Professional Developer's Tool Kit Reference Manual

Order No. AA-BT74A-TH

April 1984

This document provides reference information for some of the components of the Professional Developer's Tool Kit.It is intended for programmers developing applications for Professional 300 series personal computers.

DEVELOPMENT SYSTEM:	VAX/VMS V3.0 or later RSX-11M V4.0 or later RSX-11M-PLUS V2.0 or later P/OS V2.0
SOFTWARE VERSION:	Host Tool Kit V2.0 PRO/Tool Kit V2.0

DIGITAL EQUIPMENT CORPORATION
Maynard, Massachusetts 01754

First Printing, April 1984

CONTENTS

FIGURES

TABLES

PREFACE

Document Objectives

This manual describes some of the tools that make up the Professional Developer's Tool Kit. For some tools, this manual provides the only source of information. For others, this manual supplements other manuals. Tools not included in this manual are documented in their own manuals.

Intended Audience

This document is intended for programmers who are familiar with the Tool Kit development cycle and at least one of the Tool Kit languages. It is recommended that you read the Tool Kit User's Guide before using this manual.

Document Structure

This manual describes 11 tools, one per chapter:

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

The error message lists for ADB and FDT are extraordinarily large, thus Appendices A and B are devoted to those lists.

Appendices C and D provide tabular information related to POSRES user interface programming and are duplicated in the Tool Kit User's Guide.

Appendix E lists the error codes returned by the Professional Operating System (P/OS).

Associated Professional 300 Series Documentation

This manual is part of the Tool Kit documentation set. Related documentation is referenced throughout the book. For abstracts and order numbers of other Tool Kit documents, see the Tool Kit User's Guide.

Syntax Conventions

In this manual, syntax diagrams are presented in a format intended to make them easy to read and understand. Most languages do not require that you format your code in any particular way; therefore, you should not regard the formats used in this manual as mandatory.

Because some languages include the symbols normally used as conventions in syntax diagrams, the following conventions are used:

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 distinguish what you type from what the computer types.

CHAPTER 1

APPLICATION DISKETTE BUILDER (ADB)

The Application Diskette Builder (ADB) is a Professional 350 application that creates a master copy of an application that can be reproduced for distribution. The diskette is the distribution medium for Professional software. ADB uses the information in your application command (.INS) file to copy your application from the hard disk to one or more diskettes, from which it can be installed on other systems.

A P/OS Hard Disk application can span several diskettes. A P/OS Diskette application must be contained entirely on one diskette. Multiple applications may be copied to a single diskette if space allows. For example, you may be able to distribute the Hard Disk and Diskette versions of your application on a single diskette.

1.1 USING THE ADB

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

1. If you have not already done so, transfer all application files (task images, frame files, installation command file, and so forth) to a single directory on your Professional.

2. If you have not already done so, install the ADB. The Tool Kit includes a diskette labelled "Application Diskette Builder." Use P/OS Disk/diskette Services to install it.

3. To invoke the ADB, select it from the P/OS menu on which it was installed. The ADB dialogue is described in Section 1.2.

NOTE

During the dialogue, you can press MAIN SCREEN to return to the P/OS Main Menu.

1.2 THE ADB DIALOGUE

1. ADB displays an introductory frame. Press DO to continue.

2. **Choose the target system(s).**

   ADB displays the menu for choosing the target system for your application. Choose P/OS Hard Disk, P/OS Diskette, or both.

3. **Specify the .INS file(s).**

   ADB displays a list of installation files in the current directory. Select an installation file for each target system selected. (If you selected both target systems, ADB will prompt for two installation files.) The installation file can reside in any hard disk directory. Press the ADDTNL OPTIONS key if you want 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 file name

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

   When you have selected a file, ADB checks it for errors. Refer to Appendix A and press RESUME to try again.

4. **Ready the 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 .INS 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.

NOTE

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

5. **Create the checkpoint file, if needed.**

ADB prompts for the number of blocks to be allocated for a checkpoint file.

NOTE

A checkpoint file is needed for applications that run on P/OS Diskette systems. If your application will run on P/OS Hard Disk only, specify zero.

If your application, along with its resident libraries, callable system tasks, device drivers and graphics support will fit in the default checkpoint space of 200 blocks (100kb), simply press DO. If your application requires more than 200 blocks of memory, create a checkpoint file.

To calculate the maximum number of blocks, multiply the number of blocks that the largest task requires by the number of tasks in the application. This calculation will always give you enough checkpoint space. Allocate this many blocks if there is sufficient room on the diskette. For example, if the application consists of two tasks, one requiring a maximum of 48kb and one requiring a maximum of 20kb, you would allocate 192 blocks for the checkpoint file.

To determine the minimum number of blocks required for the checkpoint file, add the maximum size of all tasks in the application in kilobytes, and multiply by two. Do not count libraries, only tasks. For example, for the same application, with tasks requiring 48kb and 20kb, you would allocate 136 blocks for the checkpoint file.

NOTE

This second calculation determines the minimum number of blocks. More may be needed. If you use this calculation and your application fails during testing, restart the ADB and allocate more blocks at this stage.

6. **Copy the .INS file to the first diskette.**

- If you are building for a P/OS Diskette application, ADB now automatically copies the installation file to directory [ZZAPPL] as DISKETTE.INS.

- If you are building a P/OS Hard Disk application, ADB will ask for the name of the application directory to be created on the target diskette. You can specify any name; the .INS file will be copied to an .INS file with the same name.

- If you are building for both hard disk and diskette systems, and you intend to use the same application directory for both, specify the name [ZZAPPL] for the new directory of the hard disk .INS file. ADB will name the installation file ZZAPPL.INS. You can place the application files for P/OS Diskette and P/OS Hard Disk in separate directories if diskette space permits. Use separate directories only if different application tasks or files are to be used on the two systems.

If an .INS file contains errors, ADB will not copy the file and will display an error message. Check the format of the .INS file, correct the file, and start again.

7. **Copy the files to the target diskette(s).**

The ADB now copies the files listed in the FILE lines of the .INS file to the target diskette. If ADB encounters a MOUNT line (P/OS Hard Disk only), it will prompt you to insert a new diskette and will repeat the initialization process in step 3. ADB then resumes copying files with the next FILE line, and continues until all MOUNT and FILE lines have been accounted for.

ADB will check for a directory specification for each filename. If the filename includes a directory specification, ADB will look in that directory for the file to be copied. If the filename does not include a directory specification, ADB will look in the directory that contains the .INS file. In either case, if ADB cannot find the file, it will display a menu listing all hard disk directories. Select the directory in which the file resides.

If the filename includes a directory specification, ADB copies that file to that directory, creating a directory automatically if none of that name already exists on the target diskette. If the filename does not include a directory specification, ADB copies that file to the directory to which it copied the .INS file. If for any reason the ADB cannot copy a file, ADB displays an error message, then redisplays the first ADB screen. The files previously copied will, of course, still be on diskette. You may want to delete them to reclaim that space.

You will see a confirmation message when all files have been copied.

8. **Label the completed diskette(s).**

You should label your application diskettes with:

- The application's name
- The volume name (This must match the name provided in the corresponding MOUNT line, if any.)
- A brief description of the application's purpose
- The number of diskettes on which the application resides
- Minimum hardware configuration required
- Hardware options required
- Media space required

If your application spans more than one diskette, you should also indicate which is the first diskette, that is, which diskette contains the application's .INS 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 three 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 another 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 three kinds of 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:

Device Name	Represents
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 single-length 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 double-length status value in the first parameter in the call line. 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 double-length 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 has 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 routine only)
CS.FUN	2	Successful, function key exit Second word contains the key value
CE.UNO	-1	Unsupported message option
CE.MFE	-2	Message format error
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 Subsytem 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 Subsytem not attache
CE.MSC	-4	Message type out of sync
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.	CTAB error in msg, menu etc.

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. (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. The characteristic positions are as follows:

Byte 0 : Transmit Rate (Bits per sec)

Byte 1 : Receiver Rate (Bits per sec)

Byte 2 : Number of Data Bits

Byte 3 : Number of Stop Bits

Byte 4 : Parity Checking and Generation

Byte 5 : Odd or Even parity

Byte 6 : XON/XOFF Recognition

Byte 7 : Software 7-bit characters

Byte 8 : Auto answer ring count

Byte 9 : Modem Type

Byte 10 : Dial Mode

Byte 11 : Data Mode

Byte 12 : DTMF Tone Time (10 ms multiples)

Byte 13 : DTMF Interdigit Time (10 ms multiples)

Byte 14 : TMS Silence Detect Timeout

Byte 15 : Reserved

To change a line characteristic, first call CCGMC or CCLCRG to get either the current settings or the default settings. 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. Note that 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 occurs (for example, you can specify the 200 baud rate only for TMS lines).

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 communications 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 ususally 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 format characters from the phone number. (The normal format characters for a phone number are "[","]", "(",")","-" and " ". An example is [12] (345) 678-9012.)

You can set the translate table either by calling the Base System routine CCMTT (see "Base System Services" later in this chapter), or executing the PRO/Communications Setup Utility (see "PRO/Communications Services" later in this chapter).

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 startup and the Base System routines.

2.2.1 Base System Startup

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

2.2.2 Base System Routines

This section describes the communications 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

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.

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
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, ttble)

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

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.
ldb	The line descriptor block. The block size must be 16 bytes.
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
CS.NTB	Successful, no translate table defined
CE.PRM	Service call parameter error
CE.DIR	RSX Directive error Second status word contains the DSW
CE.RMS	File I/O error PRO/RMS-11 error code passed in second word
CE.IEA	Internal error

Notes:

These routines perform file I/O to the communications setup 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 setup 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 setup 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 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. No status return indicates this condition.

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

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 double-length array used as an IOSB.

tmo The time out 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 possible time out interval that you can specify is 255 decimal seconds. If the time out value is larger than 255 decimal seconds, the routine uses 255 seconds.

If you do not specify tmo, the QIO terminates immediately.

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:

Before issuing the dial, the 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 time out (or no time out parameter), the 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 non-zero time out, the request completes after the time out period or after a connection is established. If the time out period expires, then the 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 time out period expires, the routine returns IS.SUC in the first word of the I/O status block.

When dialing a call, the 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).

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
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 connections 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)

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 double-length array to be used as an IOSB.

tmo The optional time out 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 you do not specify a time out 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 time out parameter. When using CCDIAL, the driver automatically raises DTR before dialing a call.

If you specify a zero time out 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 word of the IOSB.

If you specify a non-zero value in time out, the request completes either after the time out period or after a connection is established. If the time out 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 time out period expires), the routine returns IS.SUC in the first word of the IOSB.

2.2.2.10 Hangup 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 driver drops DTR and RTS when CCHNG executes.

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.
iosb	A double-length 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 buffer becomes full. The driver has a limited amount of buffer space.

If you specify XON/XOFF support, the driver transmits XOFF when the buffer becomes three-quarters full XON when the buffer is emptied to the one-quarter point.

If the buffer becomes full, additional 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.

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 double-length 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 time out 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:

In either of the DTMF modes, the touch tone keys are presented as data characters. Further, in CODEC mode the use of the touch tone keys is signaled via an auxiliary keyboard, unsolicited input AST.

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

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 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 utility programs 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.

The following sections describe PRO/Communications startup, the utility programs you can execute, and the routines that allow you to perform Phone Book and File Transfer operations.

2.3.1 PRO/Communications Startup

When you install PRO/Communications on the Professional, the system automatically copies the utility programs to the directory [ZZCOMM], together with menu, message and help files. Also during installation, the system creates a directory to hold the PRO/Communications application. This application generates a menu through which you can select the various utility programs.

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

- loads the TMS driver, if you have installed the TMS hardware on the system
- sets up the default line characteristics for the TMS lines
- starts up the background File Transfer Subsystem.

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

2.3.2 PRO/Communications Utility Programs

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

The utility programs supplied with PRO/Communications are:

- Communications Setup - 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 utility 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 utility runs a call control task that you can use to make connections via entries stored in the Phone Book.

- Phone Book Maintenance - allows users to maintain the phone book. The phone book is a file containing information about how to connect a Professional 350 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 utility programs 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 Setup
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

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.

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.

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 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 ASYNC 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 Susbsystem 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 alread attached the File Transfer Subsystem, the Subsystem rejects further requests 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.

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:

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.

pswrd 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 Subsytem 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 double-length 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

NOTE

These routines perform file I/O to the communications setup 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 setup 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 setup 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 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', paswrd)
```

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.

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.

paswrd 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 to 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
'READ', fspec1, len1, fspec2, len2
'DELE', fspec1, len1
```

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
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 wild cards).

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

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 specify only a single file specification (no wild cards).

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

len1 The length of 'fspec1.'

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.

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 connnect 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 RAD50 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

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 file name 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 specify 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 startup in progress

CE.IEA - Internal error

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', blcknum,
                'DEVC', dev,
                'UNIT', unit,
                'TIME', elpstim,
                'OPER', operation   )
```

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

SIZE returns a double-length integer indicating the size of the file you are transferring.

filesize A double-length 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.

blcknum A double-length integer that receives the current block number of the file you are transferring.

DEVC option returns the device mnemonic for the current file operation.

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 double-length 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:

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)

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

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

2.3.4.14 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
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 startup and the TMS routines.

2.4.1 TMS Startup

The PRO/Communications application handles TMS startup. See the section on "PRO/Communications Startup" earlier in this chapter for details.

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.

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 (Touchtone 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)

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

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 routine returns an unsolicited event AST trap upon ecountering the sequence. CCDTMF reports this by setting an event flag if you have used CCATA to attach the line. The line must be attached.

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 is a Professional 350 application that allows you to install an application from a directory on the hard disk. (P/OS Disk/Diskette Services can install an application from diskette only.) 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, select Disk/Diskette Services, and select Install Application. When finished, remove and store the diskette.

2. Select Fast Install from the menu on which it was installed.

3. Fast Install prompts for the name of your application directory. Enter it and press DO.

4. Fast Install opens the .INS file and checks the format. If it finds no errors, Fast Install displays a single-choice menu consisting of a list of P/OS menus. Select the menu on which you wish to install your application and press DO.

5. After successful installation, you can run your application from the P/OS user interface.

Notes:

- Fast Install will install P/OS Diskette applications only if the REQUIRE, OPTIONS, and LOAD lines are commented out. Remember that you will then be testing a P/OS Diskette application on a hard disk system.

- If the application directory is not found, or does not not contain an installation command (.INS) file with the same name, installation fails.

- If a PRO/RMS-11 record error occurs while Fast Install is opening the .INS file, look up the error code in the PRO/RMS-11 manual set. If you cannot correct the problem, submit a Software Performance Report (SPR) to DIGITAL; be sure to include the PRO/RMS-11 error code.

- If the .INS file contains an error, Fast Install prints one of the error messages shown in Appendix A. Fast Install and the Application Diskette Builder parse installation commands the same way and thus use the same error messages.

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 that 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 which 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 V2.0 includes a vectored FCS resident library. Thus, you can move applications that use FCS to P/OS without conversion to PRO/RMS-11.

FCS is included in the Host and PRO/Tool Kits only for the purpose of moving existing applications to P/OS. Other than this chapter, FCS not described in the Tool Kit documentation set. Use PRO/RMS-11 when you write new applications.

FCS does not support as many features of P/OS as PRO/RMS-11. DIGITAL will enhance P/OS and PRO/RMS-11 but will not add these enhancements to FCS. These improvements will likely occur in the areas of networks, logical name handling and the file system. Programs using FCS will not be able to take advantage of these enhancements.

4.1 FEATURES SPECIFIC TO FCS ON P/OS

FCS for P/OS includes several features not in FCS for RSX-11M and RSX-11M-PLUS. The names in parentheses following each feature identify the modules which have been modified to support the feature. Other modules are the same as in FCS for RSX.

1. Named directories (ASCPPN, DIDFND, DIFND, DIRFND, PARDI, PARSDI)

   P/OS uses directory names consisting of one to nine alphanumeric characters in the set A to Z, 0 to 9. FCS uses these directory names also.

   The P/OS "current directory", the directory implied if a directory name is not explicitly specified, is also used by FCS.

   In addition to named directories, FCS also recognizes numeric directories in User Identification Code (UIC) form.

   Square or angle brackets can enclose directory specifications.

2. Logical device names (PARSDV)

   FCS translates P/OS logical names for devices, volume names for example, to the appropriate physical device name. However, FCS does not translate logical names which are not in the form of device names.

3. VMS-style version numbers (CBTA, ODCVT, PARSFN)

   FCS for P/OS uses version numbers in the decimal radix (as opposed to octal radix). Version numbers can be separated from file types by a period (or dot ".") or by a semicolon.

4. Other differences (FCSMAC, FCSPRE)

   To support named directories and logical device names, the File Storage Region impure data region ($$FSR2) is larger by 82 bytes than its size on RSX systems.

5. Standard P/OS devices

   FCS supports all the standard P/OS devices: the dual RX50 diskette drives as DZ1: and DZ2:, the hard disk drive (if any) as DW1:, the Professional terminal (keyboard and monitor) as TT1:, TT: or TI:, and the printer (if any) as LP: or TT2:. The Professional monitor is supported as a VT102-compatible output device.

4.2 LIMITATIONS OF FCS ON P/OS

1. Support by higher level languages

   Programs written in MACRO-11 assembly language can use FCS. Programs written in other Tool Kit languages (BASIC-PLUS-2, 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 probably work on P/OS provided that the programs are taskbuilt using the Tool Kit version of the System Object Module Library (SYSLIB) which includes FCS. Such use is not supported by DIGITAL.

   Other higher level languages available from vendors other than Digital may use FCS in their object-time systems. Please refer to product documentation of these software products.

2. Usage of resident libraries

   Vectored FCS is included with the Tool Kit V2.0. It is compatible with the FCSRES for Micro/RSX. A task built against vectored FCS will run on either system.

3. Support in future releases

   Programs using FCS must be rebuilt to run on new versions of P/OS.

CHAPTER 5

FRAME DEVELOPMENT TOOL (FDT)

The Frame Development Tool (FDT) is a special-purpose utility for creating menu, help, and message frames. These frames, when used with the POSRES User Interface Library, make up 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 OVERVIEW

FDT frame development 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 file name for the frame definition file. If you are creating a new file, specify the type of frames you want to create.

3. Enter a File Command. You can **Add**, **Delete**, **List**, **Modify**, **Name**, and **Report** on frames, or ask for **Help**. When you are finished with the current file, you can **Save*** the file on disk, **Convert** it to executable format and save it, open another **File** (saving the current file), or **Quit**, discarding the session.

4. Enter a Frame Command. You can enter the **Profile** form, the **Display** form, or the **Action** (menus only) form, or ask for **Help**. When you are finished with the frame, you can **Save*** the frame on disk or **Quit** to return to step 3.

5. While entering a form, you can use a special set of keypad keys. When you are finished with a form, you press **ENTER** to return to step 4.

---

* **Exit** is identical to **Save**.

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 5.12.

5.2 INVOKING FDT ON RSX-11M/M-PLUS (DCL)

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

```
$ FDT filename
```

If you do not specify a file name, FDT prompts for one. The default file type is .DAT.

If FDT is not an installed task, type:

```
$ RUN $FDT
```

and specify a file name when FDT prompts.

If the file does not exist, FDT assumes that you are creating a new file and prompts for the type of frame you want the file to contain:

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

Enter "H", "M", or "S".

NOTE

You can store frames of only one type in a file. Create individual files for help, messages, and single-choice menus.

5.3 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
```

If you do not specify a file name, 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 you want the file to contain:

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

Enter "H", "M", or "S".

NOTE

You can store frames of only one type in a file. Create individual files for help, messages, and single-choice menus.

5.4 FILE COMMANDS

File commands allow you to select alternate frame definition files and individual frames. They 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.

While working at file command level, the current file type and file name are displayed in a message at the top of your screen. If you enter a command and omit required parameters, FDT displays prompts requesting them.

NOTE

Do not specify frame identifiers with embedded spaces. POSRES cannot retrieve them correctly at run time.

5.4.1 ADD

The ADD command allows you to create a new frame, which 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?
  ```

  Type "M" for a help menu and "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.

5.4.2 CONVERT

The CONVERT command creates an executable frame definition file that can be read directly by POSRES menu services during program execution.

Format:

```
CONVERT output-file
```

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

Notes:

- The default file types for converted files are:

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

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

  ```
  x frames converted, y frames not converted.
  The largest frame, "<framename>", is n bytes long.
  ```

  The numbers x,y, and n are decimal. If there were 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 chapters on the

Professional Application Builder and the P/OS User Interface Services in the Tool Kit User's Guide).

5.4.3 DELETE

The DELETE command allows you to delete 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?

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

5.4.4 EXIT

The EXIT command saves the current frame definition file on disk; it performs the same function as the SAVE command. After saving a file, FDT displays a confirmation message naming the file saved and returns control to host system command level.

Format:

EXIT

5.4.5 FILE

The FILE command allows you to open another frame definition file.

Format:

FILE filename

filename The name of the frame definition file.

Notes:

- 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.

- If you did not save the current file before specifying another file, FDT prompts:

  Save current file?

  Type "Yes" to save it, or "No" to discard it.

5.4.6 HELP

The HELP command displays a list of FDT file commands.

Format:

HELP

5.4.7 LIST

The LIST command displays a list of the frames in the current definition 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.

5.4.8 MODIFY

The MODIFY command finds the specified frame in the frame definition 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.

5.4.9 NAME

The NAME command renames a frame.

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).

5.4.10 QUIT

The QUIT command returns control to host system command level without saving the current file. All changes made in the current editing session are discarded.

Format:

QUIT

Notes:

- All of the characters in the QUIT command must be typed.

5.4.11 REPORT

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

Format:

```
REPORT filename
```

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

Notes:

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

5.4.12 SAVE

The SAVE command saves the current file on disk and displays a confirmation message. If the file already exists, it is assigned the next higher version number.

Format:

```
SAVE
```

5.5 FRAME COMMANDS

Frame commands, allow you to change and save frames. They can be typed 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.

5.5.1 ACTION

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

Format:

```
ACTION  ALL
        NEW
        option
```

ALL specifies all options in the current menu. This 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).
- 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.

5.5.2 DISPLAY

The DISPLAY command invokes a Display form for the current frame. The Display form contains fields that correspond to the fields in the frame.

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.

5.5.3 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.

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. Otherwise all the changes made in this session will be discarded.

5.5.4 HELP

The HELP command displays a list of FDT frame commands.

Format:

```
HELP
```

Notes:

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

5.5.5 PROFILE

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

Format:

PROFILE

Notes:

- The Profile form for a single-choice menu contains a self-documenting description, a global help frame pointer, the default option, and the global action string.
- The Profile form for a help menu contains a self-documenting description, a previous help frame pointer, and the default option.
- The Profile form for a help text frame contains a self-documenting 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 self-documenting description.

5.5.6 QUIT

The QUIT command returns control to file editing command level without saving the current frame. Any changes made to the frame are discarded.

Format:

QUIT

Notes:

- All of the characters in the QUIT command must be typed.

5.5.7 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.

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. Otherwise all the changes made in this session will be discarded.

5.6 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:

- **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**

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

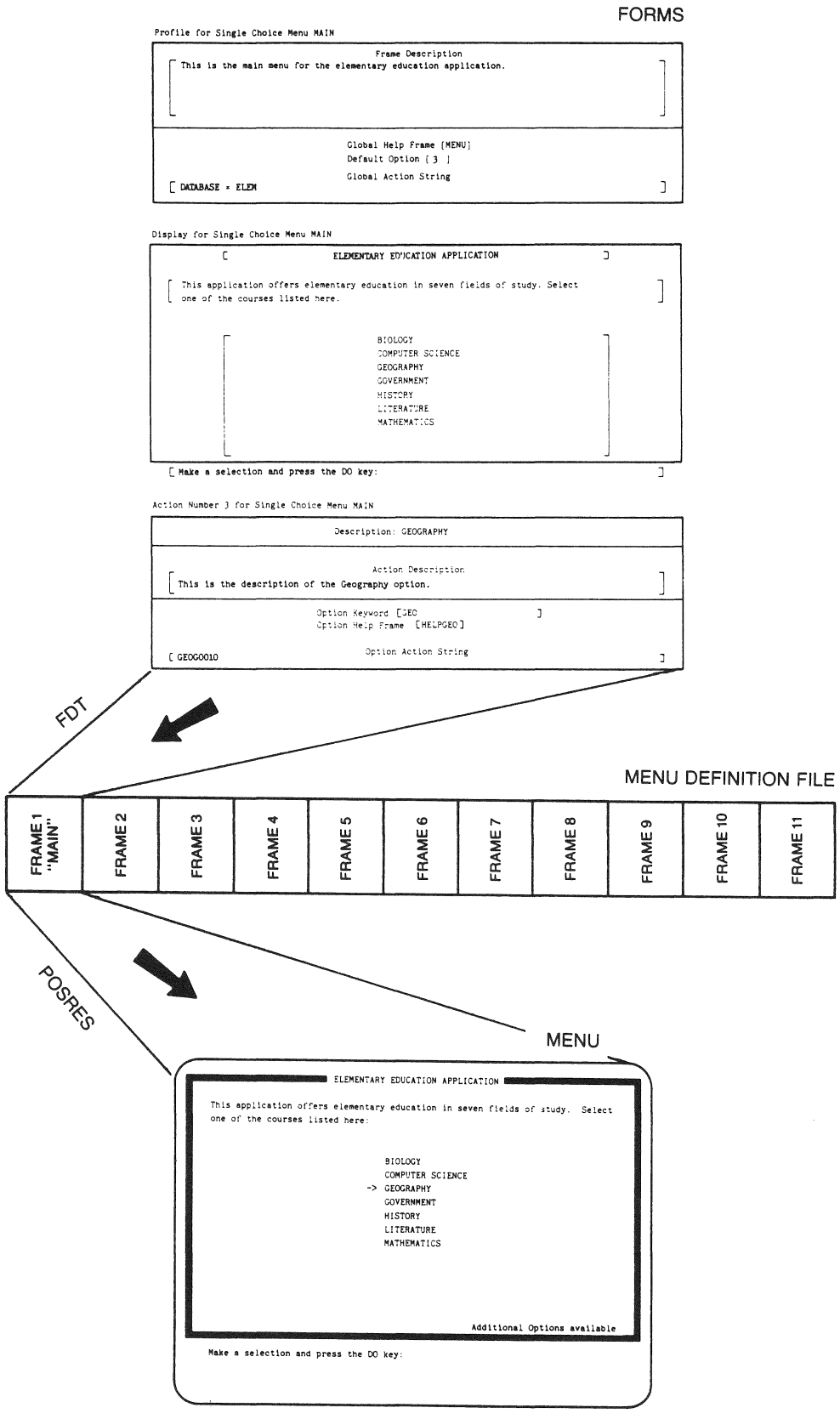

Figure 5-1: Forms for a Single-Choice Menu

Each form contains one or more fields. Some fields are self-explanatory, such as comment fields which are essentially self-documentation. Other fields are more complex and interrelated, such as help frame pointers.

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 definition file to executable format, 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 "insert mode" for entering text. There is no "insert" command. To enter text, simply type on the main keyboard.

The keypad keys operate as follows:

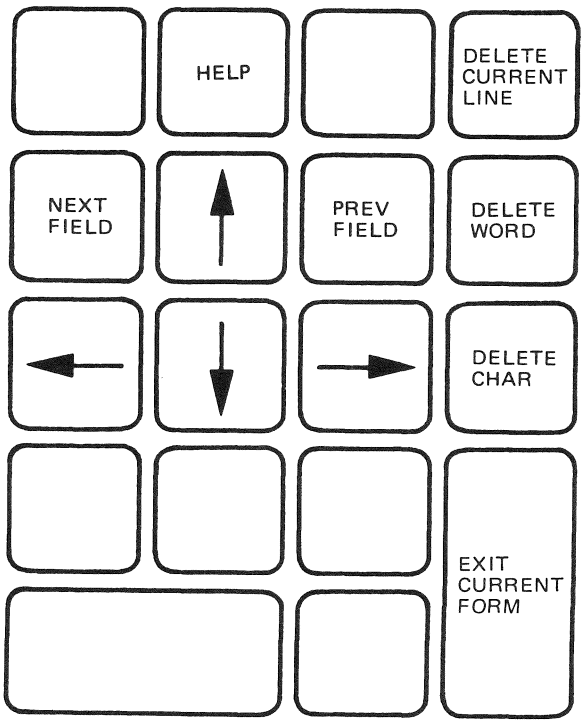

Figure 5-2: FDT Screen Editor Keypad

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 purpose.
HELP	Displays a description of the field the cursor currently points to. If pressed again, displays a description of the keypad keys.
DELETE CURRENT LINE	Deletes from the cursor to the end of the line.
DELETE WORD	Deletes from the cursor to the next word.
DELETE CHARACTER	Deletes the character the cursor is on.
NEXT FIELD	Positions the cursor at the beginning of the next field on the form.
PREVIOUS FIELD	Positions the cursor at the beginning of the previous field on the form.
EXIT CURRENT FORM	Returns control to frame editing command level unless you are filling in multiple Action forms, in which case the next Action form for the menu (if there is one) is displayed.

5.7 CREATING A SINGLE-CHOICE MENU

5.7.1 The Profile Form

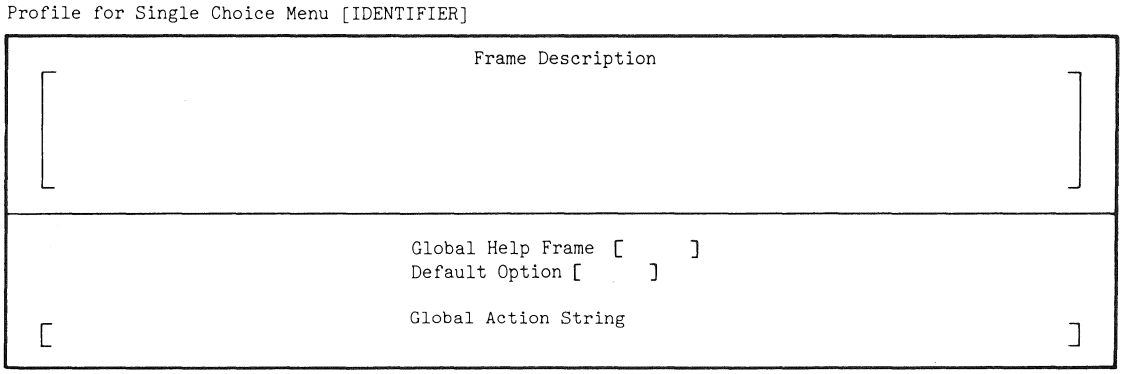

Figure 5-3: Profile Form for Single-Choice Menu

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.
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.

5.7.2 The Display Form

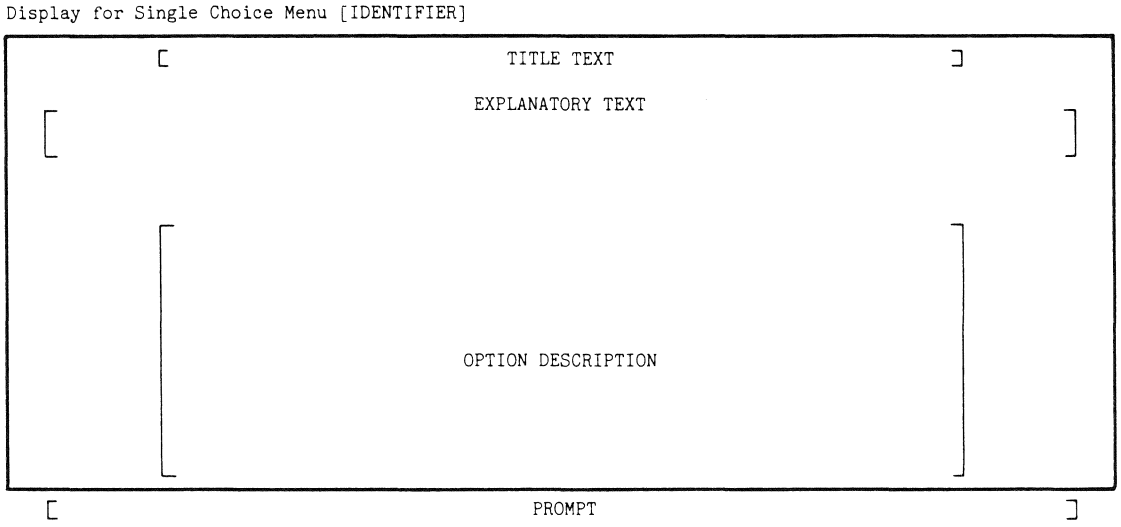

Figure 5-4: Display Form for Single-Choice Menu

Field	Type	Max. Size	Purpose
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.

5.7.3 The Action Form

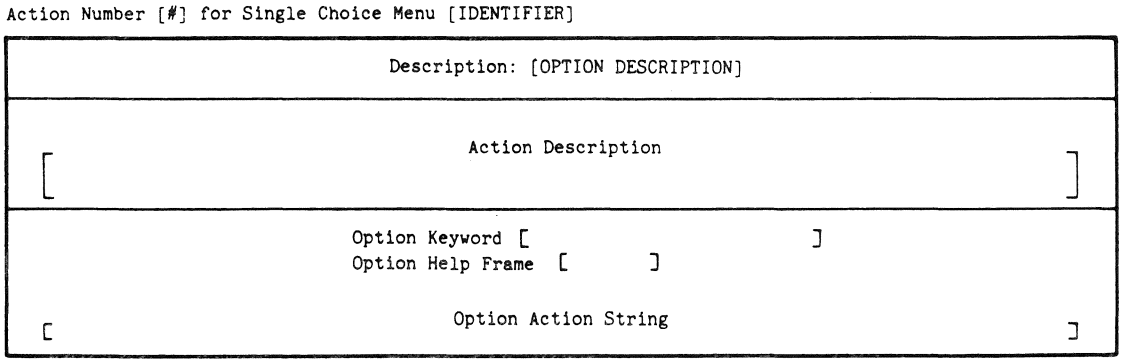

Figure 5-5: Action Form for Single-Choice Menu

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.

Field	Type	Max. Size	Purpose
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.
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.

5.8 CREATING A HELP MENU

5.8.1 The Profile Form

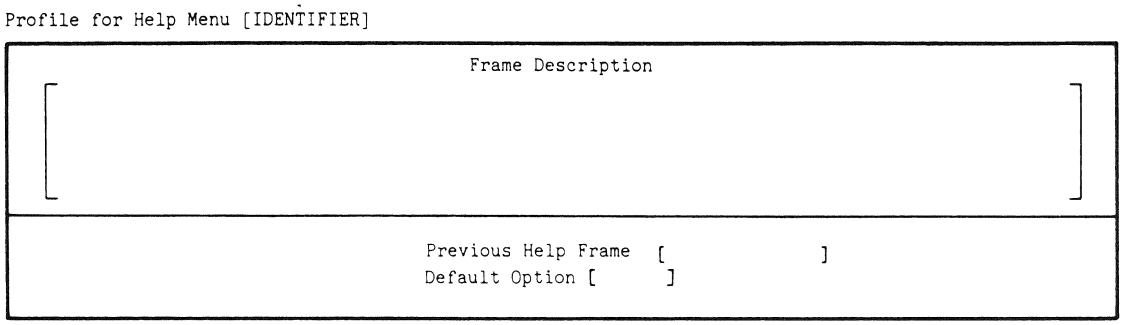

Figure 5-6: Profile Form for Help Menu

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 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.

5.8.2 The Display Form

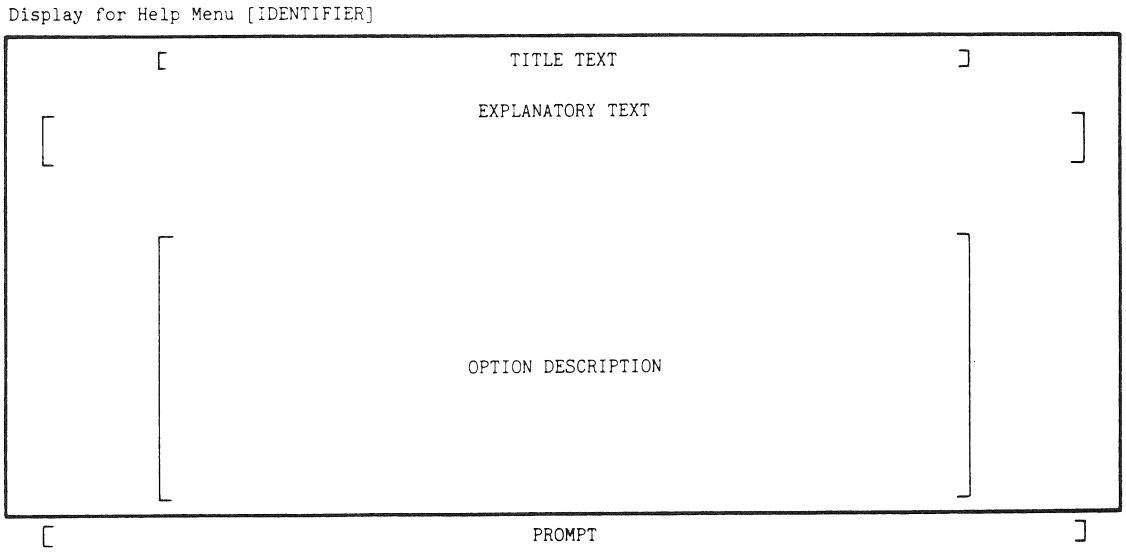

Figure 5-7: Display Form for Help Menu

Field	Type	Max. Size	Purpose
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.

5.8.3 The Action Form

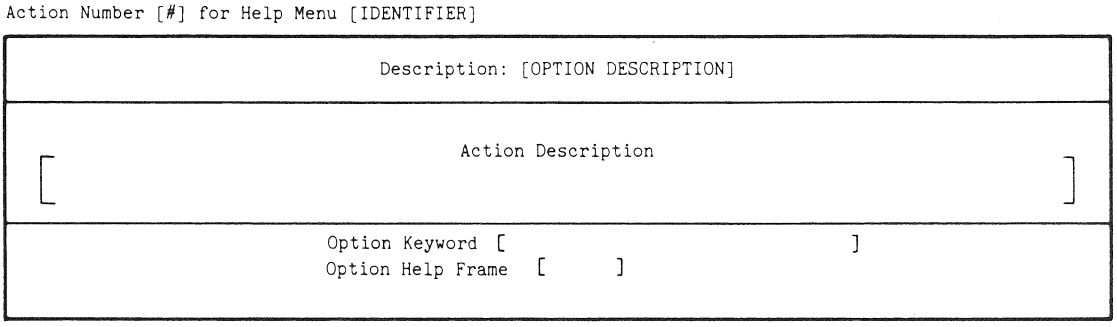

Figure 5-8: Action Form for Help Menu

Field	Type	Max. Size	Purpose
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 whether the option help frame is a menu or a text frame.
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	Required	Eight alphanumeric characters	Specify the name of the help menu or text frame to be displayed when the user selects this option.

5.9 CREATING A HELP TEXT FRAME

5.9.1 The Profile Form

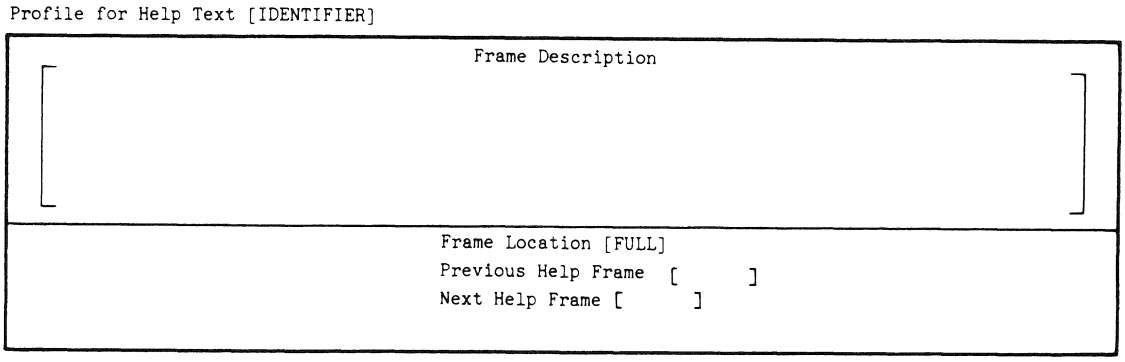

Figure 5-9: Profile Form for Help Text Frame

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 eight 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.

5.9.2 The Display Form

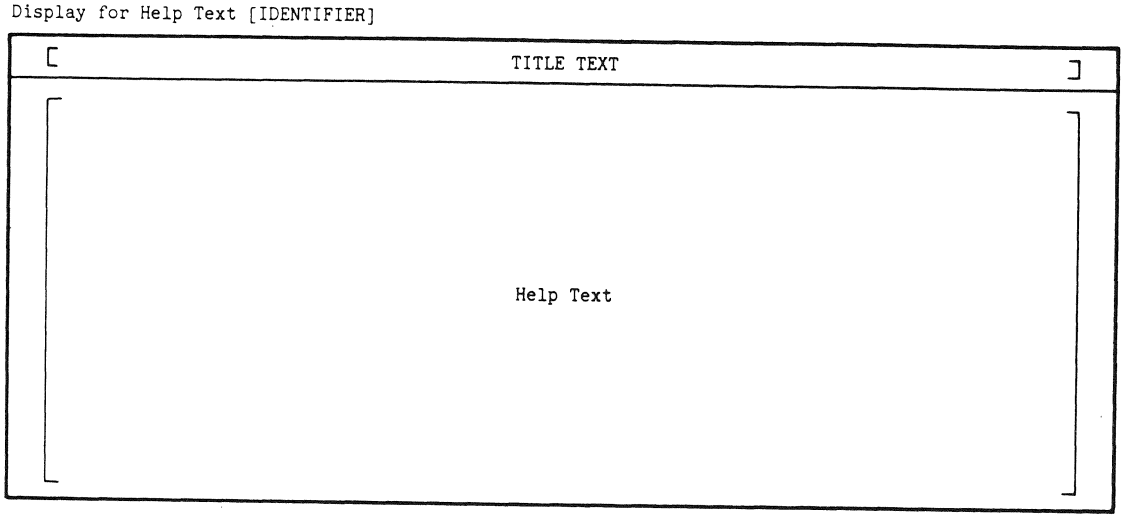

Figure 5-10: Display Form for Help Text Frame

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.

Field	Type	Max. Size	Purpose
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.

5.10 CREATING A MESSAGE TEXT FRAME

5.10.1 The Profile Form

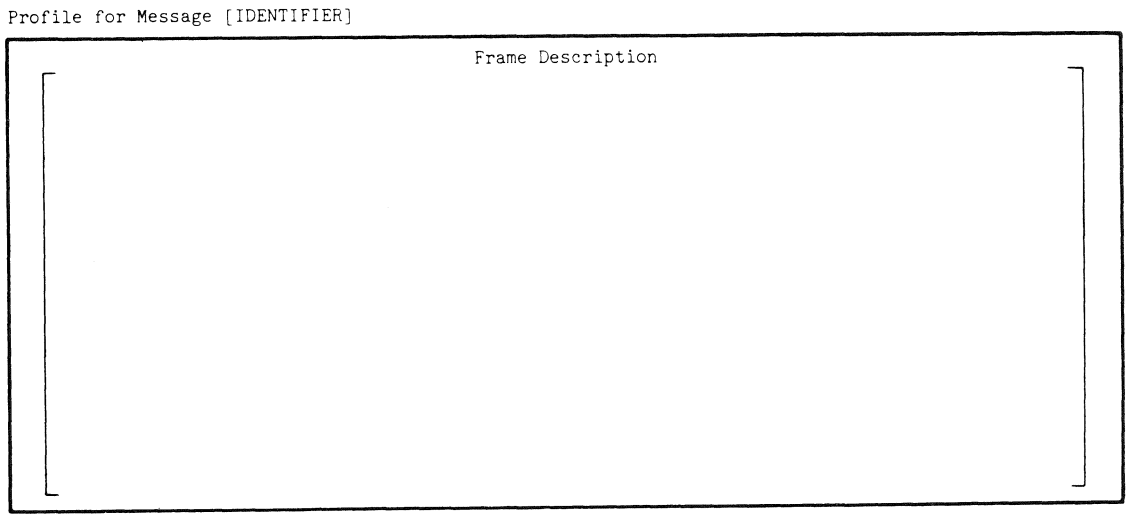

Figure 5-11: Profile Form for Message Frame

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

5.10.2 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.

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

5.11 RESOLVING ERRORS

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

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.

All other user errors produce a descriptive error messsage 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.

5.12 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 file name. 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.

```
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
```

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

The application installation (.INS) file identifies all the files and task images that make up an application. You must create separate .INS files for P/OS Hard Disk and P/OS Diskette. If your application will run on both systems, you must write two .INS files.

The .INS file is used by the following programs:

- The Application Diskette Builder uses the .INS file to identify the files and task images to copy to the application diskette(s).
- P/OS Disk/Diskette Services install service uses the .INS file to identify the files to copy to the disk at installation time.
- P/OS Diskette Prepare Application uses the .INS file to determine which files to copy from the P/OS Library diskettes onto the application diskette.
- The PRO/Dispatcher uses the .INS file at run time to identify task images (it adds the names to its list of executable tasks), to open any Menu and Help definition files (specified in ASSIGN statements), and to start the first application task.
- P/OS Disk/Diskette Services remove service uses the .INS file to identify the files to delete from the disk at application removal.

6.1 INSTALLATION COMMAND FILE FORMAT

An installation command (.INS) 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 .INS file.

The format of an .INS file is shown in Figure 6-1. Command lines can be up to 132 characters long and can have one command per line. The commands must appear in the order shown (with exceptions described under individual commands). Mandatory commands are shown in bold.

```
        NAME "name"
        FILE filename/option
1,2     MOUNT volume
        FILE filename/option
2       EXECUTE taskspec/option
3       REQUIRE filename
3       OPTIONS xxx
3       LOAD xxx
        INSTALL filename/option
        ASSIGN MENU filename
        ASSIGN HELP filename frameid
        ASSIGN LOGICAL name "equiv"
        RUN taskname

1       Required for multiple-diskette applications (see Section 6.10)

2       P/OS Hard Disk only (see individual commands by name)
3       P/OS Diskette only (see individual commands by name)
```

Figure 6-1: Installation Command File Format

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

To minimize the disk seek time and enhance the application startup process, follow these guidelines when creating an .INS file:

- Place FILE commands corresponding to INSTALL commands before any other FILE commands and arrange them in the same order as the INSTALL commands. (INSTALL commands with the /COMMON and /LIBRARY options must appear before INSTALL commands with the /TASK option.)

- Wherever possible, group FILE commands by directory. This will minimize seek time in retrieving directory information.

Subsequent sections provide descriptions of the individual commands and list the required commands for P/OS Hard Disk and P/OS Diskette applications.

6.2 ASSIGN HELP

This command assigns a default help file to your application and can be used in place of the Open Help File service routine at run time. It assumes that the specified help file is placed in APPL$DIR at installation.

Format:

ASSIGN HELP filename frameid

filename is a string of up to 13 characters that specifies the name and type of a help definition file.

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

Notes:

- The ASSIGN HELP command is optional.
- You can have only one ASSIGN HELP command per .INS file.

6.3 ASSIGN LOGICAL

This command defines a logical name at run time. (See the Tool Kit User's Guide for information about logical names.)

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.

6.4 ASSIGN MENU

This command assigns a default menu file to your application and can be used in place of the Open Menu File service routine at run time. It assumes that the specified menu file is placed in APPL$DIR at installation.

Format:

ASSIGN MENU filename

filename is a string of up to 13 characters that specifies the name and type of a help definition file.

Notes:

- The ASSIGN MENU command is optional.
- You can have only one ASSIGN MENU command per .INS file.

6.5 COMMENT (!)

An exclamation point (!) indicates that the rest of the line is a comment. Comment lines can appear anywhere in the .INS file.

Format:

! descriptive text

6.6 EXECUTE (P/OS HARD DISK ONLY)

This command causes a specified task to execute at application install time or remove time (depending on the option selected). If the task exits with status greater than one, the installation or removal aborts.

The instllation command (.INS) file executes sequentially, thus the position of the EXECUTE command relative to FILE commands is signicant, as explained below.

Format:

EXECUTE taskspec/option

taskspec the directory, file name, and file type of the task image to be executed. You can use the logical name "APPL$DST:" which equates to the application directory (see notes).

option is one of the following options:

/INS executes the specified task at application installation. This command must follow the FILE command that copies the specified task image to the hard disk. It can precede any other FILE command in order to avoid copying files needlessly on aborted installations.

/REM executes the specified task at application removal. This command must precede any FILE/DELETE commands so that if the removal aborts, no files will have been deleted.

Notes:

- The EXECUTE command is supported for P/OS Hard Disk only.
- You can place an EXECUTE command anywhere that a MOUNT or FILE command can be placed.
- The logical name APPL$DST serves the same purpose that APPL$DIR performs at run time: it equates to your application's directory. At installation or removal time, however, APPL$DIR is not defined (for your application).

6.7 FILE

This command specifies the name of an application file. You must supply a FILE command for each file to be copied at application installation time (or deleted at removal time).

Format:

FILE filename/option

filename is a string of up to 13 characters that specifies the name and type of an application file.

option must be one of the following strings (there is no default):

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.

Notes:

- You must supply at least one FILE command per .INS file.
- Do not include a FILE command for the .INS file itself.

6.8 INSTALL

This command specifies each task image or library to be installed (placed in the installed task list) at run time.

Format:

INSTALL filename/option/NOREMOVE

filename is a string that specifies the name and type of a file to be installed. You also can specify a directory name.

option 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.

NOREMOVE applies to INSTALL/TASK commands only. It specifies that the task will not be aborted or removed if the application exits or is aborted.

Notes:

- You must supply at least one INSTALL command per .INS file.
- 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 non-interactive "background" task, such as a file transfer, is not aborted inadvertantly.

  Use care with the /NOREMOVE option. If the name of the task is duplicated by another application, the second task of that name cannot be installed until the first is removed.

  Also, once you install a task with /NOREMOVE, it remains installed (it can be checkpointed if active) until another task explictly removes it (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.
- For P/OS Diskette, you must supply an INSTALL command for each task image or library listed in a REQUIRE command. (See Section 6.16 for details.)

6.9 LOAD (P/OS DISKETTE ONLY)

This command loads a device driver required for the application to run. The device driver is unloaded when the application exits. (See Section 6.16 for details.)

Format:

LOAD driver

driver is a string that specifies the physical name of the device driver to be loaded.

Notes:

- You must supply a LOAD command for each device driver required during application execution.

- P/OS Diskette V1.7 includes a configuration diskette, from which the end user can load the XK (PRO/Communications) and TMS drivers into memory. This is a user option only. Do not assume the driver will be loaded on an end user's system. If your application runs on P/OS Diskette and requires either of these drivers, include the appropriate LOAD commands in the .INS file.

6.10 MOUNT (P/OS HARD DISK ONLY)

This command specifies the names of the volumes that make up an application that is distributed on more than one diskette. At installation time, the MOUNT command specifies when the user must insert a new application diskette for installation to continue. The Application Diskette Builder copies the files named in subsequent FILE commands to the specified volume.

Format:

MOUNT volume

volume is a string of up to 12 characters that specifies a volume name (no 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.

6.11 NAME

This command specifies the application name seen by the end user when the application is installed. P/OS Hard Disk Install places this name (or one supplied by the end 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 .INS file.
- You can use single or double quotes to delimit the string.

6.12 OPTIONS (P/OS DISKETTE ONLY)

This command specifies an optional system feature that must be present on the target system for the application to run. (See Section 6.16 for details.)

Format:

```
OPTIONS feature
```

feature specifies the system feature to be included.

Notes:

- The only feature currently supported by the OPTIONS command is GRAPHICS. You must specify GRAPHICS for applications that use the CORE Graphics Library or PRO/GIDIS. Future versions of the Tool Kit and P/OS may support other options.

6.13 REQUIRE (P/OS DISKETTE ONLY)

This command specifies which system files and libraries are to be copied from the system library diskettes to the application diskette by the Prepare Application utility.

Format:

```
REQUIRE filespec
```

filespec is a string that specifies a system directory followed by a file name and type.

6.14 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 a task.

Notes:

- Include one RUN command per .INS file.
- The task name must be identical to the name specified in the TASK option (if present) in your PAB command file. If you don't use a TASK option, the task name is the first six characters of the name of the task image file on disk.

6.15 REQUIRED COMMANDS FOR P/OS HARD DISK APPLICATIONS

If application uses:	Installation file must include:
BASIC-PLUS-2	INSTALL [ZZSYS]PBFSML.TSK/LIBRARY
COBOL-81	INSTALL [ZZSYS]C81LIB.TSK/LIBRARY
DIBOL	INSTALL [ZZSYS]DBLRES.TSK/LIBRARY
FORTRAN-77	INSTALL [ZZSYS]PROF77.TSK/LIBRARY
PASCAL	INSTALL [ZZSYS]PASRES.TSK/LIBRARY
Callable editor (CET)	INSTALL [ZZSYS]PASRES.TSK/LIBRARY INSTALL [ZZSYS]CET.TSK/TASK
PRO/SORT	INSTALL [ZZSYS]PASRES.TSK/LIBRARY INSTALL [ZZSYS]PROSORT.TSK/TASK

NOTE

P/OS Hard Disk applications do not need FILE or INSTALL commands for PRO/FMS-11, PRO/GIDIS, POSRES, PRO/Communications, RMSRES, or the print utility.

6.16 REQUIRED COMMANDS FOR P/OS DISKETTE APPLICATIONS

If application uses:	Installation file must include:
(All .INS files)	REQUIRE [ZZSYS]CMAIN.TSK
	REQUIRE [ZZSYS]CTEX.MSG
	REQUIRE [ZZSYS]POSRES.TSK
BASIC-PLUS-2	REQUIRE [001002]BASIC2.ERR
	REQUIRE [ZZSYS]PBFSML.TSK
	INSTALL [ZZSYS]PBFSML.TSK/LIBRARY
Callable editor (CET)	REQUIRE [ZZSYS]CET.TSK
	REQUIRE [ZZSYS]PASRES.TSK
	REQUIRE [ZZSYS]PROSE.HLP
	REQUIRE [ZZSYS]PROSE.MSG
	REQUIRE [ZZSYS]PROSE.UDK
	INSTALL [ZZSYS]PASRES.TSK/LIBRARY
	INSTALL [ZZSYS]CET.TSK/TASK
COBOL-81	REQUIRE [ZZSYS]C81LIB.TSK
	REQUIRE [001002]C81RTE.TXT
(debugging on diskette)	REQUIRE [001002]C81DBG.HLP
	INSTALL [ZZSYS]C81LIB.TSK/LIBRARY
COMLIB	REQUIRE [ZZSYS]COMLIB.TSK
DIBOL	REQUIRE [001002]DIBOLERR.MSG
	REQUIRE [ZZSYS]DBLRES.TSK
	INSTALL [ZZSYS]DBLRES.TSK/LIBRARY
FMS	REQUIRE [001002]FMSERR.MSG
FORTRAN-77	REQUIRE [001002]PROF77.MSG
	REQUIRE [ZZSYS]PROF77.TSK
	INSTALL [ZZSYS]PROF77.TSK/LIBRARY
CORE Graphics Library	REQUIRE [ZZSYS]GDSCOM.TSK
	REQUIRE [ZZSYS]CGLFPU.TSK
	OPTIONS GRAPHICS
	INSTALL [ZZSYS]GDSCOM.TSK/LIBRARY
	INSTALL [ZZSYS]CGLFPU.TSK/LIBRARY
PRO/GIDIS	REQUIRE [ZZSYS]GDSCOM.TSK
	OPTIONS GRAPHICS
	INSTALL [ZZSYS]GDSCOM.TSK/LIBRARY
HP Plotter	REQUIRE [ZZSYS]GIHPGL.TSK
	INSTALL [ZZSYS]GIHPGL.TSK/TASK
OLDFIL/NEWFIL	REQUIRE [ZZSYS]SYSTEM.MNU

REQUIRED COMMANDS FOR P/OS DISKETTE APPLICATIONS

PASCAL	REQUIRE [001002]PASERR.MSG REQUIRE [ZZSYS]PASRES.TSK INSTALL [ZZSYS]PASRES.TSK/LIBRARY
Print services	REQUIRE [ZZSYS]CPRNT.TSK REQUIRE [ZZSYS]PRINT.MSG INSTALL [ZZSYS]CPRNT.TSK/TASK
PRODIR (POSSUM)	REQUIRE [ZZSYS]CREDEL.TSK
PROFBI (POSSUM)	REQUIRE [ZZSYS]SUMFBI.TSK
PRO/SORT	REQUIRE [ZZSYS]PROSORT.SYS REQUIRE [ZZSYS]PROSORT.TSK REQUIRE [ZZSYS]PASRES.TSK INSTALL [ZZSYS]PASRES.TSK/LIBRARY INSTALL [ZZSYS]PROSORT.TSK/TASK
TMS	REQUIRE [ZZSYS]XTDRV.TSK LOAD XT:
XK Driver	REQUIRE [ZZSYS]XKDRV.TSK LOAD XK: INSTALL [ZZSYS]COMLIB.TSK/LIBRARY

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:

```
$ RUN SYS$SYSTEM:MAC
```

or

```
$ RUN $MAC
```

7.2 INVOKING PMA ON RSX-11M/M-PLUS (DCL)

If PMA is installed on your system as "...PMA", type:

```
$ PMA command
```

where "command" is a command string in the general format "output=input" as described in the MACRO-11 documentation.

If PMA is not an installed task, type:

```
$ RUN $PMA
```

and specify a file name when FDT prompts.

7.3 INVOKING PMA ON VAX/VMS

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

symbol definition:

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

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

```
$ PMA command
```

where "command" is a command string in the general format "output=input" as described in the MACRO-11 documentation.

CHAPTER 8

POSRES USER INTERFACE LIBRARY ROUTINES

This section 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.

- 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 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.

8.2 CLOSE HELP FILE (HCLOSE)

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%() )
```

8.3 CLOSE MENU FILE (MCLOSE)

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%() )
```

8.4 DISPLAY DYNAMIC MENU (DMENU)

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 (0) 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

non-zero = 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.

Notes:

- If you specify a non-zero 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%)
```

8.5 DISPLAY HELP FRAME (HELP)

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.

Notes:

- If you specify a non-existent help frame identifier, POSRES does not return an error code in the status block.

Example:

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

8.6 DISPLAY MULTIPLE-CHOICE MENU (MMENU)

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 expresssion 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

non-zero = 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.

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%)
```

8.7 DISPLAY SINGLE-CHOICE MENU (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 (0) 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

non-zero = 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.

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)
```

8.8 FATAL ERROR (FATLER)

The Fatal Error routine informs the user of an unexpected and disabling error condition and returns control to P/OS (not 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% )
```

8.9 GET KEYSTROKE (GETKEY)

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 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
```

8.10 NEW FILE (NEWFIL)

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 file name, 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 filetype.

buflen is an integer expression that specifies the length of the filetype 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 msgl.

Notes:

- A P/OS Diskette application calling NEWFIL must also require SYSTEM.MNU in its application installation file. See Chapter 6 for details.

Example:

```
100 REM Program to test the NEW FILE Service
110 DIM Stblk%( 1% )
120 Filename$ = SPACE$( 50% )      ! Set up buffer for file name
130 Filetype$ = '.DAT'             ! Set up file name type
140 Textl$ = 'Test to check the Service of NEW FILE.'
160 Msgl$ = 'Message at bottom of NEW FILE menu.'
180 CALL Newfil BY REF (Stblk%(),
                        Filename$, 50%, Namelen%,       &
                        Filetype$, 4%, Typelen%,        &
                        Textl$, LEN(Textl$), Msgl$, LEN(Msgl$))
190 PRINT "Status = "; Stblk%( 0% ), Stblk%( 1% )
200 PRINT "Filename = "; Filename$
210 PRINT "Name length = "; Namelen%
220 PRINT "Type length = "; Typelen%
230 END
```

8.11 OLD FILE NAME (OLDFIL)

The Old File Name routine solicits the names of one or more existing files by displaying the File Selection Menu.

Format:

```
OLDFIL (status, maxfiles, files, strlens,
        wildcard, buflen, textl, buflen,
        msgl, 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.

msgl — 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 msgl.

msg2 — is a string expression of 0 to 54 characters that specifies text to be displayed on line 24.

buflen — is an integer expression that specifies the length of msg2.

Notes:

- You can use the default wild-card 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 wild-card specification. Otherwise, the same file selection menu will redisplay.

- OLDFIL requires a static buffer, a multi-buffer, and a file selection buffer, as described in the Tool Kit User's Guide.

- A P/OS Diskette application calling OLDFIL must also require SYSTEM.MNU in its application installation file. See Chapter 6 for details.

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%
```

8.12 OPEN HELP FILE (HFILE)

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.
- If your help file is in the application directory, translate the logical name "APPL$DIR:" and use the resulting file specification, including volume and directory names. A typical translation might result in an equivalence name like:

  SYSDISK:[ZZAP00012]TEST.HLP

  If you omit fields, POSRES uses the P/OS default file specification, which is almost certainly wrong. The Tool Kit User's Guide has more information about accessing application files.
- 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:', 9%, 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%)
```

8.13 OPEN MENU FILE (MFILE)

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.
- If your menu file is in the application directory, translate the logical name "APPL$DIR:" and use the resulting file specification, including volume and directory names. A typical translation might result in an equivalence name like:

  SYSDISK:[ZZAP00012]BASETEST.MNU

  If you omit fields, POSRES uses the P/OS default file specification, which is almost certainly wrong. The Tool Kit User's Guide has more information about accessing application files.

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:', 9%, FILE_SPEC$, 30%)
        !
        ! Construct filespec and open menu file
        !
        FILE_SPEC$ = FILE_SPEC$ + 'TEST.MNU'
        CALL MFILE BY REF (STATUS%(), FILE_SPEC$, LEN(FILE_SPEC$))
```

8.14 PACK DYNAMIC SINGLE CHOICE MENU (DPACK)

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 upper-case characters that specifies one of the following field identifiers:

fieldid	nn	menu field
'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
'OHLPnn'	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.

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 (OHLPnn) 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$)                   !Length of textl field
1530 Glbhlp$ = 'HELP0000'
1540 Prmtln$ = 'Select the desired option and hit DO'
1550 L3% = LEN(Prmtln$)                 !Length of prompt line
1560 Optltxt$ = 'Option 1 - the first choice'
1570 L4% = LEN(Optltxt$)                !Length of option 1 text
1580 Opt2txt$ = 'Second option - the default'
1590 L5% = LEN(Opt2txt$)                !Length of option 2 text
1600 Actstl$ = 'Action string for the first test option.'
1610 L6% = LEN(Actstl$)                 !Length of action 1 text
1620 Actst2$ = 'Action string for the second option'
1630 L7% = LEN(Actst2$)                 !Length of action 2 text
1640 Hlpfrl$ = '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$, L1%)
1680 Call Dpack BY REF &
     (Stblk%(), 'TEXT01', 6%, Txt1$, L2%)
1700 Call Dpack BY REF &
     (Stblk%(), 'GHLP', 4%, Glbhlp$, 8%, 'DFLT02', 6%)
1720 Call Dpack BY REF &
```

```
     (Stblk%(), 'PRMT', 4%, Prmtln$, L3%)
1740 Call Dpack BY REF &
     (Stblk%(), 'OPTN01', 6%, Optltxt$, L4%)
1760 Call Dpack BY REF &
     (Stblk%(), 'OPTN02', 6%, Opt2txt$, L5%)
1780 Call Dpack BY REF &
     (Stblk%(), 'ACTN01', 6%, Actstl$, L6%)
1800 Call Dpack BY REF &
     (Stblk%(), 'ACTN02', 6%, Actst2$, L7%)
1820 Call Dpack BY REF &
     (Stblk%(), 'OHLP01', 6%, Hlpfrl$, 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%)
```

8.15 PACK MULTIPLE-CHOICE MENU (MPACK)

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 upper-case characters that specifies one of the following field identifiers:

fieldid	nn	menu field
'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.

Notes:

- You can make multiple calls to MPACK.

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%)
```

8.16 PARSE STRING (PRSCSI)

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 notes).

Notes:

- 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
```

8.17 READ MENU FRAME (MFRAME)

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.

Notes:

- 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%)
```

8.18 READ MESSAGE (RDMSG)

This routine read 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. POSRES opens the message definition file in the application directory (APPL$DIR) unless you supply a directory specification.

buflen — is 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.

Example:

```
1100 CALL Rdmsg BY REF &
     (Stblk%(),'BASETEST.MSG',12%,'MESS0001',8%,B$,B%,L%)
```

8.19 SEND MESSAGE TO MESSAGE/STATUS DISPLAY (MSGBRD)

This routine sends a message to the P/OS Message/Status Display, described in the Tool Kit User's Guide.

Format:

MSGBRD (status, message, buflen)

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 non-printable characters in the message are replaced with blanks.

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

Notes:

- The MSGBRD routine is not in POSRES; it is in the system library (SYSLIB).
- You must also edit your Application Builder command (.CMD) file as described in the Tool Kit User's Guide.

Example:

```
110 REM Send a message to the message display
111 Dim Stblk%(1%)
112 Msg_file$ = "MESSAGES.MSG"
113 Frame_id$ = "12"
114 Msg$ = Space$(59%)
115 Call Rdmsg By Ref &
      (stblk%(), msg_file$, len(msg_file$), frame_id$, &
      len(frame_id$), msg$, len(msg$), length%)
116 If stblk%(0%) < 0% goto 500
140 Call Msgbrd By Ref (Stblk%(),MSG$,Length%)
150 Print Stblk%(0), Stblk%(1)
160 End
```

8.20 SPECIFY HELP FRAME (HFRAME)

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 non-existent help frame identifier, POSRES does not return an error code in the status block.

Example:

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

8.21 UNPACK MENU BUFFER (MUNPK)

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 upper-case characters that specifies one of the following field identifiers:

fieldid	nn	menu field
'TITL'		title
'TEXTnn'	01,02,03	explanatory text
'GACT'		global action string
'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
'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.

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%)
```

8.22 WAIT FOR RESUME KEY (WTRES)

This routine echoes all keystrokes except the RESUME key with a bell character. When the user presses the RESUME key, control returns to your 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

PRINT SERVICES

Print Services consists of a single callable routine that allows your application to print a file, stop, continue, abandon or restart a print job, or obtain printer status. A request to print a file creates a non-interactive task.

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:

+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 a temporary scratch pad 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 below.

-1 in the first word of the status array indicates that an error occurred. The 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, and that the second element of the array contains one of the decimal error codes show below.

Condition Codes:

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
10. Printer offline/paused
22. Print request accepted, but printer type does not match
23. Unable to determine printer status

Error Codes:

-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 Setup file. General RMS error.
-30. Error accessing Setup file. File access error.
-31. Error accessing Setup file. Error reading attributes.

-32.	Error closing Setup file.	
-33.	Error accessing Setup file.	Bad device specification.
-34.	Error accessing Setup file.	Bad directory specification.
-35.	Error accessing Setup file.	No such directory.
-36.	Error accessing Setup file.	Device not ready.
-37.	Error accessing Setup file.	File locked by another user.
-38.	Error accessing Setup file.	File not found.
-39.	Error accessing Setup file.	Bad node name.
-40.	Error accessing Setup file.	Privilege violation.
-41.	Error accessing Setup file.	File processor error.
-42.	Error accessing Setup file.	File processor error.
-43.	Error accessing Setup file.	Bad file extension.
-44.	Error accessing Setup file.	Bad file version number.
-45.	Error accessing Setup file.	Illegal wild card in merged string.
-46.	Error accessing Setup file.	Extraneous data in file specification.

Notes:

- On P/OS Hard Disk, the call to the Print Services always returns immediately with status. If a request to print a file is accepted, the print service will print the file as a non-interactive application while the calling application goes on to other processing.

- On P/OS Diskette, the call to Print Services returns status only when the print job is complete or the printer has paused for an external reason. The status will reflect this condition. Control is returned to the calling application when the print job is complete or the printer paused for some other reason.

- A P/OS Diskette application that calls Print Services must also list Print Services in its installation command file. See Chapter 6 for details.

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 file name, a temporary work file name, format options, maximum line length for text entry, and left and right margin values.

The input file name is the file to be edited. An empty or new file can also be created 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.

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 file name.

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 file name.

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 file name.

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
- non-zero — creates a new file with the name specified in outfile

format — an integer expression that specifies whether or not to retain escape sequences in the output file (see notes).

zero discards formatting data

non-zero 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

non-zero enables word wrap

lun an integer expression that must be present but is ignored by the editor task. Specify zero.

Notes:

- 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.
+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.

First	Second	Meaning
-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.
-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.

First	Second	Meaning
-13	RMS code	The UDK/Setup 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:"
        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 the last section of this chapter.

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,
        inpfile, 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.

inpfile is a string expression containing a PRO/SORT input file specification. There is no default file type.

buflen is an integer expression that specifies the number of

characters in inpfile.

status is a two-word integer array that receives status information from PRO/SORT.

Include the following in your application installation command (.INS) file:

```
INSTALL [ZZSYS]PROSORT.TSK/TASK
INSTALL [ZZSYS]PASRES.TSK/LIBRARY
```

Make sure to include a FILE command for each PRO/SORT command file. The following is a sample BASIC-PLUS-2 call to PRO/SORT:

```
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 MRS of the input file.

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.

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 may help you understand how PRO/SORT can be used to perform familiar SORT operations.

The example at the end of this section 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 how many commands are allowed in a PRO/SORT 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 ***

* Each condition in an INCLUDE line is counted 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 is counted as a WRITE line.

Comment statements in a PRO/SORT command file are indicated by an exclamation point (!) or semi-colon (;) as their first character. For example:

```
!This is a PRO/SORT comment.
;This is an alternate form of the PRO/SORT comment.
```

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 10 levels deep.

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.

AFTER Sorts the first value immediately after the second value.

If you use the COLLATE command before the DEFAULT command, the collating sequence specified by DEFAULT will supersede that specified by COLLATE. See the DEFAULT command for further details.

You can specify an unlimited number of COLLATE commands in the command file.

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, using SORT to sort commercial at (@) signs as ASCII zero (0) 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.

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, PORTUGESE, 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 that you want collated at this position. You can separate integers with commas or carriage returns.

The DEFAULT USER command is equivalent to specifying individual COLLATE commands for all 256 values of an eight-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.

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 five through nine 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: (Figure 11-1 shows how PRO/SORT interprets bytes as they occur in the input data stream).

- 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.

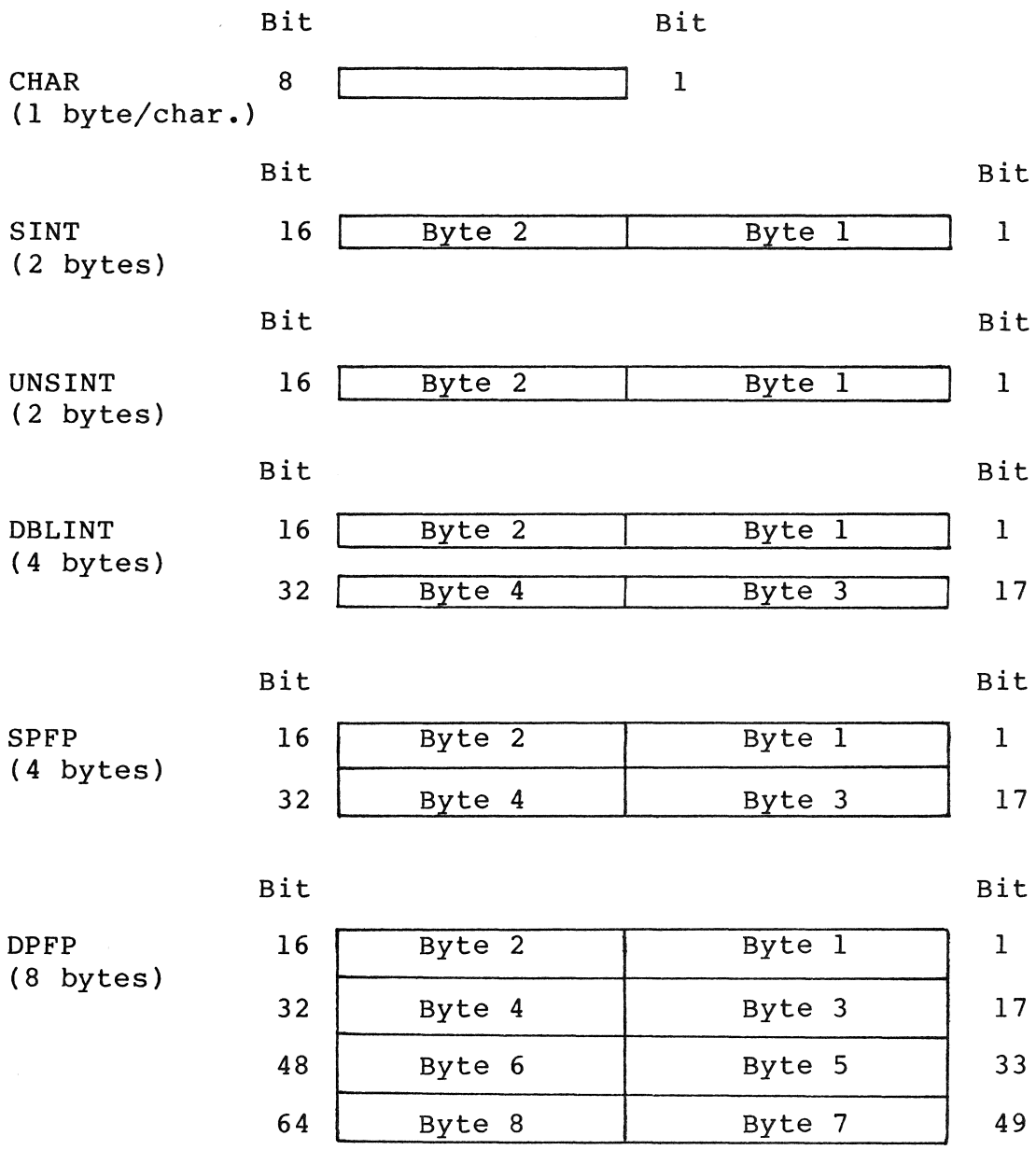

Figure 11-1: Mapping of Bytes to Fields in Input Data Stream

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), 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
```

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. Where 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 the input file so that all zip codes appear at the top of the output file:

```
FORCE ZIP TO "0"
```

You could also sort the 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 "Y"s, you would type:

```
F           50XY
```

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 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.

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.

You can specify up to 20 INCLUDE conditions in a command 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"
```

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 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 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
```

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 could not be invoked, the spawn has probably failed. Check that you have listed [ZZSYS]PROSORT.TSK/TASK and [ZZSYS]PASRES.TSK/LIBRARY on the install lines of the application installation file.

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

Error Code (decimal)	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.

-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.

-20 The PRO/SORT data file SYSDISK:[ZZSYS]PROSORT.SYS can't be opened.

-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.

-26 A field name has been referenced that does not yet exist.

-27 More than one PROCESS command in file.

-28 Too few characters specified with the DEFAULT USER command.

-29 An input record larger than 1005 bytes has been found.

-30 Insufficient disk space for temporary files.

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.

Figure 11-2: Sample 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 11-3: Sample Input File--INPUT.DAT

```
Fake    Name2  Hxyzzy
Fake    Name   Hardy
Dick    Tracy  Hero
Mary    Worth  Normal
Clark   Kent   Hero
Peter   Parker Herol
Lex     Luthor Villain
Green   Hornet Hero
Lois    Lane   Heroine
Bruce   Wayne  Herol
King    Tut    Villain
Gen.    Zod    Villain
Scooby  Doo    Herol
David   Banner Hero
Perry   White  Normal
Comm.   Gordon Normal
Mary    Jane   Normal
The     Joker  Villain
Miss    Piggy  Normal
Darth   Vader  Villain
Leia    Organa Heroine
```

The PRO/SORT command file begins with two comment lines, indicated by the exclamation points (!):

```
! 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
```

Figure 11-4: Sample Indirect Command File: FIELDS.CMD

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
```

FIELDS.CMD 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 character 8-14 receives the field name LastName, and so on. Once PRO/SORT has executed this indirect command file, the program can reference 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.

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).

Figure 11-5: Resulting Output File: OUTPUT.DAT

```
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
Hxyzzy  Fake   Name2
Normal  Comm.  Gordon
Normal  Mary   Jane
Normal  Miss   Piggy
Normal  Perry  White
Normal  Mary   Worth
```

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 .INS 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>.

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 the 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 the 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 PRO/Dispatcher must install to run this application. Correct the application installation file and restart the 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 the ADB.

Copy aborted

An error occurred while the ADB was copying files to the

Error <xxx,yyy> creating directory <dirname>.

The 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 the 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 the ADB.

Invalid task name.

A line in the application installation file that accepts task file names lists an invalid task name.

Task names can be no more than 6 characters long. Correct the application installation file and restart the 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 the ADB.

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 way in which files are copied, 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 the 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 the ADB.

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 the 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, PRO/Dispatcher executes the task listed in the RUN line. Correct the application installation file and restart the ADB.

Please enter a valid alphanumeric name.

Either no characters were entered or invalid characters were entered.

The file name is invalid

A file name in the application installation file is not a correct PRO/RMS-11 file specification.

Correct the application installation file and restart the 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 the ADB.

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 contains too many characters.

See the description of the syntax of strings in an application installation file. Correct the application installation file and restart the 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 the ADB.

Unable to bad block the diskette in <drive> -- Try another.

The diskette in the named drive could not be bad blocked 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.

Check to be sure the installation file is in the specified directory. Specify the correct application installation file.

A.2 ADB SERIOUS ERRORS

ADB<xxx,yyy> - Cannot access menu file

The ADB menu file could not be accessed.

Try running the 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 the ADB with a new diskette.

ADB<xxx,yyy> - Error while gathering additional user directories.

An error occurred while the 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 .INS file. If it does, make sure you have named the .INS file with a .INS file type. Look up the PRO/RMS-11 I/O error codes in the PRO/RMS-11 manual set, and restart the ADB. If you receive the same error message, submit an SPR.

ADB<xxx,yyy> - RMS error connecting to <filename>

The 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 the ADB again. If you receive the same error message, follow the procedures for submitting an SPR.

ADB<xxx,yyy> - RMS error creating <filename>

The 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 the ADB again. If you receive the same error message, follow the procedures for submitting an SPR.

ADB<xxx,yyy> - RMS error opening <filename>

The 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 the ADB again. If you receive the same error message, follow the procedures for submitting an SPR.

ADB<xxx,yyy> - RMS error while reading <filename>

The 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 the 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>

The 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 the 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, they result from a file operation failure. They 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:

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 hard copy listing of the file.

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, if possible.

---

* The error messages are sorted on the text, not the symbols.

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.

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

Enter "H", "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, enter a carriage return. The prompt Filename: will be displayed.

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

The named field, which is on the Action form with the indicated number, does not contain valid information. This error is detected after 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 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.

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 or 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 enter a carriage return to enter a different file name.

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 with 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.

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 in 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.

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 typograhical errors in the command line. Review the list of options on the Display form.

No new options found.

The ACTION NEW command was specified but all options have been assigned actions.

Use ACTION ALL or ACTION option-number to modify the desired options.

No option lines defined.

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

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

No value entered for required field.

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

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

Not enough 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 previous 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.

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 a form.

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

An attempt to read the specified frame failed.

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

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

FDT<xxx,yyy> - Cannot read record.

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

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

The named frame cannot be converted 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> - 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> - Closing original file after copy.

The file specified 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.

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 file specified 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 Block Words		
1	2	Description
+1	1 through 12	For menus, option selection was successful and the second word contains the ordinal option number.
	ASCII code	For GETKEY, the second word contains an ASCII decimal code representing a keyboard key.
	Undefined	For other routines, there was no error.
+2	0	A record or field was truncated.

Table C-1 (cont.)

1	2	Description
	-6	A message sent to the message board was truncated to 59 or fewer characters and displayed.
	Key code	For GETKEY, the second word contains one of the function key codes listed in Appendix D.
	Key value	For PRSCSI routine, a valid CSI sequence was entered.
-1	DSW	RSX DSW error. See the P/OS System Reference Manual for error codes.
-2	I/O code	I/O status error code returned from a QIO$ directive. See the P/OS System Reference Manual.
-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
	-12	No CSI sequence found
	-13	File extension error
-9		Calling parameter error
-10	0	Insufficient buffer space
-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.

Table C-1 (cont.)

1	2	Description
-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.

Table C-2: Menu Service Routine Errors

Word Two	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

Table C-2 (cont.)

Word Two	Description of Error
-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

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
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

FUNCTION KEY NAMES AND CODES

Code	PRO Label	P/OS Label
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
34	PF4	PF4
(Application Mode only)		
35	minus	minus
36	comma	comma
37	period	period
38	ENTER	ENTER
39	0	0
40	1	1
41	2	2
42	3	3

FUNCTION KEY NAMES AND CODES

Code	PRO Label	P/OS Label
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, the PRO/Dispatcher 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$SUC). The PRO/Dispatcher displays (in decimal numbers) the first two words of the exit status block. If the exit status (the first word) is four (EX$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.

- The system stops working and displays a picture of a Professional with none of the hardware components highlighted. In the case, the Executive displays two octal bugcheck error codes. These are listed in Section E.3. If you see one: record the codes as well as a description of the state immediately prior to the crash, and call the Hotline.

E.1 APPLICATION CANNOT BE STARTED

Elements of the Status Array

First Word	Second Word	Meaning
-1	0	The .INS file is missing or cannot be opened.
-2	nxx	The .INS file contains errors. The status number in the error code lists the .INS file line (n) which contains the error followed by numbers (xx) which mean the following:

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 .INS file line (n) which contains the error followed by numbers (xx) which mean the following:

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 illegal 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 illegal UIC
30 no pool space
31 illegal use of partition or region
32 access to common block denied
33 task image I/O error
34 too many LUNs
35 illegal 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
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 tasks installed (limit is 20)
-4	n	Reserved for internal PRO/Dispatcher use
-5	n	Reserved for internal PRO/Dispatcher 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

Elements of Status Array

First	Second	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	illegal or reserved instruction
	10	non-RSX EMT instruction
	12	TRAP instruction
	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 offline or not present)
	38	task has run over its time limit

E.3 BUGCHECK CODES

000100 P/OS Keyboard Handler error
xxxxxx

000200 Terminal Driver error
xxxxxx

100400 P/OS Terminal Subsystem error
xxxxxx

000300 P/OS Executive or other system error
000000 IOT in System State

000300 P/OS Executive or other system error
000001 Stack Overflow

000300 P/OS Executive or other system error
000002 Trace Trap or Breakpoint

000300 P/OS Executive or other system error
000003 Illegal Instruction Trap

000300 P/OS Executive or other system error
000004 Odd Address or Other Trap 4

000300 P/OS Executive or other system error
000005 Segment Fault

000300 P/OS Executive or other system error
000006 A Task on P/OS Without a Parent Aborted

000300 P/OS Executive or other system error
000007 EMT Trap

000300 P/OS Executive or other system error
000010 TRAP Trap

000300 P/OS Executive or other system error
xxxxxx Programmed call to Bugcheck macro

000400 System Startup Processing error
000001 Can't Install Task CBOOT

000400 System Startup Processing error
000002 Can't Spawn Task CBOOT

000400 System Startup Processing error
000003 Can't Spawn Task CMAIN

000400 System Startup Processing error
000007 Required File Not Found

000400 System Startup Processing error
000010 DSR corrupt

000400 System Startup Processing error
000011 Bad dispatch

000400 System Startup Processing error
000012 No way to boot via DECNA

000400 System Startup Processing error
000013 DSR allocation failure

000400 System Startup Processing error
000014 DDM vector in use

000400 System Startup Processing error
000015 Required PDV not found

000400 System Startup Processing error
000016 Required hardware not present

APPENDIX F

POSRES USER INTERFACE LIBRARY SUMMARY

Close current help definition file.

HCLOSE (status)

Close current menu file.

MCLOSE (status)

Display dynamic single-choice menu.

DMENU (status, action, buflen, strlen, display, add_opt,
msgl, buflen, msg2, buflen)

Display help frame.

HELP (status, frameid, buflen)

Display multiple-choice menu.

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

Display static single-choice menu.

MENU (status, action, buflen, strlen, display, ad_opt,
msgl, buflen, msg2, buflen)

Display "Application error..." message, followed by specified message.

FATLER (message, buflen)

Get single keystroke without echo.

```
GETKEY (status)
```

Display P/OS New File Specification form.

```
NEWFIL (status, filespec, buflen, strlen, deftype, buflen,
        strlen, text, buflen, msg, buflen)
```

Display P/OS File Selection Menu.

```
OLDFIL (status, maxfiles, files, strlens,
        wildcard, buflen, textl, buflen,
        msg1, buflen, msg2, buflen)
```

Open help definition file and set default help frame.

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

Open menu definition file.

```
MFILE (status, filespec, buflen)
```

Pack (store information in) dynamic menu buffer.

```
DPACK (status, group,...)
```

The format of a group is:

```
fieldid, buflen, fieldval, buflen

'CLRB', 4

'DFLTnn', 6

'KEYWnn', 6, offset, keylen
```

Pack (store information in) multiple-choice menu buffer.

```
MPACK (status, group,...)
```

The format of a group is:

```
fieldid, buflen, fieldval, buflen

'CLRB', 4
```

Parse string for CSI sequence.

PRSCSI (status, buff, buflen, csipos)

Read menu frame into static buffer.

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

Read message from file into buffer.

RDMSG (status, filespec, buflen, frameid, buflen,
message, buflen, strlen)

Send message to P/OS Message/Status Display.

MSGBRD (status, message, buflen)

Set default help frame.

HFRAME (status, frameid, buflen)

Unpack (return information stored in) static buffer.

MUNPK (status, group,...)

The format of a group is:

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

Wait for RESUME key.

WTRES ()

INDEX

Professional Developer's
Tool Kit Reference Manual
Order No. AA-BT74A-TH

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized?
Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) _____

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code _____
or
Country

Please cut along this line.

--- 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 300 Series Publications
DIGITAL EQUIPMENT CORPORATION
146 MAIN STREET
MAYNARD, MASSACHUSETTS 01754

--- Do Not Tear - Fold Here ---

Cut Along Dotted Line