATS, it is essential to remember that simplicity of use is the key to good office automation systems. The programming itself can be quite complex, but if that complexity becomes apparent to the user, the whole concept of office automation is defeated.

BIBLIOGRAPHY

BIBLIOGRAPHY

- John Murphy, "Office Automation," *Mini-Micro Systems*, Vol. XVI, No. 14, Dec. 1983, pp. 221-227.
- Dach Chang, "An Introduction to Integrated Software," *Byte*, Vol. 8, No. 12, Dec. 1983, pp. 103-112.
- Martin Herbach et al., "The User Interface: Two Approaches," *Byte*, Vol. 8, No. 12, Dec. 1983, pp. 247-262.
- John M. Carroll, "Presentation and Form in User-Interface Architecture," *Byte*, Vol. 8, No. 12, Dec. 1983, pp. 113-126.
- Stephen J. Kirchoff, "Development of Office Computing," *The DEC Professional*, Vol. 3, No. 3, May 1984, pp. 10-15.
- James G. Downward, "The VAX Professional Workstation," *The DEC Professional*, Vol. 3, No. 5, Sep. 1984, pp. 11-21.
- David M. Stroll, "A Common Interface, the Heart of Office Automation," *Computer Design*, Vol. 22, No. 12, pp. 69-74.
- Charles R. Miller, "Office Automation Is Not Just for Typists," *Computer Design*, Vol. 22, No. 12, pp. 75-80.
- Michael Chester, "Office Automation: Bridging the Multivendor Gap," *Systems & Software*, Vol. 3, No. 5, May 1984, pp. 110-126.
- Pamela A. Clark, "The Electronic Office," *Byte*, Vol. 8, No. 5, May 1983, p. 59.
- Dana Sohr, "Better Software Manuals," *Byte*, Vol. 8, No. 5, May 1983, pp. 286-297.
- "VAX/VMS Guide to Using Command Procedures," AA-H782A-TE, Digital Equipment Corporation, Maynard, Mass., 1980.
- "VAX/VMS Command Language User's Guide," AA-D023C-TE, Digital Equipment Corporation, Maynard, Mass., 1982.
- "VAX/VMS System Services Reference Manual," AA-D018B-TE, Digital Equipment Corporation, Maynard, Mass., 1980.
- "VAX/VMS Utilities Reference Manual," AA-H781A-TE, Digital Equipment Corporation, Maynard, Mass., 1981.
- "VAX/VMS I/O Users Guide," AA-D028B-TE, Digital Equipment Corporation, Maynard, Mass., 1980.

"ALL-IN-1 Handbook," EF-25810-52, Digital Equipment Corporation,
Maynard, Mass., 1983.

Sara Baase, VAX-11 Assembly Language Programming, Prentice-Hall,
Inc., 1983.

APPENDIXES

APPENDIX A

DIGITAL COMMAND LANGUAGE PROGRAMS

```

$!***** CALC *****!$
$   say clear_screen
$   BORDER := "*****"
$   SAY MOVE,"1;15",SCROLL,MOVE,"16;1",DIRECT,BORDER,BORDER,BORDER,BORDER
$   SAY INDEX_DOWN,REVERSE,"CALCULATOR",ATTRIBUTES_OFF
$   SAY INDEX_DOWN,"Operators: + - * / ( )"
$   SAY "Parentheses may be used. There is no order of precedence."
$   SAY "Use either keyboard or keypad for numbers."
$   SAY "Press RETURN or ENTER to display the answer."
$   SAY "Press RETURN or ENTER twice to return to OATS."
$   SAY MOVE,"1;1",DIRECT
$   ASSIGN/USER TT: SYS$INPUT
$   RUN 'OATS_DIRECTORY'DESKCALC
$   SAY CLEAR_SCREEN
$   EXIT

```

```

$!*****

```

```

$!*****

```

```

$!***** CHKQUOTA *****!$
$   SHOW QUOTA

```

```

$!*****

```

```

$!*****

```

```

$!***** CHOICELST *****!$
$   SET NOVERIFY
$!
$!   Symbol definitions for menu selection. Each symbol has the format
$!
$!       PICKnn
$!
$!   where nn is the choice number of the menu. The format of each
$!   MENU entry is:
$!
$!       MnnNAMEmmppaabbcc...zz
$!
$!   where  m indicates MENU type choice
$!           nn = the number of characters in the menu name,
$!           NAME = the name of the menu
$!           mm = the number of the menu to be shown upon exit

```

\$! pp = the number of choices in the menu
 \$! aa,bb,cc,...,zz = the menu or program numbers of the choices
 \$!

\$! The format of each PROGRAM entry is:

\$! PnnNAMEmmqqCNAME

\$! where P indicates PROGRAM type choice

\$! nn = the number of characters in the choice name,

\$! NAME = the name of the selection to be run

\$! mm = the number of the menu to be shown upon exit

\$! qq = the number of characters in the name of the COM
 file to be run

\$! CNAME = the name of the COM file to be executed

\$ PICK01 :== "M04MAIN000703020906052549"
 \$ PICK02 :== "M09REMINDERS01052204212549"
 \$ PICK03 :== "M04MAIL0106081407202549"
 \$ PICK04 :== "P03ADD0208REMINDER"
 \$ PICK05 :== "M14WORD PROCESSOR010424262549"
 \$ PICK06 :== "P10CALCULATOR0104CALC"
 \$ PICK07 :== "P18EXTERNAL MAIL LIST0308MAILPROG"
 \$ PICK08 :== "P14READ/SEND MAIL0308MAILPROG"
 \$ PICK09 :== "M04FILE010713151719232549"
 \$ PICK10 :== "M16FORMATTING STYLE0506111216182549"
 \$ PICK11 :== "P06LETTER0508WORDPROC"
 \$ PICK12 :== "P04MEMO0508WORDPROC"
 \$ PICK13 :== "P04VIEW0907FILEDOC"
 \$ PICK14 :== "P16OFFICE MAIL LIST0308MAILPROG"
 \$ PICK15 :== "P05PRINT0907FILEDOC"
 \$ PICK16 :== "P08STANDARD0508WORDPROC"
 \$ PICK17 :== "P04MOVE0907FILEDOC"
 \$ PICK18 :== "P04NONE0508WORDPROC"
 \$ PICK19 :== "P09DUPLICATE0907FILEDOC"
 \$ PICK20 :== "P09SEND MEMO0308MAILPROG"
 \$ PICK21 :== "P05ERASE0208REMINDER"
 \$ PICK22 :== "P04LOOK0208REMINDER"
 \$ PICK23 :== "P10THROW AWAY0907FILEDOC"
 \$ PICK24 :== "P12NEW DOCUMENT0508WORDPROC"
 \$ PICK25 :== "P04QUIT0004QUIT"
 \$ PICK26 :== "P12OLD DOCUMENT0508WORDPROC"
 \$ PICK49 :== "P04HELP0007HELPMSG"

\$!*****

\$!*****

\$!***** DOCSELECT *****!

\$! There is a four-level hierarchy filing system.

```

$! Drawers contain
$!     files which contain
$!     folders which contain
$!     documents, which are the actual
$!     data files.
$! The hierarchy is defined in terms of directory names as follows:
$!
$!     [USERNAME.OFFICE.drawer.file.folder]contents.OFF
$!
$! When DOCSELECT is called, a list of all "drawers" is shown
$! on the screen (i.e. all [username.office]*.dir files).
$! When the selection is made, all "files" in that drawer are
$! listed, etc., until the specifications for the entire
$! document are received. The last document accessed can
$! be selected by simply typing RETURN for the drawer name.
$! When DOCSELECT is initiated, the name of the last document
$! accessed is displayed in the proper slots. If there is
$! none (i.e. the beginning of a session), the slots are
$! blank and typing only a RETURN will produce no results.
$!
$! Initialization of symbols for picture of drawer.
$!
$BEGIN_AGAIN:
$     ON CONTROL_Y THEN GOTO Y_EXIT
$     DR_PLACE := "17;5"
$     FI_PLACE := "13;6"
$     FO_PLACE := "10;9"
$     CO_PLACE := "8;12"
$     INQ := "20;1"
$     INQM1 := "21;1"
$     BLANK := " "
$     VALID == 0
$!
$     DRAWER_NAME := ""
$     FILE_NAME := ""
$     FOLDER_NAME := ""
$     CONTENTS_NAME := ""
$!
$! Set up screen
$     SAY MOVE,"20;1",DIRECT,CLEAR_TO_CURSOR,ATTRIBUTES_OFF-
$         ,MOVE,"1;1",DIRECT,DOUBLE_WIDTH-
$         ,"Document Selection",ATTRIBUTES_OFF
$!
$! Draw picture of the drawer
$     SAY MOVE,"5;1",DIRECT
$     TYPE SYS$INPUT
$         [4m           [0m
$         / [4m           [0m /x
$         / x           x / x
$         / [4m x       [0m x/ x

```

```

      / /      x / x
    / [4m x      [0mx / x
  / /
  / /      / / /
[4m/ /      / [0m / /
x          x /
x          x /
x          x /
x          x /
[4mx      x[0m/
$!
$! Now get a directory listing of all the directory files for the drawer
$!
$      SAY ATTRIBUTES_OFF,MOVE,"3;35",DIRECT,REVERSE,"Drawers are:"-
      ,ATTRIBUTES_OFF
$!
$      DIR/OUTPUT=DIRECTORY.LST/NOHEADING/NOTAIL *.DIR
$!
$      @'oats_directory'SHOWDIRS ".DIR"
$!
$! If there was a previous document accessed, fill in the blanks
$!
$      IF WHICHLAST .EQS. "" THEN GOTO NO_LAST
$!
$      SAY MOVE,DR_PLACE,DIRECT,REVERSE,'WHICHLAST'_DRAWER,ATTRIBUTES_OFF
$      SAY MOVE,FI_PLACE,DIRECT,REVERSE,'WHICHLAST'_FILE,ATTRIBUTES_OFF
$      SAY MOVE,FD_PLACE,DIRECT,REVERSE,'WHICHLAST'_FOLDER,ATTRIBUTES_OFF
$      SAY MOVE,CO_PLACE,DIRECT,REVERSE,'WHICHLAST'_CONTENTS,ATTRIBUTES_OFF
$!
$NO_LAST:
$!
$! Ask the user for the drawer name.
$!
$DO_DRAWER:
$      SAY MOVE,INQM1,DIRECT,CLEAR_TO_EOL,MOVE,INQ,DIRECT
$      INQUIRE/GLOBAL DRAWER_NAME "Type in drawer name"
$      IF DRAWER_NAME .EQS. "HELP" THEN GOTO NEEDS_HELP
$      IF DRAWER_NAME .EQS. "QUIT" THEN EXIT 3
$      IF DRAWER_NAME .EQS. "" .AND. WHICHLAST .NES. "" THEN GOTO ALL_DONE
$!
$! Check the filename for validity
$!
$      @'oats_directory'NAMEGOOD DRAWER_NAME
$      IF $STATUS .EQ. 1 THEN GOTO D_GOOD
$!
$! The filename was not in the list of VALID_NAMES. Beep the user
$! and let him try again.
$!
$      SAY MOVE,INQM1,DIRECT,BEEP,BEEP
$      GOTO DO_DRAWER

```

```

$!
$D_GOOD:
$      SAY MOVE,DR_PLACE,DIRECT,REVERSE,"      "-
          ,MOVE,DR_PLACE,DIRECT,DRAWER_NAME-
          ,MOVE,FI_PLACE,DIRECT,"      "-
          ,MOVE,FO_PLACE,DIRECT,"      "-
          ,MOVE,CO_PLACE,DIRECT,"      ",ATTRIBUTES_OFF
$      CURRENT := 'F$DIRECTORY()'
$      LEN = 'F$LENGTH(CURRENT)'-1
$      NAMELINE := 'F$EXTRACT(0,LEN,CURRENT)'
$      NAMELINE := "'NAMELINE'.'DRAWER_NAME'"
$      SAY MOVE,"3;35",DIRECT,BLANK
$      SAY ATTRIBUTES_OFF,MOVE,"3;35",DIRECT,REVERSE,"Files are:"-
          ,ATTRIBUTES_OFF
$      DIR/OUTPUT=DIRECTORY.LST/NOHEADING/NOTAILING 'NAMELINE']*.DIR
$      @'oats_directory'SHOWDIRS ".DIR"
$!
$! Now need to get the file name
$!
$DO_FILE:
$      SAY MOVE,INQM1,DIRECT,BLANK,MOVE,INQ,DIRECT
$      INQUIRE/GLOBAL FILE_NAME "Type in file name"
$      IF FILE_NAME.EQS. "HELP" THEN GOTO NEEDS_HELP
$      IF FILE_NAME.EQS. "QUIT" THEN EXIT 3
$!
$! Check the filename for validity
$!
$      @'oats_directory'NAMEGOOD FILE_NAME
$      IF $STATUS.EQ. 1 THEN GOTO FI_GOOD
$!
$! The filename was not in the list of VALID_NAMES. Beep the user
$! and let him try again.
$!
$      SAY MOVE,INQM1,DIRECT,BEEP,BEEP
$      GOTO DO_FILE
$!
$FI_GOOD:
$      SAY MOVE,FI_PLACE,DIRECT,REVERSE,FILE_NAME,ATTRIBUTES_OFF
$      NAMELINE := " 'NAMELINE'.'FILE_NAME'"
$      SAY MOVE,"3;35",DIRECT,BLANK
$      SAY ATTRIBUTES_OFF,MOVE,"3;35",DIRECT,REVERSE,"Folders are:"-
          ,ATTRIBUTES_OFF
$      DIR/OUTPUT=DIRECTORY.LST/NOHEADING/NOTAILING 'NAMELINE']*.DIR
$      @'oats_directory'SHOWDIRS ".DIR"
$!
$! Now need to get the folder name
$!
$DO_FOLDER:
$      SAY MOVE,INQM1,DIRECT,BLANK,MOVE,INQ,DIRECT
$      INQUIRE/GLOBAL FOLDER_NAME "Type in folder name"

```

```

$      IF FOLDER_NAME .EQS. "HELP" THEN GOTO NEEDS_HELP
$      IF FOLDER_NAME .EQS. "QUIT" THEN EXIT 3
$!
$! Check the filename for validity
$!
$      @'oats_directory'NAMEGOOD FOLDER_NAME
$      IF $STATUS .EQ. 1 THEN GOTO FO_GOOD
$!
$! The filename was not in the list of VALID_NAMES. Beep the user
$! and let him try again.
$!
$      SAY MOVE,INQM1,DIRECT,BEEP,BEEP
$      GOTO DO_FOLDER
$!
$FO_GOOD:
$      SAY MOVE,FO_PLACE,DIRECT,REVERSE,FOLDER_NAME,ATTRIBUTES_OFF
$      NAMELINE :== "'NAMELINE'.'FOLDER_NAME'"
$      SAY MOVE,"3;35",DIRECT,BLANK
$      SAY ATTRIBUTES_OFF,MOVE,"3;35",DIRECT,REVERSE,"Contents are:"-
$      ,ATTRIBUTES_OFF
$      DIR/OUTPUT=DIRECTORY.LST/NOHEADING/NOTRAILING/VERSIONS=1-
$      'NAMELINE']*.OFF
$      @'oats_directory'SHOWDIRS ".OFF;"
$!
$! Now need to get the document name
$!
$DO_CONTS:
$      SAY MOVE,INQM1,DIRECT,BLANK,MOVE,INQ,DIRECT
$      INQUIRE/GLOBAL CONTENTS_NAME "Type in contents name"
$      IF CONTENTS_NAME .EQS. "HELP" THEN GOTO NEEDS_HELP
$      IF CONTENTS_NAME .EQS. "QUIT" THEN EXIT 3
$!
$! Check the filename for validity
$!
$      @'oats_directory'NAMEGOOD CONTENTS_NAME
$      IF $STATUS .EQ. 1 THEN GOTO CONTS_GOOD
$!
$! The filename was not in the list of VALID_NAMES. Beep the user
$! and let him try again.
$!
$      SAY MOVE,INQM1,DIRECT,BEEP,BEEP
$      GOTO DO_CONTS
$!
$CONTS_GOOD:
$      SAY MOVE,CO_PLACE,DIRECT,REVERSE,CONTENTS_NAME,ATTRIBUTES_OFF
$      NAMELINE :== "'NAMELINE']'CONTENTS_NAME'.OFF"
$      LAST_DRAWER :== 'DRAWER_NAME'
$      LAST_FILE :== 'FILE_NAME'
$      LAST_FOLDER :== 'FOLDER_NAME'
$      LAST_CONTENTS :== 'CONTENTS_NAME'

```

```

$!
$! Now have new valid document name.
$!
$ALL_DONE:
$      WHICHLAST := "LAST"
$
$      EXIT 1
$!
$! Here if he needs help.
$!
$NEEDS_HELP:
$      OLD_TITLE := "DOCUMENT"
$      @'oats_directory'HELPMMSG
$      IF $STATUS .EQ. 3 THEN EXIT 3
$      GOTO BEGIN_AGAIN
$!
$! Here on Control_Y
$!
$Y_EXIT:
$      WHICHLAST := ""
$      EXIT 3

$!*****

$!*****

$!***** FILEDOC *****!$
$! This program does the file manipulation. The function performed
$! is dependent on the calling argument, which is the symbol
$! CHOICE_NUMBER. The options and the corresponding value of
$! the calling argument are as follows
$!      13          VIEW -- DCL command TYPE
$!      15          PRINT -- DCL command PRINT
$!      17          MOVE -- DCL command RENAME
$!      19          DUPLICATE -- DCL command COPY
$!      23          THROW AWAY -- DCL command DELETE
$
$      ON CONTROL_Y THEN GOTO LEAVING
$      SAY CLEAR_SCREEN,MOVE,"1;22",SCROLL,MOVE,"24;72",DIRECT,REVERSE,"FILE"
$!
$      IF P1 .EQS. "13" THEN GOTO VIEW_IT
$      IF P1 .EQS. "15" THEN GOTO PRINT_IT
$      IF P1 .EQS. "17" THEN GOTO MOVE_IT
$      IF P1 .EQS. "19" THEN GOTO DUPLICATE_IT
$      IF P1 .EQS. "23" THEN GOTO THROW_IT
$VIEW_IT:
$      SAY MOVE,"24;1",DIRECT,BLINK,"View Document",ATTRIBUTES_OFF
$!
$! Select the file to be worked on.
$!

```

```

$      @'OATS_DIRECTORY'DOCSELECT
$!
$! Leave if he selected QUIT.
$!
$      IF $STATUS .EQ. 3 THEN EXIT
$!
$! Now display the name of the file being typed.
$!
$      SAY MOVE,"24;50",DIRECT,CLEAR_TO_CURSOR,MOVE,"24;1",DIRECT-
          ,"Viewing Document: ",BOLD,LAST_DRAWER," ",LAST_FILE," "-
          ,LAST_FOLDER," ",LAST_CONTENTS,ATTRIBUTES_OFF-
          ,MOVE,"0;0",SCROLL
$!
$! Now type it 20 lines at a time. The user must hit return to get
$! the next page of text displayed on the screen.
$!
$! Before typing the file, see if it is a RUNOFF file that needs
$! to be formatted first.
$!
$      RETURN := "VRET"
$      GOTO CHECK_FORMAT
$VRET:
$      OPEN TYPEFILE 'FILENAME'
$NEW_SCREEN:
$      LINE = 0
$NEXT_LINE:
$      READ TYPEFILE/END_OF_FILE=READ_DONE LINE_OF_TEXT
$      LINE = LINE + 1
$      SAY MOVE,LINE,";1",DIRECT,LINE_OF_TEXT
$      IF LINE .LT. 20 THEN GOTO NEXT_LINE
$!
$! At the end of the screen.
$!
$      SAY MOVE,"21;50",DIRECT,BLINK
$      INQUIRE IT "Press RETURN to see next page"
$      SAY CLEAR_TO_CURSOR,ATTRIBUTES_OFF
$      GOTO NEW_SCREEN
$READ_DONE:
$      SAY MOVE,"21;50",DIRECT,BLINK
$      INQUIRE IT "End of document. Press RETURN to continue"
$      CLOSE TYPEFILE
$      @'OATS_DIRECTORY'FILEFOUND FORM.TMP
$      IF $STATUS .EQ. 1 THEN DELETE FORM.TMP;*
$      SAY MOVE,"1;24",SCROLL,ATTRIBUTES_OFF,CLEAR_SCREEN
$      EXIT
$!
$!
$! Print the file specified. The only way to prevent the user
$! Seeing the message printed by the PRINT command is to blank
$! out the screen immediately.

```

```

$!
$PRINT_IT:
$      SAY MOVE,"24;1",DIRECT,BLINK,"Print Document",ATTRIBUTES_OFF
$!
$! Select the file to be worked on.
$!
$      @'DATS_DIRECTORY'DOCSELECT
$!
$! Leave if he selected QUIT.
$!
$      IF $STATUS .EQ. 3 THEN EXIT
$!
$! Now display the name of the file being printed.
$!
$      SAY MOVE,"24;50",DIRECT,CLEAR_TO_CURSOR,MOVE,"24;1",DIRECT-
          ,"Printing Document: ",BOLD,LAST_DRAWER," ",LAST_FILE," "-
          ,LAST_FOLDER," ",LAST_CONTENTS,ATTRIBUTES_OFF-
          ,MOVE,"1;3",SCROLL,MOVE,"1;1",DIRECT
$      RETURN := "PRET"
$      GOTO CHECK_FORMAT
$PRET:
$      IF FILENAME .NES. "FORM.TMP" THEN PRINT 'FILENAME'
$      IF FILENAME .EQS. "FORM.TMP" THEN PRINT/DELETE 'FILENAME'
$      SAY MOVE,"21;1",DIRECT,BLINK,CLEAR_TO_CURSOR
$      INQUIRE IT "Document queued for print. Press RETURN to continue"
$      SAY MOVE,"1;24",SCROLL,ATTRIBUTES_OFF,CLEAR_SCREEN
$      EXIT 1
$!
$!
$! Check the file to be viewed or printed to see if it begins with
$! a period. If it does, then it needs to be formatted first. The
$! formatted file (FORM.TMP) is then used and deleted.
$!
$CHECK_FORMAT:
$      FILENAME := 'NAMELINE'
$      OPEN TMP 'FILENAME'
$      READ TMP LINE
$      CODE := 'F$EXTRACT(0,1,LINE)'
$      CLOSE TMP
$      IF CODE .NES. "." THEN GOTO 'RETURN'
$      SAY MOVE,"20;1",DIRECT,BOLD,"Formatting document. Please wait"
$      RUNOFF/OUTPUT=FORM.TMP 'NAMELINE'
$      SAY MOVE,"20;1",DIRECT,CLEAR_LINE
$      FILENAME = "FORM.TMP"
$      GOTO 'RETURN'
$!
$!
$MOVE_IT:
$      SAY MOVE,"24;1",DIRECT,BLINK,"Document to be moved:",ATTRIBUTES_OFF
$!

```



```

$! Select the file to be worked on.
$!
$      @'OATS_DIRECTORY'DOCSELECT
$!
$! Leave if he selected QUIT.
$!
$      IF $STATUS .EQ. 3 THEN EXIT
$!
$! Now display the name of the file being moved.
$!
$      SAY MOVE,"24;50",DIRECT,CLEAR_TO_CURSOR,MOVE,"24;1",DIRECT-
        ,"Old Document Name: ",BOLD,LAST_DRAWER," ",LAST_FILE," "-
        ,LAST_FOLDER," ",LAST_CONTENTS,ATTRIBUTES_OFF-
        ,MOVE,"1;22",SCROLL,MOVE,"1;1",DIRECT
$!
$! Check the size of this file and make sure there's room to copy it.
$!
$      SIZE == 0
$      @'OATS_DIRECTORY'GETSIZE 'NAMELINE'
$      @'OATS_DIRECTORY'ROOMLEFT 'SIZE'
$      IF $STATUS .EQ. 1 THEN GOTO GO_AHEAD
$      SAY BEEP,BEEP,MOVE,"1;21",DIRECT,BLINK,REVERSE-
        ,"Sorry -- not enough room left. Duplication aborted."-
        ,ATTRIBUTES_OFF
$      WAIT 00:00:03
$      EXIT
$!
$! Now find out what the new Document name will be.
$!
$GO_AHEAD:
$      @'OATS_DIRECTORY'NEWNAME
$!
$! Leave if he selected QUIT.
$!
$      IF $STATUS .EQ. 3 THEN EXIT
$!
$! Now display the new file name selected.
$!
$      SAY MOVE,"22;50",DIRECT,CLEAR_TO_CURSOR,MOVE,"23;1",DIRECT-
        ,"New Document Name: ",BOLD,NEW_DRAWER," ",NEW_FILE," "-
        ,NEW_FOLDER," ",NEW_CONTENTS,ATTRIBUTES_OFF-
        ,MOVE,"1;3",SCROLL,MOVE,"1;1",DIRECT
$!
$! Now copy the file.
$!
$      COPY 'NAMELINE' 'NEWLINE'
$      LAST_DRAWER == NEW_DRAWER
$      LAST_FILE == NEW_FILE
$      LAST_FOLDER == NEW_FOLDER
$      LAST_CONTENTS == NEW_CONTENTS

```

```

$      NAMELINE == NEWLINE
$      WHICHLAST := "LAST"
$      SAY MOVE,"20;1",DIRECT,BOLD,"Document copied.",ATTRIBUTES_OFF
$      WAIT 00:00:03
$      SAY MOVE,"1;24",SCROLL
$      EXIT
$!
$!
$!
$THROW_IT:
$      SAY MOVE,"24;1",DIRECT,BLINK,"Document to be thrown away:"-
$          ,ATTRIBUTES_OFF
$!
$! Select the file to be worked on.
$!
$      @'OATS_DIRECTORY'DOCSELECT
$!
$! Leave if he selected QUIT.
$!
$      IF $STATUS .EQ. 3 THEN EXIT
$!
$! Now display the name of the file being deleted.
$      SAY MOVE,"24;50",DIRECT,CLEAR_TO_CURSOR,MOVE,"24;1",DIRECT-
$          ,"Throwing away Document : ",BOLD,LAST_DRAWER," ",LAST_FILE-
$          ," ",LAST_FOLDER," ",LAST_CONTENTS,ATTRIBUTES_OFF-
$          ,MOVE,"1;3",SCROLL,MOVE,"1;1",DIRECT
$!
$      SAY MOVE,"1;20",SCROLL,MOVE,"22;1",DIRECT,BOLD
$      INQUIRE ANS "ARE YOU SURE? (Y OR N)"
$      SAY ATTRIBUTES_OFF
$      IF ANS .NES. "Y" THEN EXIT
$      NAMELINE := "'NAMELINE';*"
$      DELETE 'NAMELINE'
$!
$! No last document accessed now.
$!
$      WHICHLAST := ""
$      SAY MOVE,"18;1",DIRECT,REVERSE,"Document Thrown Away",ATTRIBUTES_OFF
$CLEAN_UP_DIRECTORIES:
$!
$! See if any documents left in this folder.
$!
$      LOC = 'F$LOCATE("J",NAMELINE)' + 1
$      @'OATS_DIRECTORY'FILEFOUND 'F$EXTRACT(0,LOC,NAMELINE)'
$      IF $STATUS .EQ. 1 THEN GOTO NO_DELETIONS
$!
$! No documents left. Delete the folder directory.
$!
$!      SET MESSAGE/NOSEVERITY/NOTEXT/NOIDENT/NOFACILITY
$      SET PROT=OWNER:D [.'LAST_DRAWER'.'LAST_FILE'.'LAST_FOLDER'.DIR

```

```

$      DEL ['LAST_DRAWER','LAST_FILE']\LAST_FOLDER.DIR;1
$      @'DATS_DIRECTORY'FILEFOUND ['LAST_DRAWER','LAST_FILE']
$      IF $STATUS .EQ. 1 THEN GOTO NO_DELETIONS
$!
$! No folders left. Delete the file directory.
$!
$      SET PROT=OWNER:D ['LAST_DRAWER']\LAST_FILE.DIR
$      DEL ['LAST_DRAWER']\LAST_FILE.DIR;1
$      @'DATS_DIRECTORY'FILEFOUND ['LAST_DRAWER']
$      IF $STATUS .EQ. 1 THEN GOTO NO_DELETIONS
$!
$! No files left. Delete the drawer directory.
$!
$      SET PROT=OWNER:D 'LAST_DRAWER'.DIR
$      DEL 'LAST_DRAWER'.DIR;1
$NO_DELETIONS:
$      SET MESSAGE/IDENT/SEVERITY/FACILITY/TEXT
$      NAMELINE :== ""
$      WAIT 00:00:02
$      EXIT
$!
$!
$! Here to exit on control-y.
$!
$LEAVING:
$      CLOSE TYPEFILE
$      NAMELINE :== ""
$      LAST_DRAWER :== ""
$      DRAWER_NAME :== ""
$      NEWLINE :== ""
$      WHICHLAST :== ""
$      EXIT 3

$!*****

$!*****

$!***** FILEFOUND *****!$
$! This program determines if the file equated to the symbol P1 exists
$! or not. If it does, $STATUS is returned as 1. If not, $STATUS is
$! returned as 3 using the $EXIT_S utility.
$!
$      DIRECTORY/OUT=DIRECTORY.TMP/NOHEADING/NOTRAIL 'P1'
$      OPEN DIRTMP DIRECTORY.TMP
$READ_ON:
$      READ/END_OF_FILE=GOT_LAST_LINE DIRTMP LAST_LINE
$      GOTO READ_ON
$!
$GOT_LAST_LINE:
$      CODE := "3"

```

```

$      THREE := 'F$EXTRACT(0,3,LAST_LINE)'
$      IF THREE .EQS. "SYS" THEN CODE := "1"
$      DELETE DIRECTORY.TMP;*
$      CLOSE DIRTMP
$      EXIT 'CODE'

```

```

$!*****

```

```

$!*****

```

```

$!***** FORMAT *****!$

```

```

$! This program creates a file which contains basic RUNOFF commands
$! for formatting a document as a letter, memo, or standard. The
$! letter uses symbols TO_WHOM, FROM_WHOM, SAL, and DATE to set
$! the beginning of the letter. Memo uses only TO_WHOM and DATE.
$! Standard uses none of these. All are set to appropriate margins and
$! are auto-paragraphed as well as right- and left-justified.
$! The argument P1 is the document formatting type to be used.
$!
$!

```

```

$! First, open the designated file for input.
$!

```

```

$      OPEN/WRITE TMP 'NEWLINE'
$      IF P1 .EQS. "S" THEN GOTO STANDARD
$      IF P1 .EQS. "M" THEN GOTO MEMO
$      IF P1 .NES. "L" THEN EXIT 3
$      FIND = 'F$LOCATE("-",DATE)'
$      MONTH := 'F$EXTRACT(FIND+1,3,DATE)'
$      DAY := 'F$EXTRACT(0,FIND,DATE)'
$      FIND = FIND + 4
$      YEAR := 'F$EXTRACT(FIND+1,4,DATE)'
$      WRITE TMP ".LM50"
$      WRITE TMP ".RM72"
$      WRITE TMP "'MONTH'. 'DAY', 'YEAR'"
$      WRITE TMP ".LM5"
$      WRITE TMP ".RM75"
$      WRITE TMP ".LIT"
$      WRITE TMP "'TOWHOM'"
$      WRITE TMP ".END LITERAL"
$      WRITE TMP ".B3"
$      WRITE TMP "Dear 'SAL',"
$      WRITE TMP ".B1"
$      WRITE TMP ".AP"
$      WRITE TMP "*****erase this line and insert letter text here.*****"
$      WRITE TMP ".B2"
$      WRITE TMP ".LM50"
$      WRITE TMP ".LIT"
$      WRITE TMP "Sincerely,"
$      WRITE TMP " "
$      WRITE TMP "'FROMWHOM'"

```

```

$      WRITE TMP "Computer Science Department"
$      WRITE TMP "Ayres Hall"
$      WRITE TMP "University of Tennessee"
$      WRITE TMP "Knoxville, TN 37916"
$      WRITE TMP ".END LITERAL"
$      CLOSE TMP
$      EXIT

```

```
$!
```

```
$!
```

```
$MEMO:
```

```

$      WRITE TMP ".RM72"
$      WRITE TMP ".LM50"
$      WRITE TMP ".NO HEADER"
$      WRITE TMP ".LIT"
$      WRITE TMP "MEMO: "
$      WRITE TMP "TO: 'TOWHOM'"
$      WRITE TMP "FROM: 'FROMWHOM'"
$      WRITE TMP "SUBJECT: 'SUBJECT'"
$      WRITE TMP "DATE: 'DATE'"
$      WRITE TMP ".END LITERAL"
$      WRITE TMP ".AP"
$      WRITE TMP " "
$      CLOSE TMP
$      EXIT

```

```
$!
```

```
$!
```

```
$STANDARD:
```

```

$      WRITE TMP ".RM72"
$      WRITE TMP ".LMS"
$      WRITE TMP ".AP"
$      CLOSE TMP
$      EXIT

```

```
$!*****
```

```
$!*****
```

```
$!***** GETDATE *****!
```

```

$! This program asks for a date input in the format dd-mmm-yyyy. It
$! is then analyzed to see if it is a proper date. $STATUS = 1 if the
$! date is acceptable, $STATUS = 3 if it is not. After the prompt for
$! the date, the current date is displayed. This may be typed over
$! partially or wholly, or not at all( RETURN pressed). For example,
$! a typical prompt might be:

```

```
$!      DATE: 12-JAN-1985
```

```
$! The cursor would be at the "1" in "12". If the user typed in
```

```
$!      25-FEB<return>
```

```
$! the resulting date would be 25-FEB-1985.
```

```
$! Dates prior to the current date are not allowed.
```

```
$!
```

```

$      ON CONTROL_Y THEN GOTO ABORT
$!
$GET_DATE:
$      DATE[0,11] := "'F$TIME()'" !Today's date
$      SAY MOVE,"1;20",SCROLL,MOVE,"22;7",DIRECT,DATE,MOVE,"21;7",DIRECT
$      NEWDATE == DATE
$      INQUIRE NEWPART-
$          "'BOLD'Date'ATTRIBUTES_OFF'"
$      SAY MOVE,"23;1",CLEAR_LINE
$      SAY MOVE,"5;1",DIRECT
$      L = 'F$LENGTH(NEWPART)'
$      IF L .GT. 11 THEN GOTO BAD_DATE
$      IF L .EQ. 0 THEN GOTO DATE_OK
$      NEWDATE := "'F$EXTRACT(0,L,NEWPART)'"'F$EXTRACT(L,11-L,NEWDATE)'"
$!
$! Check for dashes in the proper place.
$!
$      D1 := 'F$EXTRACT(2,1,NEWDATE)'
$      D2 := 'F$EXTRACT(6,1,NEWDATE)'
$      IF D1 .NES. "-" .OR. D2 .NES. "-" THEN GOTO BAD_DATE
$!
$! Now have acceptable date format. Check each part.
$!
$      CENTURY := "'F$EXTRACT(7,2,NEWDATE)'"
$      MON := "'F$EXTRACT(3,3,NEWDATE)'"
$      DAY := "'F$EXTRACT(0,2,NEWDATE)'"
$      YEAR := "'F$EXTRACT(9,2,NEWDATE)'"
$      IF CENTURY .NES. "19" THEN GOTO BAD_DATE
$      COUNT = 0
$CHECK_MONTH:
$      TEST_MONTH := -
$          'F$EXTRACT(COUNT*3,3,"JANFEBMARAPRPMAYJUNJULAUUGSEPPOCTNOVDEC)'"
$      COUNT = COUNT + 1
$      IF MON .EQS. TEST_MONTH THEN GOTO MONTH_OK
$      IF COUNT .GE. 12 THEN GOTO BAD_DATE
$      GOTO CHECK_MONTH
$MONTH_OK:
$      M := 'MON'SYM
$      MONTH = "'F$EXTRACT(0,2,M)'"
$!
$! Check to see if month has that many days.
$!
$      NDAYS := 'F$EXTRACT(0,2,MONTH)'
$!
$! Do special things in case it's leap year.
$!
$      IF YEAR .EQS. "88" .OR. YEAR .EQS. "92" .OR. YEAR .EQS. "96" -
$          .AND. MON .EQS. "FEB" .AND. DAY .LES. "29" THEN GOTO DATE_OK
$      IF DAY .GTS. NDAYS THEN GOTO BAD_DATE
$DATE_OK:

```

```

$      SAY MOVE,"1;24",SCROLL,CLEAR_SCREEN
$      EXIT 1
$ABORT:
$      SAY MOVE,"1;24",SCROLL,CLEAR_SCREEN
$      EXIT 3
$BAD_DATE:
$      SAY MOVE,"23;1",DIRECT,BOLD,"Bad date. Try again.",BEEP,ATTRIBUTES_OFF
$      GOTO GET_DATE

```

```

$!*****

```

```

$!*****

```

```

$!***** GETSIZE *****!$
$! This program finds out the number of blocks in the most recent
$! copy of the file 'NAMELINE'.
$! The existence of the file has already been verified by the
$! Document Selection process.
$!
$      DIR/SIZE/OUT=DIRECTORY.TMP/VERSIONS=1 'NAMELINE'
$      OPEN/READ IN DIRECTORY.TMP
$MORE_READ:
$      READ/END_OF_FILE=LAST_LINE IN LINE
$      GOTO MORE_READ
$LAST_LINE:
$      CLOSE IN
$      N1 = 'F$LOCATE(", ",LINE)'
$      N2 = 'F$LOCATE("block",LINE)'
$      SIZE == 'F$EXTRACT(N1+1,N2-N1-2,LINE)'
$      SIZE == 'F$INTEGER(SIZE)'
$      DELETE DIRECTORY.TMP;*
$      EXIT

```

```

$!*****

```

```

$!*****

```

```

$!***** HELPMSG.COM *****!$
$!
$! This program displays the help message needed according to the value of
$! the symbol P1. The user erases the screen and continues by pressing
$! RETURN.
$!
$      ON CONTROL_Y THEN GOTO Y_EXIT
$      WHERE := 'F$EXTRACT(0,4,OLD_TITLE)'
$      GOTO 'WHERE'_HELP
$MAIN_HELP:
$      SAY CLEAR_SCREEN
$      TYPE SYS$INPUT
[CH#6[1m      HELP -- MAIN MENU

```

[1mMAIL [m-- Allows use of the MAIL program to send or read mail, create and edit office mail lists for use with MAIL, and create and edit MAIL messages that can be saved for future reference.

[1mREMINDERS [m-- Allows the user to create messages to himself for specific days that are displayed on the screen when OATS is begun, or when REMINDERS is selected from this menu.

[1mFILE [m-- Allow the user to access documents previously filed. These documents may be moved, thrown away, viewed, printed, or duplicated.

[1mCALCULATOR [m-- Allows the user to use the keypad keys as a calculator and have the calculations displayed on the screen:

[1mWORD PROCESSOR [m-- Gives the user access to the EDT editor to create and edit documents. They may be automatically formatted for letter, memo, or standard styles (using RUNOFF commands) or may contain no formatting. The formatted (completed) documents may be viewed or printed using FILE.

[1mHELP [m-- Displays this message on the screen.

[1mQUIT [m-- Ends the user's session on OATS.

```
$      INQUIRE X "'BOLD'Press RETURN to get back to menu"ATTRIBUTES_OFF"
$      SAY CLEAR_SCREEN
$      EXIT 1
```

\$!

\$!

\$!

\$MAIL_HELP:

```
$      SAY CLEAR_SCREEN
```

```
$      TYPE SYS$INPUT
```

```
[H#6[1m          HELP -- MAIL MENU
```

[1mREAD/SEND MAIL [m-- Allows the user to use the MAIL program on the VAX.

[1mOFFICE MAIL LIST [m-- Uses EDT to create and edit lists of user names on the VAX to whom MAIL messages might be sent. One of these lists can be specified as the recipient of any MAIL message rather than any single user by using "@listname".

[1mEXTERNAL MAIL LIST [m-- Create and maintain a list of names, addresses, and phone numbers.

[1mSEND MEMO [m-- Edit using EDT a memo to be sent to a user or group of users via MAIL.

[1mHELP [m-- Display this message.

[1mQUIT [m-- Return to the OATS main menu.

```

$      INQUIRE X "'BOLD'Press RETURN to get back to menu'"ATTRIBUTES_OFF"
$      SAY CLEAR_SCREEN
$      EXIT 1
$!
$!
$!
$WORD_HELP:
$      SAY CLEAR_SCREEN
$      TYPE SYS$INPUT
[H#6[I#      HELP -- WORD PROCESSOR

```

[# There are three steps to using the OATS Word Processor. The first is to select the document to be used. Next is the selection of the format style. Three formats have already been set up that include the necessary RUNOFF commands for a letter, a memo, and a standard format (auto-paragraph only). It is also possible to select NONE (No formatting). Once the format has been selected, the formatting commands may be changed by the user during the edit phase of the Word Processing procedure. Editing (using EDT) is next. The document is then filed using the name selected. It can then be viewed on the screen, printed, etc., using the FILE selection from the MAIN menu. It will be automatically formatted for you if it contained formatting commands.

[#OLD DOCUMENT [#-- A previously existing document is to be edited by EDT. The user will be asked for the drawer name, file name, folder name, and contents name.

[#NEW DOCUMENT [#-- A new document is to be created for editing by EDT.

[#HELP [#-- Display this message on the screen.

[#QUIT [#-- Return to the MAIN menu of OATS.

```

$      INQUIRE X "'BOLD'Press RETURN to get back to menu'"ATTRIBUTES_OFF"
$      SAY CLEAR_SCREEN
$      EXIT 1
$!
$!
$!
$FILE_HELP:
$      SAY CLEAR_SCREEN
$      TYPE SYS$INPUT
[H#6[I#      HELP -- FILE DOCUMENT

```

[#VIEW [#-- View a document on the screen. The document will be displayed 20 lines at a time. Press RETURN to see the next 20 lines. Press RETURN again to return to the FILE menu. Documents containing RUNOFF formatting commands will be formatted before being displayed on the screen.

[#PRINT [#-- Print a copy of the document (formatting done automatically).

[#MOVE [#-- Move a document from one place (drawer, file, folder, contents) and

place it in another. There is still only one copy of the document.

[1mDUPLICATE [m-- Make another copy of a document and file it in another locationg. There would then be two copies stored in different places.

[1mTHROW AWAY [m-- Allows throwing away the specified copy of a document. DATS will clean up folders, files, or drawers emptied by throwing away a document.

```
$      INQUIRE X "'BOLD'Press RETURN to get back to menu'ATTRIBUTES_OFF"
```

```
$      SAY CLEAR_SCREEN
```

```
$      EXIT 1
```

```
$!
```

```
$!
```

```
$!
```

```
$FORM_HELP:
```

```
$      SAY CLEAR_SCREEN
```

```
$      TYPE SYS$INPUT
```

```
[H#6[1m      HELP -- FORMATTING STYLE
```

[1mLETTER [m-- Creates a document with a letter format. You will be asked TO WHOM (Addressee- up to 20 characters), FROM WHOM (Signature- also up to 20 characters), and SALUTATION name (to be used in the greeting. The date is filled in automatically, as is the return address.

[1mMEMO [m-- Creates a document with a letter format. You will be asked TO WHOM, FROM WHOM, and SUBJECT (up to 70 characters).

[1mSTANDARD [m-- Creates a document with auto paragraphing (Multiple spaces or a TAB indicate new paragraph) and standard placement of left and right margins.

[1mNONE [m-- No formatting done. Document appears as is.

[1mHELP [m-- Displays this message on the screen

[1mQUIT [m-- Return to the DATS MAIN menu.

```
$      INQUIRE X "'BOLD'Press RETURN to get back to menu'ATTRIBUTES_OFF"
```

```
$      SAY CLEAR_SCREEN
```

```
$      EXIT 1
```

```
$!
```

```
$!
```

```
$!
```

```
$DOCU_HELP:
```

```
$      SAY CLEAR_SCREEN
```

```
$      TYPE SYS$INPUT
```

```
[H#6[1m      HELP -- DOCUMENT SELECTION
```

[1m The filing system for DATS is analogous to standard office filing systems. There are four levels:

```
[1m      DRAWERS [mwhich contain
```

```
      [1m      FILES [mwhich contain
```

```
      [1m      FOLDERS [mwhich contain
```

[1] CONTENTS [m] which are the actual documents.

When a document is to be selected, a picture of a file cabinet drawer is shown along with a list of the DRAWERS the user has defined and a prompt is given for the DRAWER selection. If a new DRAWER is to be created (as might be done in WORD PROCESSING a new document or DUPLICATING or MOVING a document) the new name is typed in rather than one of the existing DRAWER names.

If a document was accessed or used previously during this session and it was not thrown away, the DRAWER, FILE, FOLDER, and CONTENTS names of that document are displayed on the picture of the file drawer on the screen. If you want to select that document again, just press RETURN. The exception to this is when creating a new document. Then the DRAWER, FILE, and FOLDER (not CONTENTS) name of the last accessed file will be displayed. Pressing RETURN in that case will select only what was asked for (DRAWER, FILE, or FOLDER).

```
$      INQUIRE X "'BOLD'Press RETURN to get back to menu'"ATTRIBUTES_OFF"
$      SAY CLEAR_SCREEN
$      EXIT 1
$!
$!
$!
$Y_EXIT:
$      SAY CLEAR_SCREEN
$      EXIT 3
$!
$!
$!
$REMI_HELP:
$      SAY CLEAR_SCREEN
$      TYPE SYS$INPUT
[CH#6[1]  HELP -- REMINDERS
```

[m] You will be prompted for the date you wish to make a reminder for. The date must be in the form dd-~~mm~~-yyyy where dd is the day of the month, ~~mm~~ is the three-character abbreviation for the month, and yyyy is the year. Today's date will be printed on the screen and you can type over as much of it as is necessary to make it the date you want, and then press RETURN. What happens then depends on what you selected from the menu.

[1] ADD [m-- The messages for the date selected are displayed, then you are asked to enter the new message. Messages can be up to 70 characters long. The new message will be added to the end of the list for the day.

[1] LOOK [m-- The messages for the date selected are displayed on the screen. This is what happens for the current date when you log onto OATS. You must press RETURN to erase the screen and continue.

[1] ERASE [m-- The messages for the date selected are displayed on the

screen (they are numbered). You are then asked to type the number of the message you wish to erase.

```
$      INQUIRE X "'BOLD'Press RETURN to get back to menu'"ATTRIBUTES_OFF"
$      SAY CLEAR_SCREEN
$      EXIT 1
```

```
$!*****
```

```
$!*****
```

```
$!***** MAILPROG *****!
```

```
$! This program performs the task selected from the MAIL menu.
```

```
$!
```

```
$      SAY CLEAR_SCREEN,MOVE,"1;20;",SCROLL,MOVE,"24;50",DIRECT,REVERSE-
      , "Mail",ATTRIBUTES_OFF
$      IF P1 .EQS. "08" THEN GOTO PROGRAM
$      IF P1 .EQS. "14" THEN GOTO MAIL_LISTS
$      IF P1 .EQS. "07" THEN GOTO EXT_LISTS
$      IF P1 .EQS. "20" THEN GOTO SEND_MEMO
```

```
$!
```

```
$!
```

```
$PROGRAM:
```

```
$      LIST := "READ SEND DIRECTORY EXIT HELP QUIT"
$      SAY MOVE,"22;1",DIRECT,"MAIL COMMANDS:",MOVE,"23;1",DIRECT,BOLD-
      ,LIST,MOVE,"1;1",DIRECT,ATTRIBUTES_OFF
$      ASSIGN/USER TT: SYS$INPUT
$      MAIL
$      SAY MOVE,"1;24",SCROLL,CLEAR_SCREEN
$      EXIT 1
```

```
$!
```

```
$!
```

```
$!
```

```
$MAIL_LISTS:
```

```
$! Use WP to edit. List "lists" on screen, let user choose.
```

```
$!
```

```
$      SAY MOVE,"1;20",DIRECT,BOLD,"Internal mail distribution lists"-
      ,ATTRIBUTES_OFF
$      DIR/OUTPUT=DIRECTORY.LST/NOHEADING/NOTAIL/VERSIONS=1 *.DIS
$      VALID_NAMES := ""
$      @'DAYS_DIRECTORY'SHOWDIRS ".DIS"
```

```
$GET_NAME:
```

```
$      SAY MOVE,"22;1",CLEAR_TO_EOL,MOVE,"21;1",DIRECT
$      INQUIRE LIST_NAME "Select list name"
$      IF LIST_NAME .EQS. "HELP" THEN GOTO NEEDS_HELP1
$      IF LIST_NAME .EQS. "QUIT" THEN EXIT 1
$      IF LIST_NAME .EQS. "" THEN GOTO GET_NAME
```

```
$!
```

```
$! Check file name for validity
```

```
$!
```

```

$      @OATS_DIRECTORY'NAMEGOOD LIST_NAME
$      IF $STATUS .EQ. 1 THEN GOTO LI_GOOD
$!
$! New filename or mis-typed old one. Verify which.
$!
$      SAY MOVE,"23;1",DIRECT,"Mail List ",LIST_NAME," does not exist."
$      INQUIRE ANS "Create New List? (Y or N)"
$      IF ANS .EQS. "Y" THEN GOTO LI_GOOD
$BLANKLINE:
$      SAY MOVE,"22;1",DIRECT,CLEAR_LINE,MOVE,"23;1",DIRECT,CLEAR_LINE
$      GOTO GET_NAME
$!
$!
$LI_GOOD:
$!
$! ***go to edt with filename to create it. No formatting.
$!
$      SET NOCONTROL_Y
$      ASSIGN/USER TT: SYS$INPUT
$      EDIT/COMMAND='OATS_DIRECTORY'EDTINI.EDT 'LIST_NAME'.DIS
$      SET CONTROL_Y
$      SET MESSAGE/NOFACILITY/NOIDENTIFICATION/NOSEVERITY/NOTEXT
$      PURGE 'LIST_NAME'.DIS/KEEP=3
$      SET MESSAGE/FACILITY/IDENTIFICATION/SEVERITY/TEXT
$      EXIT 1
$!
$!
$EXT_LISTS:
$      SAY "External Mail Lists Program not available."
$      WAIT 00:00:3
$      EXIT
$!
$! Set up a memo file with edt that is then given to MAIL for
$! sending. Can use a list of receivers or a name. Validity
$! of list name is checked.
$!
$SEND_MEMO:
$      SAY MOVE,"1;1",DIRECT
$      INQUIRE TO_NAME "'BOLD'Memo to'ATTRIBUTES_OFF'"
$      IS_FILE := 'F$EXTRACT(0,1,TO_NAME)
$      TO_LIST := TO_NAME
$      IF IS_FILE .NES. "@" THEN GOTO NO_FILE
$      LEN := 'F$LENGTH(TO_NAME)
$      FNAME := 'F$EXTRACT(1,LEN,TO_NAME)
$      @OATS_DIRECTORY'FILEFOUND 'FNAME'.DIS
$      IF $STATUS .EQ. 1 THEN GOTO FILEOK
$!
$! The file does not exist.
$!
$      SAY BOLD,"Not a valid distribution list name.",ATTRIBUTES_OFF

```

```

$      GOTO SEND_MEMO
$!
$FILEOK:
$      OPEN TMP 'FNAME'.DIS
$      READ TMP UNAME
$      TO_LIST := 'UNAME'
$READ_NEXT:
$      READ/END_OF_FILE=CLOSE_IT TMP UNAME
$      IF UNAME .NES. "" THEN TO_LIST := ", 'TO_LIST' 'UNAME'"
$      GOTO READ_NEXT
$!
$! List of names is all in
$!
$CLOSE_IT:
$      CLOSE TMP
$NO_FILE:
$      INQUIRE SUBJECT "'BOLD'Subject'ATTRIBUTES_OFF'"
$      OPEN/WRITE TMP TEMPMEMO.TMP
$      WRITE TMP "To: ", TO_LIST
$      WRITE TMP "Subject: ", SUBJECT
$      WRITE TMP ""
$      CLOSE TMP
$      ASSIGN/USER TT: SYS$INPUT
$      EDIT/COMMAND='DATS_DIRECTORY'EDTINI.EDT TEMPMEMO.TMP
$!
$! Now take out the first three lines. They were for the edit only.
$!
$      OPEN TMP TEMPMEMO.TMP
$      READ TMP X
$      READ TMP X
$      READ TMP X
$      OPEN/WRITE OUT MEMO.TMP
$READ_ON:
$      READ/END_OF_FILE=ALL_DONE TMP XX
$      WRITE OUT XX
$      GOTO READ_ON
$ALL_DONE:
$      CLOSE TMP
$      CLOSE OUT
$!
$! Now mail the message.
$!
$      IF IS_FILE .EQS. "@" THEN -
$          MAIL/SUBJECT="'SUBJECT' MEMO.TMP "'TO_NAME'"
$!
$      IF IS_FILE .NES. "@" THEN -
$          MAIL/SUBJECT="'SUBJECT' MEMO.TMP 'TO_NAME'"
$!
$! Now determine the fate of the message file.
$!

```

```

$ASK_KEEP:
$      INQUIRE RES "'BOLD'Do you wish to file this memo?'ATTRIBUTES_OFF'"
$      RESP := 'F$EXTRACT(0,1,RES)'
$      IF RESP .EQS. "Y" THEN GOTO KEEP_IT
$      DELETE MEMO.TMP;*,TEMPMEMO.TMP;*
$ WAIT 00:00:05
$      SAY CLEAR_SCREEN
$      EXIT
$!
$KEEP_IT:
$      @'OATS_DIRECTORY'NEWNAME
$      IF $STATUS .EQ. 3 THEN GOTO ASK_KEEP
$      RENAME MEMO.TMP 'NEWLINE'
$      SAY CLEAR_SCREEN
$      EXIT

$!*****

$!*****

$!***** NAMEGOOD *****!$
$! This program checks the validity of the filename typed in by the user
$! by comparing it to the list of valid filenames created by SHOWDIRS.COM.
$! The symbol VALID_NAMES has the form
$!      [nfilename][nfilename]...[nfilename]
$! where n is the number of characters in the filename.
$!
$! The filename is the argument P1.
$!
$! If the filename is valid, the symbol $STATUS is given the value
$! of 1 by the EXIT call. If not valid, $STATUS is given the value
$! of 3 by the EXIT call.
$!
$!
$! Tack a zero on the end, so I'll know when I'm done.
$!
$      VALID_NAMES := "'VALID_NAMES'0"
$      LEN_RESPONSE = 'F$LENGTH(&P1)'
$      IF LEN_RESPONSE .GT. 9 THEN GOTO NO_GOOD
$      OFF_SET = 0
$NOT_YET:
$      L1:= 'F$EXTRACT(OFF_SET,1,VALID_NAMES)'
$      LENGTH = 'F$CVUI(0,4,L1)'
$      IF LENGTH .EQ. 0 THEN GOTO NO_GOOD
$!
$      IF LENGTH .EQ. LEN_RESPONSE THEN GOTO LENGTH_OK
$GO_ON:
$      OFF_SET = OFF_SET + LENGTH + 1
$      GOTO NOT_YET
$!

```

```

$! This name is the right length. Check it out.
$!
$LENGTH_OK:
$     NUM = OFF_SET + 1
$     NAME_SEE := 'F$EXTRACT(NUM,LENGTH,VALID_NAMES)'
$     IF &P1 .EQS. NAME_SEE THEN GOTO NAME_OK
$     GOTO GO_ON
$!
$NO_GOOD:
$     EXIT 3
$!
$NAME_OK:
$     EXIT 1
$!

$!*****

$!*****

$!***** NEWNAME *****!$
$! There is a four-level hierarchy filing system.
$! Drawers contain
$!     files which contain
$!     folders which contain
$!     documents, which are the actual
$!     data files.
$! The hierarchy is defined in terms of directory names as follows:
$!
$!     [USERNAME.OFFICE.drawer.file.folder]contents.OFF
$!
$! When NEWNAME is called, a list of all "drawers" is shown
$! on the screen (i.e. all [username.office]*.dir files).
$! If an existing drawer is selected, then all files in that
$! drawer are listed, as in DOCSELECT. If a new drawer is
$! selected, then user is prompted for file, folder, and
$! contents names. The new directory is then created.
$! If the response to the prompt for the drawer name,
$! a return will select the drawer, file, and folder, but not
$! the contents, previously accessed, if there is a previously
$! accessed document. If there is none (i.e. the beginning of a session),
$! the slots are blank and typing only a RETURN will produce no results.
$!
$! Initialization of logical names for picture of drawer.
$!
$BEGIN_AGAIN:
$     ON CONTROL_Y THEN GOTO Y_EXIT
$     DR_PLACE := "17;5"
$     FI_PLACE := "13;6"
$     FO_PLACE := "10;9"
$     CO_PLACE := "8;12"

```

```

$      INQ := "20;1"
$      INQM1 := 21;1"
$      BLANK := "
$      VALID = 0
$!
$      DRAWER_NAME := ""
$      FILE_NAME := ""
$      FOLDER_NAME := ""
$      CONTENTS_NAME := ""
$      NEWLINE := ""
$!
$!
$! Set up screen
$      SAY MOVE,"20;1",DIRECT,CLEAR_TO_CURSOR,ATTRIBUTES_OFF-
        ,MOVE,"1;1",DIRECT,DOUBLE_WIDTH-
        ,"Document Selection",ATTRIBUTES_OFF
$!
$! Draw picture of the drawer
$!
$      SAY MOVE,"5;1",DIRECT
$      TYPE SYS$INPUT
        [4m          [0m
        / [4m          [0m /x
        / x          x / x
        / [4m x      [0m x/ x
        //          x / x
        / [4m x      [0mx / x
        //          // /
        //          // /
[4m//          /[0m / /
x          x /
x          x /
x          x /
x          x /
[4mx          x[0m/
$!
$! Now get a directory listing of all the directory files for the drawer
$!
$      SAY ATTRIBUTES_OFF,MOVE,"3;35",DIRECT,REVERSE,"Drawers are:"-
        ,ATTRIBUTES_OFF
$!
$      DIR/OUTPUT=DIRECTORY.LST/NOHEADING/NOTAIL *.DIR
$!
$      @'DATS_DIRECTORY'SHOWDIRS ".DIR"
$!
$! If there was a previous document accessed, fill in the blanks
$!
$      IF WHICHLAST .EQS. "" THEN GOTO NO_LAST
$!
$      SAY MOVE,DR_PLACE,DIRECT,REVERSE,'WHICHLAST'_DRAWER,ATTRIBUTES_OFF

```

```

$      SAY MOVE,FI_PLACE,DIRECT,REVERSE,'WHICHLAST'_FILE,ATTRIBUTES_OFF
$      SAY MOVE,FO_PLACE,DIRECT,REVERSE,'WHICHLAST'_FOLDER,ATTRIBUTES_OFF
$      SAY MOVE,CO_PLACE,DIRECT,REVERSE,"      ",ATTRIBUTES_OFF
$      GOTO DO_DRAWER
$!
$NO_LAST:
$      SAY MOVE,DR_PLACE,DIRECT,REVERSE,"      "-
$              ,MOVE,FI_PLACE,DIRECT,"      "-
$              ,MOVE,FO_PLACE,DIRECT,"      "-
$              ,MOVE,CO_PLACE,DIRECT,"      ",ATTRIBUTES_OFF
$!
$! Ask the user for the drawer name.
$!
$DO_DRAWER:
$      SAY MOVE,INQM1,DIRECT,CLEAR_TO_EOL,MOVE,INQ,DIRECT
$      INQUIRE/GLOBAL DRAWER_NAME "Type in drawer name"
$      IF DRAWER_NAME .EQS. "HELP" THEN GOTO NEEDS_HELP
$      IF DRAWER_NAME .EQS. "QUIT" THEN EXIT 3
$      IF WHICHLAST .EQS. "" .AND. DRAWER_NAME .EQS. "" THEN GOTO DO_DRAWER
$      IF DRAWER_NAME .EQS. "" THEN DRAWER_NAME == 'WHICHLAST'_DRAWER
$      CREATE_NEW = 0
$      IF WHICHLAST .EQS. "" THEN GOTO SKIP1
$      IF DRAWER_NAME .EQS. 'WHICHLAST'_DRAWER THEN GOTO D_GOOD
$!
$! Check the filename for validity
$!
$SKIP1:
$      LEN = 'F$LENGTH(DRAWER_NAME)'
$      IF LEN .GT. 9 THEN GOTO D_BAD
$!
$      @'OATS_DIRECTORY'NAMEGOOD DRAWER_NAME
$      IF $STATUS .EQ. 1 THEN GOTO D_GOOD
$! The filename was not in the list of valid names, so it is a new one.
$! Check for a valid name.
$DRVAL:
$      LEN = LEN -1
$      CHAR := "F$EXTRACT(LEN,1,DRAWER_NAME)"
$      IF CHAR .GES. "0" .AND. CHAR .LES. "9" .OR.-
$              CHAR .GES. "A" .AND. CHAR .LES. "Z" .OR.-
$              CHAR .GES. "a" .AND. CHAR .LES. "z" -
$              THEN VALID = 1
$      IF VALID .EQ. 1 .AND. LEN .GT. 0 THEN GOTO DRVAL
$      IF VALID .EQ. 1 THEN GOTO NEW_DR
$!
$! Beep the user and let him try again.
$!
$D_BAD:
$      SAY MOVE,INQM1,DIRECT,BEEP,BEEP
$      GOTO DO_DRAWER
$!

```

```

$NEW_DR:
$      CREATE_NEW = 1
$!
$D_GOOD:
$      SAY MOVE,DR_PLACE,DIRECT,"      "
$      SAY MOVE,DR_PLACE,DIRECT,REVERSE,DRAWER_NAME,ATTRIBUTES_OFF
$      IF WHICHLAST.EQS. "" THEN GOTO SKIP2
$      IF DRAWER_NAME.NES. 'WHICHLAST'_DRAWER THEN-
$          SAY MOVE,FI_PLACE,DIRECT,REVERSE,"      "
$      IF DRAWER_NAME.NES. 'WHICHLAST'_DRAWER THEN-
$          SAY MOVE,FO_PLACE,DIRECT,"      "
$      SAY MOVE,CO_PLACE,DIRECT,"      ",ATTRIBUTES_OFF
$SKIP2:
$      THIS_DIR := 'F$DIRECTORY()'
$      LEN = 'F$LOCATE("]",THIS_DIR)'
$      HERE := 'F$EXTRACT(0,LEN,THIS_DIR)'
$      NEWLINE := "' 'HERE'. 'DRAWER_NAME'"
$      SAY MOVE,"3;35",DIRECT,BLANK
$      IF CREATE_NEW.EQ. 1 THEN GOTO NO_FI_LIST
$      SAY ATTRIBUTES_OFF,MOVE,"3;35",DIRECT,REVERSE,"Files are:"-
$          ,ATTRIBUTES_OFF
$      DIR/OUTPUT=DIRECTORY.LST/NOHEADING/NOTRAILING 'NEWLINE']*.DIR
$      @'OATS_DIRECTORY'SHOWDIRS ".DIR"
$      GOTO DO_FILE
$!
$NO_FI_LIST:
$      @'OATS_DIRECTORY'SHOWDIRS
$!
$! Now need to get the file name
$!
$DO_FILE:
$      SAY MOVE,INQM1,DIRECT,BLANK,MOVE,INQ,DIRECT
$      INQUIRE/GLOBAL FILE_NAME "Type in file name"
$      IF FILE_NAME.EQS. "HELP" THEN GOTO NEEDS_HELP
$      IF FILE_NAME.EQS. "QUIT" THEN EXIT 3
$!
$! If filename was null, no last filename, if there was one.
$!
$      IF FILE_NAME.EQS. "" .AND. CREATE_NEW.EQ. 1 THEN GOTO FI_BAD
$      IF WHICHLAST.EQS. "" THEN GOTO SKIP3
$      IF FILE_NAME.EQS. "" THEN FILE_NAME == 'WHICHLAST'_FILE
$      IF FILE_NAME.EQS. 'WHICHLAST'_FILE THEN GOTO FI_GOOD
$!
$! Check the filename for validity
$!
$SKIP3:
$      LEN = 'F$LENGTH(FILE_NAME)'
$      IF LEN.GT. 9 THEN GOTO FI_BAD
$      IF VALID_NAMES.EQS. "" THEN GOTO FIVAL
$!

```

```

$      @'OATS_DIRECTORY'NAMEGOOD FILE_NAME
$      IF $STATUS .EQ. 1 THEN GOTO FI_GOOD
$! The filename was not in the list of valid names, so it is a new one.
$! Check for a valid name.
$!
$FIVAL:
$      LEN = LEN -1
$      CHAR := "F$EXTRACT(LEN,1,FILE_NAME)"
$      IF CHAR .GES. "0" .AND. CHAR .LES. "9" .OR.-
          CHAR .GES. "A" .AND. CHAR .LES. "Z" .OR.-
          CHAR .GES. "a" .AND. CHAR .LES. "z" -
              THEN VALID = 1
$      IF VALID .EQ. 1 .AND. LEN .GT. 0 THEN GOTO FIVAL
$      IF VALID .EQ. 1 THEN GOTO NEW_FI
$!
$! Beep the user and let him try again.
$!
$FI_BAD:
$!
$      SAY MOVE,INQM1,DIRECT,BEEP,BEEP
$      GOTO DO_FILE
$!
$NEW_FI:
$      CREATE_NEW = 1
$FI_GOOD:
$      SAY MOVE,FI_PLACE,DIRECT,REVERSE,FILE_NAME,ATTRIBUTES_OFF
$      IF WHICHLAST .EQS. "" THEN GOTO SKIP4
$      IF FILE_NAME .NES. 'WHICHLAST' FILE THEN-
          SAY MOVE,FO_PLACE,DIRECT,REVERSE,"          ",ATTRIBUTES_OFF
$SKIP4:
$      NEWLINE := "'NEWLINE'.'FILE_NAME'"
$      SAY MOVE,"3;35",DIRECT,BLANK
$      IF CREATE_NEW .EQ. 1 THEN GOTO NO_FO_LIST
$      SAY ATTRIBUTES_OFF,MOVE,"3;35",DIRECT,REVERSE,"Folders are:"-
          ,ATTRIBUTES_OFF
$      DIR/OUTPUT=DIRECTORY.LST/NOHEADING/NOTAILING 'NEWLINE']*.DIR
$      @'OATS_DIRECTORY'SHOWDIRS ".DIR"
$      GOTO DO_FOLDER
$!
$NO_FO_LIST:
$      @'OATS_DIRECTORY'SHOWDIRS
$!
$! Now need to get the folder name
$!
$DO_FOLDER:
$      SAY MOVE,INQM1,DIRECT,BLANK,MOVE,INQ,DIRECT
$      INQUIRE/GLOBAL FOLDER_NAME "Type in folder name"
$      IF FOLDER_NAME .EQS. "HELP" THEN GOTO NEEDS_HELP
$      IF FOLDER_NAME .EQS. "QUIT" THEN EXIT 3
$!

```

```

$! If foldername was null, no last filename, if there was one.
$!
$      IF FOLDER_NAME .EQS. "" .AND. CREATE_NEW .EQ. 1 THEN GOTO FO_BAD
$      IF WHICHLAST .EQS. "" THEN GOTO SKIP5
$      IF FOLDER_NAME .EQS. "" THEN FOLDER_NAME == 'WHICHLAST'_FOLDER
$      IF FOLDER_NAME .EQS. 'WHICHLAST'_FOLDER THEN GOTO FO_GOOD
$!
$! Check the filename for validity
$!
$SKIP5:
$      LEN = 'F$LENGTH(FOLDER_NAME)'
$      IF LEN .GT. 9 THEN FOTO FI_BAD
$      IF VALID_NAMES .EQS. "" THEN GOTO FOVAL
$      @'OATS_DIRECTORY'NAMEGOOD FOLDER_NAME
$      IF $STATUS .EQ. 1 THEN GOTO FO_GOOD
$! The filename was not in the list of valid names, so it is a new one.
$! Check for a valid name.
$!
$FOVAL:
$      LEN = LEN -1
$      CHAR := "F$EXTRACT(LEN,1,FOLDER_NAME)"
$      IF CHAR .GES. "0" .AND. CHAR .LES. "9" .OR.-
$          CHAR .GES. "A" .AND. CHAR .LES. "Z" .OR.-
$          CHAR .GES. "a" .AND. CHAR .LES. "z" -
$          THEN VALID = 1
$      IF VALID .EQ. 1 .AND. LEN .GT. 0 THEN GOTO FOVAL
$      IF VALID .EQ. 1 THEN GOTO NEW_FO
$!
$! Beep the user and let him try again.
$!
$FO_BAD:
$      SAY MOVE,INQM1,DIRECT,BEEP,BEEP
$      GOTO DO_FOLDER
$!
$!
$NEW_FO:
$      CREATE_NEW =1
$FO_GOOD:
$      SAY MOVE,FO_PLACE,DIRECT,REVERSE,FOLDER_NAME,ATTRIBUTES_OFF
$      NEWLINE := "'NEWLINE'."FOLDER_NAME"
$      SAY MOVE,"3;35",DIRECT,BLANK
$!
$! Now need to get the document name
$!
$!
$! If this directory path already exists, chances are that
$! the user could type in an already existing document name.
$! Therefore, show him the list of already existing contents
$! to help cut down on errors.
$!

```

```

$      IF CREATE_NEW .EQ. 1 THEN GOTO NO_CO_LIST
$      DIR/OUTPUT=DIRECTORY.LST/NOHEADING/NOTRAILING/VERSIONS=1-
$              'NEWLINE']*,OFF
$      @'OATS_DIRECTORY'SHOWDIRS ".OFF;";
$      WORD_NEW := "Type in 'BLINK'NEW 'ATTRIBUTES_OFF'contents name"
$      GOTO DO_CONTENTS
$NO_CO_LIST:
$      @'OATS_DIRECTORY'SHOWDIRS
$      WORD_NEW := "Type in contents name"
$DO_CONTENTS:
$      SAY MOVE,INQM1,DIRECT,BLANK,MOVE,INQ,DIRECT
$      INQUIRE/GLOBAL CONTENTS_NAME "'WORD_NEW'"
$      IF CONTENTS_NAME .EQS. "HELP" THEN GOTO NEEDS_HELP
$      IF CONTENTS_NAME .EQS. "QUIT" THEN EXIT 3
$      IF CONTENTS_NAME .EQS. "" THEN GOTO CO_BAD
$!
$! Check the filename for validity
$!
$      LEN = 'F$LENGTH(CONTENTS_NAME)'
$      IF LEN .GT. 9 THEN GOTO CO_BAD
$COVAL:
$      LEN = LEN -1
$      CHAR := "F$EXTRACT(LEN,1,CONTENTS_NAME)"
$      IF CHAR .GES. "0" .AND. CHAR .LES. "9" .OR.-
$              CHAR .GES. "A" .AND. CHAR .LES. "Z" .OR.-
$              CHAR .GES. "a" .AND. CHAR .LES. "z" -
$              THEN VALID = 1
$      IF VALID .EQ. 1 .AND. LEN .GT. 0 THEN GOTO COVAL
$      @'OATS_DIRECTORY'NAMEGOOD CONTENTS_NAME
$      IF $STATUS .EQ. 3 THEN GOTO NEW_CO
$!
$! The file already exists. Let the user know.
$!
$ALREADY_EXISTS:
$      SAY MOVE,INQM1,DIRECT,"Document already exists."
$      WAIT 00:00:03
$!
$! Beep the user and let him try again.
$!
$CO_BAD:
$      SAY MOVE,INQM1,DIRECT,BEEP,BEEP
$      GOTO DO_CONTENTS
$!
$NEW_CO:
$      SAY MOVE,CO_PLACE,DIRECT,REVERSE,CONTENTS_NAME,ATTRIBUTES_OFF
$!
$! Now create the directory if it is a new one.
$!
$      IF CREATE_NEW .EQ. 1 THEN CRE/DIR 'NEWLINE']
$!

```

```

$! Check to see that the file does not already exist.
$!
$      NEWLINE := "'NEWLINE']'CONTENTS_NAME'.OFF"
$      NEW_DRAWER := 'DRAWER_NAME'
$      NEW_FILE := 'FILE_NAME'
$      NEW_FOLDER := 'FOLDER_NAME'
$      NEW_CONTENTS := 'CONTENTS_NAME'
$      WHICHLAST := "NEW"
$!
$! Now have new valid document name.
$!
$
$      EXIT 1
$!
$!
$NEEDS_HELP:
$      OLD_TITLE := "DOCUMENT"
$      @'OATS_DIRECTORY'HELPMMSG
$      IF $STATUS .EQ. 3 THEN EXIT 3
$      GOTO BEGIN_AGAIN
$!
$! Here on Control_Y
$!
$Y_EXIT:
$      WHICHLAST := ""
$      EXIT 3

$!*****

$!*****

$!***** OFFICE *****!$
$! This is the mail program of OATS. It controls the flow of
$! the system by determining what the user has requested -- a program or
$! another menu -- and responding accordingly.
$      ON CONTROL_Y THEN GOTO GETOUT
$!
$! Set up definitions for VT100 screen commands.
$!
$      @'OATS_DIRECTORY'SETTINGUP
$      @'OATS_DIRECTORY'SETSYMBOL
$!
$! Display reminders for today, if any.
$!
$      @'OATS_DIRECTORY'TICKLE
$!
$!
$! Initialize menu choices and program choices
$! The user may be personalized, but fall back on system list.
$!

```

```

$      @'OATS_DIRECTORY'FILEFOUND.CHOICELST.COM
$      STAT = $STATUS
$      IF STAT .EQ. 3 THEN @'OATS_DIRECTORY'CHOICELST
$      IF STAT .EQ. 1 THEN @CHOICELST
$!
$!
$!      initialization
$!
$      EXIT_MENU_NUMBER := "01"
$      CHOICE_NUMBER := "01"
$      PICKNN := 'PICK01'
$GET_MENU:
$      OLD_MENU := 'CHOICE_NUMBER'
$!
$!      Show the menu selected and get the user's new CHOICE_NUMBER
$!
$      @'OATS_DIRECTORY'SHOWMENU
$!
$GET_PROGRAM:
$      IF M_OR_P .EQS. "M" THEN GOTO GET_MENU
$      TITLE_LENGTH := 'F$EXTRACT(1,2,PICKNN)'
$      EXIT_MENU_NUMBER := 'F$EXTRACT(TITLE_LENGTH+3,2,PICKNN)'
$      IF EXIT_MENU_NUMBER .EQS. "00" THEN EXIT_MENU_NUMBER := 'OLD_MENU'
$      T = 'F$EXTRACT(TITLE_LENGTH+5,2,PICKNN)'
$      OLD_TITLE := 'TITLE'
$      TITLE := 'F$EXTRACT(TITLE_LENGTH+7,T,PICKNN)'
$      HEADER := "OFF"
$!
$!      Now execute the program called for by calling the DCL command file
$!
$      IF CHOICE_NUMBER .LES. "50" THEN -
$              @'OATS_DIRECTORY''TITLE' 'CHOICE_NUMBER'
$      IF CHOICE_NUMBER .GTS. "50" THEN -
$              @'TITLE' 'CHOICE_NUMBER'
$!
$!      Now set up to call for next menu
$!
$      IF EXIT_MENU_NUMBER .EQS. "00" THEN GOTO GETOUT
$      CHOICE_NUMBER := 'EXIT_MENU_NUMBER'
$!
$      GOTO GET_MENU
$!
$!      Here when control_y typed.
$!
$GETOUT:
$      SET MESSAGE/NOFACILITY/NOTEXT/NOSEVERITY/NOIDENT
$      DELETE *.TMP;*
$      SET MESSAGE/TEXT/SEVERITY/FACILITY/IDENT
$      SAY CLEAR_SCREEN
$      EXIT

```

\$!*****

\$!*****

\$!***** PICKMENU *****!

\$ SET NOVERIFY

\$!

\$! Symbol definitions for menu selection. Each symbol has the format

\$!

\$! PICKnn

\$!

\$! where nn is the choice number of the menu. The format of each
\$! MENU entry is:

\$!

\$! MnnNAMEmppaabbcc...zz

\$!

\$! where m indicates MENU type choice

\$! nn = the number of characters in the menu name,

\$! NAME = the name of the menu

\$! mm = the number of the menu to be shown upon exit

\$! pp = the number of choices in the menu

\$! aa,bb,cc,...,zz = the menu or program numbers of the choices

\$!

\$! The format of each PROGRAM entry is:

\$!

\$! PnnNAMEmmqqCNAME

\$!

\$! where P indicates PROGRAM type choice

\$! nn = the number of characters in the choice name,

\$! NAME = the name of the selection to be run

\$! mm = the number of the menu to be shown upon exit

\$! qq = the number of characters in the name of the COM
\$! file to be run

\$! CNAME = the name of the COM file to be executed

\$!

\$! PICK01 := "M04MAIN000703020906052549"

\$! PICK02 := "M09REMINDERS01052204212549"

\$! PICK03 := "M04MAIL0106081407202549"

\$! PICK04 := "P03ADD0208REMINDEr"

\$! PICK05 := "M14WORD PROCESSOR010424262549"

\$! PICK06 := "P10CALCULATOR0104CALC"

\$! PICK07 := "P18EXTERNAL MAIL LIST03MAILPROG"

\$! PICK08 := "P14READ/SEND MAIL0308MAILPROG"

\$! PICK09 := "M04FILE010713151719232549"

\$! PICK10 := "M16FORMATTING STYLE0506111216182549"

\$! PICK11 := "P06LETTER0508WORDPROC"

\$! PICK12 := "P04MEMO0508WORDPROC"

\$! PICK13 := "P04VIEW0907FILEDOC"

\$! PICK14 := "P16OFFICE MAIL LIST0308MAILPROG"

```

$! PICK15 := "P05PRINT0907FILEDOC"
$! PICK16 := "P08STANDARD0508WORDPROC"
$! PICK17 := "P04MOVE0907FILEDOC"
$! PICK18 := "P04NONE0508WORDPROC"
$! PICK19 := "P09DUPLICATE0907FILEDOC"
$! PICK20 := "P09SEND MEMO0308MAILPROG"
$! PICK21 := "P05ERASE0208REMINDER"
$! PICK22 := "P04LOOK0208REMINDER"
$! PICK23 := "P10THROW AWAY0907FILEDOC"
$! PICK24 := "P12NEW DOCUMENT0508WORDPROC"
$! PICK25 := "P04QUIT0004QUIT"
$! PICK26 := "P12OLD DOCUMENT0508WORDPROC"
$! PICK49 := "P04HELP0007HELPMSG"
$!!
$!! GET THE CHOICE OF MENU
$!
$ SYM := PICK'P1'
$ PICK := '&SYM'
$ TITLE_LENGTH == 'F$EXTRACT(1,2,PICK)'
$ TITLE := 'F$EXTRACT(3,TITLE_LENGTH,PICK)'
$ NEXT_NUM = TITLE_LENGTH+3
$ EXIT_MENU_NUMBER := 'F$EXTRACT(NEXT_NUM,2,PICK)'
$ NEXT_NUM = NEXT_NUM + 2
$ NUM_ENTRIES == 'F$EXTRACT(NEXT_NUM,2,PICK)'
$ IF NUM_ENTRIES .LT. 1 THEN GOTO DONE
$ CHOICE_1_NUM := 'F$EXTRACT(NEXT_NUM+2,2,PICK)'
$ SYM := PICK'CHOICE_1_NUM'
$ SYM1 := '&SYM'
$ LEN = 'F$EXTRACT(1,2,SYM1)'
$ CHOICE_1 := 'F$EXTRACT(3,LEN,SYM1)'
$ NEXT_NUM = NEXT_NUM + 2
$!
$ IF NUM_ENTRIES .LT. 2 THEN GOTO DONE
$ CHOICE_2_NUM := 'F$EXTRACT(NEXT_NUM+2,2,PICK)'
$ SYM := PICK'CHOICE_2_NUM'
$ SYM1 := '&SYM'
$ LEN = 'F$EXTRACT(1,2,SYM1)'
$ CHOICE_2 := 'F$EXTRACT(3,LEN,SYM1)'
$ NEXT_NUM = NEXT_NUM + 2
$!
$ IF NUM_ENTRIES .LT. 3 THEN GOTO DONE
$ CHOICE_3_NUM := 'F$EXTRACT(NEXT_NUM+2,2,PICK)'
$ SYM := PICK'CHOICE_3_NUM'
$ SYM1 := '&SYM'
$ LEN = 'F$EXTRACT(1,2,SYM1)'
$ CHOICE_3 := 'F$EXTRACT(3,LEN,SYM1)'
$ NEXT_NUM = NEXT_NUM + 2
$!
$ IF NUM_ENTRIES .LT. 4 THEN GOTO DONE
$ CHOICE_4_NUM := 'F$EXTRACT(NEXT_NUM+2,2,PICK)'

```

```

$      SYM := PICK'CHOICE_4_NUM'
$      SYM1 := '&SYM'
$      LEN = 'F$EXTRACT(1,2,SYM1)'
$      CHOICE_4 := 'F$EXTRACT(3,LEN,SYM1)'
$      NEXT_NUM = NEXT_NUM + 2
$!
$      IF NUM_ENTRIES .LT. 5 THEN GOTO DONE
$      CHOICE_5_NUM := 'F$EXTRACT(NEXT_NUM+2,2,PICK)'
$      SYM := PICK'CHOICE_5_NUM'
$      SYM1 := '&SYM'
$      LEN = 'F$EXTRACT(1,2,SYM1)'
$      CHOICE_5 := 'F$EXTRACT(3,LEN,SYM1)'
$      NEXT_NUM = NEXT_NUM + 2
$!
$      IF NUM_ENTRIES .LT. 6 THEN GOTO DONE
$      CHOICE_6_NUM := 'F$EXTRACT(NEXT_NUM+2,2,PICK)'
$      SYM := PICK'CHOICE_6_NUM'
$      SYM1 := '&SYM'
$      LEN = 'F$EXTRACT(1,2,SYM1)'
$      CHOICE_6 := 'F$EXTRACT(3,LEN,SYM1)'
$      NEXT_NUM = NEXT_NUM + 2
$!
$      IF NUM_ENTRIES .LT. 7 THEN GOTO DONE
$      CHOICE_7_NUM := 'F$EXTRACT(NEXT_NUM+2,2,PICK)'
$      SYM := PICK'CHOICE_7_NUM'
$      SYM1 := '&SYM'
$      LEN = 'F$EXTRACT(1,2,SYM1)'
$      CHOICE_7 := 'F$EXTRACT(3,LEN,SYM1)'
$      NEXT_NUM = NEXT_NUM + 2
$!
$      IF NUM_ENTRIES .LT. 8 THEN GOTO DONE
$      CHOICE_8_NUM := 'F$EXTRACT(NEXT_NUM+2,2,PICK)'
$      SYM := PICK'CHOICE_8_NUM'
$      SYM1 := '&SYM'
$      LEN = 'F$EXTRACT(1,2,SYM1)'
$      CHOICE_8 := 'F$EXTRACT(3,LEN,SYM1)'
$      NEXT_NUM = NEXT_NUM + 2
$!
$      IF NUM_ENTRIES .LT. 9 THEN GOTO DONE
$      CHOICE_9_NUM := 'F$EXTRACT(NEXT_NUM+2,2,PICK)'
$      SYM := PICK'CHOICE_9_NUM'
$      SYM1 := '&SYM'
$      LEN = 'F$EXTRACT(1,2,SYM1)'
$      CHOICE_9 := 'F$EXTRACT(3,LEN,SYM1)'
$!
$DONE:
$      IF NUM_ENTRIES .LT. 9 THEN CHOICE_9 := ""
$      IF NUM_ENTRIES .LT. 8 THEN CHOICE_8 := ""
$      IF NUM_ENTRIES .LT. 7 THEN CHOICE_7 := ""
$      IF NUM_ENTRIES .LT. 6 THEN CHOICE_6 := ""

```

```

$      IF NUM_ENTRIES .LT. 5 THEN CHOICE_5 := ""
$      IF NUM_ENTRIES .LT. 4 THEN CHOICE_4 := ""
$      IF NUM_ENTRIES .LT. 3 THEN CHOICE_3 := ""
$      IF NUM_ENTRIES .LT. 2 THEN CHOICE_2 := ""
$      EXIT

$!*****

$!*****

$!***** QUIT *****!$
$! This routine is reached whenever QUIT is chosen from a menu. It
$! looks at the exit menu number for the picknn entry from which
$! the current menu was called. (i.e. it goes back one menu level).
$! If the EXIT_MENU_NUMBER is "00" then an exit from the MAIN menu
$! is being requested. In that case, the user will be logged off
$! if he does not have privileges to use VAX/VMS.
$!
$      HEADER := "ON"      !Don't clear the header.
$      SYM := PICK 'EXIT_MENU_NUMBER'
$      PICKNN := '&SYM'
$      L == 'F$EXTRACT(1,2,PICKNN)'
$      EXIT_MENU_NUMBER := 'F$EXTRACT(L+3,2,PICKNN)'
$      IF EXIT_MENU_NUMBER .EQS. "00" .AND. VMS_PRIV .EQS. "NO" THEN LOGOFF
$      SET TERM/NOESCAPE
$      SAY MOVE,"1;1",DIRECT,CLEAR_SCREEN
$      EXIT

$!*****

$!*****

$!***** REMINDER *****!$
$! This program allows the user to maintain a file of messages.
$! These messages ( up to 70 characters each) are stored in
$! the file REMINDER.DTA in the user's directory. The messages
$! are stored chronologically according to date and the date
$! in the format dd-mmm-yyyy is stored at the beginning of each
$! message. Reminders cannot be entered for days already passed.
$! Likewise, all messages prior to the current day are automatically
$! deleted.
$! The options the user has (from the OATS menu) are as follows:
$!      ADD      - Add a message for the date. If others already exist
$!                  for this date, list the others and put this one
$!                  at the bottom.
$!      ERASE    - Display the messages for the date (they are numbered)
$!                  and ask for the number of the one to erase. This
$!                  is accomplished by writing the messages for the
$!                  date in a temporary file, reading them back to
$!                  delete the one desired, then writing them into

```

```

$!           the output file.
$! LOOK      - Display the messages for that date.
$!
$   ON CONTROL_Y THEN GOTO LEAVE_FAST
$   @'DATS_DIRECTORY'FILEFOUND REMINDER.DTA
$   IF $STATUS .EQ. 3 THEN CREATE REMINDER.DTA
$   SAY CLEAR_SCREEN
$   TODAY(0,11) := "'F$TIME()'"
$   MONTH := "'F$EXTRACT(3,3,TODAY)'"
$   M = 'MONTH'SYM
$   TODAY_NUMBER := -
$   "'F$EXTRACT(9,2,TODAY)'"'F$EXTRACT(2,2,M)'"'F$EXTRACT(0,2,TODAY)'"
$   IF P1 .EQS. "04" THEN WHY_HERE := "ADD"
$   IF P1 .EQS. "21" THEN WHY_HERE := "ERASE"
$   IF P1 .EQS. "22" THEN WHY_HERE := "LOOK"
$   SAY MOVE,"24;1",DIRECT,REVERSE,WHY_HERE,MOVE,"24;65",DIRECT,"Reminder"-
$   ,ATTRIBUTES_OFF
$BEGIN:
$   @'DATS_DIRECTORY'GETDATE
$   IF $STATUS .EQ. 3 THEN GOTO LEAVE_FAST
$   SAY MOVE,"1;20",SCROLL
$   MONTH = "'F$EXTRACT(3,3,NEWDATE)'"
$   M = 'MONTH'SYM
$   NEWDATE_NUMBER := -
$   "'F$EXTRACT(9,2,NEWDATE)'"'F$EXTRACT(2,2,M)'"'F$EXTRACT(0,2,NEWDATE)'"
$   IF TODAY_NUMBER .GTS. NEWDATE_NUMBER THEN GOTO YESTERDAY
$   NOW := "'F$TIME()'"
$   CLEAN_UP_TIME := "'F$EXTRACT(0,11,NOW)'"':"'F$EXTRACT(12,11,NOW)'"
$   OPEN/READ RD REMINDER.DTA
$   IF WHY_HERE .NES. "LOOK" THEN OPEN/WRITE WT REMINDER.DTA
$   ON CONTROL_Y THEN GOTO LEAVE_CLOSE
$   EOF = 0
$   P = 0
$   POS = 6
$   SAY BOLD,MOVE,"1;34",DIRECT,"Reminders for",MOVE,"3;14",DIRECT-
$   ,DOUBLE_WIDTH,NEWDATE,ATTRIBUTES_OFF
$NEXT_READ:
$   READ/END_OF_FILE=EOF_REACHED RD LINE
$   MON := "'F$EXTRACT(3,3,LINE)'"
$   M = 'MON'SYM
$   DATE_READ_NUMBER := -
$   "'F$EXTRACT(9,2,LINE)'"'F$EXTRACT(2,2,M)'"'F$EXTRACT(0,2,LINE)'"
$!
$! If "LOOK" and we've gone beyond date desired, no need to read the rest.
$!
$   IF NEWDATE_NUMBER .LTS. DATE_READ_NUMBER .AND. WHY_HERE .EQS. "LOOK" -
$   THEN GOTO EOF_REACHED
$!
$! We are either at the requested date or passed it (none there yet).
$! On ADD, go on and read until we've passed NEWDATE.

```

```

$! On ERASE, go on and read, but write in temp file if it's OATS_DIRECTORY's the
right day.
$!
$      IF NEWDATE_NUMBER .GES. DATE_READ_NUMBER THEN GOTO GO_ON
$      IF WHY_HERE .EQS. "ERASE" THEN GOTO EOF_NOT_REACHED
$      IF WHY_HERE .EQS. "ADD" THEN GOTO PUT_HERE
$!
$!
$! This may be the day, but put it at the end of this day's entries. Display
$! the other entries for that day also.
$!
$GO_ON:
$!
$! Doing some writing of the files out again for ERASE and ADD.
$!
$! Go ahead and write the line if date not history or today and if
$! its not the date desired.
$!
$      IF WHY_HERE .NES. "LOOK" -
          .AND. DATE_READ_NUMBER .NES. NEWDATE_NUMBER -
          .AND. DATE_READ_NUMBER .GES. TODAY_NUMBER THEN WRITE WT LINE
$!
$! If date read is the date desired, ADD will write it in sequence.
$! ERASE will write it in a temporary file. Make sure the file is opened
$! if this is the first one (p = 0).
$!
$      IF DATE_READ_NUMBER .NES. NEWDATE_NUMBER THEN GOTO NEXT_READ
$      IF WHY_HERE .EQS. "ADD" THEN WRITE WT LINE
$      IF WHY_HERE .EQS. "ERASE" .AND. P .EQ. 0 -
          THEN OPEN/WRITE TMP REMINDER.TMP
$      IF WHY_HERE .EQS. "ERASE" THEN ON CONTROL_Y THEN GOTO LEAVE_CLOSE_TMP
$      IF WHY_HERE .EQS. "ERASE" THEN WRITE TMP LINE
$!
$! Display the line on the screen if we're at the right date.
$!
$      LEN = 'F$LENGTH(LINE)'
$      L := 'F$EXTRACT(12,LEN-12,LINE)'
$      P = POS -5
$      SAY MOVE,POS,";1",DIRECT,P,MOVE,POS,";5",DIRECT,L
$      POS = POS + 1
$      GOTO NEXT_READ
$!
$!
$!
$! Now get the new one in and copy the current line and the rest of the file.
$!
$EOF_REACHED:
$      EOF = 1
$EOF_NOT_REACHED:
$      IF WHY_HERE .NES. "ADD" .AND. P .EQ. 0 THEN SAY -

```

```

                MOVE,POS,";1",DIRECT,BOLD,"No reminders."
$!
$      IF WHY_HERE .EQS. "LOOK" THEN GOTO NO_PUT
$PUT_HERE:
$      IF WHY_HERE .EQS. "ERASE" .AND. P .NE. 0 THEN GOTO ERASE_IT
$      IF WHY_HERE .EQS. "ERASE" THEN GOTO ERASE_OK
$      SAY MOVE,POS,";1",DIRECT
$      INQUIRE MSG "'BOLD'Message'ATTRIBUTES_OFF'"
$      LEN = 'F$LENGTH(MSG)'
$      IF LEN .LT. 70 THEN GOTO MSG_OK
$      SAY MOVE,"24;1",DIRECT-
                ,"Message too long. Must be less than 70 characters."-
                ,MOVE,POS,";1",DIRECT,CLEAR_LINE
$      GOTO PUT_HERE
$!
$!
$ERASE_IT:
$      SAY MOVE,POS,";1",DIRECT
$      INQUIRE MSG "'BOLD'Message number to erase'ATTRIBUTES_OFF'"
$      IF MSG .GTS. P THEN GOTO INVALID_NUMBER
$      CLOSE TMP
$      OPEN/READ TMP REMINDER.TMP
$      M = 0
$ER_LOOP:
$      M = M + 1
$      READ TMP TMLINE
$      IF M .NE. MSG THEN WRITE WT TMLINE
$      IF M .NE. P THEN GOTO ER_LOOP
$      CLOSE TMP
$      ON CONTROL_Y THEN GOTO LEAVE_CLOSE
$      DELETE REMINDER.TMP;*
$      GOTO ERASE_OK
$!
$INVALID_NUMBER:
$      SAY "Can't delete message number ",msg
$      GOTO ERASE_IT
$MSG_OK:
$      WRITE WT NEWDATE,";",MSG
$ERASE_OK:
$      IF EOF .EQ. 1 THEN GOTO LEAVE_OK
$WRITE_REST:
$      IF DATE_READ_NUMBER .GES. TODAY_NUMBER THEN WRITE WT LINE
$      READ/END_OF_FILE=LEAVE_OK RD LINE
$      DATE_READ := "'F$EXTRACT(0,11,LINE)'"
$      GOTO WRITE_REST
$!
$!
$! Here is leaving under duress. Destroy the copy just created.
$LEAVE_CLOSE_TMP:
$      IF WHY_HERE .EQS. "ERASE" THEN CLOSE TMP

```

```

$LEAVE_CLOSE:
$      IF WHY_HERE .NES. "LOOK" THEN CLOSE WT
$      CLOSE RD
$      IF WHY_HERE .NES. "LOOK" THEN -
$              DELETE/MODIFY/SINCE='CLEAN_UP_TIME' REMINDER.DTA;#

$LEAVE_FAST:
$      SAY CLEAR_SCREEN,MOVE,"1;24",SCROLL
$      EXIT 3
$!
$!
$! Here if leaving normally. Keep only two reminder copies around.
$!
$LEAVE_OK:
$      CLOSE WT
$NO_PUT:
$      CLOSE RD
$      SET MESSAGE/NOTEXT/NOSEVERITY/NOIDENT/NOFACILITY
$      PURGE/KEEP:2 REMINDER.DTA
$      SET MESSAGE/TEXT/SEVERITY/IDENT/FACILITY
$      SAY MOVE,"22;1",DIRECT
$      INQUIRE X "'BOLD'Press RETURN to erase screen'"ATTRIBUTES_OFF"
$      SAY CLEAR_SCREEN,MOVE,"1;24",SCROLL
$      EXIT 1
$!
$! The date entered by GETDATE was before today. Who can use old reminders!!!
$!
$YESTERDAY:
$      SAY BEEP,MOVE,"22;1",DIRECT-
$              ,BOLD,"That date has already passed. Try again.",ATTRIBUTES_OFF
$      GOTO BEGIN

$!*****

$!*****

$!***** ROOMLEFT *****!$
$! This program checks the amount of available space on the users
$! and compares it to the amount required (argument P1). EXIT 1
$! is taken if there is room, EXIT 3 if not.
$!
$      REQUIRED == 'P1'
$      @'OATS_DIRECTORY'CHKQUOTA/OUT=CHKQUOTA.TMP
$      OPEN/READ INFILE CHKQUOTA.TMP
$      READ INFILE LINE
$      CLOSE INFILE
$      N1 = 'F$LOCATE("used",LINE)'
$      N2 = 'F$LOCATE("available",LINE)'
$      LEFT = 'F$EXTRACT(N1+6,N2-N1-7,LINE)'
$      NUM = 'F$INTEGER(LEFT)'
$      DELETE CHKQUOTA.TMP;#

```

```
$      IF NUM .GT. REQUIRED THEN EXIT 1
$      EXIT 3
```

```
SYS$USER:[JANSEN.OATS]SETSYMBOL.COM;1
```

```
$!
$! Initialize symbols.
$!
$      HEADER := "OFF"
$      WHICHLAST := ""
$      NAMELINE := ""
$      NAMELIST := ""
$      LAST_DRAWER := "      "
$      LAST_FILE := "      "
$      LAST_FOLDER := "      "
$      LAST_CONTENTS := "      "
$!
$! Global symbols for menu selection
$!
$      M_OR_P := "M"
$      TITLE := "MAIN"
$      TITLE_LENGTH == 4
$      CHOICE_1 := ""
$      CHOICE_2 := ""
$      CHOICE_3 := ""
$      CHOICE_4 := ""
$      CHOICE_5 := ""
$      CHOICE_6 := ""
$      CHOICE_7 := ""
$      CHOICE_8 := ""
$      CHOICE_9 := ""
$      CHOICE_1_NUM := "00"
$      CHOICE_2_NUM := "00"
$      CHOICE_3_NUM := "00"
$      CHOICE_4_NUM := "00"
$      CHOICE_5_NUM := "00"
$      CHOICE_6_NUM := "00"
$      CHOICE_7_NUM := "00"
$      CHOICE_8_NUM := "00"
$      CHOICE_9_NUM := "00"
$!
$!
$      END_LENGTH := "00"
$      NUM_ENTRIES == 0
$!
$! Miscellaneous
$!
$      VALID_NAME := ""
$      BEEP[0,32] == %X07
$      ERROR_CODE_LIMIT[0,32] == %X0A
```

```

$      SAY := "WRITE SYS$OUTPUT"
$
$      JANSYM := "3101"
$      FEBSYM := "2802"
$      MARSYM := "3103"
$      APRSYM := "3004"
$      MAYSYM := "3105"
$      JUNSYM := "3006"
$      JULSYM := "3107"
$      AUGSYM := "3108"
$      SEPSYM := "3009"
$      OCTSYM := "3110"
$      NOVSYM := "3011"
$      DECSYM := "3112"

$!*****

$!*****

$!***** SETTINGUP *****!$
$!
$! This program sets up the various symbol values for VT-100 terminal.
$!
$      SET TERM/VT100/NOESCAPE
$!
$!
$!      Symbol definitions -- global
$!
$!      Modes to Set and reset Cursor Control
$!
$      MODE[0,32] == %X313F5B1B
$      APPLICATION := "h"
$      CURSOR := "1"
$!
$!      Auxillary keypad control
$!
$      KEY_PAD_NUM[0,32] == %X3E1B
$      KEY_PAD_APPL[0,32] == %X3D1B
$!
$      ESCAPE[0,32] == %X1B
$      CLEAR_SCREEN[0,32] == %X4A325B1B
$      CLEAR_TO_CURSOR[0,32] == %X4A315B1B
$      CLEAR_TO_EOL[0,32] == %X4B305B1B
$      CLEAR_LINE_TO_CURSOR[0,32] == %X4B315B1B
$      CLEAR_LINE[0,32] == %X4B325B1B
$      CLEAR_TO_EDSCREEN[0,32] == %X4A305B1B
$!
$      ENTER[0,32] == %X0D
$!
$!      Line size commands

```

```

$!
$      DOUBLE_HEIGHT_TOP[0,32] == %X33231B
$      DOUBLE_HEIGHT_BOTTOM[0,32] == %X34231B
$      SINGLE[0,32] == %X35231B
$      DOUBLE_WIDTH[0,32] == %X36231B
$!
$!      Cursor position report
$!
$      CURSOR_REPORT[0,32] == %X6E365B1B
$!
$!      Character attributes
$!
$      ATTRIBUTES_OFF[0,32] == %X6D305B1B
$      BOLD[0,32] == %X6D315B1B
$      UNDERSCORE[0,32] == %X6D345B1B
$      BLINK[0,32] == %X6D355B1B
$      REVERSE[0,32] == %X6D375B1B
$!
$!      Cursor movement
$!
$      MOVE[0,32] == %X5B1B
$      UP := "A"
$      DOWN := "B"
$      RIGHT := "C"
$      LEFT := "D"
$      DIRECT := "H"
$      INDEX_UP[0,32] == %X4D1B
$      INDEX_DOWN[0,32] == %X441B
$      SEE_CURSOR[0,32] == %X7673305B1B
$      HIDE_CURSOR[0,32] == %X7673315B1B
$      SAVE_CURSOR[0,32] == %X371B
$      RESTORE_CURSOR[0,32] == %X381B
$      SCROLL := "r"
$      REGULAR_CHAR_SET[0,32]==%0017
$      GRAPHICS_CHAR_SET[0,32]==%0016

$!*****

$!*****

$!***** SHOWDIRS *****!$
$! This program will read a file created by the DIRECTORY command
$! and read the filenames one at a time and print them on the
$! screen. If there is no P1 argument, the right half of the screen will
$! be cleared and exit taken. If the P1 argument is ".DIS" or ".MAI" then the
$! right half of the screen will not be cleared but the filenames in
$! DIRECTORY.LST will be displayed, as well as HELP and QUIT.
$!
$      SAY ATTRIBUTES_OFF
$      C = 33

```

```

$      L = 4
$      IF P1 .EQS. ".MAI" .OR. P1 .EQS. ".DIS" THEN GOTO NO_CLEAR
$!
$! Clear the right half of the screen.
$!
$CLSC:
$      SAY MOVE,L,";",C,DIRECT,CLEAR_TO_EOL
$      L = L + 1
$      IF L .LT. 16 THEN GOTO CLSC
$      VALID_NAMES := ""
$      IF P1 .EQS. "" THEN EXIT
$NO_CLEAR:
$      OPEN DIRIN DIRECTORY.LST
$      L = 4
$      C = 33
$!
$      NUM = 0
$      NAME := ""
$!
$MORE:
$      READ/END_OF_FILE=READ_DONE DIRIN NAME
$      IF NAME .EQS. "No files found" THEN GOTO READ_DONE
$!
$! GET RID OF ALL BUT THE FILE NAME
$!
$      BEGINN = 'F$LOCATE("J",NAME)'
$      LEN = 'F$LOCATE(P1,NAME)' - BEGINN - 1
$      WORD := 'F$EXTRACT(BEGINN+1,LEN,NAME)'
$!
$      IF WORD .EQS. "" THEN GOTO MORE
$!
$      NUM = NUM + 1
$      L = L + 1
$!
$      IF L .EQ. 16 THEN C = C + 12
$      IF L .EQ. 16 THEN L = 5
$!
$      VALID_NAMES := 'VALID_NAMES'F$CVUI(0,4,LEN)'WORD'
$      SAY MOVE,L,";",C,DIRECT,WORD
$      GOTO MORE
$!
$! All filenames listed. Now close out I/O and add "QUIT" and "HELP"
$! as choices
$!
$READ_DONE:
$      CLOSE DIRIN
$      DELETE DIRECTORY.LST;*
$      SAY MOVE,"17;33",DIRECT,BOLD,"HELP          QUIT",ATTRIBUTES_OFF
$      EXIT

```

```
$!*****
```

```
$!*****
```

```
$!***** SHOWMENU *****!
```

```
$! This program prints on the screen the menu requested by the user.
```

```
$! It then accepts the response from the user and sets up parameters
```

```
$! for return so that control is given the the selected program or
```

```
$! menu. The response expected is [char][char] where 'char' is an
```

```
$! alphabetic character.
```

```
$! If a control-Y is pressed, it is treated as if the user had
```

```
$! requested the QUIT choice on the screen.
```

```
$!
```

```
$!
```

```
$!      Clear screen and write border
```

```
$!
```

```
$      ON CONTROL_Y THEN GOTO GETOUT
```

```
$      SET TERM/ESCAPE
```

```
$      ERROR_COUNT == 0
```

```
$      IF HEADER .EQS. "ON" THEN GOTO NO_NEW_HEADER
```

```
$      SAY MOVE,"1;24",SCROLL,CLEAR_SCREEN
```

```
$      SAY ATTRIBUTES_OFF,MOVE,"1;1",DIRECT
```

```

$      TYPE SYS$INPUT
[H#3]wwqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqwwk
#4xxx                                     xxx
#3xxx                                     xxx
#4xxx                                     xxx
#3mvyqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqvj

$      SAY MOVE,"3;6",DIRECT,BOLD,REVERSE,"Office Automation Tool System"-
      ,MOVE,"4;6",DIRECT-
      ,"Office Automation Tool System"

$NO_NEW_HEADER:
$      IF HEADER .EQS. "ON" THEN SAY MOVE,"7;1",DIRECT,CLEAR_TO_EOSCREEN
$      HEADER := "ON"
$      @'DATS_DIRECTORY'PICKMENU 'CHOICE_NUMBER'
$      COLUMN = ( 40 - (TITLE_LENGTH + 4 ))/2
$      SAY ATTRIBUTES_OFF,MOVE,"18;18",SCROLL
$      SAY MOVE,"8;",COLUMN,DIRECT,TITLE,DOUBLE_HEIGHT_TOP," MENU"
$      SAY MOVE,"9;",COLUMN,DIRECT,TITLE,DOUBLE_HEIGHT_BOTTOM," MENU"
$      NUM = 1
$!
$!      Fill in the menu
$!
$NEXT_ITEM:
$      SYM := CHOICE 'F$CVUI(0,4,NUM)'
$      WHICH := '&SYM'
$      IF NUM .GT. NUM_ENTRIES THEN GOTO ALL_ON
$      KEY := 'F$EXTRACT(0,2,WHICH)'
$      END_LENGTH = 'F$LENGTH(WHICH)'-2
$      REST := 'F$EXTRACT(2,END_LENGTH,WHICH)'
$      LINE = NUM + 10
$      SAY BOLD,MOVE,LINE,";19",DIRECT,DOUBLE_WIDTH,KEY,ATTRIBUTES_OFF,REST
$      NUM = NUM + 1
$!
$      GOTO NEXT_ITEM
$!
$ ALL_ON:
$!      All choices are on the screen.
$!      Print user instructions at bottom of screen
$!
$!
$!      Now position cursor to wait for response.
$!
$GET_RESPONSE:
$      NUM = 1
$      SAY MOVE,"21;1",DIRECT,REVERSE,"_",ATTRIBUTES_OFF,-
      " Your choice -- two letters"
$      SAY MOVE,"20;1",DIRECT
$      SET NOON
$!
$      SET MESSAGE/NOTEXT/NOSEVERITY/NOFACILITY/NOIDENT

```

```

$      RUN 'OATS_DIRECTORY'ACCEPT      ! Get directions from us

$      STATUS[0,32] = '$STATUS'! Response is returned here using $EXIT_S
$      SET MESSAGE/TEXT/SEVERITY/FACILITY/IDENT
$      SET ON
$      IF STATUS .LES. ERROR_CODE_LIMIT THEN GOTO ERROR_ON_INPUT
$!
$      CHAR1 := 'F$EXTRACT(0,1,STATUS)'
$      CHAR2 := 'F$EXTRACT(1,1,STATUS)'
$!
$!      Two-character code typed in.
$!
$      CODE := 'CHAR1''CHAR2'
$!
$!      See if what was typed in is valid
$!
$MATCH_KEY:
$      IF NUM .GT. NUM_ENTRIES THEN GOTO ERROR_ON_INPUT
$      SYM := CHOICE_'F$CVUI(0,4,NUM)_'
$      WHICH := '&SYM'
$      KEY := 'F$EXTRACT(0,2,WHICH)'
$!
$!      KEY is the two-letter key
$!
$      IF CODE .EQS. KEY THEN GOTO GOT_CODE
$      NUM = NUM + 1
$      GOTO MATCH_KEY              ! Try the next one
$!
$!      Here when the matching key is found.  The value of NUM
$!      is used to determine the response.
$!
$GOT_CODE:
$      SYM := CHOICE_'F$CVUI(0,4,NUM)_'_NUM
$      CHOICE_NUMBER := '&SYM'
$LEAVE:
$      SYM := PICK'CHOICE_NUMBER'
$      PICKNN := '&SYM'
$!
$!      PICKNN is now equated to the proper symbol1 PICKnn where
$!      nn is the number of the menu or program chosen.
$!
$      M_OR_P := 'F$EXTRACT(0,1,PICKNN)'
$      SAY MOVE,"1;24",SCROLL
$      SET TERM/NOESCAPE/TYPE_AHEAD
$      EXIT
$!
$!
$!      Here CONTROL-Y was received.
$!

```

```

$GETOUT:
$      CHOICE_NUMBER := "25"
$      GOTO LEAVE
$!
$!
$!      Increment error counter.  If he's typed three in a row, give
$!      him the help message.
$!
$ERROR_ON_INPUT:
$      ERROR_COUNT == ERROR_COUNT + 1
$      IF ERROR_COUNT .LT. 3 THEN GOTO GET_RESPONSE
$!
$!      If third error, show help message and return to menu.
$!
$      CODE := "HE"
$      NUM = 1
$      GOTO MATCH_KEY
$!

$!*****

$!*****

$!***** TICKLE *****!$
$! This program supplements REMINDER.  It just does a LOOK for today.
$!
$      ON CONTROL_Y THEN GOTO LEAVE_FAST
$      @'OATS_DIRECTORY'FILEFOUND REMINDER.DTA
$      IF $STATUS .EQ. 3 THEN EXIT
$      TODAY[0,11] := "'F$TIME()'"
$      MONTH := "'F$EXTRACT(3,3,TODAY)'"
$      M = 'MONTH'SYM
$      TODAY_NUMBER := -
$      "'F$EXTRACT(9,2,TODAY)'"'F$EXTRACT(2,2,M)'"'F$EXTRACT(0,2,TODAY)'"
$      NEWDATE := 'TODAY'
$      NEWDATE_NUMBER := 'TODAY_NUMBER'
$BEGIN:
$      NOW := "'F$TIME()'"
$      CLEAN_UP_TIME := "'F$EXTRACT(0,11,NOW)'"':"'F$EXTRACT(12,11,NOW)'"
$      OPEN/READ RD REMINDER.DTA
$      ON CONTROL_Y THEN GOTO LEAVE_CLOSE
$      EOF = 0
$      P = 0
$      POS = 6
$NEXT_READ:
$      READ/END_OF_FILE=NO PUT RD LINE
$      MON := "'F$EXTRACT(3,3,LINE)'"
$      M = 'MON'SYM
$      DATE_READ_NUMBER := -
$      "'F$EXTRACT(9,2,LINE)'"'F$EXTRACT(2,2,M)'"'F$EXTRACT(0,2,LINE)'"

```

```

$!
$! If "LOOK" and we've gone beyond date desired, no need to read the rest.
$!
$      IF NEWDATE_NUMBER .LTS. DATE_READ_NUMBER THEN GOTO NO_PUT
$!
$! We are either at the requested date or passed it (none there yet).
$!
$      IF NEWDATE_NUMBER .GES. DATE_READ_NUMBER THEN GOTO GO_ON
$!
$! This may be the day, but put it at the end of this day's entries. Display
$! the other entries for that day also.
$!
$GO_ON:
$!
$      IF DATE_READ_NUMBER .NES. NEWDATE_NUMBER THEN GOTO NEXT_READ
$!
$! Display the line on the screen if we're at the right date.
$!
$      IF P .EQ. 0 THEN SAY CLEAR_SCREEN,BOLD,MOVE,"1;34",DIRECT-
          ,"Reminders for",MOVE,"3;14",DIRECT-
          ,DOUBLE_WIDTH,NEWDATE,ATTRIBUTES_OFF
$
$      LEN = 'F$LENGTH(LINE)'
$      L := 'F$EXTRACT(12,LEN-12,LINE)'
$      P = POS -5
$      SAY MOVE,POS,";1",DIRECT,P,MOVE,POS,";5",DIRECT,L
$      POS = POS + 1
$      GOTO NEXT_READ
$!
$!
$!
$! Here is leaving under duress. Destroy the copy just created.
$LEAVE_CLOSE:
$      CLOSE RD
$LEAVE_FAST:
$      SAY CLEAR_SCREEN,MOVE,"1;24",SCROLL
$      EXIT 3
$!
$!
$! Here if leaving normally. Keep only two reminder copies around.
$!
$NO_PUT:
$      CLOSE RD
$      IF P .EQ. 0 THEN EXIT 1
$      SAY MOVE,"21;1",DIRECT
$      INQUIRE X "'BOLD'Press RETURN to continue'"ATTRIBUTES_OFF"
$      SAY MOVE,"1;24",SCROLL,CLEAR_SCREEN
$      EXIT 1
$!

```

```

$!*****

```

```

$!*****
$!***** WORDPROC *****!$
$! This program serves as a word processor by utilizing the following:
$!      * The file structure of OATS
$!      * The EDT editor
$!      * The RUNOFF text-formatting program
$! The user has a choice of three styles of document formatting (letter,
$! memo, and standard) or no formatting. The letter style and memo style
$! prompt the user for the addressee and the subject. The appropriate
$! RUNOFF commands to insert those items (including date) in the proper
$! place are written into the temporary file. The text editor EDT is
$! then run with the temporary file as input and the cursor is put
$! at the end of the buffer. The user then edits the document as desired.
$! When the user exits EDT, the user is then asked to specify the
$! disposition of the document (save, format, and print; format and print
$! only; save only).
$      SAY CLEAR_SCREEN,MOVE,"1;22",SCROLL,MOVE,"24;66",DIRECT,REVERSE-
$          ,"Word Processor",ATTRIBUTES_OFF
$!
$! Verify that there's enough room to do this.
$!
$      @'OATS_DIRECTORY'ROOMLEFT 100
$      IF $STATUS .EQ. 1 THEN GOTO IS_ROOM
$      SAY BEEP,MOVE,"24;1",DIRECT,BOLD,BLINK-
$          ,"Not enough disk space left for Word Processing."
$      INQUIRE XX "Press RETURN to get back to MAIN MENU" 'ATTRIBUTES_OFF'
$      EXIT_MENU_NUMBER := "01"
$!
$! Return to main menu.
$!
$      EXIT
$!
$!
$!
$!
$! IS_ROOM:
$      IF P1 .EQS. "24" THEN GOTO PROC_NEW
$      IF P1 .EQS. "26" THEN GOTO PROC_OLD
$      IF P1 .EQS. "25" THEN EXIT
$      IF P1 .EQS. "99" THEN GOTO NEEDS_HELP
$      IF P1 .EQS. "11" THEN GOTO LETTER_STYLE
$      IF P1 .EQS. "12" THEN GOTO MEMO_STYLE
$      IF P1 .EQS. "16" THEN GOTO GET_DATE
$      IF P1 .EQS. "18" THEN GOTO GET_DATE
$!
$! Process a new document.
$!
$! PROC_NEW:
$      SAY MOVE,"24;1",DIRECT,BLINK,"New Document",ATTRIBUTES_OFF

```

```

$!
$! Get a filename for it.
$!
$      @'DATS_DIRECTORY'NEWNAME
$      IF $STATUS .EQ. 3 THEN EXIT
$!
$! Now display the filename for a second or two.
$!
$      SAY MOVE,"24:50",DIRECT,CLEAR_TO_CURSOR,MOVE,"24:1",DIRECT-
        ,"New Document: ",BOLD,NEW_DRAWER," ",NEW_FILE," "-
        ,NEW_FOLDER," ",NEW_CONTENTS,ATTRIBUTES_OFF-
        ,MOVE,"1:3",SCROLL
$!
$! Now goto menu to choose style of document.
$!
$      WAIT 00:00:03
$      SAY MOVE,"1:24",SCROLL
$      EXIT_MENU_NUMBER := "10"
$      EXIT
$!
$!
$! Now format the document.
$!
$MEMO_STYLE:
$LETTER_STYLE:
$      SAY MOVE,"24:50",DIRECT,CLEAR_TO_CURSOR,MOVE,"24:1",DIRECT-
        ,"New Document: ",BOLD,NEW_DRAWER," ",NEW_FILE," "-
        ,NEW_FOLDER," ",NEW_CONTENTS,ATTRIBUTES_OFF-
        ,MOVE,"1:3",SCROLL
$TW:
$      RETURN := "TW"
$      SAY MOVE,"10:1",DIRECT
$      INQUIRE/GLOBAL TOWHOM "'BOLD'To Whom'ATTRIBUTES_OFF'"
$      LENGTH = 'F$LENGTH(TOWHOM)'
$      IF LENGTH .GT. 20 THEN GOTO TOO_LONG
$FW:
$      RETURN := "FW"
$      INQUIRE/GLOBAL FROMWHOM "'BOLD'From Whom'ATTRIBUTES_OFF'"
$      SAY CLEAR_LINE,INDEX_UP
$      LENGTH = 'F$LENGTH(FROMWHOM)'
$      IF LENGTH .GT. 20 THEN GOTO TOO_LONG
$SB:
$      INQUIRE/GLOBAL SUBJECT "'BOLD'Subject'ATTRIBUTES_OFF'"
$      SAY CLEAR_LINE,INDEX_UP
$      IF SUBJECT .LT. 70 THEN GOTO GET_GRT_NAME
$      SAY BOLD,"Subject must be less than 70 characters long.",ATTRIBUTES_OFF
$      SAY INDEX_UP,INDEX_UP,CLEAR_LINE
$      GOTO SB
$!
$!

```

```

$TOO_LONG:
$      SAY INDEX_UP,SAVE_CURSOR,CLEAR_LINE
$      SAY MOVE,"20;1",DIRECT-
$          ,"To/From names must be less that 20 characters." -
$          ,RESTORE_CURSOR,INDEX_UP
$      GOTO 'RETURN'
$!
$!
$GET_GRT_NAME:
$      IF P1 .EQS. "12" THEN GOTO GET_DATE
$      INQUIRE/GLOBAL SAL "'BOLD'Salutation name'ATTRIBUTES_OFF'"
$      RETURN := GET_GRT_NAME:
$      LEN = 'F$LENGTH(SAL)'
$      IF LEN .GT. 20 THEN GOTO TOO_LONG
$!
$GET_DATE:
$      DATE[0,11] := "'F$TIME()'"
$      TYPE_DOC := ""
$      IF P1 .EQS. "11" THEN TYPE_DOC := "L"
$      IF P1 .EQS. "12" THEN TYPE_DOC := "M"
$      IF P1 .EQS. "16" THEN TYPE_DOC := "S"
$      IF TYPE_DOC .EQS. "" THEN GOTO NOFORM
$      @'OATS_DIRECTORY'FILEFOUND FORMAT.COM
$      ST = $STATUS
$      IF ST .EQ. 1 THEN @FORMAT 'TYPE_DOC'
$      IF ST .EQ. 3 THEN @'OATS_DIRECTORY'FORMAT 'TYPE_DOC'
$!
$! Here if formatted or not, old or new.
$!
$NOFORM:
$      NAMELINE == NEWLINE
$DOC_EDIT:
$      SET NOCONTROL_Y
$      ASSIGN/USER TT: SYS$INPUT
$      EDIT/EDT/COMMAND='OATS_DIRECTORY'EDTINI.EDT 'NAMELINE'
$      SET MESSAGE/NOTEXT/NOSEVERITY/NOFACILITY/NOIDENT
$      PURGE/KEEP:2 'NAMELINE'
$      SET MESSAGE/TEXT/SEVERITY/FACILITY/IDENT
$      SET CONTROL_Y
$      EXIT_MENU_NUMBER := "05"
$      EXIT
$!
$!
$! Here if re-editing an old document.
$!
$PROC_OLD:
$      SAY MOVE,"24;1",DIRECT,BLINK,"Old Document Name",ATTRIBUTES_OFF
$      @'OATS_DIRECTORY'DOCSELECT
$      IF $STATUS .EQ. 3 THEN EXIT
$!

```

\$! now display the filename.

\$!

\$ SAY MOVE,"24:50",DIRECT,CLEAR_TO_CURSOR,MOVE,"24;1",DIRECT-
,"New Document: ",BOLD,LAST_DRAWER," ",LAST_FILE," "-
,"LAST_FOLDER," ",LAST_CONTENTS,ATTRIBUTES_OFF-
,"MOVE,"1;3",SCROLL

\$ GOTO DOC_EDIT

APPENDIX B

MACRO ASSEMBLY LANGUAGE PROGRAM

```

.PSECT    GET_COMMAND
;
; THIS PROGRAM READS TWO CHARACTERS OF INPUT FROM THE TERMINAL. IT
; IS ASSUMED THAT THE INPUT WILL BE TWO LETTERS. THE INPUT STRING
; IS RETURNED BY USE OF THE $EXIT_S SYSTEM SERVICE ROUTINE IN THE SYSTEM
; SYMBOL $STATUS. THIS IN TURN IS READ BY THE DCL PROGRAM RUNNING THIS
; ROUTINE. ANY REMAINING CHARACTERS OR TERMINATOR TYPED IN IS LOST.
;
; This macro will evaluate the character in char and if it is not a
; letter ( either upper or lower case), control will be transferred
; to the label specified by NO_LABEL. Otherwise the normal flow
; will occur.
;
;
.MACRO    CHECK_LETTER CHAR,NO_LABEL,?LABEL
CMPB     CHAR,/^X41
BLSS     NO_LABEL          ;NOT A LETTER, LESS THAN "A"
CMPB     CHAR,/^X5A        ;"Z"
BLEQ     LABEL            ;YES, UPPER CASE LETTER
CMPB     CHAR,/^X61 ;LOWER CASE, THEN
BLSS     NO_LABEL ;NO
CMPB     CHAR,/^X7A ;LOWER CASE Z
BGTR     NO_LABEL ;NOT A LETTER
LABEL:   .ENDM    CHECK_LETTER ;HERE IF VALID LETTER
;
; This macro changes the location INBUF to the value specified
; by the argument CODE. This will be returned to the command level
; in the system service symbol $STATUS via the $EXIT_S System Service
; call.
;
;
.MACRO    SET_ERROR_CODE CODE
MOVL     #CODE,INBUF
BRW      FINISHED
.ENDM    SET_ERROR_CODE
;
;*** CHARACTER DEFINITIONS ***
;
ESC = ^X1B
ENTER = ^X0D
TTNAME:  .ASCID    /SYS$COMMAND/ ;DESCRIPTOR FOR TERMINAL NAME
TTCHAN:  .BLKW     1              ;RECEIVE CHANNEL NUMBER HERE
TTIOSB:
IOSB_STATUS: .BLKW 1              ;FIRST WORD IOSB -- STATUS
TTIOLEN:
IOSB_BUFLN: .BLKW 1              ;SECOND WORD -- OFFSET TO TERMINATOR

```

```

IOSB_TERMINATOR:      .BLKW    1          ;THIRD WORD -- TERMINATOR CHARACTER
IOSB_TERMSIZE:        .BLKW    1          ;FOURTH WORD -- SIZE OF TERMINATOR
;
INBUF: .BLKB          4          ;BUFFER TO READ INPUT
;
DEVDESC:              ;DESCRIPTOR FOR LOGICAL NAME TRANSLATION
NLEN: .LONG           63          ;LENGTH OF BUFFER
NADDR: .LONG          NAME        ;ADDRESS OF BUFFER
NAME: .BLKB           63          ;ALLOCATE 63-BYTE BUFFER TO HOLD
                                   ;EQUIVALENCE NAME OF "SYS$INPUT"
;
;
;
        .ENTRY    GET_COMMAND ^M(R6,R7)
$TRNLOG_S LOGNAM=TTNAME,RSLEN=NLEN,RSBUF=DEVDESC
CMPB      NAME,#ESC ;DOES NAME BEGIN WITH ESCAPE? (IF SO,
                                   ; IT'S A PROCESS PERMANENT FILE.)
BNEQ      10$        ;NO, SKIP
SUBL      #4,NLEN     ;OTHERWISE, SUBTRACT 4 FROM LENGTH
ADDL      #4,NADDR    ;ADD 4 TO ADDRESS
;
10$:      $ASSIGN_S DEVNAM=DEVDESC,CHAN=TTCHAN ;ASSIGN CHANNEL
BLBS      R0,NOERR6   ;NO ERROR IF SET
BSBW      ERRORASGN
;
NOERR6:   CLRL        INBUF          ;CLEAR THE INPUT BUFFER
$INPUT    CHAN=TTCHAN,LENGTH=#2,BUFFER=INBUF,IOSB=TTIOSB
BLBS      R0,NOERR7   ;NO ERROR IF SET
BSBW      ERRORINP
NOERR7:   CMPW        TTIOSB,#SS$_NORMAL ;I/O SUCCESSFUL?
BNEQ      10_ERR      ;ERROR IF NOT ...
;
        BRW          CHECK_VALIDITY ;NOW CHECK THE INPUT STRING
;
ERRORASGN: SET_ERROR_CODE 0
ERRORINP:  SET_ERROR_CODE 2
10_ERR:    SET_ERROR_CODE 4
;
;      INPUT STRING IS IN INBUF.
;
CHECK_VALIDITY:
        MOVAB        INBUF,R6 ;ADDRESS OF FIRST CHARACTER
;
;      CHECK NEEDS TO BE MADE FOR THE VALIDITY OF THE LETTERS.
;
        CHECK_LETTER (R6),ERROR_RET
        INCB         R6          ;NEXT CHARACTER
;

```

```

        CHECK_LETTER (R6),ERROR_RET
FINISHED:
        $DASSGN_S CHAN=TTCHAN ;DONE, DEASSIGN CHANNEL
        BLBS      R0,NOERR4
        BSBW      ERROR0
NOERR4:  $EXIT_S   INBUF
ERROR_RET:
        SET_ERROR_CODE 6
;
ERROR0: SET_ERROR_CODE 4
;
;       THESE ARE ERROR RETURNS.  DIGITSS ARE RETURNED TO INDICATE THE
;       ERROR STATUS OF THE INPUT OPERATION, INCLUDING INVALID INPUT.
;
        .END      GET_COMMAND

```

VITA

Jimmy Elaine Chambers Jansen was born in Bedford, Indiana, on November 11, 1945. She attended elementary schools in that city and was graduated from Bedford High School in June, 1963. The following September she entered the University of Evansville in Evansville, Indiana. In June, 1967, she received the Bachelor of Arts degree in Secondary Education with a major in Mathematics and a minor in General Science. She taught in Junior High Schools in Columbus, Indiana and Oak Ridge, Tennessee until December, 1968.

In September, 1971, she began study toward a Master of Science degree at University of Tennessee, Knoxville. From February, 1972, until February, 1974, she was employed at the Oak Ridge National Laboratory in Oak Ridge, Tennessee, as a systems programmer in the Computer Sciences Division. Studies were suspended in 1974. Mrs. Jansen re-entered the University of Tennessee, Knoxville, in September, 1980. She held a graduate assistantship in the Computer Center there during the 1980-1981 academic year. She received the Master of Science degree with a major in Computer Science in June, 1985.