

Tool Kit Reference Manual

Order No. AA-BT74B-TH

November 1985

This manual provides reference information for some of the components of the Tool Kit.

REQUIRED SOFTWARE: Host Tool Kit V3.0
or PRO/Tool Kit V3.0

OPERATING SYSTEM: P/OS V3.0



DIGITAL EQUIPMENT CORPORATION
Maynard, Massachusetts 01754-2571

First Printing, April 1984
Updated, July 1985
Revised, November 1985

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 © 1985 by Digital Equipment Corporation
All Rights Reserved

The following are trademarks of Digital Equipment Corporation:

CTI BUS	MASSBUS	Rainbow
DEC	PDP	RSTS
DECmate	P/OS	RSX
DECsystem-10	PRO/BASIC	Tool Kit
DECSYSTEM-20	PRO/Communications	UNIBUS
DECUS	Professional	VAX
DECwriter	PRO/FMS	VMS
DIBOL	PRO/RMS	VT
digital ™	PROSE	Work Processor
	PROSE PLUS	

CONTENTS

PREFACE	ix
CHAPTER 1	APPLICATION DISKETTE BUILDER (ADB)	
1.1	USING ADB	1-1
1.2	THE ADB DIALOGUE	1-2
1.2.1	Step 1: Choose Target System	1-2
1.2.2	Step 2: Choose Installation Command File	1-3
1.2.3	Step 3: Ready Target Diskette	1-3
1.2.4	Step 4: Copy Installation Command File to First Diskette	1-4
1.2.5	Step 5: Copy Files to Target Diskette	1-4
1.2.6	Step 6: Label Completed Diskette	1-4
CHAPTER 2	COMMUNICATIONS SERVICES	
2.1	OVERVIEW OF COMMUNICATIONS SERVICES	2-2
2.1.1	Status Returns	2-2
2.1.2	Line Descriptor Block	2-4
2.1.3	Autodial Modem Support	2-8
2.2	BASE SYSTEM SERVICES	2-9
2.2.1	Base System Start-Up	2-9
2.2.2	Base System Routines	2-10
2.2.2.1	Attach Line (CCATT and CCATA)	2-10
2.2.2.2	Detach Line (CCDET)	2-11
2.2.2.3	Set Line Characteristics (CCSMC)	2-11
2.2.2.4	Get Line Characteristics (CCGMC)	2-12
2.2.2.5	Get/Put Line Configuration Record (CCLCRG and CCLCRP)	2-13
2.2.2.6	Set Translate Table (CCMTT)	2-14
2.2.2.7	Dial Call (CCDIAL)	2-15
2.2.2.8	Answer Call (CCANS)	2-18
2.2.2.9	Originate Call (CCORG)	2-19
2.2.2.10	Hang Up a Call (CCHNG)	2-21
2.2.2.11	Transmit Data (CCTXD)	2-21
2.2.2.12	Receive Data (CCRXD)	2-22
2.2.2.13	Flush Input Buffer (CCFLSH)	2-23
2.2.2.14	Generate Break (CCBRK)	2-24
2.2.2.15	Kill Transfer (CCKILL)	2-25
2.3	PRO/COMMUNICATIONS SERVICES	2-25
2.3.1	PRO/Communications Start-Up	2-26
2.3.2	PRO/Communications Utilities	2-26
2.3.2.1	Spawn Communications Utility (CCSPWN)	2-27
2.3.3	Phone Book Services	2-28
2.3.3.1	Get Phone Book Record (CPHREC)	2-28
2.3.3.2	Select Phone Book Entry (CPHSEL)	2-29

2.3.4	File Transfer Services	2-30
2.3.4.1	File Transfer Initiation Protocol	2-31
2.3.4.2	Attaching the File Transfer Subsystem	2-31
2.3.4.3	Releasing the File Transfer Subsystem	2-32
2.3.4.4	Synchronous and Asynchronous Operations	2-32
2.3.4.5	File Transfer Synchronization	2-32
2.3.4.6	Set Up File Transfer Options (FTOPTG and FTOFTP)	2-33
2.3.4.7	Attach File Transfer Subsystem (FTATT)	2-35
2.3.4.8	File Transfer Operation Request (FTOPRN)	2-36
2.3.4.9	Notify On Incoming File (FTNTFY)	2-38
2.3.4.10	Synchronize with File Transfer (FTSYNC)	2-40
2.3.4.11	Unpack File Transfer Message Area (FTUNPK)	2-41
2.3.4.12	Detach File Transfer (FTDET)	2-42
2.3.4.13	Start File Transfer Server (FTSERV)	2-43
2.3.4.14	Enable/Disable File Listening (FTLISN)	2-44
2.4	TELEPHONE MANAGEMENT SYSTEM (TMS) SERVICES	2-45
2.4.1	TMS Start-Up	2-45
2.4.2	TMS Routines	2-45
2.4.2.1	Change Mode (CCMODE)	2-45
2.4.2.2	Auxiliary Keyboard Enable/Disable (CCAUXK)	2-46
2.4.2.3	Prepare to Go Voice (CCPTGV)	2-47
2.4.2.4	Set DTMF Escape Sequence (CCDTMF)	2-48

CHAPTER 3 FAST INSTALL

3.1	INSTALLING DECNET OBJECTS FOR TESTING	3-3
-----	---	-----

CHAPTER 4 FILE CONTROL SERVICES (FCS)

4.1	SUPPORT BY HIGH-LEVEL LANGUAGES	4-1
4.2	USING FCS RESIDENT LIBRARY	4-2
4.3	USING FCS IN-TASK VERSION	4-2

CHAPTER 5 FRAME DEVELOPMENT TOOL (FDT)

5.1	FDT OVERVIEW	5-1
5.2	INVOKING FDT	5-2
5.2.1	Invoking FDT on VAX/VMS	5-2
5.2.2	Invoking FDT on RSX-11M/M-PLUS (DCL)	5-3
5.2.3	Invoking FDT in PRO/Tool Kit	5-3
5.3	FILE COMMANDS	5-3
5.3.1	ADD	5-4
5.3.2	CONVERT	5-4
5.3.3	DELETE	5-5
5.3.4	EXIT	5-5

5.3.5	FILE	5-6
5.3.6	HELP	5-6
5.3.7	LIST	5-7
5.3.8	MODIFY	5-7
5.3.9	NAME	5-7
5.3.10	QUIT	5-8
5.3.11	REPORT	5-8
5.3.12	SAVE	5-8
5.3.13	WINDOW	5-9
5.4	FRAME COMMANDS	5-10
5.4.1	ACTION	5-10
5.4.2	DISPLAY	5-11
5.4.3	EXIT	5-11
5.4.4	HELP	5-12
5.4.5	PROFILE	5-12
5.4.6	QUIT	5-12
5.4.7	SAVE	5-13
5.5	PROFILE, DISPLAY, AND ACTION FORMS	5-13
5.6	CREATING A SINGLE-CHOICE MENU	5-18
5.6.1	The Profile Form	5-18
5.6.2	The Display Form	5-20
5.6.3	The Action Form	5-21
5.7	CREATING A HELP MENU	5-23
5.7.1	The Profile Form	5-23
5.7.2	The Display Form	5-24
5.7.3	The Action Form	5-25
5.8	CREATING A HELP TEXT FRAME	5-26
5.8.1	The Profile Form	5-26
5.8.2	The Display Form	5-27
5.9	CREATING A MESSAGE TEXT FRAME	5-28
5.9.1	The Profile Form	5-28
5.9.2	The Display Form	5-28
5.10	RESOLVING ERRORS	5-29
5.11	SAMPLE TERMINAL SESSION	5-29

CHAPTER 6 INSTALLATION COMMAND LANGUAGE

6.1	INSTALLATION COMMAND FILE FORMAT	6-2
6.1.1	Using .INB and .INS Files	6-3
6.1.2	Using /USER, /NETWORK, and /CLUSTER in Your .INB File	6-3
6.2	INSTALLATION FILE ERRORS	6-7
6.3	ASSIGN HELP	6-8
6.4	ASSIGN LOGICAL	6-9
6.5	ASSIGN MENU	6-10
6.6	ASSIGN MESSAGE (.INB FILE ONLY)	6-11
6.7	COMMENT	6-12
6.8	EXECUTE	6-12
6.9	FILE	6-19
6.10	INSTALL	6-21

6.11	MOUNT	6-23
6.12	NAME	6-24
6.13	RUN	6-25
6.14	IMPROVING APPLICATION PERFORMANCE	6-26
6.15	REQUIRED COMMANDS FOR APPLICATIONS	6-26

CHAPTER 7 MACRO-11 ASSEMBLER (PMA)

7.1	INVOKING PMA ON THE PRO/TOOL KIT	7-1
7.2	INVOKING PMA ON RSX-11M/M-PLUS	7-1
7.3	INVOKING PMA ON VAX/VMS	7-2

CHAPTER 8 POSRES USER INTERFACE LIBRARY ROUTINES

8.1	NOTES ON USING POSRES ROUTINES	8-1
8.2	DMENU - DISPLAY DYNAMIC MENU	8-3
8.3	DPACK - PACK DYNAMIC SINGLE CHOICE MENU	8-5
8.4	FATLER - FATAL ERROR	8-8
8.5	GETKEY - GET KEYSTROKE	8-9
8.6	HCLOSE - CLOSE HELP FILE	8-10
8.7	HELP - DISPLAY HELP FRAME	8-11
8.8	HFILE - OPEN HELP FILE	8-12
8.9	HFRAME - SPECIFY HELP FRAME	8-14
8.10	MCLOSE - CLOSE MENU FILE	8-15
8.11	MENU - DISPLAY SINGLE-CHOICE MENU	8-16
8.12	MFILE - OPEN MENU FILE	8-18
8.13	MFRAME - READ MENU FRAME	8-20
8.14	MMENU - DISPLAY MULTIPLE-CHOICE MENU	8-21
8.15	MPACK - PACK MULTIPLE-CHOICE MENU	8-23
8.16	MSGBRD - SEND MESSAGE TO A MESSAGE/STATUS DISPLAY	8-25
8.17	MUNPK - UNPACK MENU BUFFER	8-26
8.18	NEWFIL - NEW FILE	8-28
8.19	OLDFIL - OLD FILENAME	8-30
8.20	PRSCSI - PARSE STRING	8-32
8.21	RDMSG - READ MESSAGE	8-33
8.22	WTRES - WAIT FOR RESUME KEY	8-35

CHAPTER 9 CALLABLE PRINT SERVICES

9.1	CPRNT	9-1
9.2	CPRNV3	9-4
9.3	COMPATIBILITY BETWEEN CPRNT AND CPRNV3	9-8
9.4	EXAMPLE	9-8

CHAPTER 10 **PROSE TEXT EDITOR**

CHAPTER 11 **PRO/SORT**

11.1	USING PRO/SORT	11-1
11.2	VALID RMS RECORD FORMATS	11-2
11.3	PRO/SORT COMMANDS AND COMMAND FILE	11-3
11.4	COLLATE	11-4
11.4.1	COLLATE Compared to SORT ALTSEQ	11-5
11.5	DEFAULT	11-5
11.6	FIELD	11-7
11.6.1	Pseudo-Fields RRN and RFA	11-9
11.7	FORCE	11-10
11.7.1	FORCE Compared to SORT F	11-10
11.8	INCLUDE	11-11
11.8.1	INCLUDE Compared to SORT O and I	11-12
11.9	INPUT AND OUTPUT	11-13
11.10	PROCESS	11-13
11.11	SORT	11-13
11.11.1	SORT Compared to SORT N and O	11-14
11.12	WRITE	11-15
11.12.1	WRITE Compared to SORT D	11-15
11.13	PRO/SORT ERROR CODES	11-15
11.14	PRO/SORT EXAMPLE	11-18

APPENDIX A **APPLICATION DISKETTE BUILDER ERROR MESSAGES**

A.1	ADB NORMAL ERRORS	A-2
A.2	ADB SERIOUS ERRORS	A-7

APPENDIX B **FDT ERROR MESSAGES**

B.1	USER ERRORS	B-2
B.2	INTERNAL ERRORS	B-9

APPENDIX C **POSRES STATUS BLOCK CODES**

APPENDIX D **FUNCTION KEY NAMES AND CODES**

APPENDIX E **P/OS ERROR CODES**

E.1	APPLICATION CANNOT BE STARTED	E-1
E.2	SYSTEM ABORTED TASK	E-3

INDEX

FIGURES

5-1	Forms for a Single-Choice Menu	5-15
5-2	FDT Screen Editor Keypad	5-16
5-3	Profile Form for Single-Choice Menu	5-18
5-4	Display Form for Single-Choice Menu	5-20
5-5	Action Form for Single-Choice Menu	5-21
5-6	Profile Form for Help Menu	5-23
5-7	Display Form for Help Menu	5-24
5-8	Action Form for Help Menu	5-25
5-9	Profile Form for Help Text Frame	5-26
5-10	Display Form for Help Text Frame	5-27
5-11	Profile Form for Message Frame	5-28
6-1	Installation Command File Format	6-2
9-1	Sample Call to CPRNV3	9-9
11-1	Mapping of Bytes to Fields in Input Data Stream	11-8
11-2	Sample PRO/SORT Command File	11-18
11-3	Sample Input File INPUT.DAT	11-19
11-4	Sample Indirect Command File FIELDS.CMD	11-19
11-5	Resulting Output File OUTPUT.DAT	11-21

TABLES

1-1	Application File Locations for ADB	1-2
2-1	Communications Status Return Codes	2-3
2-2	Characteristic Positions in the LDB	2-5
2-3	Transmit/Receive Rate Values for LDB	2-6
3-1	Application File Locations for Fast Install	3-2
6-1	Summary of /USER, /NETWORK, and /CLUSTER Qualifiers	6-6
6-2	EXECUTE Qualifiers and Phases of Installation/Removal	6-17
6-3	Required Commands	6-27
9-1	CPRNT Condition Codes	9-2
9-2	CPRNT Error Codes	9-3
9-3	Status Returns for CPRNV3	9-6
10-1	Callable Editor - Status Return Codes	10-3
11-1	Number of PRO/SORT Commands per Command File	11-3
11-2	PRO/SORT Error Codes	11-16
C-1	POSRES Status Values	C-1
C-2	Menu Service Routine Errors	C-4
D-1	Function Key Names and Codes	D-1
E-1	Elements of the Status Array	E-1
E-2	Elements of Status Array	E-3

PREFACE

Manual Objectives

After reading this manual, you will be able to use several important tools contained in the Tool Kit. Note that some tools described in this manual have their primary documentation in other manuals; this manual supplements the primary material.

Intended Audience

This document is intended for programmers familiar with the Tool Kit development cycle and at least one Tool Kit language. Read the *Tool Kit User's Guide* before using this manual.

Structure of This Document

This manual describes 11 tools, one per chapter:

- **Chapter 1:** Application Diskette Builder (ADB)
- **Chapter 2:** Communications Services
- **Chapter 3:** Fast Install
- **Chapter 4:** File Control Services (FCS)
- **Chapter 5:** Frame Development Tool (FDT)
- **Chapter 6:** Installation Command Language
- **Chapter 7:** MACRO-11 Assembler (PMA)
- **Chapter 8:** POSRES User Interface Library Routines
- **Chapter 9:** Print Services
- **Chapter 10:** PROSE Text Editor
- **Chapter 11:** PRO/SORT

Appendices A and B are devoted to error messages returned by ADB and FDT. Appendices C and D provide tabular information related to POSRES user interface programming. Appendix E describes P/OS error codes.

Preface

Associated Documents

Throughout this manual, we refer to related documentation within the Tool Kit document set. For abstracts of other Tool Kit documents, see the *Tool Kit User's Guide*.

Conventions Used in This Document

Convention	Meaning
BOLD UPPERCASE	Bold uppercase letters indicate elements that you must use exactly as shown.
UPPERCASE LETTERS	Uppercase letters indicate elements that you can omit or use exactly as shown.
bold lowercase	Bold lowercase letters indicate elements that you must replace according to the description in the text.
lowercase letters	Lowercase letters indicate elements that you can omit or replace according to the description in the text.
bold shaded areas	Shaded areas containing bold letters indicate a list of elements from which you must select one.
shaded areas	Shaded areas containing non-bold letters indicate a list of elements from which you can optionally select one.
element,...	A comma followed by a horizontal ellipsis indicates that you can repeat the preceding element one or more times, separating the elements with commas.
.	A vertical ellipsis in a figure or example means that part has been omitted for brevity.
red letters	Red letters indicate your input to the computer.

CHAPTER 1

APPLICATION DISKETTE BUILDER (ADB)

The Application Diskette Builder (ADB) creates a master copy of an application diskette that can be reproduced for distribution. ADB is distributed with the Host Tool Kit and the PRO/Tool Kit on the Application Diskette Builder diskette.

ADB uses the information in your installation command file to copy your application from the hard disk to one or more diskettes, from which it can be installed on other systems.

An application can span several diskettes. Multiple applications can be copied to a single diskette if space allows.

1.1 USING ADB

To prepare a master distribution diskette for your application, follow these steps:

1. If you have not already installed ADB on your Professional, do so now. Insert the ADB diskette and install it as you would any application.
2. Copy your application files to the appropriate device and directory as shown in Table 1-1. The table shows how to use the qualifiers you have specified on FILE commands in your installation file to determine the destination of your files.
3. Select ADB from the menu on which it was installed. The ADB dialogue is described in Section 1.2. During the dialogue, you can press MAIN SCREEN to return to the Main Menu.

USING ADB

Table 1-1: Application File Locations for ADB

FILE Command	Copy File From
FILE fname.ext or FILE fname.ext/USER	SY000:[applicdir] where applicdir is the name of the directory containing the installation command file.
FILE [dir]fname.ext or FILE [dir]fname.ext/USER	SY000:[dir] where dir is the name of the directory that you specified in the FILE command in your installation command file.
FILE fname.ext/NETWORK	SY000:[applicdir] where applicdir is the name of the directory containing the installation command file.
FILE [dir]fname.ext/CLUSTER	LB000:[dir] where dir is the name of the directory that you specified in the FILE command in your installation command file.
FILE fname.ext/CLUSTER	SY000:[applicdir] where applicdir is the name of the directory containing the installation command file.

1.2 THE ADB DIALOGUE

ADB first displays an introductory frame. Press DO to begin the dialogue.

1.2.1 Step 1: Choose Target System

ADB displays the menu for choosing the target system for your application. Choose the appropriate target system for your application.

THE ADB DIALOGUE

1.2.2 Step 2: Choose Installation Command File

ADB displays a list of installation files contained in your current directory. If the desired installation command file is not on this list, press the ADDTNL OPTIONS key to:

- Select a file from a different directory
- Select a file from a different volume and directory
- View the next group of files
- Specify the file with an extended filename

If ADB does not find an installation command file in the current directory, it displays the Additional Options menu.

When you have selected a file, ADB checks it for errors. If ADB detects an error, refer to Appendix A, correct the problem, and try again.

1.2.3 Step 3: Ready Target Diskette

If ADB finds no errors, it displays the Diskette Drive Selection and Initialization Menu.

- If the diskette contains information that must be preserved, select one of the options that indicates copying will take place without initializing the diskette. Existing information will be saved.

NOTE

If the installation command file contains a MOUNT line for this diskette, ADB will expect the diskette to have the same volume name as that specified in the MOUNT line.

- If you do not want to preserve any existing information on the target diskette, select an option that will initialize the diskette. When ADB prompts for the volume name, enter the desired volume name. ADB will check the volume for bad blocks.

THE ADB DIALOGUE

NOTE

If your installation command file references this diskette in a MOUNT line, you will not be prompted for a volume name.

1.2.4 Step 4: Copy Installation Command File to First Diskette

ADB displays a form asking you for the name of the application directory to be created on the application diskette. If an installation command file contains errors, ADB will not copy the file and will display an error message. Check the format of the installation command file, correct the file, and start again.

1.2.5 Step 5: Copy Files to Target Diskette

The ADB now copies the files listed in the FILE lines of the installation command file to the target diskette. If ADB encounters a MOUNT line, it prompts you to insert a new diskette and repeats the diskette initialization process. ADB then resumes copying files with the next FILE line and continues until all MOUNT and FILE lines have been processed.

ADB searches for the files to copy in the locations shown in Table 1-1. If ADB cannot find a file, it displays a Directory Selection Menu. Specify the directory that contains the file.

ADB displays a confirmation message when all files are copied.

1.2.6 Step 6: Label Completed Diskette

You should label your application diskettes with:

- The application's name
- The volume name, which must match the name provided in the corresponding MOUNT line, if there is one.
- The number of diskettes on which the application resides

THE ADB DIALOGUE

- Minimum hardware configuration required
- Media space required

If your application spans more than one diskette, you should also indicate which is the first diskette; that is, indicate which diskette contains the installation command file.

CHAPTER 2

COMMUNICATIONS SERVICES

Communications Services allow you to perform communications operations on the Professional. Your application program can access the services either by calling communications routines, or by spawning tasks that execute communications utility programs. For example, your program can call a routine named CCATT to attach a phone line, or it can spawn a task that executes a terminal emulation utility.

The communications services fall into the following categories:

1. **Base System Services**, which DIGITAL provides with the P/OS operating system. These services include an asynchronous driver (XKDRV) for the Communication Port as well as a communications service library (COMLIB).
2. **PRO/Communications Services**, which DIGITAL supplies as an optional application, providing features beyond those supplied with the base system. These services include utility programs (such as a terminal emulator), as well as additional communications routines.
3. **Telephone Management System (TMS) Services**, which DIGITAL supplies as an optional application. The TMS services consist of a set of routines that allow your application to control the TMS hardware. Note that this hardware must be installed on the Professional in order to call the TMS routines.

The next section presents general information common to all the communications services. Succeeding sections describe each of the services.

OVERVIEW OF COMMUNICATIONS SERVICES

2.1 OVERVIEW OF COMMUNICATIONS SERVICES

Before executing any communications service that performs I/O, you must assign a logical unit number (LUN) to the required communication line(s). The device names for the communication lines are:

<u>Device Name</u>	<u>Represents</u>
XK0:	Communication Port
XT1:	TMS Line 1
XT2:	TMS Line 2
XT3:	TMS Voice Unit

You can assign a LUN to a communication line either as a static assignment at task build time, or by using the ALUN\$ system directive (described in the *P/OS System Reference Manual*). The system automatically incorporates the communications impure data area in the root of any task linking to the COMLIB cluster library.

If you want to service unsolicited events on the communication line, you can call the CCATA routine to attach the line. You are notified of all unsolicited events via the user asynchronous system trap (AST) routine declared in the call.

You can perform either synchronous or asynchronous I/O to the communication line driver. The event flag number (EFN)--a parameter on your call line--indicates the type of I/O performed. For asynchronous I/O, the system sets the event flag on completion of the I/O transfer.

Parameters in calls to the communications routines are all one-word integers, unless the description of a particular parameter specifies otherwise. For details on a routine's parameters, refer to the routine's description later in this chapter.

For a description of the calling conventions for all the routines, refer to the *Tool Kit User's Guide*.

2.1.1 Status Returns

All communications routines check for proper I/O termination, returning a two-word status value in the first parameter in the call line.

OVERVIEW OF COMMUNICATIONS SERVICES

The first word of the status value contains a code indicating whether or not the I/O termination was successful. The second status word contains a code indicating the specific error, if an error occurred. The routine obtains this code from the first word of the I/O status block (IOSB). You can check the first byte of this word to determine the specific error.

For routines that accept calls issuing an asynchronous QIO, your application must specify an additional two-word area for the IOSB. If you use the routine in its synchronous mode, the routine signals its termination status as described above.

However, if you use the routine's asynchronous mode, the first status word indicates only whether or not the routine has successfully issued the QIO. Your program must test the IOSB to determine if the I/O terminated successfully after the QIO completed. For some routines, the system signals QIO completion by setting an event flag that you specify in the call line.

COMLIB defines all returned status codes as global symbolic constants. For those languages that allow you to define external constants, the Professional Application Builder (PAB) resolves the values at build time.

Each status return has a symbolic name, called the *return code*. Table 2-1 shows the return codes and their values. Success returns are positive values; error returns are negative values.

For the possible return codes for a particular routine, see that routine's description later in this chapter.

Table 2-1: Communications Status Return Codes

Return Code	Value	Meaning
CS.SUC	1	Successful reply
CS.NTB	2	Successful, no translate table (CCLCRG, CCLCRP routines only)
CS.FUN	2	Successful, function key exit Second word contains key value
CE.UNO	-1	Unsupported message option
CE.MFE	-2	Message format error

OVERVIEW OF COMMUNICATIONS SERVICES

Return Code	Value	Meaning
CE.REJ	-3	Invalid message option Second word contains:
C2.DIS		1 - Operation disallowed
C2.UAB		2 - User requested abort
C2.FTB		3 - File exceeds maximum
C2.IAC		4 - Invalid action code
C2.BSY		5 - File transfer Subsystem busy
C2.BPL		6 - Bad parameter
C2.IDL		7 - No operation currently active
C2.IPW		8. - Invalid password
C2.SAA		9. - Server already active
C2.NAE		64. - File Transfer Subsystem unattached
CE.MSC	-4	Message type not synchronized
CE.DDC	-5	Data link error
CE.IEA	-6	Internal error abort
CE.RMS	-7	PRO/RMS error Second word contains the PRO/RMS error code
CE.LIB	-9.	Library mismatch error
CE.STU	-11.	File transfer startup in progress
CE.PRM	-16.	External parameter error
CE.DIR	-17.	RSX Directive error Second word contains the DSW
CE.IER	-18.	I/O termination error
CE.NNF	-19.	Name not found in Phone Book
CE.CTB	-20.	Phone Book is empty

2.1.2 Line Descriptor Block

The *line descriptor block* (LDB) stores line characteristics, which describe how voice or data travels over your communication line. You need this information to set up the line, either for a voice or data connection. You can transmit via the Communication Port or Telephone Management System (TMS) telephone lines.

OVERVIEW OF COMMUNICATIONS SERVICES

(We describe TMS later in this chapter.)

Each line characteristic consists of one byte, stored in the LDB in a position-dependent manner. That is, each value's position in the LDB indicates what characteristic it represents. Table 2-2 shows the characteristic positions in the LDB.

Table 2-2: Characteristic Positions in the LDB

Byte	Description
0	Transmit Rate (Bits per sec) For allowed values, see Table 2-3
1	Receiver Rate (Bits per sec) For allowed values, see Table 2-3
2	Number of Data Bits 8, 7, 6, or 5
3	Number of Stop Bits 1 or 2
4	Parity Checking and Generation 1 Enable 0 Disable
5	Odd or Even parity 1 Even 0 Odd
6	XON/XOFF Recognition 1 Enable 0 Disable
7	Pass 8-bit characters 1 Enable 0 Disable (pass 7-bit characters)
8	Auto answer ring count 0-9 decimal (zero means don't answer)
9	Modem Type -1 No modem, hardwired line 0 US FSK, 0-300 baud Bell 103 (TMS) 1 US DPSK, 1200 baud Bell 212 (TMS) 10 (decimal) Mini-Exchange

OVERVIEW OF COMMUNICATIONS SERVICES

Byte	Description
10	Dial Mode 0 Dial pulse, 10 pulses/second 1 DTMF 2 Dial pulse, 20 pulses/second 3 Off-hook service (external fixed number)
11	Data Mode 0 Voice telephone 1 Serial data 2 Encoded voice 3 DTMF data
12	DTMF Tone Time (10 ms multiples) 1 or 255
13	DTMF Interdigit Time (10 ms multiples) 1 or 255
14	TMS Silence Detect Timeout 1 Enable 0 Disable
15	Reserved

Table 2-3 shows the codes you can use to specify receive or transmit rate in bytes zero and one of the LDB.

Table 2-3: Transmit/Receive Rate Values for LDB

Symbol	Decimal Value	Actual baud rate (Bits per Second)
S.50	2	50
S.75	3	75
S.110	5	110
S.134	6	134.5
S.150	7	150

OVERVIEW OF COMMUNICATIONS SERVICES

Symbol	Decimal Value	Actual baud rate (Bits per Second)
S.200	8	200 (TMS only)
S.300	9	300
S.600	10	600
S.1200	11	1200
S.1800	12	1800
S.2000	13	2000
S.2400	14	2400
S.3600	15	3600
S.4800	16	4800
S.7200	17	7200
S.9600	18	9600
S.EXTA	19	External clocking
S.19.2	21	19200

To change a line characteristic, first call CCGMC or CCLCRG to get either the current settings or the default settings, respectively. Then modify the particular characteristic byte or bytes and write the new set back using CCLCRP or CCSMC. Note that CCLRP does not check for validity when you write back the default characteristics.

In general, all the settings in the LDB must be valid for the line to which you write them.

Much of the information contained in the line descriptor block is optional and depends on the type of link that you are setting up. For a voice link, the system uses only those fields that control the manner in which the call is dialed.

Some data options available via TMS are unavailable on the Communication Port, and vice versa. When you choose such an option, it is ignored, except where a functional incompatibility

OVERVIEW OF COMMUNICATIONS SERVICES

occurs. For example, you can specify the 200 baud rate only for TMS lines. Also, baud rates of 1800 and above are not supported on the TMS unit.

Note that a particular line characteristic (a byte in the LDB) corresponds to the second byte in the QIO functions SF.GMC and SF.SMC. (The communication routines that handle the line characteristics use these two QIO functions. See the *P/OS System Reference Manual* for details.)

2.1.3 Autodial Modem Support

The XK0: device handles any autodial modem that accepts a dial sequence as data passed on the primary transmit lead. This type of modem usually has both a start control sequence (indicating that a telephone number follows) and an end sequence (indicating the end of the telephone number). Not all modems require the end sequence.

You can use two methods of dialing through the Communication Port when an autodial modem is attached. The first method is to define the line as hardwired and transmit the dial sequence as data to the modem. Most autodial modems give some kind of response indicating whether or not the connection was successfully established. This can be read as data received from the modem. Note that this method will only work on XK0: (one side effect is that a hardwired line automatically answers any call because the DTR signal is held high).

The second method supports XK0:, XT1: and XT2: lines. A program that uses this method need not know the specific line type that the call is dialed on. To set up the system to support an autodial modem on XK0: you must define a translate table as the XK0: device.

The translate table contains three parts:

- The first part defines a set of character translations.
- The second part defines the start sequence for the autodial modem.
- The third part defines the end sequence.

The first part of the translate table defines a set of character translations that can occur during the dial operation. This allows you to substitute graphics characters for control characters in the phone number. Also, it allows you to remove

OVERVIEW OF COMMUNICATIONS SERVICES

format characters from the phone number. Normal format characters for a phone number are as follows:

[] () -

Additionally, the space character is a format character. A sample phone number with format characters is [12] (222) 555-4444.

You can set the translate table by calling the Base System routine CCMTT (see Section 2.2.2.6) or by executing the PRO/Communications Set-Up Utility (see Section 2.3.2).

By default (that is, if you do not override the default translate table), the XK0: device assumes that a DIGITAL DF03 autodial modem is attached.

Note that any of the parts of the table can be empty. Look in the user handbook for the start and end sequence of the particular autodial modem you are using. The translate table as stored and returned by the CCLCRP and CCLCRG routines begins at the first count-byte.

2.2 BASE SYSTEM SERVICES

The Base System Services include an asynchronous driver (XKDRV) for the Communication Port, as well as a communication service library (COMLIB). To perform communications operations, you can either issue QIO requests directly to the XKDRV (see the *P/OS System Reference Manual*), or call the routines in COMLIB.

If you are writing your program in a high-level language, it is best to use the COMLIB routines.

The Base System Services allow you to set up and control a telephone connection for data communication. Additionally, the Base Services allow you to handle voice communication if you install the optional Telephone Management System (TMS).

The following sections describe the Base System start-up and the Base System routines.

2.2.1 Base System Start-Up

When you power up a Professional and boot P/OS, a Base System start-up program executes. The program is present on all systems. This program loads the asynchronous driver, XKDRV, and

BASE SYSTEM SERVICES

sets up the default line characteristics. It also loads the TMS Driver, if the hardware is present, and sets the default line characteristics of TMS lines.

2.2.2 Base System Routines

This section describes the communication routines that DIGITAL provides with the P/OS operating system.

2.2.2.1 Attach Line (CCATT and CCATA) - There are two attach routines. The CCATA routine attaches a line and declares an AST routine that your application uses for unsolicited events. The CCATT only attaches the line. The caller specifies the AST routine that CCATA uses.

Format:

CCATT (status, lun)

CCATA (status, lun, ast, param)

status A two-word status block as described in Section 2.1.1.

lun The logical unit number for the communication line.

ast The entry point for an unsolicited event AST.

param A number that identifies this line as the input source upon entry to an unsolicited event AST routine.

Status:

CS.SUS 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

CE.IEA Internal error

BASE SYSTEM SERVICES

Note:

The routines CCATT and CCATA issue asynchronous QIOs to perform the attach, which always completes immediately. However, if another task has attached the line prior to your request, you receive the status return code CE.IEA, with the second status word equal to zero. The zero indicates that the attach request will pend until the line is free. When the line is free, the driver updates the status parameters with the completion status of the attach QIO (that is, the driver asynchronously writes the IOSB to the status block).

To avoid data corruption upon completion of the attach, you might want to use a separate status block when calling CCATT or CCATA.

2.2.2.2 Detach Line (CCDET) - This routine detaches a line previously attached by the CCATT or CCATA routines.

Format:

CCDET (status, lun)

status A two-word status block as described in Section 2.1.1.

lun The logical unit number for the communication line.

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
CE.IEA	Internal error

2.2.2.3 Set Line Characteristics (CCSMC) - This routine modifies line characteristics to the required settings. The routine writes the characteristics stored in the ldb parameter (see the call line) to the appropriate driver.

BASE SYSTEM SERVICES

Format:

CCSMC (status, lun, ldb)

status A two-word status block as described in Section 2.1.1.

lun The logical unit number for the communication line.

ldb The line descriptor block containing the new line characteristics. The block size must be 16 bytes.

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
CE.IEA	Internal error

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

Format:

CCGMC (status, lun, ldb)

status A two-word status block as described in Section 2.1.1.

lun The logical unit number for the communication line.

ldb A buffer into which the line characteristics may be returned in ldb format. The buffer size must be 16 bytes.

Status:

CS.SUC	Successful
CE.PRM	Service call parameter error
CE.DIR	RSX Directive error Second status word contains the DSW

BASE SYSTEM SERVICES

CE.IER I/O Termination error
 Second status word contains the IOSB first word

CE.IEA Internal error

2.2.2.5 Get/Put Line Configuration Record (CCLCRG and CCLCRP) -
You can use these routines to retrieve and modify the line configuration record for the specified line.

Format:

CCLCRG (status, dev, unit, ldb, ttable)

CCLCRP (status, dev, unit, ldb, ttable)

status A two-word status block as described in Section 2.1.1.

dev A two-byte ASCII string containing the device mnemonic.

unit The unit number in binary format.

ldb The line descriptor block. The block size must be 16 bytes.

ttable A 64-byte translate table for converting internal telephone numbers to a format accepted by an autodial modem. The table begins at the first count-byte.

Status:

CS.SUC Successful

CS.NTB Successful, no translate table defined (CCLCRG only)
 or
 Successful, ttable parameter not supplied
 (CCLCRG and CCLCRP)

CE.PRM Service call parameter error

CE.DIR RSX Directive error, second status word contains DSW

CE.RMS File I/O error
 PRO/RMS-11 error code passed in second word

CE.IEA Internal error

Notes:

These routines perform file I/O to the communications set-up data

BASE SYSTEM SERVICES

file. (This file stores default line characteristics, translate tables, and other information used by the communications services.) In order to perform the file I/O, you must assign a logical unit number to the set-up data file. Insert the following global definition command in your PAB command (.CMD) file:

```
GBLDEF = CM$LUN:lun
```

where "lun" is the logical unit number that the system associates with the communications set-up file.

See the section on command files in this manual for further information. Also, see the description of GBLDEF in the *RSX-11M/M-PLUS and Micro/RSX Task Builder Manual*.

If you specify "ttble" without having previously defined a translate table, CCLCRG assigns the value zero to the first three bytes of the translate table buffer. The CS.NTB status return indicates this condition. Note that both CCLCRG and CCLCRP also return CS.NTB if you omit the ttble parameter.

You can set up a translate table either by calling CCLCRP or by using the PRO/Communications application to "Set modem characteristics."

2.2.2.6 Set Translate Table (CCMTT) - This routine allows you to set a translate table when you are using an autodial modem.

Format:

CCMTT (status, lun, ttble)

status A two-word status block as described in Section 2.1.1.

lun The logical unit number for the communication line.

ttble A 64-byte translate table for converting internal telephone numbers to a format accepted by an autodial modem. The table begins at the first count-byte.

Status:

CS.SUC Successful

CE.PRM Service call parameter error

CE.DIR RSX Directive error
Second status word contains the DSW

BASE SYSTEM SERVICES

CE.IER I/O Termination error
 Second status word contains the IOSB first word

CE.IEA Internal error

Notes:

Some autodial modems require that a delay be introduced between sending the start sequence and presenting the telephone number. You can introduce a one-second delay by adding an "!" to the end of the start sequence. An additional "!" adds an additional one-second delay.

2.2.2.7 Dial Call (CCDIAL) - This routine dials a call in data mode on a designated communication line. The type of call you establish (voice or data) depends on the current data mode of the device. Use the CCSMC routine to set the data mode.

Format:

CCDIAL (status, lun, tel, len, efn, iosb, tmo)

status A two-word status block as described in Section 2.1.1.

lun The logical unit number for the communication line.

tel A string up to 32 bytes long representing a telephone number.

len The length of the telephone number.

efn An event flag the system sets on completion of the operation, or zero if you perform the operation synchronously.

iosb A two-word array used as an IOSB.

tmo The timeout period for the dial to complete. Specify the value as follows:

 bits 0-7 Number of ten-second intervals, up to 255 decimal

 bits 8-15 Number of one-second intervals, up to 255 decimal

The longest timeout interval is 255 (decimal) seconds. If the timeout value is larger than 255 seconds, the routine uses 255 seconds.

BASE SYSTEM SERVICES

If you do not specify tmo, the driver uses 60 seconds as the timeout value.

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
CE.IEA	Internal error

Notes:

Before issuing the dial, the Communication Port driver raises DTR and RTS. These may be dropped depending upon the termination status of the dial, as described in the following paragraphs.

If you specify a zero timeout (or no timeout parameter), the Communication Port driver waits 60 seconds to establish carrier. If at the end of this period carrier is not raised, then DTR and RTS are dropped.

If you specify a nonzero timeout, the request completes after the timeout period or after a connection is established. If the timeout period expires, then the Communication Port driver drops DTR and RTS and the routine returns an IE.DNR error in the first word of the I/O status block. If the modem raises carrier before the timeout period expires, the routine returns IS.SUC in the first word of the I/O status block.

When dialing a call, the Communication Port driver automatically deletes the format effectors. These include left and right parentheses, the dash, and the space character. The driver also inserts the appropriate start and end codes for a DF03 modem, provided that you use the default translate table. For other types of format effector editing, or for support of autodial modems with different start/end codes, you must set up a new translate table (see the description of the CCMTT routine earlier in this chapter).

If you are dialing TMS, you can specify the following characters in your telephone number.

BASE SYSTEM SERVICES

! An exclamation mark causes an access pause and waits for an initial or intermediate dial tone. This pause is required by some telephone systems.

If TMS receives a dial tone, it continues dialing the remaining characters in the dial string. After six seconds, if TMS has not received a dial tone, it continues dialing as if it had received a dial tone.

You can use this feature where a second dial tone is needed. For example, to dial from within a company's phone system that requires a "9" followed by a dial tone, the Phone Book entry could be: 9!(222)555-4444.

!! Two exclamation marks cause a longer access pause (maximum of 40 seconds) that waits for an intermediate dial tone. This is used when more than a few seconds of network switching time is needed.

, A comma provides a fixed, two-second delay.

The pound sign symbol is used in two cases. First, if you are dialing in Touch-Tone® mode, it transmits the "#" signal, which may be used if your telephone recognizes it (for example, as a terminator at the end of variable-length international telephone numbers).

The pound sign can also be used where mixed-mode dialing is necessary. For example, if your phone service allows only rotary dial, and you need to communicate with a device that requires Touch-Tone® signals, the pound sign switches from rotary dialing to Touch-Tone® dialing; the pound sign itself is not sent. For example 555-4444# could dial in rotary, and all communication after the pound sign would occur in Touch-Tone®.

*ABCD These additional signals are required by special communications networks and applications. The asterisk is available if you have Touch-Tone® service.

If you are dialing the Mini-Exchange, you can specify the following characters in your telephone number. Note that you must have specified Mini-Exchange as your modem type in byte 9 of the line descriptor block (see Section 2.1.2 for details).

BASE SYSTEM SERVICES

- 1-8 The Mini-Exchange ports are numbered 1 through 8. To connect to a Mini-Exchange port, use the port number instead of a phone number.
- R The R character specifies that a modem is attached to port 8. It can be followed by a telephone number that the modem will dial. For example, R555-4444.
- HR HR specifies that a modem operating at 1200 baud is attached to port 8. It can be followed by a telephone number that the modem will dial. For example, HR555-4444.
- LR LR specifies that a modem operating at 300 baud is attached to port 8. Mini-Exchange will set the baud rate to that of the modem. It can be followed by a telephone number that the modem will dial. For example, LR555-4444.
- An underscore indicates that the modem of one Mini-Exchange is connecting to the port of another Mini-Exchange (the two Mini-Exchanges must be connected by modems). For example, if 555-4444 were the number of the other Mini-Exchange, R555-4444_6 would dial it and connect to its port number 6. The port selector following a phone number can be 1 through 7 only.
- P The P character following a port number indicates that a DIGITAL printer is attached to the specified port. Note that a telephone number cannot accompany the P code.

When the Professional accepts a connection through Mini-Exchange, its baud rate is automatically set to that of the originator.

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

Format:

CCANS (status, lun)

status A two-word status block as described in Section 2.1.1.

lun The logical unit number for the communication line.

Status:

CS.SUC Successful

BASE SYSTEM SERVICES

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

CE.IEA Internal error

Notes:

If no connection occurs within 60 seconds, the routine returns an error status. You should call this routine in response to a "ring indicator" unsolicited event.

In the case of a TMS line where the call has already been established in voice mode, this routine causes the TMS software to connect the designated modem in answer mode when a change mode command is issued to place the line in ASYNC mode. The corresponding party must have placed their modem in originate mode. (Use the CCORG routine, followed by a change mode to ASYNC, to place a modem in originate mode.)

The converse of this is also true. That is, to change a line from ASYNC mode to voice mode, use the following procedure:

1. Use CCPTGV to indicate that the line is not to be disconnected when carrier loss occurs.
2. Use CCMODE to set the desired mode, which does not take effect until you issue a CCANS or CCORG call.
3. Use CCANS or CCORG to effect the mode change.

2.2.2.9 Originate Call (CCORG) - This routine initiates the connection in originate mode, once the line has already been connected. You can use the routine to determine whether or not a connection currently exists, or to raise DTR and RTS before making a manual connection.

Format:

CCORG (status, lun, efn, iosb, tmo)

BASE SYSTEM SERVICES

status A two-word status block as described in Section 2.1.1.

lun The logical unit number for the communication line.

efn An event flag the system sets on completion of the operation, or zero if you perform the operation synchronously.

iosb A two-word array to be used as an IOSB.

tmo The optional timeout parameter. See the description of the tmo parameter in the description of CCDIAL.

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
CE.IEA	Internal error

Notes:

If you do not specify a timeout parameter, the driver raises DTR and RTS. You can then initiate a manual connection in the normal manner. If you have specified that the line has a modem, DTR is not raised until you issue CCORG with no timeout parameter. When using CCDIAL, the driver automatically raises DTR before dialing a call.

If you specify a zero timeout parameter, the request completes immediately. When carrier is down, the driver drops DTR and RTS and the routine returns an IE.DNR error in the first word of the IOSB. When carrier is up, the routine returns IS.SUC in the first byte of the IOSB. You can use this to determine if a connection exists on a line.

If you specify a nonzero value in timeout, the request completes either after the timeout period or after a connection is established. If the timeout period expires, the driver drops DTR and RTS and the routine returns an IE.DNR error in the first word of the IOSB. If carrier is up (or comes up before the timeout period expires), the routine returns IS.SUC in the first word of the IOSB.

BASE SYSTEM SERVICES

2.2.2.10 Hang Up a Call (CCHNG) - This routine disconnects a call previously established on a specified line.

Format:

CCHNG (status, lun)

status A two-word status block as described in Section 2.1.1.

lun The logical unit number for the communication line.

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

CE.IEA Internal error

Note:

The Communication Port driver drops DTR and RTS when CCHNG executes. The TMS driver puts the telephone line on-hook.

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

Format:

CCTXD (status, lun, efn, iosb, stadd, size)

status A two-word status block as described in Section 2.1.1.

lun The logical unit number for the communication line.

efn An event flag the system sets on completion of the operation, or zero if you perform the operation synchronously.

BASE SYSTEM SERVICES

iosb A two-word array used for the IOSB.

stadd A buffer that contains the data to be transferred.

size An integer specifying the amount of data to be transferred.

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

CE.IEA Internal error

2.2.2.12 Receive Data (CCRXD) - This routine reads data from a communication line. You can either read all the data currently buffered in the driver or wait a specific period of time and read all the data accumulated up to that point. Alternatively, you can read a fixed amount of data from the line. Input always terminates when your task's buffer becomes full.

If you specify XON/XOFF support, the driver transmits XOFF when the driver's buffer becomes three-quarters full, and transmits XON when the driver's buffer is emptied to the one-quarter point. The driver has a limited amount buffer space.

If the driver's buffer becomes full, additional incoming characters are lost.

Note that the amount of data received is in the second word of the IOSB.

Format:

CCRXD (status, lun, efn, iosb, stadd, size, tmo)

status A two-word status block as described in Section 2.1.1.

BASE SYSTEM SERVICES

lun The logical unit number for the communication line.

efn An event flag the system sets on completion of the operation, or zero if you perform the operation synchronously.

iosb A two-word array for the IOSB.

stadd A buffer into which the routine can place the data.

size An integer specifying the input buffer size.

tmo If you do not specify this parameter, input is not complete until "size" characters have been written to the buffer.

If tmo is zero, the request returns immediately after transferring as many characters as are available up to the input buffer size. If tmo is not equal to zero, the request completes after the timeout period specified. See CCDIAL for the format of tmo.

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
CE.IEA	Internal error

Note:

If you are using a TMS line in DTMF mode, the Touch-Tone® keys are presented as data characters. (See the *TMS Programmer's Manual*.) Further, in CODEC mode the use of the Touch-Tone® keys is signalled by an AST (if the line is attached for ASTs).

2.2.2.13 Flush Input Buffer (CCFLSH) - This routine flushes the input buffer for the specified communication line.

BASE SYSTEM SERVICES

Format:

CCFLSH (status, lun)

status A two-word status block as described in Section 2.1.1.

lun The logical unit number for the communication line.

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

CE.IEA Internal error

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

Format:

CCBRK (status, lun, brk)

status A two-word status block as described in Section 2.1.1.

lun The logical unit number for the communication line.

brk Specify 0 to send a break. Specify 1 to send a long space.

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

BASE SYSTEM SERVICES

Second status word contains the IOSB first word

CE.IEA Internal error

2.2.2.15 Kill Transfer (CCKILL) - This routine kills any outstanding transfers. You can use it only with asynchronous I/O transfers. The outstanding transfer terminates in the normal manner (the system sets the event flag you specified at initiation).

Format:

CCKILL (status, lun)

status A two-word status block as described in Section 2.1.1.

lun The logical unit number for the communication line.

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

CE.IEA Internal error

2.3 PRO/COMMUNICATIONS SERVICES

PRO/Communications services consist of:

- An application that you can invoke from the P/OS menu system.
- A set of utilities that you can call from your application program.
- A set of routines that allow you to perform Phone Book and File Transfer operations.

As with the Base System Services, you can access PRO/Communications services by calling the COMLIB routines.

PRO/COMMUNICATIONS SERVICES

The following sections describe PRO/Communications start-up, the utilities you can execute, and the routines that allow you to perform Phone Book and File Transfer operations.

2.3.1 PRO/Communications Start-Up

Once you have installed PRO/Communications on your system, every time you power up a Professional and boot P/OS, a PRO/Communications Start-Up Program executes. It performs the following operations:

- installs necessary PRO/Communications tasks
- starts up the background File Transfer Subsystem

The PRO/Communications Start-Up Program is present on your system only if you have installed PRO/Communications.

2.3.2 PRO/Communications Utilities

Any application program can use the PRO/Communications utilities. If you install PRO/Communications on the machine, the system installs all utilities as part of the start-up procedure.

The utilities supplied with PRO/Communications are:

- Communications Set-Up - allows the user to specify parameters that control the functioning of the communications services. To allow users to change the parameters, your application program can either invoke this service or call the Base System routines.
- Terminal Emulator - allows the Professional to act as a VT102, VT125, VT52 or native mode terminal connected to a host operating system. Your program can invoke the terminal emulator to allow the user to communicate with a host system. The host system, in turn, can terminate the emulator and return control to the application.
- Call Services - allows users to dial and answer calls, using either TMS or an autodial modem attached to the Communication Port. This service runs a call control task that you can use to make connections via entries stored in the Phone Book.

PRO/COMMUNICATIONS SERVICES

- Phone Book Maintenance - allows users to maintain the phone book. The phone book is a file containing information about how to connect a Professional to another computer system. Note that the routines you can use to access the phone book do not allow maintenance. The phone book runs two programs as tasks: a file maintenance program and a server program that reads or displays entries in the phone book. You can call the server program through the phone book services.

Each of the utilities has a user interface. See the *Pro/Communications Manual* for a description of the utilities and their user interfaces. Also, see the *Terminal Subsystem Manual* for additional details on the terminal emulator.

You use the CCSPWN routine (see below) to make a request to initiate a utility.

2.3.2.1 Spawn Communications Utility (CCSPWN) - This routine spawns a PRO/Communications utility task. On termination the routine returns the exit status to the requestor.

Format:

CCSPWN (status, name)

status A two-word status block as described in Section 2.1.1.

name A four-byte string that identifies the requested utility.

EMUL	Terminal Emulator
CSET	Communications Set-Up
CALL	Call Services
PHNE	Phone Book Maintenance

Status:

CS.SUC	- Successful
CE.PRM	- Service call parameter error
CE.DIR	- RSX Directive error Second status word contains the DSW
CE.IEA	- Internal error

PRO/COMMUNICATIONS SERVICES

Note:

In the case of a successful return, the routine returns the exit status of the utility in the second status word, as follows:

1.	-	Successful
2-4	-	Error exit
5-9.	-	Unused
9.	-	MAIN SCREEN key was used to terminate program
10.	-	EXIT key was used to terminate program

2.3.3 Phone Book Services

The services described in this section allow an application to read specified entries from the phone book, or display the phone book and allow the user to select an entry. To perform phone book maintenance, you must use the phone book maintenance program.

2.3.3.1 Get Phone Book Record (CPHREC) - This service returns a specified record from the phone book.

Format:

CPHREC (status, name, number, numlen, descr, descrlen, ldb)

status	A two-word status block as described in Section 2.1.1.
name	A 15-byte, space-filled string in which you specify the name of the entry to be retrieved.
number	A string in which the routine returns the telephone number. The string can be up to 48 bytes.
numlen	An integer the routine returns indicating the length of the telephone number.
descr	A string in which the routine returns the description stored in the phone book. The string can be up to 40 bytes.
descrlen	The length of the description string.
ldb	The line descriptor block, a 16-element byte string in which the routine returns the line characteristics required to set up the call.

PRO/COMMUNICATIONS SERVICES

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
CE.NNF	- Name not found
CE.IEA	- Internal error

Note:

If you call this routine with a blank name parameter, it functions identically to CPHSEL.

2.3.3.2 Select Phone Book Entry (CPHSEL) - This service displays all the phone book entries and allows the user to select a specific entry from those available. Note that the task that calls CPHSEL must be running from TT1:, or the system returns the error CE.IEA.

Format:

CPHSEL (status, name, number, numlen, descr, descrlen, ldb)

status	A two-word status block as described in Section 2.1.1.
name	A 15-byte string in which the routine returns the name of the entry to be retrieved.
number	A string in which the routine returns the telephone number. The string can be up to 48 bytes.
numlen	An integer the routine returns indicating the length of the telephone number.
descr	A string in which the routine returns the description stored in the phone book. The string can be up to 40 bytes long.
descrLen	The length of the description string.
ldb	The line descriptor block, a 16-element byte string in

PRO/COMMUNICATIONS SERVICES

which the routine returns the line characteristics required to set up the call.

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
CE.NNF	- Name not found
CE.IEA	- Internal error
CE.CTB	- CTAB display error (empty phone book)

2.3.4 File Transfer Services

This section describes services that allow you to use the File Transfer Subsystem. Using the File Transfer Subsystem, you can transfer files between Professionals through callable service routines. (Note that PRO/Communications already includes a Professional-to-Professional file transfer application, which itself uses the File Transfer Subsystem.)

The File Transfer Subsystem consists of the following components:

- Communications Manager Program - controls the initiation and termination of the file transfer programs.
- File Transfer Server - executes in response to a request from another system to initiate a file transfer. The server communicates with the "host" program to perform a file transfer. The host program currently runs on RSX-11M/11M+, VMS or the Professional.
- File Transfer Host Program - initiates a file transfer. It communicates with the server program.
- Listener Program - monitors the following communication lines in the background: the Communication Port, TMS line 1 and TMS line 2. The program monitors lines in data mode only for ASYNC transfers. When the Professional receives unsolicited

PRO/COMMUNICATIONS SERVICES

ASYNCR data over a monitored line, the listener program attempts to interpret the data as an ANSI escape sequence, provided the line is not currently attached and no outstanding reads are present. The listener processes only file transfer requests.

You can initiate a file transfer to the Professional either when the Professional is in terminal emulation mode or when the listener intercepts the file transfer escape sequence. The latter case constitutes a background transfer.

The File Transfer Subsystem always runs asynchronously in the background; however, the program issuing calls to it can either synchronize with each call to the Subsystem or run file transfer operations asynchronously.

2.3.4.1 File Transfer Initiation Protocol - Receipt of the escape sequence initiates file transfer. The escape sequence is:

```
ESC [ 1 ! ~
```

Prior to sending the escape sequence, the file transfer host program tries to determine if the Professional is able to perform the transfer. It does this by sending a device status request (DSR):

```
ESC [ 5 n
```

The Professional must respond with a "ready, no malfunctions" response report:

```
ESC [ 0 n
```

After accepting the file transfer escape sequence, your application should request the file transfer server task. At the end of the file transfer, a further DSR escape sequence will be received and the "ready, no malfunctions" report described above must be returned.

2.3.4.2 Attaching the File Transfer Subsystem - In order to initiate a file transfer, the requesting task must attach the File Transfer Subsystem. This applies both to transfers initiated by the local system and requests to activate the server in response to requests from a remote system. Because of processor limitations, only one file transfer can be active in the system at any time. If another task has already attached the File Transfer Subsystem, the Subsystem rejects further requests

PRO/COMMUNICATIONS SERVICES

for attachment. The process of attachment informs the File Transfer Subsystem of the line on which it performs the transfer. Also, attachment can optionally send a password to the remote system (if the local system is the initiator).

2.3.4.3 Releasing the File Transfer Subsystem - You automatically release the File Transfer Subsystem upon terminating the task that attached it. Also, you can explicitly request the release of the Subsystem (using the FTDET routine). Note that any queued requests continue to be processed, even if you have released the Subsystem.

2.3.4.4 Synchronous and Asynchronous Operations - A program that initiates an operation to the File Transfer Subsystem has two options. It either can wait for the operation to complete, or can synchronize with the return message at a later point in time. This is called an asynchronous operation; you can use it to initiate multiple file operations.

Each call to the file transfer services (except FTDET) results in a message sent to the File Transfer Subsystem. For each message sent, the Subsystem returns a response to the requesting task (unless suppressed). You can synchronize these responses by calling the FTSYNC routine. Use the msgid parameter to identify the response to the requesting task.

2.3.4.5 File Transfer Synchronization - The file transfer service calls contain "reply" and "sync" parameters to enable synchronization between your application program and the File Transfer Subsystem. Calls to the file transfer routines that contain a "reply" parameter cause action messages to be sent to the File Transfer Subsystem. The Subsystem generates response messages depending on the value you specify for "reply." These messages are passed using the variable send/receive data directives. You can synchronize the caller with response messages from the Subsystem by either setting the sync parameter when issuing a call and setting the reply flag to 2 (running the Subsystem synchronously), or calling the FTSYNC routine, which also uses the sync parameter.

In either case, setting the sync flag causes the calling task to stop for a response message from the File Transfer Subsystem. The user's impure data area holds the response; you can access it by calling FTUNPK.

PRO/COMMUNICATIONS SERVICES

The possible values for "reply" are:

Bit(s)	Value	Meaning
0-1	0	No response message.
	1	Response message requested.
	2	Response message and unstop requested. File Transfer Subsystem will issue unstop directive for requesting task. You must use this value if sync flag = 1.
2-3	0	Do not write message to system message board.
	1	Write message to system message board.
	2	Write message to system message board only on error.
	3	Write message to system message board only if successful.
3-15		Reserved.

NOTE

The File Transfer Subsystem writes messages to the system message board when a task that has requested a response message has exited.

2.3.4.6 Set Up File Transfer Options (FTOPTG and FTOPTP) - Two routines allow you to configure the file transfer options 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.

Format:

FTOPTG (status, flags, pswrd, vol, direc, max_file)

FTOPTP (status, flags, pswrd, vol, direc, max_file)

status A two-word status block as described in Section 2.1.1.

flags Option Flags:

PRO/COMMUNICATIONS SERVICES

- bit 0 1 Enable remote file copy
0 Disable remote file copy
- bit 1 1 Enable remote file receive
0 Disable remote file receive
- bit 2 1 Supersede existing files
0 Do not supersede existing files
- bit 3 1 Enable password security
0 Disable password security
- bit 4 1 Local delete enabled
0 Local delete disabled

Bits 5 through 15 are reserved.

- pswr A nine-character password to be used by a remote Professional. It must be space filled if you have disabled password security.
- vol A 12-character name of the volume to which the File Transfer Subsystem places files received from another Professional. The Subsystem uses this parameter only if the incoming file specification contains no volume specification.
- direc A nine-character name of the directory into which the File Transfer Subsystem places files being received from another Professional. The Subsystem uses this parameter only if the incoming file specification contains no directory specification.
- max_file A two-word integer specifying the maximum size, in blocks, of a local file that the remote Professional can create. Note that a zero value indicates no limit.

Status:

- CS.SUC Successful
- CE.PRM Service call parameter error
- CE.RMS File I/O error
PRO/RMS-11 error code passed in second word
- CE.IEA Internal error

PRO/COMMUNICATIONS SERVICES

NOTE

These routines perform file I/O to the communications set-up data file. (This file stores default line characteristics, translate tables, and other information used by the communications services.) In order to perform the file I/O, you must assign a logical unit number to the set-up data file. You do this by inserting the following global definition command in your PAB command (.CMD) file:

```
GBLDEF = CM$LUN:lun
```

lun is the logical unit number that the system associates with the communications set-up file.

See the section on command files in this manual for further information. Also, see the description of GBLDEF in the *RSX-11M/M-PLUS* and *Micro/RSX Task Builder Manual*.

2.3.4.7 Attach File Transfer Subsystem (FTATT) - This routine attaches the designated line to the issuing task for use with the File Transfer Subsystem. You must call this routine prior to invoking any of the other File Transfer Subsystem routines, except as noted in the routine definitions. The issuing task can optionally specify a password to gain access to the remote system.

Format:

FTATT (status, msgid, reply, sync, dev, unit, 'PSWD', paswr)

status A two-word status block as described in Section 2.1.1.

dev A two-byte ASCII string containing the device mnemonic.

unit The unit number.

reply A reply flag.

msgid An unsigned integer that identifies the return message for this call.

PRO/COMMUNICATIONS SERVICES

sync A synchronization flag: an integer that determines whether or not the calling program will stop if no response message is queued.

- 0 - Do not stop for response
- 1 - Stop for response message

PSWD An optional keyword that specifies the presence of a password parameter immediately following.

paswrđ A optional nine-byte string containing the password for the remote system, space-filled. You must immediately precede this parameter with the PSWD keyword.

Status:

- CS.SUC - Successful
- CE.PRM - Service call parameter error
- CE.DIR - RSX Directive error
Second status word contains the DSW
- CE.IEA - Internal error

Notes:

If a task issues two calls to FTATT without an intervening call to FTDET, an error occurs on the second call to FTATT.

2.3.4.8 File Transfer Operation Request (FTOPRN) - This routine issues file operation requests to the File Transfer Subsystem. The routine allows you to: transfer files between systems, determine the status of file transfers currently in progress, or abort the current transfer. You can specify only one file operation request in any call to the routine.

Format:

FTOPRN (status, msgid, reply, sync, request)

The format of a request is:

```
'ABRT',  
'STAT',  
'SEND', fspec1, len1, fspec2, len2, protcl  
'READ', fspec1, len1, fspec2, len2, protcl  
'DELE', fspec1, len1, protcl
```

PRO/COMMUNICATIONS SERVICES

status A two-word status block as described in Section 2.1.1.

msgid An unsigned integer that identifies the return message for this call.

reply Reply flags.

sync A synchronization flag: an integer that determines whether or not the calling program will stop if no response message is queued.

0 - Do not stop for response
1 - Stop for response message

ABRT Allows you to abort the current file operation. If no file operation is in progress, the routine returns an error in the "status" parameter.

STAT Causes the routine to return statistics on the current file transfer operation to the caller. You can access the statistics via the FTUNPK routine.

SEND Allows you to transfer files to the remote system. You can use only a single file specification (no wildcards).

READ Allows you to transfer files from the remote system. You can use only a single file specification (no wildcards).

fspec1 A string buffer holding the file specification of the file to be transferred.

len1 The length of fspec1.

fspec2 A string buffer holding the destination file specification of the file to be transferred.

len2 The length of fspec2. By default, the destination file has the same name and type as fspec1. If no change is required, len2 should be zero. If you want to place the file on a specific device or in a specific directory, you must specify fspec2; otherwise, the routine uses the system's default destination.

DELE Allows you to delete a file at a remote system. You can only specify a single file specification (no wildcards).

PRO/COMMUNICATIONS SERVICES

fspec1 A string buffer holding the file specification of the file to be deleted.

len1 The length of **fspec1**.

protcl A one-word integer indicating the protocol to be used on the operation. It can have one of two values:

- 0 - Use existing file transfer protocol (default)
- 1 - Use encoded protocol

In order to use encoded protocol, both systems involved in the file transfer must be running PRO/Communications V3.0 or later.

Status:

- CS.SUC - Successful
- CE.PRM - Service call parameter error
- CE.UNO - Unsupported message option
- CE.MFE - Message format error
- CE.IMO - Operation rejected
- 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.NFT - No file transfer
- CE.IEA - Internal error

2.3.4.9 Notify On Incoming File (FTNTFY) - This routine requests the File Transfer Subsystem to notify a specified task when a file is received from a remote system. Upon arrival of each file, the Subsystem sends a duplicate of the response message that it returns to the task requesting the server activation. This indicates the full specification of the incoming file and termination statistics.

PRO/COMMUNICATIONS SERVICES

Format:

FTNTFY (**status, msgid, reply, sync, notify,**
'TASK', taskid, 'CNCL')

- status** A two-word status block as described in Section 2.1.1.
- msgid** An unsigned integer that identifies the return message for this call.
- reply** Reply flags.
- sync** Synchronization flag: an integer that determines whether or not the calling program stops if no response message is queued. This parameter applies only to the File Transfer Subsystem's initial verification to indicate acceptance of the request.
- 0 - Do not stop for response
 - 1 - Stop for response message
- notify** Notification flag: an integer that determines how the File Transfer Subsystem notifies the task that the specified file has arrived.
- 1 - Send the notification message to the specified task.
 - 2 - Send notification message and unstop the specified task.
 - 3 - Send the notification message using the request and connect directive.
- TASK** Requests the File Transfer Subsystem to direct notification messages to the task specified in taskid. If you omit this option, the routine by default uses the current task name.
- taskid** Task to be notified, in Radix-50 format.
- CNCL** Cancels previous requests for file transfer notification. The only other optional parameter that you can use with 'CNCL' is 'TASK.'

Status:

- CS.SUC** - Successful
- CE.IMO** - Operation Rejected
- CE.PRM** - Service call parameter error
- CE.IEA** - Internal error

PRO/COMMUNICATIONS SERVICES

CE.DIR - RSX Directive error
 Second status word contains the DSW

Notes:

The File Transfer Subsystem sends response messages to the specified task. You can interpret these messages by using the FTSYNC and FTUNPK routine to access the filename and termination statistics of the file received. Setting the sync flag when calling FTSYNC requires having used a notification flag = 2. The original msgid you specified when calling FTNTFY is preserved in all subsequent notification messages. The task being notified need not be active.

2.3.4.10 Synchronize with File Transfer (FTSYNC) - Use this routine to synchronize your task with an asynchronous request previously issued by a call to FTATT, FTDET, FTOPRN or FTNTFY. It places a response message from the File Transfer Subsystem into the calling program's impure data area. You should immediately unpack the response message by calling FTUNPK.

Format:

FTSYNC (status, msgid, sync)

status A two-word status block as described in Section 2.1.1.

msgid A message identifier that the File Transfer Subsystem returns. You pass this parameter in the original call.

sync Synchronization flag: an integer that determines whether the program stops if no message is currently queued.

0 - Do not stop
1 - Stop for response message

Status:

CS.SUC - Successful

CE.PRM - Service call parameter error

CE.IMO - Operation Rejected

CE.STU - File transfer start-up in progress

CE.IEA - Internal error

PRO/COMMUNICATIONS SERVICES

2.3.4.11 Unpack File Transfer Message Area (FTUNPK) - This routine unpacks the message components from the impure data area. You specify each message component with a keyword. Note that the calling task need unpack only those components that are of interest to it. You must immediately precede a call to FTUNPK with a call to FTSYNC.

NOTE

Any call to the other communication service routines between calls to FTSYNC and FTUNPK can overwrite the response message held in the application's impure data area.

Format:

```
FTUNPK (status, 'SIZE', filesize,  
          'FILE', fspec, len,  
          'BLCK', blknum,  
          'DEVC', dev,  
          'UNIT', unit,  
          'TIME', elpstim,  
          'OPER', operation )
```

status A two-word status block as described in Section 2.1.1.

SIZE Returns a two-word integer indicating the size of the file you are transferring.

filesize A two-word integer into which the routine places the file's size.

FILE Returns the name of the current file on which you are performing the file operation.

fspec A buffer into which the routine can place the file specification of the file being transferred. This buffer should be at least 50 bytes long.

len The length of the file specification returned in the fspec buffer.

BLCK Returns the current block number of the file you are transferring.

blknum A two-word integer that receives the current block number of the file you are transferring.

DEVC Returns the device mnemonic for the current file operation.

PRO/COMMUNICATIONS SERVICES

dev A two-byte ASCII string that receives the device mnemonic.

UNIT Returns the unit number for the current file operation.

unit An integer that receives the unit number.

TIME Returns the elapsed time for the current file transfer.

elpstim A two-word integer that receives the elapsed time in seconds.

OPER Returns the current file operation.

operation A single-length integer that receives the code for the operation in progress:

- 1 - READ
- 2 - SEND
- 3 - LOCAL DELETE
- 4 - REMOTE DELETE

Status:

CS.SUC - Successful

CE.PRM - Service call parameter error

CE.IMO - Operation rejected

CE.MSC - Message type out of sync

CE.IEA - Internal error

2.3.4.12 Detach File Transfer (FTDET) - This routine detaches the File Transfer Subsystem from use by the requesting task.

Format:

FTDET (status, msgid, reply, sync)

status A two-word status block as described in Section 2.1.1.

msgid An unsigned integer that identifies the return message for this call.

reply Reply flags.

Status:

PRO/COMMUNICATIONS SERVICES

CS.SUC - Successful

CE.PRM - Service call parameter error

CE.DIR - RSX Directive error
Second word contains the DSW

CE.IEA - Internal error

2.3.4.13 Start File Transfer Server (FTSERV) - This routine requests the file transfer server on the local system. You should call this routine in conjunction with initiation of the host file transfer program.

Format:

FTSERV (status, msgid, reply, sync, protcl)

status A two-word status block as described in Section 2.1.1.

msgid An unsigned integer that identifies the return message for this call.

reply Reply flags.

sync Synchronization flag: an integer that determines whether or not the calling program will stop if no response message is queued.

- 0 - Do not stop for response message
- 1 - Stop for response message

protcl A one-word integer indicating the protocol to be used on the operation. It can have one of two values:

- 0 - Use existing file transfer protocol (default)
- 1 - Use encoded protocol

In order to use encoded protocol, both systems involved in the file transfer must be running PRO/Communications V3.0 or later.

Status:

CS.SUC - Successful

CE.PRM - Service call parameter error

CE.UNO - Unsupported message option

PRO/COMMUNICATIONS SERVICES

CE.MFE - Message format error
CE.IMO - Operation rejected
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.NFT - No file transfer
CE.IEA - Internal error

2.3.4.14 Enable/Disable File Listening (FTLISN) - Calling this routine with the enable flag set causes the File Transfer Subsystem to monitor an unattached line for unsolicited data. The listener program reads unsolicited data from the specified line. Upon receiving a valid file transfer request, the listener automatically initiates a file transfer.

Note that other applications can ATTACH or DETACH the File Transfer Subsystem; however, the listening task will not be able to initiate file transfers while the Subsystem is attached.

Format:

FTLISN (status, dev, unit, flag)

status A two-word status block as described in Section 2.1.1.

dev A two-byte ASCII string containing the device mnemonic.

unit The unit number.

flag A flag that indicates the type of request:

0 - Disable listening
1 - Enable listening

Status:

CS.SUC - Successful
CS.FLE - File listening already enabled with enable request
CS.FLD - File listening already disabled with disable request

PRO/COMMUNICATIONS SERVICES

- CE.PRM - Service call parameter error
- CE.DIR - RSX Directive error
Second word contains the DSW
- CE.IEA - Internal error

Note:

If you call FTLISN with just the status parameter, the second status word contains the current listener status. Bit zero set means listening is enabled on XK0:.

2.4 TELEPHONE MANAGEMENT SYSTEM (TMS) SERVICES

This section describes services that enable your application program to control the TMS hardware.

Note the following if you are using CODEC input on a TMS line: TMS digitizes the voice signal at 4 kilobytes per second, and so you will lose voice data unless you perform asynchronous I/O and double buffer the input requests. Each input request should specify a 4 kilobyte buffer; this provides 1 second for your application to process the first buffer before the second buffer fills.

The following sections describe TMS start-up and routines.

2.4.1 TMS Start-Up

Base system start-up loads the TMS device driver and sets the default line characteristics for each TMS line.

2.4.2 TMS Routines

This section describes the routines that you can call to control the operation of the TMS hardware. Note that you must have installed the hardware in order to call any of the routines.

2.4.2.1 Change Mode (CCMODE) - This routine changes the current mode of the line. The line must be attached. Also, the mode change does not take effect until you issue a CCANS or CCORG call.

TELEPHONE MANAGEMENT SYSTEM (TMS) SERVICES

Format:

CCMODE (status, lun, mode)

status A two-word status block as described in Section 2.1.1.

lun The logical unit number for the communication line.

mode One of the following available mode settings:

- 0 = VOICE
- 1 = ASYNC (Modem)
- 2 = CODEC
- 3 = DTMF Keypad (Touch-Tone® pass all keys)

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
- CE.IEA - Internal error

Note:

After calling CCMODE, if you want to change the current mode to ASYNC, you must establish a connection by calling CCORG. In the case where your application will be communicating with another TMS unit, one unit must be in originate mode (using CCORG) and the other TMS unit must be in answer mode (using CCANS).

2.4.2.2 Auxiliary Keyboard Enable/Disable (CCAUXK) - This routine enables or disables the auxiliary keyboard for the specified TMS line. The line must be attached. An AST trap signals input from the auxiliary keyboard. You must use the CCATA routine to attach the line.

Format:

CCAUXK (status, lun, flag)

TELEPHONE MANAGEMENT SYSTEM (TMS) SERVICES

status A two-word status block as described in Section 2.1.1.

lun The logical unit number for the communication line.

flag This flag indicates whether the keyboard is being enabled/disabled:

- 0 - Disabled
- 1 - Enabled

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

CE.IEA - Internal error

2.4.2.3 Prepare to Go Voice (CCPTGV) - This routine warns the specified TMS line that it is about to go into VOICE mode. TMS does not disconnect the call when the change in mode occurs. The line must be attached.

Format:

CCPTGV (status, lun)

status A two-word status block as described in Section 2.1.1.

lun The logical unit number for the communication line.

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

TELEPHONE MANAGEMENT SYSTEM (TMS) SERVICES

CE.IEA - Internal error

Note:

You can call CCHNG to cancel CCPTGV and disconnect the telephone line.

2.4.2.4 Set DTMF Escape Sequence (CCDTMF) - This routine sets a DTMF escape sequence on the specified TMS line. The line must be attached. When the TMS unit recognizes the DTMF escape sequence the driver causes an unsolicited event AST for the user task to indicate the event. (See the *TMS Programmer's Manual*.)

Format:

CCDTMF (status, lun, dtmf, dtmflen)

status A two-word status block as described in Section 2.1.1.

lun The logical unit number for the communication line

dtmf A string that contains the DTMF escape sequence. The string terminates with a zero byte.

dtmflen The length of the DTMF escape sequence.

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

CE.IEA - Internal error

CHAPTER 3

FAST INSTALL

Fast Install allows you to install your application without having to build application diskettes. It saves time when you are testing your application.

Fast Install does the following:

- Checks the syntax and organization of your installation command file.
- Processes EXECUTE/INS commands in your installation command file. (It does not process EXECUTE commands with the /CLS, /NTW, or /USR commands.) See Chapter 6 for details on the EXECUTE command.
- Places the application name into the P/OS menu system.

Fast Install is distributed with the Host Tool Kit and the PRO/Tool Kit on the Application Diskette Builder (ADB) diskette.

To use Fast Install, follow these steps:

1. If you have not already installed Fast Install on your Professional, do so now. Insert the ADB diskette and install it as you would any application.
2. Copy your application files to the appropriate device and directory as shown in Table 3-1. The table shows how to use the qualifiers you have specified on FILE commands in your installation file to determine the destination of your files.
3. Select Fast Install from the menu on which it was installed.
4. Fast Install prompts for the name of your application directory and for the file type (.INS or .INB) of your installation command file. Enter the information and press DO.

FAST INSTALL

5. Fast Install opens the installation command file, checks the format, and displays a list of application groups. Select the group in which you want to install your application and press DO.
6. After successful installation, you can run your application from the P/OS user interface.

Table 3-1: Application File Locations for Fast Install

FILE Command	Copy File To
FILE fname.ext or FILE fname.ext/USER	SYSDISK:[applicdir] where applicdir is the name of your application directory, which must be the same as the name of your installation command file.
FILE [dir]fname.ext or FILE [dir]fname.ext/USER	SYSDISK:[dir] where dir is the name of the directory that you specified in the FILE command in your installation command file.
FILE fname.ext/NETWORK	SYSDISK:[applicdir] where applicdir is the name of your application directory, which must be the same as the name of your installation command file.
FILE [dir]fname.ext/CLUSTER	LB000:[dir] where dir is the name of the directory that you specified in the FILE command in your installation command file.
FILE fname.ext/CLUSTER	For details on this case, see Section 3.1.

FAST INSTALL

Notes:

- If Fast Install cannot find the application directory, or the application directory does not contain an installation command file with the same name, installation fails.
- If a PRO/RMS-11 record error occurs while Fast Install is opening the installation command file, look up the error code in the PRO/RMS-11 manual set. If you cannot correct the problem, contact your local DIGITAL software support representative.
- If the installation command file contains errors in syntax, Fast Install prints them.

3.1 INSTALLING DECNET OBJECTS FOR TESTING

Applications that contain a DECnet object use a /CLUSTER file named DECNET.ODS, and possibly other /CLUSTER files, in the FILE command of the installation file. If you specify the directory in the FILE command for these files, naming conflicts can occur with the /CLUSTER files of other applications.

To avoid such conflict, the system creates a directory for these /CLUSTER files during normal installation. However, because Fast Install does not copy any files, you must manually create the directory, copy the /CLUSTER files into it, and place an entry in the DECnet object database.

Follow these steps:

- Log in to a privileged account. On P/OS Server systems, you must log in to the Server.
- Create a new directory on device LB000:.
- Copy all /CLUSTER files that do not specify a directory in the FILE command of the installation command file to the directory you created in the previous step.
- Using DCL on your Professional, issue the following commands:

```
$ ASSIGN LB000:[directory] APPL$DST
$ ASSIGN LB000:[directory] APPL$DST:
$ ASSIGN "applic_name" DST$NAME
$ RUN LB000:[ZZDECNET]DAI/COMMAND="/INS/CLS"
```


CHAPTER 4

FILE CONTROL SERVICES (FCS)

File Control Services (FCS) is a set of file management routines for use on the RSX-11 family of operating systems. It was designed for the RSX-11D operating system in the early 1970s, and has been implemented subsequently for IAS, RSX-11M, RSX-11M-PLUS, the RSX Application Migration Executive on VMS, and for the RSX run-time system on RSTS/E. FCS can be considered the precursor of Record Management Services (RMS-11), which is implemented on all of the above systems as well as on P/OS (PRO/RMS-11).

NOTE

FCS is described in detail in the *IAS/RSX-11 I/O Operations Reference Manual*, which is part of the RSX-11M/M-PLUS documentation set.

Originally, P/OS applications could use only PRO/RMS-11 for file system operations. All applications targeted for P/OS had to use PRO/RMS-11. Applications that had been written to use FCS had to be converted to use PRO/RMS-11 instead.

With the release of the PRO/Tool Kit V1.0 and the Host Tool Kit V1.7, the FCS routine library became part of the P/OS system module library. Tool Kit V3.0 also includes a vectored FCS resident library. Thus, you can move applications that use FCS to P/OS without conversion to PRO/RMS-11.

Although P/OS provides a full implementation of FCS, you are urged to always use RMS in new applications. Use FCS to port applications designed to run on RSX systems when such applications already use FCS.

4.1 SUPPORT BY HIGH-LEVEL LANGUAGES

Programs written in MACRO-11 assembly language can use FCS. Programs written in other Tool Kit languages (BASIC-PLUS-2,

File Control Services (FCS)

COBOL-81, PRO/DIBOL, FORTRAN-77 and PASCAL) do not work with FCS because the Object Time Systems (OTS) of these languages on P/OS use PRO/RMS-11 rather than FCS. The RSX versions of some of these languages will work on P/OS provided that the programs are linked using the Tool Kit version of the System Library (SYSLIB) which includes FCS.

High-level languages available from vendors other than DIGITAL can use FCS in their object-time systems. Please refer to product documentation of these software products.

4.2 USING FCS RESIDENT LIBRARY

Vectored FCS is included with the Tool Kit V3.0. It is compatible with the FCSRES for Micro/RSX and RSX-11M-PLUS. A task linked with vectored FCS can perform FCS operations on either system without relinking.

4.3 USING FCS IN-TASK VERSION

FCS is also available as modules that you can include in the task's virtual address space.

NOTE

We recommend that you use the resident library if possible, since that ordinarily enables the task to acquire new functionality without relinking.

If it is not possible or desirable to use the resident library, you can automatically include the same FCS modules from SYSLIB. However, note that the FCS modules in SYSLIB can handle only those logical names that translate to devices; the modules do not handle logical names that translate to any other fields in a file specification.

Perform the following steps if you are using the in-task version of FCS and you require the enhanced logical name capability:

1. Make copies of LB:[1,5]RSXMAC.SML and LB:[1,5]SYSLIB.OLB. These two libraries will be modified in the following steps. It will be easier to return the current capabilities if you save these libraries.

File Control Services (FCS)

The next three steps show DCL commands that you invoke to perform the modifications.

2. \$ LIBRARY/REPLACE LB:[1,5]RSXMAC.SML FCSMAC.NEW
3. \$ LIBRARY/REPLACE LB:[1,5]SYSLIB.OLB FCSEP.NEW
4. \$ LIBRARY/REPLACE/NOGLOBALS LB:[1,5]SYSLIB.OLB FCSNOEP.NEW
5. Assemble and taskbuild your programs.

The replacement modules were not included in the distributed version of SYSLIB because they require more virtual address space than the existing FCS modules.

Should it be necessary to restore RSXMAC and SYSLIB to their prior states, replacement modules are provided. Use the following DCL commands to restore RSXMAC and SYSLIB:

1. \$ LIBRARY/REPLACE LB:[1,5]RSXMAC.SML FCSMAC.OLD
2. \$ LIBRARY/REPLACE LB:[1,5]SYSLIB.OLB FCSEP.OLD
3. \$ LIBRARY/REPLACE/NOGLOBALS LB:[1,5]SYSLIB.OLB FCSNOEP.OLD

CHAPTER 5

FRAME DEVELOPMENT TOOL (FDT)

The Frame Development Tool (FDT) is a special-purpose utility for creating menu, help, and message frames. When these frames are used with the POSRES User Interface Library, they provide a user interface for your P/OS application. Designing a user interface and programming with POSRES are described in the *Tool Kit User's Guide*.

5.1 FDT OVERVIEW

Creation or modification of a frame file follows these general steps:

1. Plan your frames. You should have a good idea of what frames your menu structure and help structure will require before you use FDT.
2. Start FDT and specify a filename for the frame definition file. If you are creating a new file, specify the type of frames (menu, help or message) you want to create. This leaves you at File Command level.
3. Enter a File Command. When creating a new frame file, you would typically use the **Add** command. When modifying a frame file, you might use commands such as **Add, Delete, Modify**.
4. If you are adding or modifying a frame, enter a Frame Command. A frame is defined by three separate forms that you fill in or modify. While entering data in a form, you use a special set of keypad keys.

When you are finished with a form, you press **ENTER**. When you are finished adding or modifying the frame, you enter the **Save** or **Quit** command.

FDT OVERVIEW

5. You are back at File Command level, where you can repeat Steps 3 and 4 with another frame, or you can supply a File Command that uses the whole frame file, such as **Convert**, **List**, **Report**, **Window**.

When you are finished with the file, you enter **Save** or **Quit**.

The following sections describe how to run FDT, the File commands, the Frame commands, and the forms in more detail. If you would like to try out a sample FDT editing session before moving on, turn to Section 5.11.

5.2 INVOKING FDT

FDT is invoked in different ways depending on whether you are running it on a VAX, PDP-11, or your Professional. The commands required to start it are discussed in the following sections.

In each case, FDT requires the name of the frame file. If you do not include the filename, FDT prompts for one. The default file type is .DAT. If the file does not exist, FDT assumes that you are creating a new file and prompts for the type of frame that you will store in the file.:

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

You enter H, M, or S.

NOTE

You can only store frames of the same type in a file. You must create separate files for help, messages, and single-choice menus.

5.2.1 Invoking FDT on VAX/VMS

For convenience, edit your LOGIN.COM file and insert the following symbol definition:

```
$ FDT ::= $FDT
```

Once the symbol is defined, you can invoke FDT by typing:

```
$ FDT filename
```

INVOKING FDT

5.2.2 Invoking FDT on RSX-11M/M-PLUS (DCL)

If FDT is installed on your system as "...FDT", enter:

```
$ FDT filename
```

If FDT is not an installed task, enter:

```
$ RUN $FDT
```

5.2.3 Invoking FDT in PRO/Tool Kit

You must install the FDT task. You might edit your start-up command file with the appropriate command to install FDT from the directory in which it resides.

To start FDT, enter:

```
$ FDT filename
```

If FDT is not an installed task, enter:

```
$ RUN $FDT
```

5.3 FILE COMMANDS

Some file commands operate on all frames in the frame file. Other file commands operate on individual frames; these require that you supply the name of the frame (called the frame identifier, or frameid) with the command. File commands are entered in response to the prompt:

File Command:

With the exception of the QUIT command, you can abbreviate any file command to a single character. If you enter a command and omit required parameters, FDT displays prompts requesting them.

While working at file command level, the current file type and filename are displayed in a message at the top of your screen.

NOTE

Do not embed spaces in frame identifiers. POSRES routines cannot retrieve them correctly at run time.

FILE COMMANDS

5.3.1 ADD

The ADD command begins the creation of a new frame. New frames can be added to a new or existing frame definition file.

Format:

ADD frameid

frameid A string of up to eight alphanumeric characters that identifies the new frame.

Notes

- A frame identifier must consist of alphanumeric ASCII characters. FDT converts all characters to uppercase.
- If you are editing a help file, FDT prompts as follows:

Help (M)enu or (T)ext?

Respond with M for a help menu or T for a help text frame.

- FDT displays a Profile form, which is the first form you must fill in when you are creating a new frame.

5.3.2 CONVERT

The CONVERT command creates a converted frame file that can be read directly by POSRES menu services during program execution.

Format:

CONVERT output-file

output-file The name of the converted frame file.

Notes:

- The default file types for converted files are:

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

FILE COMMANDS

- All required fields on frames must be filled in before the frame is converted. If a frame lacks required information, an error message appears and the frame is not converted.
- FDT displays these messages:

x frames converted, y frames not converted.
The largest frame, "<framename>", is n (m octal) bytes long.

The numbers x, y, and n, are decimal. The number m is the octal equivalent of n. If there are unconverted frames, other messages will appear.

Record the size of the largest frame converted. You can use this number to compute the minimum amount of buffer space required for frames of the type stored in this file (see the discussions of the Professional Application Builder and the P/OS User Interface Services in the *Tool Kit User's Guide*).

5.3.3 DELETE

The DELETE command deletes the specified frame from the current file.

Format:

DELETE frameid

frameid The identifier of the frame to be deleted.

Notes:

- Before deleting the frame, FDT shows the Display form for the specified frame and prompts:

Delete this frame?

Enter **YES** to delete the frame; enter **NO** to avoid deleting it. Control returns to the File Command prompt.

5.3.4 EXIT

The EXIT command saves the current frame definition file on disk and exits from FDT. (The EXIT command is identical to the SAVE command.)

FILE COMMANDS

Format:

EXIT

Notes:

- If the file already exists, it is assigned the next higher version number.
- After saving a file, FDT displays a confirmation message naming the file saved and returns control to system command level.

5.3.5 FILE

The FILE command terminates processing of the current file and opens a new frame file.

Format:

FILE filename

filename The name of the new frame file. The default file type is .DAT.

Notes:

- If you have modified the current file, FDT prompts:
Save current file?
Enter YES to save it, or NO to discard it.
- If you are creating a new frame file, FDT prompts for the type of file you want to create:
Create (H)elp, (M)essage, or (S)ingle-choice menu file?
Enter H, M, or S; or press RETURN to exit.

5.3.6 HELP

The HELP command displays a list of FDT file commands.

Frame Development Tool

Format:

HELP

5.3.7 LIST

The LIST command displays a list of the frames in the current file, by frame identifier.

Format:

LIST

Notes:

- The frames are listed in the order they were created, although the order may change as a result of the addition or deletion of frames.

5.3.8 MODIFY

The MODIFY command begins the process of modifying an existing frame in the frame file.

Format:

MODIFY frameid

frameid The identifier for the frame to be modified.

Notes:

- FDT displays a message indicating the frame is available for modification and displays the "Frame Command" prompt.

5.3.9 NAME

The NAME command renames a frame.

FILE COMMANDS

Format:

NAME old-frameid new-frameid

old-frameid Is the frameid that is to be renamed.

new-frameid Is the new name (up to eight ASCII alphanumeric characters).

5.3.10 QUIT

The QUIT command returns control to system command level without saving the current file. All changes made to the current file in the current editing session are discarded. (All of the characters in the QUIT command must be typed.)

Format:

QUIT

5.3.11 REPORT

The REPORT command creates a printable file containing information about each frame in the current frame file.

Format:

REPORT filename

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

Notes:

- The default file type for the report file is .RPT.
- If you do not specify a filename, FDT uses the current frame filename by default.

5.3.12 SAVE

The SAVE command saves the current frame definition file on disk and exits from FDT. (The SAVE command is identical to the EXIT command.)

Frame Development Tool

Format:

SAVE

Notes:

- If the file already exists, it is assigned the next higher version number.
- After saving a file, FDT displays a confirmation message naming the file saved and returns control to system command level.

5.3.13 WINDOW

The WINDOW command converts the current file to a Synergy source frame file. (See the *Synergy Programmer's Manual*.)

Format:

WINDOW filename

Notes:

- The default file type is .SFF.
- The text that appears in the Display form becomes the text in the Synergy window. Other text fields, such as action strings, become option values or comments in the Synergy frame file.
- Frame identifiers are not changed. (The Synergy Frame Compiler Tool diagnoses numeric names and names that are longer than six characters.)
- The conversion cannot be complete, since the frame files of FDT and FCT do not provide identical functionality. However, FCT will diagnose most of the problems during the first attempt to compile the Synergy frame file. If you encounter strange errors, use the CONVERT command to discover and correct missing fields before trying the WINDOW command a second time.

Frame Development Tool

- Combine the SFF files that you derive from your MNU and HLP files before using the Synergy Frame Compiler, so that references to help frames will be resolved.

5.4 FRAME COMMANDS

Frame commands allow you to modify frames. They are entered in response to the prompt:

Frame Command:

With the exception of the QUIT command, you can abbreviate any frame command to a single character. If you enter a command and omit required parameters, FDT displays prompts requesting them.

5.4.1 ACTION

The ACTION command invokes Action forms for the options on the current menu. An Action form specifies a keyword, a help frame pointer, and (for single-choice menus only) an action string for each option.

Format:

```
ACTION ALL  
NEW  
option
```

ALL Specifies all options in the current menu. ALL is the default.

NEW Specifies only those options for which you have not yet entered an action form.

option Specifies a particular option by ordinal number (1 to 12).

Notes:

- Action information can be assigned only to menus (not text frames).

Frame Development Tool

- Each option must be assigned action information.
- If you specify an option that has already been assigned action information, the current description is displayed on the Action form. You can then enter new information or change existing information.

5.4.2 DISPLAY

The DISPLAY command invokes a Display form for the current frame. The Display form contains the text that the user sees on the screen.

Format:

DISPLAY

Notes:

- A Display form for a menu contains the title, explanatory text, options, and prompt.
- A Display form for a help text frame contains the title and text.
- A Display form for a message frame contains the message text.

5.4.3 EXIT

The EXIT command saves the current frame on disk in the current file, displays a confirmation message, and returns control to file editing level. (The EXIT command is identical to the SAVE command.)

Format:

EXIT

CAUTION

This command saves the current frame, not the file. To save the file, enter an EXIT or SAVE command at file command level when you are finished editing.

Frame Development Tool

5.4.4 HELP

The HELP command displays a list of FDT frame commands.

Format:

HELP

Notes:

- If you request help at the Frame Command prompt while you are working on a text or message frame, the ACTION command is not listed.

5.4.5 PROFILE

The PROFILE command displays a Profile form for the current frame. The Profile form contains data that documents the frame and describes the operation of the frame to menu services. This information is not visible to the end user.

Format:

PROFILE

Notes:

- The Profile form for a single-choice menu contains a frame description, a global help frame pointer, the default option, and the global action string.
- The Profile form for a help menu contains a frame description, a previous help frame pointer, and the default option.
- The Profile form for a help text frame contains a frame description, the frame's screen location, a previous help frame pointer, and a next help frame pointer.
- The Profile form for a message frame contains a frame description.

5.4.6 QUIT

The QUIT command returns control to file command level without

Frame Development Tool

saving the current frame. Any changes made to the frame are discarded.

Format:

QUIT

Note:

All four characters of the QUIT command must be entered.

5.4.7 SAVE

The SAVE command saves the current frame on disk in the current file, displays a confirmation message, and returns control to file editing level. (The SAVE command is identical to the EXIT command.)

Format:

SAVE

CAUTION

This command saves the current frame, not the file. To save the file, enter an EXIT or SAVE command at file command level when you are finished editing.

5.5 PROFILE, DISPLAY, AND ACTION FORMS

FDT provides access to frame descriptions by means of forms. When you enter the ADD command to create a new frame, FDT displays a series of forms for you to fill in. When you enter the MODIFY command to edit an existing frame, FDT allows you to specify which forms you want to modify.

As shown in Figure 5-1, there are three types of forms:

PROFILE, DISPLAY, AND ACTION FORMS

- **The Profile Form**

The Profile form (invoked by the PROFILE command) contains data that describes the purpose and operation of the frame to POSRES and to your task. This information is not visible to the user.

- **The Display Form**

The Display form (invoked by the DISPLAY command) contains fields that correspond to the fields in the frame. It describes how the form will appear to the user.

- **The Action Form**

The Action form (invoked by the ACTION command) is for menu frames only. It specifies a keyword, help frame pointer, and (for single-choice menus) an action string for each option.

PROFILE, DISPLAY, AND ACTION FORMS

FORMS

Profile for Single Choice Menu MAIN

<p>Frame Description</p> <p>[This is the main menu for the elementary education application.]</p>
<p>Global Help Frame [MENU]</p> <p>Default Option [3]</p> <p>Global Action String</p> <p>[DATABASE = ELEM]</p>

Display for Single Choice Menu MAIN

<p>ELEMENTARY EDUCATION APPLICATION</p> <p>[This application offers elementary education in seven fields of study. Select one of the courses listed here.]</p> <p> BIOLOGY COMPUTER SCIENCE GEOGRAPHY GOVERNMENT HISTORY LITERATURE MATHEMATICS </p> <p>[Make a selection and press the DO key:]</p>
--

Action Number 3 for Single Choice Menu MAIN

<p>Description: GEOGRAPHY</p>
<p>Action Description</p> <p>[This is the description of the Geography option.]</p>
<p>Option Keyword [GEO]</p> <p>Option Help Frame [HELPGEO]</p>
<p>Option Action String</p> <p>[GEOG0010]</p>

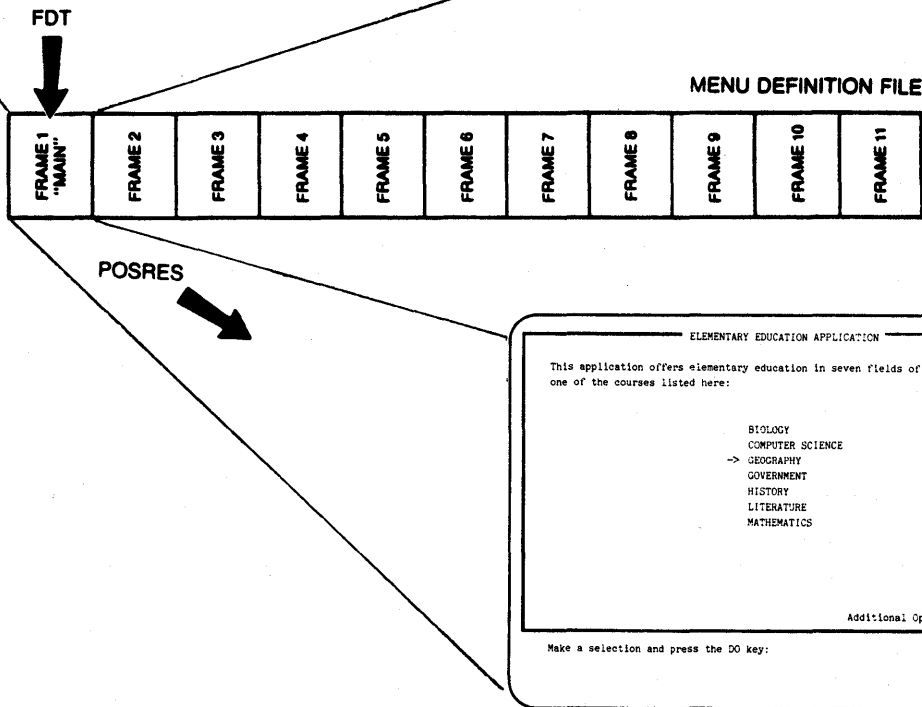


Figure 5-1: Forms for a Single-Choice Menu

PROFILE, DISPLAY, AND ACTION FORMS

Each form contains one or more fields. The meaning of some fields is clear from the field name or prompt. Other fields, such as help frame pointers, are interrelated and are more complex.

Some fields are optional; others are required. If you fail to fill in a required field, FDT accepts the frame. However, when you try to CONVERT the frame file, an error occurs and the conversion fails.

You can use your keyboard's numeric keypad and ARROW keys to manipulate text within a field. Figure 5-2 shows the layout of the numeric keypad. Like other screen editors, FDT uses an insert mode for entering text. There is no insert command. To enter text, simply type on the main keyboard.

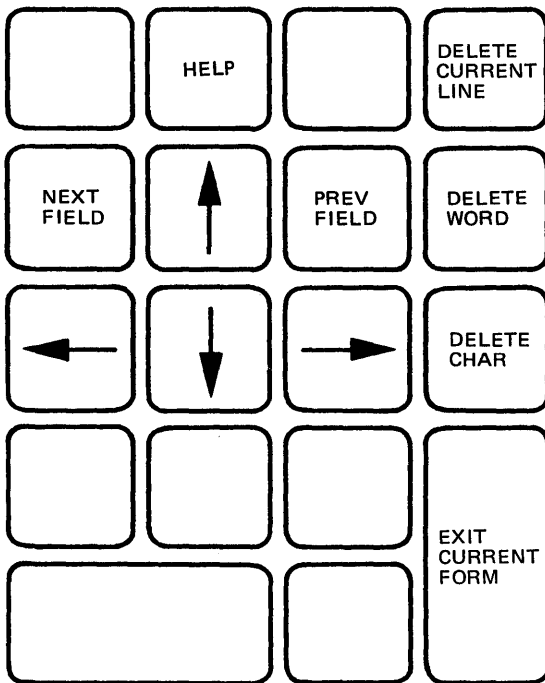


Figure 5-2: FDT Screen Editor Keypad

PROFILE, DISPLAY, AND ACTION FORMS

The keypad keys operate as follows:

ARROW keys

Move the cursor forward, backward, up, or down, within and between fields. You can use the 6, 4, 8, and 2 keys (respectively) on the numeric keypad for the same purposes.

HELP

Displays a description of the field the cursor currently points to. If pressed again, the HELP key displays a description of the keypad keys.

**DELETE
CURRENT LINE**

Deletes from the cursor to the end of the line.

DELETE WORD

Deletes from the cursor to the next word.

DELETE CHAR

Deletes the character the cursor is on.

**NEXT
FIELD**

Positions the cursor at the beginning of the next field on the form.

**PREV
FIELD**

Positions the cursor at the beginning of the previous field on the form.

**EXIT CURRENT
FORM**

Returns control to frame editing command level unless you are filling in multiple Action forms, in which case the next Action form for the menu (if there is one) is displayed.

CREATING A SINGLE-CHOICE MENU

5.6 CREATING A SINGLE-CHOICE MENU

5.6.1 The Profile Form

Profile for Single Choice Menu [IDENTIFIER]

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

Figure 5-3: Profile Form for Single-Choice Menu

Field	Type	Max. Size	Purpose
Frame Description	Optional	Six lines of 72 characters	Enter a description of the overall purpose and operation of the menu. For example, you can document the content and meaning of the global action string.
Global Help Frame	Optional	Eight alphanumeric characters	Specify the help frame to be displayed when the selector is at the rest position or if no option frame pointer exists.
Default option	Optional	Two digits	Specify the ordinal number of the option on which the selector is to begin when the menu is displayed. If you leave this field blank, the selector begins at the rest position.

CREATING A SINGLE-CHOICE MENU

<u>Field</u>	<u>Type</u>	<u>Max. Size</u>	<u>Purpose</u>
Global action string	Optional	72 characters	Specify a string that associates some useful data with this menu. When the MFRAME routine reads this menu into memory, it returns the global action string to your task, which can use or ignore it.

CREATING A SINGLE-CHOICE MENU

5.6.2 The Display Form

Display for Single Choice Menu [IDENTIFIER]

The diagram shows a rectangular display form for a single-choice menu. At the top, there is a title bar containing the text "TITLE TEXT" between square brackets. Below this is a section for "EXPLANATORY TEXT", also between square brackets. The main body of the form is for "OPTION DESCRIPTION", which is flanked by square brackets on both sides. At the bottom of the form is a "PROMPT" field, also flanked by square brackets.

Figure 5-4: Display Form for Single-Choice Menu

<u>Field</u>	<u>Type</u>	<u>Max. Size</u>	<u>Purpose</u>
Title	Required	64 alphanumeric characters	Enter the title to be displayed on the first line of the menu.
Explanatory text	Optional	Three lines of 72 characters	Enter text introducing the user to the options listed on this menu.
Option descriptions	Required	Twelve lines of 64 characters	Enter the option list. You must specify at least one option.
Prompt	Required	72 characters	Enter a string that requests the user to make a selection.

CREATING A SINGLE-CHOICE MENU

5.6.3 The Action Form

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

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

Figure 5-5: Action Form for Single-Choice Menu

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.

<u>Field</u>	<u>Type</u>	<u>Max. Size</u>	<u>Purpose</u>
Action Description	Optional	Two lines of 72 characters	Enter a description of the overall purpose and operation of the option. For example, you can document the content and meaning of the option action string.
Option keyword	Required	30 characters	Enter any contiguous substring of the option. This keyword appears in bold when the menu is displayed in order to indicate which characters the user can type to select this option.
Option Help Frame	Optional	Eight alphanumeric characters	Specify the help frame to be displayed when the user presses the HELP key while the selector is positioned at this option.

CREATING A SINGLE-CHOICE MENU

<u>Field</u>	<u>Type</u>	<u>Max. Size</u>	<u>Purpose</u>
Option action string	Optional	72 characters	Specify a string that associates some useful data with this option. When the user selects an option, the MENU routine returns this string to your task, which can use or ignore it.

CREATING A HELP MENU

5.7 CREATING A HELP MENU

5.7.1 The Profile Form

Profile for Help Menu [IDENTIFIER]

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

Figure 5-6: Profile Form for Help Menu

<u>Field</u>	<u>Type</u>	<u>Max. Size</u>	<u>Purpose</u>
Frame Description	Optional	Six lines of 72 characters	Enter a description of the overall purpose and operation of the menu. For example, you can document how it fits into the larger help structure.
Previous Help Frame	Optional	Eight alphanumeric characters	Specify the help frame to be displayed when the user presses the PREV SCREEN key.
Default option	Optional	Two digits	Specify the ordinal number of the option on which the selector is to begin when the menu is displayed. If you leave this field blank, the selector begins at the rest position.

CREATING A HELP MENU

5.7.2 The Display Form

Display for Help Menu [IDENTIFIER]

The diagram shows a rectangular frame representing the display form. It is divided into four sections by brackets: 'TITLE TEXT' at the top, 'EXPLANATORY TEXT' in the upper middle, 'OPTION DESCRIPTION' in the lower middle, and 'PROMPT' at the bottom. The 'EXPLANATORY TEXT' and 'OPTION DESCRIPTION' sections are each enclosed in a vertical bracket on both sides, indicating they are optional or multi-line fields.

Figure 5-7: Display Form for Help Menu

<u>Field</u>	<u>Type</u>	<u>Max. Size</u>	<u>Purpose</u>
Title	Required	64 alphanumeric characters	Enter the title to be displayed on the first line of the menu.
Explanatory text	Optional	Three lines of 72 characters	Enter text introducing the user to the options listed on this menu.
Option descriptions	Required	Twelve lines of 64 characters	Enter the option list. You must specify at least one option.
Prompt	Required	72 characters	Enter a string that requests the user to make a selection.

CREATING A HELP TEXT FRAME

5.8 CREATING A HELP TEXT FRAME

5.8.1 The Profile Form

Profile for Help Text [IDENTIFIER]

Frame Description <div style="border: 1px solid black; height: 60px; margin: 5px 0;"></div>
Frame Location [FULL] Previous Help Frame []] Next Help Frame []]

Figure 5-9: Profile Form for Help Text Frame

Field	Type	Max. Size	Purpose
Frame Description	Optional	Six lines of 72 characters	Enter a description of the overall content of the frame.
Frame Location	Required	Specific options	Specify "FULL", "TOP", or "BOTTOM", according to where you want this frame to appear. A full frame has 16 lines of text; a top or bottom frame has 8 lines. The default is "FULL". To change it, position the cursor on the "F" and press the DELETE WORD key on the keypad.
Previous Help Frame	Optional	Eight alphanumeric characters	Specify the help frame to be displayed when the user presses the PREV SCREEN key.
Next Help Frame	Optional	Eight alphanumeric characters	Specify the help frame to be displayed when the user presses the NEXT SCREEN key.

CREATING A HELP TEXT FRAME

5.8.2 The Display Form

Display for Help Text [IDENTIFIER]

[TITLE TEXT]
[Help Text]

Figure 5-10: Display Form for Help Text Frame

Full screen and top or bottom half help text frames 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.

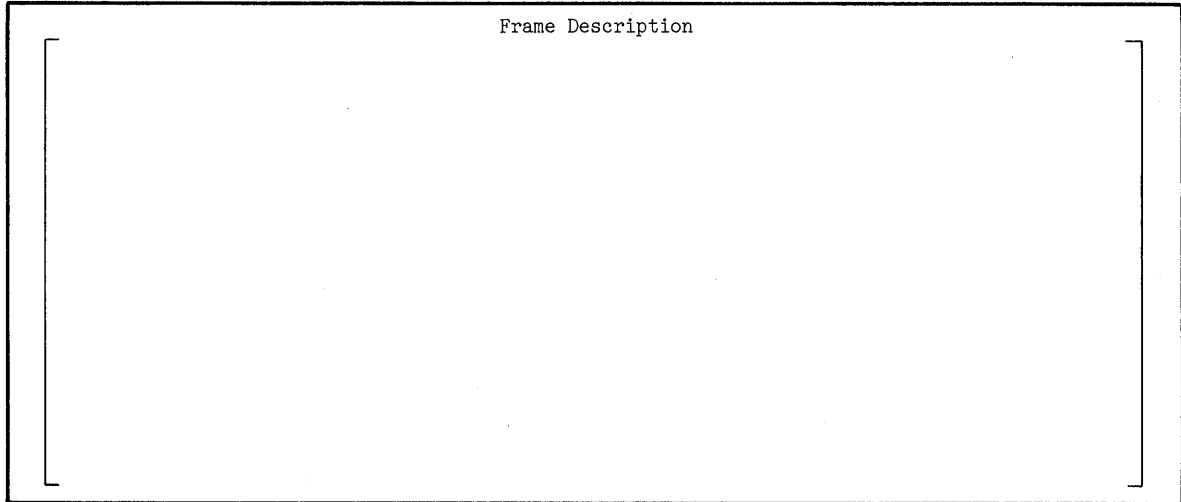
<u>Field</u>	<u>Type</u>	<u>Max. Size</u>	<u>Purpose</u>
Title	Required	64 alphanumeric characters	Enter the title to be displayed on the first line of the frame.
Help text	Required	16 lines of 72 characters	Enter at least one line of text that provides help on some aspect of your application.

CREATING A MESSAGE TEXT FRAME

5.9 CREATING A MESSAGE TEXT FRAME

5.9.1 The Profile Form

Profile for Message [IDENTIFIER]



The image shows a rectangular profile form for a message frame. At the top center of the form, the text "Frame Description" is displayed. The rest of the form is a large, empty rectangular area, bounded by a thin black line, intended for entering a description of the message's contents and purpose.

Figure 5-11: Profile Form for Message Frame

<u>Field</u>	<u>Type</u>	<u>Max. Size</u>	<u>Purpose</u>
Frame description	Optional	17 lines of 72 characters	Enter text that documents the contents and purpose of the message.

5.9.2 The 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.

<u>Field</u>	<u>Type</u>	<u>Max. Size</u>	<u>Purpose</u>
Message text	Required	21 lines of 79 characters	Enter the message text. You must enter at least one line.

RESOLVING ERRORS

5.10 RESOLVING ERRORS

All FDT user errors produce a short beep from the keyboard. For example, while you are filling in forms, if you press a key that is not on the FDT keypad or attempt to type over field boundaries, the keyboard beeps and the key action is ignored.

Errors other than the example given above produce a descriptive error message following the keyboard beep. FDT Error messages are described in Appendix B.

Certain invalid keystrokes disrupt the FDT screen display. For example, a line feed entered while an FDT form is displayed scrolls the screen upward. Also, a backspace typed to the File or Frame Command prompt allows typing over the prompt, causing valid commands typed at this point to be rejected. If you enter an invalid keystroke that alters a form display, type **CTRL/W** to refresh the screen and continue. If you enter an invalid keystroke that alters a File or Frame Command prompt, enter valid commands until one is accepted.

5.11 SAMPLE TERMINAL SESSION

The following is a sample of a terminal session using the Frame Development Tool on an RSX-11M/M-PLUS system. It is recommended that you use the single-choice menu shown in Figure 5-1 to fill in the forms.

```
  $ RUN $FDT
```

FDT starts and prompts for a filename. In this example, FDT creates a new file.

```
Filename: SAMPLE
```

FDT prompts for the type of frames to be stored in the file.

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

The file **SAMPLE.DAT** will contain single-choice menus. FDT prompts for a file command.

```
File Command: ADD MAIN
```

The **ADD** command creates a new frame called **MAIN**. Because it is a new frame, FDT displays a Profile form. When you finish the form, press the **ENTER** key. **MAIN** is now the current frame. FDT prompts for a frame command.

SAMPLE TERMINAL SESSION

Frame Command: **DISPLAY**

The **DISPLAY** command creates a Display form for **MAIN**. When you finish the form, press the **ENTER** key. **FDT** prompts for another frame command.

Frame Command: **ACTION**

The **ACTION** command creates Action forms for each option on the display form for **MAIN**. When you finish an Action form, press the **ENTER** key. When all the Action forms are done, **FDT** prompts for another frame command. Suppose that you want to correct a mistake on the Action form for the third option.

Frame Command: **ACTION 3**

FDT displays the action form for option three with the information you entered before. When you finish the form, press the **ENTER** key. **FDT** prompts for another frame command.

Frame Command: **SAVE**

MAIN is saved on disk, and a confirming message is displayed. **FDT** prompts for a file command.

File Command: **SAVE**

FDT saves the file **SAMPLE.DAT**, which contains **MAIN**, on disk and returns control to host system command level.

Suppose that you decide to modify the Profile form for **MAIN**.

\$ **RUN \$FDT**

Filename: **SAMPLE**

File Command: **MODIFY MAIN**

FDT extracts **MAIN**, displays a message confirming that **MAIN** is available for modification, and prompts for a frame command.

Frame Command: **PROFILE**

The **PROFILE** command invokes the Profile form for **MAIN**, the current frame. The information you entered before is still there. When you finish the form, press the **ENTER** key. **FDT** prompts for a frame command.

Frame Command: **SAVE**

SAMPLE TERMINAL SESSION

FDT saves MAIN with the new Profile form and prompts for a file command.

File Command: QUIT

FDT discards all changes made to the file SAMPLE.DAT (including the new Profile form) and returns control to host system command level.

CHAPTER 6

INSTALLATION COMMAND LANGUAGE

Every application requires an *installation command file*. This file, whose filename contains either a .INB or .INS extension, is important for creating application diskettes and for installing, running, and removing an application.

The system uses an installation file in the following ways:

- **Building Application Diskettes**

When you start the Application Diskette Builder (ADB) to build the distribution diskettes for your application, ADB reads the installation file to identify the files to copy to the application diskette(s).

- **Copying Application to Public Library**

The system manager on a P/OS system can copy applications into the Public Library. The installation file directs the transfer of files from application diskette to the Public Library on hard disk.

- **Installing an Application**

When the user installs your application, P/OS reads the installation file to identify the files to copy to the hard disk at installation time. If you have tasks that should execute during the installation process, they can be run by the EXECUTE command in the installation file.

- **Running an Application**

When a user runs your application, P/OS reads the installation file to identify task images (it adds the names to its list of executable tasks), to open any Menu, Message, and Help frame files (specified in ASSIGN commands), and to start the first application task.

Installation Command Language

- **Removing an Application**

When the user removes your application, P/OS reads the installation file to identify the files to delete from the hard disk.

- **Deleting Application from Public Library**

When the system manager deletes applications from the Public Library, the system reads the installation file to determine which application files to delete from the Library.

6.1 INSTALLATION COMMAND FILE FORMAT

An installation command file is an ordinary text file that you can create with any editor. Some Tool Kit languages provide a facility for creating a general-purpose installation file. Note the following requirements for command files:

- Command lines can be up to 132 characters long.
- There is one command per line.
- Commands must appear in a prescribed order.

The format of an installation file is shown in Figure 6-1. Boldface indicates a mandatory command.

NOTES	COMMAND LINES
	! A sample install file.
	NAME "name"
	FILE filespec/option1/option2
1	MOUNT volume
	EXECUTE taskspec/option1/option2/option3
	INSTALL filespec/option1/option2/NOREMOVE/FIX
	ASSIGN MESSAGE filespec/option
	ASSIGN MENU filespec/option
	ASSIGN HELP filespec/option frameid
	ASSIGN LOGICAL name "equiv"
2	RUN taskname

Figure 6-1: Installation Command File Format

INSTALLATION COMMAND FILE FORMAT

Notes to Figure 6-1

1. The MOUNT command is required for multiple-diskette applications. (See Section 6.11.)
2. ADB and Fast Install generate a warning message if you omit the RUN command. Although P/OS does not require the RUN command to be present, most applications must use it.

The order of the FILE commands determines the order in which files are copied from the application diskettes or Public Library. The order of the INSTALL commands determines the order in which task image files are processed when an application is started.

6.1.1 Using .INB and .INS Files

For applications that will run on P/OS V3.0 you should use the .INB extension on your installation filename. This will allow your application to take advantage of additional features present in P/OS V3.0 and later versions.

To run on V2.0A and earlier P/OS systems, your application must supply a .INS file. These systems do not recognize .INB files as installation files.

You can supply both an .INB and .INS file for any application. In general, we recommend that you supply both forms of the installation file so that your application is compatible with the maximum number of systems.

The following are valid only in .INB files:

- ASSIGN MESSAGE command
- /USR, /NTW, and /CLS qualifiers on the EXECUTE command
- /USER, /NETWORK, and /CLUSTER qualifiers on any file specification
- /TASK=name qualifier on the INSTALL command

6.1.2 Using /USER, /NETWORK, and /CLUSTER in Your .INB File

You use the /USER, /NETWORK, and /CLUSTER qualifiers on any command whose format includes a file specification. These

INSTALLATION COMMAND FILE FORMAT

qualifiers tell P/OS where on the system each application file normally resides; thus they allow you to control access to the files.

A description of each qualifier follows:

- **/USER--Separate file available for each user**

Application files with the /USER qualifier are available separately to each user who installs the application. Such files cannot be shared among different users. An example of a /USER file is a file containing individual set-up information. On P/OS V2.0A or earlier systems, all application files were treated as /USER files.

Files given the /USER qualifier always reside on the device SYSDISK:, in either a dynamically-created directory named [ZZAPnnnnn] or in the directory specified in the installation command.

During run time, your application can refer to any /USER files either by using the logical APPL\$USER: in conjunction with the filename and extension (preferred), or by using the complete file specification. However, for compatibility with P/OS V2.0A and earlier systems, you must instead translate the logical APPL\$DIR and use the resulting device and directory in your file specification.

Sample references to a /USER file:

```
APPL$USER:SETUP.DAT
SYSDISK:[MYAPPLDIR]SETUP.DAT
```

- **/NETWORK--file can be shared among users**

Application files with the /NETWORK qualifier can be shared among users executing the same application. This saves disk space, since the system can use a single copy of the file for all users executing your application on a particular system. Most read-only files are good candidates for the /NETWORK qualifier. Also, a read-write database could be a /NETWORK file, provided that your application properly manages any concurrent accesses.

Files having the /NETWORK qualifier reside on the disk specified by the system manager when they are copied from diskette. The system places /NETWORK files in a dynamically-created directory of the form [ZZAPnnnnn.ZZPUBAP].

INSTALLATION COMMAND FILE FORMAT

During run time, your application must refer to any /NETWORK files by using the logical APPL\$NETWORK: in conjunction with the filename and extension.

Sample reference to a /NETWORK file:

```
APPL$NETWORK:DATABASE.DAT
```

- **/CLUSTER--file can be shared among different applications**

Application files with the /CLUSTER qualifier can be shared among users executing *different applications*. Only a Server manager can install /CLUSTER files on a system.

As with /NETWORK files, a read-only file or a properly-managed read-write database is a good candidate for the /CLUSTER qualifier. A callable editor task would be a very good candidate.

Files given the /CLUSTER qualifier reside on LB000:, the boot device of the Server that down-line loaded the system on which your application is running.

Usually, you specify the directory in the installation command. However, if you omit the directory, P/OS places the file in a dynamically-created directory of the form [ZZCLnnnn]. In general, we strongly recommend that you supply the directory, since your application cannot determine the dynamically-created name at run time. See Section 3.1 for an exception that applies to applications using DECnet.

During run time, your application must refer to any /CLUSTER files by using the actual file specification. No logical names are available for /CLUSTER files.

Sample reference to a /CLUSTER file:

```
LB000:[ZZPRQDCL]PIP.TSK
```

Table 6-1 summarizes the three qualifiers.

INSTALLATION COMMAND FILE FORMAT

Table 6-1: Summary of /USER, /NETWORK, and /CLUSTER Qualifiers

Qualifier	Location	Logical Name
/USER	<p>SYSDISK:[ZZAPnnnnn] or SYSDISK:[directory]</p> <ul style="list-style-type: none"> ● SYSDISK: points to the user home. ● nnnnn is created dynamically by the system. ● directory is specified in the installation command. 	APPL\$USER:
/NETWORK	<p>device:[ZZPAnnnnn.ZZPUBAP]</p> <ul style="list-style-type: none"> ● device is the device chosen by the workstation or Server manager. ● nnnnn is created dynamically by the system. 	APPL\$NETWORK:
/CLUSTER	<p>LB000:[directory] or LB000:[ZZCLnnnnn]</p> <ul style="list-style-type: none"> ● LB000: points to the system library device. ● directory is specified in the installation command. ● nnnnn is created dynamically by the system. 	None

INSTALLATION FILE ERRORS

6.2 INSTALLATION FILE ERRORS

Errors resulting from an incorrect installation command file are documented as follows:

- Errors returned during operation of ADB or Fast Install are documented in Appendix A.
- Errors returned to the end user when installing, removing, or transferring your application to or from the Public Library are documented in Appendix E.

ASSIGN HELP

6.3 ASSIGN HELP

This command assigns a filename as the default help file for your application. If you have a default help file, you need not call the Open Help File service routine at run time.

This command is used only when your application is started by the user.

Format:

ASSIGN HELP filespec/option frameid

filespec Specifies the name of a help frame file.

option Is optional and can only be specified in .INB files. It can be one of /USER, /NETWORK, or /CLUSTER. The qualifier you specify indicates where the help frame file resides. See Section 6.1.2 for details.

frameid Is a string of up to eight characters that specifies the default help frame.

Notes:

- You must use the same filespec that you used in the the FILE command.
- You can have only one ASSIGN HELP command per installation file.
- The ASSIGN HELP command is optional. However, we strongly recommend that you supply it in your installation command file. If you do not supply an ASSIGN HELP command, POSRES does not provide any default help file.
- If you have multiple help files, always use ASSIGN HELP to specify your primary help file as the default. In an HFILE call to open a different help file, specify only the file name of the other help file, not the complete file specification. POSRES obtains any missing fields of the file specification from the value given in ASSIGN HELP.

ASSIGN LOGICAL

6.4 ASSIGN LOGICAL

This command defines a logical name at run time. (See the *Tool Kit User's Guide* and *P/OS System Reference Manual* for information about logical names.) This command is used only when your application is started by the user.

Format:

ASSIGN LOGICAL name "equiv"

name Is a string that specifies the logical name that is to be defined.

equiv Is a string that specifies the equivalence name.

Notes:

- The ASSIGN LOGICAL command is optional.
- Multiple ASSIGN LOGICAL commands can be used.

ASSIGN MENU

6.5 ASSIGN MENU

This command assigns a file as the default menu file for your application. If you have a default menu file, you need not call the Open Menu File (MFILE) service routine at run time. This command is used only when your application is started by the user.

Format:

ASSIGN MENU filespec/option

filespec Specifies a menu frame file.

option Is optional and can only be specified in .INB files. It can be one of /USER, /NETWORK, or /CLUSTER. The qualifier you specify indicates where the menu file resides. See Section 6.1.2 for details.

Notes:

- You must use the same filespec that you used in the the FILE command.
- You can have only one ASSIGN MENU command per installation file.
- The ASSIGN MENU command is optional. However, we strongly recommend that you supply it in your installation command file. If you do not supply an ASSIGN MENU command, POSRES does not provide any default menu file.
- If you have multiple menu files, always use ASSIGN MENU to specify your primary menu file as the default. In an MFILE call to open a different menu file, specify only the file name of the other menu file, not the complete file specification. POSRES obtains any missing fields of the file specification from the value given in ASSIGN MENU.

ASSIGN MESSAGE (.INB FILE ONLY)

6.6 ASSIGN MESSAGE (.INB FILE ONLY)

This command assigns a file as the default message file for your application. This command is used only when your application is started by the user.

Format:

ASSIGN MESSAGE filespec/option

filespec Specifies a message frame file.

option Is optional and can only be specified in .INB files. It can be one of /USER, /NETWORK, or /CLUSTER. The qualifier you specify indicates where the message file resides. See Section 6.1.2 for details.

Notes:

- You must use the same filespec that you used in the the FILE command.
- You can have only one ASSIGN MESSAGE command per installation file.
- This command is valid only for P/OS V3.0 or later versions.
- You can have only one ASSIGN MESSAGE command per installation file.
- The ASSIGN MESSAGE command is optional. However, we strongly recommend that you supply it in your installation command file. If you do not supply an ASSIGN MESSAGE command, POSRES uses APPL\$DIR as the default device and directory specification for all message file operations.
- If you have multiple message files, always use ASSIGN MESSAGE to specify your primary message file as the default. In an RDMSG call to open a different message file, specify only the file name of the other message file, not the complete file specification. POSRES obtains any missing fields of the file specification from the value given in ASSIGN MESSAGE.

COMMENT

6.7 COMMENT

An exclamation point (!) indicates that the line is a comment. The exclamation point must be the first nonblank character on the line.

Format:

! text

6.8 EXECUTE

This command causes a specified task to execute during the following operations:

- Install application
- Remove application
- Copy application to the Public Library
- Delete application from the Public Library

If the task exits with status greater than one, the install or copy operation aborts.

P/OS acts on the EXECUTE command as soon as it is read from the installation file. Thus, the position of the EXECUTE command relative to FILE commands is significant, as explained below.

Format:

EXECUTE taskspec/option1/option2/option3

taskspec Is the task image file in one of the following forms for both .INB and .INS files:

[directory]file.TSK

APPL\$DST:file.TSK

Additionally, the following form of taskspec is allowed for .INB files:

file.TSK

file Is the filename for the task image to be executed.

EXECUTE

directory Is the directory containing the task image to be executed.

APPL\$DST Points to the destination device:[directory] for files being transferred.

/option1 Is required. It is one of the following:

/INS Executes the specified task either during application installation, or when the workstation or Server manager is copying application files into a shared area (such as the Public Library or LB000:). This command must follow the FILE command that copies the specified task image to the hard disk. It can precede other FILE commands in order to avoid copying files needlessly in the event that the task exits unsuccessfully.

/REM Executes the specified task at application removal, or when the workstation or Server manager is deleting application files from a shared area (such as the Public Library or LB000:). This command must precede the FILE/DELETE command that refers to the same task image file.

/option2 Is optional and allowed only for .INB files. This option further specifies when the task should be executed.

Omitting option2 executes the task as follows:

- **During installation** (you have also specified the /INS qualifier): The system executes the task when copying /USER files from the application diskette into APPL\$USER:.
- **During removal** (you have also specified the /REM qualifier): The system executes the task when deleting from APPL\$USER: those /USER files originally copied from application diskette.

/USR as option2 executes the task as follows:

- **During installation** (you have also specified the /INS qualifier): The system executes the task when copying /USER files from the Public Library into APPL\$USER:.

EXECUTE

- **During removal** (you have also specified the /REM qualifier): The system executes the task when deleting from APPL\$USER: those /USER files originally copied from the Public Library.

/NTW as option2 executes the task as follows:

- **When copying** (you have also specified the /INS qualifier): The system executes the task when copying /NETWORK files from application diskette to APPL\$NETWORK:.
- **When deleting** (you have also specified the /REM qualifier): The system executes the task when deleting /NETWORK files from APPL\$NETWORK:.

/CLS as option2 executes the task as follows:

- **When copying** (you have also specified the /INS qualifier): The system executes the task when copying /CLUSTER files from the application diskette onto LB000:
- **When deleting** (you have also specified the /REM qualifier): The system executes the task when deleting /CLUSTER files from LB000:.

/option3 is optional and is allowed only for .INB files. This option specifies the location of the task image, if you did not specify it in the taskspec. If present, option3 is one of:

/USER Specifies that the task image is found in APPL\$USER:. If the corresponding FILE line has a /USER qualifier, the EXECUTE line should also have a /USER qualifier.

/NETWORK Specifies that the task image is found in APPL\$NETWORK:. If the corresponding FILE line has a /NETWORK qualifier, the EXECUTE line should also have a /NETWORK qualifier.

/CLUSTER Specifies that the task image is found in the system library, LB0:. If the corresponding

EXECUTE

FILE line has a /CLUSTER qualifier, the EXECUTE line should also have a /CLUSTER qualifier.

Notes on Combining Qualifiers:

In most cases, combinations of the EXECUTE qualifiers option2 and option3 are straightforward. The following examples are typical:

EXECUTE ATASK.TSK/INS/USR/USER

This command executes ATASK.TSK during application installation (see /INS), at the point when the system is copying /USER files from the Public Library (see /USR) into APPL\$USER:.. The system expects to find ATASK.TSK in APPL\$USER:--that is, the corresponding FILE command precedes the EXECUTE command. (See /USER.)

EXECUTE XTASK.TSK/INS/USER

This command executes XTASK.TSK during application installation (see /INS), at the point when the system is copying /USER files from the application diskette (no option2 qualifier present) into APPL\$USER:.. The system expects to find ATASK.TSK in APPL\$USER:.. (See /USER.)

EXECUTE BTASK.TSK/REM/NTW/NETWORK

This command executes BTASK.TSK during application removal (see /REM), at the point when the system is removing /NETWORK files (see /NTW) from APPL\$NETWORK:.. The system expects to find ATASK.TSK in APPL\$NETWORK:.. (See /NETWORK.)

The following examples are somewhat more complicated.

EXECUTE CTASK.TSK/INS/USR/NETWORK

This command executes CTASK.TSK during application installation (see /INS), at the point when the system is copying /USER files from the Public Library (see /USR) into APPL\$USER:.. The system expects to find CTASK.TSK in APPL\$NETWORK:.. (See /NETWORK.) The file CTASK.TSK is a shared component that the workstation or Server manager has previously copied to APPL\$NETWORK:..

EXECUTE [ZZWHA]DTASK.TSK/INS/USR/CLUSTER

This command executes DTASK.TSK during application installation (see /INS), at the point when the system is copying /USER files from the Public Library (see /USR) into

EXECUTE

APPL\$USER:. The system expects to find DTASK.TSK on LB000: (see /CLUSTER). The file DTASK.TSK is a shared component that the Server manager has previously copied to LB000:.

Examples showing invalid combinations of option2 and option3 follow.

EXECUTE [SOMEDIR]ETASK.TSK/INS/USR/NETWORK

INVALID COMMAND: The /NETWORK file ETASK.TSK cannot have an explicit directory.

EXECUTE [ZZSYS]FTASK.TSK/INS/CLS/USER

INVALID COMMAND: The desire is to execute the task when the Server manager is copying /CLUSTER files from diskette onto LB000:. (See /CLS.) However, the /USER qualifier tells the system to locate the file in the user-specific application directory, which does not exist at the time the Server manager is copying /CLUSTER files.

EXECUTE [ZZSYS]GTASK.TSK/INS/CLS/NETWORK

INVALID COMMAND: This example is similar to the previous one. The /NETWORK area, in which the system will search for GTASK.TSK, does not exist when the Server manager is copying /CLUSTER files from diskette onto LB000:.

General Notes:

- The EXECUTE command is not supported on P/OS Hard Disk prior to Version 2.0.
- The qualifiers /option2 and /option3 are supported for .INB files only.
- To execute a task whenever the installation process copies /USER files into APPL\$USER:, you must provide two EXECUTE commands: one that specifies the /USR qualifier as option2, and one that omits option2 altogether.
- The task referred to by taskspec in the EXECUTE command cannot be linked with libraries that must be installed during application installation. This is because all EXECUTE commands must precede all INSTALL commands in the installation file. Consequently, tasks written in high-level languages, which normally require installation of a run-time library, cannot be executed in the installation file.

EXECUTE

NOTE

A solution is to execute a task written in MACRO-11 that installs any required tasks or libraries, and then spawns other tasks written in high-level languages.

- Your executing task can determine the phase of installation or removal by issuing a GMCR\$ directive (documented in the *P/OS System Reference Manual*). This directive returns the option1 and option2 qualifiers present on the EXECUTE command. Table 6-2 describes the phases associated with different combinations of the qualifiers.
- You can place an EXECUTE command anywhere that a MOUNT or FILE command can be placed.
- APPL\$DST points to the destination device:[directory] for files being transferred during the current phase of installation.
- At installation, copying, or removal time the logical names APPL\$DIR, APPL\$USER:, and APPL\$NETWORK: are not defined for your application.
- The logical name APPL\$SRC: points to the source device:[directory] of the most recent file copied during the current phase of installation.

Table 6-2: EXECUTE Qualifiers and Phases of Installation/Removal

option1/option2	Installation or Removal Phase
/INS	End user is installing from diskette
/INS/USR	End user is installing from Public Library
/INS/NTW	System manager is installing the "entire application" or "application-specific components" from diskette into Public Library
/INS/CLS	System manager is installing "entire application" or "system-wide components" from diskette onto LB000:.

EXECUTE

option1/option2	Installation or Removal Phase
/REM	End user is removing application
/REM/USR	End user is removing application
/REM/NTW	System manager is removing application from Public Library
/REM/CLS	System manager is removing application from Public Library

FILE

6.9 FILE

The FILE command specifies the name of an application file. You must supply a FILE command for each file that is to be copied to your diskettes during application building and that is to be copied to or removed from the hard disk during application installation or removal.

This command is used by ADB when your application diskettes are built, and by P/OS when your application is installed or removed.

Format:

FILE filespec/option1/option2

filespec Is the file specification in the following form for both .INB and .INS files:

file.TSK

[directory]file.TSK

During the execution of ADB, the file specification is used to locate the file so that it can be copied to the diskette. The details of this search for the file to be copied are described in Chapter 1.

The file is copied to the diskette in the application directory, or in the directory that is named in the file specification. A directory is created on the diskette, if it does not already exist. Thus, FILE [ABC]FOO.DAT causes the file to be placed in [ABC] on the diskette.

During the installation of the application, the FILE command is used to determine where to place the file that is copied from the diskette.

/option1 Is required. It must be one of the following:

/KEEP Prevents the file (and the application directory) from being deleted when the application is removed.

/DELETE Allows the file (and the application directory, if there are no files remaining) to be deleted when the application is removed.

FILE

/option2 Is optional and allowed only for .INB files. It determines where the file is to be placed. Option2 must be one of the qualifiers /USER, /NETWORK, or /CLUSTER. The default is /USER, but you should always explicitly specify the qualifier. See Section 6.1.2 for details.

Note:

Do not include a FILE command for the installation file itself.

INSTALL

6.10 INSTALL

This command specifies each task image or library to be installed (placed in the installed task list) at run time. This command is used only by P/OS when the user selects your application to be run.

Format:

INSTALL filespec/option1/option2/NOREMOVE/FIX

filespec Is the file specification in the following form for both .INB and .INS files:

[directory]file.TSK

file.TSK

/option1 Is required. It must be one of the following strings:

/LIBRARY Specifies that the file is a read-only resident library.

/COMMON Specifies that the file is a read-write resident library.

/TASK Specifies that the file is an executable task image.

/TASK=name Is valid for .INB files only. It specifies that the file is an executable task image, and the name of the task image should be replaced with "name". The assigned name must be a one- to six-character string of characters in the Radix-50 character set.

option2 Is optional and is allowed only in .INB files. It can be one of /USER, /NETWORK, or /CLUSTER. The qualifier you specify indicates where the task image or library to be installed resides. The default is /USER, but you should always explicitly specify this qualifier. See Section 6.1.2 for details.

/NOREMOVE Can be specified only with the /TASK option. It specifies that the task will not be aborted or removed when the application exits (or is aborted).

/FIX Can be specified only with the /TASK option. It specifies that the task is to be fixed in memory during execution (not swapped).

INSTALL

Notes:

- The maximum number of INSTALL commands allowed in an installation file is 31.
- All INSTALL commands with /LIBRARY and /COMMON options must precede any INSTALL command with a /TASK option.
- The /NOREMOVE option can be used to ensure that a noninteractive background task, such as a file transfer, is not aborted on request.

Use care with the /NOREMOVE option. The task remains installed after your application terminates. When any application starts up, if it attempts to install a task with the same name as the task that you left installed, the new install command is ignored provided the INSTALL command has a /NOREMOVE option.

This is convenient for your application, since its installation file has such an option on the INSTALL line. But if another application happens to use the same task name, and its INSTALL does not use the /NOREMOVE option (the normal case), the INSTALL command fails and the other application cannot be run.

Once you install a task with /NOREMOVE, it remains installed (it can be checkpointed if active) until another task explicitly removes it (using the PRO/Tool Kit DCL REMOVE command, for example), or until the system is powered down.

For more information about installing tasks, refer to the PROTSK routine in the P/OS System Reference Manual.

MOUNT

6.11 MOUNT

This command specifies the name of the next volume in a multiple-diskette application.

When you are preparing your distribution diskettes, the MOUNT command signals ADB to prompt you to insert a new diskette, and causes ADB to copy subsequent files to the new volume.

When the user installs the application from your distribution kit, the MOUNT command specifies when the user should be prompted to insert the next diskette for installation to continue.

Format:

MOUNT volume

volume Is a string of up to 12 characters that specifies a volume name (without a colon).

Notes:

- The MOUNT command is optional for applications that reside entirely on one diskette.
- No MOUNT command is required for the first diskette of a multiple-diskette application.

NAME

6.12 NAME

This command specifies the application name seen by the user when the application is installed. P/OS places this name (or a replacement name supplied by the user) on a menu.

Format:

NAME "name"

name Is a string of up to 40 characters that identifies your application.

Notes:

- You must supply exactly one NAME command per installation file.
- You can use single or double quotes to delimit the string.

RUN

6.13 RUN

This command specifies the name of the first task to execute at run time.

Format:

RUN taskname

taskname is a string of up to six characters that specifies the name of an installed task.

Notes:

- You can include no more than one RUN command per installation file.
- The task name must be identical to the name specified in the TASK option in your PAB command file, or in the /TASK=name option. (If you did not use a TASK option when you built the application, the task name defaulted to the first six characters of the name of the task image file.)

IMPROVING APPLICATION PERFORMANCE

6.14 IMPROVING APPLICATION PERFORMANCE

To minimize the disk seek time and enhance the application start-up process, follow these guidelines when creating an installation file:

- Place FILE commands that have corresponding INSTALL commands before any other FILE commands. Arrange the INSTALL commands in the same order as their corresponding FILE commands. INSTALL commands with the /COMMON and /LIBRARY options must appear before INSTALL commands with the /TASK option.
- Wherever possible, group FILE commands and INSTALL commands by directory. This will minimize seek time for retrieving directory information.

6.15 REQUIRED COMMANDS FOR APPLICATIONS

Table 6-2 shows command lines that you might need to include in your installation command file, depending on other software that your application uses.

Note the following:

- For .INS files, omit the /CLUSTER qualifier.
- Applications do not need FILE or INSTALL commands for PRO/FMS-11, PRO/GIDIS, POSRES, PRO/Communications, RMSRES, or the print utility.

REQUIRED COMMANDS FOR APPLICATIONS

Table 6-3: Required Commands

If Application Uses:	Installation File (.INB) Must Include:
BASIC-PLUS-2	INSTALL [ZZSYS]PBFMSL.TSK/LIBRARY/CLUSTER
COBOL-81	INSTALL [ZZSYS]C81LIB.TSK/LIBRARY/CLUSTER
DIBOL	INSTALL [ZZSYS]DBLRES.TSK/LIBRARY/CLUSTER
FORTRAN-77	INSTALL [ZZSYS]PROF77.TSK/LIBRARY/CLUSTER
PASCAL	INSTALL [ZZSYS]PASRES.TSK/LIBRARY/CLUSTER
CET	INSTALL [ZZSYS]PASRES.TSK/LIBRARY/CLUSTER INSTALL [ZZSYS]CET.TSK/TASK/CLUSTER
PRO/SORT	INSTALL [ZZSYS]PASRES.TSK/LIBRARY/CLUSTER INSTALL [ZZSYS]PROSORT.TSK/TASK/CLUSTER

CHAPTER 7

MACRO-11 ASSEMBLER (PMA)

Professional Tool Kit MACRO-11 (PMA) is the PDP-11 relocatable assembly language processor. Refer to the *PDP-11 MACRO-11 Language Reference Manual* for detailed information about PMA.

7.1 INVOKING PMA ON THE PRO/TOOL KIT

To invoke PMA on a Professional running the PRO/Tool Kit, type:

```
$ MACRO
```

See the description of the MACRO command in the *PRO/Tool Kit Command Language and Utilities Manual* for details on the MACRO command.

You can also invoke PMA on the PRO/Tool Kit by entering the following command in DCL:

```
$ RUN $PMA
```

You receive the PMA> prompt, indicating that you are in the PMA environment. See the *PDP-11 MACRO-11 Language Reference Manual* for command line format.

7.2 INVOKING PMA ON RSX-11M/M-PLUS

If PMA is installed on your system as "...PMA" and your terminal Command Language Interpreter (CLI) is MCR, type the following command:

```
> PMA command
```

where "command" is a command string. See the *PDP-11 MACRO-11 Language Reference Manual* for command line format.

INVOKING PMA ON RSX-11M/M-PLUS

If PMA is installed on your system as "...PMA" and your terminal CLI is DCL, type the following command:

```
$ MACRO
```

See the *RSX-11M/M-PLUS Command Language Manual* for further information on the MACRO command.

If PMA is not an installed task (whether your terminal CLI is MCR or DCL), type:

```
$ RUN $PMA
```

You receive the PMA> prompt, indicating that you are in the PMA environment. See the *PDP-11 MACRO-11 Language Reference Manual* for command line format.

7.3 INVOKING PMA ON VAX/VMS

For convenience, edit your LOGIN.COM file and insert the following symbol definition:

```
$ PMA ::= MCR PMA
```

Once the symbol is defined, you can invoke PMA by typing:

```
$ PMA command
```

where "command" is a command string. See the *PDP-11 MACRO-11 Language Reference Manual* for command line format.

CHAPTER 8

POSRES USER INTERFACE LIBRARY ROUTINES

This chapter describes the routines provided by POSRES, the P/OS User Interface Library.

8.1 NOTES ON USING POSRES ROUTINES

- You can call POSRES routines from MACRO-11 or from high-level languages. POSRES uses the PDP-11 calling sequence convention, which is described in the *Tool Kit User's Guide*. POSRES preserves the stack pointer (SP); it does not preserve any other registers.
- Tasks that use POSRES must include some specific information in their PAB files, as described in the *Tool Kit User's Guide*.
- All POSRES routines accept a two-word integer parameter named "status". The values returned in the status block vary according to the routine called and are listed in Appendix C.
- Some POSRES routine parameters specify values (input parameters), others return values (output parameters), and some do both. Output parameters must be variables. Input parameters, however, can be variables, constants, or constant expressions, depending on your language. If it does not support constant expressions as actual parameters, read "expression" as "constant."
- Several POSRES routines accept parameters consisting of variable-length character (byte) string buffers. All string buffers are accompanied by an integer value that specifies the length of the buffer. Because these values mean the same thing, they are all named "buflen" and directly follow the buffer parameter.

NOTES ON USING POSRES ROUTINES

- String buffers that return values are accompanied by an extra parameter that returns an integer value representing the actual length of the string returned in the buffer. Because these integer variables all mean the same thing, they are all named "strlen" and directly follow the "buflen" parameter.
- Some POSRES routines have parameters that can be omitted from the right. For example, if the fourth parameter is omitted, then parameters five, six, and seven must also be omitted.
- Some POSRES routines have parameter groups with a variable format within the group. Short parameter lists (no more than three or four groups) are easier to read and debug. Long parameter lists, however, are more efficient because they require fewer calls.
- POSRES parameter list run-time checking is somewhat limited. Check your source code against the lists in this chapter carefully. Invalid parameters can cause unpredictable results.

DMENU - DISPLAY DYNAMIC MENU

8.2 DMENU - DISPLAY DYNAMIC MENU

This routine displays the single-choice menu in the dynamic buffer.

Format:

**DMENU (status, action, buflen, strlen, display, add_opt,
msg1, buflen, msg2, buflen)**

- status** Is a two-word integer array that returns a code indicating the results of the routine call. (See Appendix C.)
- action** Is a string buffer of up to 80 characters that returns the action string associated with the option selected. The action string is truncated or filled with blanks to fit the buffer.
- buflen** Is an integer expression that specifies the length of the action buffer.
- strlen** Is an integer variable that returns the actual length of the string in the action buffer.
- display** Is reserved for future use. Specify an integer value of zero for this parameter.
- add_opt** Is an integer expression that specifies whether to display an Additional Options flag on the menu and whether to return control if the user presses the ADDTNL OPTIONS key. Values:
- zero = no flag
 - nonzero = display Additional Options flag
- msg1** Is a string expression of up to 80 characters that specifies a message to be displayed on line 23.
- buflen** Is an integer expression that specifies the length of msg1.
- msg2** Is a string expression of up to 80 characters that specifies a message to be displayed on line 24.
- buflen** Is an integer expression that specifies the length of msg2.

DMENU - DISPLAY DYNAMIC MENU

Note:

If you specify a nonzero value for the Additional Options flag, POSRES returns (-14/14) in the status block when the user presses the ADDTNL OPTIONS key.

Example:

```
1880 REM *** Displaying a single-choice Dynamic Menu. ***
1890 Act$ = SPACE$(80) !Buffer area of the action string
1900 Msg1$ = 'This is the first message line for the dynamic menu.'
1910 Len1% = LEN(Msg1$) !Length of the first message line
1920 Msg2$ = 'This is the second message line for the dynamic menu.'
1930 Len2% = LEN(Msg2$) !Length of the second message line
1940 CALL Dmenu BY REF &
      (Stblk(),Act$,80%,Len3%,0%,0%,Msg1$,Len1%,Msg2$,Len2%)
```

DPACK - PACK DYNAMIC SINGLE CHOICE MENU

8.3 DPACK - PACK DYNAMIC SINGLE CHOICE MENU

This routine packs (stores information in) the dynamic menu buffer.

Format:

DPACK (status, group,...)

The format of a group is:

```
fieldid, buflen, fieldval, buflen
'CLRB', 4
'DFLTnn', 6
'KEYWnn', 6, offset, keylen
```

status Is a two-word integer array that returns a code indicating the results of the routine call (see Appendix C).

fieldid Is a string expression of up to six uppercase characters that specifies one of the following field identifiers:

<u>fieldid</u>	<u>nn</u>	<u>menu field</u>
'TITL'		title
'TEXTnn'	01,02,03	explanatory text
'GHLP'		global help frame identifier
'PRMT'		prompt text for line 21
'OPTNnn'	01-12	option text for option nn
'ACTNnn'	01-12	action string for option nn
'OHLNnn'	01-12	help frame identifier for option nn

buflen Is an integer expression that specifies the length of the fieldid.

fieldval Is a string variable of any length that specifies the contents of the field. For example, if fieldid is 'TITL' then fieldval specifies the title text.

buflen Is an integer expression that specifies the length of the field value.

CLRB Is a string constant containing a command that clears the dynamic buffer.

DPACK - PACK DYNAMIC SINGLE CHOICE MENU

DFLTnn Is a string constant containing a command that sets the default option number to nn (01-12).

KEYWnn Is a string constant containing a command that specifies the keyword of option nn (01-12).

offset Is an integer expression that specifies the beginning of an option keyword as an offset from the beginning of the option (range zero to one less than the length of the option text).

keylen Is an integer expression that specifies the length of the option keyword.

Notes:

- The order in which you pack menu fields is not significant.
- If you specify an option help frame identifier (OHLNnn) with an invalid (greater than 12) number, POSRES does not return an error code in the status block.

Example:

```
1480 REM *** Packing the Dynamic menu buffer.
1490 Title$ = 'Title for DYNAMIC Menu'
1500 L1% = LEN(Title$)           !Length of title field
1510 Txt1$ = 'This menu is a shorter version of the FRAME001 Menu.'
1520 L2% = LEN(Txt1$)           !Length of text1 field
1530 Glbhlp$ = 'HELP0000'
1540 Prmtln$ = 'Select the desired option and hit DO'
1550 L3% = LEN(Prmtln$)         !Length of prompt line
1560 Opt1txt$ = 'Option 1 - the first choice'
1570 L4% = LEN(Opt1txt$)       !Length of option 1 text
1580 Opt2txt$ = 'Second option - the default'
1590 L5% = LEN(Opt2txt$)       !Length of option 2 text
1600 Actst1$ = 'Action string for the first test option.'
1610 L6% = LEN(Actst1$)        !Length of action 1 text
1620 Actst2$ = 'Action string for the second option'
1630 L7% = LEN(Actst2$)        !Length of action 2 text
1640 Hlpfr1$ = 'HELP0001'
1650 Hlpfr2$ = 'HELP0002'

1655 ! The Dpack call can be specified on multiple lines.
      ! However, the first parameter must always be Status

1657 REM *** Packing the new Dynamic Single Choice Menu.***
1660 Call Dpack BY REF &
      (Stblk%(), 'CLRB', 4%, 'TITL', 4%, Title$, L1%)
1680 Call Dpack BY REF &
```

DPACK - PACK DYNAMIC SINGLE CHOICE MENU

(Stblk%(), 'TEXT01', 6%, Txt1\$, L2%)
1700 Call Dpack BY REF &
(Stblk%(), 'GHLP', 4%, Glbhlp\$, 8%, 'DFLT02', 6%)
1720 Call Dpack BY REF &
(Stblk%(), 'PRMT', 4%, Prmtln\$, L3%)
1740 Call Dpack BY REF &
(Stblk%(), 'OPTN01', 6%, Opt1txt\$, L4%)
1760 Call Dpack BY REF &
(Stblk%(), 'OPTN02', 6%, Opt2txt\$, L5%)
1780 Call Dpack BY REF &
(Stblk%(), 'ACTN01', 6%, Actst1\$, L6%)
1800 Call Dpack BY REF &
(Stblk%(), 'ACTN02', 6%, Actst2\$, L7%)
1820 Call Dpack BY REF &
(Stblk%(), 'OHLP01', 6%, Hlpfr1\$, 8%)
1840 Call Dpack BY REF &
(Stblk%(), 'OHLP02', 6%, Hlpfr2\$, 8%)
1860 Call Dpack BY REF &
(Stblk%(), 'KEYW01', 6%, 0%, 8%, 'KEYW02', 6%,0%,6%)

FATLER - FATAL ERROR

8.4 FATLER - FATAL ERROR

The Fatal Error routine informs the user of an unexpected and disabling error condition and returns control to P/OS (not to your task). It blanks line 22, displays the message "Application error. Press RESUME to return to Main Menu." on line 23, and displays an application-defined message on line 24. That message can tell the user why the application failed and where to look for recovery information.

Format:

FATLER (message, buflen)

message Is a string expression of up to 80 characters that specifies an error message.

buflen Is an integer expression that specifies the length of the message parameter.

Example:

```
100 ON ERROR GOTO 1120
.
.
.
1120 REM Fatal error handling
1130 Linenum$ = STR$(ERL) ! Get line number
1140 Errormsg$ = ERT$(ERR) ! Get error message number
1150 Rtnmsg$ = 'Fatal error occurred at ' &
+ Linenum$ + '. ' + Errormsg$
1160 L% = LEN(Rtnmsg$)
1170 CALL Fatler BY REF( Rtnmsg$, L% )
```

GETKEY - GET KEYSTROKE

8.5 GETKEY - GET KEYSTROKE

Gets a single keystroke from the terminal. The keystroke is not echoed on the screen.

Format:

GETKEY (status)

status Is a two-word integer array containing one of the following codes:

First Word	Second Word
+1	A DEC Multinational decimal character code, representing a main keyboard key.
+2	A code representing one of the function keys shown in Appendix D.
n	Indicates an error has occurred. See Appendix C for status values.

Example:

```
100 DIM Stblk%(1%)
PRINT "Enter a key "; \ CALL Getkey BY REF( Stblk%( ) )
SELECT Stblk%(0%)
CASE 1%
PRINT "Main keyboard key = "; CHR$(Stblk%(1%))
CASE 2%
PRINT "Function or Keypad key = "; Stblk%(1%)
CASE ELSE
PRINT "Error. Status = "; Stblk%( 0% ), Stblk%( 1% )
END SELECT
GOTO 100
```

HCLOSE - CLOSE HELP FILE

8.6 HCLOSE - CLOSE HELP FILE

This routine closes the current help definition file.

Format:

HCLOSE (status)

status Is a two-word integer array that returns a code indicating the results of the routine call. (See Appendix C.)

Example:

```
1990 REM *** Closing help definition file ***
2000 CALL Hclose BY REF( Stblk%() )
```


HELP - DISPLAY HELP FRAME

8.7 HELP - DISPLAY HELP FRAME

Displays the default help frame or a specified help frame.

Format:

HELP (status, frameid, buflen)

status Is a two-word integer array that returns a code indicating the results of the routine call. (See Appendix C.)

frameid Is a string of up to eight characters that identifies the help frame to display.

buflen Is an integer expression that specifies the length of frameid.

Note:

If you specify a nonexistent help frame identifier, POSRES returns immediately without an error code in the status block.

Example:

```
1180 REM *** Display help text frame ***
1190 CALL Help BY REF(Stblk%(), 'HELP0000', 8%)
```

HFILE - OPEN HELP FILE

8.8 HFILE - OPEN HELP FILE

This routine opens the specified help definition file and sets the default help frame.

Format:

HFILE (status, filespec, buflen, frameid, buflen)

status Is a two-word integer array that returns a code indicating the results of the routine call. (See Appendix C.)

filespec Is a string expression that specifies the help definition file.

buflen Is an integer expression that specifies the length of the help file specification.

frameid Is a string expression of one to eight characters that specifies the name of the default help frame.

buflen Is an integer expression that specifies the length of the default help frame string.

Notes:

- Only one help file can be open at any time. If another help file is open, HFILE closes it before the requested file is opened.
- You should provide a complete file specification for your help file, unless your installation command file contains an ASSIGN HELP command. If your installation command file does contain an ASSIGN HELP command, you do not have to specify a device and directory in the file specification.

Use the appropriate system logical name to represent the device and directory. The location of the help file depends on the qualifiers you use on the FILE command in your installation file.

For example, suppose you specify the following FILE command in your .INB file:

```
FILE TEST.HLP/NETWORK
```

In your HFILE call, the file specification for TEST.HLP is:

```
APPL$NETWORK:TEST.HLP
```

HFILE - OPEN HELP FILE

Note that if you omit fields in the file specification, POSRES uses the P/OS default file specification, which is almost certainly wrong.

See the *Tool Kit User's Guide*, for details regarding system logical names and accessing application components.

- You must supply a default help frame. Failure to do so results in a fatal error.

Example:

```
100    DIM SCB%(7%), STATUS%(1%)
      !
      ! Get name of application directory
      !
      FILE_SPEC$ = SPACE$(29) ! buffer for equivalence name
      CALL PROLOG BY REF &
          (SCB%(), 4%, 'APPL$DIR', 8%, FILE_SPEC$, 30%)
      !
      ! Construct filespec and open help file
      !
      FILE_SPEC$ = FILE_SPEC$ + 'TEST.HLP'
      CALL HFILE BY REF &
          (STATUS%(), FILE_SPEC$, LEN(FILE_SPEC$), 'FOO', 3%)
```

HFRAME - SPECIFY HELP FRAME

8.9 HFRAME - SPECIFY HELP FRAME

Specifies the frameid of the frame to be displayed when help is requested. This call does not result in a display; it simply defines the default frame.

Format:

HFRAME (status, frameid, buflen)

status Is a two-word integer array that returns a code indicating the results of the routine call. (See Appendix C.)

frameid Is a string expression of up to eight characters that specifies the default help frame identifier. A fatal error results if frameid is specified as blank or of zero length.

buflen Is an integer expression that specifies the length of the frameid string.

Notes:

- The help frame definition file must be open before calling HFRAME.
- If you specify a nonexistent help frame identifier, POSRES returns immediately without an error code in the status block.

Example:

```
1140 REM *** Calling help frame ***  
1150 CALL Hframe BY REF(Stblk%), 'HELP0001', 8%)
```

MCLOSE - CLOSE MENU FILE

8.10 MCLOSE - CLOSE MENU FILE

This routine closes the current menu file.

Format:

MCLOSE (status)

status Is a two-word integer array that returns a code indicating the results of the routine call. (See Appendix C.)

Example:

```
2020 REM *** Closing menu definition file ***  
2030 CALL Mclose BY REF( Stblk%() )
```

MENU - DISPLAY SINGLE-CHOICE MENU

8.11 MENU - DISPLAY SINGLE-CHOICE MENU

Displays a frame from the default static menu buffer.

Format:

**MENU (status, action, buflen, strlen, display, ad_opt,
msg1, buflen, msg2, buflen)**

- status** Is a two-word integer array that returns a code indicating the results of the routine call. (See Appendix C.)
- action** Is a string buffer of up to 80 characters that returns the action string associated with the option selected. The action string is truncated or filled with blanks to fit the buffer.
- buflen** Is an integer expression that specifies the length of the action buffer.
- strlen** Is an integer variable that returns the actual length of the string in the action buffer.
- display** Is reserved for future use. Specify an integer value of zero for this parameter.
- add_opt** Is an integer expression that specifies whether to display an Additional Options flag on the menu and whether to return control if the user presses the ADDTNL OPTIONS key. Values:
- zero = no flag
- nonzero = display Additional Options flag
- msg1** Is a string expression of up to 80 characters that specifies a message to be displayed on line 23.
- buflen** Is an integer expression that specifies the length of msg1.
- msg2** Is a string expression of up to 80 characters that specifies a message to be displayed on line 24.
- buflen** Is an integer expression that specifies the length of msg2.

MENU - DISPLAY SINGLE-CHOICE MENU

Example:

```
1350 REM *** Calling menu routine ***
1360 Action$ = Space$( 80% ) !Buffer area for the action string
1370 Msg1$ = 'This is the first message line'
1385 L1% = LEN(Msg1$) !Length of the first message
1390 Msg2$ = 'This is the second message line'
1400 L2% = LEN(Msg2$) !Length of the second message
1410 !0% = display (full)
1420 CALL Menu BY REF &
      (Stblk%(),Action$,80%,Length%,0%,0%,Msg1$,L1%,Msg2$,L2%)
1430 !0% = add. options (no)
```

MFILE - OPEN MENU FILE

8.12 MFILE - OPEN MENU FILE

This routine opens the specified menu definition file.

Format:

MFILE (status, filespec, buflen)

status Is a two-word integer array that returns a code indicating the results of the routine call. (See Appendix C.)

filespec Is a string expression that specifies the menu definition file.

buflen Is an integer expression that specifies the length of the menu file specification.

Notes:

- Only one menu file can be open at any time. MFILE closes any open menu file before opening the requested file.
- You should provide a complete file specification for your menu file, unless your installation command file contains an ASSIGN MENU command. If your installation command file does contain an ASSIGN MENU command, you do not have to specify a device and directory in the file specification.

Use the appropriate system logical name to represent the device and directory. The location of the menu file depends on the qualifiers you use on the FILE command in your installation file.

For example, suppose you specify the following FILE command in your .INB file:

```
FILE TEST.MNU/USER
```

In your MFILE call, the file specification for TEST.MNU is:

```
APPL$USER:TEST.MNU
```

Note that if you omit fields in the file specification, POSRES uses the P/OS default file specification, which is almost certainly wrong.

MFILE - OPEN MENU FILE

See the *Tool Kit User's Guide*, for details regarding system logical names and accessing application components.

Example:

```
100    DIM SCB%(7%), STATUS%(1%)
      !
      ! Get name of application directory
      !
      FILE_SPEC$ = SPACE$(29) ! buffer for equivalence name
      CALL PROLOG BY REF &
          (SCB%(), 4%, 'APPL$DIR', 8%, FILE_SPEC$, 30%)
      !
      ! Construct filespec and open menu file
      !
      FILE_SPEC$ = FILE_SPEC$ + 'TEST.MNU'
      CALL MFILE BY REF (STATUS%(), FILE_SPEC$, LEN(FILE_SPEC$))
```

MFRAME - READ MENU FRAME

8.13 MFRAME - READ MENU FRAME

Reads the specified menu frame into the static menu buffer.

Format:

MFRAME (*status*, *frameid*, *buflen*, *action*, *buflen*, *strlen*)

status Is a two-word integer array that returns a code indicating the results of the routine call. (See Appendix C.)

frameid Is a string of one to eight characters that specifies a menu frame identifier.

buflen Is an integer expression that specifies the length of the *frameid*.

action Is a string variable of any length that returns the global action string associated with the menu frame, if present. The global action string is truncated or padded with spaces to fit the buffer.

buflen Is an integer expression that specifies the maximum length of the action parameter.

strlen Is an integer variable that returns the actual length of the returned global action string.

Note:

The menu definition file must be open before calling MFRAME.

Example:

```
1220 Gloact$ = Space$( 80% ) !Buffer for global action string
1230 CALL Mframe BY REF &
      ( Stblk%(), 'FRAME001', 8%, Gloact$, 80% , Len%)
```

MMENU - DISPLAY MULTIPLE-CHOICE MENU

8.14 MMENU - DISPLAY MULTIPLE-CHOICE MENU

This routine displays a multiple-choice menu.

Format:

```
MMENU (status, opt_buff, opt_len, opt_count, max_opt,  
       resp_count, resp_array, add_opt,  
       msg1, buflen, msg2, buflen)
```

status Is a two-word integer array that returns a code indicating the results of the routine call. (See Appendix C).

opt_buff Is a string buffer that specifies the options on this menu, in the form of a concatenated string.

opt_len Is an integer expression that specifies the length of an option (up to 64 characters). All options must have the same length. Before concatenation, use trailing space characters to pad all option strings to this length.

opt_count Is an integer expression that specifies the number of options in the buffer.

max_opt Is an integer expression that specifies the maximum number of options that the user can select.

resp_count Is an integer variable that returns the number of options actually selected.

resp_array Is an integer array of the size specified in **max_opt** that returns the ordinal numbers of the selected options in ascending order.

add_opt Is an integer expression that specifies whether to display an Additional Options flag on the menu and whether to return control if the user presses the ADDTNL OPTIONS key. Values:

zero = no flag

nonzero = display Additional Options flag

msg1 Is a string expression of up to 80 characters that specifies text to be displayed on line 23.

buflen Is an integer expression that specifies the length of **msg1**.

MMENU - DISPLAY MULTIPLE-CHOICE MENU

msg2 Is a string expression of up to 80 characters that specifies text to be displayed on line 24.

buflen Is an integer expression that specifies the length of msg2.

Example:

```

2300 REM *** Setting up to display the Multiple-choice Menu ***
2310 Optlen% = 6%                !Maximum option length
2320 Optct% = 9%                !Option count.
2330 Opttxt$(0%) = "Red  "      !Option text. Note that
2340 Opttxt$(1%) = "Orange"    !Tool Kit BP2 zero-based arrays
2350 Opttxt$(2%) = "Yellow"    !are returned in Resary%
2360 Opttxt$(3%) = "Green  "   !as 1 through 9, not
2370 Opttxt$(4%) = "Blue  "    !0 through 8.
2380 Opttxt$(5%) = "Purple"
2390 Opttxt$(6%) = "Brown "
2400 Opttxt$(7%) = "Black  "
2410 Opttxt$(8%) = "White  "
2430 ! Now concatenate all the options
2435 ! in one string (Optct% by Optlen%)
2440 Opt$ = Opttxt$(0%) + Opttxt$(1%) + Opttxt$(2%) + Opttxt$(3%) + &
      Opttxt$(4%) + Opttxt$(5%) + Opttxt$(6%) + Opttxt$(7%) + &
      Opttxt$(8%)
2450 Lmt% = 9%                  !Max. number of options
2460 Msg1$ = SPACE$(75%)       !Buffer area for message line one.
2470 Msg2$ = SPACE$(75%)       !Buffer area for message line two.

2475 REM *** Displaying the Dynamic Multi-Choice Menu. ***
2480 CALL Mmenu BY REF &
      (Stblk(), Opt$, Optlen%, Optct%, Lmt%, Res%, Resary(), &
      0%, Msg1$, 75%, Msg2$, 75%)

```

MPACK - PACK MULTIPLE-CHOICE MENU

8.15 MPACK - PACK MULTIPLE-CHOICE MENU

This routine packs (stores information in) the multiple-choice menu buffer.

Format:

MPACK (status, group,...)

The format of a group is:

```
fieldid, buflen, fieldval, buflen  
'CLRB', 4
```

status Is a two-word integer array that returns a code indicating the results of the routine call. (See Appendix C.)

fieldid Is a string expression of up to six uppercase characters that specifies one of the following field identifiers:

<u>fieldid</u>	<u>nn</u>	<u>menu field</u>
'TITL'		title
'TEXTnn'	01,02,03	explanatory text
'GHLP'		global help frame identifier
'PRMT'		prompt text for line 21

buflen Is an integer expression that specifies the length of the field identifier.

fieldval Is a string variable of any length that specifies the contents of the field. For example, if fieldid is 'TITL' then fieldval specifies the title text.

buflen Is an integer expression that specifies the length of the field value string.

CLRB Is a string constant containing a command that clears the multi-buffer before use.

Note:

You can make multiple calls to MPACK.

MPACK - PACK MULTIPLE-CHOICE MENU

Example:

```
2050 REM *** Setting up to pack the multi-buffer **
2060 Title$ = 'Title for Multiple-choice Menu '
2070 L1% = LEN(Title$)           !Length of the title field
2080 Txt1$ = 'This menu is an example of a Multiple-choice menu.
2090 L2% = LEN(Txt1$)           !Length of the first text line
2100 Txt2$ = 'Use the SELECT key to choose your favorite colors.'
2110 L3% = LEN(Txt2$)           !Length of the second text line
2140 Glbhlp$ = 'HELP0000'
2150 Prmtln$ = 'Use the SELECT key to make a choice and press DO'
2160 L5% = LEN(Prmtln$)         !Length of the prompt line
2170 REM *** Packing the multi-buffer. ***
2180 Call Mpack BY REF &
      (Stblk%(), 'CLRB', 4%, 'TITL', 4%, Title$, L1%)
2200 Call Mpack BY REF &
      (Stblk%(), 'TEXT01', 6%, Txt1$, L2%)
2220 Call Mpack BY REF &
      (Stblk%(), 'TEXT02', 6%, Txt2$, L3%)
2260 Call Mpack BY REF &
      (Stblk%(), 'GHLP', 4%, Glbhlp$, 8%)
2280 Call Mpack BY REF &
      (Stblk%(), 'PRMT', 4%, Prmtln$, L5%)
```

MSGBRD - SEND MESSAGE TO A MESSAGE/STATUS DISPLAY

8.16 MSGBRD - SEND MESSAGE TO A MESSAGE/STATUS DISPLAY

This routine sends a message to the P/OS Message/Status Display, described in the *Hard Disk System User's Guide*. In a P/OS Server environment, the Message/Status Display can belong to the user, the workstation, or the Server.

Format:

MSGBRD (status, message, buflen, destin)

status Is a two-word integer array that returns a code indicating the results of the routine call. (See Appendix C.)

message Is a string expression of up to 59 characters that specifies text to be sent to the Message/Status Display. All nonprintable characters in the message are replaced with blanks.

buflen Is an integer expression that specifies the length of the message string.

destin Is an integer expression that specifies which Message/Status Display is to receive the message. If omitted, the destination defaults to the user's Message/Status Display.

<u>Value</u>	<u>Global</u>	<u>Destination</u>
0	u.def	user's Message/Status Display
1	w.def	workstation's Message/Status Display

Note:

The MSGBRD routine is not in POSRES; it is in the system library (SYSLIB).

Example:

```
111 Dim Stblk%(1%)
114 Msg$ = "A sample message for posting."
117 REM Post message on workstation's message display.
140 Call Msgbrd By Ref (Stblk%(),Msg$,Len(Msg$),1%)
150 Print Stblk%(0), Stblk%(1)
160 End
```

MUNPK - UNPACK MENU BUFFER

8.17 MUNPK - UNPACK MENU BUFFER

This routine unpacks (returns the information stored in) the static menu buffer.

Format:

MUNPK (status, group,...)

The format of a group is:

```
fieldid, buflen, fieldval, buflen, strlen
'DFLT', 4, defopt
'KEYWnn', 6, offset, keylen
```

status Is a two-word integer array that returns a code indicating the results of the routine call. (See Appendix C.)

fieldid Is a string expression of up to six uppercase characters that specifies one of the following field identifiers:

<u>fieldid</u>	<u>nn</u>	<u>Menu Field</u>
'TITL'		title
'TEXTnn'	01,02,03	explanatory text
'GACT'		global action string
'GHELP'		global help frame identifier
'PRMT'		prompt text for line 21
'OPTNnn'	01-12	option text for option nn
'ACTNnn'	01-12	action string for option nn
'OHLPnn'	01-12	help frame identifier for option nn

buflen Is an integer expression that specifies the length of the field identifier string.

fieldval Is a string variable of any length that returns the value of the field. For example, if fieldid is 'TITL', then fieldval returns the title text.

buflen Is an integer expression that specifies the length of the field value buffer.

strlen Is an integer variable that returns the length of the field value string.

MUNPK - UNPACK MENU BUFFER

- DFLT** Is a string constant containing a command that requests the default option number.
- defopt** Is an integer variable that returns the default option number (01-12).
- KEYWnn** Is a string constant containing a command that requests the offset and length of the keyword of option nn (01-12).
- offset** Is an integer variable that returns the beginning of an option keyword as an offset from the beginning of the option (range zero to one less than the length of the option text).
- keylen** Is an integer variable that returns the length of the keyword.

Example:

```
1290 Title$ = Space$( 40% )           !Buffer area for the menu title
1300 CALL MUNPK BY REF(Stblk%(), 'TITL', 4%, Title$, 40%, Length%)
```

NEWFIL - NEW FILE

8.18 NEWFIL - NEW FILE

Solicits the name of a new file to be created by displaying the P/OS New File Specification form.

Format:

**NEWFIL (status, filespec, buflen, strlen, deftype,
buflen, strlen, text, buflen, msg, buflen)**

- status** Is a two-word integer array that returns a code indicating the results of the routine call. (See Appendix C.)
- filespec** Is a 50-character string buffer that returns the file specification entered by the user. If you specify a default filename, it can be no longer than nine characters and must be followed by a null character or a space.
- buflen** Is an integer expression that specifies the length of the filename buffer (50).
- strlen** Is an integer variable that returns the actual length of the file specification.
- deftype** Is a four-character string variable that specifies a default file type. If the user changes the file type via Additional Options, NEWFIL returns the updated file type.
- buflen** Is an integer expression that specifies the length of the file type buffer (4).
- strlen** Is an integer variable that returns the length of the string returned in deftype.
- text** Is a string expression of 0 to 72 characters that specifies text to appear at the top of the form.
- buflen** Is an integer expression that specifies the length of the text string.
- msg** Is a string expression of 0 to 80 characters that specifies text to appear on line 23.
- buflen** Is an integer expression that specifies the length of msg1.

NEWFIL - NEW FILE

Example:

```
100 REM Program to test the NEW FILE Service
110 DIM Stblk%( 1% )
120 Filename$ = SPACE$( 50% )      ! Set up buffer for filename
130 Filetype$ = '.DAT'            ! Set up filename type
140 Text1$ = 'Test to check the Service of NEW FILE.'
160 Msg1$ = 'Message at bottom of NEW FILE menu.'
180 CALL Newfil BY REF (Stblk%(),
                       Filename$, 50%, Namelen%,      &
                       Filetype$, 4%, Typelen%,      &
                       Text1$, LEN(Text1$), Msg1$, LEN(Msg1$))
190 PRINT "Status = "; Stblk%( 0% ), Stblk%( 1% )
200 PRINT "Filename = "; Filename$
210 PRINT "Name length = "; Namelen%
220 PRINT "Type length = "; Typelen%
230 END
```

OLDFIL - OLD FILENAME

8.19 OLDFIL - OLD FILENAME

The Old Filename routine solicits the names of one or more existing files by displaying the File Selection Menu.

Format:

**OLDFIL (status, maxfiles, files, strlens,
wildcard, buflen, text1, buflen,
msg1, buflen, msg2, buflen)**

- status** Is a two-word integer array that returns a code indicating the results of the routine call. (See Appendix C.)
- maxfiles** Is an integer variable that specifies the maximum number of files the user can select and returns the number of files actually selected.
- files** Is a string buffer that returns the selected file specifications in concatenated 50-character strings, left-justified and blank-filled. Thus, the size in characters is 50 times the number specified in maxfiles.
- strlens** Is an integer array of the size specified in maxfiles that returns the actual length of each file specification string.
- wildcard** Is a string expression that specifies the file selection criteria.
- buflen** Is an integer expression that specifies the length of the wildcard string.
- text** Is a string expression of 0 to 72 characters that specifies text to be displayed at the top of the screen.
- buflen** Is an integer expression that specifies the length of text.
- msg1** Is a string expression of 0 to 54 characters that specifies text to be displayed on line 23.
- buflen** Is an integer expression that specifies the length of msg1.
- msg2** Is a string expression of 0 to 54 characters that specifies text to be displayed on line 24.

OLDFIL - OLD FILENAME

buflen Is an integer expression that specifies the length of msg2.

Notes:

- You can use the default wildcard specification (*.*) by supplying a zero-length string, in order to display the latest versions of all files in the user's current directory.
- The Additional Options "show all versions" and "show only the latest versions" work only when you use the default wildcard specification. Otherwise, the same file selection menu will reappear.

Example:

```
100 DIM STBLK%(1%), SIZEARRAY%(1%)
120 NUMCHOICE% = 2%
130 FILEBUF$ = SPACE$(100%)
140 WILDSPEC$ = '*.TSK'
160 TEXT1$ = 'Choose a file spec and Press DO'
180 MSG1$ = 'This is the first message'
200 MSG2$ = 'This is the second message'
230 CALL OLDFIL BY REF                                     &
      (STBLK%(), NUMCHOICE%, FILEBUF$, SIZEARRAY%(),      &
       WILDSPEC$, LEN(WILDSPEC$), TEXT1$, LEN(TEXT1$),    &
       MSG1$, LEN(MSG1$), MSG2$, LEN(MSG2$))
250 PRINT "Status: "; STBLK%(0%), STBLK%(1%)
260 PRINT "Files chosen: "; NUMCHOICE%
270 PRINT MID$(FILEBUF$, (I% * 50%) - 49%, SIZEARRAY%(I%-1%)) &
      FOR I% = 1% TO NUMCHOICE%
```

PRSCSI - PARSE STRING

8.20 PRSCSI - PARSE STRING

This routine parses a string for a CSI sequence. The characters in the buff parameter are scanned from the left until a CSI character is found. The sequence following the CSI character is parsed and translated into a value indicating which function key terminates the string.

Format:

PRSCSI (status, buff, buflen, csipos)

status Is a two-word integer array. In the first element of the array, +2, indicates that a valid CSI sequence was found and that the second element contains one of the function key values in Appendix D. A value less than 0 indicates an error occurred. (See Appendix C.)

buff Is a string variable of any length.

buflen Is an integer expression that specifies the length of buff.

csipos Is an integer variable that returns the position of the CSI character, counting from one. (See the notes.)

Note:

The csipos parameter was described in previous Tool Kit documentation as the length of the string, up to but not including the CSI character. The value returned is actually the length of the string, up to and including the CSI character.

Example:

```
10      Dim Stblk%(1%)
30      Buffer$ = "This is an example of PARSE" + ESC + "[29~"
40      Call Prscsi By Ref &
          (stblk%(), buffer$, len(buffer$), csipos%)
50      If stblk%(0%) < 0 goto 100
60      Buffer$ = left$(buffer$, csipos% - 1%)
70      Print buffer$
80      Print Stblk%(1%)
90      Goto 200
100     Print "error"; stblk%(0%); stblk%(1%)
200     End
```

RDMSG - READ MESSAGE

8.21 RDMSG - READ MESSAGE

This routine reads a message from a message definition file into a buffer.

Format:

**RDMSG (status, filespec, buflen, frameid, buflen,
message, buflen, strlen)**

status Is a two-word integer array that returns a code indicating the results of the routine call. (See Appendix C.)

filespec Is a string specifying the message definition file.

buflen Is an integer expression that specifies the length of the filespec parameter.

frameid Is a string expression of 1 to 8 characters that identifies the message frame to read from the message file. A fatal error results if frameid is specified as blank or of zero length.

buflen Is an integer expression that specifies the length of the frameid string.

message Is a string buffer of any length that returns the specified message. The message is truncated or filled with blanks to fit the buffer.

buflen Is an integer expression that specifies the length of the message buffer.

strlen Is an integer variable that returns the actual length of the returned message string.

Notes:

- If you do not include an ASSIGN MESSAGE command in your installation file, and you do not provide a complete file specification for your message file in the RDMSG call, POSRES attempts to open the message definition file in APPL\$USER: (or APPL\$DIR for older systems).
- If you do include an ASSIGN MESSAGE command in your installation command file, you do not have to specify a device and directory in the file specification. POSRES uses the qualifiers specified on the ASSIGN MESSAGE command to determine where to look for the the message definition file.

RDMSG - READ MESSAGE

For example, suppose you specify the following ASSIGN MESSAGE command in your .INB file:

```
ASSIGN MESSAGE TEST.MSG/USER
```

POSRES searches for TEST.MSG using the following file specification:

```
APPL$USER:TEST.MSG
```

See the *Tool Kit User's Guide*, for details regarding system logical names and accessing application components.

Example:

```
1100 CALL Rdmsg BY REF &  
      (Stblk%(),'BASETEST.MSG',12%,'MESS0001',8%,B$,B%,L%)
```


WTRES - WAIT FOR RESUME KEY

8.22 WTRES - WAIT FOR RESUME KEY

This routine echoes all keystrokes except the RESUME key with a bell character. When the user presses the RESUME key, control returns to the task. You can use this routine to allow the user to read something on the screen or change a diskette, for example, before proceeding. Before calling WTRES, display a message such as "Press RESUME to continue." on the screen.

Format:

WTRES ()

Example:

```
1180 PRINT "Calling HELP frame. Press RESUME to continue."  
1190 CALL Wtres BY REF( ) ! Allow user to read message  
1120 CALL Help BY REF(Stblk%(), 'HELP0000', 8%)
```


CHAPTER 9

CALLABLE PRINT SERVICES

Print Services consists of two callable routines that allow your application to print a file, stop, continue, abandon or restart a print job, or obtain printer status.

The two routines, CPRNT and CPRNV3, differ in their capabilities and in their compatibility with different operating system versions. See Section 9.3 for complete details.

CPRNV3 provides an additional feature: the ability to submit print requests to a particular print queue. Print queues are available only on P/OS V3.0 systems or later.

If you need compatibility across all system versions, use CPRNT instead of CPRNV3.

For both routines, a request to print a file executes a noninteractive task whose installed name is C\$PRNT. Also, both routines use the R5 calling convention to pass parameters. For each call, register 5 must contain the address of your parameter buffer. For details on the R5 calling convention, see the *P/OS System Reference Manual*.

9.1 CPRNT

Format:

CPRNT (status, request, filespec, len)

status A two-word integer array used to return a code indicating the results of the service request (see below).

request A 75-word integer array. The first word specifies one of the following request values:

CPRNT

- +1 Print file
- +2 Abandon current print job
- +3 Pause current print job
- +4 Continue the paused print job
- +5 Restart the current print job
- +6 Report printer condition

The remainder of the array is used as temporary storage by the CPRNT routine.

filespec An ASCII string expression that contains the file specification of the file to be printed if the print file service (+1) is requested. Specify zero if a service other than print file is requested.

len An integer expression that contains the number of characters in filespec. Specify zero if the filespec parameter was omitted.

Status Values:

- +1 In the first word of the status array indicates that the request was accepted. If the request was "report printer condition," the second word returns one of the decimal condition codes shown in Table 9-1.
- 1 In the first word of the status array indicates that a directive error occurred. The Directive Status Word (DSW) error is contained in the second element of the array.
- 21 In the first word of the status array indicates that a Print Service error occurred. The second element of the array contains one of the decimal error codes shown in Table 9-2.

Table 9-1: CPRNT Condition Codes

Code	Description
1.	Print job accepted
2.	Stop print job accepted
3.	Pause print job accepted
4.	Continue print job accepted
5.	Restart current file accepted
6.	Print job active
7.	Print job paused
8.	Print job inactive
9.	Printer offline

CPRNT

Code	Description
10.	Printer offline/paused
22.	Print request accepted, but printer type does not match
23.	Unable to determine printer status

Table 9-2: CPRNT Error Codes

Code	Description
-11.	Print job already active
-12.	Printer busy
-13.	Printer already attached
-14.	Print job not in progress
-15.	Print job not paused
-16.	Print job already paused
-17.	Parameter out of range
-18.	Printer is offline
-19.	More than 9 files selected
-20.	Printer offline / print job paused
-21.	Illegal file specification length
-24.	Port currently in use
-25.	Mini-Exchange not connected
-26.	Mini-Exchange printer currently in use
-27.	Print Server directive error encountered
-28.	Previous Mini-Exchange connection not broken
-29.	Error accessing set-up file. General RMS error.
-30.	Error accessing set-up file. File access error.
-31.	Error accessing set-up file. Error reading attributes.
-32.	Error closing set-up file.
-33.	Error accessing set-up file. Bad device specification.
-34.	Error accessing set-up file. Bad directory specification.
-35.	Error accessing set-up file. No such directory.
-36.	Error accessing set-up file. Device not ready.
-37.	Error accessing set-up file. File locked by another user.
-38.	Error accessing set-up file. File not found.
-39.	Error accessing set-up file. Bad node name.
-40.	Error accessing set-up file. Privilege violation.

CPRNT

Code	Description
-41.	Error accessing set-up file. File processor error.
-42.	Error accessing set-up file. File processor error.
-43.	Error accessing set-up file. Bad file extension.
-44.	Error accessing set-up file. Bad file version number.
-45.	Error accessing set-up file. Illegal wildcard in merged string.
-46.	Error accessing set-up file. Extraneous data in file specification.

Notes:

- On P/OS Hard Disk, the call to the CPRNT always returns immediately with status. If a request to print a file is accepted, then CPRNT prints the file as a noninteractive application while the calling application goes on to other processing.
- For compatibility with earlier versions of P/OS, you can call CPRNT on systems running P/OS V3.0 or later. See Section 9.3 for details.

9.2 CPRNV3

Format:

CPRNV3 (**status, request, subfun, copynum, reserv
qnam, qlen, filespec, flen, jobnum**)

status Is an eight-word integer array used to return status information. CPRNV3 currently uses only the first four words.

request Is a 75-word integer array. The first word contains the request:

CPRNV3

<u>Symbol</u>	<u>Value</u>	<u>Request</u>
RQ.PRN	1	Submit a set of files to print
RQ.ABO	2	Abort a print request
RQ.HLD	3	Hold a print request
RQ.REL	4	Release a held print request
RQ.STS	5	Report status of a print request

The remainder of the array is used as temporary storage by the CPRNV3 routine.

subfun Reserved.

copynum Is a one-word integer representing the number of copies of the job to print. The default value is one, and the maximum value is 100.

reserv Is reserved. You must supply a place holder, which CPRNV3 ignores. qname Is a variable-length string representing the name of the queue in which to place the print job. The maximum length is 39. If you specify zero for this parameter, CPRNV3 places the job in the default queue, if defined.

Note that, to send a request to the default print queue, the user who is running your application must have previously specified a default queue. You should query the user for the print queue name if your program receives the error CE.QNF (default print queue not defined).

qlen Is a one-word integer representing the length of the qname parameter.

fspec Is a variable-length string representing the file specification of the file to be printed. The maximum length is 65. CPRNV3 ignores this parameter if the request is other than to print files.

flen Is a one-word integer representing the length of the fspec parameter.

jobnum Is an integer word specifying the job number of the job to be aborted, held, released or reported. CPRNV3 ignores this parameter if the request is to print files.

CPRNV3

Status Values:

Table 9-3 shows the status returns that you can receive after a call to CPRNV3.

For a successful return (PS.SUC in word 0 of the return block), CPRNV3 places the job number in word 1 of the return block. If there is a directive error during processing, CPRNV3 returns the Directive Status Word (DSW) in word 1. For information on DSW values, see the *P/OS System Reference Manual*.

The error PE.PRN in word 0 indicates that an error occurred in the C\$PRNT task. The specific C\$PRNT error code is in word 1. All other error codes in word 0 indicate that the error occurred in the CPRNV3 routine.

Whenever CPRNV3 returns a value of -12. in word 2 of the return block, an RMS error has occurred; CPRNV3 returns the RMS STS and STV status in words 3 and 4, respectively. See the *PRO/RMS-11 Macro Programmer's Guide* for information on RMS errors.

Table 9-3: Status Returns for CPRNV3

Word 0	Word 1	Word 2	Word 3	Description
PS.SUC (+1)	Job number			Success
PE.DSW (-1)	DSW			Directive error
CE.EXT (-3)				Server exited before completing function
CE.BAB (-4)				Invalid argument block
CE.BQL (-5)				Invalid print queue name length
CE.BFL (-6)				Invalid file spec length
PE.PRN (-2)	CE.BFN (-2)			Invalid or unsupported request function

CPRNV3

Word 0	Word 1	Word 2	Word 3	Description
PE.PRN (-2)	CE.BRP (-3)			Incomplete request packet
PE.PRN (-2)	CE.QNF (-4)			(Default) print queue not defined
PE.PRN (-2)	CE.QFU (-5)			Print queue is full
PE.PRN (-2)	CE.NSJ (-6)			No such job
PE.PRN (-2)	CE.DNF (-8.)			Data file not found
PE.PRN (-2)	CE.ADF (-9.)	-12.	RMS error	Error accessing data file
PE.PRN (-2)	CE.PDF (-10.)	-12.	RMS error	Error parsing data file specification
PE.PRN (-2)	CE.VRC (-12.)	DSW		Error issuing VRCX directive
PE.PRN (-2)	CE.BQN (-13.)			Invalid print queue name
PE.PRN (-2)	CE.QUE (-14.)	-12.	RMS error	Error queueing (creating request file)
PE.PRN (-2)	CE.CPY (-15.)	-12.	RMS error	Error copying a data file
PE.PRN (-2)	CE.FVE (-16.)			Bad file version string
PE.PRN (-2)	CE.APS (-17.)	-12.	RMS error	Error accessing private set-up file
PE.PRN (-2)	CE.BJN (-19.)			Invalid job number
PE.PRN (-2)	CE.AQF (-20.)	-12.	RMS error	Error accessing queue file
PE.PRN (-2)	CE.ARQ (-21.)	-12.	RMS error	Error accessing entry file

CPRNV3

Notes:

- The CPRNV3 routine defines all request (RQ.xxx) and status (PE.xxx and CE.xxx) symbols. When the task builder resolves a reference to CPRNV3, it automatically resolves these symbols as well.
- After a call to CPRNV3, there might be a waiting period before Print Services processes the request. If your application queues a set of files to print and immediately follows with a status report request, the call may return a "No such job" error (CE.NSJ) because the print request has not yet been processed. In such cases, your application should wait for some time and retry.
- To indicate that you want the default value for a parameter, specify a place holder whose value is zero for a numeric parameter, or a null string (a string whose length is zero) for a string parameter.

9.3 COMPATIBILITY BETWEEN CPRNT AND CPRNV3

Use CPRNT if you design an application to run on all P/OS versions. However, when calling CPRNT on P/OS V3.0 systems or later, note that you can only specify the print request (RQ.PRN) in the call to CPRNT. No other requests are valid on P/OS V3.0 systems or later.

9.4 EXAMPLE

Figure 9-1 shows a sample MACRO-11 program that calls CPRNV3.

EXAMPLE

```

PRMBLK: .WORD    10.           ; Number of parameters
        .WORD    STATUS       ; Status block
        .WORD    REQUEST     ; Print request
        .WORD    SUBFUN      ; Subfunction
        .WORD    NUMCOP      ; Number of copies to print
        .WORD    BANNER      ; Banner page flag
        .WORD    QUEUE       ; Print queue name
        .WORD    LQUEUE      ; Length of the print queue name
        .WORD    FILEN       ; Address of the filename(s) to print
        .WORD    FILENL      ; Length of the filename(s)
        .WORD    JOBNUM      ; Job number

STATUS: .BLKW    8.           ; Status block
REQUEST: .WORD   RQ.PRN      ; Print file(s) request
        .BLKW    74.         ; Required buffer space for CPRNV3
SUBFUN:  .WORD   0           ; Subfunction value (reserved)
NUMCOP:  .WORD   1           ; Print 1 copy of the file
BANNER:  .WORD   1           ; Print with a banner page
QUEUE:   .WORD   1           ; Print queue name
;        .ASCII  /PRO$$PROVOLUME$LA50Q/ ;Example print queue name
LQUEUE:  .WORD   0           ; No length if using default queue
FILEN:   .ASCII  /FILE.TYP/  ; File name to print
TMP      = . - FILEN        ; Actual length of the filename
        .EVEN              ; Ensure on an even boundary
FILENL:  .WORD   TMP        ; Length of the filename
JOBNUM:  .WORD   0           ; No job number needed to print

;+
; Code
;-

.MCALL  EXIT$$

START:  MOV      #PRMBLK, R5 ; R5 => Parameter block pointer
        CALL    CPRNV3      ; Call Print Services to print a file

        EXIT$$              ; Exit the task

        .END    START

```

Figure 9-1: Sample Call to CPRNV3

CHAPTER 10

PROSE TEXT EDITOR

PROSE is the text editor supplied with P/OS. It offers facilities for entering and editing text to create documents, source programs, and memos or similar text files. Editing keys on the Professional keyboard allow text manipulation. The end user documentation describes PROSE and the PROSE user interface.

By calling the PROSE callable editor task (CET), an application can offer the text editor for use within its own context. For example, an electronic mail application might use the editor to provide editing services for message creation or modification. All the editor functions offered to the end user are available in the callable form of the editor.

Application tasks invoke the callable editor task by calling the CET subroutine that resides in SYSLIB. This subroutine invokes a separate editor server task to perform the actual editing session.

Parameters to the CET subroutine define the input and output filename, a temporary work filename, format options, maximum line length for text entry, and left and right margin values.

The input filename is the file to be edited. An empty or new file can also be created with the newfile parameter. After editing, the callable editor opens the output file with the specified name to create the edited version of the file.

The CET subroutine passes the specified parameters to the callable editor by spawning it as a task. The subroutine waits until the editor server task terminates before returning control to the calling application. When the editing session terminates, the server task returns a two-word status block to the CET routine, which in turn passes the status back to the application. The callable editor uses EFN (event flag number) 32, so the calling application should not use it.

PROSE Text Editor

Prior to calling the CET subroutine, the calling task should detach from the terminal since the callable editor attaches to the terminal during execution. After the callable editor task exits, the calling task can reattach to the terminal. For example, if the application is written in BASIC-PLUS-2 and attaches the terminal for CTRL/C ASTs, it should use RCTRLC to detach the terminal before CET is invoked.

List the name of the callable editor task in the application installation file to make sure that it is installed when the application task calls it. (See Chapter 6 for details.)

Format:

**CET (status, infile, inlen, outfile, outlen,
workfile, wklen, newfile, format, maxline,
initleft, initright, initwrap, lun)**

status	A two-word integer array that returns the results of the editing session as shown in Table 10-1.
infile	A string expression that specifies an input file. The default file type is .DOC.
inlen	An integer expression that specifies the number of characters in the input filename.
outfile	A string expression that specifies an output file. The default file type is .DOC. If the output file is the same as the input file, P/OS creates a new version of the file.
outlen	An integer expression that specifies the number of characters in the output filename.
workfile	A string expression that specifies a temporary work file. The editor task deletes this file on exit.
wklen	An integer expression that specifies the number of characters in work filename.
newfile	An integer expression that specifies whether a new source file should be created.
zero	Uses an existing file with the name specified in infile
nonzero	Creates a new file with the name specified in outfile

PROSE Text Editor

format An integer expression that specifies whether or not to retain escape sequences in the output file (see notes).

zero Discards formatting data

nonzero Retains formatting data

maxline An integer expression in the range 2 to 160 that specifies the maximum number of characters on a line that can be saved by a user.

initleft An integer expression in the range 1 to 131 that specifies the initial left margin value for a new file.

initright An integer expression that specifies the initial right margin value for a new file.

initwrap An integer expression that specifies the default word wrap setting for a new file.

zero Disables word wrap

nonzero Enables word wrap

lun An integer expression that must be present but is ignored by the editor task. Specify zero.

Note:

During the editing session, the user can define margin settings for different regions of text that result in the inclusion of nonprinting characters, called escape sequences, in the output file. The presence of escape sequences in files may cause errors if the file is used with host system utilities. If you remove the escape sequence from a file, it will print as expected. However, if you edit the file again, the margins will be lost.

Table 10-1: Callable Editor - Status Return Codes

First	Second	Meaning
+1	0	The editing session completed normally.

PROSE Text Editor

First	Second	Meaning
+2	0	The editing session completed normally, and the output file it produced contains escape sequences indicating formatting information (for example, margin settings).
-1	DSW code	The CET subroutine could not invoke the server task.
-2	POSRES code	The Callable Editor Task could not open its help file. The second status word contains the second word of the status block that was returned from the execution of the HFILE routines in POSRES.
-3		The Callable Editor Task encountered a problem related to the parameters passed to it. The second status word has a specific code:
	-1	The server encountered an error while trying to receive the parameters from the CET subroutine.
	-2	The parameters contain invalid values. For example, the initial left margin value is greater than or equal to the initial right margin value.
-4	RMS code	The Callable Editor Task could not open the specified input file. The second status word contains the RMS error code returned in the STS field of the FAB when the open failed.
-5	RMS code	The Callable Editor Task could not create the specified output file. The second status word contains the RMS error code returned in the STS field of the FAB when the create failed.
-6	RMS code	The Callable Editor Task could not create the specified work file. The second status word contains the RMS error code returned in the STS field of the FAB when the create failed.

PROSE Text Editor

First	Second	Meaning
-8		The server task encountered a fatal run-time error (for example, an odd address trap).
-9		The input file is not an RMS sequential file.
-10	RMS code	An I/O error occurred when writing the output file.
-11	RMS code	An I/O error occurred when performing I/O to the work file.
-12	RMS code	An I/O error occurred when reading the input file.
-13	RMS code	The UDK/Set-Up file could not be opened. The STS field of the FAB used to access the file is returned in the second status word.
-14	DSW code	CET could not create the dynamic region used to hold the editor's XBUF buffer pool.
-15	DSW code	CET could not create the address window needed to map to XBUF.
-16	DSW code	An error occurred during a mapping operation to access CET's XBUF region.

Example:

```

10      DIM StatBlk%(1%)      ! Status return data
      PRINT "Calling CET ..." ! Calling the editor task
20      CALL CET BY REF      &
      (StatBlk%(),          ! Return status block.          &
      "", 0%,               ! No name since creating new file.  &
      "MESG.EXC", 8%,       ! Name of resulting output file.  &
      "SY:W.FIL", 8%,       ! Work file used by editor.      &
      1%,                   ! We're creating a new file.      &
      0%,                   ! Allow margins; discard ESCapes  &
      160%,                 ! Let user create long lines.    &
      5%, 72%, 1%,         ! Initial margins 5, 72, word wrapped &
      0%)                   ! placeholder - value is ignored
      PRINT
      PRINT "Back from editor.  Status block:"

```

PROSE Text Editor

```
PRINT "      Word 0 =", StatBlk%(0%)  
PRINT "      Word 1 =", StatBlk%(1%)  
END
```

CHAPTER 11

PRO/SORT

PRO/SORT is a general-purpose sorting utility that runs on P/OS. This section discusses how to use PRO/SORT and the commands it supports. Error messages are listed in Section 11.13. For a sample PRO/SORT command file, see Section 11.14.

11.1 USING PRO/SORT

Application tasks use the PROSRT routine to invoke and pass to PRO/SORT the name of a command file containing sorting commands. The parameter list can include the names of an input file (containing records to be sorted) and an output file (containing the sorted records or indices to the sorted records).

Format:

```
PROSRT (cmdfile, buflen,  
        outfile, buflen,  
        infile, buflen, status)
```

- cmdfile Is a string expression containing a PRO/SORT command file specification. The default file type is ".CMD".
- buflen Is an integer expression that specifies the number of characters in cmdfile.
- outfile Is a string expression containing a PRO/SORT output file specification. There is no default file type.
- buflen Is an integer expression that specifies the number of characters in outfile.
- infile Is a string expression containing a PRO/SORT input file specification. There is no default file type.

USING PRO/SORT

buflen Is an integer expression that specifies the number of characters in inpfiler.

status Is a two-word integer array that receives status information from PRO/SORT.

In your application installation command file, make sure to include a FILE command for each PRO/SORT command file. Also, include the following commands in the order shown:

```
INSTALL [ZZSYS]PASRES.TSK/LIBRARY/CLUSTER
INSTALL [ZZSYS]PROSORT.TSK/TASK/CLUSTER
```

A sample BASIC-PLUS-2 call to PRO/SORT follows:

```
1000 DIM STATUS%(1%)
      CALL PROSRT BY REF ('TEST.CMD', 8%, 'DATA.OUT', 8%,
                        'SOURCE.INP', 10%, STATUS%())
```

Programmers using MACRO-11 should note that PRO/SORT uses event flag #31.

11.2 VALID RMS RECORD FORMATS

The PRO/SORT input and command files can be of variable-length or fixed-length sequential record format, independently of each other. For example, the input file can have variable-length records and the command file fixed-length records. The record type for the output file matches that of the input file.

If the input file uses variable-length records, the maximum record size (MRS) of the output file equals the input file MRS. If the input file uses fixed-length records, there are two possibilities for the output file:

- If the PRO/SORT command file has WRITE commands, the size of the output file equals the total length of all fields specified by the WRITE commands.
- If the PRO/SORT command file has no WRITE commands, the entire record is copied. Therefore the input and output record sizes are identical.

The maximum record length for a RECORD sort is 400 bytes. The maximum record length for a TAG sort is 1005 bytes. The maximum record length for an INDEX sort is 1005 bytes. The default is PROCESS RECORD. If you use a record that is too long, PRO/SORT returns an appropriate error message.

PRO/SORT COMMANDS AND COMMAND FILE

11.3 PRO/SORT COMMANDS AND COMMAND FILE

The following sections describe the commands supported by PRO/SORT. Where appropriate, the corresponding PDP-11 SORT specification lines are also described. If you are familiar with PDP-11 SORT, the comparison between SORT and PRO/SORT command syntax will be helpful.

Figure 11-2 on page 11-18 illustrates the structure of a PRO/SORT command file. During program execution, the command file is passed to the sort program and the sort is performed.

The maximum command line length is 80 characters. Table 11-1 lists the maximum number of commands allowed in a command file.

Table 11-1: Number of PRO/SORT Commands per Command File

Command	Maximum Number of Commands per File
COLLATE	no limit
DEFAULT	1
FIELD	20
FORCE	30
INCLUDE	20 *
INPUT	1 **
OUTPUT	1 **
PROCESS	1
SORT	16
WRITE	30 ***

Notes for Table 11-1:

- * Each condition in an INCLUDE line counts as an INCLUDE command.
- ** You can use the INPUT and OUTPUT statements in the command file only if no input or output file is specified in the CALL PROSRT parameter list.
- *** Each field name in a WRITE statement counts as a WRITE line.

Comment statements in a PRO/SORT command file are indicated by an exclamation point (!) or semicolon (;) as their first character. For example:

```
!This is a PRO/SORT comment.  
;This is an alternate form of the PRO/SORT comment.
```

PRO/SORT COMMANDS AND COMMAND FILE

By comparison, SORT comments are indicated by an asterisk (*) in column seven. For example:

```
*This is a SORT comment.
```

You can nest indirect command files within the PRO/SORT command file, to group related commands. To invoke an indirect command file from within the PRO/SORT command file, use the following format:

```
@filespec
```

filespec Is the name of the indirect command file to be accessed. The default file type is ".CMD".

Indirect command files can be nested as many as ten levels.

11.4 COLLATE

Use the COLLATE command to sort and output specific characters in a new position relative to the second value specified. For example, you can use the COLLATE command to perform country-specific collating sequences by changing the logical ordering of a character set. COLLATE operates on CHAR fields only.

Format:

```
COLLATE entity, ...
```

The format of an entity is:

```
value 

|        |
|--------|
| BEFORE |
| AS     |
| AFTER  |

 value
```

value A number or a character. Characters must be enclosed in single or double quotes.

BEFORE Sorts the first value immediately before the second value.

AS Sorts the first value as though it were the same as the second value.

COLLATE

AFTER Sorts the first value immediately after the second value.

Notes:

- If you use the COLLATE command before the DEFAULT command, the collating sequence specified by DEFAULT supersedes that specified by COLLATE. See the DEFAULT command for details.
- A command file can contain an unlimited number of COLLATE commands.

11.4.1 COLLATE Compared to SORT ALTSEQ

SORT performs alternate sequencing of records in a specification file through the ALTSEQ command. The format of an ALTSEQ record in SORT is:

```
ALTSEQ aaabbbxxxyyy...
```

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

The PRO/SORT COLLATE command provides a superset of the same function.

For example, to sort commercial "at" signs (@) as ASCII zero characters you would type:

```
ALTSEQ 100060
```

Using PRO/SORT, you would type:

```
COLLATE "@" AS "0" or COLLATE 100 AS 60
```

11.5 DEFAULT

Use the DEFAULT command to set up the default collating sequence. If you use the COLLATE command before the DEFAULT command, the collating sequence specified by DEFAULT will supersede that specified by COLLATE.

DEFAULT

The DEFAULT command has one of two formats. If you want to specify a country-specific collating sequence, use the following command format:

Format:

```
DEFAULT language  
USER n1,n2,n3,...,256
```

language Any of the national collating sequences supported by PRO/SORT: MULTINAT (DEC Multinational), DANISH, DUTCH, ENGLISH, FINNISH, FRENCH, GERMAN, ITALIAN, NORWEGIAN, PORTUGUESE, SPANISH, SWEDISH, and BINARY. The default is MULTINAT.

USER Specifies a user-defined collating sequence.

n Is an integer representing the binary value of the eight-bit character to be collated at this position. Separate integers with commas or carriage returns.

The DEFAULT USER command is equivalent to specifying individual COLLATE commands for all 256 values of an 8-bit character. Therefore, the list must always have 256 integer values. If you choose to have all characters treated distinctly (no "collate as"), the value of "n" ranges from 0 to 255. For example,

```
DEFAULT USER  
4,3,0,1,2,5,6,7,8,9  
10,11,12,...  
:  
:  
:  
...248,249  
250,251,252,253,254,255
```

In the above example, character 4 is collated first, followed by character 3. The remaining characters are collated normally, in order of increasing binary value up to the last character (255).

If you want to force a character to collate as the preceding character in the collating sequence, you must add 1000 to its normal value. For example,

```
DEFAULT USER  
1,2,0,3,7,8,12,1013,14,15,4,...
```

In this example, character 0 is collated after characters 1 and 2. Character 13 is collated equivalent to character 12.

You can specify only one DEFAULT command in the command file.

FIELD

11.6 FIELD

The FIELD command assigns a symbolic name to a positional range within an input record. For example, you could assign the field name "ZIP" to columns 5 through 9 in each record, the field name "Address" to characters 10-25, and so on. You can define up to 20 fields with FIELD commands.

Once the FIELD command has assigned a symbolic name to a field, other PRO/SORT commands reference that field by field name only. You must define a field before it can be referenced by other PRO/SORT commands.

Format:

FIELD fname datatype start end

fname A field name (must be a character string).

datatype One of the following six datatypes:

CHAR Specifies character data. The field width for this data type is equal to the number of characters.

SINT Specifies signed binary integer format. The field width must be given as two bytes per number.

UNSINT Specifies unsigned binary integer format. The field width must be given as two bytes per number.

DBLINT Specifies double precision binary integer format. The field width must be given as four bytes per number.

SPFP Specifies single precision floating point. The field width must be given as four bytes per number.

DPFP Specifies double precision floating point. The field width must be given as eight bytes per number.

start Is an integer representing the starting column number of the field within the record.

end Is an integer representing the ending column number of the field within the record.

FIELD

Figure 11-1 shows how PRO/SORT interprets bytes as they occur in the input data stream.

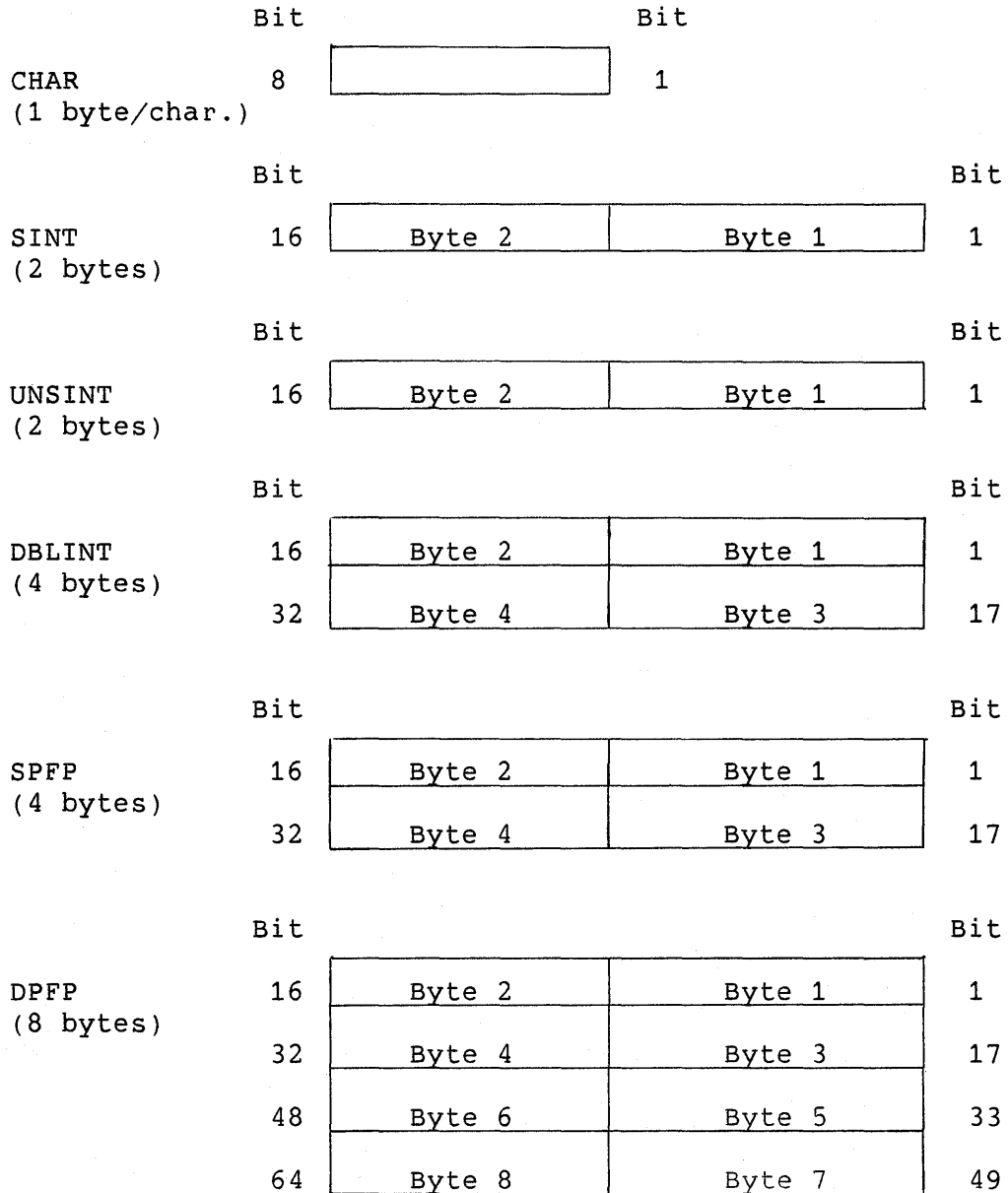


Figure 11-1: Mapping of Bytes to Fields in Input Data Stream

FIELD

Notes:

- For both the SPFP and DPFP data types, the sign appears in bit 16 (byte 2), the exponent appears in bits 8 through 15 (bytes 1 and 2), and the mantissa appears in the remaining bits. This conforms to the standard PDP-11 data type convention.
- For SINT, UNSINT, DBLINT, the most significant bit is 16, 16, or 32, respectively. For SPFP and DPFP, bit 15 is the most significant bit of the exponent and bit 7 is the most significant bit of the mantissa.

11.6.1 Pseudo-Fields RRN and RFA

Two pseudo-fields are predefined as part of PRO/SORT: relative record number (RRN) and record file address (RFA). The relative record number is defined as if your command file included the line:

```
FIELD RRN SINT x x+1
```

where x is the pseudo starting column of the field. This two-byte number identifies the record's position in the input file. Use RRN to perform random indexing of fixed length sequential files.

The record file address of a record is defined as if your command file included the line:

```
FIELD RFA CHAR x x+5
```

where x is the pseudo starting column of the field. This 6-character identifier provides the block and byte offset of the record in the input file. RFAs assist in the random access within a file that contains either fixed or variable length records. You can use RFA access to retrieve records randomly.

Use RRN and RFA in any other PRO/SORT command to obtain pointers to desired records. For example, the following commands would select the first 19 records in the input file and then write the each record's location (RFA) and key (KEYFIELDNAME) to the output file.

```
INCLUDE RRN < '20'  
FIELD KEYFIELDNAME CHAR 1 10  
WRITE RFA, KEYFIELDNAME
```

FORCE

11.7 FORCE

Use the FORCE command to force any record containing a particular field (or a field with a particular value) to a specified position in the collating sequence.

Format:

```
FORCE fname TO val IF val
```

fname The field name (must be a character string).

val A number or a character. Characters must be enclosed in single or double quotes.

You can use FORCE only on CHAR fields. While COLLATE affects all fields equally, FORCE operates only on the first character of the specified field. FORCE IF operates on that field only if its current value is equal to the value specified in the IF clause of the command.

For example, you could sort an input containing zip codes so that all zip codes appear at the top of the output file:

```
FORCE ZIP TO "0"
```

You could also sort such an input file so that only the zip codes beginning with 5 appear at the top of the list; all other zip codes would be sorted as usual:

```
FORCE ZIP TO "0" IF "5"
```

You can specify up to 30 FORCE commands in the command file.

11.7.1 FORCE Compared to SORT F

With the SORT F field record specification, you specify the column to be forced and the character that column should be forced to. An optional trigger character specifies that the force should not occur unless the indicated column contains that character.

The SORT "F" field specifier can also logically combine conditions through a FORCE command. PRO/SORT cannot do this.

For example, using SORT to sort all X characters in column 50 as if they were Ys, you would type:

```
F          50XY
```

FORCE

To perform a similar sort in PRO/SORT, you would type:

```
FIELD NAME CHAR 50 59
.
.
FORCE NAME TO 'Y' IF 'X'
```

11.8 INCLUDE

Use the INCLUDE command to sort and write to the output file only records containing fields that match certain conditions.

Format:

```
INCLUDE condition,...
```

The format of a condition is:

```
fname operator fname
                  constant
```

The format of an operator is:

EQ or =	(equal to)
NE or <>	(not equal to)
LT or <	(less than)
LE or <=	(less than or equal to)
GT or >	(greater than)
GE or >=	(greater than or equal to)

fname A field name (must be a character string enclosed in single or double quotes).

For example, to select and sort all records whose in an input file whose zip code field, ZIP, begins with 0, you could use the command:

```
INCLUDE ZIP < "10000"
```

Fields of data type CHAR can only be compared to other CHAR fields. Numeric fields can be compared to any other numeric fields. If a constant is specified along with a numeric field, it is assumed to be numeric.

When you specify more than one condition with a single INCLUDE command, PRO/SORT will combine the conditions logically with AND. Only records that meet all the INCLUDE line's conditions will appear in the output file.

INCLUDE

You can specify up to 20 INCLUDE conditions in a command file.

When you specify more than one condition with multiple INCLUDE lines, PRO/SORT will combine the conditions logically with OR. A record need only match one of the INCLUDE lines to appear in the output file.

For character fields, INCLUDE can also be used as follows, but only in the case of an EQ or NE comparison. It cannot be used for LT, LE, GT, or GE. The following examples will illustrate this special case.

```
INCLUDE fname = 'XYZ...'
```

Include any record that has a field starting with 'XYZ'.

```
INCLUDE fname = '...XYZ'
```

Include any record that has a field containing 'XYZ'.

11.8.1 INCLUDE Compared to SORT O and I

SORT and PRO/SORT use the same logical relationship tests to determine whether an output file will omit or include a given record. When you use the SORT I specification, you may either omit or include portions of the input file to be sorted. PRO/SORT provides one way to achieve either result through an INCLUDE command.

For example, using SORT to omit all records with column 10 to 14 equal to XYZZY you would type:

```
O C 10 14EQXYZZY
```

With PRO/SORT, you would type:

```
FIELD PLUGH CHAR 10 14  
INCLUDE PLUGH NE "XYZZY"
```

or:

```
INCLUDE PLUGH <> "XYZZY"
```

INPUT AND OUTPUT

11.9 INPUT AND OUTPUT

The INPUT command specifies which input file contains the data to be sorted. The OUTPUT command specifies which file the sorted records (or indices to sorted records, for the PROCESS INDEX command) are to be written to. You can use the INPUT and OUTPUT statements in the command file only if no input or output file is specified in the CALL PROSRT parameter list.

Format:

INPUT filespec

OUTPUT filespec

filespec A string that specifies the file being sorted or written to.

You can specify that file be output to your terminal screen by using TI: as the filename.

11.10 PROCESS

The PROCESS command specifies whether PRO/SORT uses a RECORD or TAG sort.

Format:

PROCESS **RECORD**
TAG

For a RECORD or TAG sort, the PROCESS command operates like the /PR switch in SORT. For an INDEX sort, the PROCESS command operates like the SORTA address routing sort.

In a RECORD sort, PRO/SORT manipulates actual records. Use a RECORD sort for small sorting jobs. In a TAG sort, PRO/SORT manipulates pointers to records. The TAG option can be significantly faster; use it for longer records. Switch to TAG if RECORD fails.

The default is PROCESS RECORD.

11.11 SORT

The SORT command specifies the key fields which will be used to reorder the records in the output file. Specify the key fields

SORT

in the order of their importance to the sort (primary, secondary, etc.).

Format:

SORT field,...

The format of a field is:

```
  + fname  
  -
```

fname The field name (must be a character string).

A leading plus or minus sign before the field name controls the order of the sort for the field. A leading plus sign indicates that the sort of that field should be performed in ascending order. A leading minus sign indicates that the sort should be performed in descending order.

To specify multiple key fields, you can use more than one SORT command or combine more than one key field on a line. Up to 16 key fields can be specified in a command file.

11.11.1 SORT Compared to SORT N and O

Both SORT and PRO/SORT permit key structures (primary, secondary, etc.). However, with SORT you can specify a maximum of 10 key fields. SORT uses N ("normal") and O ("opposite") in column 7 of the specification file to indicate ascending or descending sort per key. PRO/SORT uses the plus (+) and minus (-) signs.

For example, using SORT to sort the primary key ascending from columns 1 to 5, secondary descending from 34 to 37, you would type:

```
FNC 1 5  
FOC 34 37
```

Using PRO/SORT, you would type:

```
FIELD PRIMARY CHAR 1 5  
FIELD SECONDARY CHAR 34 37  
SORT +PRIMARY, -SECONDARY
```


WRITE

11.12 WRITE

The WRITE command creates an output file with a different arrangement of fields from the input file. You can use the WRITE command to omit a field from the output file, for example, or you can retain all the fields but output them in a new order.

Format:

```
WRITE fname,...
```

fname A field name (must be a character string).

You can specify up to 30 fields with WRITE commands in the command file.

11.12.1 WRITE Compared to SORT D

Both SORT and PRO/SORT permit the creation of an output file whose records differ in field order from the input file. In SORT, 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> by typing:

```
FDC 1 19
FDC 20 39
FDC 40 44
```

Using PRO/SORT, you would type:

```
FIELD NAME CHAR 1 19
FIELD ADDRESS CHAR 20 39
FIELD ZIP CHAR 40 44
WRITE ZIP, ADDRESS, NAME
```

11.13 PRO/SORT ERROR CODES

When PRO/SORT returns control to your program, the first element of the status array (element 0) contains one of the following codes:

```
1 = normal successful completion
-1 = error occurred in processing
-2 = could not invoke PRO/SORT
```

PRO/SORT is a separate task image on the P/OS system. If it

PRO/SORT ERROR CODES

could not be invoked, the spawn has probably failed. Check that you have listed the following lines (in the order shown) in the application installation file:

```
INSTALL [ZZSYS]PASRES.TSK/LIBRARY/CLUSTER
INSTALL [ZZSYS]PROSORT.TSK/TASK/CLUSTER
```

If an error occurred in processing, the second element of the status array (element 1) contains an error code.

A code that is greater than zero indicates a PRO/SORT internal error. Refer to the *PDP-11/SORT Reference Manual*.

A code that is less than zero indicates an error detected by the PRO/SORT task. Table 11-2 lists the error codes (decimal) and descriptions.

Table 11-2: PRO/SORT Error Codes

Decimal Error Code	Description of Error Condition
-1	A command in the sorting command file contains an invalid parameter, for example a string parameter where a numeric parameter is needed.
-2	The sort command file contains a line that is not recognized as a valid sort command.
-3	Can't open the specified sort command file.
-4	An INCLUDE command describes a relation between two fields of different data type.
-5	Two FIELD commands use the same name for their field.
-6	Multiple INPUT commands. Either the parameter list and the command file both contain an INPUT command or the the command file contains more than one.
-7	Multiple OUTPUT commands. Either the parameter list and the command file both contain an OUTPUT command or the the command file contains more than one.

PRO/SORT ERROR CODES

Decimal Error Code	Description of Error Condition
-8	A command has spurious text following the normal end of the command.
-9	A fatal error has been found in the logic of the PRO/SORT utility. Call the Hot Line.
-10	Too many FIELD commands have been given.
-11	Too many FORCE commands have been given.
-12	Too many INCLUDE commands have been given.
-13	Too many SORT commands have been given.
-14	Too many WRITE commands have been given.
-15	Ellipses were used with a relational operator other than = or <>.
-16	No INPUT command was found in the command file stream.
-17	No OUTPUT command was found in the command file stream.
-18	A command omitted a required parameter.
-19	FORCE commands may only be used on fields of type character, but a FORCE command was found in violation of this.
-15	The PRO/SORT data file be opened (SYSDISK:[ZZSYS]PROSORT.SYS).
-21	The input file is empty.
-22	Can't open the input file.
-23	Can't open the output file.
-24	The total size of all key fields added up is too large.
-25	The user's command file pool has too many levels of indirect command files.

PRO/SORT ERROR CODES

Decimal Error Code	Description of Error Condition
-26	A field name that does not yet exist has been referred to.
-27	More than one PROCESS command in file.
-28	Too few characters specified in the DEFAULT USER command.
-29	An input record larger than 1005 bytes was found.
-30	Insufficient disk space for temporary files.
-100 to -5200	An internal error reported by the PASCAL run-time system used by PRO/SORT.

11.14 PRO/SORT EXAMPLE

Figures 11-2 through 11-5 show several files associated with a sorting example. Figure 11-2 shows the PRO/SORT command file. It is designed to sort the information in the input file INPUT.DAT (Figure 11-3), rearrange it, and output it in a new order to the output file OUTPUT.DAT (Figure 11-5). The indirect command file FIELDS.CMD (Figure 11-4) is invoked by the PRO/SORT command file.

```
! Command file for sorting the INPUT.DAT file.
! Define the Fields:
@FIELDS
! Set up sorting parameters:
INPUT INPUT.DAT
OUTPUT OUTPUT.DAT
SORT Type, LastName
INCLUDE Type <> "Herol"
FORCE Type to "0" if "V"
WRITE Type, Space, FirstName, LastName
```

Figure 11-2: Sample PRO/SORT Command File

PRO/SORT EXAMPLE

```
Fake   Name2   Hxyzzy
Fake   Name    Hardy
Dick   Tracy   Hero
Mary   Worth   Normal
Clark  Kent     Hero
Peter  Parker   Hero1
Lex    Luthor   Villain
Green  Hornet   Hero
Lois   Lane    Heroine
Bruce  Wayne   Hero1
King   Tut      Villain
Gen.   Zod     Villain
Scooby Doo   Hero1
David  Banner   Hero
Perry  White    Normal
Comm.  Gordon   Normal
Mary   Jane     Normal
The    Joker   Villain
Miss   Piggy   Normal
Darth  Vader    Villain
Leia   Organa   Heroine
```

Figure 11-3: Sample Input File INPUT.DAT

The PRO/SORT command file begins with two comment lines:

```
! Command file for sorting the INPUT.DAT file.
! Define the Fields:
```

The first comment identifies the PRO/SORT command file. The second introduces the next command line, which invokes and executes the indirect command file FIELDS.CMD to define the fields in the input file:

```
@FIELDS
```

```
FIELDS.CMD
```

```
! Field specification for the input file INPUT.DAT
```

```
Field FirstName Char 1,7
Field LastName Char 8,14
Field Type Char 15,21
Field Space Char 7,7
```

Figure 11-4: Sample Indirect Command File FIELDS.CMD

PRO/SORT EXAMPLE

FIELDS.COMD defines each field in INPUT.DAT and assigns it a symbolic name. The field of characters 1-7 receives the field name FirstName; the field of characters 8-14 receives the field name LastName, and so on. Once PRO/SORT has executed this indirect command file, the program can refer to any field in the input file by the field name alone.

The next line in the PRO/SORT command file is another comment:

```
! Set up sorting parameters:
```

This line identifies the subsequent commands as those which set up the sorting parameters for the data file. The sorting parameters determine how fields in the data file will be sorted.

The first sorting parameter specifies the input file:

```
INPUT INPUT.DAT
```

It identifies INPUT.DAT as the file containing the data to be sorted. The next sorting parameter specifies the output file:

```
OUTPUT OUTPUT.DAT
```

It identifies OUTPUT.DAT as the file to which the sorted records will be written.

If INPUT.DAT and OUTPUT.DAT are specified in the CALL PROSRT parameter list, they can't be specified also in the PRO/SORT command file.

The next sorting parameter is a SORT command:

```
Sort Type, LastName
```

It specifies which key fields in the input file are pertinent to this sort, and in what order they will be sorted. The symbolic names Type and LastName have already been assigned by the indirect command file FIELDS.COM. They are now identified as the primary and secondary fields, which will be sorted in the default, or ascending, order.

The next line is an INCLUDE command:

```
Include Type <> "Herol"
```

It specifies which records will be sorted and written to the output file. In this case, only records whose field "Type" is not equal to "Herol" will be handled by PRO/SORT. Records with the Type "Herol" will be ignored.

PRO/SORT EXAMPLE

The next line is a FORCE command:

```
Force Type to "0" if "v"
```

It specifies that any record with a field type that begins with V should be sorted and written to the top (0) of the output file.

The last line is a WRITE command:

```
Write Type, Space, FirstName, LastName
```

Using this command, the programmer could have omitted any of the fields from the output record. Instead, the programmer has retained all fields but specified that they appear in the new order.

When this program is run, PRO/SORT creates the output file, OUTPUT.DAT. (See Figure 11-5.)

Villain	The	Joker
Villain	Lex	Luthor
Villain	King	Tut
Villain	Darth	Vader
Villain	Gen.	Zod
Hardy	Fake	Name
Hero	David	Banner
Hero	Green	Hornet
Hero	Clark	Kent
Hero	Dick	Tracy
Heroine	Lois	Lane
Heroine	Leia	Organa
Hxyzy	Fake	Name2
Normal	Comm.	Gordon
Normal	Mary	Jane
Normal	Miss	Piggy
Normal	Perry	White
Normal	Mary	Worth

Figure 11-5: Resulting Output File OUTPUT.DAT

APPENDIX A

APPLICATION DISKETTE BUILDER ERROR MESSAGES

The Application Diskette Builder displays an error message if it cannot continue building a diskette or if it finds an error in the installation command file. In the latter case, ADB may respond with more than one error message for the same error. For example, if the RUN command has an error, then two error messages result:

Invalid task name

A 'RUN' line must be present.

The normal ADB error messages are listed in alphabetic order in Section A.1.

Some error messages report conditions that you might not be able to resolve. These error messages indicate that a serious error has occurred. In most cases, the errors are the result of a failed file operation. These errors are listed in Section A.2 and appear in the following form:

ADB<xxx,yyy> - Message text

The values xxx and yyy represent error status numbers. These numbers correlate to a specific error, which is generally described by the message text. These values are positive. Look up the error status code in the PRO/RMS-11 manual set. If you cannot determine the cause of the error, submit a Software Performance Report (SPR) to DIGITAL. Record on the SPR the error message exactly as it was displayed on the terminal, including the values represented here as <xxx,yyy>.

ADB NORMAL ERRORS

A.1 ADB NORMAL ERRORS

A 'NAME' line must be present

The application installation file does not contain a NAME "menu name" line.

A NAME "menu name" must be present in each application installation file. The NAME "menu name" line assigns a default name to this application. Correct the application installation file and restart ADB.

A 'RUN' line must be present

The application installation file does not contain a RUN taskname line.

The application installation file must have, as the last line of the file, a RUN taskname line. When the application is invoked, P/OS executes the task listed in the RUN line. Correct the application installation file and restart ADB.

An 'INSTALL/TASK' line must be present

Either the application installation file does not contain an INSTALL line, or the INSTALL line(s) contain syntax errors.

INSTALL lines list all the programs that P/OS must install to run this application. Correct the application installation file and restart ADB.

At least one 'FILE' line must be present

Either the application installation file does not contain a FILE line or the FILE line(s) contain syntax errors.

FILE lines list all the files that must be copied to the target diskette. Correct the application installation file and restart ADB.

Copy aborted

An error occurred while ADB was copying files.

ADB NORMAL ERRORS

Error <xxx,yyy> creating directory <dirname>.

ADB could not create a directory specified in the FILE or REQUIRE command line.

Check the installation file and try again.

Invalid number or too large. -- Try again.

The number of blocks to allocate for a checkpoint file was entered incorrectly.

Invalid switch for this command line

The application installation file has an invalid switch.

Correct the application installation file and restart ADB.

Invalid syntax for this command

The application installation file contains unrecognized characters at the end of a line.

Correct the application installation file and restart ADB.

Invalid task name

A line in the application installation file that accepts task filenames lists an invalid task name.

Task names can be no more than 6 characters long. Correct the application installation file and restart ADB.

/LIBRARY and /COMMON switches must precede /TASK switches on INSTALL lines

An INSTALL/TASK line in the application installation file occurs before an INSTALL/LIBRARY or INSTALL/COMMON line.

Correct the application installation file and restart ADB.

ADB NORMAL ERRORS

Not enough contiguous space on the volume for <filename>

The file <filename> requires contiguous space and could not be copied to the remaining free space on the diskette.

Use a different diskette with more free space on it, or try to use the same diskette by changing the order in which files are copied.

To reorder the which file copying, list task images and files that are built to reside in contiguous or adjoining blocks first in the FILE lines of the application installation file. After contiguous files are copied, the remaining fragmented space can be used to store files that do not require a contiguous area.

Not enough room on this diskette for the application directory

The main ADB screen is displayed.

Not enough space left on the diskette for <filename>

During the copy operation the diskette filled up before the file <filename> could be copied to it.

Start over with another diskette.

Only one 'ASSIGN HELP' line is allowed

The application installation file contains multiple ASSIGN HELP lines. Only one line of this type is permitted.

The ASSIGN HELP line assigns a default help definition file to this application. Correct the application installation file and restart ADB.

Only one 'ASSIGN MENU' line is allowed

The application installation file contains multiple ASSIGN MENU lines. Only one line of this type is permitted.

The ASSIGN MENU line assigns a default menu definition file to this application. Correct the application installation file and restart ADB.

ADB NORMAL ERRORS

Only one 'NAME' line is allowed

The application installation file contains multiple NAME "menu name" lines.

Only one line of this type is permitted. The NAME "menu name" line assigns a default menu name to this application. Correct the application installation file and restart ADB.

Only one 'RUN' line is allowed

The application installation file contains multiple RUN taskname lines.

The application installation file must have, as the last line of the file, a RUN taskname line. When the application is invoked, P/OS executes the task listed in the RUN line. Correct the application installation file and restart ADB.

Please enter a valid alphanumeric name

Either no characters or invalid characters were entered.

The filename is invalid

A filename in the application installation file is not a correct PRO/RMS-11 file specification.

Correct the application installation file and restart ADB.

The frame identifier is incorrect

A frame identifier (frameid) in the application installation file is not in the correct format.

Check the help definition file to make sure you are spelling the frameid correctly. Correct the frame identifier in the application installation file and start again.

The Installation file command lines are in an incorrect order

The application installation file contains lines that are not in correct order.

Review the application installation file format and correct the errors in the file. Restart ADB.

ADB NORMAL ERRORS

The string must be quoted and within length boundaries

A string, such as the "menu name" string in the NAME line, is not surrounded by quotes, or it contains too many characters.

See the description of string syntax for strings appearing in the offending command in Chapter 6. Correct the application installation file and restart ADB.

The volume name is invalid

The volume name in a MOUNT line is not a 1- to 12-character alphanumeric string. A volume name in a MOUNT line cannot include a colon (:).

Correct your volume name and try again.

This command is invalid

An invalid command is listed in the application installation file.

Check the list of valid commands in the section on application installation files. Correct the application installation file and restart ADB.

Unable to bad block the diskette in <drive> -- Try another

The diskette in the named drive could not be checked for bad blocks as part of the initialization.

Replace the diskette and try the operation again.

Unable to locate the file <filename>

The application installation file could not be located in either the current directory or in the specified new directory.

Make sure the installation file is in the specified directory. Specify the correct application installation file.

ADB SERIOUS ERRORS

A.2 ADB SERIOUS ERRORS

ADB<xxx,yyy> - Cannot access menu file

ADB menu file could not be accessed.

Try running ADB again. If you receive the same error message, submit an SPR.

ADB<xxx,yyy> - Error creating this directory -- Try another

The destination directory could not be created on the target diskette, possibly because the diskette has bad blocks or format errors. The error code corresponds to either a PRO/RMS-11 error code or an I/O error code.

Either try another directory, or restart ADB with a new diskette.

ADB<xxx,yyy> - Error while gathering additional user directories

An error occurred while ADB was gathering additional user directories.

Submit an SPR.

ADB<xxx,yyy> - RMS error accessing the installation file

The application installation file could not be opened because of a record access error.

Check the diskette to make sure it contains an installation command file. If it does, make sure you have named the installation command file with the .INS or .INB file type. Look up the PRO/RMS-11 I/O error codes in the PRO/RMS-11 manual set, and restart ADB. If you receive the same error message, submit an SPR.

ADB<xxx,yyy> - RMS error connecting to <filename>

ADB could not access the file listed as <filename>.

Look up the PRO/RMS-11 I/O error codes in the PRO/RMS-11 manual set. Try running ADB again. If you receive the same error message, follow the procedures for submitting an SPR.

ADB SERIOUS ERRORS

ADB<xxx,yyy> - RMS error creating <filename>

ADB could not create the file listed as <filename>.

Look up the PRO/RMS-11 I/O error codes in the PRO/RMS-11 manual set. Try running ADB again. If you receive the same error message, follow the procedures for submitting an SPR.

ADB<xxx,yyy> - RMS error opening <filename>

ADB could not open the file listed as <filename>.

Look up the PRO/RMS-11 I/O error codes in the PRO/RMS-11 manual set. Try running ADB again. If you receive the same error message, follow the procedures for submitting an SPR.

ADB<xxx,yyy> - RMS error while reading <filename>

ADB could not read the file listed as <filename>.

Look up the PRO/RMS-11 I/O error codes in the PRO/RMS-11 manual set. Try running ADB again. If you receive the same error message, follow the procedures for submitting an SPR.

ADB<xxx,yyy> - RMS error while writing to <filename>

ADB could not write to the file listed as <filename>.

Look up the PRO/RMS-11 I/O error codes in the PRO/RMS-11 manual set. Try running ADB again. If you receive the same error message, follow the procedures for submitting an SPR.

ADB<xxx,yyy> - Unable to access directory listing

The directory of the specified device could not be obtained. A fatal error message is displayed.

Check to be sure the directory resides on the specified device and that the device is properly loaded. Try the operation again. If you receive the message again, submit an SPR.

APPENDIX B

FDT ERROR MESSAGES

FDT displays two types of error messages: user errors and internal errors.

User error messages are intended to be self-explanatory. If you need clarification or advice, this appendix contains an alphabetic* list of user error messages and possible solutions. A symbol enclosed in left and right angle brackets (< and >) represents a field name, frame identifier, or file specification that FDT has copied from your command or retrieved from stored information. Internal error messages indicate that a serious error has occurred in FDT processing. In most cases, it results from a file operation failure. Error messages appear in the following form:

FDT<xxx,yyy> - Message text.

The values xxx and yyy represent PRO/RMS-11 I/O error status numbers and correlate to a specific error which is generally described by the message text. When you see an internal error message, do the following:

1. Record the error message exactly as it was displayed on your terminal, including the numbers enclosed in angle brackets.
2. Exit FDT (enter the SAVE or EXIT command).
3. Invoke FDT and use the REPORT command to obtain a hardcopy listing of the file.

* The error messages are sorted by text, not symbols.

FDT Error Messages

4. Look up the error status code in the PRO/RMS-11 manual set. If you cannot determine the cause of the error, submit a Software Performance Report (SPR) to DIGITAL. Include a detailed description of the user-FDT dialogue up to and including the error message, and the report file.

B.1 USER ERRORS

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 - name of frame unchanged.

The NAME command specified a previously assigned frame identifier as a new frame identifier.

Use the LIST or REPORT commands to check the names of existing frame identifiers.

<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 has been created. Control returns to host system command level.

USER ERRORS

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 zero to xx, where xx is the total number of option description lines on the Display form. This error condition is detected after use of the CONVERT command.

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 use of the CONVERT command.

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, Name, 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, press RETURN. Then use the LIST command to display the frame identifiers.

Enter "H", "M" or "S" to select a file type, or a return.

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, press 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 use of the CONVERT command.

USER ERRORS

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 use of the CONVERT command.

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 use of the CONVERT command.

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. DELETE command is invalid.

An attempt was made to delete a frame that doesn't exist. No frame is deleted.

File <filename> is empty. File not written.

There are no frames in a file that was to be saved. 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> is empty. MODIFY command is invalid.

The MODIFY command was typed, but no frames are available for modification.

USER ERRORS

File <filename> is empty. NAME command is invalid.

The NAME command was typed, but no frames are available to rename.

File <filename> is not an FDT source file - try again.

The specified file is not a valid FDT definition file. FDT will prompt you again for the filename.

File <filename> not found.

Either the specified file is not a valid FDT definition file, or the file does not exist.

Create a new FDT definition file using the specified name or press RETURN to enter a different filename.

No frames written to converted file.

The CONVERT command specified a definition file but it contains no valid frames that may be converted. All 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 identifier may only have alphanumeric characters.

The specified frame identifier contained characters that are not in the ASCII alphanumeric character set.

Frame identifiers must be specified by ASCII alphanumeric characters only (A through Z and 0 through 9).

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.

USER ERRORS

Invalid command. Enter HELP for a list of commands.

An invalid command was entered.

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 and retype if necessary. Refer to the *Tool Kit User's Guide* or your host system documentation for more information on file specifications.

Invalid identifier.

A frame identifier was specified with more than eight characters, or with characters that are not from the ASCII alphanumeric character set.

Enter a frame identifier with no more than eight characters in the ASCII alphanumeric character set (A through Z and 0 through 9).

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.

USER ERRORS

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 use of the CONVERT command.

Alter the keyword for one or both of the specified options. Do not delete an option description line on the Display form to alter it. If you delete the line, 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 use of the CONVERT command.

Use the MODIFY command to fill in 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 were assigned actions.

Use ACTION ALL or ACTION option-number to modify the desired options.

USER ERRORS

No option lines defined.

The ACTION command was used but no option description lines were 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 for the previous field should be and enter that value in the field before attempting to convert the file.

Not enough room to insert a line.

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

Use the ARROW keys, or the NEXT and PREV FIELD keys on the FDT editor keypad to make room.

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.

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, or Bottom 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 and enter a correct value.

USER ERRORS

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 neither an integer between zero and 12, nor 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 selector will rest by the default option. Enter a valid numeric value for the field or leave it blank.

B.2 INTERNAL ERRORS

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

The output file specified in the CONVERT command cannot be written because of 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 form.

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

An attempt to read the specified frame failed.

INTERNAL ERRORS

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

A record is not accessible to 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 because of a processing error. FDT continues to convert the remaining records in the file.

FDT<xxx,yyy> - Cannot update index record for frame <frame-id>.

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

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

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

FDT<xxx,yyy> - Could not read file index zz.

The specified file cannot be opened because the file index cannot be read.

INTERNAL ERRORS

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 because of a processing error.

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

The current file cannot be saved in a permanent file because of 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 because of a processing error.

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

The named frame cannot be saved in the permanent file because of 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 specified file cannot be opened because of a processing error in the file manager.

APPENDIX C

POSRES STATUS BLOCK CODES

POSRES uses the status block parameter to return error and status information to the calling program. It is recommended that your task check the status block after each POSRES call.

Decimal status values are returned to the calling task in a two-word integer array. The first column of Table C-1 shows the values returned in the first word of the status array. The second column lists the values returned in the second word of the status array, except for menu routine errors, which are shown in Table C-2.

In Table C-1, the numbers one and two represent the first and second status block words, respectively. In your application, the first word may be array element zero, one, or n, depending on which programming language you are using. For example, BASIC-PLUS-2 numbers arrays from zero while PASCAL lets you define your own numbering scheme.

Table C-1: POSRES Status Values

Status Word 1	Status Word 2	Description
+1	1 through 12	For menus: option selection was successful, and the second word contains the ordinal option number.
+1	ASCII code	For GETKEY, the second word contains an ASCII decimal code representing a keyboard key.

POSRES STATUS BLOCK CODES

Status Word 1	Status Word 2	Description
+1	Undefined	For other routines, there was no error.
+2	0	A record or field was truncated.
+2	-6	A message sent to the message board was truncated to 59 or fewer characters and displayed.
+2	Key code	For GETKEY, the second word contains one of the function key codes listed in Appendix D.
+2	Key value	For PRSCSI routine, a valid CSI sequence was entered.
-1	DSW	RSX DSW error. See the <i>P/OS System Reference Manual</i> for error codes.
-2	I/O code	I/O status error code returned from a QIO\$ directive. See the <i>P/OS System Reference Manual</i> .
-3		File access error:
	-1	Index record not 256 bytes long.
	-2	No match during index operation.
	-3	File index record is greater than one block.
	-4	File is not open.
	-5	Frameid is not in Radix-50 character set.
-4	See Table C-2	Error executing help routine.
-5	See Table C-2	Error executing menu routine.
-6	See Table C-2	Error executing dynamic menu routine.
-7	-11	Invalid CSI sequence found
-7	-12	No CSI sequence found.
-7	-13	File extension error.
-9		Calling parameter error.
-10	0	Insufficient buffer space.

POSRES STATUS BLOCK CODES

Status Word 1	Status Word 2	Description
-11		Short message error.
	-1	No matching entry number.
-12	RMS error code	PRO/RMS-11 File access error. See the PRO/RMS-11 manual set for error codes.
-13	See Table C-2	Error executing menu unpack routine.
-14	Key code	Option selection failed. The second word contains one of the function key codes listed in Appendix D. For example, the value 14 indicates that the user pressed the ADDTNL OPTIONS key.
-15	See Table C-2	Error displaying frame specified with frameid.
-16	See Table C-2	Error executing multiple-choice menu routine.
-17	See Table C-2	Error executing menu pack or unpack routine.
-18		Error executing OLDFIL routine:
	-1	No choices made.
	-2	No files found.
	-3	Error in wildcard selection.
-19		Buffer error. The buffers FL\$BUF and MM\$BUF are not large enough.
-20	-1	Message rejected. The Message/Status Display is full. The Display lists up to 255 messages.

POSRES STATUS BLOCK CODES

Table C-2: Menu Service Routine Errors

Status Word 2	Description of Error
-1	Option number greater than maximum.
-2	Multiple once-only fields.
-4	Error in keyword definition.
-5	Title or text field length error.
-6	Text field length error.
-8	Argument error or unknown fieldid.
-9	Buffer error.
-10	Text or option line number greater than maximum.
-11	Error in menu packing.
-12	No options for multiple-choice menu.
-13	Multiple-choice menu limits responses to zero.
-14	More responses allowed than options.
-15	No help available.

APPENDIX D
FUNCTION KEY NAMES AND CODES

Table D-1 shows the function keys on the Professional keyboard. The table gives the following information:

- o **Decimal code:** The POSRES routine GETKEY returns this value in the status array parameter.
- o **PRO label:** The Professional keyboard names each key.
- o **P/OS label:** The keyboard label strip that comes with P/OS provides alternative names for each key.

Table D-1: Function Key Names and Codes

Code	PRO Label	P/OS Label
1	F1	F1
2	F2	F2
3	F3	BREAK
4	F4	SETUP
5	F5	F5
6	F6	Reserved
7	F7	RESUME
8	F8	CANCEL
9	F9	MAIN SCREEN

FUNCTION KEY NAMES AND CODES

Code	PRO Label	P/OS Label
10	F10	EXIT
11	F11	F11
12	F12	F12
13	F13	F13
14	F14	ADDTNL OPTIONS
15	HELP	HELP
16	DO	DO
17	F17	F17
18	F18	F18
19	F19	F19
20	F20	F20
21	FIND	FIND
22	INSERT HERE	INSERT HERE
23	REMOVE	REMOVE
24	SELECT	SELECT
25	PREV SCREEN	PREV SCREEN
26	NEXT SCREEN	NEXT SCREEN
27	UP ARROW	UP ARROW
28	LEFT ARROW	LEFT ARROW
29	DOWN ARROW	DOWN ARROW
30	RIGHT ARROW	RIGHT ARROW
31	PF1	PF1
32	PF2	PF2
33	PF3	PF3

FUNCTION KEY NAMES AND CODES

Code	PRO Label	P/OS Label
34	PF4	PF4
(Application Mode only)		
35	-	-
36	,	,
37	.	.
38	ENTER	ENTER
39	0	0
40	1	1
41	2	2
42	3	3
43	4	4
44	5	5
45	6	6
46	7	7
47	8	8
48	9	9

APPENDIX E

P/OS ERROR CODES

P/OS displays error codes in the following situations:

- An application cannot be started. In this case, P/OS displays an error code in the form of two decimal numbers. These error codes are listed in Section E.1.
- An application task exits abnormally (with exit status other than EX\$SUC). P/OS displays (in decimal numbers) the first two words of the exit status block. If the exit status (the first word) is 4 (EX\$SEV), something in the task caused the system to abort. In that case, the second word is also significant. The error codes for this case are listed in Section E.2.

E.1 APPLICATION CANNOT BE STARTED

Table E-1: Elements of the Status Array

First Word	Second Word	Meaning
-1	0	The installation command file is missing or cannot be opened.
-2	nxx	The installation command file contains errors. The status number in the error code lists the installation command file line (n) which contains the error followed by numbers (xx) that mean the following:

APPLICATION CANNOT BE STARTED

First Word	Second Word	Meaning
		00 RUN line expected
		01 Invalid parameter count
		02 Invalid first keyword
		03 Invalid filename spec
		05 Invalid string on line (improperly quoted)
		06 Invalid frameid
		07 Extra characters on line
		08 Invalid task name
-3	nxx	An application task cannot be installed. The status number in the error code lists the installation command file line (n) which contains the error followed by PROTSK error numbers (xx).
		01 Task name in use
		02 File not found
		03 Specified partition too small
		04 Task and partition base mismatch
		07 Length mismatch common block
		08 Base mismatch common block
		09 Too many common block requests
		11 Checkpoint area too small
		13 Not enough APRs for task image
		14 File not a task image
		15 Base address must be on 4K boundary
		16 Invalid first APR
		18 Common block parameter mismatch
		20 Common block not loaded
		22 Task image virtual address overlaps common block
		23 Task image already installed
		24 Address extensions not supported
		26 Checkpoint space too small, using checkpoint file
		27 No checkpoint space, assuming not checkpointable
		29 Invalid UIC
		30 No pool space
		31 Invalid use of partition or region
		32 Access to common block denied
		33 Task image I/O error
		34 Too many LUNs
		35 Invalid device
		36 Task may not be run
		37 Task active
		39 Task fixed
		40 Task being fixed
		41 Partition busy
		43 Common/task not in system

APPLICATION CANNOT BE STARTED

First Word	Second Word	Meaning
		44 Region or common fixed
		45 Cannot do receive
		47 Invalid request
		48 Cannot return status
		49 Error encountered on file open operation
		50 Error encountered on file close operation
		51 Cannot get file LBN to process label blocks
		99 Too many INSTALL commands in installation command file (maximum is 31)
-4	n	Reserved for internal use
-5	n	Reserved for internal use
-6	n	Error occurred during Main Menu display
-7	n	Error occurred while loading PRO/Communications software driver
-8	n	Error occurred while loading OPTIONS Graphics software

E.2 SYSTEM ABORTED TASK

Word 2 is reported to the end user as positive, but returned by P/OS as a negative number.

Table E-2: Elements of Status Array

First Word	Second Word	Meaning
4		
	0	Odd address and other traps to 4
	2	Memory protect violation
	4	Break point instruction (BPT) or trace trap (T-bit)
	6	IOT instruction
	8	Invalid or reserved instruction
	10	NonRSX EMT instruction
	12	TRAP instruction

SYSTEM ABORTED TASK

First Word	Second Word	Meaning
	14	11/40 floating point exception
	16	SST abort - bad stack
	18	AST abort - bad stack
	20	Abort via directive (ABRT\$)
	22	Task load read failure
	24	Task checkpoint read failure
	26	Task exit with outstanding I/O
	28	Task memory parity error
	30	Task aborted with PMD request
	32	TI: virtual terminal was eliminated
	34	Task installed in 2 different systems
	36	Task aborted due to bad affinity (required Bus runs are off-line or not present)
	38	Task has run over its time limit

INDEX

- Action form, 5-10
 - help menu, 5-25
 - single-choice menu, 5-21
- Action string
 - global, 8-26
 - option, 8-3, 8-16, 8-26
- ADB
 - see Application Diskette Builder
- Additional Options flag, 8-3, 8-4, 8-16, 8-21
- ALUN\$ directive
 - Communications, 2-2
- APPL\$DIR, 6-8
- APPL\$DST, 6-13, 6-17
- Application
 - multiple diskette, 1-5
- Application directory, 1-1
 - and Fast Install, 3-1
- Application Diskette Builder, 1-1
 - to 1-5, 6-23
 - and Fast Install, 3-3
 - and installation command file, 1-1
 - bad block checking, 1-3
- Application installation file
 - see Installation
- Background task, 6-22
- Bad block
 - checking by ADB, 1-3
- BASIC-PLUS-2, 6-27
- CALLABLE PRO/SORT, 11-1
- CET
 - see PROSE
- Checkpoint file, A-3
- COBOL-81, 6-27
- CODEC input on TMS line, 2-45
- Collating sequence
 - PRO/SORT, 11-6
- Communications
 - Base System Services, 2-1
 - CCANS routine, 2-18, 2-45, 2-46
 - CCATA routine, 2-2, 2-10, 2-11, 2-46, 2-48
 - CCATT routine, 2-10, 2-11
 - CCAUXK routine, 2-46
 - CCBRK routine, 2-24
 - CCDET routine, 2-11
 - CCDIAL routine, 2-15
 - CCDTMF routine, 2-48
 - CCFLSH routine, 2-23
 - CCGMC routine, 2-7, 2-12
 - CCHNG routine, 2-21, 2-48
 - CCKILL routine, 2-25
 - CCLCRG routine, 2-7, 2-9, 2-13
 - CCLCRP routine, 2-7, 2-9, 2-13
 - CCMODE routine, 2-45
 - CCMTT routine, 2-9, 2-14
 - CCORG routine, 2-19, 2-45, 2-46
 - CCPTGV routine, 2-47
 - CCRXD routine, 2-22
 - CCSMC routine, 2-7, 2-11
 - CCSPWN routine, 2-27
 - CCTXD routine, 2-21
 - COMLIB, 2-1, 2-2, 2-3, 2-9
 - CPHREC routine, 2-28
 - CPHSEL routine, 2-29
 - device names, 2-2
 - FTATT routine, 2-35, 2-40
 - FTDET routine, 2-32, 2-36, 2-40, 2-42
 - FTLISN routine, 2-44
 - FTNTFY routine, 2-38, 2-40
 - FTOPRN routine, 2-36, 2-40
 - FTOPTG routine, 2-33
 - FTOPTP routine, 2-33
 - FTSERV routine, 2-43
 - FTSYNC routine, 2-32, 2-40, 2-41
 - FTUNPK routine, 2-32, 2-40, 2-41
 - getting line characteristics, 2-12
 - line characteristics, 2-5
 - line descriptor block, 2-4
 - modem support, 2-8
 - service categories, 2-1
 - status value, 2-2, 2-4
 - translate table, 2-8
 - XKDRV, 2-1, 2-9, 2-10
- Copying application to Public Library, 6-1

INDEX

- CPRNT
 - condition codes, 9-2
 - error codes, 9-3
- CSI sequence
 - parsing, 8-32
- Deleting application from Public Library, 6-2
- DIBOL, 6-27
- Diskette
 - labelling, 1-4
- Display form, 5-5, 5-11, 5-14
 - help menu, 5-24
 - help text frame, 5-27
 - message frame, 5-28
- Dynamic menu
 - displaying, 8-3
- Escape sequence
 - PROSE, 10-3
- Event flag
 - Communications, 2-2
- Fast Install, 3-1 to 3-3
 - and Application Diskette Builder, 3-3
- Fatal error handling, 8-8
- FCS
 - see File Control Services
- FDT
 - see Frame Development Tool
- File Control Services, 4-1 to 4-3
 - and Micro/RX, 4-2
 - support for various languages, 4-2
- File Selection Menu, 8-30
- File specification
 - P/OS default, 8-13, 8-19
 - use of, 8-28, 8-30
- Filename
 - input routine, 8-28
 - selection routine, 8-30
- FORTRAN-77, 6-27
- Frame
 - computing buffer size, 5-5
- Frame Development Tool, 5-1 to 5-31
 - ACTION command, 5-1, 5-10, 5-12, 5-14, 5-30
 - ADD command, 5-1, 5-4, 5-13, 5-29
 - and POSRES, 5-1
 - CONVERT command, 5-4
 - DELETE command, 5-1, 5-5
 - DISPLAY command, 5-11, 5-14, 5-30
 - DISPLAY command, 5-1
 - error messages, B-1
 - Errors, 5-29
 - EXIT command, 5-5, 5-11
 - FILE command, 5-1, 5-6
 - file types, 5-6
 - HELP command, 5-1, 5-6, 5-12
 - LIST command, 5-1, 5-7
 - MODIFY command, 5-1, 5-7, 5-13, 5-30
 - NAME command, 5-1, 5-7
 - on PRO/Tool Kit, 5-3
 - on RSX-11M/M-PLUS, 5-3
 - on VAX/VMS, 5-2
 - PROFILE command, 5-1, 5-12, 5-14, 5-30
 - QUIT command, 5-1, 5-3, 5-8, 5-10, 5-12, 5-31
 - REPORT command, 5-1, 5-8
 - SAVE command, 5-1, 5-8, 5-13, 5-30
 - use of keypad, 5-16
 - WINDOW command, 5-9
- Function key
 - codes, D-1
- GBLDEF option
 - Communications, 2-35
- Help file
 - closing, 8-10
 - opening, 8-12
- Help frame
 - default, 8-12, 8-13
 - specifying, 8-14
 - displaying, 8-11
- I/O status block
 - Communications, 2-3
- Input
 - single keystroke, 8-9
- Installation
 - and Application Diskette Builder, 6-1
 - and running, 6-1
 - Application, 6-2

INDEX

- ASSIGN HELP command, 6-8, A-4
 - ASSIGN LOGICAL command, 6-9
 - ASSIGN MENU command, 6-10, A-4
 - ASSIGN MESSAGE command, 6-11
 - command file, 6-1 to 6-27
 - and Fast Install, 3-2
 - errors, A-1
 - format, 6-2
 - PRO/SORT, 11-2
 - PROSE, 10-2
 - used by ADB, 1-1
 - comment delimiter, 6-12
 - EXECUTE command, 6-12
 - FILE command, 1-4, 6-3, 6-12, 6-19, 6-23, A-2
 - INSTALL command, 6-3, 6-21, A-2, A-3
 - options, 6-26
 - MOUNT command, 1-3, 1-4, 6-23, A-6
 - NAME command, 6-24, A-2, A-6
 - of application, 6-1
 - optimization, 6-26
 - order of commands, 6-3
 - RUN command, 6-25, A-1, A-2, A-5
- Keypad
- use with FDT, 5-1
- Keystroke
- input routine, 8-9
- Library
- Public, 6-1, 6-2
- Line descriptor block
- Communications, 2-4
- Logical name
- in Installation Command File, 6-9
- Logical unit number
- Communications, 2-2
- MACRO-11, 7-1
- and FCS, 4-2
 - and PRO/SORT, 11-2
- Menu file
- closing, 8-15
 - opening, 8-18
 - reading from, 8-20
- Message
- sending, 8-25
- Message file
- reading from, 8-33
- Message/Status Display, 8-25, C-2, C-3
- Micro/RSX
- and FCS, 4-2
- Mini-Exchange
- code characters, 2-17
- Modem
- support for, 2-8
- Multiple diskette application
- P/OS Hard Disk, 1-4
- Multiple-choice menu
- displaying, 8-21
 - packing, 8-23
- New File Specification form, 8-28
- P/OS
- error codes, E-1
- PASCAL, 6-27
- PMA
- see Professional Macro Assembler
- POSRES, 8-1 to 8-35
- and FDT, 5-1
 - and registers, 8-1
 - DMENU routine, 8-3
 - DPACK routine, 8-5
 - FATLER routine, 8-8
 - GETKEY routine, 8-9, C-1, C-2
 - HCLOSE routine, 8-10
 - HELP routine, 8-11
 - HFILE routine, 6-8, 8-12
 - HFRAME routine, 8-14
 - MCLOSE routine, 8-15
 - MENU routine, 8-16
 - MFILE routine, 6-10, 8-18
 - MFRAME routine, 8-20
 - MMENU routine, 8-21
 - MPACK routine, 8-23
 - MSGBRD routine, 8-25
 - MUNPK routine, 8-26
 - NEWFIL routine, 8-28
 - OLDFIL routine, 8-30, C-3
 - omitting parameters, 8-2
 - parameter checking, 8-2
 - parameter format, 8-2
 - parameter types, 8-1
 - PRSCSI routine, 8-32, C-2
 - RDMSG routine, 6-11, 8-33

INDEX

- status block, 8-1
- status codes, C-1
- valid frame identifiers, 5-4
- WTRES routine, 8-35
- Print Services, 9-1 to 9-9
 - CPRINT routine, 9-1
 - CPRV3 routine, 9-4
 - P/OS Hard Disk, 9-4
- PRO/Communications Services, 2-1
- PRO/RMS-11, 4-1
 - and Fast Install, 3-3
- PRO/SORT
 - calling, 11-1
 - COLLATE command, 11-3, 11-4, 11-5, 11-10
 - command file
 - nesting, 11-4
 - specifying, 11-1
 - comment, 11-3
 - data types, 11-8
 - DEFAULT command, 11-3, 11-5
 - error codes, 11-15
 - event flag number, 11-2
 - example, 11-18
 - FIELD command, 11-3, 11-7, 11-9
 - FILE command, 11-2
 - FORCE command, 11-3, 11-10
 - INCLUDE command, 11-3, 11-11
 - INPUT command, 11-3, 11-13
 - maximum record lengths, 11-2
 - OUTPUT command, 11-3, 11-13
 - PROCESS command, 11-2, 11-3, 11-13
 - pseudo-fields, 11-9
 - record formats, 11-2
 - SORT command, 11-3, 11-13
 - WRITE command, 11-2, 11-3, 11-9, 11-15
- Professional Application Builder, 6-25, 8-1
 - Communications, 2-3
- Professional Macro Assembler, 7-1 to 7-2, 8-1
 - PRO/Tool Kit, 7-1
 - RSX-11M/M-PLUS, 7-1
 - VAX/VMS, 7-2
- Profile Form, 5-14
- Profile form, 5-4, 5-12
- help menu, 5-23
- help text frame, 5-26
- message frame, 5-28
- single-choice menu, 5-18
- PROSE, 10-1 to 10-6
 - Callable Editor Task, 6-27, 10-1
 - event flag number, 10-1
 - margin settings, 10-3
 - status codes, 10-3
- PROTSK routine (POSSUM), 6-22
- Public Library
 - copying application to, 6-1
 - deleting application from, 6-2
- RDMSG
 - default, 8-34
- REMOVE command (DCL), 6-22
- RESUME key
 - programming, 8-35
- RSX-11M-PLUS
 - and FCS, 4-2
- Single-choice menu
 - displaying, 8-16
 - dynamic
 - packing, 8-5
 - static
 - unpacking, 8-26
- SORT-11
 - ALTSEQ command, 11-5
- SYSLIB, 8-25
- Task name, A-3
- TASK option, 6-25
- Telephone Management System, 2-1, 2-9
- Terminal
 - detaching, 10-2
- TMS
 - see Telephone Management System
 - code characters, 2-16
- Translate table
 - format, 2-8
- Wildcard
 - use of, 8-30, 8-31

Do Not Tear - Fold Here and Tape

digital

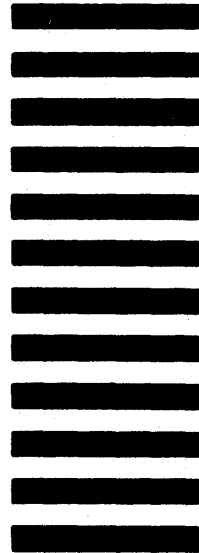


No Postage
Necessary
if Mailed in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

Professional Workstations Publications
DIGITAL EQUIPMENT CORPORATION
146 Main Street, MLO21-2/T76
Maynard, Massachusetts 01754-2571



Do Not Tear - Fold Here

Cut Along Dotted Line