

Draft

TOOL KIT  
USER'S GUIDE

XT

*Restricted Distribution*

---

**digital**

PROFESSIONAL DEVELOPER'S  
TOOL KIT USER'S GUIDE

Document: 01-0033-03

June 1982

This Guide explains how to use the Tool Kit to write application software that may be integrated with the Professional Operating System (P/OS). The information in this document is preliminary and subject to change without notice.

PRELIMINARY DRAFT

RESTRICTED DISTRIBUTION



The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may only be used or copied in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by DIGITAL or its affiliated companies.

The specifications and drawings, herein, are the property of Digital Equipment Corporation and shall not be reproduced or copied or used in whole or in part as the basis for the manufacture or sale of items without written permission.

Copyright (c) 1982 by Digital Equipment Corporation

All Rights Reserved

The following are trademarks of Digital Equipment Corporation:

DEC	DECnet	DECsystem-10
DECSYSTEM-20	DECUS	DECwriter
DIBOL	DIGITAL	EduSystem
IAS	MASSBUS	PDP
PDT	RSTS	RSX
UNIBUS	VAX	VMS
VT		

RESTRICTED DISTRIBUTION

## PREFACE

## PART I OVERVIEWS

## CHAPTER 1 PROFESSIONAL DEVELOPER'S TOOLKIT

1.1	INTRODUCTION TO THE PROFESSIONAL . . . . .	1-1
1.2	INTRODUCTION TO THE PROFESSIONAL DEVELOPER'S TOOL KIT . . . . .	1-2
Figure ? Tool Kit Environment		

## CHAPTER 2 P/OS CONCEPTS FOR APPLICATION DEVELOPERS

2.1	SYSTEM CONVENTIONS . . . . .	2-1
2.1.1	Program Types . . . . .	2-2
2.1.2	Program States . . . . .	2-2
2.1.2.1	Interactive Programs . . . . .	2-3
2.1.2.2	Noninteractive Programs . . . . .	2-3
2.1.2.3	Application Multitasking . . . . .	2-3
2.2	USER APPLICATIONS . . . . .	2-3
2.3	P/OS SYSTEM SERVICES . . . . .	2-4
2.3.1	File, Print, And Disk Services . . . . .	2-4
2.3.2	PROSE And P/OS COMMUNICATIONS FACILITY . . . . .	2-4
2.4	USER INTERFACE SERVICES . . . . .	2-6
2.4.1	Menu Services . . . . .	2-6
	Figure ? : DISK-BASED P/OS Menu System	
	Figure ? : DISKETTE-BASED P/OS Menu System	
2.4.2	Help Services . . . . .	2-9
2.4.3	Message Services . . . . .	2-9
2.4.4	Message/Status Board . . . . .	2-10
2.4.5	Function Keys . . . . .	2-10
	Figure ? : Professional Keyboard	
2.4.6	Operation Of Function Keys In P/OS . . . . .	2-12
	Table ? : Description of Function Keys	
2.5	P/OS SYSTEM COMPONENTS . . . . .	2-15
Figure: Flow of Control At Run-Time		
2.5.1	Resident Clustered Libraries . . . . .	2-17
2.5.2	ProDispatcher . . . . .	2-17
2.5.3	RSX/Exec . . . . .	2-18

## PART II APPLICATION DEVELOPMENT CYCLE

Figure: Program Development Cycle

## CHAPTER 3 SELECTING SOFTWARE COMPONENTS FOR A PROFESSIONAL APPLICATION

3.1	PROFESSIONAL HARDWARE CONFIGURATIONS . . . . .	3-1
-----	--	-----

3.2	PROFESSIONAL SOFTWARE CONFIGURATIONS . . . . .	3-1
Table 3-1: Application Packages		

#### CHAPTER 4 TOOLS FOR WRITING THE APPLICATION

4.1	PROGRAMMING LANGUAGES . . . . .	4-1
4.1.1	PRO/BASIC-PLUS-2 . . . . .	4-2
4.1.2	PRO/DIBOL . . . . .	4-2
4.1.3	PDP-11 MACRO-11 . . . . .	4-3
4.2	FORMS MANAGEMENT SYSTEM (PRO/FMS-11) . . . . .	4-3
4.3	PRO/RECORD MANAGEMENT SERVICES (RMS-11) . . . . .	4-4
4.4	PRO/SORT . . . . .	4-5
4.5	P/OS SERVICES LIBRARY . . . . .	4-5
4.6	GRAPHICS LIBRARY . . . . .	4-5
4.7	RSX MACRO LIBRARY AND SYSLIB (SYSTEM OBJECT LIBRARY) . . . . .	4-6
4.8	FRAME DEVELOPMENT TOOL (FDT) . . . . .	4-6

#### CHAPTER 5 TASK BUILDING THE APPLICATION ON THE HOST

#### CHAPTER 6 CREATING AN APPLICATION INSTALLATION FILE

#### CHAPTER 7 TRANSFERING THE TASK TO THE PROFESSIONAL

#### CHAPTER 8 INSTALLING, EXECUTING, AND DEBUGGING THE APPLICATION

8.1	FAST INSTALL . . . . .	8-1
8.2	EXECUTE THE APPLICATION . . . . .	8-2
8.3	DEBUG THE APPLICATION . . . . .	8-2

#### CHAPTER 9 APPLICATION DISKETTE BUILDER

9.1	RUN THE APPLICATION DISKETTE BUILDER . . . . .	9-1
9.2	LABEL DISKETTES . . . . .	9-2

### PART III THE TARGET SYSTEM AND SOFTWARE TOOLS

#### CHAPTER 10 PROGRAMMING CONVENTIONS FOR P/OS

10.1	FUNCTION KEY OPERATION . . . . .	10-1
10.1.1	Settings . . . . .	10-1
10.1.2	Assigning Settings To The Function Keys . . . . .	10-2
10.1.3	Function Key Operation During Application Execution . . . . .	10-2
Table ? : Application Execution		

10.2	APPLICATION MULTITASKING . . . . .	10-3
10.3	THE PROFESSIONAL TERMINAL CHARACTER SET . . . . .	10-3
	Figure ? : DEC Multinational Character Set	

## CHAPTER 11 CALLABLE SERVICES

11.1	FILE, DISK, AND PRINT SERVICES . . . . .	11-1
11.1.1	Conventions . . . . .	11-1
	Table ? : User-visible File Types	
	Table ? : System/Application File Types	
	Table ? : Physical and Logical Device Names	
11.1.2	Services . . . . .	11-4
11.2	OVERVIEW OF P/OS COMMUNICATIONS FACILITY . . . . .	11-5
11.2.1	Communications Services Calling Conventions . . . . .	11-7
11.2.2	Returned Status Information . . . . .	11-7
11.2.2.1	I/O Status Block . . . . .	11-7
11.2.2.2	File Transfer Statistics Buffer . . . . .	11-9
11.2.3	Call Control Services . . . . .	11-9
11.2.3.1	Enable/Disable Communications Option (CCENBL And CCDSBL) . . . . .	11-9
11.2.3.2	Call Control Block Allocation (CCBALC) . . . . .	11-11
11.2.3.3	Attach Line (CCATT And CCATA) . . . . .	11-12
11.2.3.4	Detach Line (CCDET) . . . . .	11-13
11.2.3.5	Set Line Characteristics (CCSMC) . . . . .	11-14
11.2.3.6	Get Line Characteristics (CCGMC) . . . . .	11-15
11.2.3.7	Get/Put Line Configuration Record (CCLCRG And CCLCRP) . . . . .	11-16
11.2.3.8	Dial Call (CCDIAL) . . . . .	11-17
11.2.3.9	Answer Call (CCANS) . . . . .	11-18
11.2.3.10	Hangup A Call (CCHNG) . . . . .	11-19
11.2.3.11	Transmit Data (CCTXD) . . . . .	11-20
11.2.3.12	Receive Data (CCRXD) . . . . .	11-21
11.2.3.13	Flush Input Buffer (CCFLSH) . . . . .	11-22
11.2.3.14	Generate Break (CCBRK) . . . . .	11-23
11.2.3.15	Kill Transfer (CCKILL) . . . . .	11-24
11.2.4	File Transfer Services . . . . .	11-25
11.2.4.1	Setup File Transfer Options (FTOPTG And FTOPTP) . . . . .	11-25
11.2.4.2	Send A File (FTSND) . . . . .	11-27
11.2.4.3	Send A File And Wait (FTSNDw) . . . . .	11-28
11.2.4.4	Retrieve A File (FTGET) . . . . .	11-29
11.2.4.5	Get A File And wait (FTGETW) . . . . .	11-30
11.2.4.6	Get Transfer Status (FTSTS) . . . . .	11-31
11.2.4.7	Abort Current Transfer (FIABT) . . . . .	11-32
11.2.5	TMS Services . . . . .	11-33
11.2.5.1	Change Mode (CCMODE) . . . . .	11-33
11.2.5.2	Originate Call (CCORG) . . . . .	11-34
11.2.5.3	Auxiliary Keyboard Enable/Disable (CCAUXK) . . . . .	11-35
11.2.5.4	Turn Speaker On/Off (CCSPKR) . . . . .	11-36
11.2.5.5	Prepare To Go Voice (CCPTGV) . . . . .	11-37
11.2.5.6	Set DTMF Escape Sequence (CCDTMF) . . . . .	11-37
11.3	PROSE, TEXT EDITOR . . . . .	11-38
11.3.1	Callable Editor Task . . . . .	11-38
11.3.2	Example Calling Sequences . . . . .	11-40

11.4	CALLABLE SORT PROGRAM (PRO/SORT)	11-41
11.4.1	PRO/Sort Commands	11-41
11.4.2	Comment Statement	11-42
11.4.3	Indirect Command File Command	11-42
11.4.4	COLLATE Command	11-42
11.4.5	DEFAULT Command	11-42
11.4.6	FIELD Command	11-43
11.4.7	FORCE Command	11-43
11.4.8	INCLUDE Command	11-43
11.4.9	INPUT Command	11-44
11.4.10	OUTPUT Command	11-44
11.4.11	SORT Command	11-44
11.4.12	WRITE Command	11-44
11.4.13	Comparison Of PRO/Sort And PDP-11 SORT-11	11-44
11.4.13.1	Comment Statement	11-45
11.4.13.2	COLLATE Command Compared With ALTSEQ Records	11-45
11.4.13.3	FORCE Command Compared With F Command	11-45
11.4.13.4	INCLUDE Command Compared With O And I Record Specifications	11-46
11.4.13.5	INPUT And OUTPUT Command Compared With Command Line Format	11-46
11.4.13.6	SORT Command Compared With N And O Field Specifications	11-47
11.4.13.7	WRITE Command Compared With D Field Specifications	11-47

## CHAPTER 12 FRAME DEVELOPMENT TOOL (FDT)

12.1	PLANNING THE FRAMES	12-2
12.1.1	Menus	12-2
	Figure ??: A Single-choice Menu	
	Figure ??: A Help Menu	
	Figure ??: A Multiple-choice Menu	
	Figure ??: A Short-form Menu	
12.1.2	Help And Message Text Frames	12-9
	Figure ??: A Help Text Frame	
	Figure ??: A Message Text Frame	
12.2	DEVELOPING THE FRAMES	12-12
12.2.1	Overview	12-12
12.2.2	Sample Terminal Session	12-13
12.2.3	Frame Development Tool Commands	12-16
12.2.3.1	File Editing	12-19
12.2.3.2	Frame Editing	12-23
	Figure ??: Screen Editor Keypad	
12.3	CREATING A SINGLE-CHOICE MENU	12-27
	Figure ??: Profile Form for Single-choice Menu	
	Figure ??: Display Form for Single-choice Menu	
	Figure ??: Action Form for Single-choice Menu	
12.4	CREATING HELP MENUS AND HELP TEXT FRAMES	12-34
	Figure ??: Profile Form for Help Menu	
	Figure ??: Display Form for Help Menu	
	Figure ??: Action Form for Help Menu	
	Figure ??: Profile Form for Help Text Frame	
	Figure ??: Display Form for Help Text Frame	



12.5	CREATING MESSAGE TEXT FRAMES . . . . .	12-45
	Figure 7: Profile Form for Message Frame	
12.6	RESOLVING ERROR CONDITIONS . . . . .	12-48
12.6.1	Format Of Messages . . . . .	12-48
12.6.2	User Error Messages . . . . .	12-49
12.6.3	FDT Internal Errors . . . . .	12-54

## CHAPTER 13 P/OS SERVICES LIBRARY

13.1	DETAILED OVERVIEW . . . . .	13-1
13.2	CONVENTIONS . . . . .	13-2
13.3	MENU SERVICE ROUTINES . . . . .	13-2
13.4	HELP SERVICE ROUTINES . . . . .	13-16
13.5	MESSAGE SERVICE ROUTINES . . . . .	13-21
13.5.1	Values Returned In Status . . . . .	13-22
13.6	MISCELLANEOUS SERVICES . . . . .	13-24

## APPENDIX A SUMMARY OF ENTRY POINTS TO P/OS

A.1	SUMMARY OF ENTRY POINTS . . . . .	A-1
A.1.1	File Services . . . . .	A-1
A.1.2	Disk Services . . . . .	A-1
A.1.3	Communications . . . . .	A-1
A.1.4	Editor . . . . .	A-1
A.1.5	User Interface . . . . .	A-1
A.1.6	Miscellaneous System Services . . . . .	A-2

## APPENDIX B THE DEC MULTINATIONAL CHARACTER SET

## GLOSSARY



## PREFACE

## INTENDED AUDIENCE

The Tool Kit User's Guide is for application developers who are creating software for P/OS. This manual explains how to use the Professional Developer's Tool Kit. This manual assumes that you are familiar with the host system environment (RSX-11M, RSX-11M-PLUS, or VAX/VMS) and that you understand the needs of your own end user.

## DOCUMENT DESIGN

Part I provides two overviews: the first overview introduces the Tool Kit. The second overview introduces the target operating system, P/OS.

Part II describes the software development tools in the Tool Kit and the program development cycle for applications.

Part III describes programming conventions for P/OS, callable services, Frame Development Tool, and P/OS Services library.

## DOCUMENTATION CONVENTIONS

Certain conventions are used in the manual to distinguish different kinds of information. These conventions are as follows:

- o Examples consist of computer output wherever possible.
- o The term filespec denotes a file specification, which includes a device name, a directory specification, a file name, file type, and if applicable, a version number.
- o User input is printed in red to distinguish it from computer output.
- o Required characters are in UPPERCASE letters. Although you must type required characters -- for example, a command name -- in the sequence shown, you can type those characters in either uppercase or lowercase characters.
- o Variables, for which you must supply a value, are in lowercase characters. You can type variables in either uppercase or lowercase characters.
- o Vertical or horizontal ellipsis points (: or ...) indicate omission of optional repetitive information. You can repeat the item that precedes the ellipsis.



## PART I OVERVIEWS

Chapter 1 presents an overview of the Professional 300 Family and the Professional Developer's Toolkit.

Chapter 2 presents an overview of the Professional Operating System (P/OS).





## CHAPTER 1

### PROFESSIONAL DEVELOPER'S TOOL KIT

The Professional Developer's Tool Kit is a set of software development tools designed to be used by application developers to build applications for the Professional 300 family and computers. The Tool Kit contains software tools to help you develop standard application software for the Professional Operating System (P/OS). P/OS is the operating system for the Professional 325 and 350 hardware.

#### 1.1 INTRODUCTION TO THE PROFESSIONAL

A Professional is a computer that belongs to a family of single-user, desk-top computers that can be used in many ways. For example, a Professional can be used as a:

1. Personal computer for Professional engineers, executives, scientists, journalists, and others
2. Small-business system to perform accounts receivable, accounts payable, payroll, inventory, personnel records, correspondence, and so forth
3. VAX/VMS or RSX-11M/M-PLUS workstation for general time-sharing, program development, large-scale data processing, and so forth

All members of the Professional family share the following key features:

- o PDP-11 based processor
- o 256KB memory
- o Dual diskette drive for 400KB diskettes

These common features form the Professional 325 and 350. However, the Professional 325 does not accept additional options. Of primary interest to application developers are the following differences in

## models:

1. The Professional 350 may have, in addition to the 400KB dual diskette drive, a 5MB disk.
2. P/OS may be either diskette-based or disk-based depending upon the hardware configuration.

P/OS software is a successor of on the RSX-11M-PLUS operating system. However, the P/OS user interface replaces the RSX command interface. The user interface includes on-line help, keyboard and screen conventions, and a menu tree with a branch reserved for the applications you develop. The user interface can be used by people with differing levels of knowledge about computers. Inexperienced computer users can quickly start using P/OS services through the menus.

The menus and screen prompts guide the user in creating files, printing letters, and writing programs in BASIC. Displays of instructional information appear on the screen at the touch of the HELP key. P/OS users who are familiar with other computers can quickly learn Professional keyboard and P/OS system conventions. A full complement of user documentation is provided with the system.

P/OS is a foundation for application software; therefore, you can write programs that work well with its operating system and blend in with the complete system. Application programs that use the features of P/OS will provide the end user with a consistent way to interact with the system. Using the Tool Kit, programmers can create P/OS applications in which the user interface is consistent with P/OS services provided by DIGITAL.

## 1.2 INTRODUCTION TO THE PROFESSIONAL DEVELOPER'S TOOL KIT

The Tool Kit software consists of application development software:

- o A program development language that generates programs capable of executing on the Professional:
  1. PRO/BASIC-PLUS-2 Extended BASIC Compiler and Debugger
  2. PRO/DIBOL Business-Oriented programming language
  3. MACRO-11 Assembly Language
- o The Professional Operating System (P/OS) Services Library (Object) including interfaces to menus, help, messages, and graphics

- o FMS-11 Forms Management System
- o RMS-11 File Management System with multikey ISAM
- o PRO/Sort Callable sorting task
- o Frame Development Tool (FDT) for creating menus, on-line help text, and message frames
- o RSX Task Builder for building RSX-based programs
- o P/OS Macro Interfaces Library
- o PRO/Diskette Builder to create Professional Application Diskettes
- o Program development documentation
- o On-line Debugging Technique (ODT)

You can use RSX-11M, RSX-11M-PLUS, or VAX/VMS to create an application that will run on the target system. On VAX/VMS, the program development tools run under the Applications Migration Executive (AME). Any supported configuration of these systems is appropriate for P/OS application development.

In addition to the above software and documentation, to complete the developer's Tool Kit development environment you must obtain the Professional 325 or 350, P/OS, and Professional Communications Facility (with cabling to the host). The Professional with P/OS is the target run-time system for applications you develop.

With P/OS Communications Facility, you can use the Terminal Emulator and File Transfer program on the Professional to create an executable Professional application on the host system. In terminal emulation mode, the Professional behaves like a terminal on the host system. When you are ready to run and debug the application on the Professional, the you can transfer the files over the communication line to the Professional. Now you can test and debug the application in the target environment.

When your application is ready to be given to an end user, you Professional Diskettes on the Professional with the Application Diskette Builder.

Figure ? Tool Kit Environment

TO BE SUPPLIED





## CHAPTER 2

### P/OS CONCEPTS FOR APPLICATION DEVELOPERS

This chapter gives a conceptual overview of the Professional Operating System (P/OS) and introduces important terms.

This chapter discusses the following topics:

- o System Conventions
- o User Applications
- o P/OS System Services
- o User Interface Services
- o P/OS System Components

#### 2.1 SYSTEM CONVENTIONS

P/OS is a single-user operating system. Menus and forms are the user's command interface to system services and applications.

System conventions dictate the types of programs that P/OS supports and the resources each type of program can use. These are introduced in the next section.

System conventions also govern the number of applications that can be in use at one time and the state of an application when it is not in use. Programs may be interactive or noninteractive. These conventions are described in Section ?

### 2.1.1 Program Types

P/OS supports two types applications and utilities:

1. User applications: These applications perform functions related to the user's work, since they are directed towards the specific problems that the end user wants to solve. Your application will most likely be in this category.
2. P/OS services: These services help the user manage Professional computer resources. Your application can use these services during program execution. Services are described in Section ?

P/OS programs are categorized according to use in the list below. User applications that are appropriate for P/OS are listed in the lefthand column and P/OS System Services are listed in the righthand column.

#### USER APPLICATIONS

Phone dialer  
General Ledger  
Checkbook Program  
Mortgage Program

#### P/OS SYSTEM SERVICES

File Services  
Disk Services  
Print Services  
Communications Facility  
PROSE text editor

### 2.1.2 Program States

P/OS is a new operating system that has its roots in the RSX-11M-PLUS operating system. As such, it employs many of the same methods of managing resources as the RSX family of systems. For example, P/OS uses checkpointing to move tasks from memory to disk to make effective use of memory, it runs tasks on a priority basis to respond to real-time events, and it allows multitasking so multiple tasks can run simultaneously. The facilities for using these features are fully described in the *System Reference Manual*. This section introduces the important concepts.

Programs may be either interactive or noninteractive, and the end user cannot alter this property of an application. Applications you develop must be interactive. Noninteractive programs, such as Print services and the Communications Facility, are provided by DIGITAL.

P/OS allows multiple noninteractive programs (such as file printing and file transfer to another system) to run simultaneously with a single interactive program (such as PROSE). The interactive application has direct control over the keyboard and screen. While the noninteractive application is in operation, it is not directly

connected to the keyboard and screen.

2.1.2.1 Interactive Programs - while interactive programs are running, the function keys on the Professional keyboard are passed to the running task, except for the system function keys. The Professional keyboard and programming conventions are introduced in a subsequent section of this chapter.

2.1.2.2 Noninteractive Programs - Noninteractive Programs operate, for the most part, without interacting with the end user. After they are invoked from a menu, noninteractive applications can interact with the end user only indirectly through the Message/Status Board. P/OS reserves this execution environment for its own use.

More than one noninteractive application can run in parallel with an interactive application. Noninteractive applications do not have access to the screen or the keyboard.

2.1.2.3 Application Multitasking - One program can activate another by using an executive system service and through language-specific features, such as CHAIN and CALL in Pro/BASIC-PLUS-2. Thus, an interactive application may include more than one program. Chapter 10 discusses this in more detail.

## 2.2 USER APPLICATIONS

An application is a computer program that addresses the needs of an end user. The executable part of a user application is one or more tasks.

A task is the runnable portion of an application. To create a task, you must first write a source program and, by processing it through a language processor, create an object module. You create the task by submitting the object module to the Task Builder. The Task Builder creates a file containing a task image which may be executed on P/OS.

To create the task, the Task Builder accesses system and user-specified libraries to resolve references in it. Application tasks can access commonly used P/OS subroutines which are contained in resident libraries. Resident libraries are built and installed into the system separately from tasks that might reference them. To build a task that references subroutines contained in a resident library, for each library you must specify a symbol definition file in the task builder command line. The symbol definition file contains links that resolve calls from within the referencing task to locations within the library.

In addition to tasks, the P/OS application has disk or diskette files. The files contain the application menu, message, and help files, as well as disk overlays for tasks in memory. These files reside on disk or diskette during application execution. In Figure 2, they reside on the RD50 disk. The menu, message, and help files provide the user interface for an application.

The P/OS application also has an installation file on disk with the other files. The installation file contains information which directs ProDispatcher to install all the tasks associated with this application. The installation file also names all the application-related files. The installation file provides enough information to ProDispatcher so it can start the application, or interrupt and terminate it.

## 2.3 P/OS SYSTEM SERVICES

P/OS offers many services for manipulating files on disk or diskette, and for printing stored files or displayed screen images. These services are called File, Print, and Disk services. P/OS provides a text editor (PROSE) and Communications Facility as well.

P/OS also offers ProBasic for end users who want to write programs. However, since it is neither an application development tool nor a callable system service, ProBasic is not discussed here.

### 2.3.1 File, Print, And Disk Services

File services allow the end user to copy, rename, delete, backup, restore, append, purge, display and delete-protect files.

Files can be printed if the system includes a printer. Print services accepts commands to start, suspend, stop, and continue print jobs.

Disk services allow the end user to initialize diskettes, format the hard disk, and duplicate volumes. Directories can be created, deleted, and listed.

Certain File, Print, and Disk services are callable at run-time by application software. Chapter 3 describes interfaces to callable services.

### 2.3.2 PROSE And P/OS COMMUNICATIONS FACILITY

PROSE is the P/OS text editor. PROSE is intended for a business environment and is suitable for creating memos or entering data for applications. The end user presses arrow keys on the P/OS keyboard to move the cursor and edit or enter text. In addition to the basic editing functions, PROSE allows the end user to move text segments,

duplicate text segments, set margins, and perform global search and replace of specified text segments.

P/OS Communications Facility includes File Transfer, Terminal Emulation, and Telephone Management System.

To use Terminal Emulation, you can connect the Professional to a host computer via a local connection or a telephone line connection. You can use the Professional as a VT102 or a VT125 terminal through the Communications Facility. As a VT125 terminal, the Professional interprets REGIS code sent by the host. The Communications Phone Book is used to store and retrieve communications line and host characteristics.

The File Transfer service allows data files to be passed between two Professionals or between a Professional and a host running VAX/VMS or RSX-11M/M-PLUS. To use File Transfer, the Professional must be in Terminal Emulation mode.

The Telephone Management System (TMS) integrates the use of the telephone for voice and data communications into the Professional. Any of the functions that can be performed using the Kernal Communication port can be done with the Telephone Management System. This includes file transfer and terminal emulation. The Phone Book can be used to initiate voice and data calls.

PROSE and Communications Facility are callable at run-time by application software. Chapter 11 describes interfaces to callable services.

## 2.4 USER INTERFACE SERVICES

The P/OS user interface enables the end user to interact with applications and P/OS services through a series of menus and forms. It provides a simple method for selecting activities by positioning the pointer to items on menus or by typing the item on a command line, filling in blanks on forms, and pressing labeled function keys. On-line text provides helpful information whenever it is needed. Messages are displayed to inform the user of errors or confirm completion of an activity.

Your application can use P/OS Service Routines to provide the same user interface for your applications. Service routines to access menu, help, and message services are described in Chapter 13.

### 2.4.1 Menu Services

The end user interacts with P/OS services through menus. Menus are an efficient command interface for both novice and experienced computer users.

A menu is a list of commands. Like command lines in other systems, the menu options are used to direct P/OS to perform useful functions.

The menus supplied with P/OS are organized in a hierarchy. The top level Main Menu branches out to related menus. The Main Menu lists categories of P/OS services. After the end user signs on, the main menu appears and the end user selects a submenu or P/OS service. When the end user stops using a service or an application, the display returns to the Main Menu.

Figure ? shows the disk-based P/OS menu system as it appears before any optional applications have been added.

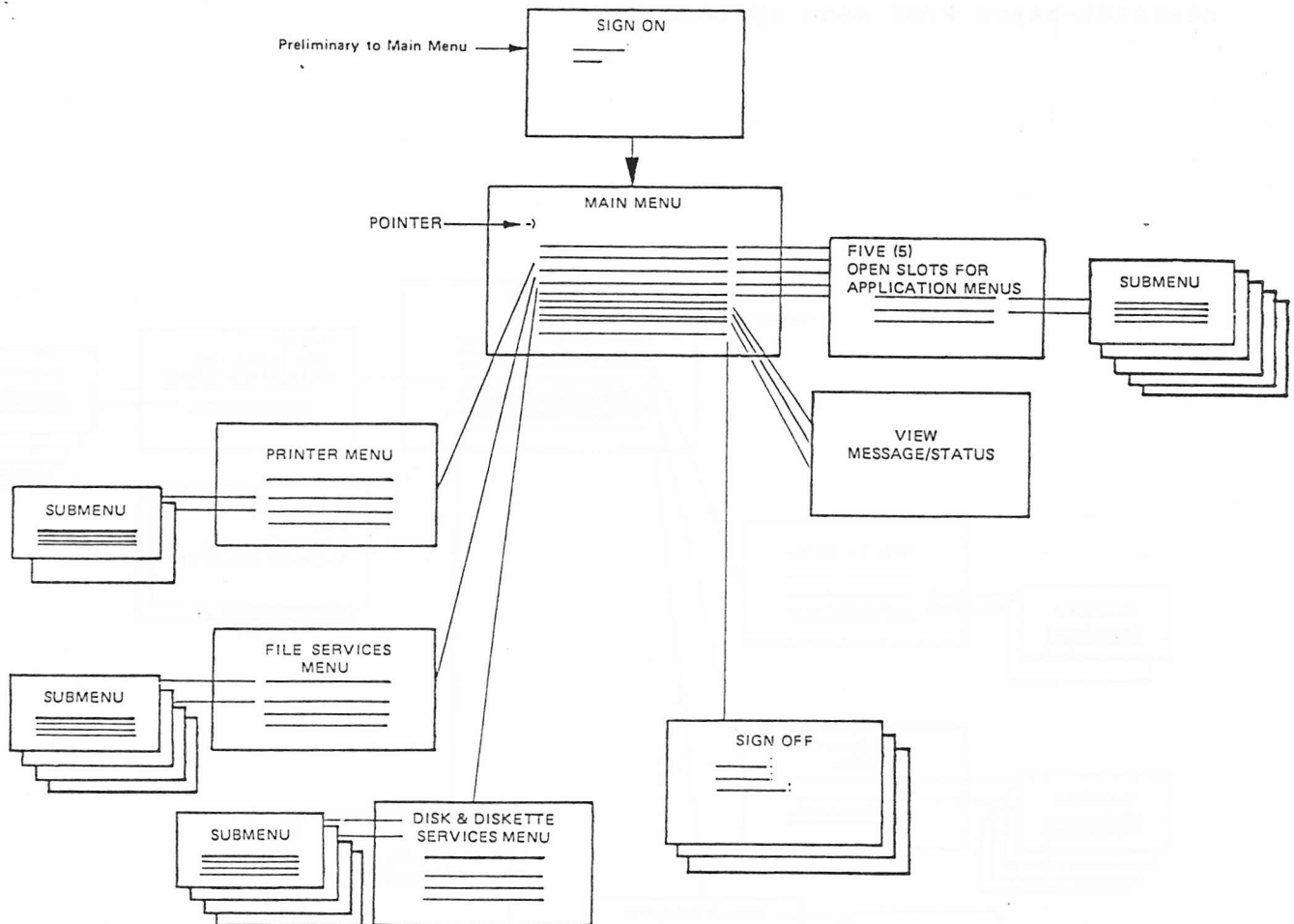


Figure 2: Disk-based P/OS Menu System

After purchasing a P/OS application, the end user integrates it into his system with the Install program. The Install program adds the name of the application to a menu so it can be invoked. The end user may add the application name to one of five slots in the Main Menu or on an Additional Applications submenu. This group of menus is reserved for user-installed applications on disk-based P/OS (the *User's Guide* describes the application installation service).

On diskette-based P/OS, applications are pre-installed by the application developer. The Main Menu has an entry for using applications contained on another diskette. To run an application residing on a separate diskette, the end user selects that option, removes the current system diskette, and replaces it with the

alternate diskette. In addition, the P/OS main menu has two open slots. When you create an application for P/OS, you must place an entry for your application on the P/OS main menu. Figure 7 shows the diskette-based P/OS menu system.

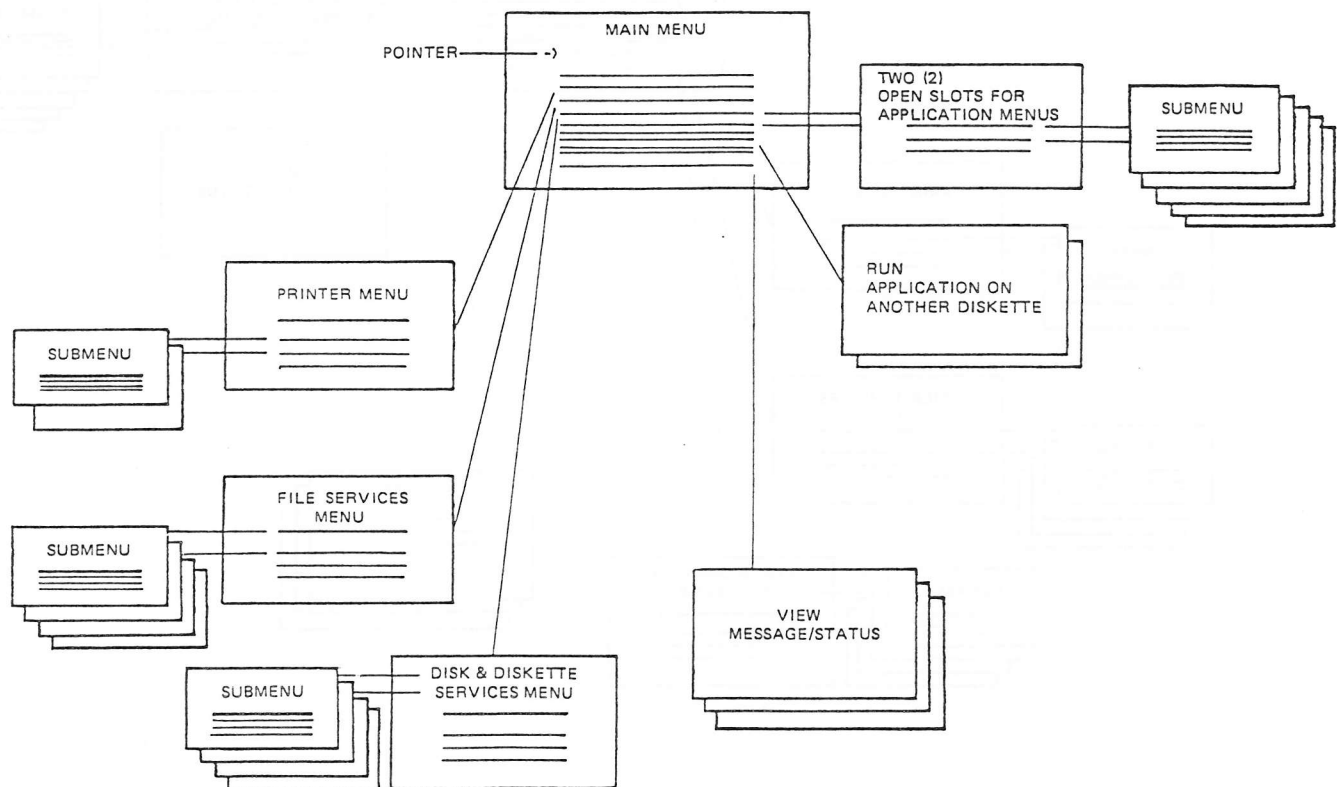


Figure 2: Diskette-based P/OS Menu System

Each application must have an entry on either the main menu in diskette-based P/OS or either on the main menu or the Additional Applications menu on disk-based P/OS. The end user invokes an application by selecting the entry for it from one of the prescribed menus. Most applications require additional menus or submenus through which the user interacts with the application.

ISSUES:



1. How does application developer put application name on diskette-based P/OS main menu?

#### 2.4.2 Help Services

P/OS displays helpful information about the current activity at the touch of the HELP key. The help information is either general information about the current application, or specific information about a single command.

You can use P/OS help services to display help information for your application. Help is displayed in frames which can be organized in parallel to the application menus. That is, each menu for your application can have an associated Help frame. While using your menus, the end user can press the HELP key to display the associated Help frame.

In addition to having a Help frame associated with each menu, you can have a Help menu that lists the different subjects for which on-line Help is available. A Help menu can have a hierarchical structure, using the PREV SCREEN and NEXT SCREEN keys to display the appropriate Help frames.

If the user presses the RESUME key while a Help menu or Help frame is being displayed, the screen reverts to the state it was in before the HELP key was pressed.

Applications that do not have menus may use P/OS Service routines to display help frames directly. In this case, if the user presses the RESUME key while a Help menu or Help frame is being displayed, the application must restore the screen to the state it was in before the HELP key was pressed.

If the application uses PRO/FMS-11 forms, help may be provided through the FMS-11 help structure. The FMS-11 Form Driver offers two levels of help if the user requests it: help for the form field in which the cursor is located and help for the entire form. The former is displayed the first time the user presses the help key; the latter is displayed the second time the user presses the help key. P/OS help services may be used in conjunction with FMS-11 help, or it may be used instead of FMS-11 help. The PRO/FMS-11 Documentation Supplement describes PRO/FMS-11 help in more detail.

#### 2.4.3 Message Services

Your application can use P/OS Message Services to store and retrieve text strings. These messages can be displayed on the screen during program execution or they may be used for inter-task communication.

#### 2.4.4 Message/Status Board

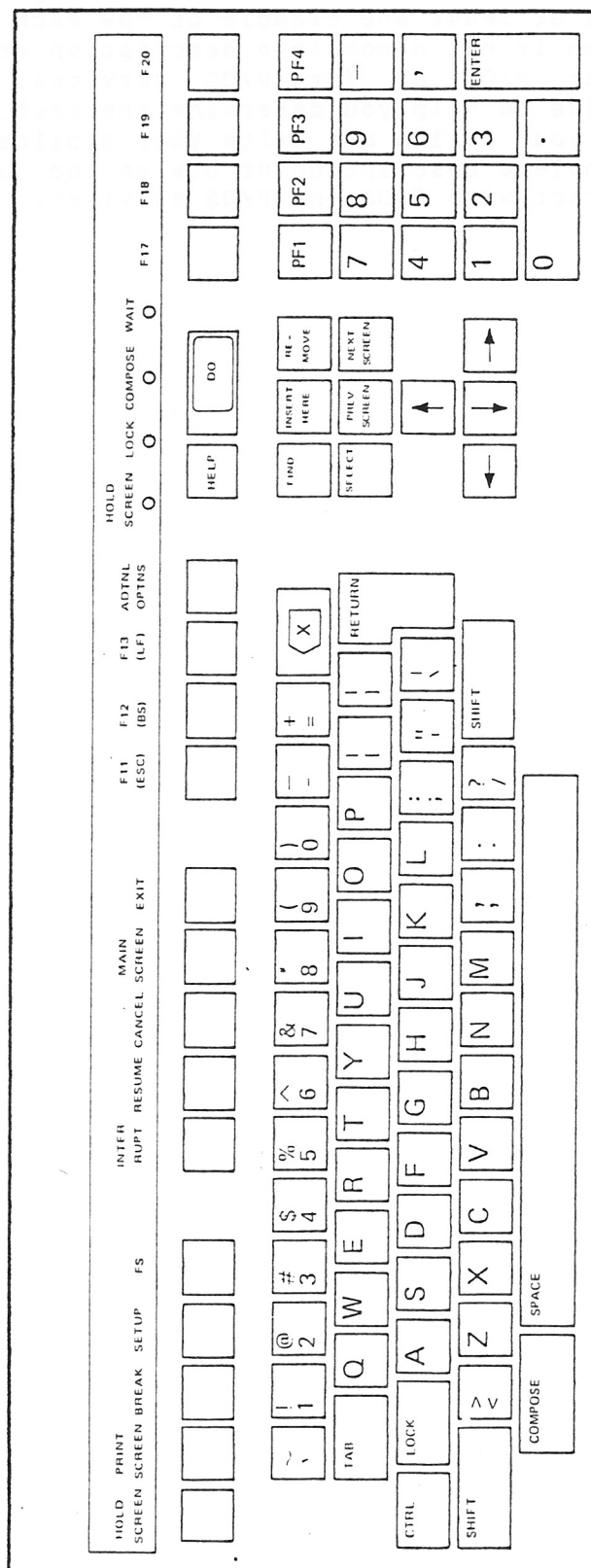
The Message/Status Board displays system status information, such as the names of the volumes in the dual diskette drive, as a service to the end user.

Your application can use the Message/Status Board during program execution to post messages directed to the end user. The P/OS Service routine to do this is in the P/OS Services Library, described in Chapter ?

#### 2.4.5 Function Keys

The Professional keyboard is shown in Figure ? (Figure ? shows the U.S. version of the Professional keyboard. Illustrations of the other keyboards supported by the Professional can be found in the Owner's Manual.) The keyboard has the main typewriter keyboard, an editing keypad, a numerical keypad, and keys that may be assigned a specific function by application software. These function keys are on the top row of the keyboard and in a cluster between the main keyboard and the numerical keypad. The label strip above the keys identifies their functions.

Figure 7: Professional Keyboard



#### 2.4.6 Operation Of Function Keys In P/OS

Table ? lists the general operation of function keys in the P/OS environment and gives at least one example of how each key is used in P/OS. This information is not a complete description of the operation of function keys for P/OS or for P/OS services. Rather, it is intended only as a guide to help you determine the best usage for the function keys when you design and write your application. See the User's Guide for a complete description of how an end user uses the function keys to interact with P/OS and P/OS services.

Table 7: General Description of Function Keys

KEY	RESULTING ACTION
-----	-----
ADDTNL OPTIONS	A key used to display additional options for the current menu. If additional options have been defined for the current single-choice menu, pressing this key displays a separate menu with the additional options.  Used in PROSE to offer options for text formatting.
Arrow Keys	The up and down arrows are used in P/OS to move the selector on menus.
BREAK	By industry convention, this function key is only used in communications software to cause a break condition on a communications line.
CANCEL	Used in all P/OS menus to discard anything typed so far and to reset the selection process.
DO	Indicates that a selection from a menu has been made. The current P/OS service takes action if an unambiguous selection was made.
EXIT	Used by PROSE to display an Exit menu, from which the end user leaves the text editor and returns to the P/OS Main Menu.  Also used by ProBasic to leave the ProBasic interpreter and return to the main menu.
FIND	Used in PROSE, the text editor, to search for particular text segments.
HELP	A key used to display information and instruction to the end user.  At the P/OS Main Menu, displays a Help frame describing how to use the menu system. A subsequent request for help results in a display of an P/OS Help menu. The Help menu provides access to Help for each P/OS service and function key.
HOLD SCREEN	The screen image stays still so it can be read. Output to the

screen continues when any key other than PRINT SCREEN is pressed.

**INSERT HERE**

Used in PROSE to allow text to be inserted rather than overtyped.

**INTERRUPT**

Stops the current activity in P/OS system services if following by DO key. Otherwise, no action.

**MAIN SCREEN**

A function key used to return the display to a central point in the current context.

In the P/OS menu tree, returns the display to the P/OS Main Menu.

**NEXT SCREEN**

A function key used to view a display that logically follows the current display.

Used in PROSE to move the cursor forward one screenful in the text file to display the next screenful of text.

**Numbered  
function  
keys**

In P/OS, these keys are not used.

**PREV SCREEN**

A function key used to view a display that logically precedes the current display.

Used in PROSE to move the cursor back one screenful in the text file to display the next screenful of text.

**PRINT SCREEN**

The screen remains still so the image can be printed. If the printer supports this feature and is connected, in working order and is not being used, the current screen is printed, preceded by a form feed. If there is no printer connected or if the printer is not working, an error message is displayed. If the printer is busy, a message reports that to the end user. Input and output to the screen continues when any key other than HOLD SCREEN is pressed.

**REMOVE**

Used by PROSE to erase text.

**RESUME**

A function key used to resume an activity.

	In P/OS, this key is used to return to the current activity after use of the HELP and ADDTNL OPTIONS keys.
SELECT	Used in multiple-choice menus to choose more than one item before pressing the DO key.
	Used in PROSE to mark some section of text for a subsequent operation.
SETUP	Displays the SETUP menu if pressed when the P/OS Main Menu is displayed. The SETUP menu allows you to change certain Professional hardware characteristics.

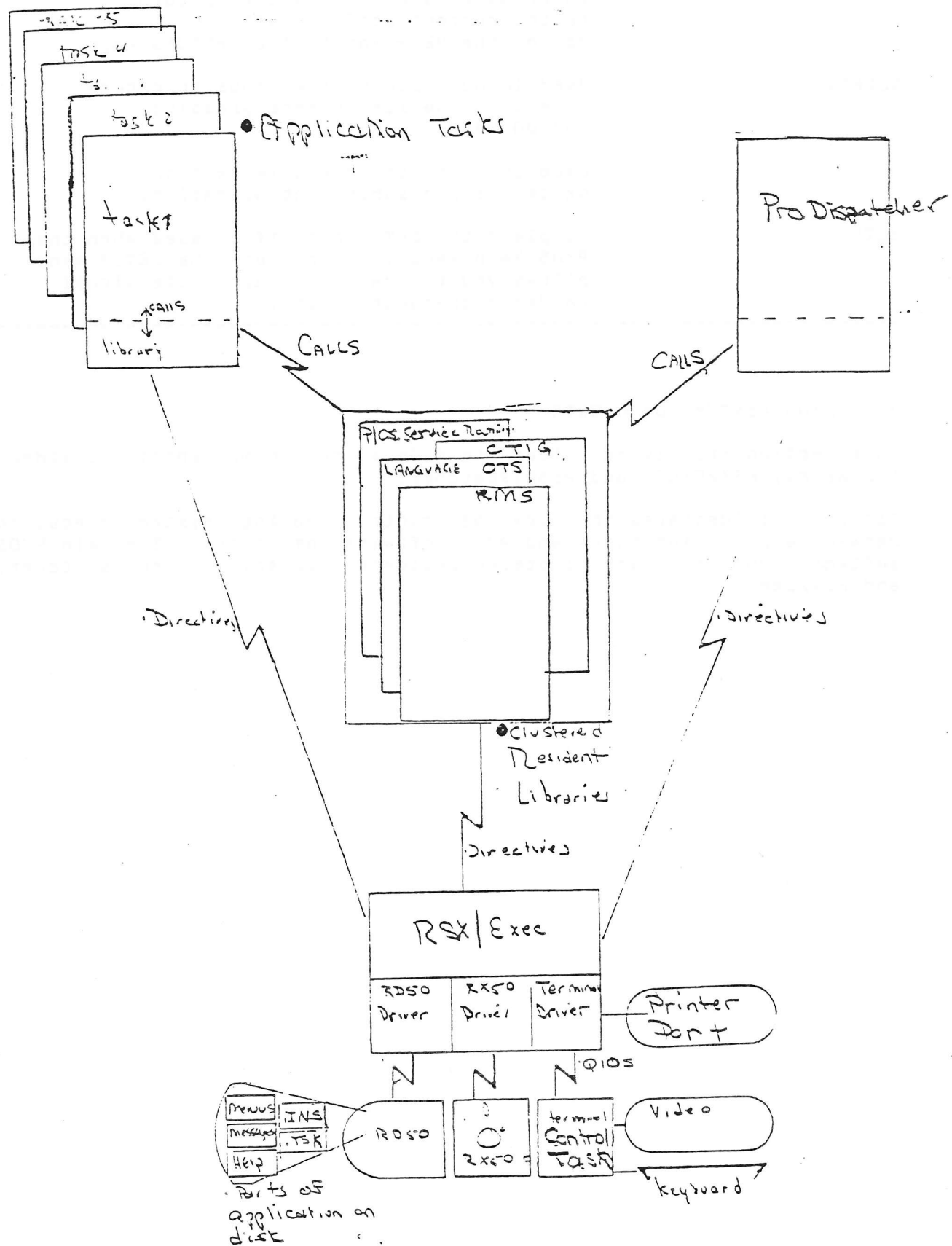
---

## 2.5 P/OS SYSTEM COMPONENTS

This section briefly describes the P/OS system components: resident libraries, RSX/EXEC, and ProDispatcher.

Figure ? illustrates the flow of control during system execution between application tasks and P/OS software components. The main P/OS software components are clustered resident libraries, ProDispatcher, and RSX/Exec.

Figure: Flow of Control At Run-Time





### 2.5.1 Resident Clustered Libraries

Resident libraries provide a single copy of commonly used subroutines in a way which can be shared by several tasks. Application tasks may be task built against these libraries on the host system. The libraries are not part of your application's task image. The P/OS resident clustered libraries are as follows:

1. Record Management System (RMS-11) providing record management services
2. Language Object Time System (OTS) providing language-specific run-time support to high-level languages
3. Graphics Primitives providing subroutines for creating graphics
4. P/OS Service Routines providing subroutines for displaying menus, help text, and messages on the screen from a data base on disk or diskette

Resident clustered libraries have the following characteristics:

1. Prebuilt as an integral part of the operating system. The application developer cannot add resident libraries to the system.
2. Clustered so all libraries are mapped into the same virtual address space.
3. Not part of application task image but do use virtual address space

#### Issues:

1. Define clustered from RSX TKB V2 manual.
2. Do Applications task build against RCL using command files?

### 2.5.2 ProDispatcher

The ProDispatcher controls the P/OS main menu and user interface services through which the end user interacts with the system services and applications. The ProDispatcher performs the following:

1. Adds the name of the application to the user-specified menu through an interactive program called INSTALL/REMOVE. By selecting the application name from the menu, the end user can run the application.

2. Adds the name of the application tasks to the list of executable tasks, called the system task directory (STD). Figure 7 shows ProDispatcher issuing directives to RSX/Exec to accomplish this job.
3. Invokes, aborts, and removes tasks. When the end user selects an application from a menu, ProDispatcher starts the first task for that application. If the end user presses the INTERRUPT key followed by the DO key, ProDispatcher aborts the running application and removes any tasks left running.

ProDispatcher uses the Application Installation file (through the Install program) to record the name of the first task to be invoked for this application, and to keep track of all related tasks. If the application developer specified default menu, help, and message files in the installation file, ProDispatcher issues calls to the P/OS Services Library to open these files when the application is invoked.

4. Recovers the system after disabling errors have occurred in a running application. If a task fails, control returns to ProDispatcher and the main menu is displayed.

### 2.5.3 RSX/Exec

RSX/Exec is the controlling program in the P/OS operating system. Its main duties are as follows:

1. Provides an intertask communication mechanism to ProDispatcher allowing it to carry out its application-related jobs
2. Contains executive directives which are accessible through macro calls. Executive directives may be used by applications to obtain system information, such as the date and time, and to control application execution, such as for multitasking.
3. Provides access through device drivers to the RX50 dual diskette drive and the RD50 hard disk.

When a diskette is inserted in a drive, RSX/Exec automatically adds its name to the list of available volumes. When the diskette is removed, the name is removed. The end user is not required to perform device mount and dismount. P/OS Disk services provide access to the end user to media.

4. Provides the on-disk file structure and record management system (RMS-11). P/OS File, Disk, and Print services provide these facilities to the end user.

5. Provides access through the terminal driver to the terminal control task. The terminal control task controls the video image and the keyboard. Applications can use ProDispatcher services and P/OS Graphics to display images on the screen and interpret keyboard input.
6. Provides access through the terminal driver to the printer. The PRINT key and P/OS print services allow the end user to print files on the printer.

**ISSUES:**

1. Add more information relating activities to the figure.



## P/OS CONCEPTS FOR APPLICATION DEVELOPERS

### PART II APPLICATION DEVELOPMENT CYCLE

Chapters 3 to 9 describe the program development cycle for P/OS applications. The steps in P/OS application development are as follows:

1. Identify the Professional software and hardware configuration for your application
2. Design the application
3. Design the application's accompanying documentation
4. Write the application, application user interface and, optionally, Software Security Program
5. Task build the Application on host
6. Write the application installation file
7. Transfer the application to the Professional
8. Install, execute, and debug the application on P/OS
9. Create a P/OS diskette with the Application Diskette Builder

First, you must decide whether to create a diskette or disk-based P/OS application, or both. The factors involved in this decision are presented in the next chapter.

Next, you design the application and its documentation. The Application Design Guide provides insights into design issues. It includes a sample application which illustrates the discussion.

To guide you in writing documentation for your application, the Tool Kit includes the Documentation Guidelines.

After deciding which system you want to create your application for, you can write the application and the application user interface. Chapter 4 describes all the tools you will need to accomplish this step. Many of the application development tools are described in separate documents or document sets. Others are fully described in Part III of this manual. Chapter 4 lists documentation references for each tool.

The Tool Kit provides an optional mechanism which allows Producers to protect their applications from unauthorized reproduction. The producer can build a subprogram into his application which will automatically be called by P/OS when the end-user attempts to install or run the program. The subprogram is passed the Hardware CPU ID number as a parameter. The subprogram performs an encryption on the CPU ID which yields an absolute key. When the end-user installs the application, he is required by the INSTALL program to enter a key

## P/OS CONCEPTS FOR APPLICATION DEVELOPERS

(called the "tentative key") which he has received from the vendor. If the tentative key does not match the absolute key, the application will not be installed.

If the keys match, then the tentative key is stored in the application as it is installed. Each time the user attempts to run the application, the stored key is retrieved and re-validated against the CPU ID. If the validation fails, the application is scratched from memory, and the user is advised to re-install the package. This prevents a user from sharing his diskettes with another user.

The Producer must provide each authorized user with the key which will unlock the application so it may run on a given CPU. The key may be distributed in many ways. For mail-order business, the purchaser should be requested to enclose his CPU ID number on his order form. The resulting key can be printed on a label and sent with the distribution kit. For volume/retail business, a telephone registration system may be the best method for tracking sales and distributing keys. The vendor will call the registration center, giving the name and CPU ID of the purchaser. The Producer's center will record the customer information, and run the end-user's CPU ID through the encryption algorithm. The resulting key will be read to the vendor, who can then give it to the user.

Application protection is optional. If you wish to distribute unprotected copies of your application, you need not write the encryption program. In this case, in the Application Diskette Build phase of program development answer "NO" to the question, "Is Key Protection desired?".

Before you go on, consider the terminal you will be using to create the application. One way to create a file that contains program language source statements is to log on to the host from a terminal and use a text editor to create the file. Another way to do this is to use the Professional. You can either use the Professional as a workstation (as a terminal to the host) or as an independent system.

As a Workstation, through P/OS Communications Facility you can log on to the host and use any editor to create source code. Without leaving the Professional, you can create, compile, and build the task image on the host. To run and test the application, you can transfer it to the Professional using the P/OS File Transfer Facility.

As an independent system, you can use PROSE (the P/OS Editor) to create your source file on the Professional and store it in a disk file. You can examine, delete, and change it until you are ready to compile and build the application. Then, use P/OS File Transfer to move your file to the host and build the task image.

To determine whether to use the Professional or another terminal to create the application, consider whether you want to use the 8-bit characters from the DEC Multinational Character Set which the Professional supports.

The DEC Multinational Character Set represents characters with an

## P/OS CONCEPTS FOR APPLICATION DEVELOPERS

8-bit code. Tool Kit hosts and their utilities support 7-bit character representation. However, all of the languages and utilities in the Tool Kit support 8-bit characters. (Note that MACRO-11 does not support 8-bit characters as input in source programs.)

To generate 8-bit character codes, either in a file or as interactive input to a Tool Kit utility, you must use the Professional. If you are using a VT100 or another terminal to create applications on the host, you cannot enter or display 8-bit characters on it. Chapter 2 describes the DEC Multinational Character Set in more detail.

When you are ready to create application tasks, you must use the host Task Builder. Chapter 5 illustrates the task build procedure.

To run the application on P/OS, the application must have an application installation file. Chapter 6 describes how to create one.

To run and test the application, transfer it to the Professional using P/OS File Transfer. Chapter 7 illustrates this procedure.

After transferring the application tasks and installation file, you can install, execute, and debug the application on P/OS. Chapter 8 describes how to do this.

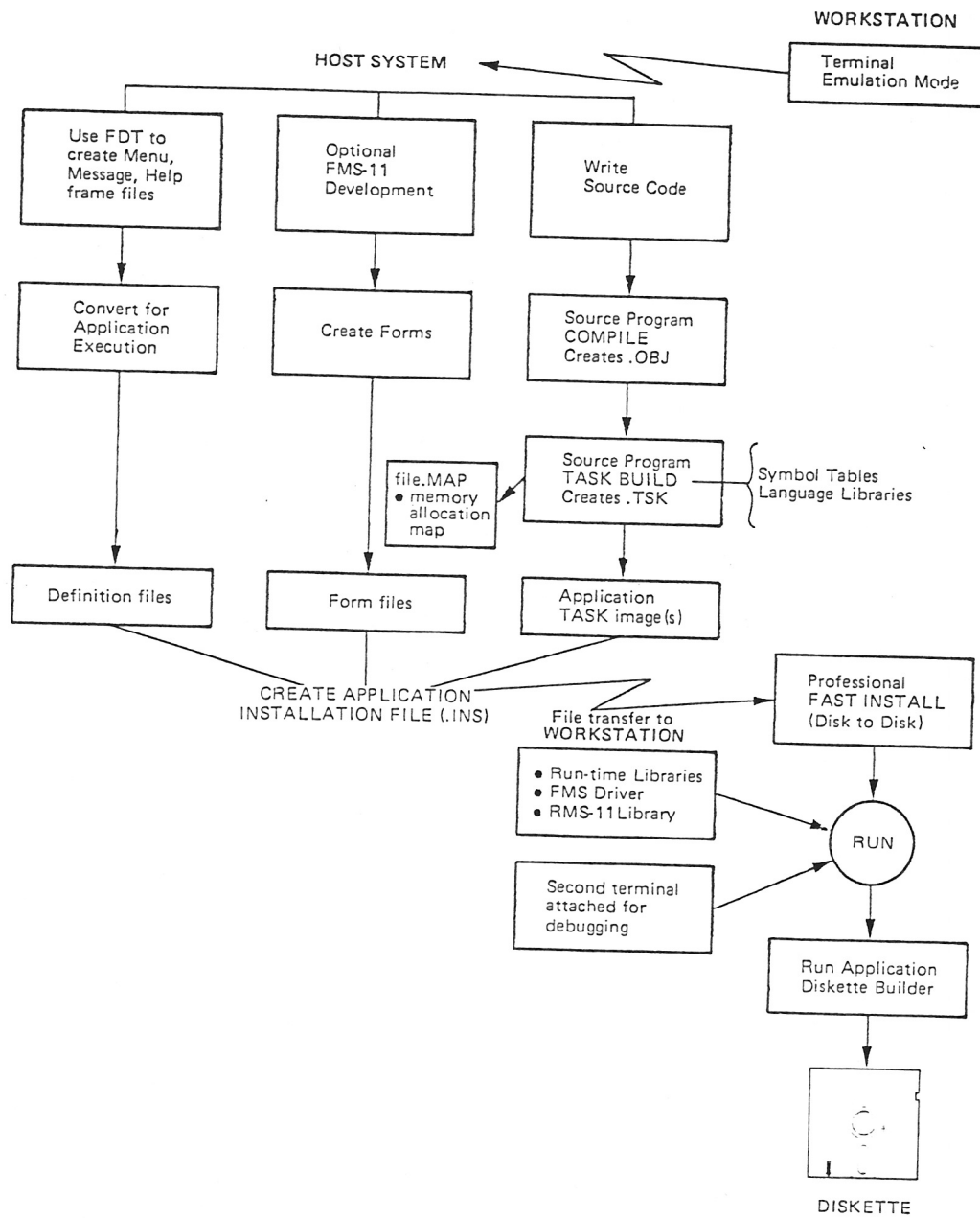
When the application is debugged, you can create a P/OS diskette with the Application Diskette Builder. Chapter 9 describes this procedure.

The following figure shows how to use the Workstation to create a P/OS application program.





Figure: Program Development Cycle





## CHAPTER 3

### SELECTING SOFTWARE COMPONENTS FOR A PROFESSIONAL APPLICATION

This chapter presents options an application developer has in combining an application with required and optional software for any individual hardware configuration.

#### 3.1 PROFESSIONAL HARDWARE CONFIGURATIONS

The Pro325 hardware supports the diskette drive only. The Pro350 hardware supports the dual diskette drive and the optional 5 MB disk. The Professional operating system (P/OS) may be configured as diskette or disk-based to support the Professional hardware configurations.

#### 3.2 PROFESSIONAL SOFTWARE CONFIGURATIONS

Diskette-based P/OS runs on both Pro325 and Pro350. Disk-based P/OS runs on Pro350 if the optional hard disk is added.

For diskette-based P/OS, the system and services reside on the base diskette. For most operations, the base or system diskette must be inserted and remain in the drive while P/OS is running. For other operations, such as diskette-to-diskette file copy, system software is fixed in memory so the base diskette may be removed and replaced by a data diskette.

Application tasks targeted for the diskette-based system must be pre-installed on the base diskette with the operating system. If a user wants to use an application that does not reside on the base diskette, he selects RUN APPLICATION FROM ANOTHER DISKETTE from the menu. To run the application, the second diskette must be inserted and started. To allow use of a data diskette, an application task may fix system software in memory or it may spawn a task from the second diskette, allowing that diskette to be removed. The System Reference Manual explains the FIX\$ directive and Parent/offspring tasking in detail.

For disk-based P/OS, the system resides on the disk so the dual diskette drive is free for use for data diskettes. The disk in this

configuration is equivalent to the base diskette; the disk is the base or system disk. The disk-based system has an Install program to copy applications from a diskette to the disk. As applications are installed, the application name is entered on a menu. Thereafter, the user invokes the application from the menu and saves the diskette as a backup copy of the application.

Application tasks targeted for diskette-based P/OS must be bundled with the system on a diskette. You may select one or more of the callable P/OS services, one configuration of RMS, and one OTS to be task built with the application task. All selected components must be copied to the diskette.

Application tasks targeted for disk-based P/OS may reference one or more of the callable P/OS services and be built against the required OTS and full RMS. However, except for Communications, the target system includes these software components. They are not built into your application's task image. (Task Builder options to reference these libraries are presented in Chapter ?)

Table 3-1 shows the possible configurations of software components on media for disk and diskette-based P/OS.

For diskette-based P/OS, the notation On base diskette indicates that a software component must reside on the base or system diskette.

For disk-based P/OS, system software components reside on the system or base disk. The notation On Disk indicates that a software component is part of the target system. The notation On Application Diskette indicates that a software component is part of the application task which is installed into the system.

In the table, the lefthand column lists the software components which can be combined to create an application package. The righthand columns list the systems.

Table 3-1: Application Packages

SOFTWARE COMPONENT	DISKETTE-BASED P/OS	DISK-BASED P/OS
Application task image(s)	On base diskette or on supplementary diskettes	On application diskette to be installed on disk
Application's Menu, help, message files	On base diskette	On application diskette to be installed on disk
Application's data files, form files	On base diskette or on supplementary diskette	On application diskette to be installed on disk
P/OS (RSX/Exec) (ProDispatcher)	On base diskette	On disk
Resident libraries		
Language Object Time System	On base diskette	On disk
P/OS Services	On base diskette	On disk
RMS Full	Option on base diskette	On disk
RMS Subset	Option on base diskette	Not an option
Callable P/OS Services		
File Services	Optional on diskette	On disk
Print Services	Optional on diskette	On disk
Disk Services	Optional on diskette	On disk
Editor	Optional on diskette	On disk
Communications facilities	Optional on diskette	Optional on application diskette to be installed on disk
Callable Sort	Optional on diskette	Optional on application diskette to be installed on disk

If you are writing an application for both the disk and diskette-based systems, two versions of the application must be created. Most of the source code can be used for both versions; however, at a minimum, the task building process is different for each version of the application. Chapter 7 explains how to use the Task Builder to

rebuild an application to run on both systems.

The applications may be packaged on diskettes according to their size. If they fit, applications for both hardware configurations may reside on a single diskette and be sold for one price.

#### ISSUES:

1. For diskette-based P/OS, does the developer select on language Object time system to include on diskette?
2. What is the RMS subset?
3. Application Diskette Builder has to ask which language OTS the application requires.
4. Are File and Disk services callable?
5. This table will show in-core sizes of P/OS libraries and modules; disk layout.
6. Does diskette-based system have an INSTALL program?
7. What else needs to be said here?

## CHAPTER 4

### TOOLS FOR WRITING THE APPLICATION

This chapter introduces the software tools you will need to develop P/OS applications. Each software development tool is either fully described in this manual or it is described in a separate manual or manual set. References are provided to complete documentation for each software tool. The Tool Kit Documentation Directory contains abstracts for the separate manuals referenced here.

If you have never developed a program on RSX-11M/M-PLUS or VAX/VMS, the RSX-11M/M-PLUS Guide to Program Development and VAX-11/RSX-11 User's Guide provide orientation.

The Tool Kit offers the following program development tools:

1. Programming Languages
2. Forms Management System (PRO/FMS-11)
3. Record Management Services (PRO/RMS-11)
4. PRO/Sort
5. P/OS Services Library
6. PRO/Graphics Library
7. RSX Macro Library and SYSLIB (System Object Library)
8. Frame Development Tool (FDT)

#### 4.1 PROGRAMMING LANGUAGES

Tool Kit development languages include PRO/BASIC-PLUS-2, PRO/DIBOL, and PDP-11 MACRO-11.

#### 4.1.1 PRO/BASIC-PLUS-2

PRO/BASIC-PLUS-2 is a programming language that uses familiar words in simple statements and familiar mathematical notations to perform operations.

If your Tool Kit host is VAX/VMS, use PRO/BASIC-PLUS-2 in Compatibility mode. The related documentation listed at the end of this section describes operational differences between BASIC-PLUS-2 on RSX-11M and BASIC-PLUS-2 on VAX/VMS in Compatibility mode.

The PRO/BASIC-PLUS-2 language processor is composed of a compiler and an Object-Time System Library. In addition to the elementary BASIC statements, PRO/BASIC-PLUS-2 features terminal-format files, virtual arrays, RMS I/O, extensive string manipulation functions, matrix package handling operations, and long variable names. Statement types include IF...THEN...ELSE combinations, ON ERROR condition handlers, and the statement modifiers IF, WHILE, UNLESS, UNTIL, and FOR. User-defined functions are supported. Multiple statements may be included on one line, and a statement may cover multiple lines. In addition, external subprograms may be accessed through the SUB, CALL, COMMON statements.

##### Documentation

##### PRO/BASIC-PLUS-2 Documentation Supplement

The PRO/BASIC-PLUS-2 Documentation Supplement describes the special features of PRO/BASIC-PLUS-2.

##### PDP-11 BASIC-PLUS-2 Installation Guide/Release Notes

##### BASIC Reference Manual

##### BASIC User's Guide

##### BASIC on RSX-11M/M-PLUS Systems

#### 4.1.2 PRO/DIBOL

PRO/DIBOL is DIGITAL's Business Oriented Language. It is similar to COBOL in that it has a DATA DIVISION and it uses English-like procedural statements. DIBOL enables data manipulation, arithmetic expression evaluation, table subscripting, record redefinition, external and internal calls to other programs, intertask communication, and random, sequential, and indexed access to files. Additionally, DIBOL includes a comprehensive online debugging utility with which the program developer can quickly isolate and correct programming errors.

##### Documentation:

##### Introduction to the PRO/DIBOL Language



PRO/DIBOL Language Reference Manual

PRO/DIBOL Tool Kit User's Guide

PRO/DIBOL Release Notes and Installation Guide

#### 4.1.3 PDP-11 MACRO-11

MACRO-11 is a PDP-11 assembly language processor supplied with the host system. The assembler processes source statements sequentially, generating binary machine instructions and data words or performing assembly-time operations (such as macro expansions) for each statement.

Documentation:

PDP-11 MACRO-11 Language Reference Manual

#### 4.2 FORMS MANAGEMENT SYSTEM (PRO/FMS-11)

The Forms Management System (PRO/FMS-11) is used by programmers to create forms for handling user inquiry or response during application execution. PRO/FMS-11 can be used to format user input for use by the application, and to display application output in an easily understandable style. PRO/FMS-11 features include:

- o Forms can be designed directly and interactively on the video screen, eliminating the need to lay out the form on paper, code form language statements, compile, debug, enter corrections, and so forth.
- o Forms can be modified without having to recompile or relink the application, or reprocess collected data. This simplifies modifications and reduces program maintenance requirements.
- o The storage of form libraries on disk reduces program memory requirements.
- o The complexities of terminal I/O are removed from the application program. Each field in a form includes help, display, protection, and validation information to ensure the integrity of data returned to the application program.

PRO/FMS-11 includes the following software components:

- o The Form Editor (FED) is used to create and modify video forms. You create forms by typing them on the screen as they are to appear to the application user. The Form Editor runs on the host system only.

- o The Form Utility (FUT) is a multifunction utility program for creating form description files which can be printed and form libraries (runs on the host system only).
- o The Form Driver (FDV) is a set of subroutines called from application programs to display forms, perform input and output operations, respond to HELP requests, and so forth. The Form Driver runs on the Professional.

The Form Driver calls for the Professional terminal are different from FMS-11 calls for the VT100 terminal, changing the way in which an end user interacts with a form.

The EMS/Pro Documentation Supplement describes the special features of FMS/Pro.

Documentation:

EMS-11/RSX Release Notes

EMS/RSX-11 Software Reference Manual

EMS/RSX Mini-Reference

EMS/Pro Documentation Supplement

#### 4.3 PRO/RECORD MANAGEMENT SERVICES (RMS-11)

Record Management Services (RMS-11) provides a set of run-time service routines and utilities that enable direct, sequential and multi-keyed access to data files and that enable your program to define, populate, update and maintain files on direct access devices. RMS-11 supports sequential and multiple keyed access to data files.

Record Management Services (RMS-11) provide a record interface between the operating system and your application program. RMS-11 comprises a set of run-time service routines and utility programs that enable keyed access data files to be defined, populated, updated, and maintained on direct access storage devices. The utility programs are not part of the Tool Kit.

Documentation:

RSX-11M/M-PLUS RMS-11 Primer

RSX-11M/M-PLUS RMS-11 User's Guide

RSX-11M/M-PLUS RMS-11 Macros and Symbols

ISSUES:

1. If utilities are not part of Tool Kit we should pull the RMS-11 Utilities manual from the set. (RSX-11M/M-PLUS RMS-11 Utilities)

#### 4.4 PRO/SORT

PRO/Sort is a general-purpose sorting program for use by applications on the Professional. Application tasks can use an RSX/Exec directive (SPWNS) to invoke PRO/Sort, passing the name of a file containing sorting commands. PRO/Sort is not available to P/OS end users. It must be bundled with the application when you create the application tasks on the host system.

Documentation:

Chapter ?

#### 4.5 P/OS SERVICES LIBRARY

The P/OS Services Library contains subroutines for displaying menus, help, and message frames. Other routines provide facilities for obtaining input and output file names from the end user, and for translating character sequences from keystrokes into meaningful tokens. The P/OS Services library is one of the resident clustered libraries in the P/OS system. Chapter ? lists the services in the library.

Documentation:

Chapter ?

ISSUES:

1. What are the contents of the P/OS library on the host?

#### 4.6 GRAPHICS LIBRARY

P/OS Graphics is a library of graphics routines for the application programmer to use to create graphics. Features include:

1. Graphics primitives based on the SIGGRAPH CORE standard, callable from the Tool Kit languages
2. Graphics text

3. Variable character attributes and user definable fonts
4. Picture segmentation
5. window and viewport functions
6. Eight simultaneous colors with the Extended Bit Map Option

Documentation:

Graphics Manual

#### 4.7 RSX MACRO LIBRARY AND SYSLIB (SYSTEM OBJECT LIBRARY)

To be supplied.

Documentation:

RZDS System Reference Manual

#### 4.8 FRAME DEVELOPMENT TOOL (FDT)

The Frame Development Tool (FDT) allows the application developer to create a user interface for an application. Through a series of forms, you specify the actual text to be displayed on menus and help frames and information relating to the manner and timing of the frame displays.

Documentation:

Chapter ?

## CHAPTER 5

### TASK BUILDING THE APPLICATION ON THE HOST

The RSX Task builder creates a task image file that can be loaded into memory and executed. The Task Builder performs the following:

1. Links object modules
2. Resolves references to system or user libraries of object modules
3. Allocates virtual address space to the task
4. Produces an optional Task Builder map that describes memory allocation, object modules, and global symbol references
5. Produces an optional Symbol Definition File
6. Builds an overlaid task
7. Maps the task to shared regions of memory

PRO/BASIC-PLUS-2 generates its own task builder command files. You must edit the BASIC-PLUS-2 command files to reference the libraries required to perform the task build procedure.

Examples of Commands.

TO BE SUPPLIED.

Related Documentation

RSX-11M/M-PLUS Task Builder Manual



## CHAPTER 6

### CREATING AN APPLICATION INSTALLATION FILE

Before you can run an application on the Professional, you must create an application installation file. The application installation file identifies all the application files and the hardware configuration required by the application.

The installation file is used by the disk-based P/OS Install program to add the application to the list of executable tasks. The end user is given the option of installing applications into the Additional Applications Menu or into an open slot on the main menu on the disk-based P/OS.

Chapter 7 explains how to use the Install program.

The application installation file has a .INS file type. It has the following format:

```
!This is the installation file
FILE filename
FILE filename/value
NAME menu name
INSTALL filename/LIBRARY
INSTALL filename/TASK
OPTIONS option,option...
ASSIGN MENU filename
ASSIGN MESSAGE filename
ASSIGN HELP filename
ASSIGN LOGICAL name string
RUN filename
```

Commands in the Application Installation File must appear in the order shown and each line must have the following syntax:

!Text                      The exclamation point (!) precedes an optional comment.

FILE filename

filename                  A one to nine character file name identifying each file to be copied to the diskette. The installation

file can contain multiple lines of this type.

#### FILE filename/value

/value

There is no default for /value. The value must be one of the following:

/KEEP keep this file if the application  
is removed  
/DEL delete this file if the  
application is removed

#### NAME menu name

menu name

A 1 to 40 character application name as it will appear in the end user's menu. The end user has the option of changing this name when the application is installed.

#### INSTALL filename /LIBRARY

filename

A nine character file name of a resident library associated with this application. The file may contain multiple lines of this type, all of which must appear before lines with /TASK. If your application uses the callable services, each of them must be installed with this command line.

#### INSTALL filename /TASK

filename

A one to nine character file name of an application task image. The file may contain multiple lines of this type.

#### Options,option,option

Optional. Specifies the hardware options required by this application. Valid options include: TMS (Telephone Management System)  
FPP (Floating Point Processor)  
LA100 (Graphics output printer)  
VR241 (Color Monitor)

#### ASSIGN MENU filename

filename

Optional. A one to nine character name of the file containing your application's menus. This command may be used to assign a default menu definition file to your application and may replace use of the open menu file directive at application run-time.

#### ASSIGN MESSAGE filename

filename

Optional. A one to nine character name



of the file containing error and status messages for your application. This command may be used to assign a default message definition file to your application and may replace use of the open message file directive at application run-time.

#### ASSIGN HELP filename

filename	Optional. A one to nine character name of the file containing help information for your application. This command may be used to assign a default help definition file to your application and may replace use of the open help file directive at application run-time.
----------	---

#### ASSIGN LOGICAL logical name "string"

logical name	A name to be assigned a string name. The name could be a device or file name.
"string"	The name to be assigned to the logical name.

#### RUN filename

filename	A one to nine character name of the task image file which should be executed first for this application.
----------	--

#### ISSUES:

1. ASSIGN LOGICAL name in Installation File -- what is this used for?
2. What are valid hardware options?
3. Install for disk-based P/OS only?
4. What does the application developer do to create a menu entry for diskette-based applications?
5. What are the names of the callable services for command line INSTALL filename/TASK?



## CHAPTER 7

### TRANSFERRING THE TASK TO THE PROFESSIONAL

To execute the application on the Professional, use Professional Communications Facilities to transfer the file from the host to the target.

The Communications Facilities Manual describes how to use Professional communications for file transfer in terminal emulation mode. The file transfer program provides an easy way to transfer the application task and its associated files to the Professional.

When the transfer is completed, the application task image (and any other requested files) reside on the Professional disk.

During program development, you can use the Terminal Emulation Program to work on the host, then switch back to local mode to execute the application. The Terminal Emulation program enables the Professional to behave like a terminal to the host software development system.

Example of transferring a task image to Professional.

TO BE SUPPLIED.



## CHAPTER 8

### INSTALLING, EXECUTING, AND DEBUGGING THE APPLICATION

After you have transferred the task image of your application and the application installation file to the Professional, you can use the Fast Install program to install your application tasks. The Fast Install program is similar to the Install program provided to end users.

Both Fast Install and Install identify all the executable images and application data files to the system. As applications are installed, the application name is placed on a menu. Afterward, the application can be invoked from the menu.

The end user Install program installs applications from diskette; Fast Install installs applications from disk.

#### 8.1 FAST INSTALL

The Install program performs the following:

1. Adds the name of the application as an entry in the specified menu
2. Optionally assigns Help, Message, and Menu file so they can be located during run time. If these files are not assigned when the application is installed, they are assigned at run time.

Example of how to use Fast INSTALL.

TO BE SUPPLIED.

ISSUES:

1. what is the user interface for Fast INSTALL?

## 8.2 EXECUTE THE APPLICATION

After you have installed the application, you can execute it by selecting it from a menu.

### ISSUES:

1. Which menu does Fast INSTALL put the application name on?
2. What options does Fast INSTALL offer for placement of application name on menu?

## 8.3 DEBUG THE APPLICATION

After you have located errors in logic or execution in the application, you can use the Terminal Emulation program and File Transfer to repeatedly edit, compile and task build the application on the host, transferring the task image back to the Professional when you want to retest it.

The high-level languages provide interactive debugging commands to help you locate run-time errors in your program. These commands allow you to check program operation and make corrections. The reference documentation describes these commands and their use.

The Professional has a special feature that allows you to debug an application using the Professional and another terminal. To use a terminal for debugging, attach it to the printer port on the back of the Professional System Unit.

You can redirect debugger I/O away from the Professional to the other terminal using language-specific commands. Program I/O continues to be displayed on the Professional. This allows graphics or forms on the Professional to be displayed without disruption from debugger messages and prompts.

### ISSUES:

1. BASIC-PLUS-2 has the REDIRECT command to use the printer port. What do FORTRAN and DIBOL use?

## CHAPTER 9

### APPLICATION DISKETTE BUILDER

After you have fully debugged your application, you can create a P/OS diskette with the Application Diskette Builder. The end user installs or runs the application from the diskette.

The Application Diskette Builder is an interactive program that runs on the Professional. The Diskette Builder copies the task image of your application and related files to a formatted diskette. If you have created more than one application or if you have created a diskette and disk-based version of your application, each application may reside on its own diskette or together on one diskette. The packaging of applications on diskettes is limited only by their size.

#### 9.1 RUN THE APPLICATION DISKETTE BUILDER

If the application is error-free, run the Application Diskette Builder.

The Application Diskette Builder performs the following tasks:

1. If the target system is the diskette-based system, copies the ready-made bootable RX50 to the diskette.
2. Copies the application installation file to the diskette
3. Copies the application files named in the installation file to the diskette

Example of dialogue:

TO BE SUPPLIED

ISSUES:

1. Is this what the Application Diskette Builder does?

2. what's the dialogue/user interface for the Diskette Builder?

## 9.2 LABEL DISKETTES

When your application is error-free and you have created diskettes, put a label on the diskette with a description of the application. The label should state the following:

1. Application Title
2. Software Identification Number
3. Minimum hardware configuration required to run application
4. Hardware options required
5. Media space required by application.

### ISSUES:

1. What needs to be said here?



## PART III THE TARGET SYSTEM AND SOFTWARE TOOLS

The chapters in Part III describe programming conventions for the P/OS operating system (Chapter ?), callable P/OS and Tool Kit Services (Chapter ?), the Frame Development Tool (Chapter ?), and P/OS Services Library (Chapter ?).



## CHAPTER 10

### PROGRAMMING CONVENTIONS FOR P/OS

This chapter describes the following topics:

- o Function Key Usage
- o Application Multi-tasking
- o DEC Multinational Character Set

#### 10.1 FUNCTION KEY OPERATION

Pressing a function key passes a unique key character sequence to the software. The key character sequence identifies the key that was pressed, and the software performs some action based on the key's setting.

##### 10.1.1 Settings

There are two possible settings for function keys:

1. System function: The key character sequences for these keys are handled by the system; therefore, the application can ignore these keys. The following keys have a system function: PRINT SCREEN, INTERRUPT, and HOLD SCREEN.
2. Type-Ahead: The key character sequences for these keys are passed first to the type-ahead buffer and then to the application's input buffer. Keys not listed as system function keys are passed to the application in the type-ahead buffer.

## 10.1.2 Assigning Settings To The Function Keys

To be supplied.

## 10.1.3 Function Key Operation During Application Execution

Table ? shows how the Function Keys operate when an application is using menu services during application execution.

Table ? : Application Execution

KEY	ACTION
-----	-----
ADDTNL OPTIONS	Displays an additional options menu. If no additional options were defined for this menu, an error message is displayed.
Arrow keys	Up and down arrows move the pointer up or down the list of options.
BREAK	Passed to the application as type-ahead.
CANCEL	Cancels selection.
DO	Passes control to the executing application indicating that the user has made a selection.
EXIT	Passed to the application as type-ahead.
FIND	Passed to the application as type-ahead.
HELP	If defined in menu definition file, menu services displays a help menu or help frame.
HOLD SCREEN	The screen remains still until another key is pressed.
INSERT HERE	Passed to the application as type-ahead.
INTERRUPT	Regardless of the context, if followed by the DO key, stops the interactive application and displays the P/OS Main menu. If the DO key is not pressed, the INTERRUPT key is ignored and the next key is used.
MAIN SCREEN	Passed to the application as type-ahead.
NEXT SCREEN	If defined in help definition file, menu services displays the next help screen.
Numbered function keys	Passed to the application as type-ahead.

PREV SCREEN	If defined in help definition file, menu services displays the previous help screen.
PRINT SCREEN	The screen remains stationary while it is printed on printer.
REMOVE	Passed to the application as type-ahead.
RESUME	If defined, menu services handles while menus are being used.
SELECT	Marks a selection from a multiple-choice menu.
SETUP	Passed to the application as type-ahead.
-----	

## 10.2 APPLICATION MULTITASKING

Tasks can activate one another using an executive service and through language-specific features, such as CHAIN and CALL in PRO/BASIC-PLUS-2.

### ISSUES:

1. For diskette-based P/OS, explain how the application can use an RSX/Exec directive to fix parts of the system in memory to run with the system diskette out. (Compare with floppy-to-floppy file copy.) Reference the
2. Chaining: use RPOI RSX Directive (alternative to exit, see RSX Exec Reference Manual). (PRO/BASIC-PLUS-2 and PRO/DIBOL)
3. Use SPWNS directive to activate tasks
4. Installation file names all tasks to be installed with task; callable services must be named here. An option in installation file indicates which tasks should be left active after parent task exits.

## 10.3 THE PROFESSIONAL TERMINAL CHARACTER SET

The Professional terminal supports the DEC Multinational Character Set, which includes the ASCII Graphic Character set and the DEC Supplemental Graphic set. (Note that a graphic character is defined as a noncontrol character, having a visual representation normally handwritten, printed, or displayed.)

The character set is represented as an 8-bit code. The set of character codes gained by using the eighth bit supports alphabets used in major European languages, but not commonly used in English.



For the ASCII subset, the eighth bit is set to 0. For the remaining characters, the 8th bit is set to 1.

Figure 7 shows the Professional character set in a code table. The code table is a matrix of 16 rows and 8 columns for a seven-bit code, and 16 rows and 16 columns for an eight-bit code. The rows correspond to the low four bits of the character code, and the columns correspond to the upper three or four bits of the code.

The DEC Multinational Character Set is in columns 0 to 15. The ASCII Graphic Character Set is in columns 0 to 7. The DEC Supplemental Graphic Set is in columns 10 to 15.

The ASCII control characters appear in columns 0 and 1; the additional control character set is in columns 8 and 9.

Figure 7: DEC Multinational Character Set

	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
0	NUL	DLE	SP	0	@	P	'	p		DCS		°	À		à	
1	SOH	DC1	!	1	A	Q	a	q		PU1	i	±	Á	Ñ	á	ñ
2	STX	DC2	"	2	B	R	b	r		PU2	¢	²	Â	Ò	â	ò
3	ETX	DC3	#	3	C	S	c	s		STS	£	³	Ã	Ó	ã	ó
4	EOT	DC4	\$	4	D	T	d	t	IND	CCH			Ä	Ô	ä	ô
5	ENO	NAK	%	5	E	U	e	u	NEL	MW	¥	μ	Å	Õ	å	õ
6	ACK	SYN	&	6	F	V	f	v	SSA	SPA		¶	Æ	Ö	æ	ö
7	BEL	ETB	'	7	G	W	g	w	ESA	EPA	§	•	Ç	Ɔ	ç	œ
8	BS	CAN	(	8	H	X	h	x	HTS		⌘		È	Ø	è	ø
9	HT	EM	)	9	I	Y	i	y	HTJ		©	1	É	Ù	é	ù
10	LF	SUB	*	:	J	Z	j	z	VTS		ª	º	Ê	Ú	ê	ú
11	VT	ESC	+	;	K	[	k	{	PLD	CSI	<<	>>	Ë	Û	ë	û
12	FF	FS	,	<	L	\	l		PLU	ST		¼	Ï	Ü	ï	ü
13	CR	GS	-	=	M	]	m	}	RI	OSC		½	Î	Ý	í	ÿ
14	SO	RS	.	>	N	^	n	~	SS2	PM			Î		î	
15	SI	US	/	?	O	_	o	DEL	SS3	APC		¿	Ï	ß	ï	





## CHAPTER 11

### CALLABLE SERVICES

While your program is executing, it can invoke some of the facilities offered by P/OS to end users. In addition, the Tool Kit includes a callable sorting utility which is not available to end users.

The callable services include File, Disk, Print Services, Communications, PROSE (the text editor), and ProSort. The next sections introduce these facilities and describe the calls your application can use to invoke them.

#### 11.1 FILE, DISK, AND PRINT SERVICES

The P/OS File Services use Record Management Services (RMS) to perform file and disk manipulations. Using RMS-11, your application can access these file and disk services directly, through RMS, or it can call on the P/OS File services to perform these operations.

Files are created by applications, and all applications must follow the system conventions for file manipulation.

##### 11.1.1 Conventions

The P/OS file specification is as follows:

device:[directory]filename.typ;version

Each element of the file specification is described in this section.

**File Name** A P/OS file name is up to nine characters long. File names can contain only the characters A to Z and the numerals 0 to 9. Uppercase and lowercase letters are valid (lowercase letters are converted to uppercase). Embedded spaces, punctuation, and other special characters are not allowed in file names.

**File Type** A P/OS file type is a 3-character extensions to the file name. By convention, the file type describes the internal organization of the file. The internal organization of a file

determines which application the end user can use to work with it. For example, the printer cannot print an executable file.

The end user chooses a file type from a list of meaningful names, and File Services translates the name into an internal file type. The meaningful name is displayed in a file directory on the screen. If no meaningful name is chosen, the 3-letter file type is displayed.

Table ? shows the user-visible file types that are supported:

Table ? : User-visible File Types

FILE TYPE	INTERNAL REPRESENTATION	USE
BASIC Program	.BAS	user/BASIC
data	.DAT	user/BASIC
document	.DOC	user/Editor
text	.txt	user/Editor

Table ? shows a partial list of file types that are supported and used only by the system and by applications:

Table ? : System/Application File Types

FILE TYPE	INTERNAL REPRESENTATION	USE
forms library	.FLB	application
system file	.SYS	system
task image	.TSK	system/application
converted help file	.HLP	application
converted message file	.MSG	application
converted menu file	.MNU	application
Basic virtual array file	.VIR	ProBasic
DIBOL data file	.DDF	PRO/DIBOL

**File Size** The size of a file is limited by the medium on which it resides. Files cannot be spread over more than one medium, except when a file residing on the disk is saved to a set of diskettes (file backup).

**Media** Files reside either on the disk or on a diskette. For diskette-based P/OS, the base diskette must reside in one diskette slot during system use. A storage diskette can be loaded in the other slot. Programs cannot be executed from storage diskettes on disk-based P/OS.

The P/OS system software always resides on the disk. The disk and diskettes can both be used for the user's data storage and files.

**Media Names** Each disk and diskette has a physical, logical, and volume name. Physical and logical names are properties of the media and their matching disk drives. By convention, a logical name is a meaningful equivalent of a physical device name. Physical and logical names do not change. Your application can use either the physical or logical device name for a device assignment (device:) in a file specification.

The end user assigns a volume name to a storage diskette when the diskette is first used. Volume names are assigned to system diskettes and application diskettes by the P/OS or by an application. The disk has the volume name BIGVOLUME. System diskettes produced by DIGITAL begin with the letters "DEC." Volume names of storage diskettes can be changed.

Table ? lists physical device names in the left-hand column, P/OS logical names in the middle column, and hardware types in the right-hand column.

Table ? : Physical and Logical Device Names

Physical Name	Logical Name	Front Panel	Hardware Type
DZ1:	DISKETTE1	1	RX50 diskette
DZ2:	DISKETTE2	2	RX50 diskette
DW0:	BIGDISK		RD50 disk

**File Directory** Files are organized in groups called directories. Directory names are one to nine characters in length and may be enclosed in brackets or in the signs < and >.

A directory is an index of a group of files that belong together. Each disk contains at least one directory. The directory contains an entry for each file in that directory and maintains a record of each file's name, type, size, protection status, location, version number, date created, date of last change, and date of last backup.

Directory names can be assigned by the user. The preassigned directory, USERFILES, is a starting point; this directory can be deleted or not used. Initially, this directory contains the names of all the files created by the end user.

When the end user starts working on the system, files created during the session are automatically assigned to the USERFILES directory, unless the default directory has been changed to a different directory. This assignment can be changed during the session.

You may create a directory from your application. If your application creates files that serve no useful purpose after the application terminates, the application should delete those files before exiting. If your application passes through several phases, opening and closing files in the user's directory to pass data between phases, then the application should provide an easy method to the user for deleting those files.

**Version Numbers** When a file is created, it is assigned version number 1. When the file is subsequently modified, such as when it is edited, a new version, 2, of the file is created. Now version 1 and version 2 are on the disk. As versions of a file are created, their predecessors remain on disk until the end user or your application deletes them.

The version number must be separated from the file type by a semicolon (;) or by a period (.).

**Defaults** The system assumes certain file characteristics if the user does not specify them. File location (disk name and file directory), protection code, and type are characteristics that can be assumed. Changes to default characteristics can be saved and applied during subsequent sessions.

The defaults may be overridden during each file manipulation through explicit specification by the application.

#### 11.1.2 Services

**Copy file** Duplicates a file on the disk or diskette. The current disk, directory, and file names are the defaults for the output file. The output file must be copied to a different version number.

**Append file** Attaches a copy of one file to the end of another file. To be concatenated in this way, files must have sequential file organization and the same file type. In addition, the disk on which the file being appended resides must have enough space to contain both files.

**Rename file or change file type** Changes the name or file type of a file. The disk and directory names cannot be changed.

**Delete file** Erases a file.

**Protect file** Sets the protection status of a file. The default protection code gives a file a protection status that allows change, delete, or change and delete.

**List directory** Displays a list about all files in the specified directory. The list shows file name, file type, file size, date of creation, protection code, and date of last backup. This command can show a list of files, list of directories, and a full directory of files.

**Print** Initiates a print job by submitting one or more file names to Print Services.

## 11.2 OVERVIEW OF P/OS COMMUNICATIONS FACILITY

P/OS communications includes the following features:

1. Terminal Emulation
2. System-to-system file transfer
3. Telephone Management System

Application writers can use Communications run-time services as building blocks for new communications applications. Application programs can access many of the communications facilities through system service routines. Other facilities may be invoked as tasks by the application during program execution. Communications facilities are fully described in the *Communications Manual*.

Communications facilities use the Phone Book to establish connections between systems for file and data transfer, terminal emulation, and voice telephone management.

The data base is available to any application program that supports RMS indexed files. For example, it is used by a communications application to find the number of a person to dial. The terminal emulator uses it to store host system phone numbers and characteristics.

The P/OS communications software interfaces described in this section are supported by the following functional components:

1. Communication Services: A library of routines which may be called directly from an application program to perform communications services on the Professional. These services provide a set of services nearly equivalent to those available through the user interface menus for the Communication Facility.
2. Communication Utility Task (CUT): A utility that implements library service calls and handles unsolicited telephone connect requests and file transfer requests.
3. Dumb Terminal Emulator (DTE): A program that handles the VT102/VT125 file emulation.
4. File Transfer programs: A program that handles the transfer of files between the Professional and a host or between two Professionals.

### 11.2.1 General Description

with the exception of those routines described in the section on TMS services, all communications services may be used to request functions on both the communications port and TMS lines in a transparent manner. The user is first required to call CCBALC to request the allocation of a call control block, the address of which is passed back by the CCBALC routine. The call control block must be used in subsequent calls to communications service routines. An application program using communications services must incorporate the communications impure area into the task by linking the module "COMROT" into the task root. A linkage to the impure area is set up via the SVEXT pointer.

An AST routine is located in the impure area. If the user wishes to service unsolicited events on the communications line, the CCATA routine may be called to attach the line. All unsolicited events are notified via this AST routine. The AST routine saves the event notification in a user-designated word defined when the CCATA routine is called. In addition the unsolicited event flag, which the user specified in the call to the CCATA routine, is set. The program may wait on this event flag for an unsolicited event.

If you want to control more than one communications line simultaneously, CCBALC must be called for each line being used. The address of a new Call Control Block is returned each time (maximum of three). The task may wait for unsolicited events on one or more lines by using the WTLOS directive.

The user may perform either synchronous or asynchronous I/O to the communications line driver. The I/O type is indicated by the 'efn' parameter in the CCTXD and CCRXD routine call lines. For asynchronous I/O, the event flag specified by the user is set on completion of the I/O transfer. Note that because the two-word status reply is not set until completion of I/O it should not be tested until the termination event flag is set.

If you are using CODEC input on a TMS line, note that the voice signal is digitised at 4kb per second. It is only possible to avoid loss of voice data if asynchronous I/O is performed and the input requests are double buffered. It is recommended that each input request specifies a 4kb buffer. This allows the user application 1 second to process the buffer before the second buffer fills.

If the FTSND, FTSNDw, FTGET or FTGETW routines are being used for file transfer then the communications line must not be attached because the transfer will never be started.

### 11.2.2 Status Returns

All routines return a two-word status, the status parameter is always passed as the first parameter in the call line. The first word contains a code indicating success, or the source of any error. The second status word contains a code indicating the specific error.

Some services translate into QIO calls to the appropriate driver; while others, such as file transfer requests, use the callable image communication routines to request the communications utility task to perform functions on behalf of the application.

All routines return some status information that indicate the degree of success or failure. Those routines that issue QIO calls to the driver return an I/O Status Block (IOSB) the service routine checks that the I/O has terminated successfully and indicates this in the first status word as either a success or error status. In the case of an error, the second status word is loaded with the contents of the first word of the IOSB. The application may check the first byte of this word to determine the specific error.

If a service routine can be called to issue an asynchronous QIO, the application must specify an additional two word area for the IOSB. If the service routine is used in its synchronous mode, the termination status is signaled as described above. However, if the service is used in its asynchronous mode the first status word only indicates whether the QIO has been successfully issued. The program must test the IOSB to determine if the I/O terminated successfully after the QIO has completed. This is signaled by the setting of an event flag specified in the original call to the service.

The status codes returned are defined as global symbolic constants in the communications services library. For those languages that allow the definition of external constants, the values will be resolved by the task builder at build time. This document defines all the status returns by their symbolic names. All success returns are positive values, all error returns are negative values.

### 11.2.3 Communications Services Calling Conventions

All subroutines utilize the R5 calling convention and are entered with:

JSR PC,SUBR

The parameter pack is pointed to by R5 and has the following format:

WD.00	byte 0 : Number of arguments
	byte 1 : Undefined
WD.01	Address of arg 1
.	.
.	.
WD.NN	Address of arg N

Registers R0-R5 are undefined on return, SP is restored to its state at entry before exit from a routine. All parameters are passed by reference and data type conversion is performed by the service routine. Use the following format to call communications services:



CALL SUBR (parm1,parm2,... ,parmN)

Optional parameters are shown in [] and may be dropped from the call line of the service described. All routines return a two-word status which indicates the success or failure of the function being requested.

11.2.3.1 File Transfer Statistics Buffer - The file transfer statistics buffer is a 36 byte array in which the file transfer statistics data is returned.

The statistics block has the following format:

WD.00	File size in blocks when complete
WD.01	Current block being written/read from the file
WD.02	Current file function
WD.03	Device mnemonic for the line
WD.04	Unit number
WD.05	Bytes received, double length integer
WD.07	Bytes sent, double length integer
WD.09	Data blocks sent, double length integer
WD.11	Data blocks received, double length integer
WD.13	Byte 0: Data errors inbound Byte 1: Data errors outbound
WD.14	Byte 0: Remote reply timeouts Byte 1: Local reply timeouts
WD.15	Local buffer allocation failures
WD.16	Elapsed time for transfer in seconds as a double length integer

#### 11.2.4 Call Control Services

The services described in this section enable an application program to setup and control a telephone connection for the purpose of voice or data communication. Voice communication is only available via the TMS option.

11.2.4.1 Enable/Disable Communications Option (CCENBL And CCDSBL) - These two routines are used to load and unload the communication port and TMS device drivers. Before you use either of these routines, you must install the device driver tasks into the system. The communications utility task activates or deactivates the designated driver. On activation the driver is given the default characteristics defined for the lines associated with it. If file transfer is enabled for remote file copy or receive, the communications utility task monitors unsolicited connects.

Call:



CALL CCENBL (status,option)

and

CALL CCDSBL (status,option)

Parameters:

status	CS.SUC	Successful
	CE.PRM	Service call parameter error
	CE.DIR	RSX Directive error
		Second status word contains the DSW
	CE.DRV	Failed to load/unload driver
		C2.INS     Driver not installed
		C2.ACT     Driver already active/not active
option	Specifies the driver to be loaded or unloaded.	
	Valid values are:	
	1	communication port Driver
	2	TMS Driver

11.2.4.2 Call Control Block Allocation (CCBALC) - This routine allocates a Call Control Block (CCB) and initializes it with the parameters passed in the call line. The communications line is assigned to the specified lun. The CCB is used to pass parameters between the various communications services.

Call:

CALL CCBALC (status,ccb,lun,dev,unit)

Parameters:

status	CS.SUC	Successful
	CE.PRM	Service call parameter error
	CE.DIR	RSX Directive error
		Second status word contains the DSW
ccb		An integer which is used to return the address of the call control block in the impure data area
lun		Logical unit number for the communications line.
dev		Two byte ASCII string containing the device mnemonic.
unit		Unit number in binary.

11.2.4.3 Attach Line (CCATT And CCATA) - These two routines provide to the application task a mechanism to attach to the communications line. The CCATT routine attaches a line. The CCATA routine attaches a line and declares an AST routine to be used for unsolicited events. The AST routine used by CCATA is copied to the impure area when it is being initialized. If an unsolicited event is signaled, the event code is written to word zero of the ccb for the line on which it occurred and the event flag specified by the user in the call to CCBINI is set.

Call:

CALL CCATT (status,ccb)

and

CALL CCATA (status,ccb,efn)

Parameters:

status	CS.SUC	Successful
	CE.PRM	Service call parameter error
	CE.DIR	RSX Directive error
		Second status word contains the DSW
	CE.IER	I/O Termination error
		Second status word contains the
		IOSB first word
ccb		An integer which contains the address of the call control block.
efn		An event flag to signal unsolicited events.

11.2.4.4 Detatch Line (CCDET) - The CCDET routine detatches a line that has been attached by the CCATT or CCATA routines.

Call:

CALL CCDET (status,ccb)

Parameter:

status	CS.SUC	Successful
	CE.PRM	Service call parameter error
	CE.DIR	RSX Directive error Second status word contains the DSW
	CE.IER	I/O Termination error Second status word contains the IOSB first word
ccb	An integer which contains the address of the call control block.	

11.2.4.5 Set Line Characteristics (CCSMC) - This routine modifies line characteristics to the desired settings.

Call:

CALL CCSMC (status,ccb,ldb,size)

Parameters:

status	CS.SUC	Successful
	CE.PRM	Service call parameter error
	CE.DIR	RSX Directive error Second status word contains the DSW
	CE.IER	I/O Termination error Second status word contains the IOSB first word
ccb	An integer which contains the address of the call control block.	
ldb	Line descriptor block containing the new line characteristics.	
size	The size of the characteristics buffer in bytes.	

11.2.4.6 Get Line Characteristics (CCGMC) - This routine returns the current line characteristics for the specified line.

Call:

CALL CCGMC (status,ccb,ldb,size)

Parameters:

status	CS.SUC	Successful
	CE.PRM	Service call parameter error
	CE.DIR	RSX Directive error
		Second status word contains the DSW
	CE.IER	I/O Termination error
		Second status word contains the
		IOSB first word
ccb		An integer which contains the address of the call control block.
ldb		A buffer into which the line characteristics may be returned in ldb format.
size		The size of the characteristics buffer in bytes.

11.2.4.7 Get/Put Line Configuration Record (CCLCRG And CCLCRP) -  
These routines may be used to retrieve and modify the line  
configuration record for the specified line.

Call:

CALL CCLCRG (status,ccb,ldb,[ttble])

and

CALL CCLCRP (status,ccb,ldb,[ttble])

Parameters:

status	CS.SUC	Successful
	CS.TTB	Successful, no translate table defined
	CE.PRM	Service call parameter error
	CE.DIR	RSX Directive error
		Second status word contains the DSW
CE.RMS	File I/O error	
	RMS error code passed in second word	
ccb	An integer which contains the address of the call control block.	
ldb	Line descriptor block.	
ttble	This optional parameter contains a translate table for converting internal telephone numbers to that accepted by an autodial modem. Its primary use is for auto-dial modems attached to the communication port and editing out format effectors in the telephone number.	

11.2.4.8 Dial Call (CCDIAL) - This routine dials a call on the designated communications line. Communication line types are designated with the CCSMC communications service, and detected by the current 'data mode' of the device.

Call:

CALL CCDIAL (status,ccb,tel)

Parameters:

status	CS.SUC	Successful
	CE.PRM	Service call parameter error
	CE.DIR	RSX Directive error
		Second status word contains the DSW
	CE.IER	I/O Termination error
		Second status word contains the
		IOSB first word
ccb		An integer which contains the address of the call control block.
tel		Telephone number.



11.2.4.9 Answer Call (CCANS) - This routine answers a call on the specified telephone line. The call is answered in the current mode of the line, either voice or data.

For a TMS line, for which the call has already been established in VOICE mode, when a change mode (CCMODE) command is issued to place the line in ASYNC mode the Answer routine causes TMS to connect the designated modem in answer mode. The corresponding party must have established their modem in 'originate' mode. To establish originate mode with P/OS Communication services, use the CCORG routine.

Call:

CALL CCANS (status,ccb)

Parameters:

status	CS.SUC	Successful
	CE.PRM	Service call parameter error
	CE.DIR	RSX Directive error Second status word contains the DSW
	CE.IER	I/O Termination error Second status word contains the IOSB first word
ccb	A integer which contains the address of the call control block.	

11.2.4.10 Hangup A Call (CCHNG) - This routine disconnects a call which has previously been established on a specified line.

Call:

CALL CCHNG (status,ccb)

Parameters:

status	CS.SUC	Successful
	CE.PRM	Service call parameter error
	CE.DIR	RSX Directive error
		Second status word contains the DSW
CE.IER	I/O Termination error	
		Second status word contains the IOSB first word
ccb	An integer which contains the address of the call control block.	

11.2.4.11 Transmit Data (CCTXD) - This routine transmits a data buffer to the destination system at the other end of a communications line. The data is delivered to the remote system as a character string with no intermediate line protocol.

Call:

CALL CCTXD (status,ccb,[efn],stadd,size)

Parameters:

status	CS.SUC	Successful
	CE.PRM	Service call parameter error
	CE.DIR	RSX Directive error
		Second status word contains the DSW
	CE.IER	I/O Termination error
		Second status word contains the
		IOSB first word
ccb		An integer which contains the address of the call control block.
efn		An event flag to be set on completion of the I/O transfer. This is only required if asynchronous I/O is being used. For synchronous I/O the parameter may be absent or have a value of zero.
stadd		A string that contains the data to be transferred.
size		An integer specifying the amount of data to be transferred.

11.2.4.12 Receive Data (CCRXD) - This routine reads data from a communications line. The user may either read all the data currently buffered in the driver or wait a specified period of time and read all the data accumulated up to that point. Alternatively, the user may read a fixed amount of data from the line. Input will always terminate when the user's buffer becomes full. The driver has a limited amount of buffer space (512 bytes). If XON/XOFF support has been specified, the driver will transmit an XOFF when the buffer becomes three-quarters full and an XON when the buffer is emptied to the one-quarter point. If the buffer becomes full, additional characters will be lost.

Call:

CALL CCRXD (status,ccb,[efn],stadd,size,[tmo])

Parameters:

status	CS.SUC	Successful
	CE.PRM	Service call parameter error
	CE.DIR	RSX Directive error
		Second status word contains the DSW
	CE.IER	I/O Termination error
		Second status word contains the
		IOSB first word
ccb		An integer which contains the address of the call control block.
efn		An event flag to be set on completion of the I/O transfer. This is only required if asynchronous I/O is being used. For synchronous I/O the parameter may be absent or have a value of zero.
stadd		A string into which the data may be placed.
size		An integer specifying the amount of data received.
tmo		An optional timeout count in one-second intervals. If zero is specified, no timeout occurs and the read request does not terminate until the buffer is full. If tmo is less than zero, the request returns immediately after transferring as many characters as are available. If tmo is greater than zero the request will timeout after tmo seconds (less than or equal to 127 secs). Again the number of characters available is returned.

11.2.4.13 Flush Input Buffer (CCFLSH) - This routine will flush the input buffer for the specified communications line.

Call:

CALL CCFLSH (status,ccb)

Parameters:

status	CS.SUC	Successful
	CE.PRM	Service call parameter error
	CE.DIR	RSX Directive error
		Second status word contains the DSW
	CE.IER	I/O Termination error
		Second status word contains the
		IOSB first word
ccb		An integer which contains the address of the call control block.

11.2.4.14 Generate Break (CCBRK) - This routine generates a break or a long space on the line.

Call:

CALL CCBRK (status,ccb,brk)

Parameters:

status	CS.SUC	Successful
	CE.PRM	Service call parameter error
	CE.DIR	RSX Directive error
		Second status word contains the DSW
CE.IER	I/O Termination error	
		Second status word contains the IOSB first word
ccb	An integer which contains the address of the call control block.	
brk	If this is zero then a break is sent, if this has the value one then a long space is sent.	

11.2.4.15 Kill Transfer (CCKILL) - This routine destroys any outstanding transfers. This can only be used with asynchronous I/O transfers. The outstanding transfer terminates in the normal manner by setting the event flag specified by the user at initiation.

Call:

CALL CCKILL (status,ccb)

Parameters:

status	CS.SUC	Successful
	CE.PRM	Service call parameter error
	CE.DIR	RSX Directive error
	CE.IER	I/O Termination error

Second status word contains the DSW

Second status word contains the IOSB first word

ccb      An integer which contains the address of the call control block.

### 11.2.5 File Transfer Services

The routines in this section are used to initiate and control file transfer between two Professionals. The actual file transfer is performed by a file transfer task running in the background. The file transfer task handles the link protocol and the file transfer protocol. The file transfer task ensures the correct delivery of data between the source and destination systems.

**11.2.5.1 Setup File Transfer Options (FTOPTG And FTOPTP) - Two routines are provided which allow the file transfer options to be configured for automatic file transfer initiated by a remote system. The routine FTOPTG returns the current option settings. The routine FTOPTP modifies the current option settings.**

Call:

CALL FTOPTG (status,ccb,flags,pswrd,vol,direc,maxfile)

and

CALL FTOPTP (status,ccb,flags,pswrd,vol,direc,maxfile)

Parameters:

status	CS.SUC	Successful
	CE.PRM	Service call parameter error
	CE.RMS	File I/O error
		RMS error code passed in second word
ccb	An integer which contains the address of the call control block.	
flags	Option Flags	
	bit 0	= 1 Enable remote file copy 0 Disable remote file copy
	bit 1	= 1 Enable remote file receive 0 Disable remote file copy
	bit 3	= 1 Superceed existing files 0 Do not superceed existing files
	bits 4 to 15 Reserved	
pswrd	Nine character password to be used by a remote Professional. Space filled if password security disabled.	
vol	Nine character name of the volume to which files being received from another Professional are placed. Only used if no volume specification appears on the	



incomming file spec.

**direc** Nine character name of the directory into which files being received from another Professional are placed. Only used if no directory specification appears on the incomming file spec.

**max\_file** An unsigned integer specifying the max size local file that can be created by the remote Professional. Note a zero value indicates no limit. The file size is in blocks.

**Issues:**

1. Status: To be decided.

11.2.5.2 Send A File (FTSND) - This routine is used to send a file to another Professional. The file transfer is initiated in the background. A message is written to the message board when the transfer terminates.

Call:

CALL FTSND (status,ccb,fspec1,[fspec2])

Parameters:

status	CS.SUC	Successful
	CE.PRM	Service call parameter error
	CE.UNO	Unsupported message option
	CE.MFE	Message format error
	CE.IMU	Invalid message option
	CE.MSC	Message type out of sync
	CE.DDC	DDCMP error
	CE.RMS	File I/O error
		RMS error code passed in second word
	CE.IEA	Internal error abort
ccb		An integer which contains the address of the call control block.
fspec1		The source file descriptor string.
fspec2		The destination file descriptor string, if not supplied this will be the same as the source file.

11.2.5.3 Send A File And Wait (FTSNDW) - This routine is used to send a file to another Professional. The file transfer is initiated in the background and the program waits until the transfer is complete before continuing.

Call:

CALL FTSNDW (status,ccb,fspec1,[fspec2],bfstats)

Parameters:

status	CS.SUC	Successful
	CE.PRM	Service call parameter error
	CE.UNO	Unsupported message option
	CE.MFE	Message format error
	CE.IMO	Invalid message option
	CE.MSC	Message type out of sync
	CE.DDC	DDCMP error
	CE.RMS	File I/O error
		RMS error code passed in second word
CE.IEA	Internal error abort	
ccb	An integer which contains the address of the call control block.	
fspec1	The source file descriptor string.	
fspec2	The destination file descriptor string, if not supplied this is the same as the source file.	
bfstats	File transfer statistics buffer, this is a 36 byte array in which the file transfer statistics data is returned.	

11.2.5.4 Retrieve A File (FTGET) - This routine is used to retrieve a file from another Professional. The file transfer is initiated in the background. A message will be written to the message board when the transfer terminates.

Call:

CALL FTGET (status,ccb,fspec1,[fspec2],[pswrd])

Parameters:

status	CS.SUC	Successful
	CE.PRM	Service call parameter error
	CE.UNO	Unsupported message option
	CE.MFE	Message format error
	CE.IMO	Invalid message option
	CE.MSC	Message type out of sync
	CE.DDC	DDCMP error
	CE.RMS	File I/O error
		RMS error code passed in second word
CE.IEA	Internal error abort	
ccb	An integer which contains the address of the call control block.	
fspec1	The source file descriptor string.	
fspec2	The destination file descriptor string, if not supplied this is the same as the source file.	
pswrd	A string containing a password to be presented to the remote Professional system.	

11.2.5.5 Get A File And wait (FTGETW) - This routine is used to retrieve a file from another Professional. The file transfer is initiated in the background and the program waits until the transfer is complete before continuing.

Call:

CALL FTGETW (status,ccb,fspec1,[fspec2],[pswrd],bfstats)

Parameters:

status	CS.SUC	Successful
	CE.PRM	Service call parameter error
	CE.UNO	Unsupported message option
	CE.MFE	Message format error
	CE.IMO	Invalid message option
	CE.MSC	Message type out of sync
	CE.DDC	DDCMP error
	CE.RMS	File I/O error
		RMS error code passed in second word
	CE.IEA	Internal error abort
	CE.PwE	Password error

ccb      An integer which contains the address of the call control block.

fspec1    The source file descriptor string.

fspec2    The destination file descriptor string, if not supplied this is the same as the source file.

pswrd     A string containing a password to be presented to the remote Professional system.

bfstats   File transfer statistics buffer, this is a 36 byte array in which the file transfer statistics data is returned.

11.2.5.6 Get Transfer Status (FTSTS) - This routine returns the status of a file transfer currently in progress on a specified line. Note the status of the transfer being reported may not have been initiated by the current application.

Call:

CALL FTSTS (status,ccb,bfstats)

Parameters:

status	CS.SUC	Successful
	CE.PRM	Service call parameter error
	CE.UNO	Unsupported message option
	CE.MFE	Message format error
	CE.IMO	Invalid message option
	CE.MSC	Message type out of sync
	CE.DDC	DDCMP error
	CE.RMS	File I/O error
		RMS error code passed in second word
	CE.IEA	Internal error abort
ccb	An integer which contains the address of the call control block.	
bfstats	File transfer statistics buffer. This is a 36 byte array in which the file transfer statistics data is returned.	

11.2.5.7 Abort Current Transfer (FTABT) - This routine stops the current file transfer in progress on a given communications line. The transfer is terminated regardless of whether or not it was initiated by this program.

Call:

CALL FTABT (status,ccb,bfstats)

Parameters:

status	CS.SUC	Successful
	CE.PRM	Service call parameter error
	CE.UNO	Unsupported message option
	CE.MFE	Message format error
	CE.I/O	Invalid message option
	CE.MSC	Message type out of sync
	CE.DDC	DDCMP error
	CE.RMS	File I/O error
		RMS error code passed in second word
CE.IEA	Internal error abort	
ccb	An integer which contains the address of the call control block.	
bfstats	File transfer statistics buffer. This is a 36 byte array in which the file transfer statistics data is returned.	

## 11.2.6 TMS Services

The services described in this section are specific to the TMS hardware and may return an error status if attempted for the Kernal Communications Port.

11.2.6.1 Change Mode (CCMODE) - This service changes the current mode of the line.

Call:

CALL CCMODE (status,ccb,mode)

Parameters:

status	CS.SUC	Successful
	CE.PRM	Service call parameter error
	CE.DIR	RSX Directive error
		Second status word contains the DSw
	CE.IER	I/O Termination error
		Second status word contains the
		IOSB first word
ccb	An integer which contains the address of the call control block.	
mode	The following mode settings are available	
	0	= VOICE
	1	= ASYNC (Modem)
	2	= CODEC
	3	= DTMF Keypad (Touchtone pass all keys)
	4	= DTMF Block (Touchtone with # as CR)



11.2.6.2 Originate Call (CCORG) - This routine is the converse of CCANS for a TMS line which is currently in voice mode and about to go into ASYNC mode. When the change mode command is issued, the designated modem is placed in 'originate' mode. Note that one party should be in 'answer' mode and the other party in 'originate' mode.

Call:

CALL CCORG (status,ccb)

Parameters:

status	CS.SUC	Successful
	CE.PRM	Service call parameter error
	CE.DIR	RSX Directive error
		Second status word contains the DSW
	CE.IER	I/O Termination error
		Second status word contains the
		IOSB first word
ccb		An integer which contains the address of the call control block.

11.2.6.3 Auxiliary Keyboard Enable/Disable (CCAUXK) - This routine enables or disables the auxiliary keyboard for the specified TMS line.

Call:

CALL CCAUXK (status,ccb,flag)

Parameters:

status	CS.SUC	Successful
	CE.PRM	Service call parameter error
	CE.DIR	RSX Directive error Second status word contains the DSW
	CE.IER	I/O Termination error Second status word contains the IOSB first word
ccb	An integer which contains the address of the call control block.	
flag	This flag indicates whether the keyboard is being enabled/disabled.  0 - Disabled 1 - Enabled	

11.2.6.4 Turn Speaker On/Off (CCSPKR) - This routine turns the speaker ON or OFF for the specified TMS line.

Call:

CALL CCSPKR (status,ccb,flag)

Parameters:

status	CS.SUC	Successful
	CE.PRM	Service call parameter error
	CE.DIR	RSX Directive error
	CE.IER	I/O Termination error
		Second status word contains the DS w
		Second status word contains the
		IOSB first word
ccb		An integer which contains the address of the call control block.
flag		This flag indicates whether the speaker is being turned ON or OFF.
	0	- OFF
	1	- ON

11.2.6.5 Prepare To Go Voice (CCPTGV) - This routine warns the specified TMS line that it is about to go into VOICE mode. When the change mode takes place the call is not disconnected.

Call:

CALL CCPTGV (status,ccb,flag)

Parameters:

status	CS.SUC	Successful
	CE.PRM	Service call parameter error
	CE.DIR	RSX Directive error
		Second status word contains the DSW
	CE.IER	I/O Termination error
		Second status word contains the
		IOSB first word
ccb	An integer which contains the address of the call control block.	
flag	This flag indicates whether a 'prepare to go voice' command is being issued or a previous one is being cancelled.	
	0	- Cancel
	1	- Issue 'Prepare To Go Voice'

11.2.6.6 Set DTMF Escape Sequence (CCDTMF) - This routine sets a DTMF escape sequence on the specified TMS line. An unsolicited event AST trap is returned when the sequence is encountered. This is reported by setting an event flag if CCATA has been used to attach the line.

Call:

CALL CCDTMF (status,ccb,dtmf)

Parameters:

status	CS.SUC	Successful
	CE.PRM	Service call parameter error
	CE.DIR	RSX Directive error
		Second status word contains the DSW
	CE.IER	I/O Termination error
		Second status word contains the
		IOSB first word
ccb	An integer which contains the address of the call control block.	
dtmf	A string terminated by a zero byte that contains the DTMF escape sequence.	

### 11.3 PROSE, TEXT EDITOR

The P/OS text editor is called PROSE. PROSE offers facilities for entering and editing text to create documents, BASIC programs, memos or similar text files. Ten editing keys on the Professional keyboard allow text manipulation while it is being entered. The end user documentation describes PROSE features and user interface.

The editor can be part of a larger application. For example, an electronic mail application might use the editor to provide editing services for message creation or modification. In this form, PROSE is called the callable editor task. All of the editor functions offered to the end user are available in the callable form of the editor.

#### 11.3.1 Callable Editor Task

To use the callable editor, application tasks invoke it as a separate task by calling the CET subroutine. Parameters to the CET subroutine define the input and output file name, a temporary work file name, format options, maximum line length for text entry, and left and right margin values.

The input file name is the file to be edited. An empty or new file can also be created. After editing, the callable editor opens the output file with the specified name to create the edited version of the file. Your application performs initial and final operations before and after the editor is invoked.

The CET subroutine passes the parameters to the callable editor by spawning it as a task. The subroutine waits until the editor task terminates before returning control to the calling application. When the callable editor task exits, it returns a status word indicating the results of the editing session.

Call:

```
CALL CET (infil,inlen,outfil,outlen,wkfil,wklen,  
format,maxline,initleft,initright,lun,status)
```

Parameters must be specified in the order shown:

Parameters:

infil	The input file name. Specify in the format dev:[dir]filename.typ. The device and directory are supplied by default if you don't specify them.
inlen	the length of the input file name.
outfil	The output file name. Specify in the format dev:[dir]filename.typ. The device and directory are supplied

by default if you don't specify them.  
The output file name may be  
the same as the input file name,  
in which case a new version is created.

outlen

The length of the output file name.

wkfil

The CET temporary work file name.  
The callable editor deletes this file  
when the task exits.

wklen

The length of the CET temporary work  
file name.

format

This parameter determines whether  
escape sequences are retained in the  
output file.

To retain the formatting data,  
use a non-zero value for this parameter.  
To discard the formatting data,  
use a zero value for this parameter.

During the editing session, the user can  
define margins settings for different  
regions of text. The margin settings  
result in the inclusion of nonprinting  
characters, called escape sequences,  
in the file. Although the format of  
the file is unaffected and margins  
are properly set, the escape  
sequences may be excluded from the  
output file.

The presence of escape sequences in  
files may provoke errors if the  
file is used with host system  
utilities.

maxline

The maximum number of characters on  
a line that may be entered by a user.  
Valid values are in the range 2 to 237.

initleft

The initial left margin value.  
This setting is used for text entered  
during the current editing session.

initright

The initial right margin value.  
This setting is used for text entered  
during the current editing session.

lun

The logical unit number required for  
I/O.

status

The parameter to which is returned

the results of the editing session.  
Significant bits are:

Bit 0        If this bit is set on return, it indicates that a fatal processing error occurred during the editing session. The output file is probably corrupt.

Bit 1        If this bit is set on return, it indicates that the output file contains escape sequences indicating some internal formatting information. For example, margin settings are present.

Bit 15       If this bit is set, it indicates that the callable editor task could not be invoked. For example, if the task is not installed the Spawn will fail. Note that the other bits in the status word are not meaningful if this bit is set.

If this bit is clear, it means the callable editor task was invoked.

### 11.3.2 Example Calling Sequences

To be supplied.

#### ISSUES:

1. The parameters to the CET subroutine are passed in accordance with the CTAB Parameter Passing Specification for character strings and integers.

### 11.4 CALLABLE SORT PROGRAM (PRO/SORT)

PRO/Sort is a general-purpose sorting utility which runs on P/OS. Application tasks can use the Spawn directive (SPWNS) to invoke PRO/Sort, passing the name of a file in the MCR command line buffer containing the sort commands to PRO/Sort. Only a few sorting commands

are needed to provide a wide range of sort processing. These commands are described in the following sections.

#### ISSUES:

1. INCLUDE EXAMPLES.RNO TO ILLUSTRATE PRO/Sort

#### 11.4.1 PRO/Sort Commands

PRO/Sort commands are put in a PRO/Sort Command file. During program execution, the command file is passed to the sort program and the sort is performed.

PRO/Sort supports the following commands:

!Comment statement

;Comment statement

@ <indirect command filespec>

COLLATE <val> AS <val> { , <val> AS <val> }

DEFAULT

FIELD <name> <datatype> <start> <end>

FORCE <name> TO <val> ( IF <val> )

[ EQ | = ]

[ NE | <> ]

INCLUDE <name> [ LT | < ] <name>| "constant" {,<another condition>}

[ LE | <= ]

[ GT | > ]

[ GE | >= ]

INPUT <filespec>

OUTPUT <filespec>

SORT +/- <name> { , +/- <name> }

WRITE <name> { , <name> }

#### Definitions:

<val> Can be a number or a character.  
Characters must be enclosed  
in double quotes or apostrophes.

<start>, <end> Must be a number.



<name>                    Must be a character string.

#### 11.4.2 Comment Statement

Comment statements in a command file are indicated by an exclamation point (!) or semi-colon (;) as their first character. Lines beginning with these symbols are ignored by PRO/Sort.

#### 11.4.3 Indirect Command File Command

A PRO/Sort indirect command file is executed when an at sign (@) precedes the indirect command file name. The file must have a .CMD file type. Indirect command files may be nested.

Files may be nested 10 levels deep.

#### 11.4.4 COLLATE Command

The COLLATE command may be used to change the logical ordering of a character set. The purpose of this command is to enable country-specific collating sequences. See also the Default command.

Format:

COLLATE AS

COLLATE BEFORE

COLLATE AFTER

#### 11.4.5 DEFAULT Command

The DEFAULT command may be used to set up the default collating sequence for a variety of national languages. The DEFAULT command supports the following national languages: Basque, French, Norwegian, and English. If a COLLATE command is used prior to a DEFAULT command, the collating sequence specified in the COLLATE command is superseded by the DEFAULT command.

ISSUES:

1. Which national languages are supported?

#### 11.4.6 FIELD Command

The FIELD command allows you to identify multiple fields of an input record.

With the FIELD command, a symbolic name may be given to a positional range of a record. For example the "ZIP" field could be assigned to columns five through nine in each record.

The only valid data type for the FIELD command is CHAR. A maximum of 30 fields may be defined. When subsequent references are made, they reference field names only. A field must be defined before it is referenced, for example with the INCLUDE statement.

#### ISSUES:

1. Does an upper limit restrict the number of fields which may be defined?

#### 11.4.7 FORCE Command

The FORCE command allows you to change how one field of a record is ordered. This differs from the COLLATE command which affects all fields equally.

For example, the field "ZIP" could be sorted such that all zip codes beginning with 5 are placed at the bottom (or top) of the list. The FORCE command only operates on the first character of the specified field.

#### 11.4.8 INCLUDE Command

The INCLUDE command allows you to sort and write to the output file only those records matching certain conditions.

For example, the INCLUDE command could match all records of employees making less than \$100 dollars per week.

When more than one condition is specified in a single INCLUDE command, the conditions are logically combined (with AND). A record is included only if it met all conditions in the INCLUDE command. When more than one INCLUDE command is used, records need only match one of the lines in an INCLUDE command to be included. Up to 30 conditions may be specified with INCLUDE commands in a file.

#### 11.4.9 INPUT Command

The INPUT command specifies the input file containing the data to be sorted.

#### 11.4.10 OUTPUT Command

The OUTPUT command specifies the file that the sorted records are to be written to.

#### 11.4.11 SORT Command

The SORT command specifies what fields are pertinent to the sort process. Data fields (as opposed to key fields) are not the target of the sort process and are not specified in the SORT command. The SORT command also specifies the order of significance of fields to be sorted, that is, primary key, secondary key, etc.

More than one SORT command is permitted or the successive key fields may be combined on one line. Up to 16 fields may be specified with SORT commands in a file.

A leading plus (+) or minus (-) sign may be specified before the field name to control the order of the sort for this field.

A leading plus sign is the default if no sign is specified. It indicates that the sort should be performed in ascending order.

A leading minus (-) sign indicates that the sort should be performed in descending order.

#### 11.4.12 WRITE Command

The WRITE command creates output records with a different arrangement of fields from the input records. Using the WRITE command, a field in the input records could be omitted from the output records, for example, or all the fields could be retained in a new order.

#### 11.4.13 Comparison Of PRO/Sort And PDP-11 SORT-11

If you are familiar with PDP-11 SORT-11, you might wish to compare it with PRO/Sort to gain a better understanding of PRO/Sort functionality.

This section illustrates the differences between the command syntax for SORT-11 and PRO/Sort. For each PRO/Sort command, the corresponding SORT-11 specification line is described. This shows how

PRO/Sort can be used to perform familiar SORT-11 operations.

11.4.13.1 Comment Statement - In SORT-11, a comment in a specification file is indicated by an asterisk character (\*) in column seven. Comment lines in SORT-11 and PRO/Sort follow as examples:

SORT-11:

\*This is a line with a comment in it.

PRO/Sort:

!This is a line with a comment in it.

;This is an alternate form of a comment line.

11.4.13.2 COLLATE Command Compared with ALTSEQ Records - SORT-11 allows alternate sequencing of records in a specification file through the ALTSEQ (ALternate SEquence) command. The COLLATE command provides a superset of the same function in PRO/Sort.

For example, the format of an ALTSEQ record in SORT-11 is:

"ALTSEQ aaabobxxxxyy..."

The value aaa is the octal value of a character. The value bbb is the value of the character to use in the sort.

The value xxx is a character. The value yyy is the value of the character to use in the sort, etc.

The following examples show the format for both SORT-11 and PRO/Sort for sorting at signs (@) as ASCII zero (0) characters:

SORT-11:

ALTSEQ 100060

PRO/Sort:

COLLATE "@" AS "0"

or

PRO/Sort:

COLLATE 100 as 60

11.4.13.3 FORCE Command Compared with F Command - The PRO/Sort FORCE command is functionally based on the SORT-11 "F" field record specification, in which the user specifies the column to be forced, the character to force that column to, and optionally a 'trigger' character. The trigger character specifies that the force should not occur unless the indicated column contains the trigger character. For example, all "X" characters in column 50 could be sorted as if they

were "Y"s. The trigger character in this case is "X".

The SORT-11 "F" field specifier can also logically combine conditions through a FORCE command. For example, if column 50 contains an "X" then force it to a "Y". If it does not contain an "X" then force it to a "Z". PRO/Sort does not do this.

Examples:

SORT-11:

F 50XY

PRO/Sort:

FIELD NAME CHAR 50 59

FORCE NAME TO "Y" IF "X"

11.4.13.4 INCLUDE Command Compared with O And I Specifications -  
SORT-11 allows you omit or include portions of the input file to be sorted. PRO/Sort provides one way to achieve either result through an INCLUDE command.

Both SORT-11 and PRO/Sort use the same logical relationship tests to determine if a given record will actually be omitted or included from the output file.

For example, to omit all records with column 10 to 14 equal to "XYZZY":

SORT-11:

O C 10 14EGXYZZY

PRO/Sort:

FIELD PLUGH CHAR 10 14  
INCLUDE PLUGH NE "XYZZY"

or

PRO/Sort:

INCLUDE PLUGH <> "XYZZY"

11.4.13.5 INPUT, OUTPUT Command Compared with Command Line Format -  
SORT-11 uses the standard PDP-11 command line format "input=output" to specify the input and output files. In contrast to this, PRO/Sort has the user specify the input and output files in separate free form commands.

For example:

SORT-11:

SRT NEW.DAT = OLD.DAT

PRO/Sort:  
INPUT OLD.DAT  
OUTPUT NEW.DAT

11.4.13.6 SORT Command Compared With N And O Field Specifications - Both SORT-11 and PRO/Sort permit the same kind of key structures (primary, secondary, etc.). SORT-11 is limited to a maximum of 10 levels of keys. PRO/Sort is limited to 16 levels.

PRO/Sort uses the plus (+) and minus (-) signs to indicate ascending or descending sort per key. SORT-11 uses N ("normal") and O ("opposite") in column 7 of the specification file. The following example sorts the primary key from columns 1 to 5, secondary descending from 34 to 37:

SORT-11:  
FNC 1 5  
FOC 34 37

PRO/Sort:  
FIELD PRIMARY CHAR 1 5  
FIELD SECONDARY CHAR 34 37  
SORT +PRIMARY, -SECONDARY

#### ISSUES:

1. Interpret this example.

11.4.13.7 WRITE Command Compared With D Field Specifications - Both sort programs permit creating output files whose records differ in field order from the input file. For example, if the input consists of a name in columns 1 to 19, address in 20 to 39, and zip in 40 to 44, the output records could be rearranged in the order <zip>,<address>,<name>:

SORT-11:  
FDC 1 19  
FDC 20 39  
FDC 40 44

PRO/Sort:  
FIELD NAME CHAR 1 19  
FIELD ADDRESS CHAR 20 39  
FIELD ZIP CHAR 40 44  
WRITE ZIP, ADDRESS, NAME

## CHAPTER 12

### FRAME DEVELOPMENT TOOL (FDT)

The Frame Development Tool (FDT) in the Tool Kit is a special-purpose utility for creating interactive displays for your P/OS application. With the Frame Development Tool, you can create:

- o Single-choice menus through which the end user interacts with your application
- o Help menus that list subjects for which on-line help is available
- o On-line help text that is available whenever the end user presses the HELP key
- o A file of messages

An application may have many menus that allow interaction between it and the end user. In addition, you may write on-line instructions to help the user perform complex operations and to display error messages if the user does something incorrectly. You may also want to post messages to confirm completion of operations. Each menu, display of Help text, and message is displayed separately, one at a time. The term frame denotes any single-screen display.

Frames are stored on disk with other frames of the same type in a definition file. Thus, your application menus will all be stored together in a Menu Definition File; the help frames will be in a Help Definition File; messages will be in a Message Definition File.

You can create frames on the host with the Frame Development Tool either by using a VT100 or the Professional in terminal emulation mode. If you want to have 8-bit characters in your frames, you must use the Professional in terminal emulation mode.

FDT error messages are listed at the end of this chapter.

## 12.1 PLANNING THE FRAMES

Before you start the Frame Development Tool, determine what type of frames you want to create for your application. For example, you can create single-choice menus for full or short-form, Help text frames, and message text frames.

### 12.1.1 Menus

There are three types of menus:

1. Single-choice: The end user chooses one item from a list of up to 12 items. Single-choice menus allow end user to perform application-related activities. Single-choice menus can be full or short-form.
2. Help menus: Help menus list subjects about which help text frames are available.
3. Multiple-choice: The end user chooses one or more items from a list. The list may be of any length, possibly covering multiple screens. Multiple-choice menus are not created with the Frame Development Tool. Multiple-choice menus can be created only under program control during execution using menu service routines described in Chapter 9.

Each application you write may have a file of menus which can be used to interact with the application. You specify the exact wording and order of items on a menu. You can either use menu service routines to display the menus and frames or you can display them directly from your application. If you use menu services to display them, P/OS conventions for visual layout and function key usage apply.

Every menu has the following fields:

- o A menu title.
- o A few lines of introductory or explanatory text.
- o A list of options -- the command interface to your application. The maximum number of options is 12. At your direction, some number of characters in each option are in boldface, indicating the ones the user must type to make an unambiguous choice. The bolded characters are called the keyword for the option.
- o A pointer which points to one or more options or (when not in use) is at the rest position just above the options.



- o A prompt line requesting user response.
- o Two cursor lines on which user responses are typed.
- o An error and information line where messages may be displayed.
- o A flag indicating the presence of Additional Options, accessed by pressing the ADDTNL OPTIONS key.
- o An indication of unread messages on the Message/Status Board

Single-choice menus may be defined either in a Menu Definition File created with the Frame Development Tool or dynamically (during program execution) using menu service routines. Chapter 13 explains how to use menu service routines to create dynamic single-choice menus.

Figure ? is an example of a single-choice menu. The fields listed above are pointed out in this figure.

Figure ? : A Single-choice Menu

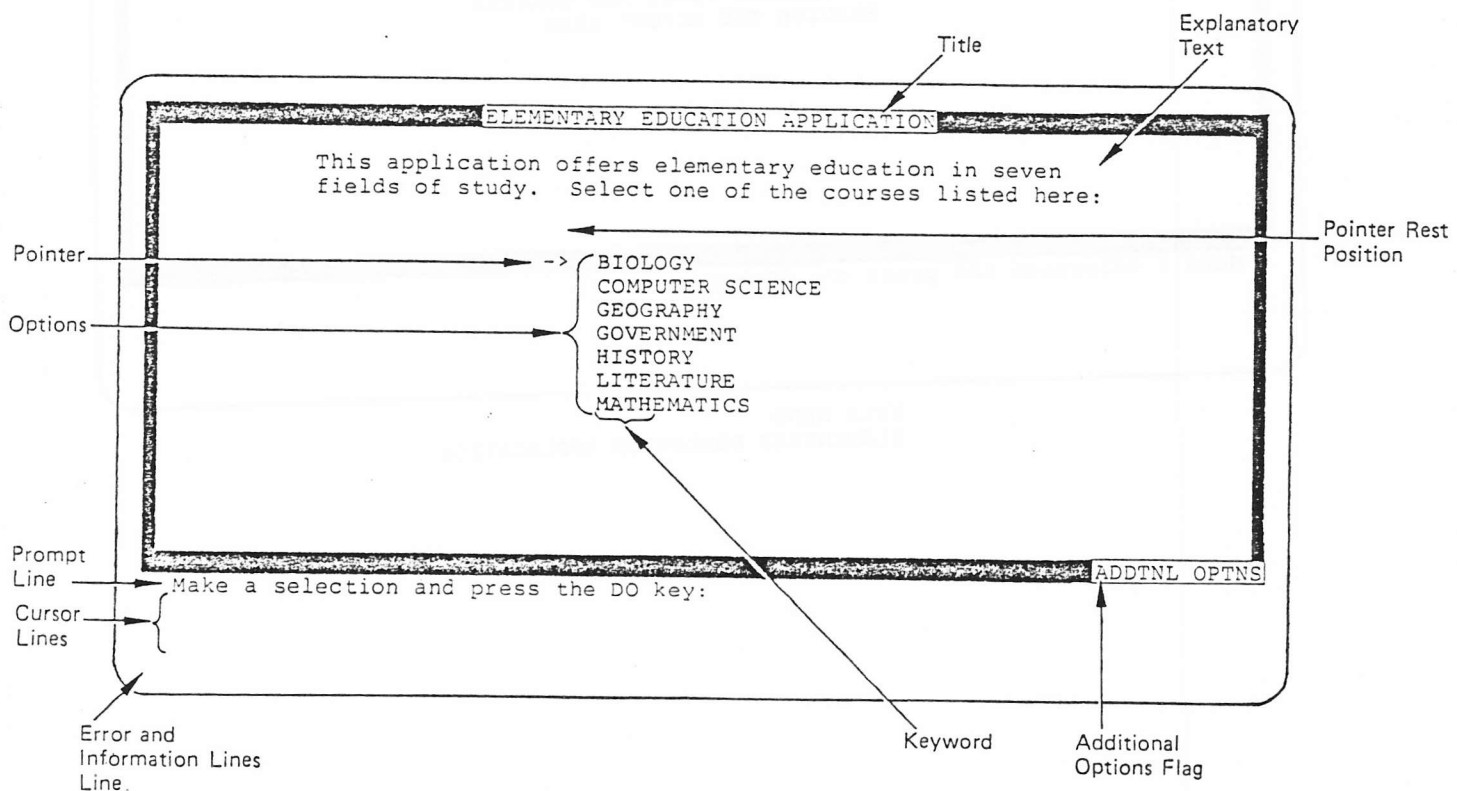
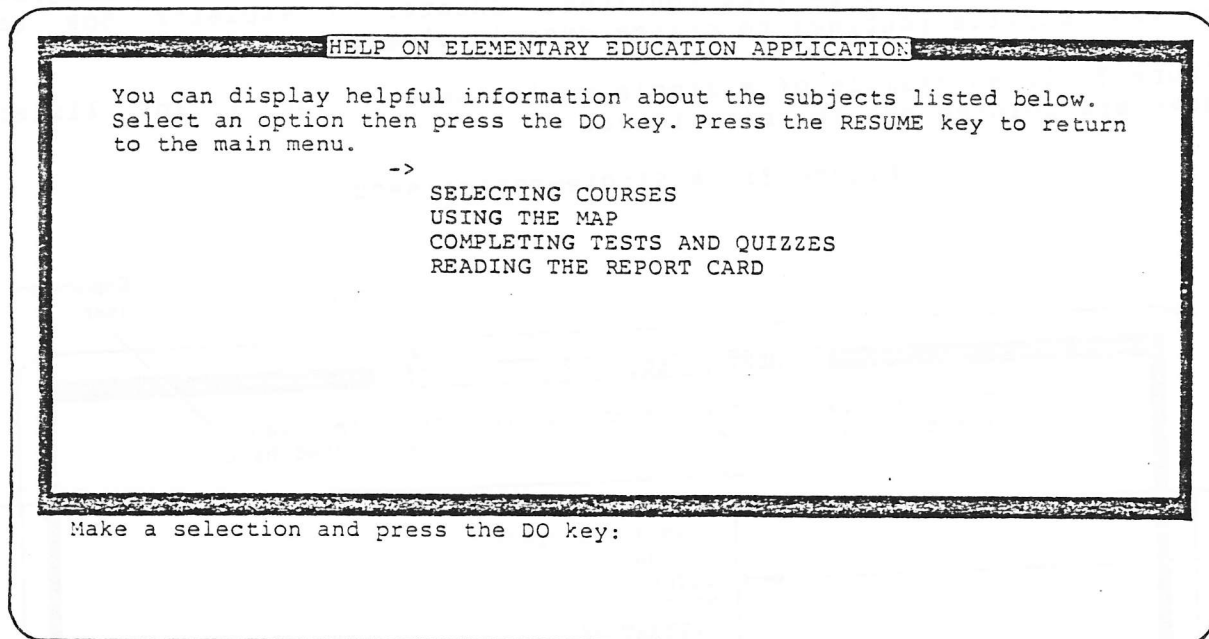


Figure ? shows the Help menu that was displayed when the end user pressed the HELP key. When the HELP key was pressed, the pointer was positioned at the rest position on the single-choice menu shown in Figure ?.

Figure ? : A Help Menu



HELP MENU  
ELEMENTARY EDUCATION APPLICATION

Multiple-choice menus can be created only under program control during execution using menu service routines described in Chapter 7. Figure 7-1 is an example of a multiple-choice menu.

Figure 7-1: A Multiple-choice Menu

COURSE: GEOGRAPHY OF NORTH AMERICA

This course covers rivers, lakes, and mountains. Our first topic is mountains. The mountains listed below are located in different parts of the world. Select all the mountains located in North America:

->

- MT. RUSHMORE
- MT. WASHINGTON
- MT. HOOD
- GRAND TETON
- MT. RAINIER
- MT. FUJI
- KILIMANJARO
- MT. ST. HELENS
- MT. MCKINLEY

Choose one or more options with the SELECT key and press the DO key:

Short-form menus -- abbreviated single-choice menus -- are derived from full single-choice menu text. However, short-form menus cover four lines, rather than 24.

To display a single-choice menu in short form, use the menu service routine described in Chapter ?. Figure ? shows the short-form of the single-choice menu shown in Figure ?

An short-form menu has the following format:

- o A menu title
- o A two-line list of options in up to six columns, only the bolded characters are displayed. Up to ten characters for each option are displayed.
- o A prompt and echo line with three fields for a prompt, display of typed characters, and a message
- o A flag indicating the presence of Additional Options, accessed by pressing the ADDTNL OPTIONS key

Figure ? : A Short-form Menu

COURSES FOR ELEMENTARY EDUCATION APPLICATION

BIO	COMP	GEO	GOV
HIST	LIT	MATH	

SELECTION:

The end user selects options from menus and corrects typing errors with function keys.

Pressing the delete key erases typed characters.

Pressing the CANCEL key restarts the selection process. Any characters typed thus far on the response line are cleared, and the pointer returns to the rest position.

To make a selection, the user positions the pointer with the arrow keys so it points to an option. The user presses the DO key to indicate that a selection has been made. Any other function key terminates the selection process with no selection made.

Any user error results in a beep or a short explanatory message.

The user selects options in the full single-choice menu by positioning the pointer or by typing the option keyword on the response line. The pointer moves only when the characters identify a unique choice. The pointer is one line high and never covers multiple choices. When the user presses the DO key, the selection is completed.

In the multiple-choice menu, the user selects options by positioning the pointer or by typing the option keyword on the response line and then pressing the SELECT key. The options may reside on the current screen or on related screens not currently being displayed. (The pointer does not move in response to typed characters since the matching options may not currently be on the screen.) When options are chosen with the SELECT key, multiple pointers point to all the selections. If necessary, the display on the screen is scrolled to select the option. Pressing the SELECT key for a previously selected option removes that option from the group of selected options.

In the short-form menu, no pointer is used. The user makes a selection by typing characters on the response line. This type of menu enables application writers to provide a command interface similar to a command language for an application.

#### ISSUES:

1. what is the flag for additional options?
2. what is the flag for message/status board unread messages?

### 12.1.2 Help And Message Text Frames

Help text frames provide on-line instruction while the end user is using your application. Your on-line instruction may be up to 16 lines of text. Help text frames can be displayed on the screen as follows:

- o Full covering up to 16 lines
- o Top half covering up to 8 lines
- o Bottom half covering up to 8 lines
- o Last line (on the error and information message line)

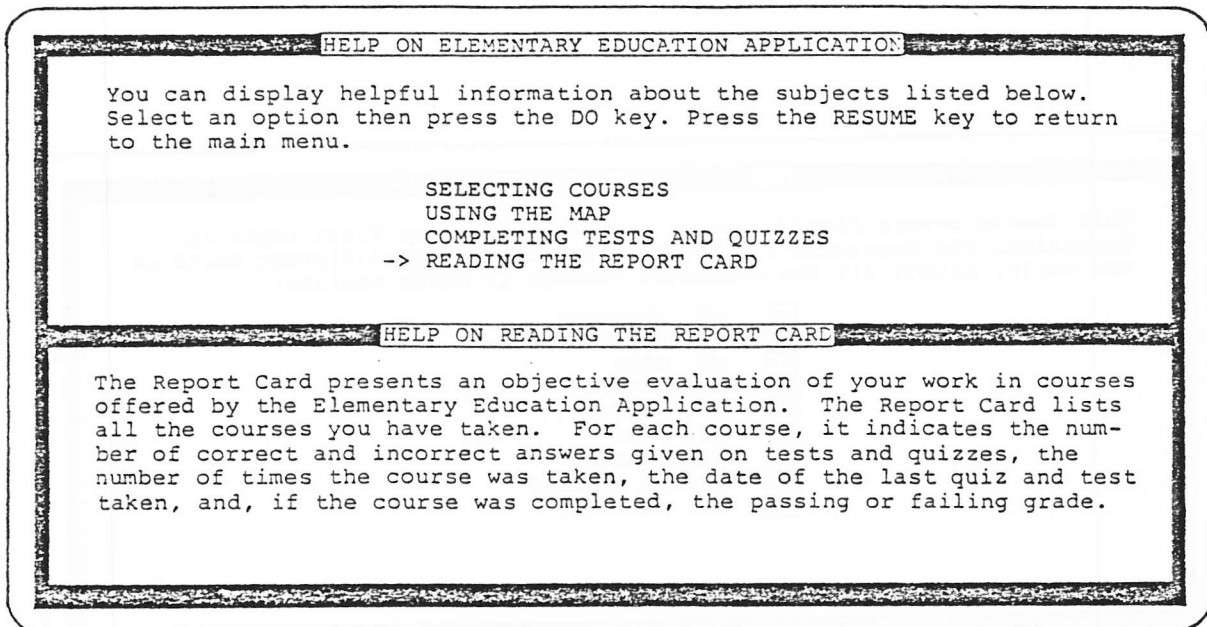
Frames displayed on half the screen do not alter the current contents of the screen.

When you create a single-choice menu, you can associate a help frame or help menu with it. When the single-choice menu is displayed and the end user presses the HELP key, the Help text frame or Help menu which are associated with the menu are displayed. When the end user presses the RESUME key, the menu system refreshes the screen.

If the application does not have menus, Help text and Help frames may be displayed directly using Help Service Routines. In this case, the application must restore the screen after the end user presses the RESUME key.

Figure ? shows the Help text frame displayed on the bottom half of the screen when the user pressed the HELP key. When the HELP key was pressed, the pointer was positioned on the option "Reading the Report Card" in Figure ?

Figure 2: A Help Text Frame



Message text frames can contain up to 20 lines of text. The Frame Development Tool provides an orderly way of storing messages; however, your application must retrieve and display a message frame. Figure ? shows the message text frame displayed when the end user selected Kilimanjaro from the multiple-choice menu in Figure ?

Figure ??: A Message Text Frame

COURSE: GEOGRAPHY OF NORTH AMERICA

This course covers rivers, lakes, and mountains. Our first topic is mountains. The mountains listed below are located in different parts of the world. Select all the mountains located in North America:

<input type="checkbox"/>	MT. RUSHMORE
<input type="checkbox"/>	MT. WASHINGTON
<input type="checkbox"/>	MT. HOOD
<input type="checkbox"/>	GRAND TETON
<input type="checkbox"/>	MT. RAINIER
<input type="checkbox"/>	MT. FUJI
<input type="checkbox"/>	KILIMANJARO
<input type="checkbox"/>	MT. ST. HELENS
<input type="checkbox"/>	MT. MCKINLEY

Choose one or more options with the SELECT key and press the DO key:  
Kilimanjaro is not in North America. It is in Tanzania, Africa.  
At 19,340 feet, it is the highest point in Africa.



## 12.2 DEVELOPING THE FRAMES

After you have planned the frames, you can start the Frame Development Tool and create them. The following sections give an overview of the FDT command procedure and illustrates the procedure with a sample terminal session. Subsequent sections provide reference material on FDT commands.

### 12.2.1 Overview

To develop frames, follow these steps:

1. Start the Frame Development Tool (FDT) and enter a file name for the frame definition file. The definition file is opened and it becomes the current frame definition file.
2. FDT displays a prompt requesting the type of frames you want to create. You can respond with an H for Help, an M for message, or an S for single-choice. After assigning a file type, you can create frames of that type and store them in it. For example, if the file is assigned single-choice menus, you can create as many menus as you need. You can then close the file and open another file called HELP for help frames.

You can store frames of one type only in the current definition file. For example, if you specify menus as the frame type, only menus can be placed in the current frame definition file. To create help frames after creating menus, save the current definition file and open a new one.

3. FDT displays a prompt requesting what you want to do with a frame. You can add, delete, modify, or convert a frame for execution with the application. You can also get reports about frames in the current definition file. The first step is to create a frame.

Frames are created in an interactive session using forms. To create a new frame, fill out a Profile, Display, and, for menus only, one Action form for each option on the menu to define the frame and its operation.

1. A Profile form contains operational information about the frame. Information recorded on the Profile form provides direction to menu services and to your application about the purpose and operation of the frame when it is displayed. This information is not displayed on the frame.
2. A Display form contains the text to be displayed on the screen for this frame. For example, user instruction and a list of options are recorded on a Display form. When this frame is needed by the application, the information

recorded on the Display form is displayed on the screen.

3. An Action form specifies the actions to be taken for an option if the end user selects that option. Menus require one Action form for each option because for each option on the menu, a related action may be taken. For example, one might be "Create a File". When the user selects this, the application creates the file. The action form indicates to the executing application what action to take based upon the option selected.
4. Save the frame in the current frame definition file. When you save the frame, you save the two to fourteen forms that you filled out. For example, if you created a single-choice menu then you are saving a Profile and Display form, and one Action form for each option on the menu. The forms define the display and the context in which it is used.
5. Use the ADD command to create remaining frames for this definition file.
6. Use the REPORT command to make that sure the frames are correct; use the MODIFY command to alter them, if necessary. The REPORT command compiles information about frames you have created. You can use the report to check the frames before you use them with your application. If you find an error in a frame, modify the frame by changing the appropriate form. You may have to change the Profile, the Display, or one of the Action forms to correct the error.
7. After creating all required frames, use the CONVERT command to format your definition files. After formatting, a definition file is ready to install and run with your application on P/OS.

### 12.2.2 Sample Terminal Session

The following is a sample of a terminal session using the Frame Development Tool on a terminal set to DCL.

\$ RUN FDT

FDT starts.

FDT prompts for a file name and, in this example, a new file is specified.

Filename: MENUS

FDT prompts for the file type.

Create (H)elp, (M)essage, or (S)ingle-choice menu file? S

The letter "S" was entered. This definition file will contain single-choice menus.

Now FDT prompts for a command.

File Command: ADD FRAME1

The Add command is entered to create a new frame called FRAME1.

Since no information is known about the frame, FDT displays a Profile form and the user fills it in. To signal that the Profile form is complete, the user presses the EXIT CURRENT FORM key on the FDT keypad.

FDT prompts for another command to work with FRAME1.

Frame Command: DISPLAY

The Display command is entered to create a display for the new frame FRAME1.

FDT displays a facsimile of a menu. Using the FDT editing keypad, the user fills in the Display form with the text for this frame.

The user presses the EXIT CURRENT FORM key on the editor keypad and control returns to frame editing command level.

FDT prompts for another command to work with FRAME1.

Frame Command: ACTION

The Action command is entered to assign actions for each option on the display form for FRAME1.

A series of Action forms, one for each option on the Display form, are displayed. The EXIT CURRENT FORM key is used to proceed from one Action form to the next.

When actions have been assigned to all options, control returns to frame editing command level.

Frame Command: ACTION 3

The Action command specifies the action form for option 3. The user wants to correct a mistake on the form.

The action form for option 3 is displayed. When the user exits from the form, control returns to frame editing command level.

Frame Command: SAVE

FRAME1 is saved on disk and a message is displayed confirming it.

File Command: SAVE

The current frame definition file, MENUS, is saved with the new frame, FRAME1, in it.

Control returns to host system command level. The user decides to modify the Profile form for FRAME1.

\$ RUN FDT

Filename: MENUS

File Command: MODIFY FRAME1

FDT obtains FRAME1 and displays a message confirming that FRAME1 is available for modification. The user must specify whether he wants to work on the Profile, Display, or Action forms for FRAME1.

Frame Command: PROFILE

The Profile command is entered. The existing Profile form is displayed, and changes are made to it.

Frame Command: SAVE

FRAME1 is saved with the new Profile form.

File Command: QUIT

The user decided not to save the new FRAME1. The changes are not saved. Control returns to host system command level.

### 12.2.3 Frame Development Tool Commands

Frame Development Tool commands are presented in Sections ? and ? If FDT has been installed on your system, to start type:

`$FDT filename`

If FDT has not been installed on your system, to start type:

`$RUN FDT`

If you don't specify the file name or if you use the Run command to start FDT, the Frame Development Tool prompts for the name of the frame definition file to open. The opened file is referred to as the current frame definition file. The default file type is .DAT.

For new files, the Frame Development Tool then prompts for the type of frame definition file you want to create. Valid file types are help, message, and single-choice menu file. Help frame definition files may contain help menus or help text frames.

Create (H)elp, (M)essage, or (S)ingle-choice menu file?

Section ? lists commands used to select a particular file and frame after you start the Frame Development Tool. These commands are typed to the Frame Development Tool prompt File Command:.

Section ? lists commands that are used to work on forms that define frame contents and frame operation. These commands are typed to the Frame Development Tool prompt Frame Command:.

The following general-purpose commands can be typed in response to either prompt.

#### Exit

The EXIT command saves the current work, and returns control to the previous command level.

When typed to the File Command prompt, the EXIT command saves the current frame definition file on disk; it performs the same function as the SAVE command. After typing EXIT, control returns to host system command level.

When typed to the Frame Command prompt, the EXIT command saves the current frame on disk in the current file. Only the current frame, not the file, is saved. If you do not save the current file when you leave the program, none of the changes made in the session will be saved. After typing EXIT, the frame is saved, and control returns to file editing command level.

After saving a frame or file, FDT displays a confirmation message naming the frame or file saved.

## Help

The HELP command displays a list of Frame Development Tool commands.

If you type HELP to the Frame Command prompt while you are working on a text frame, the ACTION command will not be listed.

## QUIT

The QUIT command discards the current work and returns control to the previous command level.

When typed to the File Command prompt, the QUIT command returns control to host system command level without saving the current file. Any changes made to any frame in the file are discarded.

When typed to the Frame Command prompt, the QUIT command returns control to file editing command level without saving the current frame. Any changes made to the frame are discarded.

## Save

The SAVE command preserves the current work and returns control to the previous command level.

When typed to the File Command prompt, the SAVE command saves the current file on disk. If the file is new, it is assigned the next higher version number. If the file already existed, it is assigned the next higher version number.

When typed to the Frame Command prompt, the SAVE command saves the current frame on disk in the current file. Only the current frame, not the file, is saved. If you do not save the file when you stop the Frame Development Tool, all the changes made in this session will be discarded. After you type SAVE, control returns to file editing command level.

After saving a frame or file, FDT displays a confirmation message naming the frame or file saved.

In the following sections, each command is introduced in a command line in boldface type. The command line shows the command name as it must be typed and the correct placement of parameters. The following documentation conventions are used to define syntax for Frame Development Tool commands and parameters:

- UPPERCASE** In commands, must be typed; the minimum abbreviation of a command.
- lowercase characters** In commands and command options, not required.
- In command parameters, a variable for which you must supply a value.
- If required parameters are not specified, the Frame Development Tool displays a prompt requesting the required parameter.
- []** Enclose optional parameters -- include or omit the item in brackets.
- |** Separates multiple parameters. Use only one from a series of parameters separated by vertical bars.

12.2.3.1 File Editing - File editing commands allow you to select the frame definition file and frame that you want to create or work with, to save the file and frame, and to open another frame definition file. File editing commands can be typed in response to the prompt:

File Command:

When you are working at this command level, the current file type and file name are displayed in a message at the top of your screen.

Add frame-identifier

The ADD command allows you to create a new frame, which can be added to a new or existing frame definition file. When you type ADD, the Frame Development Tool displays a Profile form. The Profile form is the first form you must fill out when you are creating a new frame.

Parameters:

frame-identifier	The 8-character identifier of the new frame.
------------------	--

If you specify H, for help file, a further prompt is displayed:

Help (M)enu or (T)ext?

Type M for Help menu and T for help text frame. Both types of help frames reside in a help definition file.

The frame-identifier is used in P/OS service routines to display the frame. It is also used to establish relationships between frames. For example, a main menu for an application may list the frame-identifier of a help frame that describes how to select an option from a menu.

If you do not specify the frame-identifier, the Frame Development Tool displays a prompt requesting it.

Convert output-filename

The CONVERT command creates a formatted file from your frame definition file. After using the CONVERT command, you can transfer your frame definition file to the P/OS for application execution. The CONVERT command removes extraneous information, such as comments, from the frame definition file to create the formatted executable file. After formatting, a frame definition file can be read directly by menu services during program execution.

Parameters:

output-filename	The name of the executable frame definition file.
-----------------	---

If you do not specify the file name, the Frame Development Tool



displays a prompt requesting it. The default file types for converted files are as follows:

Help definition file	.HLP
Message definition file	.MSG
Menu definition file	.MNU

All required fields on frames must be filled in before the frame is converted. If a frame lacks required information, an error message is displayed when you type CONVERT, and the frame is not converted. After conversion, FDT displays a message indicating the number of frames converted and the number of frames not converted.

#### Delete frame-identifier

The DELETE command allows you to delete the specified frame from the current file.

##### Parameter:

frame-identifier	The identifier of the frame to be deleted.
------------------	--

Before deleting the frame, the Frame Development Tool shows the Display form for the specified frame and prompts:

Delete this frame?

If you type Yes, the frame is discarded; if you type No, it is not. After you respond to the prompt, control returns to the File Command prompt.

#### File filename

The File Command allows you to open another frame definition file.

##### Parameters:

filename	The name of the frame definition file.
----------	--

If you do not specify a file name, the Frame Development Tool prompts:

Filename:

If you are creating a new frame file, the Frame Development Tool prompts for the type of file you want to create:

Create (H)elp, (M)essage, or (S)ingle-choice menu file?

Valid frame file types are help, message, and single-choice menu file. Type a carriage return if you want to return to the Filename: prompt.

If you did not save the current file before specifying another file,

the Frame Development Tool prompts:

Save current file?

Type Yes to save it or No to discard it.

### List

The LIST command displays an alphabetized list of the frames in the current definition file by frame identifier. The identifier is the name which was assigned to the frame when it was created with the ADD command.

### Report filename

The REPORT command creates a file about specified frames in the current frame definition file. You specify the frames you want a report on by selecting them individually from a list.

#### Parameter:

filename	The name of the output file that will contain the report.
----------	---

If you do not specify the output file name, the Frame Development Tool prompts:

Output file name?

The default file type for the report file is .RPT.

### Modify frame-identifier

The MODIFY command finds the specified frame in the frame definition file. After you have specified the frame-identifier, the Frame Command prompt is displayed, and you select the form you want to modify for this frame. valid responses are listed Section ?

#### Parameters:

frame-identifier	The identifier for the frame to be modified.
------------------	--

If you do not specify the frame-identifier, the Frame Development Tool displays a prompt requesting it. After specifying a frame identifier, FDT displays a message indicating the frame is available for modification.

### ISSUES:

1. How many frames can be placed in a menu definition file?  
Development Tool prompt for this? 80 OR 100?

12.2.3.2 Frame Editing - Frame Editing commands, which allow you to create and save frames, can be typed in response to the prompt:

Frame Command:

#### Profile

The Profile command displays a form that you can use to record information about the operation of the frame. Information on a Profile form directs menu services when the frame is displayed. This information is not displayed on the screen with the frame.

A Profile form contains fields, both required and optional, you must fill in. If you do not fill in a required field, an error message is displayed when you try to convert the frame definition file for execution; frames lacking required information are not converted. Single-choice menus, Help menus and Help frames, and messages have different Profile forms, shown in Section ?

The Frame Development Tool offers a simple screen editor for entering and manipulating text on a Profile form. The screen editor is described at the end of this section.

#### Display

The Display command allows you to record on a form the text to be displayed on the screen.

Information about the frame you record on a Profile or Action form may have an effect on what a menu looks like when it is displayed. For example, by assigning keywords to options, you cause those characters to appear in boldface in the display.

A Display form contains fields, both required and optional, you must fill in. The fields on a Display form correspond to the fields on a menu. These fields are illustrated in this chapter in Figure ? If you do not fill in a required field, an error message is displayed when you try to convert the frame definition file for execution; frames lacking required information are not converted. Single-choice menus, Help menus and Help frames, and message frames have different Display forms. These forms are shown in Section ?

The Frame Development Tool offers a simple screen editor for entering and manipulating text on a Display form. The screen editor is described at the end of this section.

#### Action [All | New | option-number]

The ACTION command displays a form requesting the action to be taken when an option is selected from the menu. Each option on a menu must be assigned an action. During application execution, action information is either passed back to the application program or used to display a Help frame for the option. This information is not

displayed on the screen with the frame.

Note that action information can be assigned only to menus. Help and message text frames cannot be assigned actions. Every menu must have at least two options; each option must be assigned an action.

The ACTION command parameters specify the options to be assigned action information.

Parameters:

- |               |  |
|---------------|--|
| ALL           | The ALL option refers to all options currently specified. The default option for ACTION is ALL.              |
| NEW           | The NEW parameter refers only to those options for which you have not entered information on an action form. |
| option-number | The option-number is the ordinal position, 1 to 12, of this option in the list of options.                   |

If the Action parameter specifies an option that has already been assigned action information, the current description is displayed on the Action form. You can then enter new action information or change existing action information.

An Action form contains fields, both required and optional, you must fill in. If you do not fill in a required field, an error message is displayed when you try to convert the frame definition file for execution; frames lacking required information are not converted. Single-choice menus and Help menus have different Action forms. These forms are shown in Section ?

The Frame Development tool offers a simple screen editor for entering and manipulating text on an Action form. The screen editor is described next.

## Frame Development Tool - Editing Keypad

When you fill in the Profile, Display, and Action forms to create a frame, you can use the Frame Development Tool's editing keypad to manipulate text. The editing keypad for the Frame Development Tool consists of the keys on the editing keypad on the VT100 family of terminals, or the numerical keypad on the Professional. All text entering is done in insert mode. Figure ? shows the layout of the screen editor keypad.

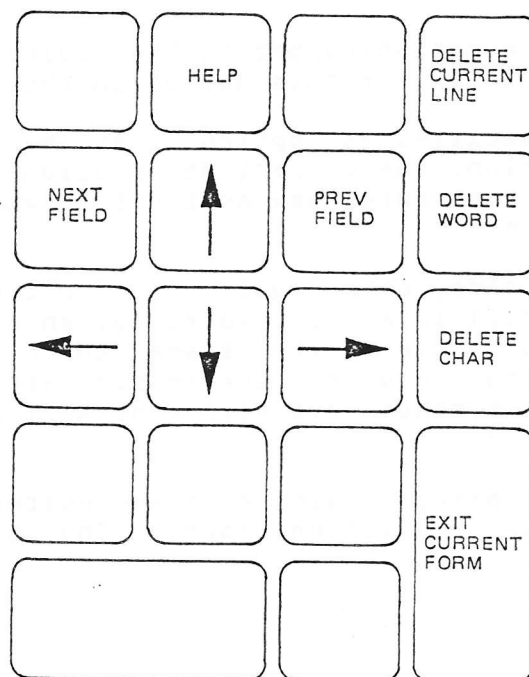


Figure ? : Screen Editor Keypad

The screen editor keys operate as follows:

Arrow keys	Move the cursor forward, backward, up, or down within a field, but not between fields. The arrow keys on the P/OS editing keypad can also be used for the same purpose, as well as the arrow keys on a VT100-compatible terminal.
HELP	Displays a description of the field the cursor currently points to. If pressed again, displays a description of the keypad keys.
DELETE CURRENT LINE	Deletes from cursor to end of line.
DELETE WORD	Deletes from cursor to next word.
DELETE CHARACTER	Deletes the character the cursor is on.
NEXT FIELD	Positions the cursor at the beginning of the next field on the form.
PREVIOUS FIELD	Positions the cursor at the beginning of the previous field on the form.
EXIT CURRENT FORM	Returns control to frame editing command level (to the prompt Frame Command:). If you are filling out Action forms and you press this key, the next Action form for the menu (if there is one) is displayed.

## 12.3 CREATING A SINGLE-CHOICE MENU

The first step in creating a single-choice menu is to fill in a Profile form (see Figure ?).

Figure ?: Profile Form for Single-choice Menu

Profile for Single Choice Menu [IDENTIFIER]

Frame Description	
Global Help Frame [    ] Default Option [    ] Global Action String	



The fields on a Profile form for a single-choice menu are defined as follows:

Frame Description	Optional field. Enter 4 lines of 72 characters. Use this field to document the overall operation of the menu for later reference.
Global Help Frame	Optional field. Enter 8 alphanumeric characters. Enter the frame identifier of the help frame that will provide help about the entire menu. When the menu is displayed, if the pointer is in the rest position and the HELP key is pressed, the help frame identified in this field is displayed.
Default option	Optional field. Enter a one or two-digit number. This field designates an option as the default option for this menu. Enter the ordinal number of the option that is to be the default for this menu. If you designate a default option, when the menu is displayed the pointer automatically points to the default option.
Global action string	Optional field. Enter 1 line of 72 characters. The global action string is returned to the application when the menu is displayed. Example uses: set flags before menu displays or set function keys.

Next, fill in a Display form (see Figure ?).

Figure ? : Display Form for Single-choice menu

Display for Single Choice Menu [IDENTIFIER]

[		TITLE TEXT		]	
[		EXPLANATORY TEXT		]	
[		OPTION DESCRIPTION		]	
[				]	

The fields on a Display form for a single-choice menu are defined as follows:

Title	Required field. Enter up to 40 alphanumeric characters. The title is displayed on the first line of the menu.
Explanatory text	Optional field. Enter 3 lines of 72 characters. The text introduces the end user to the options listed on the menu.
Option description	Required field. Enter at least two options with up to 64 characters. The options are the command interface to the application.
Prompt	Optional field. Enter up to 72 alphanumeric characters. Enter a message to the end user prompting for a menu selection.

Third, fill in one Action form for each option on the menu (see Figure ?). The message line at the top of the form shows the ordinal number of the option for which action is being assigned. It also shows the identifier of the menu on which the option is displayed. You must fill in an Action form for each option on the menu.

Figure ? : Action Form for Single-Choice Menu

Action Number [#] for Single Choice Menu [IDENTIFIER]

Description: [OPTION DESCRIPTION]	
[	Action Description ]
Option Keyword [ ] Option Help Frame [ ]	
[	Option Action String ]

The fields on an Action form for a single-choice menu are defined as follows:

Action Description	Optional field. Enter 2 lines of 72 characters. Use this field to document the overall operation of the menu for later reference.
Option keyword	<p>Required field. Enter 1 to 30 characters.</p> <p>Some number of contiguous characters in the option must be designated the keyword. You must designate one keyword for each option. An end user who types characters rather than moves the pointer must type the keyword to make a selection. When the menu is displayed, the keyword appears in boldface.</p> <p>At least 1 character must appear in boldface; all characters may be in boldface. The keyword must be unique; no keyword can be a subset of another keyword. For example, the number 1 is a subset of the number 10. These two numbers would not be valid keywords if they were together on the same menu.</p> <p>When a short-form menu is displayed, a maximum of 10 characters in boldface are displayed for each item.</p> <p>A verb is often appropriate as the keyword, as for commands. The keyword may be at the beginning of an option, or it may be the whole option, as in the following example:</p> <p style="margin-left: 40px;">Delete a file Copy a file</p> <p>However, the keyword may appear anywhere in the string, as in the following example:</p> <p style="margin-left: 40px;">File delete File copy</p> <p>If you cannot easily differentiate your menu entries, you may use numbered entries, although, this is not recommended. For example:</p> <p style="margin-left: 40px;">1 Write checks 2 Write balance</p>
Option Help Frame	Optional field. Enter 8 alphanumeric

characters of the help frame for this option. When the menu on which this option appears is displayed, if the end user presses the HELP key while the pointer is on this option, the help text frame is displayed.

Option Action String

Optional field. Enter up to 72 characters. The option action string is returned to the application when the option is selected.

## 12.4 CREATING HELP MENUS AND HELP TEXT FRAMES

The first step in creating a help menu is to fill in a Profile form (see Figure ?).

Figure ? : Profile Form for Help Menu

Profile for Help Menu [IDENTIFIER]

Frame Description	
[ ]	
Previous Help Frame [ ]:	
Default Option [ ]	

The fields on a Profile form for a help menu are defined as follows:

Frame Description	Optional field. Enter 4 lines of 72 characters. Use this field to document the overall operation of the menu for later reference.
Previous Help frame	Optional field. Enter 8 alphanumeric characters. Enter the identifier of the previous help frame.
Default option	Optional field. This field designates an option as the default for this menu. Enter the ordinal number of the option. When a menu is displayed, the pointer automatically points to the default option.



The next step in creating a help menu is to fill in a Display form (see Figure ?).

Figure ? : Display Form for Help Menu

Display for Help Menu [IDENTIFIER]

[		TITLE TEXT	]
[	EXPLANATORY TEXT		]
[		OPTION DESCRIPTION	]
[			]

The fields on a Display form for a help menu are defined as follows:

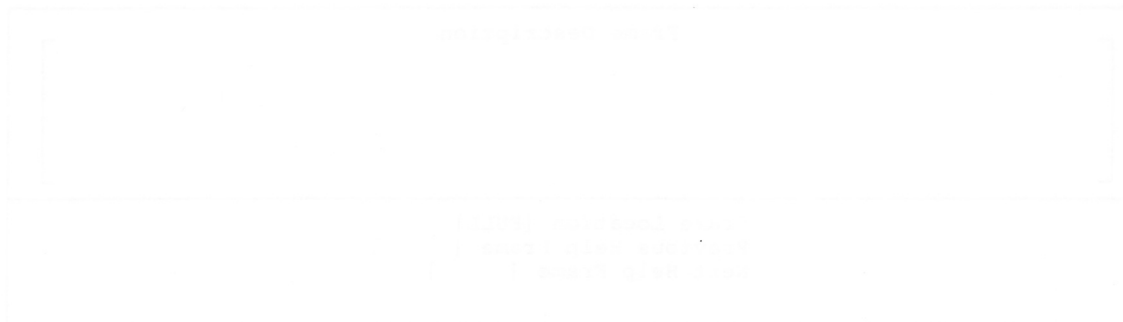
Title	Required field. Enter up to 40 alphanumeric characters. The title is displayed on the first line of the menu.
Explanatory text	Optional field. Enter 3 lines of 72 characters. The text introduces the end user to the options listed on the menu.
Option description	Required field. Enter at least two options with up to 64 characters each. The options are the command interface to the application.
Prompt	Optional field. Enter up to 72 alphanumeric characters. Enter a message to the end user prompting for a menu selection.



The fields on an Action form for a help menu are defined as follows:

Action Description	Optional field. Enter 2 lines of 72 characters. Use this field to document the overall operation of the menu for later reference.
Option keyword	<p>Required field. Enter 1 to 30 characters</p> <p>Some number of contiguous characters in the option must be designated the keyword. You must designate one keyword for each option. An end user who types characters rather than moves the pointer must type the keyword. When the menu is displayed, the keyword appears in boldface.</p> <p>At least 1 character must appear in boldface; all characters may be in boldface. The keyword must be unique; no keyword can be a subset of another keyword. For example, the number 1 is a subset of the number 10. These two numbers would not be valid keywords if they were together on the same menu.</p> <p>When a short-form menu is displayed, a maximum of 10 characters in boldface are displayed for each item.</p> <p>A verb is often appropriate as the keyword, such as for commands. The keyword may be at the beginning of an option, or it may be the whole option, as in the following example:</p> <p style="margin-left: 40px;">Delete a file Copy a file</p> <p>However, the keyword may appear anywhere in the string, as in the following example:</p> <p style="margin-left: 40px;">File delete File copy</p> <p>If you cannot easily differentiate your menu entries, you may use numbered entries, although, this is not recommended. For example:</p> <p style="margin-left: 40px;">1 Write checks 2 Write balance</p>
Option Help frame	Optional field. Enter 8 alphanumeric characters of the help frame for this option.

When the menu on which this option appears is displayed, if the end user presses the HELP key while the pointer is on this option, the help text frame is displayed.



The first step in creating a help text frame is to fill out a Profile form (see Figure ?).

Figure ? : Profile Form for Help Text Frame

Profile for Help Text [IDENTIFIER]

[ Frame Description ]
Frame Location [FULL] Previous Help Frame [     ] Next Help Frame [     ]

The fields on a Profile form for a help text frame are defined as follows:

Frame Description	Optional field. Enter 4 lines of 72 characters. Use this field to document the overall operation of the menu for later reference.
Frame location	Required field. Valid entries in this field are FULL, TOP, BOTTOM, and LINE. A full frame has 16 lines. A top or bottom half frame has 8 lines. A line frame has one line. One line frames have no title. To enter a value other than FULL, position the cursor on the "F" in FULL and press the DELETE WORD key on the FDT keypad. Enter one of the other values.
Previous Help frame	Optional field. Enter 8 alphanumeric characters. Enter the identifier of the previous help frame.
Next Help Frame	Optional field. Enter 8 alphanumeric characters. Enter the identifier of the next help frame.

Full screen, top or bottom half help text, and single line help text have different Display forms representing the difference in the number of lines in the frame. However, they all share the same fields and the full screen help text frame is representative.

Figure ? shows a Display form for a full screen help text frame.

Figure ? : Display Form for Help Text Frame

Display for Help Text [IDENTIFIER]

[	TITLE TEXT	]
[	Help Text	]



The fields on a Display form for a help text frame are defined as follows:

Title	Required field. Enter up to 40 alphanumeric characters. The title is displayed on the first line of the menu.
Help text	Optional field. Enter 16 lines of 72 characters. The text explains how to perform an application-related activity, such as supplying a command in the proper format or displaying or using data.

## 12.5 CREATING MESSAGE TEXT FRAMES

First, to create a message text frame fill in a Profile form (see Figure ?).

Figure ? : Profile Form for Message Frame

Profile for Message [IDENTIFIER]

Frame Description	

The fields on a Profile form for a message text frame are defined as follows:

Frame Description

Optional field. Enter 21 lines of 79 characters. Use this field to document the overall operation of the menu for later reference.

Second, for a message text frame fill in a Display form. Message text frames are displayed with no border or prompt. The Display form for this type of frame is simply a blank screen with the usual message line at the top identifying the current frame.

## 12.6 RESOLVING ERROR CONDITIONS

All user errors occurring with FDT produce a short beep from the keyboard.

While you are filling in forms, if a key that is not on the FDT keypad is pressed or if you attempt to type over field boundaries, the keyboard beeps and the key action is ignored.

All other user errors produce a descriptive error message following the keyboard beep. FDT Error messages are described here.

### 12.6.1 Format Of Messages

In this section, each FDT error message appears in the same form as on your display terminal. The messages appear in the following form:

Message text.

or

Message text about <frame or file or name>.

In the documentation, left and right angle brackets (< and >) enclose a symbol. The symbol represents a reference to a value that FDT copies from your commands or retrieves from stored information. The references are to field names or to file or frame specifications. These references may appear as the first, middle, or last element in an error message.

When you receive a message, look it up in the following alphabetized section, read the short explanation about the reason for the message, and apply the remedies that are described. The symbols in angle brackets are not used to alphabetize messages.

Some error messages report error conditions that cannot be resolved by you. These error messages indicate that a serious error has occurred in FDT processing. In most cases, the errors are the result of a failed file operation. These errors appear in the following form:

FDTxxx,yyy - Message text.

The values xxx and yyy represent RMS-11 error status numbers. These numbers correlate to a specific error which is generally described by the message text. These two values may be positive or negative.

When you receive a message of this type, you should submit a Software Performance Report (SPR) to DIGITAL. An SPR is a form that most customers (who are in-warranty or have purchased support services) can use to report faults in the software and suggest product improvements. These messages are listed together in Section ?

### 12.6.2 User Error Messages

Action form deleted.

An option description line was deleted or the keyword in an option description line was divided onto two lines on a Display form. The action form for the option description line was deleted by FDT. An Action form is deleted if the option it describes no longer exists.

If you deleted the option description line, create another one. When you create the new one, an action form will be set up for it. If you divided a keyword onto two lines on the Display form, enter it on only one line. A new action form will be set up for it.

<Frame-id> already exists - no new frame created.

The frame identifier of an existing frame was specified in an Add command. Use the Modify command to alter the existing frame or enter a unique frame identifier.

<filename> already saved.

The current file was not changed, so no new copy of the file is created. Control returns to host system command level.

Current file full - frame not added.

An attempt was made to add another frame to the current file with the Add command but the file contains the maximum number of frames.

Delete a frame in the current file to make room for the new frame or start a new file.

Default option is not in the range 0 to xx.

The default option field on a Profile form has a value outside the valid range. The valid range is 0 to xx, where xx is the total number of option description lines on the Display form. This error condition is detected after the Convert command was used to convert a definition file.

Use the MODIFY command to alter the Profile form to contain a value in the range.

Default option specified with no options.

A value was specified in the default option field on the Profile form but no options were listed on the Display form. This error condition is detected after the Convert command was used to convert a definition file.

Create two or more options in a Display form. Check to make sure the default option field on the Profile form is correct.

<frame-id> does not exist.

A frame identifier was specified with the Modify or Delete command but it does not exist in the current file.

Check for typographical errors in the frame identifier. To display the list of frames contained in the current definition file, type a carriage return. Then use the List command to display the frame identifiers.

Enter (H)elp, (M)essage, or (S)ingle-choice to select a file type.

A character other than H, M, or S was entered in response to the previous prompt.

To select a file type, respond to the prompt by entering the single character in parentheses in the message. The file type determines the type of frames that may be created and stored in the current definition file. To specify a new definition file, enter a carriage return. The prompt Filename: will be displayed.

Field <name> invalid in action form <number>.

The named field, which is on the Action form with the indicated number, does not contain valid information. This error is detected after the Convert command was used to convert a definition file.

Use the MODIFY command to fix the specified field. The most likely reason for this error message is a blank value for a required field.

Field <name> invalid in display form.

The named field, which is on a Display form, does not contain valid information. This error is detected after the Convert command was used to convert a definition file.

Use the MODIFY command to fix the specified field. The most likely reason for this error message is a blank value for a required field.

Field <name> invalid in profile form.

The named field, which is on the Profile form, does not contain valid information. This error is detected after the Convert command was used to convert a definition file.

Use the MODIFY command to fix the specified field. The most likely reason for this error message is a blank value for a required field.

File <filename> is empty. File not written.

An attempt was made to save a file with no frames in it. The file is not saved and control returns to host system command level.

Make sure the correct file was specified or use the QUIT command to leave FDT after opening an empty file.

File <filename> not found.

The specified file is either not a valid FDT definition file or the file does not exist.

Create a new FDT definition file using the specified name or enter a carriage return to enter a different file name.

File not written - no frames to write.

The Convert command specified a definition file but it contains no valid frames that may be converted. Any frames in the current definition file contain errors.

Fix the errors in the frames, and then use the Convert command to convert the file.

Frame <frame-id> not converted.

The named frame was not converted because it contained invalid information or it lacked required information in one or more fields.

Use the Modify command to fix the listed errors. Then use the Convert command to convert the frames.

Invalid command. Enter HELP for a list of commands.

You entered an invalid command in response to the file or frame command prompt.

Review the list of valid commands in the FDT documentation or type Help to display the list of valid commands.

Invalid file specification <filename>.

An error was made in the format of the file specification.

Check for typographical errors in the file specification. Verify that the dev:filename.typ file specification format was used. Retype the file specification.



Invalid identifier.

A frame identifier was specified with more than 8 characters.

Enter a frame identifier with no more than 8 characters.

Keyword deleted.

The keyword for the option description line that was just altered was deleted. Keywords are deleted whenever the keyword no longer matches the option description line.

Enter a new keyword for the new option description line.

Keyword does not match description line.

The value entered in the keyword field on an Action form is invalid. It does not match any segment of contiguous characters in the option description line on the Display form.

You must either leave the option keyword field blank or enter a unique, matching keyword in the keyword field.

Keywords not unique for options xx and yy.

Two matching keywords were specified for the named options. Either keyword xx is a subset of keyword yy or vice versa, or they are identical. For example, "cat" is a subset of "catch" and the number 1 is a subset of the number 10. This error condition is detected after the Convert command was used to convert a definition file.

Alter the keyword for one or both of the specified options. If you decide to alter an option description line on the Display form, do not delete the line because the action form for the option will be deleted.

No action form found for option xx.

An action form was not filled in for the named option. This error condition is detected after the Convert command was used to convert a definition file.

Use the MODIFY command to fill out the Action form.

No frames to report.

A report on the current definition file was requested but the file contains no frames.

Make sure that the correct file is open.

No matching option found.

A nonexistent option was specified in the Action command.

Check for typographical errors in the command line. Review the list of options on the Display form.

No new options found.

The Action New command was specified but all options have been assigned actions.

Use the Action All or Action option-number commands to modify the desired actions.

No option lines defined.

The Action command was used but no option description lines have been specified.

Use the Display command to enter the option description lines, and then use the Action command to assign actions to each option.

No value entered for required field.

This is a warning message indicating that the previous field requires a value before this frame can be converted.

Determine what the value should be for the previous field and enter that value in the field before attempting to convert the file.

Not enough options defined - at least two are required.

A menu frame with less than two options cannot be converted.

Add at least two options to the menu.

Not enough room to insert a line.

There is not enough room in the field to open a new line.

Use the arrow keys, the next and previous field keys, or the exit key on the FDT editor keypad to achieve the desired result.

Please answer Yes or No.

You have responded incorrectly to a prompt requiring a Yes or No response.

Respond with a Y for Yes, or an N for No.

Please enter FULL, TOP, BOTTOM, LINE.

The value entered for the field is not one of the valid values listed.

You cannot proceed to the next field without entering a correct response. Enter Full, Top, Bottom, or Line to specify the display location of a help text frame. Initially, FULL is the value in the location field. To replace FULL or an entered value, use the arrow keys to position the cursor on the F, then press the DELETE WORD key on the FDT keypad. Enter a correct value.

Please respond with an "M" or "T".

Before you can add a frame in a help definition file, you must specify whether you are adding a help menu or a help text frame.

Enter an "M" to add a menu, a "T" to add a text frame, or carriage return to display the file command prompt.

Value must be between 0 and 12.

A value was entered for the default option field but the value is not an integer between 0 and 12, or a blank.

You cannot proceed to the next field without entering a correct response. The maximum number of options on a menu is 12. You can assign any one of the 12 options to be the default option. If you assign an option to be a default, when the menu is displayed the pointer will rest by the default option. Enter a valid numeric value for the field or leave it blank.

### 12.6.3 FDT Internal Errors

For all of the following error messages, the remedy is as follows: Attempt to exit FDT. If the error didn't occur when you were saving a definition file, attempt to save it. Obtain a hard copy listing of the file using the REPORT command. Submit an SPR to DIGITAL. Include with the SPR a detailed description of the user-FDT dialogue preceding the error message, and a listing of the definition file. Record on the SPR the error message exactly as it was displayed on your terminal, including the values represented here as <xxx,yyy>.

FDT<xxx,yyy> - Cannot create output file.

The output file specified in the Convert command cannot be written due to a processing error.

FDT<xxx,yyy> - Cannot create temporary file.

The file specified cannot be opened because of a processing error in the file manager.

FDT<xxx,yyy> - Cannot display graphics frame.

A processing error occurred while FDT was attempting to display a form.

FDT<xxx,yyy> - Cannot read frame <frame-id>.

An attempt to read the specified frame failed.

FDT<xxx,yyy> - Cannot read frame from temporary file.

This error indicates that a record is inaccessible from the MODIFY command.

FDT<xxx,yyy> - Cannot read record.

A record in the current file cannot be accessed for conversion.

FDT<xxx,yyy> - Cannot read record <frame-id> from temporary file.

The named frame cannot be converted due to a processing error. FDT continues to convert the remaining records in the file.

FDT<xxx,yyy> - Cannot write index of converted file.

The output file from the Convert process has been corrupted. Under no circumstances should the output file be used on the Professional.

FDT<xxx,yyy> - Cannot write record <frame-id>.

The named frame cannot be converted due to a processing error. FDT continues to convert the remaining records in the file.

FDT<xxx,yyy> - Closing original file after copy.

The file specified cannot be opened because of a processing error in the file manager.

FDT<xxx,yyy> - Frame <frame-id> unreadable.

The specified frame exists in the file but it cannot be read.

FDT<xxx,yyy> - Index not written to data file.

The current file cannot be saved in a permanent file due to a processing error.

FDT<xxx,yyy> - New file cannot be created.

The current file cannot be saved in a permanent file due to a processing error.

FDT<xxx,yyy> - Read #zz during initial copy.

The specified file cannot be opened because of a processing error in the file manager.

FDT<xxx,yyy> - Record <frame-id> could not be written to data file.

The named frame cannot be saved in the permanent file due to a processing error.

FDT<xxx,yyy> - Record <frame-id> unreadable from temporary file.

The named frame cannot be saved in the permanent file due to a processing error.

FDT<xxx,yyy> - Unable to write record.

The frame editor cannot write the last record to the file.

FDT<xxx,yyy> - Write #zz during initial copy.

The file specified cannot be opened because of a processing error in the file manager.



## CHAPTER 13

### P/OS SERVICES LIBRARY

P/OS menu services provides facilities for displaying and operating the menu, help, and message frames that you create with the Frame Development Tool. These facilities, called service routines, are provided in a shared resident library called the P/OS Services Library. They are used for displaying frames from the previously defined frame definition files. These services can be called from high-level languages.

#### 13.1 DETAILED OVERVIEW

To be supplied.

## 13.2 CONVENTIONS

Certain documentation conventions are used in this chapter.

Routine names shown are the actual global symbol names as known by the task-builder (TKB).

Parameter names are for illustration purposes only. The exact names are defined by the application developer, subject to the restrictions of the language being used.

Parameters shown inside square brackets ([ and ]) are optional and may be omitted. Parameters may only be omitted from the right, so that if parameter <n> is omitted, then parameter <n+1> must also be omitted.

Each programming language may call the routines in a manner appropriate to the language.

Fortran should use the standard CALL statement.

Basic-Plus-2 should use CALL xxx BY REF. parameters.

In all cases, registers are NOT preserved by the called routine except for the stack pointer which is preserved.

## 13.3 MENU SERVICE ROUTINES

This section describes the service routines used for displaying single and multiple-choice menus.

Menu and help services use internal buffers for temporary storage of menu frames. There are three buffers for menus:

1. Static buffer for static single-choice menus
2. Dynamic buffer for dynamic single-choice menus
3. Multi buffer for multi-choice menus



## OPEN MENU FILE

Opens the specified menu definition file. Only one menu file can be open at any time. If another menu file was open, it is closed before the requested file is opened.

Instead of issuing this call when the application executes, a default menu definition file may be specified in the application installation file. When the end user installs the application, the default menu file is assigned to the application and automatically opened when the application is invoked.

## Call:

CALL MFILE (status,filename,maxlen)

## Parameters:

status	A two-word integer field used to return a code indicating the results of the requested operation. Values for status are listed at the end of this section.
filename	A string variable or literal of any length specifying the menu definition file name and type. The directory used for the menu file is the directory in which the application resides.
maxlen	Integer variable or constant. This is the maximum length of the immediately preceding buffer or string parameter. For read-only parameters, this may be the length of the significant data only.

## CLOSE MENU FILE

Closes an open menu file. To read another menu frame into the buffer, use the OPEN MENU FILE service routine.

## Call:

CALL MCLOSE (status)

Parameters: none

Status return codes:

See Section ?

## Errors:

none

## READ MENU FRAME

Reads the specified menu frame into the static menu buffer. Using one of the display or pack routines, the frame in the static buffer may be displayed or modified. The global action string associated with the frame, if present, is returned to the application.

## Call:

CALL MFRAME (status,frameid,maxlen,globalstring,maxlen length)

## Parameters:

status	A two-word integer field used to return a code indicating the results of the requested operation. Values for status are listed at the end of this section.
frameid	An 8-character string containing the frame identifier of the desired menu frame. All characters in frameid must be in uppercase letters.
maxlen	Integer variable or constant. This is the maximum length of the immediately preceding buffer or string parameter. For read-only parameters, this may be the length of the significant data only.
globalstring	<p>An optional string variable of any length. This parameter is used to return the global action string associated with the menu frame, if present. If the buffer supplied is shorter than the global action string, the string is truncated to fit; if the buffer is longer, it is blank-filled.</p> <p>If this parameter is not specified, the global action string is not returned, even if it is present in the menu frame.</p>
maxlen	Integer variable or constant. This is the maximum length of the immediately preceding buffer or string parameter. For read-only parameters, this may be the length of the significant data only.
length	An integer variable. This parameter is a returned value indicating how much of the buffer or string parameter was actually used by the operation.

## SHOW SINGLE-CHOICE MENU

Displays a frame from the current static menu buffer, and accepts the keystrokes entered in response by the end user. Menu interaction is terminated when the user presses the DO key or a function key that does not have meaning within a menu. The menu service returns the user's response to the application in the status parameter.

If the end user presses the HELP key while a menu is being displayed, the current help frame is read from the help definition file and displayed. If the HFRAME service routine was not used to define the current help frame, no help frame is displayed. If other help frames are defined for display from the current HFRAME, they are displayed when the end user presses the HELP key again. The application should change the current frame designation with HFRAME each time a different menu is displayed. The single-choice menu is redisplayed when the end user presses the RESUME key.

Before using this service routine to show the menu, the application must use OPEN MENU FILE and READ MENU FILE to place the frame into the menu buffer. (A default menu file assigned when the application was installed may have been used instead of OPEN MENU FILE.)

During the execution of this routine, menu services handles the following function keys:

- CANCEL
- DO
- ADDITNL OPTNS
- HELP
- arrow keys

If the HELP key is pressed, the following keys are handled:

- RESUME
- PREV SCREEN
- NEXT SCREEN

If the executing program had assigned function key ASTs to these keys, the AST routines are overridden. The state of the application at the time of the call is saved and restored before control returns to it.

Other function keys terminate the menu display, and return a flag to the executing application indicating which function key was pressed.

Call:

CALL MENU (status,action,maxlen,length,displayflag,addoptflag,  
.br  
msg1,maxlen,msg2,maxlen)

Parameters:

status            A two-word integer field used to return  
                  a code indicating the results of

the requested operation. Values for status are listed in Section ?

action	A string variable of any length. This parameter is used to return the action string for the selected option. If the action string is longer than the string provided, it is truncated; if it is shorter, the buffer is blank-filled.
maxlen	Integer variable or constant. This is the maximum length of the immediately preceding buffer or string parameter. For read-only parameters, this may be the length of the significant data only.
length	An integer variable. This parameter is a returned value indicating how much of the buffer or string parameter was actually used by the operation.
displayflag	An integer variable or constant. This parameter indicates what type of menu to show:  0 = full screen menu display 1 = abbreviated menu at top of screen 2 = abbreviated menu at bottom of screen
addoptflag	An integer variable or constant. This parameter indicates whether to show an Additional Options flag on the screen. Values are:  0 = No flag 1 = Display Additional Options flag
msg1	A string variable or literal of any length. This parameter contains the text to be displayed as a message at the bottom of the screen below the menu border.
maxlen	Integer variable or constant. This is the maximum length of the immediately preceding buffer or string parameter. For read-only parameters, this may be the length of the significant data only.
msg2	A string variable or literal of any length. This parameter contains the text to be displayed as a message at the bottom of the screen below the menu border.

#### ISSUES:

1. what happens if HFRAME not used to specify current help frame and HELP is pressed?

## UNPACK MENU BUFFER

The MUNPK routine unpacks the STATIC buffer into its component parts. The parts may be modified then put into the DYNAMIC buffer using DPACK.

MUNPK accepts any number of parameter groups, plus an initial status parameter. The general case is that fieldid is followed by some number of associated parameters. This group may then be followed by another group starting with another fieldid. Special-case parameters may appear anywhere in the call.

## Call:

```
CALL MUNPK (munstatus,fieldid,maxlen,buff,maxlen,length
.br
[,fieldid,maxlen,buff,maxlen, length ...])
```

## Parameters:

munstatus	A two-word integer field used to return a code indicating the results of the requested operation. Values for status are listed in Section ?
fieldid	A string variable or literal of 6 characters. The fieldid specifies what field is to be filled or returned to pack or unpack a menu buffer.
	Valid fieldid values are:
	<pre>TITL      title field TEXTnn    explanatory text line 1, 2 or 3 GACT      global action string GHLP      global help frame id DFLnn     default option number PRMT      prompt text line OPTnnnn   option text for option nn (01-12) ACTnnnn   action string for option nn (01-12) UHLPrnn   help frame id for option nn (01-12) KEYwnnn   specification of key portion of option            nn (01-12) text</pre>
maxlen	Integer variable or constant. This is the maximum length of the immediately preceding buffer or string parameter. For read-only parameters, this may be the length of the significant data only.
buff	A string variable of any length. This parameter specifies the value of a field to be filled or returned to pack or unpack a menu buffer.
maxlen	Same as previous maxlen.

length            An integer variable. This parameter is a returned value indicating how much of the buffer or string parameter was actually used by the operation.

The following special cases are defined:

[DFLT, maxlen, defopt]            Gets the default option number. defopt is an integer variable which returns the default option number (1-12) for the menu being unpacked.

[KEYwnn, maxlen, offset, length]    Gets the offset and length for the key portion of option nn.

## PACK DYNAMIC SINGLE CHOICE MENU

CALL DPACK (status,fieldid,maxlen,buff,maxlen  
[,fieldid,maxlen,buff,maxlen...])

This routine packs the DYNAMIC menu buffer using specified components. DPACK accepts any number of parameter groups, plus an initial status parameter. The general case is that the fieldid parameter is followed by some number of associated parameters. This group may then be followed by another group starting with another fieldid.

## Parameters:

status           A two-word integer field used to return a code indicating the results of the requested operation. Values for status are listed in Section ?

fieldid          A string variable or literal of 6 characters. The fieldid specifies what field is to be filled or returned to pack or unpack a menu buffer.

Valid fieldid values are:

TITL	title field
TEXTnn	explanatory text line 1, 2 or 3
GACT	global action string
GHLP	global help frame id
DFLTnn	default option number
PRMF	prompt text line
OPTWnn	option text for option nn (01-12)
ACTNnn	action string for option nn (01-12)
GHLPnn	help frame id for option nn (01-12)
KEYWnn	specification of key portion of option nn (01-12) text

maxlen          Integer variable or constant. This is the maximum length of the immediately preceding buffer or string parameter. For read-only parameters, this may be the length of the significant data only.

buff            A string variable of any length. This parameter specifies the value of a field to be filled or returned to pack or unpack a menu buffer.

maxlen          Same as previous maxlen.

The following special-case parameters are defined:

[DFLTnn, maxlen]	Sets the default option number to option nn.
------------------	--

[KEYWnn,maxlen,offset,length]	Sets the offset and length for the key portion of option nn.
-------------------------------	--



[CLRB]

If status is the only parameter  
or if a fieldid of CLRB  
is encountered, then clear  
the dynamic buffer.

Note that only one menu may be packed at a time because there is only one dynamic buffer. The dynamic buffer must be cleared before packing begins. The order of supplying fields with fieldid is not significant.

## DISPLAY DYNAMIC MENU FRAME

Displays a single-choice menu constructed from the dynamic buffer. The operation is the same as for the MENU routine.

Before using this service routine to show the menu, the application must use OPEN MENU FILE and READ MENU FILE to place the frame into the menu buffer. (A default menu file assigned when the application was installed may have been used instead of OPEN MENU FILE.) After filling the buffer, use the UNPACK MENU BUFFER to access and modify the fields in the buffer. Use the display routine to show the resulting menu. These dynamic menus are created under program control during application execution.

If the user pressed the HELP key, the help display is initiated, and this menu is redisplayed when the user presses the RESUME key.

During the execution of this routine, the following function keys are handled:

- CANCEL
- HELP
- DO
- ADDTNL OPTIONS
- arrow keys

If the HELP key is pressed, the following keys are also handled:

- RESUME
- PREV SCREEN
- NEXT SCREEN

If the executing application had assigned function key ASTs to these keys, they are overridden. The application state at the time of the call is saved and restored before return.

Call:

```
CALL DMENU (status,action,maxlen,length,display,addoptflag,  
.br  
msg1,maxlen,msg2,maxlen)
```

Parameters:

status	A two-word integer field used to return a code indicating the results of the requested operation. Values for status are listed in Section ?
action	A string into which is returned the action string associated with the option selected. If the action string is longer than the string provided, it is truncated; if it is shorter, the

buffer is blank-filled.  
Maximum length of an action  
string is 80 characters.

maxlen Integer variable or constant. This is the maximum length of the immediately preceding buffer or string parameter. For read-only parameters, this may be the length of the significant data only.

length An integer variable. This parameter is a returned value indicating how much of the buffer or string parameter was actually used by the operation.

display An integer flag indicating the screen display requested by the caller.

0 = full screen menu display  
1 = short-form menu, top of screen  
2 = short-form menu, bottom of screen

adoptflag An optional parameter which, if present, is an integer flag indicating the presence of additional options for this menu. If the parameter is present, a zero value indicates that additional options are not expected by the application, and the additional options indicator is not displayed. A value of 1 indicates that additional options may be chosen, and the additional options flag is displayed with the menu frame. In any case, the additional options key is treated as any other function key. The display is terminated and the key character sequence is passed to the application either through a function key AST (if the application has previously so requested) or in the status return code.

msg1 A string variable or literal of any length. This parameter contains the text to be displayed as a message at the bottom of the screen below the menu border.

maxlen Same as previous maxlen.

msg2 Same as msg1.

maxlen Same as previous maxlen.

## PACK MULTI CHOICE (DYNAMIC) MENU

Packs the multi buffer with peripheral parts of the multi-choice menu, such as the title.

```
CALL MPACK (status,fieldid,maxlen,buff,maxlen  
.br  
[,fieldid,maxlen,buff,maxlen...])
```

## Parameters:

**status** A two-word integer field used to return a code indicating the results of the requested operation. Values for status are listed in Section ?

**fieldid** A string variable or literal of 6 characters. The fieldid specifies what field is to be filled or returned to pack or unpack a menu buffer.

Valid fieldid values are:

TITL	title field
TEXTnn	explanatory text line 1, 2 or 3
GACT	global action string
GHLP	global help frame id
DFLinn	default option number
PRMT	prompt text line
OPTNnn	option text for option nn (01-12)
ACTNnn	action string for option nn (01-12)
UHLPnn	help frame id for option nn (01-12)
KEYwnn	specification of key portion of option nn (01-12) text

**maxlen** Integer variable or constant. This is the maximum length of the immediately preceding buffer or string parameter. For read-only parameters, this may be the length of the significant data only.

**buff** A string variable of any length. This parameter specifies the value of a field to be filled or returned to pack or unpack a menu buffer.

**maxlen** Same as previous maxlen.

## DISPLAY MULTI-CHOICE (DYNAMIC) MENU FRAME

Displays a multi choice menu from the multi-buffer.

CALL MMENU (status,optioncount,maxlen,option-array,limit,  
responses,responsearray,adoptflag,msg1,maxlen,msg2,maxlen)

status	A two-word integer field used to return a code indicating the results of the requested operation. Values for status are listed in Section ?
optioncount	An integer variable or constant. This parameter is one element of optionarray.
maxlen	Integer variable or constant. This is the maximum length of the immediately preceding buffer or string parameter. For read-only parameters, this may be the length of the significant data only.
option-array	A two-dimensional array of characters; dimensioned with the parameters maxlen by optioncount. The characters 1 to maxlen must be contiguous in memory.
limit	An integer variable or constant. This parameter specifies the maximum number of options that may be chosen in a multi-choice menu.
responses	An integer variable which returns the number of options actually selected from a multi-choice menu.
responsearray	An integer array with the length of the value in the limit parameter. This parameter is returned with a list of the option numbers selected from a multi-choice menu.
adoptflag	An optional parameter which, if present, is an integer flag indicating the presence of additional options for this menu. If the parameter is present, a zero value indicates that additional options are not expected by the application, and the additional options indicator is not displayed. A value of 1 indicates that additional options may be chosen, and the additional options flag is displayed with the menu frame. In any case, the additional options key is treated as any other function key. The display is terminated and the key character sequence is passed to the application either through a function key AST (if the application has previously so requested) or in the status return code.

msg1            A string variable or literal of any length.  
                 This parameter contains the text to be  
                 displayed as a message at the bottom of the  
                 screen below the menu border.

maxlen          Same as previous maxlen.

msg2            Same as msg1.

maxlen          Same as previous maxlen.

#### 13.4 HELP SERVICE ROUTINES

P/OS provides access to the Help services through the following service routines.

##### OPEN HELP FILE

Opens the specified help definition file. Only one help file may be open at any time. If another help file is open, it is closed before the requested file is opened.

Instead of issuing this call when the application executes, a default help definition file may be specified when the application is installed. In this case, when the end user requests help for this application, the default help file is used.

This call does not result in a display. It only defines the current help file name and the current frame.

Call:

CALL HFILE (status,filename,maxlen,frameid,maxlen)

Parameters:

status	A two-word integer field used to return a code indicating the results of the requested operation. Values for status are listed in Section ?
filename	A string specifying the file name and type for the help file. (The directory used for the help file is the directory in which the application resides.)
maxlen	Integer variable or constant. This is the maximum length of the immediately preceding buffer or string parameter. For read-only parameters, this may be the length of the significant data only.
frameid	An eight-byte string containing the key to be used in retrieving the desired help frame. Frameid may not be blank or of zero length; a valid frameid must be specified.
maxlen	Same as previous maxlen.

**CLOSE HELP FILE**

Closes the help file if one was previously open. P/OS displays the following message if the HELP key is subsequently pressed:

"SORRY, NO HELP AVAILABLE"

Call:

CALL HCLOSE (status)

Parameters:

status	A two-word integer field used to	return
a code indicating the results of	the	requested
operation. values for status are listed in Section ?		



## SPECIFY HELP FRAME

Specifies the identifier of the frame to be displayed when help is requested. If a default help definition file was assigned when the application was installed, this call is unnecessary. This call does not result in a display; it simply defines the current frame.

## Call:

CALL HFRAME (status,frameid,maxlen)

## Parameters:

status	A two-word integer field used to return a code indicating the results of the requested operation. Values for status are listed in Section ?
frameid	An eight-byte string containing the key to be used in retrieving the desired help frame. Frameid may not be blank or of zero length; a valid frameid must be specified.
maxlen	Integer variable or constant. This is the maximum length of the immediately preceding buffer or string parameter. For read-only parameters, this may be the length of the significant data only.

## DISPLAY HELP FRAME

This routine retrieves and displays the current help frame from the current help file. The user response is accepted. The last specified help frame is the one named in a SPECIFY HELP FRAME service routine. The help frame specified is read into the help buffer. The executing application must restore the previous screen display after using this routine.

A help file must be open. If no help file is open or no current frame has been defined, the message NO HELP IS AVAILABLE is displayed. If the current frame as specified by a previous call is to be used, then the frameid parameter may be omitted or may be specified as all blanks or of zero length.

A status code indicating that the message was displayed is returned to the application. The user's response may lead to the display of additional help frames. When the user presses the RESUME key, control returns to the executing application. During the execution of this routine, the following function keys are handled:

RESUME  
PREV SCREEN

If a help frame is displayed, the following keys are handled:

NEXT SCREEN

If a help menu is displayed, the following keys are handled:

CANCEL  
DO  
arrow keys

If the executing application had specified function key ASTs, these are overridden. The application state at the time of the call is saved and restored before control returns to it.

Call:

CALL HELP (status[,frameid,maxlen])

Parameters:

status	A two-word integer field used to return a code indicating the results of the requested operation. Values for status are listed in Section 7.
frameid	An eight character string containing the key to be used in retrieving the desired help frame. Frameid may not be blank or of zero length; a valid frameid must be specified.

maxlen Integer variable or constant. This is the maximum length of the immediately preceding buffer or string parameter. For read-only parameters, this may be the length of the significant data only.

### 13.5 MESSAGE SERVICE ROUTINES

The following service routines provide access to the message file during program execution.

#### READ MESSAGE

Reads the specified message from the specified message file into a buffer supplied by the application.

The message file to be used must be specified for each call to RDMSG. This allows multiple message files to be in use by one application, although only one file will be open at any given time. For example, the application can have a message file which is distinct from that used by the language-specific Object Time System.

If the message file cannot be opened or read, the buffer is filled with the message "CAN'T OPEN MESSAGE FILE."

If the buffer is shorter than the message record, the record is truncated; if the buffer is longer, it is blank-filled. This facility may not be accessible during asynchronous processing.

Instead of issuing this call when the application executes, a default message definition file may be specified when the application is installed. The assigned message file is used if the file name parameter is not specified. This is for application use only. Layered products, such as language Object Time Systems (OTS routines) or similar programs, must supply the file specification as a parameter with this call.

#### Call:

CALL RDMSG (status,filename,maxlen,frameid,maxlen,buffer,maxlen,length)

#### Parameters:

status	A two-word integer field used to return a code indicating the results of the requested operation. Values for status are listed in Section ?
filename	A string specifying the file name and type for the help file. (The directory used for the help file is the directory in which the application resides.)
maxlen	Integer variable or constant. This is the maximum length of the immediately preceding buffer or string parameter. For read-only parameters, this may be the length of the significant data only.
frameid	An eight-byte string containing the key to be used in retrieving the

desired help frame. Frameid may not be blank or of zero length; a valid frameid must be specified.

maxlen Same as previous maxlen.

buffer A string variable of any length. This parameter specifies the value of a field to be filled or returned to pack or unpack a menu buffer.

maxlen Same as previous maxlen.

length An integer variable. This parameter is a returned value indicating how much of the buffer or string parameter was actually used by the operation.

## 13.5.1 Values Returned In Status

The status parameter is used to return error indications to a calling program. The first word of the two-word array indicates the source of the error; the second word indicates the actual error. All values below are decimal.

status value	meaning
-----	-----
+1	No error; for menus, DO was pressed.
+2	Qualified success: for menus, some function key besides DO was pressed; for other operations, a record or field was truncated.
-1	RSX DSW error (DSW in STATUS_+2).
-2	RMS I/O status error (I/O code in STATUS_+2).
-4	P/OS error in HELP.
-5	P/OS error in menus.
-6	I/O error in terminal initialization.
-7	Parsing error.
-8	Process terminated by key other than DO.
-9	Improper parameters.
-10	Insufficient buffer space.
-----	-----

If STATUS is +2 for a menu display operation, then STATUS+2 indicates which function key was pressed as follows. Function key values (decimal) are as follows:

VALUE	KEY
-----	-----
1	HOLD SCREEN
2	PRINT SCREEN
3	BREAK
4	SETUP
5	F5
6	INTERUPT
7	RESUME
8	CANCEL
9	MAIN SCPEEN
10	EXIT
11	F11
12	F12
13	F13
14	ADDITIONAL OPTIONS
15	HELP
16	DO
17	F17
18	F18
19	F19
20	F20
21	FIND
22	INSERT HERE
23	REMOVE
24	SELECT

25	PREV SCREEN
26	NEXT SCREEN
27	cursor up
28	cursor left
29	cursor down
30	cursor right
31	PF1
32	PF2
33	PF3
34	PF4

-----

The following decimal values correspond to keypad keys in application mode.

VALUE	KEY
35	minus
36	comma
37	period
38	ENTER
39	0
40	1
41	2
42	3
43	4
44	5
45	6
46	7
47	8
48	9

-----

If STATUS is -4 or -5, then the P/OS error is indicated in STATUS+2 as follows:

-1	option # greater than maximum
-2	text line # greater than max.
-3	multiple occurrence of once-only field
-4	erroneous frame buffer contents
-5	title field length error
-6	text field length error
-7	frame error
-8	dynamic parse error (unknown field id)
-9	multiple occurrence of once-only fields in dynamic menu frame
-10	unknown escape sequence

25	PREV SCREEN
26	NEXT SCREEN
27	cursor up
28	cursor left
29	cursor down
30	cursor right
31	PF1
32	PF2
33	PF3
34	PF4

---

The following decimal values correspond to keypad keys in application mode.

VALUE	KEY
35	minus
36	comma
37	period
38	ENTER
39	0
40	1
41	2
42	3
43	4
44	5
45	6
46	7
47	8
48	9

---

If STATUS is -4 or -5, then the P/OS error is indicated in STATUS+2 as follows:

-1	option # greater than maximum
-2	text line # greater than max.
-3	multiple occurrence of once-only field
-4	erroneous frame buffer contents
-5	title field length error
-6	text field length error
-7	frame error
-8	dynamic parse error (unknown field id)
-9	multiple occurrence of once-only fields in dynamic menu frame
-10	unknown escape sequence



### 13.6 MISCELLANEOUS SERVICES

#### FATAL ERROR

An application program should call this routine in the event of an unexpected and disabling error condition which prevents the application from continuing.

If this routine is called, line 22 of the screen is blanked and the message "Application error. Press RESUME to return to Main Menu." is displayed on line 23. The message specified in this routine is displayed on line 24.

When the user presses the RESUME key, the application is aborted and the display returns to the main menu. The message string should contain information which will tell the application developer where and why the application failed. The end user can report this information to the Support Services Hotline. This routine should NOT be used to avoid handling user errors in a friendly way.

CALL FATLER (message, msglength)

#### Parameters:

message	string variable or literal, any length
msglength	integer constant or variable

## GET KEYSTROKE

Gets one keystroke. Status indicates the type of key and key name. The keystroke is not echoed on the screen.

CALL GETKEY (status)

status      A two-word integer array.  
Value for status are:  
+1          the key is a single ASCII character  
            status +2 contains the ASCII code  
+2          the key is a function key  
            status +2 contains a code for the  
            function key; see token table above  
< 0        see "Values for STATUS" table in  
            Section ?  
            status=-7, status+2=-11 means that  
            an invalid CSI sequence was found

## PARSE STRING

Parse a string for a CSI sequence. The characters in the parameter buff are scanned from the left until a CSI character is found. The sequence which follows is parsed and translated into a token value indicating what function key terminates the string. The length of the string up to the CSI character is returned in the parameter msglength.

CALL PRSCSI (status, buff, buflen, msglength)

## Parameters:

status	Two-word integer array. Values for status are: +2      a valid CSI sequence was found status +2 contains the token value for that sequence; see the token table above < 0      see values for status table in Section ? status=-7, status_+2=-12 means that no CSI sequence was found
buff	A string variable of any length.
buflen	An integer indicating the length of buff.
msglength	An integer variable.

## NEW FILE

Solicit the name of a new file to be created. The new file specification form is displayed and the user is allowed to enter the name portion of a file specification. The filetype must be a default file type of the form .xxx and this field may not be omitted.

On return of this routine, the filename parameter contains a full DCL file specification, including device and directory. File type is modified to reflect any change made by the user with an Additional Option. See Section ? for possible values for the parameter status.

```
CALL NEWFIL (status,filename,namelen,filetype,typelen,  
.br  
text1,textilen,msg1,msgilen)
```

## Parameters:

status	A two-word integer array.
filename	A string variable up to 50 bytes.
namelen	An integer indicating the length of the filename.
filetype	A string variable of 4 bytes.
typelen	An integer indicating the length of filetype.
text1	A string variable or literal of any length. The string text1 is written at the top of the form.
textilen	An integer indicating the length of text1.
msg1	A string variable or literal of any length. The string is written below the form.
msgilen	An integer indicating the length of msg1.

## OLD FILE NAME

Solicits the names of one or more existing files. The caller specifies a selection criterion for file names from the current directory, the maximum number of choices the operator will be allowed to make, and several pieces of text to be displayed on the menu (text1, msg1, msg2). The routine returns a list of selected file names (filestrbuf) and the number of choices actually made (numchoices). The array (sizearray) contains the length of each file specification in filestrbuf. File names begin every 50 bytes in filestrbuf with blank-fill to the right as needed; the lengths in sizearray indicate the actual length of each name.

CALL OLDFIL (status,numchoices,filestrbuf,sizearray,wildcardspec,  
.br  
wcslen,text1,textilen,msg1,msgilen,msg2,msg2len)

## Parameters:

status	A two-word integer array.
numchoices	An integer variable indicating the maximum number of choices the user may make.
filestrbuf	A string variable listing the selected file names. File names begin every 50 bytes with blank fill to the right as needed.
sizearray	An integer array indicating the actual length of each file specification in filestrbuf.
wildcardspec	A string variable or literal that specifies the file selection criteria.
wcslen	An integer indicating the length of wilocardspec.
text1	A string variable or literal of text to be displayed at the top of the screen.
textilen	An integer indicating the length of text1.
msg1	A string variable or literal of text to be displayed at the bottom of the screen.
msg1len	An integer indicating the length of msg1.
msg2	Same as msg1.
msg2len	An integer indicating the length of msg2.

## CURRENT DATE AND TIME

Return the current date and time. Date is returned in datestr in, on of the following formats:

dd-mmm-yy (example: 08-May-82)

dd/mm/yy (example: 05/12/82)

Time is returned in timestr in one of the following formats:

hh:mm:ss (example: 13:45:50)

hh:mm:ss x.m. (example: 1:45:50 p.m.)

The format is determined by Setup options. These formats are truncated, if necessary, to fit the string variable provided. Extra length in these variables is unmodified, not blank-filled. A string length may be zero if either date or time is not wanted, but all parameters must still be provided.

CALL XTIME (datestr, datelen, timestr, timelen)

## Parameters:

datestr	A string variable.
datelen	An integer indicating the length of datestr.
timestr	A string variable.
timelen	An integer indicating the length of timestr.

## WAIT FOR RESUME KEY

This routine echoes all keystrokes with a bell character except the RESUME key. When the RESUME key is pressed, the routine returns to the caller.

CALL WTRES ( )

## GET USER NAME

Obtains the name with which the user signed on. This name may be up to 40 characters. It is truncated to fit the space provided.

CALL USRNAM (username, namelength)

## Parameters:

username            A string variable.

namelength        An integer indicating the length of username.



## DISPLAY ON MESSAGE BOARD

Put a message on the P/OS message board. This call enters the message into a queue and rings the keyboard bell. The user sees the message when Display message/status is selected from the Main Menu.

CALL MSGBRD (msg, msglen)

## Parameters:

msg            A string variable or literal.

msglen        An integer indicating the length of msg.

## GET CONFIGURATION TABLE

Obtains a copy of the Professional hardware configuration table. This table is built by the Professional diagnostic firmware every time the Professional is powered on. This call copies the last segment of words of this table (segment is determined by tablelength) into the caller's buffer. See ? for a definition of the configuration table.

CALL CONFIG (configtable, tablelength)

## Parameters:

configtable      An integer array.

tablelength      An integer indicating the length of configtable.

## Issues:

1. where is the configuration table defined?



SUMMARY OF ENTRY POINTS TO P/OS

APPENDIX A

SUMMARY OF ENTRY POINTS TO P/OS

A.1 SUMMARY OF ENTRY POINTS

To be supplied.

A.1.1 File Services

To be supplied.

A.1.2 Disk Services

To be supplied

A.1.3 Communications

To be supplied.

A.1.4 Editor

To be supplied.

A.1.5 User Interface

To be supplied.

A.1.6 Miscellaneous System Services

To be supplied.

# APPENDIX B

## THE DEC MULTINATIONAL CHARACTER SET

Decimal Code	8 -Bit Octal Code	Character	Remarks
0	000	nul.	Null, tape feed, shift, 'P
1	001	SOH	Start of heading, start of message, 'A
2	002	STX	Start of text, end of address, 'B
3	003	ETX	End of text, end of message, 'C
4	004	EOT	End of transmission, shuts off TWX machine, 'D
5	005	ENQ	Enquiry, WRU, 'E
6	006	ACK	Acknowledge, RU, 'F
7	007	BEL	Bell, 'G
8	010	BS	Backspace, format effector, 'H
9	011	HT	Horizontal tab, 'I
10	012	LF	Line feed, 'J
11	013	VT	Vertical tab, 'K
12	014	FF	Form feed, page, 'L
13	015	CR	Carriage return, 'M
14	016	SO	Shift out, 'N
15	017	SI	Shift in, 'O
16	020	DLE	Data link escape, 'P
17	021	DC1	Device control 1, 'Q
18	022	DC2	Device control 2, 'R
19	023	DC3	Device control 3, 'S
20	024	DC4	Device control 4, 'T

(continued on next page)

Decimal Code	8-Bit Octal Code	Character	Remarks
21	025	NAK	Negative acknowledge, ERR. 'U
22	026	SYN	Synchronous idle, 'V
23	027	ETB	End-of-transmission block, logical end of medium, 'W
24	030	CAN	Cancel, 'X
25	031	EM	End of medium, 'Y
26	032	SUB	Substitute, 'Z
27	033	ESC	Escape, prefix, shift, 'K
28	034	FS	File separator, shift, 'L
29	035	GS	Group separator, shift, 'M
30	036	RS	Record separator, shift, 'N
31	037	US	Unit separator, shift, 'O
32	040	SP	Space
33	041	!	Exclamation point
34	042	"	Double quotation mark
35	043	#	Number sign
36	044	\$	Dollar sign
37	045	%	Percent sign
38	046	&	Ampersand
39	047	'	Apostrophe
40	050	(	Left parenthesis
41	051	)	Right parenthesis
42	052	*	Asterisk
43	053	+	Plus sign
44	054	,	Comma
45	055	-	Minus sign, hyphen
46	056	.	Period, dot
47	057	/	Slash, statement separator
48	060	0	Zero
49	061	1	One
50	062	2	Two
51	063	3	Three
52	064	4	Four
53	065	5	Five
54	066	6	Six
55	067	7	Seven
56	070	8	Eight
57	071	9	Nine
58	072	:	Colon
59	073	;	Semicolon
60	074	<	Left angle bracket
61	075	=	Equal sign
62	076	>	Right angle bracket
63	077	?	Question mark
64	100	@	At sign
65	101	A	Upper-case A
66	102	B	Upper-case B
67	103	C	Upper-case C

(continued on next page)

Decimal Code	8-Bit Octal Code	Character	Remarks
68	104	D	Upper-case D
69	105	E	Upper-case E
70	106	F	Upper-case F
71	107	G	Upper-case G
72	110	H	Upper-case H
73	111	I	Upper-case I
74	112	J	Upper-case J
75	113	K	Upper-case K
76	114	L	Upper-case L
77	115	M	Upper-case M
78	116	N	Upper-case N
79	117	O	Upper-case O
80	120	P	Upper-case P
81	121	Q	Upper-case Q
82	122	R	Upper-case R
83	123	S	Upper-case S
84	124	T	Upper-case T
85	125	U	Upper-case U
86	126	V	Upper-case V
87	127	W	Upper-case W
88	130	X	Upper-case X
89	131	Y	Upper-case Y
90	132	Z	Upper-case Z
91	133	[	Left bracket, shift K
92	134	\	Backslash, shift L
93	135	]	Right bracket, shift M
94	136	^	Caret, circumflex
95	137	_	Underscore
96	140	`	Accent grave
97	141	a	Lower-case a
98	142	b	Lower-case b
99	143	c	Lower-case c
100	144	d	Lower-case d
101	145	e	Lower-case e
102	146	f	Lower-case f
103	147	g	Lower-case g
104	150	h	Lower-case h
105	151	i	Lower-case i
106	152	j	Lower-case j
107	153	k	Lower-case k
108	154	l	Lower-case l
109	155	m	Lower-case m
110	156	n	Lower-case n
111	157	o	Lower-case o
112	160	p	Lower-case p
113	161	q	Lower-case q
114	162	r	Lower-case r
115	163	s	Lower-case s

(continued on next page)



Decimal Code	8 -Bit Octal Code	Character	Remarks
116	164	t	Lower-case t
117	165	u	Lower-case u
118	166	v	Lower-case v
119	167	w	Lower-case w
120	170	x	Lower-case x
121	171	y	Lower-case y
122	172	z	Lower-case z
123	173	{	Left brace
124	174		Vertical line
125	175	}	Right brace
126	176	-	Tilde
127	177	DEL	Delete, rubout

## DEC SUPPLEMENTAL GRAPHIC SET

Decimal Code	8-Bit Octal code	Character	Remarks
128	200		Reserved
129	201		Reserved
130	202		Reserved
131	203		Reserved
132	204	IND	Index
133	205	NEL	Next line
134	206	SSA	Start of selected area
135	207	ESA	End of selected area
136	210	HTS	Horizontal tab set
137	211	HTJ	Horizontal tab with justify
138	212	VTs	Vertical tab set
139	213	PLD	Partial line down
140	214	PLU	Partial line up
141	215	RI	Reverse index
142	216	SS2	Single shift two
143	217	SS3	Single shift three
144	220	DCS	Device control string
145	221	PU1	Private use one
146	222	PU2	Private use two
147	223	STS	Set transmit state
148	224	CCH	Cancel character
149	225	MW	Message waiting
150	226	SPA	Start of protected area
151	227	EPA	End of protected area
152	230		Reserved
153	231		Reserved
154	232		Reserved
155	233	CSI	Control sequence introducer

Decimal Code	8-bit Octal code	Character	Remarks
156	234	ST	String terminator
157	235	OSC	Operating system command
158	236	PM	Privacy message
159	237	APC	Application program command
160	240		Reserved
161	241	!	Inverted exclamation mark
162	242	¢	Cent sign
163	243	£	Pound sign
164	244		Reserved
165	245	¥	Yen sign
166	246		Reserved
167	247	§	Section sign
168	250	¤	General currency sign
169	251	©	Copyright sign
170	252	₊	Feminine ordinal indicator
171	253	«	Angle quotation mark left
172	254		Reserved
173	255		Reserved
174	256		Reserved
175	257		Reserved
176	260	°	Degree sign
177	261	±	Plus/minus sign
178	262	²	Superscript 2
179	263	³	Superscript 3
180	264		Reserved
181	265	μ	Micro sign
182	266	¶	Pilcrow, paragraph sign
183	267	.	Middle dot

Decimal Code	8-Bit Octal code	Character	Remarks
184	270		Reserved
185	271	¹	Superscript 1
186	272	º	Masculine ordinal indicator
187	273	»	Angle quotation mark right
188	274	¼	Fraction one quarter
189	275	½	Fraction one half
190	276		Reserved
191	277	¿	Inverted question mark
192	300	À	Capital A with grave accent
193	301	Á	Capital A with acute accent
194	302	Â	Capital A with circumflex accent
195	303	Ã	Capital A with tilde
196	304	Ä	Capital A with diaeresis or umlaut mark
197	305	Å	Capital A with ring
198	306	Æ	Capital AE diphthong
199	307	Ç	Capital C with cedilla
200	310	È	Capital E with grave accent
201	311	É	Capital E with acute accent
202	312	Ê	Capital E with circumflex accent
203	313	Ë	Capital E with diaeresis or umlaut mark
204	314	Ì	Capital I with grave accent
205	315	Í	Capital I with acute accent
206	316	Î	Capital I with circumflex accent
207	317	Ï	Capital I with diaeresis or umlaut mark
208	320		Reserved
209	321	Ñ	Capital N with tilde
210	322	Ò	Capital O with grave accent

Decimal Code	8-Bit Octal code	Character	Remarks
211	323	Ō	Capital O with acute accent
212	324	Ô	Capital O with circumflex accen
213	325	Õ	Capital O with tilde
214	326	Ö	Capital O with diaeresis or umlaut mark
215	327	Œ	Capital OE ligature
216	330	Ø	Capital O with slash
217	331	Ù	Capital U with grave accent
218	332	Ú	Capital U with acute accent
219	333	Û	Capital U with circumflex accen
220	334	Ü	Capital U with diaeresis or umlaut mark
221	335	Ÿ	Capital Y with diaeresis or umlaut mark
222	336		Reserved
223	337	ß	German small sharp s

Decimal Code	8-bit Octal Code	Character	Remarks
224	340	à	Small a with grave accent
225	341	á	Small a with acute accent
226	342	â	Small a with circumflex accent
227	343	ã	Small a with tilde
228	344	ä	Small a with diaeresis or umlaut mark
229	345	å	Small a with ring
230	346	æ	Small ae diphthong
231	347	ç	Small c with cedilla
232	350	è	Small e with grave accent
233	351	é	Small e with acute accent
234	352	ê	Small e with circumflex accent
235	353	ë	Small e with diaeresis or umlaut mark
236	354	ì	Small i with grave accent
237	355	í	Small i with acute accent
238	356	î	Small i with circumflex accent
239	357	ï	Small i with diaeresis
240	360		Reserved for future use
241	361	ñ	Small n with tilde
242	362	ò	Small o with grave accent
243	363	ó	Small o with acute accent
244	364	ô	Small o with circumflex accent
245	365	õ	Small o with tilde
246	366	ö	Small o with diaeresis or umlaut mark
247	367	oe	Small oe ligature
248	370	ø	Small o with slash

Decimal Code	8-bit Octal Code	Character	Remarks
249	371	ü	Small u with grave accent
250	372	ú	Small u with acute accent
251	373	û	Small u with circumflex accent
252	374	ü	Small u with diaeresis or umlaut mark
253	375	ÿ	Small y with diaeresis or umlaut mark
254	376		Reserved for future use
255	377		Reserved for future use

## TOOL KIT GLOSSARY

## Application Developer

A firm or individual using the Professional Developer's Tool Kit to develop application programs for P/OS. The end user installs application software on P/OS.

## Application Diskette Builder

The Tool Kit software component that transforms the task image of an application into a P/OS application on a diskette. The program runs on the Professional.

## Application Program

A computer program that meets some specific user needs -- for example, to control an inventory or to monitor a manufacturing process. The term "application program" is often shortened to "application." A Professional owner can buy a variety of applications.

Some classes of applications have generic names in the computer industry. For example, an editor is an application that creates and edits text.

## PRO/BASIC-PLUS-2

A programming language used to develop application software for P/OS. It is provided with the Professional Developer's Tool Kit.

## Character

A member of a set of elements used to organize, control, or represent text. Characters used in computers are graphic characters and control characters. The Professional terminal uses the DEC Multinational coded character set having an 8-bit code representation. This character set includes:

- o Control characters producing a function that initiates, modifies, or stops the recording, processing, transmission, or interpretation of data.
- o Graphic characters, which do not include control characters, having a visual representation normally handwritten, printed, or displayed.

## End User

An ultimate user of a Professional -- for example, those who are self-employed, managers, technicians, and clerks.

## PRO/FMS-11 (Forms Management System)

PRO/FMS-11 creates forms for displays for interactive data gathering



