

Professional™ 300series

CP/M-80® User's Guide
Hard Disk System

AA
PRO
V449A

digital

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

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

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

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

Copyright © 1983 by Digital Equipment Corporation
All Rights Reserved

CP/M* is a registered trademark of Digital Research Inc.

The following are trademarks of Digital Equipment Corporation:

CTI BUS
DEC
DECmate
DECsystem-10
DECSYSTEM-20
DECUS
DECwriter
DIBOL

digital

MASSBUS
PDP
P/OS
PRO/BASIC
PRO/Communications
Professional
PRO/FMS
PRO/RMS
PROSE
PROSE PLUS

Rainbow
RSTS
RSX
Tool Kit
UNIBUS
VAX
VMS
VT
Work Processor

94-0036049/32

Preface

INTENDED READER

This manual is intended for first-time users running CP/M-80* on the Professional 350 hard disk system. (CP/M is a registered trademark of Digital Research Inc.) You do not need to know how to program to use this manual. Your Professional can run the CP/M operating system when the Z80 board and CP/M applications software are installed. If you are the first user of a newly installed system, refer to the *Professional 300 Series User's Guide for Hard Disk System* before reading this manual.

MANUAL STRUCTURE

Chapter 1 introduces you to the Professional's CP/M. Chapter 1 also tells how to use CP/M menus, start operating sessions, end operating sessions, and copy your PRO-CP/M-80 SYSTEM DISKETTE.

Chapter 2 explains various CP/M operations and gets you started using CP/M. It provides practice examples of some of the basic CP/M commands. Try the exercises provided throughout this chapter to practice using the CP/M operating system.

Chapter 3 describes the CP/M commands. This chapter is a reference section for all CP/M commands except those used for creating separate diskette areas for multiple users, assembly language programming, and debugging.

Chapter 4 discusses what to do if you have a problem with the Professional or CP/M. It tells you how to isolate the problem, and what to do to correct it.

Appendix A presents the procedure for compressing virtual diskettes. (Virtual diskettes are described in greater detail later in this manual.)

Appendix B describes the error messages that are most likely to occur while you are working with CP/M.

Appendix C explains some special system calls that can be used with the Professional CP/M application. (Note: Appendix C should be used by assembly language programmers only.)

CONVENTIONS USED IN THIS GUIDE

- In examples of dialogue between you and the Professional, what the Professional displays is shown in black. What you type is shown in red.
- <RETURN> means that you must press the RETURN key on your keyboard.
- The term “application diskette” refers to the diskette that is labeled “PRO-CP/M-80 APPLICATION DISKETTE.”
- The term “system diskette” refers to the diskette that is labeled “PRO-CP/M-80 SYSTEM DISKETTE.”

INFORMATION SOURCES FOR ADVANCED CP/M USERS

This guide tries to answer questions likely to be posed by beginning users who want to run CP/M applications programs available from many software vendors. Such programs cover an extremely wide range—from number manipulation to business planning to word processing. Because users of such programs need little specific computer knowledge, this guide concentrates on those aspects of the Professional and CP/M that are useful for running applications programs and handling the data that such programs produce. The CP/M commands used in file creation and maintenance are covered in detail, as are those that direct input and output to external devices.

The CP/M software for assembling and debugging programs is not covered in this guide. Digital Research, however, produces a CP/M manual that includes detailed information on the CP/M assembler, debugger, and related programs. Readers who want to write assembly-language programs should obtain this manual by ordering:

CP/M Operating System Manual, Digital Research, P.O. Box 579, Pacific Grove, California, 93950.

The advanced CP/M user may also find the following books useful.

Thom Hogan, *Osborne CP/M User Guide*, Osborne/McGraw Hill, Berkeley, California, 1981

Rodnay Zaks, *The CP/M Handbook with MP/M*, Sybex Inc., 1980

Contents

CHAPTER 1 GETTING STARTED WITH CP/M

Introducing CP/M	1
Preparing Your Professional for CP/M	2
Using Words	3
The HELP Key	3
The MARK, BUREAU, and END Keys	4
The F10 Key	4
Invoking CP/M Programs	5
Configuring I/O Devices	5
Configuring Virtual Disks	6
Start CP/M	6
Checking Your Directory	7
Working Smoothly with CP/M	8
Copying Programs	9

CHAPTER 2 GETTING THINGS DONE WITH CP/M

Working with CP/M	11
Using the Keyboard with CP/M	12
How to Watch CP/M Responses	12
How to Use CP/M Commands	13

APPENDIX C SYSTEM CALLS

Retrieving System Information	103
Call to Specify P/OS File Names	104

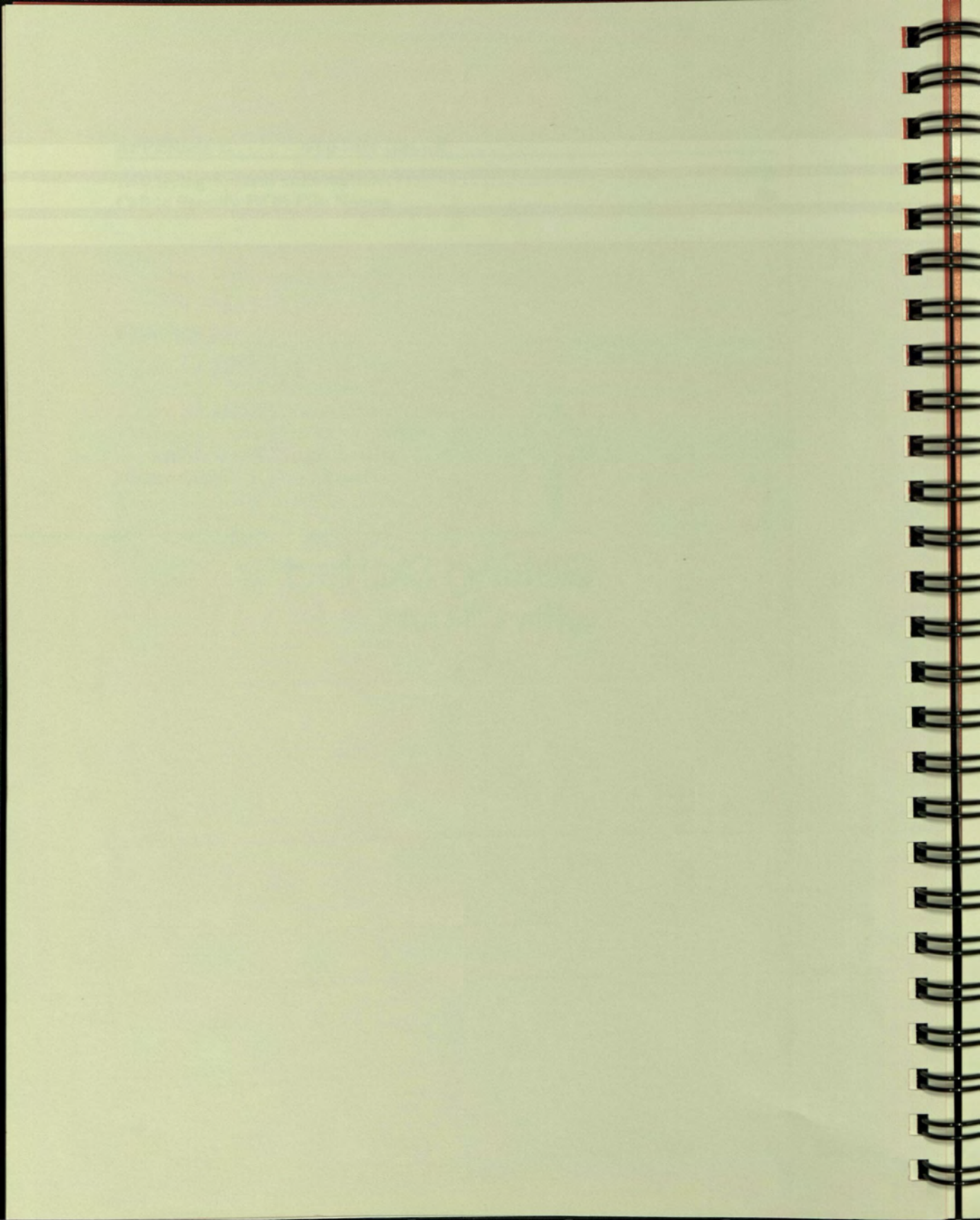
INDEX

FIGURES

Figure 1-1: CP/M Main Help Menu	4
Figure 1-2: CP/M Main Menu	5
Figure 1-3: CP/M Diskette Initialization Menu	6
Figure 1-4: CP/M I/O Device Configuration Menu	9
Figure 1-5: CP/M Virtual Diskette Configuration Menu	12
Figure 1-6: CP/M Boot Screen	14
Figure 1-7: CP/M Directory	15
Figure 2-1: Applying a Write-Protect Tab	22
Figure 2-2: Tracks and Sectors on a Diskette	30
Figure 2-3: The ED Text Buffer	30

1

Getting Started
with CP/M



Chapter 1

Getting Started with CP/M

INTRODUCING CP/M

CP/M is the Control Program for Microprocessors. Such a program is called an operating system because it governs all activities taking place in a computer. When you run CP/M applications on the Professional, they are under CP/M's control.

When you want the Professional to do something, you communicate with it by typing commands to CP/M on the keyboard. CP/M accepts commands for a variety of actions including:

- Listing the names of files on a diskette
- Copying the entire contents of diskettes
- Copying individual files
- Creating text files
- Displaying text files
- Printing files on a printer
- Deleting files
- Running programs

PREPARING YOUR PROFESSIONAL FOR CP/M

Before you use CP/M, you must first install the CP/M Z80 board and the CP/M application (contained on the diskette labeled "PRO-CP/M-80 APPLICATION DISKETTE"). The *Professional 300 Series Installation Instructions for CP/M Module* manual contains instructions for installing the Z80 board. The *Professional 300 Series User's Guide for Hard Disk System* contains instructions for installing applications.

You will want to set up CP/M on your Professional before you start a CP/M operating session. This chapter describes:

- How to use CP/M menus
- How to initialize CP/M diskettes
- How to configure virtual diskettes (virtual diskettes are described in greater detail later in this chapter)
- How to start and end CP/M operating sessions
- How to back up your "PRO-CP/M-80 SYSTEM DISKETTE" (this diskette is referred to as the "system diskette" from now on)

You will find it helpful to read through this chapter and use the examples provided before you begin to run CP/M applications programs and manipulate files.

NOTE: Before you do the examples provided in this chapter, see the *Professional 300 Series User's Guide for Hard Disk System* for instructions on how to handle and use diskettes.

USING MENUS

Your CP/M application includes several menus that help you organize your CP/M files and use certain CP/M capabilities. You will always use at least one of the menus—the CP/M Main Menu—each time you use CP/M. You will use the other

CP/M menus whenever you need to do specific tasks, but you will not necessarily use them every time you enter a CP/M operating session. The menus allow you to do the following tasks:

- Start CP/M processing
- Initialize diskettes so they can be used for CP/M processing
- Configure I/O devices
- Configure disk devices

The HELP Key

When you are not sure what to do, press **HELP**. You can press **HELP** from any CP/M menu. Information will appear to aid you. Help is available to you from the menus only. Once you start a CP/M operating session, you can no longer press the **HELP** key to display a help message.

You can press **HELP** in several different situations:

1. First, you can display general information about each CP/M menu. To do so, press **HELP** while the pointer is resting above the list of menu options. Press **RESUME** to return to the menu you were using.
2. Second, you can display the CP/M Main Help Menu, which allows you to choose from a list of Help topics. To do so, press **HELP** from the CP/M Main Menu when the pointer is resting above the list of menu options. The Help for CP/M Application Menu is displayed. Now press **NEXT SCREEN**. The CP/M Main Help Menu (Figure 1-1) is displayed. From it you can choose the topic for which you want further information. After you have displayed the information that you want, press **RESUME** to return to the CP/M Main Menu, or press **PREV SCREEN** to redisplay the CP/M Main Help Menu.
3. Third, you can display specific information about the options on each menu. Simply move the pointer to the option you want and press **HELP**. Press **RESUME** to return to your original work. (If you press **HELP** for any options on the CP/M Main Menu, you can also press **PREV SCREEN** to display the CP/M Main Help Menu.)

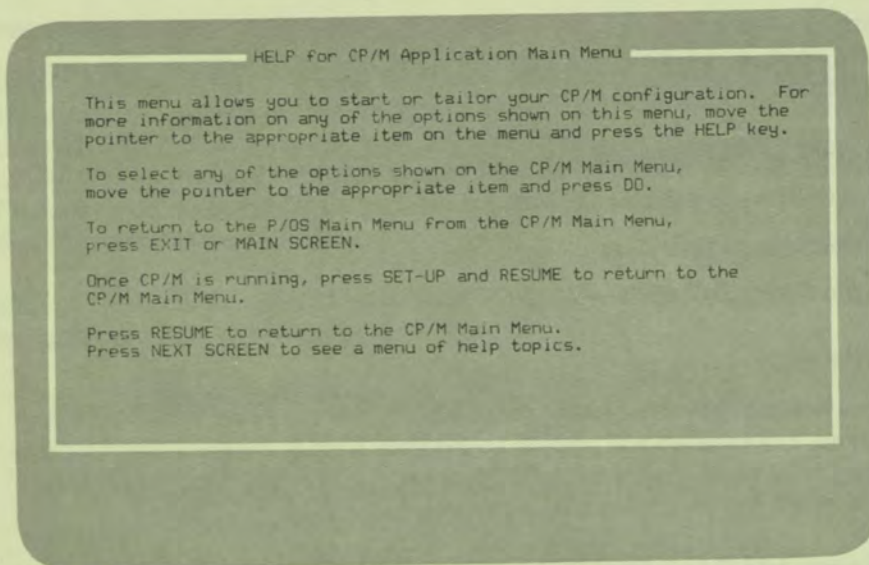


Figure 1-1
CP/M Main Help Menu

The MAIN SCREEN and EXIT Keys

While you are working with the CP/M Main Menu, pressing the MAIN SCREEN and EXIT keys returns you to the P/OS Main Menu. If you press MAIN SCREEN from any CP/M menu (**except the CP/M Main Menu**), the CP/M Main Menu is redisplayed. If you press EXIT from any CP/M menu (**except the CP/M Main Menu**), the next higher menu appears.

The CP/M Main Menu

When you want to use CP/M, you must first display the CP/M Main Menu on your screen. The procedure below tells you how to display the CP/M Main Menu.

1. Turn the Professional on by pressing the power switch to the "1" position. After a minute or so, the Professional's P/OS Main Menu is displayed. If you installed your CP/M application on the P/OS Main Menu, go to step 6. Otherwise, go to step 2.
2. From the P/OS Main Menu, press the ↓ key to move the pointer to "Additional applications" or type the boldface letters for the "Additional applications" menu option.
3. Press the DO key. The Applications Group Menu appears.

4. Press the ↓ key to move the pointer to the group that contains your CP/M application.
5. Press the DO key. The Applications Within a Group menu appears.
6. Type "CP/M" or press the ↓ key to move the pointer to "CP/M for the Professional 350." (If you chose another name for the CP/M application when you did the installation procedure, the name you chose will appear on the menu instead of "CP/M for the Professional 350.")
7. Press DO. The Professional displays the CP/M Main Menu (Figure 1-2).

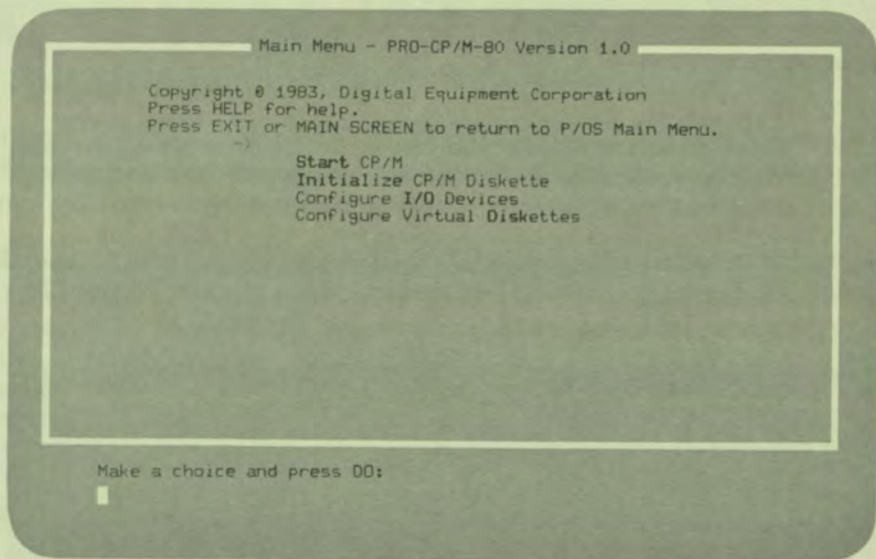


Figure 1-2
CP/M Main Menu

Look at the CP/M Main Menu that is displayed on your screen now. This menu lists four options that allow you to perform different tasks with CP/M. The sections below describe the four options on the CP/M Main Menu.

Normally, when you begin a CP/M operating session, you only need to choose "Start CP/M" from the CP/M Main Menu. The first time that you use CP/M, however, you must work with the other options on the CP/M Main Menu to initialize diskettes, configure virtual diskettes (described in more detail below), and configure I/O devices. After you have set up your diskettes, virtual diskettes, and I/O devices, you can begin a CP/M operating session simply by

selecting “Start CP/M” from the CP/M Main Menu. The sections below are therefore organized so that the “Start CP/M” option is described last.

Initialize CP/M Diskette

First, look at the “Initialize CP/M Diskette” option on the CP/M Main Menu. You use this option to do two things:

- Initialize diskettes and
- Initialize virtual diskettes (virtual diskettes are described in more detail below.)

Before you use any diskettes, including virtual diskettes, for the first time, you must initialize them.

CAUTION: The initialization procedure erases the contents of your diskette. Before you initialize diskettes, make sure you have backed up any files you want to save onto other diskettes. (This is true for diskettes only. The initialization procedure does not erase the contents of any existing virtual diskettes.)

Move the pointer to “Initialize CP/M Diskette” or type the boldface letters. Press DO. The CP/M Diskette Initialization Menu appears (Figure 1-3).

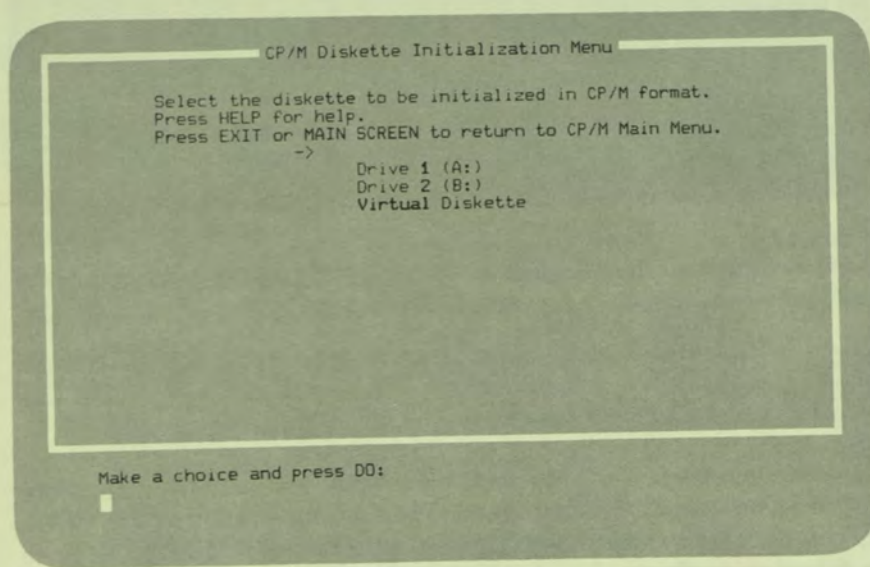


Figure 1-3
CP/M Diskette Initialization Menu

The menu allows you to initialize CP/M diskettes in drive slot 1 or drive slot 2. To initialize a diskette, insert that diskette into a diskette drive. Then select from the menu the drive that contains the diskette to be initialized and press DO. The system displays the following message:

CAUTION: Initializing will destroy all data on the diskette

Press DO to initialize diskette

Press CANCEL to return to menu

If you press DO, the system displays:

Initializing diskette....

The system then redisplay the CP/M Diskette Initialization Menu with the following message at the bottom of the screen:

Diskette initialized

If you press CANCEL, the CP/M Diskette Initialization Menu is redisplayed. The diskette is not initialized, and no message appears at the bottom of the screen.

Diskettes that have been initialized for CP/M processing cannot be used for P/OS processing. Likewise, diskettes that have been initialized for P/OS processing cannot be used for CP/M processing. Make sure that you label your diskettes carefully so that you can distinguish CP/M diskettes from P/OS diskettes.

The CP/M Diskette Initialization Menu also lets you initialize a “virtual diskette.” You have not yet worked with virtual diskettes on your Professional, so they are described in some detail below.

The term “virtual diskette” refers to P/OS files on the hard disk that are created specifically for storing CP/M files. You work with virtual diskettes in the same ways that you normally work with diskettes. You can change, protect, delete, and copy files, just as you can those on a diskette. The only difference is that you do not physically remove the hard disk at the end of an operating session (the way you remove diskettes from diskette drives).

When you initialize a virtual diskette, you are actually creating a new P/OS file for storing CP/M files on the hard disk. You can create as many virtual diskettes as you want, but you can use only four virtual diskettes at a time.

You can store more than one CP/M file on a virtual diskette if you want. The amount of space used by each virtual diskette expands to accommodate the files that it contains. The space used by a virtual diskette does not decrease, however, if you delete all or parts of the files contained there.

For example, suppose you create a virtual diskette that contains a 10-block file. If you then delete the file from the virtual diskette, the virtual diskette still takes up enough space on the hard disk to contain a 10-block file, even though it no longer actually contains any file.

Because virtual diskettes do not decrease in size when you delete files, you could waste valuable space on your hard disk by keeping virtual diskettes that have had files deleted. You will therefore want to compress your virtual diskettes from time to time. Appendix A describes how to compress virtual diskettes.

Initialize a virtual diskette now. Move the pointer to "Virtual diskette" or type the boldface letters. Press DO. The Name a File Form appears. This is the same form that you use to name files when you are working with P/OS files on your Professional. (For more information about the Name a File Form, refer to your *Professional 300 Series User's Guide for Hard Disk System*.) Enter the name CPMTEST and press DO. (Note that the system automatically assigns a file type of ".CPM" to your file. The full name of the virtual diskette you are initializing will be "CPMTEST.CPM.")

The CP/M Diskette Initialization Menu reappears when you press DO.

A message displayed at the bottom of the screen lets you know that the file has been created. Press EXIT or MAIN SCREEN to return to the CP/M Main Menu.

When you initialize a virtual diskette, you do not erase the contents of any virtual diskettes that you have already created. Even if you initialize a virtual diskette and give it the same name as an existing one, a file with a new version number of the virtual diskette is created. The file with the old version number is saved and any CP/M files that are contained on it are preserved. (For your own convenience, however, you generally should give each virtual diskette a different name.)

Configure I/O Devices

The third selection listed on the CP/M Main Menu is "Configure I/O Devices." You use this selection to name files that will allow you to transfer information between P/OS and CP/M.

From the CP/M Main Menu, move the pointer to “Configure I/O Devices” or type the boldface letters. Press DO. The CP/M I/O Device Configuration Menu appears (Figure 1-4).

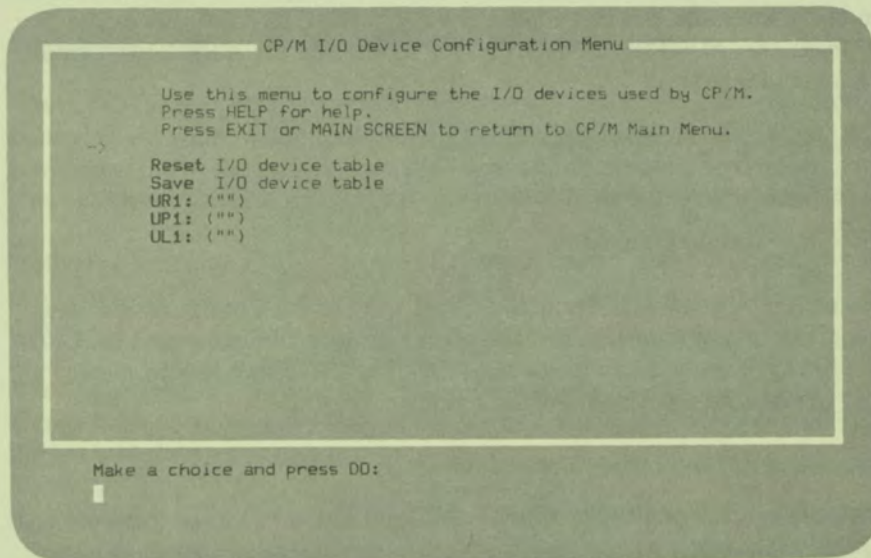


Figure 1-4
CP/M I/O Device Configuration Menu

First look at the last three options on the menu. These are:

- UR1: (“BIGVOLUME:[USERFILES]FILE.UR1”)
- UP1: (“BIGVOLUME:[USERFILES]FILE.UP1”)
- UL1: (“BIGVOLUME:[USERFILES]FILE.UL1”)

UR1, UP1, and UL1 are the names of the devices that CP/M lets you configure. After you configure a device, the volume, directory, and file name that you assign to the device appear inside the parentheses beside the device name.

UR1, UP1, and UL1 are devices that you can configure to move information back and forth between CP/M and P/OS. The first device, UR1, enables you to transfer files from P/OS to CP/M. The second device, UP1, enables you to move files from CP/M to P/OS. The third device, UL1, also enables you to transfer files from CP/M to P/OS. In other words, UR1 is used for sending input to CP/M. UP1 and UL1 are used for sending output from CP/M.

You will find it easiest to think of UR1, UP1, and UL1 as files to which you can send information. When you configure UR1, UP1, and UL1, you are actually assigning them file names.

When you choose UR1 from the CP/M I/O Device Configuration Menu, the File Selection Menu appears. You can choose the name for UR1 from this menu. You can also press the **ADDTNL OPTIONS** key to display an Additional Options Menu. This menu lets you:

1. Choose a different directory/volume
2. Display the next group of files
3. Use an extended file name

When you choose UP1 or UL1 from the CP/M I/O Device Configuration Menu, the Name a File Form appears. You can either enter a file name and let CP/M assign the file type, or you can press the **ADDTNL OPTIONS** key to display an Additional Options Menu. This menu lets you:

1. Choose a different directory/volume
2. Specify or change the file type
3. Use an extended file name

Configure UR1 now. Move the pointer to UR1 or type the boldface letters. Press **DO**. The File Selection Menu appears.

For practice, create a new file called **USERTEST.TRY** for UR1. Press the **ADDTNL OPTIONS** key. The Additional Options Menu appears.

Select "Use extended file name." Press **DO**. The Extended File Name screen appears. Type **BIGVOLUME:[USERFILES]USERTEST.TRY** and press **DO**. The CP/M I/O Device Configuration Menu reappears. (Notice that the name of UR1 is now **USERTEST.TRY**. It is stored in the **USERFILES** directory on **BIGVOLUME**.)

Now look at the first two options on the menu. These options are:

- Reset I/O device table
- Save I/O device table

Look at the “Save I/O device table” option first.

“Save I/O device table” saves the file names for all three devices exactly as you named them while using the CP/M I/O Device Configuration Menu. After you have chosen the file names that you want to remain for the devices, you select this option so that the names that you assigned become permanent.

Note that when you select “Save I/O device table,” you automatically save the file names for all three devices. You cannot save just one or two of the device names.

The other option, “Reset I/O device table,” allows you to change the device names back to the names that you assigned when you last selected “Save I/O device table.”

You have set up a file name for UR1. To save that name, move the pointer to “Save I/O device table” or type the boldface letters. Press DO. After a few seconds, the system redisplay the CP/M I/O Device Configuration Menu. (In this exercise, you named only UR1. When you select “Save I/O device table,” you save UR1’s new name, and you save the original names for UP1 and ULL.) Now press EXIT or MAIN SCREEN to return to the CP/M Main Menu.

Configure Virtual Diskettes

The next selection listed on the CP/M Main Menu is “Configure Virtual Diskettes.” When you select this option, a screen is displayed that allows you to choose the virtual diskettes that you want to use while you are working with CP/M. Move the pointer to “Configure Virtual Diskettes” or type the boldface letters. Press DO. The CP/M Virtual Diskette Configuration Menu appears (Figure 1-5).

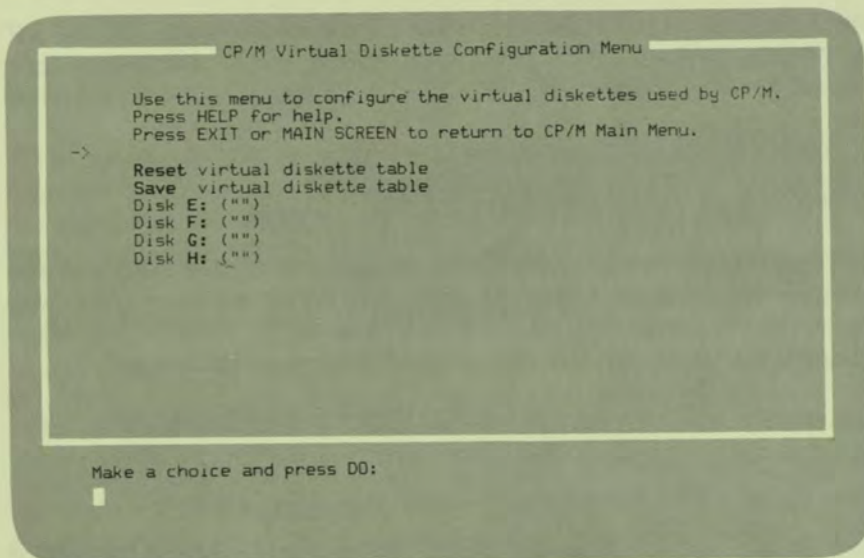


Figure 1-5
CP/M Virtual Diskette Configuration Menu

The CP/M Virtual Diskette Configuration Menu offers you six choices. Look at the last four selections on the menu first, because you will be working with them before you work with the others. These last four choices refer to the virtual diskettes. You must configure a virtual diskette before you can use it during a CP/M operating session. Generally speaking, when you configure a virtual diskette, you are telling the Professional that you want to work with a specific file as that virtual diskette.

You can work with only four virtual diskettes at a time, because CP/M will allow you to configure only four at a time. You can also choose not to configure any virtual diskettes if you do not need to use them during an operating session.

You have already initialized a virtual diskette, so now you can configure it. Move the pointer to one of the virtual diskette selections (Disk E for example), or type the boldface letters. Press DO. The File Selection Menu appears. This is the same menu that you use when you are selecting P/OS files. Move the pointer to the file named CPMTEST and press DO. The CP/M Virtual Diskette Configuration Menu reappears.

NOTE: The File Selection Menu lists all files in the current directory. Choosing a P/OS file that is not a virtual diskette will produce unpredictable results.

Now look at the first two options on the CP/M Virtual Diskette Configuration Menu. These are:

- Reset virtual diskette table
- Save virtual diskette table

Look at the “Save virtual diskette table” option first.

“Save virtual diskette table” saves the files that you selected as the virtual diskettes for the CP/M operating session. After you have chosen the file names that you want to remain for your virtual diskettes, you select this option so that the names that you assigned become permanent. Each time you select “Save virtual diskette table” from the CP/M Virtual Diskette Configuration Menu, you save the file names exactly as they appear on the screen.

Note that when you select “Save virtual diskette table,” you automatically save the file names for all four diskettes. You cannot save just one or two of the virtual diskette names.

The other option, “Reset virtual diskette table,” allows you to change the virtual diskette names back to the file names that you assigned when you last selected “Save virtual diskette table.”

To save the file names as they appear on the CP/M Virtual Diskette Configuration Menu, move the pointer to “Save virtual diskette table” or type the boldface letters. Press DO. After a few seconds, the system redisplay the CP/M Virtual Diskette Configuration Menu. (In this exercise, you configured only Diskette E. When you select “Save virtual diskette,” you save Diskette E as you specified it and you save the other virtual diskettes as being unspecified.) Now press EXIT or MAIN SCREEN to return to the CP/M Main Menu.

Start CP/M

The “Start CP/M” option enables you to begin your CP/M operating session. Whenever you choose “Start CP/M” from the CP/M Main Menu, you must insert the system diskette into drive 1 before you can actually start working with CP/M. Follow the steps below to start your CP/M operating session.

1. Move the pointer to “Start CP/M” or type the boldface letters.
2. Press DO. The CP/M Boot screen (Figure 1-6) appears.

3. Insert the system diskette into drive 1.
4. Press the RESUME key. The following message appears.

```
CP/M 2.2 for P/OS V. 1.0  
Copyright © 1981, Digital Research, Inc.
```

```
A>
```

A > is a prompt that indicates that CP/M is ready to accept commands. CP/M can access up to six drives, so it always prompts you with the name of the drive it is currently accessing (A, B, E, F, G, or H) and a right angle bracket (>).

For example, if the current, or active, drive is Drive B (drive 2), CP/M displays:

```
B>
```

It waits for a command. You can issue commands to CP/M whenever it displays the prompt.

NOTE: Drive A and Drive B refer to drives 1 and 2, respectively. Drives E, F, G, and H refer to the four virtual diskettes that you can configure for a CP/M operating session.

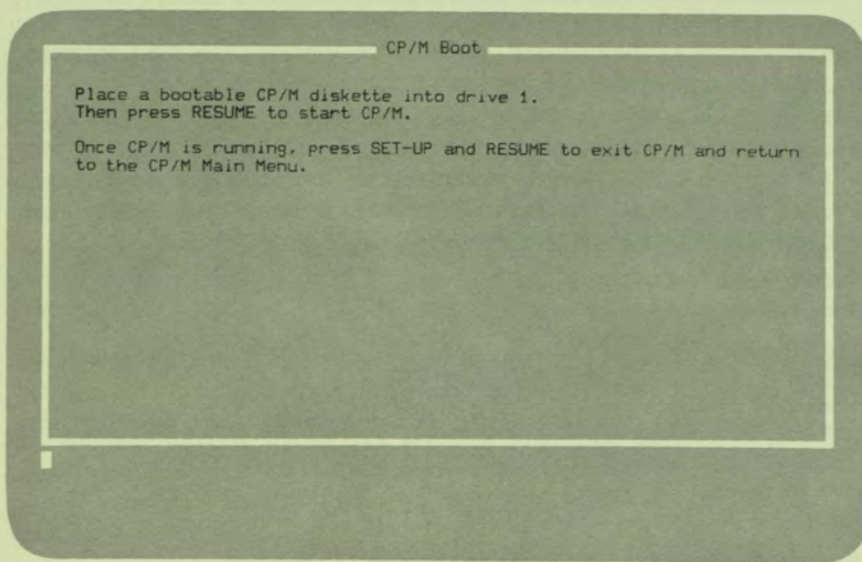


Figure 1-6
CP/M Boot Screen

CHECKING YOUR DIRECTORY

After you have performed the procedure described in the previous section, "Start CP/M," you should check your CP/M directory to make sure that your system diskette contains all the software included with your CP/M application option.

To do this, type **DIR** after the **A>** prompt. Then press the **RETURN** key. The system should display the contents of your system diskette. Check the names displayed against those listed in Figure 1-7. If you discover any omissions, contact your vendor.

```
A>DIR
A: ASM      COM : DDT      COM : DUMP      ASM : DUMP      COM
A: ED       COM : LOAD     COM : PIP       COM : STAT      COM
A: SUBMIT   COM : XSUB      COM : SYSGEN    COM : FILEXFER  COM
A>
```

Figure 1-7
CP/M Directory

ENDING A SESSION WITH CP/M

Follow the procedure below to end a CP/M operating session.

1. At the prompt (in this case **A>**), press the **SET-UP** key. The system displays:

```
CP/M cancelled on user request
Press RESUME to return to CP/M Main Menu
```
2. Press the **RESUME** key. The system displays the CP/M Main Menu.
3. Press **MAIN SCREEN** or **EXIT** to return to the P/OS Main Menu.
4. Remove the system diskette from drive 1 and put it away. Remove and put away any other diskettes that you have been using.

Once you have reached the P/OS Main Menu, you can either use any of the Professional's other capabilities (described in the *Professional 300 Series User's Guide for Hard Disk System*) or you can turn off the system by pressing the power switch to the "0" position.

COPYING DISKETTES

As you use CP/M, you will want to make backup copies of your diskettes and virtual diskettes. In doing so, you will ensure that you have extra copies of your files in case files on your working diskettes or hard disk become lost or damaged.

Before you start to use CP/M, you should copy your application and system diskettes. This will ensure that you have copies that you can use in case you accidentally lose or erase either diskette.

The procedure below tells you how to copy diskettes. You can copy any diskette, **except the system diskette**, by following the first 8 steps of the procedure. To copy the system diskette, follow all 15 steps of the procedure.

This procedure assumes that you have the CP/M Main Menu displayed on your screen. When you do this procedure, you will type some commands so that CP/M can copy the diskette. These commands are discussed in later chapters. For now, however, you don't have to understand how they work.

CAUTION: Press the RETURN key only when directed to do so. Otherwise, you may erase valuable information. You may, from habit, want to press RETURN each time you enter information during this procedure. Several steps in this procedure require you to enter information **without** pressing RETURN. To avoid any errors, do not press RETURN unless the instructions tell you to.

1. Insert the diskette to which you want to copy the system diskette in drive 2 (Drive B) of the the system unit.

NOTE: This procedure erases the files on the diskette in Drive B. You should therefore choose either a blank diskette or one that contains files that you wish to erase.

CAUTION: NEVER INITIALIZE YOUR SYSTEM DISKETTE. If you do so, you will erase all the files on your diskette.

2. From the CP/M Main Menu, select "Initialize CP/M Diskette." Press DO. The CP/M Diskette Initialization Menu is displayed.
3. Move the pointer to Drive 2 (Drive B) and press DO. For a few seconds, you will hear some activity in the diskette drive. Then the CP/M Diskette Initialization Menu is redisplayed with the following message at the bottom of the screen.

Diskette Initialized

4. Press **MAIN SCREEN** or **EXIT** to return to the CP/M Main Menu.
5. Move the pointer to "Start CP/M" and press **DO**.
6. Insert your system diskette into drive 1 (Drive A). (Reminder: the diskette that you just initialized should still be in Drive B.)
7. Press **RESUME**.
8. At the prompt (A>), type:

```
A> PIP B: = A: *.*[OVR]
```

Press **RETURN**. The following message appears:

```
COPYING—
```

The files on the system diskette are then listed on your monitor as they are copied. The copying process takes several minutes. After all files have been copied, the prompt (A>) is redisplayed.

If you are copying your system diskette, go to step 9. Otherwise, stop here. All files from Diskette A have been copied to Diskette B.

9. (**System diskette only.**) At the A> prompt, type:

```
A> SYSGEN
```

Press **RETURN**. The following message is displayed:

```
SYSGEN Version 2.0 for the Professional 300 Series  
Enter source drive name (or press RETURN to skip)
```

10. Type:

```
A
```

DO NOT PRESS RETURN. The following message appears:

```
Source on A, then press RETURN
```

11. Press **RETURN**. The following message is displayed:

```
Function complete  
Enter destination drive name (or press RETURN to reboot)
```

12. Type:

```
B
```

DO NOT PRESS RETURN. The following message appears:

```
Destination on B, then press RETURN
```

13. Press RETURN. The following message appears:

Function complete

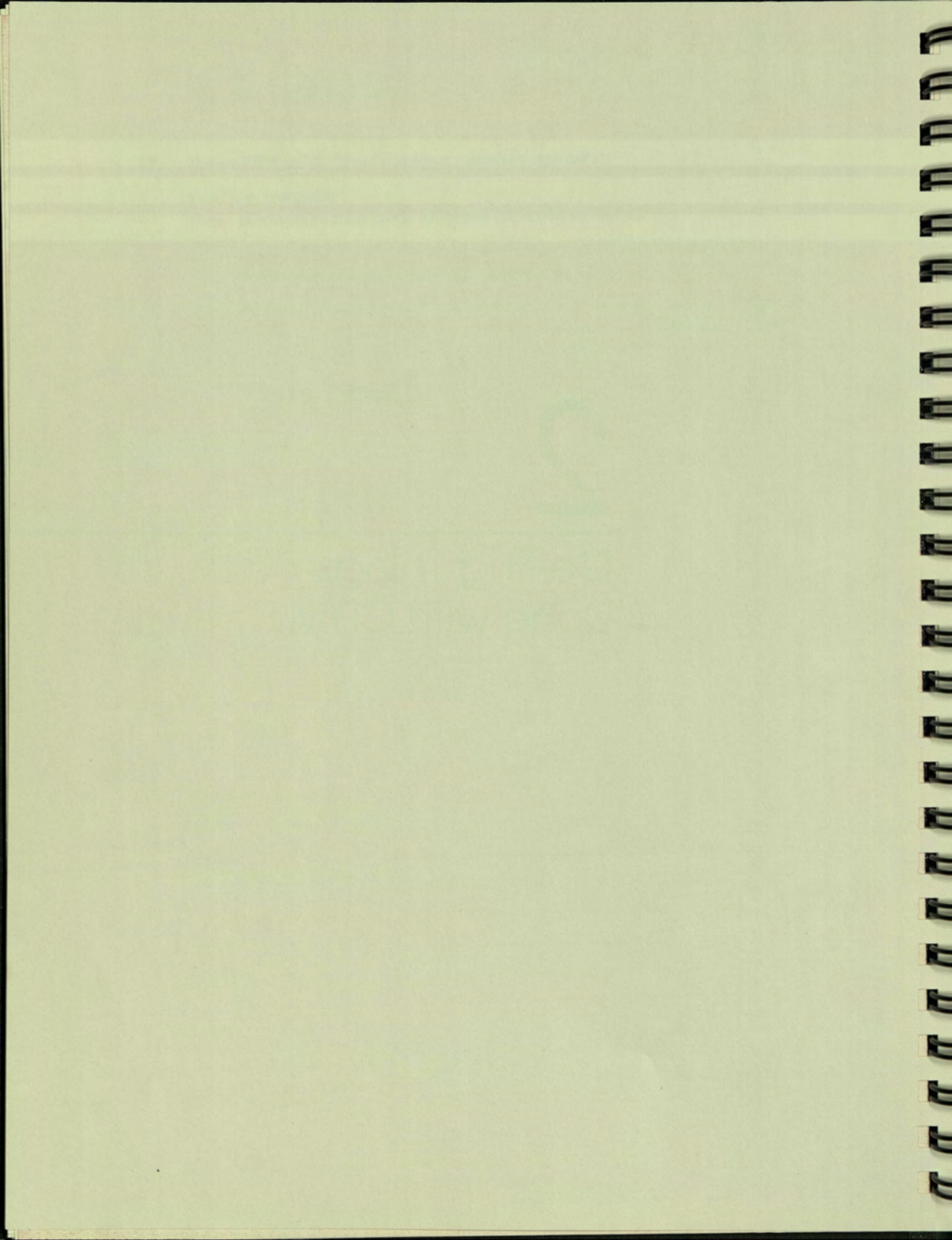
Enter destination drive name (or press RETURN to reboot)

14. Press RETURN. When the prompt (A >) is redisplayed, the system diskette has been copied onto the diskette in Drive B.
15. Remove the diskette from Drive B and label it as your backup system diskette. Because the CP/M files on the diskette are copyrighted, you should also mark the backup CP/M system diskette as follows:

© 1981 Digital Research

2

Getting Things
Done with CP/M



Chapter 2

Getting Things Done with CP/M

The Professional's CP/M is a useful tool for running applications programs. You can also use it to manipulate and store the data that such programs often require for input or produce as output. An applications program helps you perform a particular task, such as planning a budget or analyzing data. Many applications programs are available for use with the Professional and CP/M.

You can best learn CP/M by using it. This chapter explains various CP/M operations and provides practice examples. You can get a good idea of how CP/M works on your Professional by reading the rest of this chapter and trying the examples.

CONVERSING WITH CP/M

You converse with CP/M by typing commands when CP/M displays the prompt (A>). Whenever the A> prompt is displayed on your monitor, you are conversing with CP/M at "command level." In other words, you are talking directly to CP/M when you issue commands at the A> prompt.

Generally, after typing a command, you must press the RETURN key to tell CP/M that you want it to execute the command. Some CP/M programs display messages such as CR (for carriage return), or RET to indicate that you should press the RETURN key.

A few CP/M programs ask you questions requiring a single-character response. Some of these programs process your response as soon as you type it, without waiting for you to press RETURN. In such instances, you must be careful because a typing error could alter or destroy your software.

This guide alerts you to such programs, but your best protection against trouble is to write-protect diskettes where instructed to do so, and to keep copies of your software. You can protect any diskette from being written on by any program if you use the self-sticking write-protect tabs provided by the manufacturer. Apply these tabs as shown in Figure 2-1. This mechanism prevents the inadvertent destruction of data by you or anyone else. You can remove the write-protect tab at any time and return the diskette to its writable state.

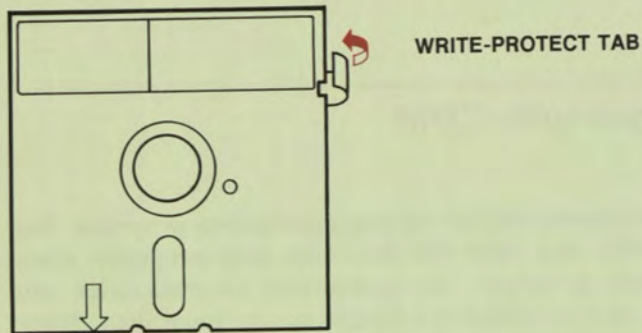


Figure 2-1
Applying a Write-Protect Tab

USING THE KEYBOARD WITH CP/M

As you use your CP/M application, you will discover that the keys on your Professional's keyboard sometimes have different functions than they do when you are working in P/OS. This section summarizes how CP/M interprets the keys as you type them.

Keys to Which CP/M Responds

SET-UP—**SET-UP** is used to end a CP/M operating session. (See "Ending an Operating Session" in Chapter 1.)

SET-UP is also used if the keyboard becomes locked (the **WAIT** light is on) and nothing you type is entered. Pressing **SET-UP** is the only way to clear this condition. When you press **SET-UP**, you also exit from CP/M.

HOLD SCREEN—**HOLD SCREEN** keeps the screen display from changing so that you can read what is on it. Your Professional will not display any new text after you press **HOLD SCREEN** once. When you press **HOLD SCREEN**, the **HOLD** light on the keyboard lights.

You can release the screen in two different ways. First, if you want to see the rest of the file, press **HOLD SCREEN** again. The **HOLD** light goes out, and the remainder of the file is displayed.

Second, if you wish to exit from the file, press any key except for the following:

HOLD SCREEN
 PRINT SCREEN
 SET-UP
 SHIFT
 COMPOSE CHARACTER
 CTRL
 LOCK

The **A >** prompt is redisplayed. (Some keys will produce extra characters to be displayed beside the **A >** prompt. Simply delete the characters and continue processing.)

COMPOSE CHARACTER—This key is used to enter 8-bit characters. (For more information about 7- and 8-bit characters, see the *Professional 300 Series Terminal Subsystem Manual*, order number AA-N623B-TK.)

If you inadvertently press **COMPOSE CHARACTER** when you are entering 7-bit characters, press **<X>** to cancel it. (If you do not press **<X>**, the terminal beeps after the second keystroke. You can then continue typing.)

RETURN, LF (LINE FEED), and ENTER—The **RETURN**, **LF**, and **ENTER** keys do the same thing. Use them to mark the ends of commands that you issue to CP/M, and to indicate the end of a line when you are using **ED**, the text editor.

BS (BACKSPACE) and <X> (DELETE)—The **BS** and **<X>** keys erase errors. When you press either of these keys, the character to the left of the cursor is erased and the cursor moves one space to the left. CP/M lets you erase characters until you reach the **A >** prompt. When you reach the **A >** prompt, however, CP/M ceases to respond to these keys. You cannot erase the prompt or any characters on the lines above the prompt.

When you are working with ED and some other CP/M applications, the $\langle X \rangle$ key works slightly differently. When you press $\langle X \rangle$, the character to the left of the cursor is erased from memory, but CP/M echoes the erased character on the screen. If you erase a series of characters using the $\langle X \rangle$ key, you may find the display on your screen to be confusing. You can redisplay the line with its corrections by typing CTRL/R.

CTRL—The CTRL key is typed simultaneously with a letter key to generate a special form of the letter called a control code. CP/M recognizes the control codes listed below.

This guide documents such combinations as CTRL/X to indicate that you should simultaneously press the CTRL key and some other letter (in this case, X). Such combinations are called control characters. When typed, some control characters are displayed on the screen before causing any action to occur, and some are not. A control character that is displayed appears as a caret (^) followed by the letter you typed. For example, when you type CTRL/C to reset the system, the screen displays it as ^C before resetting the system.

Control Code	Function
CTRL/C	Resets the system. (This is sometimes called a “warm start” or a “warm boot.”)
CTRL/E	Permits you to start a new line while deferring execution of the preceding line.
CTRL/H	Works the same as the BS (BACKSPACE) key.
CTRL/J	Works the same as the LF (LINE FEED) key.
CTRL/M	Works the same as the RETURN and ENTER keys.
CTRL/P	Echoes screen display on the current LST: device until you type CTRL/P a second time or until a warm start occurs. (This is not the same as the PRINT SCREEN key.) Chapter 3 discusses the CP/M devices in greater detail.
CTRL/R	Redisplays the current line with corrections.
CTRL/S	Keeps the screen display from changing so you can read what is on it. To display the remainder of the file, type CTRL/S again, or type any letter or number key on the keyboard (do not use the auxiliary keypad to type a number).

CTRL/U	Erases the current line from CP/M's memory, displays a # sign, moves the cursor to the next line, and waits for further input. The line image stays on the screen.
CTRL/X	Works the same as CTRL/U but erases the line image from the screen.
CTRL/Z	Ends input from the keyboard when using ED. Indicates the end of a string when using PIP.

Keys That CP/M Displays but Does Not Understand

CP/M does not understand the following keys:

1. Any keys in the editing keypad
2. The four keys at the top of the auxiliary keypad (PF1-PF4)
3. The special function keys in the top row, except for HOLD SCREEN, SET-UP, ESC, BS, LF, and PRINT SCREEN (if your Professional communicates with a local printer).

If you press any of these keys, CP/M displays some characters but takes no further action. Although CP/M itself does not understand these keys, some applications programs do. The way an applications program responds to these keys depends on the program.

The keys in the editing keypad display the following characters:

Key	Display	
	7-bit mode	8-bit mode
FIND	^[1~	CSI1~
INSERT HERE	^[2~	CSI2~
REMOVE	^[3~	CSI3~
SELECT	^[4~	CSI4~
PREV SCREEN	^[5~	CSI5~
NEXT SCREEN	^[6~	CSI6~
↑	^[A	CSIA
↓	^[B	CSIB
→	^[C	CSIC
←	^[D	CSID

The four keys at the top of the auxiliary keypad display the following characters:

Key	Display	
	7-bit mode	8-bit mode
PF1	^[OP	SS3P
PF2	^[OQ	SS3Q
PF3	^[OR	SS3R
PF4	^[OS	SS3S

The special function keys in the top row display the following characters:

Key	Display	
	7-bit mode	8-bit mode
HOLD SCREEN	Pressed once, holds screen display; pressed twice, releases the hold	Same as 7-bit mode
PRINT SCREEN	Prints the screen; invisible if you have no printer	Same as 7-bit mode
SET-UP	"CP/M cancelled on user request; Press RESUME to return to CP/M Main menu"	Same as 7-bit mode
BREAK	^[13~	CSI13~
F5	^[15~	CSI15~
INTERRUPT	^[17~	CSI17~
RESUME	^[18~	CSI18~
CANCEL	^[19~	CSI19~
MAIN SCREEN	^[20~	CSI20~
EXIT	^[21~	CSI21~
ESC	Understood as ESC	Same as 7-bit mode
BS	Understood as BS	Same as 7-bit mode
LF	Understood as LF	Same as 7-bit mode
ADDTNL OPTIONS	^[26~	CSI26~
HELP	^[28~	CSI28~
DO	^[29~	CSI29~
F17	^[31~	CSI31~
F18	^[32~	CSI32~
F19	^[33~	CSI33~
F20	^[34~	CSI34~

CORRECTING TYPING ERRORS IN COMMAND LINES

If you mistype a command or type the wrong command, and you notice the error before pressing RETURN, you can make corrections in several ways. The following sections describe the various methods of making corrections.

BS

The BS key in the top row of function keys is the BACKSPACE key. Pressing the BS key erases the last character you typed from CP/M's memory and from the screen, and moves the cursor back a space. (The cursor is the blinking symbol on your screen. It will be either a rectangle (█) or an underscore (_)). For example, type:




```
A>PROFESSIONSK
```



Now press the BS key twice. The characters K and S disappear, leaving you with PROFESSION. Type:

```
A>PROFESSIONAL
```

The screen displays PROFESSIONAL. Now press the BS key 12 times to erase PROFESSIONAL.

DELETE Key

The DELETE key is the  key located above the RETURN key. If you are at command level, you erase the character to the left of the cursor when you press the  key. (At command level, the  key works the same way that BS works.)

NOTE: The  key does not work this way when you use some applications, such as the editor (ED). If you press the  key while using these applications, the character to the left of the cursor is deleted from memory, but the deleted character is echoed at your screen. You might find it easier to use the BS key when editing.

CTRL/X

CTRL/X erases the entire command line back to the prompt. For example, type:

```
A>PERSONAL OFFICE COMPUTER
```

Now type CTRL/X by holding down the CTRL key and then pressing the letter X. The entire line is erased, and only the prompt remains.

CTRL/U

CTRL/U cancels the current command line and allows you to begin again on the line below. The line remains visible on the screen, but is deleted from memory. To see how CTRL/U works, type:

```
A>THIS IS FUN
```

Now type CTRL/U by holding down the CTRL key and pressing the letter U. The system displays a # sign at the end of the line. The cursor then moves to the next line, but the A> is not redisplayed:

```
A>THIS IS FUN#

```

You erased the words “THIS IS FUN” from memory when you typed CTRL/U, but you did not erase them from the screen. When the cursor moves to the next line, you can enter a CP/M command just as you would if the cursor were beside the prompt.

ERROR MESSAGES

When CP/M displays its prompt, it is waiting for you to type a command. If you type something that CP/M does not understand, it displays an error message. For example, if you type:

```
A>TIPE <RETURN>
```

when you mean to issue the TYPE command, CP/M displays the message:

```
TIPE?
```

The error message signifies that CP/M does not recognize the command.

Error messages can occur for a variety of reasons. For example, you get an error message if you type a command incorrectly, if the command is not a valid CP/M command, or if CP/M lacks some information it needs to process the command. It is not possible to anticipate all the conditions that can cause error messages. However, if you get an error message you can:

- Check for spelling errors and then retype the command correctly if you find any errors.
- Check the list of commands at the beginning of Chapter 3 to determine whether or not the command you typed is a valid CP/M command.
- Refer to the discussion of the command in Chapter 3.

ERROR CONDITIONS

Sometimes the system issues an error message, but it does not redisplay the A> prompt. If this happens, you can take one of two actions:

- Type CTRL/C to reset the system. When you type CTRL/C, the A> prompt is redisplayed. This is called a “warm start” or “warm boot.”
- Press SET-UP and then RESUME to exit to the CP/M Main Menu.

DISKETTES AND FILES

CP/M handles a broad spectrum of information including programs, text, and data. Information is organized in the form of files, and the files are stored on diskettes and virtual diskettes. You might find it helpful to think of the information contained on a diskette or virtual diskette as being similar to the folders contained in a file cabinet.

Storing Information on Diskettes

The computer stores and retrieves files by referring to tracks and sectors on a diskette (see Figure 2-2). Professional diskettes have 80 tracks (numbered 0 through 79), each composed of 10 sectors. Sectors store blocks of bytes, each byte representing one character such as a letter, a digit, or a symbol. Because each sector has a unique location on the diskette, the computer can find a particular sector on a particular track and store information in it or retrieve information from it.

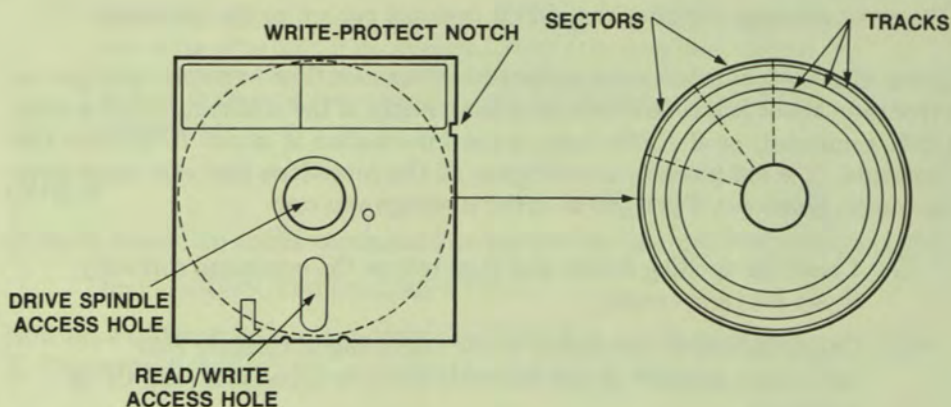


Figure 2-2
Tracks and Sectors on a Diskette

Storing Information on Virtual Diskettes

CP/M stores information on virtual diskettes in much the same way as it stores information on RX-50 diskettes. The space available on a virtual diskette is substantially larger than the space available on RX-50 diskettes. (Virtual diskettes can be as large as approximately 1 megabyte.) In addition, you can use as many as four virtual diskettes at once, whereas you can use only two RX-50 diskettes at once.

Using Diskettes and Virtual Diskettes

Because information on a diskette or a virtual diskette can be corrupted by accident, human error, or temperature and humidity, it is important to copy or "back up" diskettes and virtual diskettes. Thus, you generate working copies and store the originals, or master diskettes, as insurance against corrupted files.

Chapter 1 describes how to back up diskettes. If you have not already done so, copy and label the system diskette according to the procedure in Chapter 1. Other diskettes and virtual diskettes that contain programs, the data generated by programs, or text should be copied periodically.

You may occasionally install a diskette that does not seat itself properly on the drive hub as you close the drive door. This can cause unreliable operation resulting in data loss or inaccessible data. To guard against this, always insert the diskette carefully, pushing it gently back in the drive as far as it can go. Shut the drive door securely. If you encounter any difficulty inserting the diskette or closing the drive door, remove the diskette and try again.

Using Commands and Files

When you issue a command, you usually have to include a file specification. The file specification tells CP/M the file to which the command applies and the file's location (that is, the drive that contains the diskette on which the file resides). If you do not specify a drive name, CP/M assumes the file is on the diskette in the current drive.

File Names

CP/M files have a “first” name, called the file name, and an optional “last” name, called the file type (or extension). The file name and file type are separated by a period. The following conventions apply to file names:

- A file name and type can include any combination of letters, numbers, and printable symbols except for any characters generated by the COMPOSE CHARACTER key and the characters shown below:

< > . , ; : = * ? [] |

- The file name must be separated from the file type by a period.
- A file name can include from one to eight characters.
- A file type can include from zero to three characters. File types are often used to denote a class of files. For example, the file type .COM identifies CP/M programs that you can run by typing the file name.

Following are some examples of valid file names:

XYZ.COM	X.Y	1
GAMMA.RAY	1.1	GAMMA
GAMMA.1	X	PAY-ROLL.+

Looking at the File Directory—The DIR Command

Each diskette that contains files also contains a file directory. Just as a telephone directory identifies what subscribers exist and where they can be found, the diskette directory tells CP/M what files exist on a given diskette. To obtain information about the contents of a diskette, examine its directory by using the DIR command. To see what is on a diskette, type:

A>DIR <RETURN>

The Professional displays all the files on the diskette:

```
A>DIR
A: ASM      COM : DDT      COM : DUMP      ASM : DUMP      COM
A: ED       COM : LOAD     COM : PIP       COM : STAT      COM
A: SUBMIT   COM : XSUB      COM : SYSGEN    COM : FILEXFER  COM
A>
```

The files are displayed without a period between the file name and file type, but you must always include the period when creating and referring to files. Note that files on each line in the directory are separated from one another by a colon (:).

The DIR command lists the contents of the currently active diskette. To examine the contents of another diskette, specify its drive name. For example, to examine the contents of a diskette on Drive B, first make sure there is an initialized diskette in the drive, then type:

```
A>DIR B: <RETURN>
```

CP/M lists the contents of the diskette on Drive B.

If there are no files on the initialized diskette when you issue the DIR command, CP/M replies:

```
NO FILE
```

If no diskette is loaded in the drive that you specify, or the drive door is open, the system issues the following message:

```
A>DIR B:
? I/O error attempting to mount disk foreign, I/O status code = 375
  Device not ready
Bdos Err On B: Select █
```

If this happens, type CTRL/C to redisplay the A> prompt. Then insert an initialized diskette in the drive, close the door, and retype the command. CP/M will then display the directory for the selected drive.

You can also use the DIR command to obtain a directory of the virtual diskettes that you have configured for the current operating session. Type DIR E: for a directory of the virtual diskette on Drive E, DIR F: for a directory of the virtual diskette on Drive F, and so on.

If you have not configured the virtual diskette, the system issues the message:

```
A>DIR H:
? No file name for virtual diskette
Bdos Err On H: Select █
```

If this happens, type CTRL/C to redisplay the A> prompt and continue processing.

Changing the Currently Active Drive

Unless you type a drive name when you issue a command, CP/M assumes that the files upon which you want the command to act reside on the diskette in the currently active drive. The current drive is sometimes called the default drive, and commands are said to default to the current drive.

There are two ways of making commands apply to other drives:

1. Specify the drive name with the command, as you did when you examined the directory on Drive B. This method is useful when you want to do something briefly on another drive and then continue working on the current one.
2. Specify another drive name as the currently active one before you issue the command. (See the example below.) This method is more convenient if you want to work on the other drive for a length of time, perhaps to run a program and store its results on the diskette on that drive.

Any time the CP/M prompt is displayed, you can access a different drive by typing the drive name (A, B, E, F, G, or H) followed by a colon (:). **If there is an initialized diskette in that drive**, then that drive becomes active, and CP/M displays the appropriate prompt.

To access Drive B before issuing the DIR command, **be sure there is an initialized diskette** in Drive B. Type:

```
A>B: <RETURN>
```

CP/M displays:

```
B>
```

Drive B becomes the current drive. If you now issue a command without specifying a location, the command defaults to Drive B. For instance, type:

```
B>DIR <RETURN>
```

to get a directory of the diskette on Drive B. If the diskette in Drive B contains some files, then a directory of those files is displayed on your screen.

Now get a directory of the system diskette on Drive A. Type:

```
B>DIR A: <RETURN>
```

The directory of the CP/M diskette on Drive A appears.

You can use DIR to determine whether a specific file resides on any given diskette. For example, to find the FILEXFER.COM file, which is on the diskette in Drive A, specify the location and the name of the file. Type:

```
B>DIR A:FILEXFER.COM
```


CP/M displays:

```
A: FILEXFER COM
```

Switch back to Drive A by typing:

```
B>A: <RETURN>
```

CP/M again accesses Drive A and displays its prompt:

```
A>
```

Using File References

A file reference identifies a particular file or group of files on a diskette. CP/M recognizes references to two kinds of file names, unambiguous and ambiguous. An unambiguous file name identifies a specific file such as FILEXFER.COM or PIP.COM. An ambiguous file name identifies one or more files. Ambiguous file names are useful when, for example, you are searching for a file whose exact name you have forgotten, or when you want a command to act on several files at once.

You use an ambiguous reference by substituting a question mark (?) or an asterisk (*) for parts of an unambiguous file name. Because they can replace actual characters, these symbols are sometimes called wildcards. The question mark replaces individual characters in the same position as the question mark. For example, give the DIR command followed by an S, seven question marks, and the file type .COM. Type:

```
A>DIR S???????.COM <RETURN>
```

CP/M lists all files on the diskette in Drive A that start with S and have a file type of .COM.

The asterisk matches any unspecified remainder of a file name or file type. For this reason, typing:

```
A>DIR *.COM <RETURN>
```

causes CP/M to list all files (on Drive A, the current drive) with a file type of .COM.

If you type:

```
A>DIR *.* <RETURN>
```

you get a list of all files on the diskette in Drive A.

This is the same as typing:

A> DIR <RETURN>

Note that while a file name cannot contain a question mark (?) or an asterisk (*), a file reference can.

THE CP/M COMMANDS

CP/M recognizes two kinds of commands, resident and transient. Resident commands are built into CP/M. You will not see them in the directory, but the programs for them are part of the CP/M operating system. Any time the system is running, these commands are in memory, and you can use them. The resident commands are:

DIR	Displays a directory.
ERA	Deletes a file or files from a diskette.
REN	Renames a file.
SAVE	Saves portions of memory in a diskette file (not covered in this manual).
TYPE	Displays the contents of a file.
USER	Creates separate diskette areas for multiple users (not covered in this manual).

Transient commands are not usually in memory when CP/M is running. Because the programs for them are longer than those for the resident commands, transient commands exist as files on a diskette and must be brought into the computer's memory before they can be used. The files for transient commands have the file type .COM. If such a file is on the diskette in the current drive, you can invoke it by typing the file name without the file type. (For example, you invoke the program PIP.COM by typing PIP.) Otherwise, you must find which diskette it is on, load the diskette in a drive, and specify that drive when you type the file name. Alternatively, you can copy the file to the current diskette if space permits. Examples of transient commands that you will use most frequently are:

ED	Allows you to create and modify text files.
PIP	Allows you to copy files.
STAT	Provides information about files and diskettes.
FILEXFER	Allows you to copy files between P/OS and CP/M.

The remainder of this chapter introduces a variety of the CP/M commands in the form of examples that you can practice at your terminal. However, the chapter does not provide extensive information about the commands, nor does it cover every one. For detailed information on each CP/M command, refer to Chapter 3.

Using CP/M Commands

Saving information in electronic files requires some understanding about how they are created, stored, and used. It is equally important to know what operations can be performed on files to make the most efficient use of diskette space and to aid in the transfer of information between diskettes.

To see how various commands affect files, create a test file on the system diskette. Make sure that the system diskette is loaded in Drive A. If the system diskette is write-protected, remove it and remove the write-protect tab. Then insert the diskette in Drive A. (If you have not copied the system diskette yet, do so according to the instructions in Chapter 1, before you try any of the following procedures.)

Creating and Editing Files—The ED Command

Aside from running programs, you may want to write reports, memos, or letters and store them in files. Files of this sort are called text or ASCII files because they consist of ordinary characters: letters, numbers, and symbols such as punctuation marks. However, not all files are text files. Some files exist in forms understandable only to the computer. These have various names such as machine language, binary, or executable files. All of the programs for CP/M's transient commands are stored in this form.

To create a file, invoke ED, the CP/M editor. An editor is a program for creating new text files or for modifying old ones. The following example demonstrates elementary editing commands. First, the example creates a new text file, which contains intentional errors. Next, the example shows you how to move around in the text and fix the errors.

Create the file TEST.TXT by typing:

```
A>ED TEST.TXT
```

When you type the command, ED allocates an area in memory called the text buffer, which serves as a temporary workspace (see Figure 2-3). At the end of the editing session, ED transfers the file from the text buffer to the diskette and frees the area in memory. If no other file with the name TEST.TXT exists on the diskette, ED informs you that you have made a

NEW FILE

and creates the file TEST.TXT on the diskette, along with a temporary file called TEST.###. Both are empty. ED also prompts you with a colon (:), a space, and an asterisk (*), signifying it is ready to accept commands:

: *

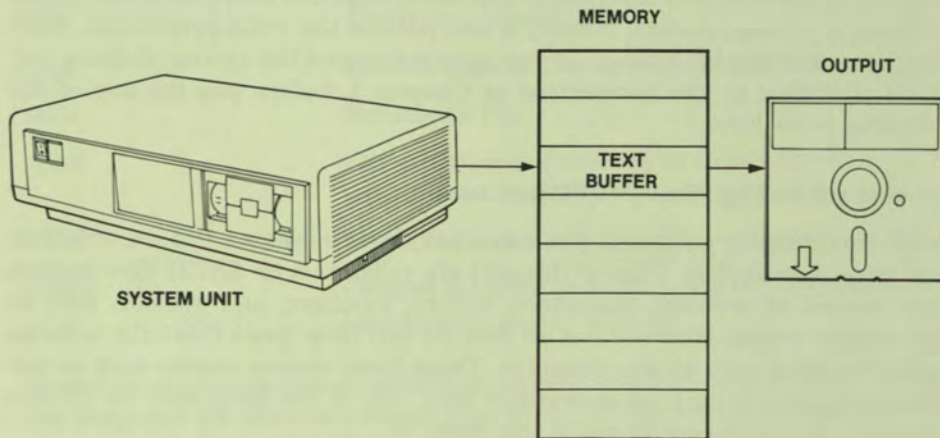


Figure 2-3
The ED Text Buffer

Tell ED you want to insert text by typing:

: *I <RETURN>

ED gives you a blank line, numbered 1, and accepts all subsequent input as new text until you type CTRL/Z. Each time you press RETURN, ED gives you another numbered line. The line numbers help you locate information when you want to make changes, but they do not exist in the file when the editing session is over:

1:

Insert the text printed in red in the following example, typing everything exactly as you see it. The text contains intentional errors for you to fix later. If you make additional errors, correct them using **BS** or **CTRL/X** if you can, or just leave them for the time being.

```
A>ED TEST.TXT
NEW FILE
: *I
1: THIS REPORT OUTLINES COSTS FOR
2: LABORATORY MICEE AND EQUIPMENT USED
3: EXPERIMENTALLY FOR
4: IN EXPERIMENTS DURING JANUARY.
5: █
```

Typing **CTRL/Z** signals ED that you are through inserting text and causes ED to display its prompt (: *), which means it is waiting for further commands. Now you can correct errors and insert additional text. To help you move around in the buffer and locate portions of the text that require change, ED provides a character pointer, called the CP. This is an imaginary arrow that marks where you are in the buffer. The CP will always be located:

- Before the first character in the buffer, or
- After the last character in the buffer, or
- Between two characters.

When you finish inserting text, the CP is at the bottom of the text buffer; that is, after the last character in the last line. It may help to think of the text buffer as a page in a book. Move the CP to the beginning of the buffer (page) by pressing:

```
: *B <RETURN>
```

ED displays:

```
1: *
```

to indicate that the CP is now at the beginning of the buffer. From anywhere in the buffer, you can always reach the beginning by pressing B, or the end by typing -B.

If you move to the end of the buffer, ED displays:

```
: *
```

to signal that it is at the end and is awaiting further commands.

Use the T (Type) command to display lines from the buffer. Pressing T causes ED to display the current line starting from wherever the CP is currently located. Typing 0T (use a **zero** not the letter O) causes ED to display the line from the beginning to the CP. You can easily display the whole line by typing 0TT. If you want ED to display the whole buffer, move the CP to the beginning of the buffer, then type:

```
1: *#T <RETURN>
```

ED displays the entire contents of the buffer.

Although ED just displayed the whole buffer, the CP is still at the top of the buffer (that is, before the first character in line 1). Line 2 in the file contains a misspelling. MICEE should be MICE. Use the S command to make ED search for and correct the misspelled word. Type:

```
1: *SMICEE<CTRL/Z>MICE<CTRL/Z> <RETURN>
```

ED moves the CP to the word MICEE and replaces it with the word MICE. Now examine the line to confirm that ED made the change. Remember that the CP is located in the middle of the line, after the word MICE. To display the whole line, use the 0TT command. Type:

```
2: *0TT <RETURN>
```

The corrected line is displayed.

Sometimes it is necessary to delete entire lines from the file. Line 3, for example, is extraneous and should be removed. The L command tells ED how many lines in the buffer to move the CP. Pressing L moves it one line ahead; typing 2L moves it two lines ahead, typing -2L moves it two lines back, and so on. Because the CP is positioned in the middle of line 2, you should move it to the beginning of line 3 and type it out to confirm it is really the line you want to delete. To do this, type:

```
2: *LT <RETURN>
```

ED moves the CP ahead one line and displays:

```
3: EXPERIMENTALLY FOR
3: *
```

indicating that the CP is positioned at the beginning of line 3.

Note that the T command does not change the position of the CP. Use the K (Kill) command to delete the line. Press:

```
3: *K <RETURN>
```


ED displays:

3: *

To confirm that the line was deleted, use the B command to move the CP to the beginning of the buffer. Then use the #T command to display the entire contents of the buffer. Press:

3: *B <RETURN>

1: *#T <RETURN>

ED displays the buffer. The extraneous line has been deleted, and all the lines have been renumbered.

The entire editing sequence is shown below.

```
A>ED TEST.TXT
NEW FILE
: *I
1: THIS REPORT OUTLINES COSTS FOR
2: LABORATORY MICEE AND EQUIPMENT USED
3: EXPERIMENTALLY FOR
4: IN EXPERIMENTS DURING JANUARY.
5:
: *B
1: *#T
1: THIS REPORT OUTLINES COSTS FOR
2: LABORATORY MICEE AND EQUIPMENT USED
3: EXPERIMENTALLY FOR
4: IN EXPERIMENTS DURING JANUARY.
1: *SMICEE^ZMICE^Z
2: *OTT
2: LABORATORY MICE AND EQUIPMENT USED
2: *LT
3: EXPERIMENTALLY FOR
3: *K
3: *B
1: *#T
1: THIS REPORT OUTLINES COSTS FOR
2: LABORATORY MICE AND EQUIPMENT USED
3: IN EXPERIMENTS DURING JANUARY.
1: *█
```

End the editing session by using the E (Exit) command. This tells ED you are finished and want to save the file on diskette. Press:

1: *E <RETURN>

When you exit the ED program, CP/M displays its prompt. Type:

A>DIR <RETURN>

Look at the directory to confirm that the files TEST.TXT and TEST.BAK are there. TEST.BAK is a **backup copy** of the file, containing the previous version. Because you just created a new file, there is no previous version, and TEST.BAK is empty. However, when you make future edits, the previous version becomes TEST.BAK and the new version becomes TEST.TXT. This feature ensures that you always have the last version of the file as well as the next-to-the-last version. (Refer to the ED command in Chapter 3 for more information about backup files.)

Examining Files—The TYPE Command

Text files (files composed of printable characters) can be displayed on the screen by the TYPE command. Use the command to examine TEST.TXT and TEST.BAK:

```
A>TYPE TEST.TXT <RETURN>
```

CP/M displays the contents of TEST.TXT on the terminal as shown below. Notice that the line numbers are gone.

```
A>TYPE TEST.TXT
THIS REPORT OUTLINES COSTS FOR
LABORATORY MICE AND EQUIPMENT USED
IN EXPERIMENTS DURING JANUARY.
```

```
A> █
```

When you use TYPE to display a long file, you may find that it scrolls too quickly for you to read it. In that case, you can temporarily halt the scrolling by pressing the HOLD SCREEN key. To resume the display, press the HOLD SCREEN key again.

Now type the contents of TEST.BAK:

```
A>TYPE TEST.BAK <RETURN>
```

Because there is nothing in the file, there is nothing to display. The prompt is displayed and CP/M waits for another command.

Changing File Names—The REN Command

It is a good idea, when creating a file, to give it a name that will help you remember what is in the file. TEST.TXT is very general and gives no clue as to the file's contents. Because the report is supposed to enumerate operating expenses for the month of January, it might be better to call it something like COSTS.JAN.

The REN command lets you change the name of a file. To change the name of TEST.TXT to COSTS.JAN, type:

```
A>REN COSTS.JAN=TEST.TXT <RETURN>
```

Now use the DIR command to confirm that the change was made. (Note that TEST.TXT is no longer in the directory, but TEST.BAK is.)

If you change the name of a file, be sure to rename the backup file as well. In this case, change TEST.BAK to COSTS.BAK. Type:

```
A>REN COSTS.BAK=TEST.BAK <RETURN>
```

You can issue another DIR command, if you want, to verify that the backup file was also renamed.

```
A>REN COSTS.JAN=TEST.TXT
A>DIR
A: TEST      BAK : ASM      COM : DDT      COM : DUMP     ASM
A: DUMP      COM : ED       COM : LOAD     COM : PIP      COM
A: STAT      COM : SUBMIT   COM : XSUB     COM : COSTS    JAN
A: SYSGEN    COM : FILEXFER COM
A>REN COSTS.BAK=TEST.BAK
A>DIR
A: COSTS     BAK : ASM      COM : DDT      COM : DUMP     ASM
A: DUMP      COM : ED       COM : LOAD     COM : PIP      COM
A: STAT      COM : SUBMIT   COM : XSUB     COM : COSTS    JAN
A: SYSGEN    COM : FILEXFER COM
A>
```

Getting Information About Files—The STAT Command

As its name suggests, the STAT command provides statistics on files, such as the amount of space (in bytes) they occupy on a diskette. STAT can also tell you how much free space (unused bytes) is on a particular diskette. Get some information about the current diskette by typing:

```
A>STAT <RETURN>
```

STAT should return a message similar to the one shown below. The message indicates that the diskette is read/write; that is, the diskette can be both read from and written to. The message also indicates that some number of bytes (318K in this example) of unused space remains on the diskette. You will recall from the discussion of storing information on diskettes that a byte is the amount of memory required to store a single character (1K byte equals 1024 bytes).

```
A>STAT
A: R/W, Space: 318k
A>
```

Now get some information about COSTS.JAN. Type:

```
A> STAT COSTS.JAN <RETURN>
```

Your screen will show information similar to that shown below. (This is only an example. You may get different numbers, but the form of the message should be the same.)

```
A>STAT COSTS.JAN
  Recs  Bytes  Ext Acc
    1    2k    1 R/W A:COSTS.JAN
Bytes Remaining On A: 318k
A> █
```

Rec means *record*. CP/M builds files by stringing records together, each of which contains 128 bytes. *EXT* means *extent*. An extent is a block of records. The term *Acc* refers to the file's *access attribute*, which indicates whether the file is a read-only file or a read-write file. In this case, the file can be read from and written to. STAT also informs you that the file resides on Drive A, and that Drive A has 318K bytes of free space.

Copying Files—The PIP Command

The PIP (Peripheral Interchange Program) command enables you to create copies of a file, either on the current diskette or on another diskette. Such copying is useful if you want to change a file without altering the original, or if you want to make a backup copy of the file on another diskette. Backup copies provide protection in case of accidental damage to the original diskette, or in case you accidentally delete the original file. PIP also enables you to send files to printers and other devices connected to the Professional.

To see how PIP works, make a copy of COSTS.JAN on a diskette in Drive B. Load an initialized diskette in Drive B. Type:

```
A> PIP B:COSTS.JAN = A:COSTS.JAN <RETURN>
```

This command copies COSTS.JAN on Drive A to COSTS.JAN on Drive B.

When you type the PIP command, you must include the drive name whenever it differs from the currently active drive. In the example above, Drive A was the currently active drive. You were copying a file to a diskette in Drive B, so you had to specify Drive B in the command (B:COSTS.JAN). Because the file you were copying already existed on Drive A, however, you could have issued the same command as follows:

```
A> PIP B:COSTS.JAN = COSTS.JAN <RETURN>
```


Now examine the directory of Drive B to verify that the copy was made. Type:

```
A> DIR B: <RETURN>
```

COSTS.JAN should appear in the directory. Also examine the directory of Drive A. COSTS.JAN is there, too.

```
A>DIR B:
B: COSTS      JAN
A>DIR A:
A: COSTS      BAK : ASM      COM : DDT      COM : DUMP      ASM
A: DUMP       COM : ED      COM : LOAD     COM : PIP      COM
A: STAT       COM : SUBMIT   COM : XSUB     COM : COSTS    JAN
A: SYSGEN     COM : FILEXFER COM
```

When you copy a file, you can, if you want, give the copy a different name from the original. For instance, to make a copy of COSTS.JAN and call the copy REPORT.JAN, type the following command:

```
A> PIP REPORT.JAN = COSTS.JAN <RETURN>
```

Examining the directory will show both files. The files are identical in content but different in name.

```
A>PIP REPORT.JAN=COSTS.JAN
A>DIR
A: COSTS      BAK : ASM      COM : DDT      COM : DUMP      ASM
A: DUMP       COM : ED      COM : LOAD     COM : PIP      COM
A: STAT       COM : SUBMIT   COM : XSUB     COM : COSTS    JAN
A: REPORT     JAN : SYSGEN   COM : FILEXFER COM
```

CAUTION: If you copy a file and give it the name of a file that already exists, the existing file is deleted and replaced with the new one as soon as the copy operation is complete.

Deleting Files—The ERA Command

Eventually, space on a diskette becomes scarce as you continue to create text files or to generate data by running programs. Therefore, it becomes necessary from time to time to free space by deleting files that are no longer needed. The ERA command erases those files which you no longer need. For example, to delete REPORT.JAN, type:

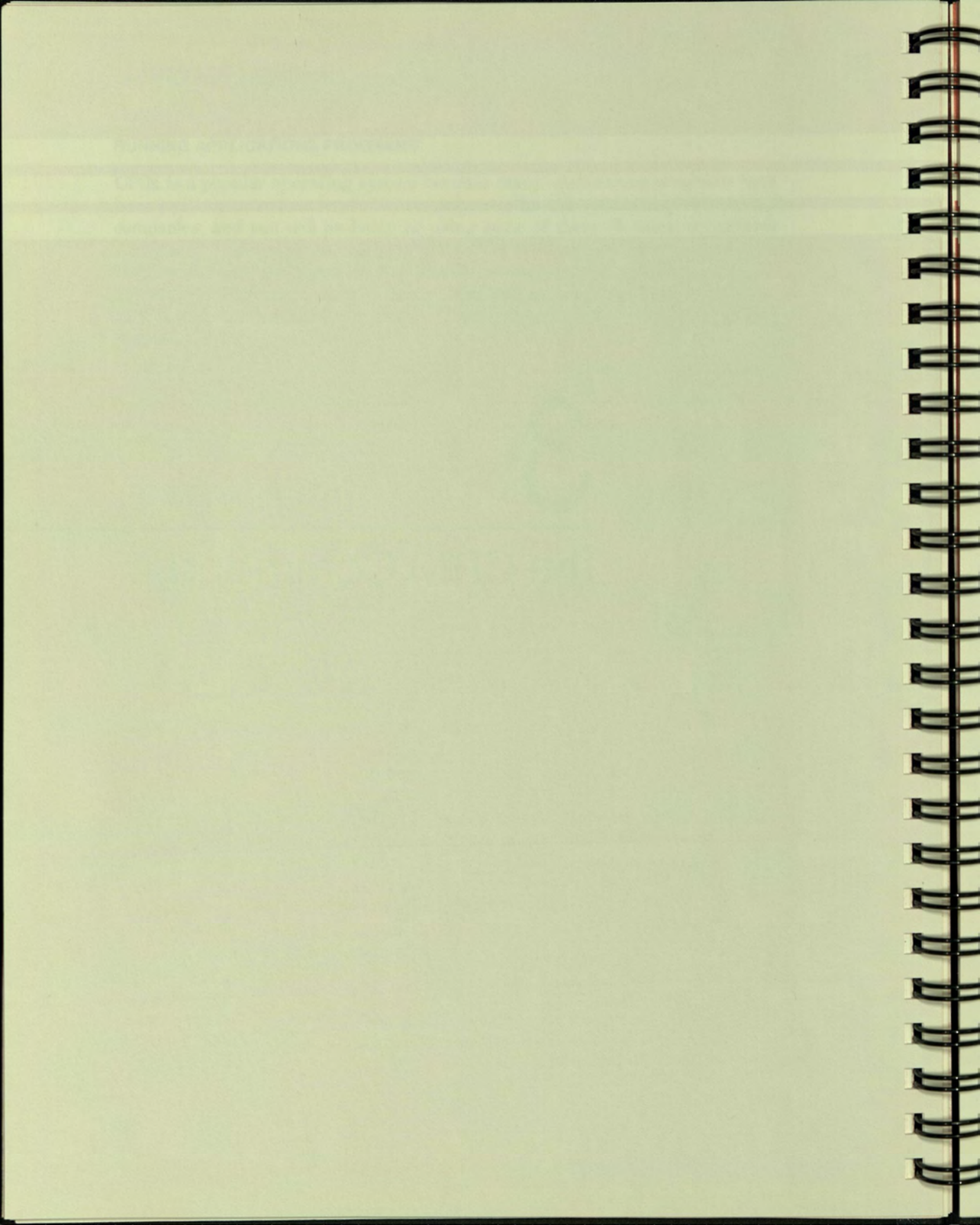
```
A> ERA REPORT.JAN <RETURN>
```

RUNNING APPLICATIONS PROGRAMS

CP/M is a popular operating system because many applications programs have been written to run under it. These programs have been developed by many companies, and you will probably be using some of them. Because procedures for running such programs can vary, you will have to consult the documentation that comes with the programs for detailed operating instructions. In general, however, CP/M makes it easy to run an applications program. Usually an applications program has the file type .COM, and you can invoke it simply by typing its name.

3

The CP/M Commands



Chapter 3

The CP/M Commands

INTRODUCTION

This chapter describes two types of CP/M commands: resident and transient.

- Resident commands are built into the CP/M operating system. You will not see them in the directory, but any time the system is running, these commands are in memory, and you can use them.
- Transient commands are not usually in memory when CP/M is running. Because the programs that execute them are longer than those for resident commands, transient commands exist as files on a diskette and must be brought into the computer's memory before they can be used. Transient commands reside on the system diskette and are visible in the directory as files of the type .COM. Each time you invoke a transient command, CP/M reads the named file from the diskette into memory and then proceeds to execute the computer instructions contained in the file. Transient commands need not be permanent parts of the operating system. You can copy the transient command files you want to use to other diskettes. Thus, you can build working diskettes using only those transient commands you expect to use frequently—and release diskette space for programs and data. (The PIP command, discussed in this chapter, enables you to copy files from one diskette to another.)

Both resident and transient commands are issued by typing the command name in uppercase or lowercase or even in a mixture of the two. (This guide illustrates

all commands in uppercase, but you need not follow that practice.) CP/M requires that you terminate each command by pressing the RETURN key. Here is an example:

```
A>FILEXFER <RETURN>
```

Certain CP/M transient commands contained on your system diskette are associated with assembly language programming and debugging—an area of the Professional CP/M application intentionally omitted from this guide.

The resident and transient CP/M commands provided by your CP/M system diskette are listed below. Transient command names are preceded by an asterisk (*) to distinguish them from names of resident commands.

<i>Command</i>	<i>Function</i>
*ASM	Runs ASM.COM, an assembly program (not covered in this guide).
*DDT	Runs DDT.COM, a CP/M program-debugging utility program (not covered in this guide).
DIR	Lists file names in the directory.
*DUMP	Runs DUMP.COM, CP/M's file-examining utility program (not covered in this guide).
*ED	Runs ED.COM, the CP/M text-editing program.
ERA	Erases or deletes a file.
*FILEXFER	Runs FILEXFER.COM, a program that transfers files between P/OS and CP/M.
*LOAD	Runs LOAD.COM, a program that converts files from HEX format into executable .COM files (not covered in this guide).
*PIP	Runs PIP.COM, the Peripheral Interchange Program (for transporting files from one system device to another).
REN	Renames a file.
SAVE	Permits user to save portions of memory in a diskette file (not covered in this guide).

*STAT	Runs STAT.COM, a CP/M program that provides information about and certain controls over diskette, terminal, and printer allocations.
*SUBMIT	Runs SUBMIT.COM, which permits the execution of multiple, previously generated commands.
*SYSGEN	Runs SYSGEN.COM, which facilitates copying new systems (not covered in this guide).
TYPE	Displays the contents of a text file on the terminal.
USER	Permits creation and use of separate diskette areas for multiple users (not covered in this guide).
*XSUB	Runs XSUB.COM, a subset of SUBMIT.COM. Permits line input to programs that are invoked by SUBMIT (not covered in this guide).

Detailed information about the commands covered in this guide follows.

DIR

The DIR (directory) command displays the names of files on a diskette. This command can be typed alone, or it can be accompanied by file specifications, diskette drive specifications, or both.

DIR <RETURN>

lists all files on the currently active diskette. It is the same as the command DIR *.* <RETURN> (for information about using the wildcard specifiers, "*" and "?," see "Using File References" in Chapter 2). Following are some examples of valid DIR commands.

DIR X.Y <RETURN>	Displays the directory of the file X.Y.
DIR X?Z.C?M <RETURN>	Displays the directory of all files with three-letter names that begin with X and end with Z and three-letter extensions that begin with C and end with M.
DIR *.COM <RETURN>	Displays a directory of all files with a .COM extension.

The file name can be preceded by a drive name. The following DIR commands cause CP/M to list a directory of files on a specific drive. (If you don't specify a drive with the DIR command, CP/M automatically lists the directory of the currently active drive.)

- DIR B: <RETURN> Displays the directory of all files on the diskette in Drive B.
- DIR E:X.Y <RETURN> Displays the directory of the file X.Y on the virtual diskette in Drive E.
- DIR B:*.* <RETURN> Displays the directory of all files on the diskette in Drive B that have a three-letter extension that begins with C and ends with M.

ED

The ED command runs ED.COM, a program that lets you create and modify text files. ED is a *line-oriented editor*. That is, its primary focus is a single line within the text. If you want information about more than the line you are currently working on, you must issue commands to display the lines above and below your present location. This distinguishes ED from the *screen-oriented editors* usually used by word-processing systems. These editors show you the context in which you are working by automatically displaying a full screen of up-to-date text each time you change your working location.

If you intend to use your Professional's CP/M extensively to create and modify large documents, you should contact your dealer for information about one of the screen editors or word-processing programs currently available to run under CP/M. If, however, you expect to create only short text files, ED may be adequate.

The following sections provide you with the basic facts about ED. If you have never run this program before, you may find it helpful to do the sample session on ED in Chapter 2 before you read these sections.

Starting an Editing Session

To get started, type the ED command together with an unambiguous file name and type:

```
A>ED TEST.TXT <RETURN>
```


If TEST.TXT does not exist, ED will open a file of that name and then will display:

```
NEW FILE
: *
```

The asterisk is ED's prompt character. It indicates that the program is waiting for a command. At this point you have a variety of options that will be discussed later in this section. First, however, you need to learn some basics.

The Text Buffer Versus the Diskette File

When you begin creating a document by typing characters for ED to process, those characters are initially stored in the Professional's memory. This allows you to make any necessary changes before ED transfers the file to a diskette. Because memory can be quickly and easily altered, ED puts all the text it can in memory before it requires any transfers to diskette.

The portion of memory that ED allocates for this temporary storage task is called the text buffer. In the Professional, ED's text buffer has room for about 39,000 characters. (This amounts to about 13 pages of single-spaced text.)

When you invoke ED with a new file name, it not only opens a text buffer in memory, but also creates two new files on the diskette—in this example, TEST.TXT and TEST.***. Both are empty. When you exit from the editing session, ED does the following:

- Renames TEST.TXT to TEST.BAK
- Moves text from the buffer into TEST.***
- Renames TEST.*** to TEST.TXT

TEST.BAK thus becomes a backup copy containing the previous version of the file. If you have just created a new file and there is no previous version, TEST.BAK is empty. However, when you make future edits, the previous version becomes TEST.BAK and the new version becomes TEST.TXT.

This feature ensures that you always have both the last version and the next-to-the-last version of the file.

Routine Editing Operations

Most of your work with ED can be done easily if you familiarize yourself with the following concepts, discussed in the remainder of this section:

- The character pointer (CP)
- Commands for getting text into the buffer
- Commands for displaying text
- Commands for deleting text from the buffer
- Commands for moving the character pointer
- Commands for transferring text from the buffer to diskette
- Commands for ending an editing session

NOTE: In the following sections, the letter “n,” when used with a command, stands for a number that you choose to type together with the command. The word “string” refers to a word or group of words that you wish to manipulate by issuing the command.


The Character Pointer (CP)

The majority of operations that ED performs on text in the buffer are done with reference to the character pointer, called the CP. This is an imaginary arrow that marks where you are in the buffer. It sits either before the first character in the buffer, behind the last character in the buffer, or between two characters in the buffer. You always insert text to the immediate right of the CP, and you display text forward or backward from the CP.

Getting Text into the Buffer

The following commands put text into the buffer.

<i>Command</i>	<i>Function</i>
nA <RETURN>	Appends lines from diskette file to buffer. Moves n lines. (#A moves maximum number of 65,535 lines; 0A moves lines until file is totally transferred or until the buffer is half full.)
I <RETURN>	Enters insert mode. All subsequent text goes into the buffer until you type CTRL/Z. If V is selected (see following), each new line is assigned a number

(which does not go into the buffer). BS, CTRL/H, CTRL/R CTRL/U, and CTRL/X perform the same erase functions as in the CP/M monitor. The key erases the character to the left of the cursor from memory and echoes the character on your screen. (If you use the  key, you can type CTRL/R to redisplay the line with corrections.)

Istring <CTRL/Z>	Inserts the characters composing the string to the right of the CP; no RETURN is appended.
Istring <RETURN>	Inserts the characters composing the string and a RETURN to the right of the CP.

Displaying Text

The following commands display text in the buffer.

<i>Command</i>	<i>Function</i>
nT/-nT <RETURN>	Types lines. If n is positive, displays n lines after the CP. If n is negative, displays n lines before the CP. If the CP is in the middle of a line, 0T displays the text from the beginning of the line to the CP, and T displays the text from the CP to the end of the line. Thus, 0TT displays the whole line.
n/-n <RETURN>	Moves forward or backward n lines and displays the line found. (If n is positive, the CP moves forward. If n is negative, the CP moves backward.)
V/-V <RETURN>	Controls the form of the ED display. The normal display mode numbers the lines on the screen. These numbers, combined with ED's prompt symbol, take up space on the left side of the screen. The presence of the numbers prohibits formatting lines longer than 72 columns. However, invoking -V eliminates line numbering and permits displaying lines up to the full 80 columns provided by the terminal.

Deleting Text From the Buffer

The following commands delete text from the buffer.

Command	Function
nD/-nD <RETURN>	Deletes the n characters after or before the CP. (If n is positive, the n characters after the CP are deleted. If n is negative, the n characters before the CP are deleted.)
nK/-nK <RETURN>	Deletes the n lines after or before the CP. (If n is positive, the n lines after the CP are deleted. If n is negative, the n lines before the CP are deleted.) If the CP is in the middle of a line, 0K will delete all characters to the start of the line; -1K will delete those characters and all characters on the preceding line. Similarly, K or 1K will delete all characters between the CP and the beginning of the next line (including the RETURN that separates the lines).

Moving the Character Pointer

The following commands move the CP.

Command	Function
B/-B <RETURN>	Moves the CP to the beginning or to the end of the buffer. (B moves the CP to the beginning of the buffer. -B moves the CP to the end of the buffer.)
nC/-nC <RETURN>	Moves the CP n characters either forward or backward. (If n is positive, the CP moves forward. If n is negative, the CP moves backward.)
nL/-nL <RETURN>	Moves the CP n lines forward or backward.
: <RETURN>	Moves the CP to the beginning of the text buffer and displays the first line in the file.
n:	Moves the CP to the beginning of line number n.

- `:m <RETURN>` Defines the end of a range. For example, `:22T` would type from the present location through the end of line 22; `3::25T` would move the CP to the beginning of line 3 and type the buffer from that point through the end of line 25.
- `nP/-nP <RETURN>` Moves the CP forward and backward `n` pages, then displays each page.
- `nFstring <CTRL/Z> <RETURN>`
Finds, in the current buffer, the `n`th occurrence of a specified string of characters after the current CP position. Leaves the CP at the end of character string.
- `nNstring <CTRL/Z> <RETURN>`
Finds the `n`th occurrence of the specified string of characters after the current CP position, even if that string is not contained in the current buffer. If this occurrence cannot be found in the current buffer, ED writes the buffer to the output file, reads in and searches the next buffer and so on until the string is found or the end of the file is reached.
- `nSstring <CTRL/Z> substitute <CTRL/Z> <RETURN>`
Starting at the current CP position, finds all occurrences on `n` lines of the specified string of characters, and replaces each with the specified substitute string.
- `nJstring <CTRL/Z> insert <CTRL/Z> terminate <CTRL/Z> <RETURN>`
Starting at the current CP position, finds `n` occurrences of the specified string, places the insert string after each occurrence, and deletes all characters up to (but not including) the terminate string.

NOTE: Because RETURN terminates a search string, you cannot use RETURN as a character within the search string. If you need to include RETURN in a search string, use CTRL/L to represent it.

Moving Text from the Buffer to Diskette

The following commands move text from the buffer to a diskette file.

<i>Command</i>	<i>Function</i>
nW <RETURN>	Writes n lines from the buffer to the temporary file and deletes those lines from the buffer. Always begins with the first line in the buffer. If you replace n with a pound sign (#), the entire buffer is written to the diskette. If n is 0, writing is done until the buffer is only half full.
H <RETURN>	Moves the file into the temporary file and redisplay the : * ED prompt. In doing so, ED, in effect, performs the E (Exit) command (described below) followed by an ED file name command. The result is quick protection against operator error or equipment failure.

Ending an Editing Session

The following commands end an editing session.

<i>Command</i>	<i>Function</i>
E <RETURN>	Ends the editing session; copies the buffer and all remaining source file data to the temporary file; changes the source file type to .BAK; changes the temporary file type to that of the original source file. Returns to the CP/M prompt.
O <RETURN>	Returns to the original state of the source file, discarding any edits made in the current session. ED asks "O-(Y/N)?" before it proceeds, because all changes made during the current session will be discarded.
Q <RETURN>	Quits the editing session. Leaves the source file unchanged and deletes the temporary file. ED asks "Q-(Y/N)?" before it proceeds because all changes made during the current session will be discarded.

Advanced Editing Operations

The following commands are somewhat more complex and specialized than the routine commands just covered. You will probably not use the advanced commands on a regular basis, although they may occasionally be very useful.

<i>Command</i>	<i>Function</i>
nX <RETURN>	Transfers a block of text (the n lines following the CP) to a temporary file (X\$\$\$\$\$.LIB) called the block move file. If that file has previously been written into during the current session, the present block will be added to what the file already contains. Because you can read from this file (with the R command), this procedure allows you to move lengthy text passages from one location to another.
R <RETURN>	Reads the block move file (see the X command, above) into the buffer to the left of the CP.
Rname <RETURN>	Reads a previously created file called name.LIB into the buffer to the left of the CP. If you have boilerplate text that you routinely repeat, this command enables you to create it once and access it when needed. (You can create .LIB files with ED in the same way that you create any other files.)
U/-U <RETURN>	Converts all lowercase alphabetic input (from the keyboard or the source file) to uppercase. When U is negative, no conversion occurs.
OV <RETURN>	Displays two numbers, the first identifying the free bytes left in the buffer, the second identifying the total size of the buffer.

ERA

The ERA (erase) command erases a file or a group of files from a diskette directory and frees diskette space so that you can use it again. The following examples show several different cases of legal ERA commands. (For information about the use of the wildcard specifiers, “*” and “?,” see “Using File References” in Chapter 2.)

Erase the file EXAMPLE.DOC from Drive A (the currently active drive):

```
A>ERA EXAMPLE.DOC <RETURN>
```

Erase all files on Drive A with the name GONE:

```
A>ERA GONE.* <RETURN>
```

With Drive A as the currently active drive, erase all the files with the type .BAS from the diskette on Drive B:

```
A>ERA B:*.BAS <RETURN>
```

Erase all files from Drive B (Drive A is the currently active drive). If you do this, make sure that the diskette in Drive B contains files that you do not wish to keep.

```
A>ERA B:*. * <RETURN>
ALL (Y/N)?Y <RETURN>
```

PIP

The PIP command invokes PIP.COM, the CP/M Peripheral Interchange Program, which you can use to copy files from one system device to another. PIP allows you to transfer files between diskette drives, to output files, to printers and other devices connected to the serial ports on the rear of the system unit, and even to create a file directly from the keyboard.

The following discussion covers PIP as it is used in normal file transfer and maintenance. PIP has several attributes, however, that are primarily designed for programmers or unusual CP/M applications. In the interest of simplicity, these have not been covered in this section.

PIP can be invoked in either of two forms. For simple transfers that can be specified on one line, you can use PIP in what is called *command mode*. For more complex transfer operations, you will probably find *program mode* more convenient. These two modes are described next.

Command Mode

In this mode you type PIP, then a space, then a description of what you wish to do—destination first, then an equals sign (=), then the source. Following are some examples of using PIP in command mode to manipulate files.

Copy an existing file to a new file:

```
A>PIP ABC.NEW = XYZ.OLD <RETURN>
```


This command makes a copy of XYZ.OLD and gives it the name ABC.NEW. Both files are on Drive A, the currently active drive. The file, XYZ.OLD, is unaffected by the operation.

PIP allows the mixing of drives in one transfer specification and also allows multiple source files to be merged into one destination file. Create a new file on Drive B by merging two existing files on Drive A, the currently active drive:

```
A> PIP B:ABC.NEW = DEF.OLD,XYZ.OLD <RETURN>
```

This command copies DEF.OLD and XYZ.OLD from Drive A into a new file on Drive B, called ABC.NEW. (Note that when you use PIP to copy more than one file into a new file, PIP copies the files in the order that you specify when you type the command. PIP reads the source file specifications that you type from left to right and transfers them to the destination file in the order in which they are encountered.)

Wildcard specifiers are acceptable to PIP. (For information about the use of the wildcard specifiers, “*” and “?,” see “Using File References” in Chapter 2.)

```
A> PIP A: = B: * .COM
```

PIP copies all files with a file type of .COM from Drive B to Drive A.

PIP copies files from a diskette to an external device.

```
A> PIP LST: = TEST.TXT
```

This command copies the file, TEST.TXT, to the current list device. If this is a printer, PIP causes the file to be printed.

NOTE: Typing PIP CON:=TEST.TXT is equivalent to typing TYPE TEST.TXT because CON: is the logical device name for the terminal. (Logical and physical devices are covered in more detail in “Statistics on Devices” later in this chapter.)

Program Mode

You can also read the program PIP.COM into memory and run it. Simply type PIP <RETURN>. When you do this, PIP.COM is loaded into the Professional's memory and started. It displays an asterisk (*), then waits for you to specify a transfer operation and then press RETURN. (If you press RETURN without such a description, you will exit the PIP program, and the CP/M prompt will be redisplayed.) Following is an example of a possible PIP sequence.

```
A>PIP <RETURN>
*B:TEX3.TXT=A:TEX1.TXT,B:TEX2.TXT <RETURN>
*<RETURN>
A>
```

The first line calls and runs PIP.COM. PIP displays its prompt at the head of the second line. Here, the user types the command to merge the files TEX1.TXT (from Drive A) and TEX2.TXT (from Drive B) into TEX3.TXT (on Drive B). PIP performs this operation and returns its prompt. The user exits the PIP program by pressing RETURN.

PIP Parameters

PIP allows you to modify the action of any transfer by the inclusion of several symbols, which are called parameters. PIP parameters are single characters (or single characters followed by arguments), which are inserted between brackets ([]) at the end of the transfer command, as shown below.

```
A>PIP B: = FILENAME.TYP[E] <RETURN>
```

Several parameters can be combined in one set of brackets with no delimiting punctuation. When one or more parameters are thus included in a PIP command line, PIP's normal action will be modified as indicated in the following table.

<i>Parameter</i>	<i>Effect</i>
B	Performs a block mode transfer. PIP copies data into a buffer until CTRL/S is typed. PIP transfers that data from the buffer and then copies more data.
Dn	Deletes characters that extend past column n (counted from the last RETURN) as characters are transmitted. Use this parameter to shorten long lines being sent to a narrow printer or console device. (Devices are discussed in more detail later in this chapter.)

- E** Echoes (displays) all transfer operations on the console device as they occur. (This is a useful option only for text files.)
- F** Removes form feeds (page breaks) from the file. (You can use the P parameter simultaneously to insert new page breaks with a different number of lines per page. See Pn below.)
- Gn** Copies the file from user area n.
- H** Checks the data for proper hexadecimal format, then transfers the hexadecimal data. (Do not use with I parameter.)
- I** Checks the data for proper hexadecimal format. Ignores “:00” records. (Do not use with H parameter.)
- L** Translates uppercase characters to lowercase.
- N** Adds line numbers to each line transferred to the destination, starting at 1 and incrementing by 1. Leading zeroes are suppressed and the number is followed by a colon. If you specify N2, leading zeroes are included, and a tab is inserted following the number. If T is set (see the “Tn” entry in this list), the tab is expanded according to its specification.
- O** Transfers an object (nontext) file. Necessary for transferring program or binary data (but not .COM) files.
- Pn** Inserts form feeds (page breaks) after every n lines. If n = 1 or is omitted, form feeds occur every 60 lines. If the F parameter is used simultaneously, form feed suppression occurs before the new form feeds are inserted.

Qstring <CTRL/Z>	Quits copying from the source device or file when a specified string of characters (terminated by CTRL/Z) is encountered. Allows partial files to be copied. (Can be used with the S parameter to define a file segment that begins after the beginning of the source file and terminates before the end.) PIP must be entered in program mode if lowercase character strings are to be matched.
R	Copies system files.
Sstring <CTRL/Z>	Starts copying from the source device or file when the character string (terminated by CTRL/Z) is found. PIP must be in program mode, if lowercase character strings are to be matched.
Tn	Expands tabs to every nth column during the transfer of characters.
U	Translates lowercase characters to uppercase characters during the transfer.
V	Verifies that data has been copied correctly by rereading after the write operation. (This option works only if the transfer is to a diskette file.)
W	Overwrites a file that is set to read/only status. (Read/only status is discussed in the section that covers the STAT command.)
Z	Sets the parity bit to zero.

The following are examples of valid PIP commands that specify parameters in the file transfer.

```
A>PIP B:XX.TXT = A:RH.TXT[V] <RETURN>
```

This PIP command transfers XX.TXT from Drive A to Drive B and verifies that the transfer is correct.

NOTE: Notice that in the example above, A is the active drive. Therefore, you could rewrite this command as PIP B:XX.TXT=RH.TXT[V]<RETURN>. If the file you are copying is on the diskette in the active drive, you do not have to specify the drive name with that file. Likewise, if the new file is to be created on the diskette in the active drive, you do not have to specify the drive name when you name the new file.


```
PIP LST: = DATA.MAY[NT8U] <RETURN >
```

This PIP command transfers DATA.MAY to the LST: device. While doing so, it numbers each line, expands tabs to every eighth column, and translates lowercase alphabets to uppercase.

REN

The REN (Rename) command allows you to change the names of diskette files. The command form is REN NEWNAME.TYP=OLDNAME.TYP, where "NEWNAME" and "OLDNAME" are unambiguous file names that can also contain drive specifications. The following example changes the file name REPORT.JAN to COSTS.JAN.

```
A > REN COSTS.JAN = REPORT.JAN <RETURN >
```

You can precede either or both file names with a drive name; however, there are some rules:

- If you specify a drive name for the new name in the command line (COSTS.JAN), then the file for the old name in the command line (REPORT.JAN) is assumed to be on the diskette in that drive.
- If you specify a drive name for both files, you must specify the same drive name in both cases.

The following example changes the name REPORT.JAN to COSTS.JAN on Drive B.

```
A > REN B:COSTS.JAN = REPORT.JAN <RETURN >
```

The next example also changes the name REPORT.JAN to COSTS.JAN on Drive B.

```
A > REN B:COSTS.JAN = B:REPORT.JAN <RETURN >
```

STAT

The STAT command displays statistics for and permits certain controls over diskette files and system physical and logical devices. The following discussion will first concentrate on how STAT is used to get information about and change the status of files, then on how STAT can be used with devices.

This section covers STAT as it is used in normal CP/M operation. STAT has several attributes, however, that are primarily designed for programmers or unusual CP/M applications. In the interest of simplicity, these have not been covered in this section.

Statistics on Files

STAT commands perform the following operations on files.

- Display the amount of free space (in bytes) and the attributes of all diskettes used since the last start.
- Display the amount of free space on a specified diskette.
- Display the size and attributes of a single file or a group of files.
- Assign file access and/or directory attributes to a single file or group of files.

File Access Attributes

An R/O (read-only) file can be accessed only for reading. It cannot be changed or erased.

An R/W (read-write) file can be read, changed, and erased. R/W is the normal (default) attribute for CP/M files.

STAT's terms are defined as follows:

- Recs = Records = 128-byte units by which CP/M stores data.
- Bytes = Bytes = Basic storage unit (8 bits).
- Ext = Extents = Storage unit by which CP/M organizes data.
- Acc = file access attribute = R/W (read-write); R/O (read-only).
- K = kilo = 1024 bytes.

Examples Using STAT on Files

The examples in this section show you how to issue various commands using STAT. The examples also show the types of responses you get from CP/M when you issue STAT commands. The CP/M responses are intended only as examples. The statistics for your diskette will probably be different than the statistics shown in these examples.

The following examples use Drive A as the currently active drive. You can use STAT commands to access any drive on your system that contains an initialized diskette or virtual diskette.

Display the attributes and amount of free space (in bytes) on the diskettes in all drives accessed since the last start:

```
A>STAT <RETURN>
```

```
A: R/W, Space: nK
```

```
B: R/O, Space: nK
```

Display the amount of free space (in bytes) for the diskette on Drive A:

```
A>STAT A: <RETURN>
```

```
Bytes Remaining on A: 100K
```

Display the amount of space (in records, bytes, and extents) occupied by the file TEST.TXT:

```
A>STAT TEST.TXT <RETURN>
```

```
Recs Bytes Ext Acc
  24  3K  1 R/W A:TEST.TXT
```

```
Bytes Remaining on A: 100K
```

Display the amount of space occupied by each file with the type .TXT:

```
A>STAT *.TXT <RETURN>
```

```
Recs Bytes Ext Acc
  24   3K  1 R/W A:TEST.TXT
  12   2K  1 R/W A:NEXT.TXT
```

```
Bytes Remaining on A: 100K
```

Assign the attribute R/O to TEST.TXT:

```
A>STAT A:TEST.TXT $R/O <RETURN>
```

STAT replies:

```
TEST.TXT SET TO R/O
```

TEST.TXT is now set to read-only. It can be read but not written into or erased. This protection will be maintained until you change it again with STAT.

Assign the attribute R/W to TEST.TXT:

```
A>STAT TEST.TXT $R/W <RETURN>
```

STAT replies:

```
TEST.TXT SET TO R/W
```

TEST.TXT can now be both read from and written into. Attributes can be assigned to groups of files with wildcards. For example:

```
A> STAT B:*. * $R/O <RETURN>
```

sets all files on Diskette B to read-only.

Statistics on Devices

The Professional routinely communicates with three kinds of physical devices: the monitor, the keyboard, and the diskette drives. In addition, it can communicate with devices that you connect to it, such as printers, modems, and external computers. This communication takes place through a two-level access system involving logical and physical device names.

A logical device name identifies one of four generic processing functions:

<i>Name</i>	<i>Function</i>
CON:	This is the user console device. It interacts with CP/M, accepting input from a keyboard and displaying output on a video screen or paper.
RDR:	This device receives information (input only)
PUN:	This device sends information (output only).
LST:	This device lists information (output only).

Each logical device name can be associated (by means of STAT) with any one of several physical devices. CP/M allows 11 physical devices. This group is listed below.

<i>Name</i>	<i>Physical Device</i>
TTY:	Console
CRT:	Undefined
UC1:	Undefined
PTR:	Undefined
PTP:	Undefined
UR1:	File
UR2:	Undefined

UP1:	File
UP2:	Undefined
LPT:	Printer port
UL1:	File

You can communicate with physical devices (usually by means of PIP) by invoking either their physical or their logical names. The valid physical-to-logical device assignments are as follows.

<i>Logical Device</i>	<i>Physical Device</i>	<i>Default</i>
CON:	TTY:, CRT:, BAT:, UC1:	TTY:
RDR:	TTY:, PTR:, UR1:, UR2:	TTY:
PUN:	TTY:, PTP:, UP1:, UP2:	TTY:
LST:	TTY:, CRT:, LPT:, UL1:	TTY:

BAT: is a pseudophysical device. The name stands for batch processor. If BAT: is assigned to be the console device, CON:, then input comes from the current RDR: device, and output goes to the current LST: device.

STAT commands perform the following operations on physical and logical devices.

- Display the current physical devices and their connections to the logical devices.
- Assign physical devices to logical devices.
- Display diskette characteristics.
- Assign a temporary read-only status to a diskette drive.
- Display the possible physical-to-logical device assignments.

Examples of Using STAT on Devices

The examples in this section show you how to issue various commands using STAT. The examples also show the types of responses you get from CP/M when you issue STAT commands. The CP/M responses are intended only as examples. The statistics for your diskette will probably be different than the statistics shown in these examples.

Display the current physical-to-logical device assignments:

```
A> STAT DEV: <RETURN>
```

```
CON: is TTY:
```

```
RDR: is TTY:
```

```
PUN: is TTY:
```

```
LST: is TTY:
```

Display the possible physical-to-logical device assignments:

```
A> STAT VAL: <RETURN>
```

```
Temp R/O Disk: d: = R/O
```

```
Set Indicator: d:filename.typ $R/O $R/W $SYS $DIR
```

```
Disk Status : DSK: d:DSK
```

```
User Status : USR:
```

```
lobyte Assign:
```

```
CON: = TTY: CRT: BAT: UC1:
```

```
RDR: = TTY: PTR: UR1: UR2:
```

```
PUN: = TTY: PTP: UP1: UP2:
```

```
LST: = TTY: CRT: LPT: UL1:
```

The lines above convey the following information:

- To create a temporary read-only drive, type
STAT d: = R/O <RETURN>
("d" = drive name).
- To change the attributes of a file or files, type
STAT x:filename.type \$atr <RETURN>
("\$atr" is one of the attributes,
R/O, R/W, SYS, or DIR)

- To see the storage characteristics of drive x, type
`STAT x:DSK:<RETURN>`
 (“x” is the name of the drive in question; “STAT DSK:
`<RETURN>`” requests characteristics of all drives accessed since the
 last warm start.)
- To see the current user status, type
`STATUSR:<RETURN>`
- “Iobyte Assign:” introduces a list of possible physical-to-logical device
 assignments.

The following examples use Drive A as the currently active drive. You can use STAT commands to access any drive on your system that contains an initialized diskette or virtual diskette.

Assign the physical device TTY: to the logical device CON:

```
A> STAT CON: = TTY:<RETURN>
```

Assign the physical device CRT: to the logical device LST:, and the physical device TTY: to the logical device RDR::

```
A> STAT LST: = CRT:,RDR: = TTY:<RETURN>
```

Display the characteristics of the diskette in drive A:

```
A> STAT A:DSK:<RETURN>
```

```

A: Drive Characteristics
1368: 128 Byte Record Capacity
171: Kilobyte Drive Capacity
64: 32 Byte Directory Entries
64: Checked Directory Entries
128: Records/ Extent
8: Records/ Block
36: Sectors/ Track
2: Reserved Tracks
```

Assign a temporary read-only status to the diskette in Drive A:

```
A> STAT A: = R/O<RETURN>
```

Using STAT to write-protect a diskette is probably not advisable, because such protection is retained only in computer memory and is nullified when you restart the operating session. Some CP/M commands (PIP, for example) execute a warm start each time they return control to CP/M, so the protection is liable to disappear unexpectedly.

Write-protecting files is a more reliable procedure than write-protecting diskettes, because the diskette is actually modified (the write-protect bit associated on the diskette with the file in question is set). This modification persists through warm and cold starts until the protection is nullified by:

```
STAT drv:name.typ $R/W <RETURN >
```

Until a protected file is set back to read/write status, any effort to erase, rename, or overwrite it (with PIP) will result in an error message.

TRANSFERRING FILES BETWEEN P/OS AND CP/M—THE FILEXFER PROGRAM

When you are working with CP/M, you will probably want to transfer files back and forth between CP/M and P/OS. Your Professional's CP/M allows you to do this with a file transfer program called FILEXFER. When you use FILEXFER, you type some simple commands and you specify the file that you want to transfer. The FILEXFER program then uses the UR1 and UP1 devices to transfer the file.

NOTE: CP/M allows you to transfer only ASCII files between P/OS and CP/M. You **cannot** transfer binary files.

The FILEXFER program includes six commands to help you transfer files. They are:

- FILEXFER
- EXIT
- COPY
- ECHO
- HELP
- NAME

Each of these commands is discussed in this section.

In the previous chapter, you created a CP/M text file called COSTS.JAN. You will be using your COSTS.JAN file to practice transferring files in the examples below.

The FILEXFER command allows you to enter the FILEXFER program from the CP/M prompt. At the A > prompt type:

```
A> FILEXFER
```

CP/M displays:

```
P/OS-CP/M File Copy Program
```

```
Version 1.00
```

```
*
```

The asterisk (*) indicates that you are talking to the FILEXFER program. You can issue any of the following five FILEXFER commands (EXIT, COPY, ECHO, HELP, and NAME) whenever the asterisk is displayed.

The EXIT Command

The EXIT command allows you to leave the FILEXFER program. At the asterisk, type:

```
* EXIT
```

Then press RETURN. The A > prompt is redisplayed. You have exited from the FILEXFER program.

The COPY Command

The COPY command allows you to copy files between P/OS and CP/M. Return to the FILEXFER program to practice the COPY command. Now try to copy your COSTS.JAN file to P/OS. At the asterisk, type:

```
* COPY [USERFILES]TRANSFER.DOC/PRO = COSTS.JAN
```

Then press RETURN. The file is copied within several seconds. Then the asterisk is redisplayed. You have now copied your COSTS.JAN file from your CP/M directory into your P/OS USERFILES directory. You have named your P/OS file "TRANSFER.DOC."

You can check your P/OS directory to see that the new file, TRANSFER.DOC, is there. To do this, type EXIT at the asterisk and press RETURN. Then, at the A > prompt, press the SET-UP key. When you are prompted to do so, press the RESUME key. The CP/M Main Menu is redisplayed.

Press **MAIN SCREEN** to return to the P/OS Main Menu. Choose "File Services." If **USERFILES** is your current directory, select "Show current directory." You will see **TRANSFER.DOC**, the CP/M file that you transferred to P/OS, listed there. (If **USERFILES** is not your current directory, you will have to change your directory to **USERFILES** before you select "Show current directory.")

Now restart your CP/M operating session and return to the **FILEXFER** program to continue practicing the **COPY** command. Type **FILEXFER** at the **A >** prompt and press **RETURN**.

Now try to transfer your **TRANSFER.DOC** file to an initialized CP/M diskette on Drive B. (Make sure you have placed an initialized diskette in Drive B before you type the command.) At the asterisk, type the following command.

```
*COPY B:TEST.CPM =[USERFILES]TRANSFER.DOC/PRO
```

Press **RETURN**. The file is copied within several seconds. Then the asterisk is redisplayed.

You have now copied the **TRANSFER.DOC** file from your P/OS **USERFILES** directory into the directory for your CP/M diskette on Drive B and named it **TEST.CPM**.

You can check your directory for Drive B to see that the **TEST.CPM** file is there. At the asterisk, type **EXIT** and press **RETURN**. Then type **DIR B:** at the **A >** prompt. The directory for Drive B is displayed, and the new file, **TEST.CPM**, is listed there.

Now that you have used the **FILEXFER** program to transfer files, go back to the CP/M I/O Device Configuration Menu to see how it has changed. At the **A >** prompt, press **SET-UP** and then **RESUME**. The CP/M Main Menu is redisplayed.

Move the pointer to "Configure I/O devices," and press **DO**. When the CP/M I/O Device Configuration Menu is displayed, you will notice that the names for **UP1** and **UR1** have changed. **UP1**, which transferred the **COSTS.JAN** file to P/OS and named it **TRANSFER.DOC**, is now called **TRANSFER.DOC**. **UR1**, which transferred the **TRANSFER.DOC** file from P/OS, is also called **TRANSFER.DOC**.

CAUTION: The COPY command copies to whatever file name you specify on the destination diskette. If you are copying a file to CP/M and a file with the same name already exists on the CP/M diskette, it will be replaced by the new file **without comment**. If you are copying a file to P/OS and a file with the same name already exists, a new version of the P/OS file is created **without comment**.

The ECHO Command

When you use the ECHO command, the files that you request to be transferred from P/OS to CP/M will be displayed on your screen as they are being transferred. ECHO acts like a toggle switch. That is, if you type ECHO once, the ECHO capability is turned on. If you type ECHO a second time, the ECHO capability is turned off.

Try the ECHO command while transferring a P/OS file to CP/M. Restart your CP/M operating session and return to the FILEXFER program. At the asterisk, type ECHO. A message is displayed that tells you the ECHO is now on. Another asterisk is then immediately displayed. At the second asterisk, type:

```
*COPY ECHO.TST=[USERFILES]TRANSFER.DOC/PRO
```

The TRANSFER.DOC file is transferred to CP/M and named ECHO.TST. While the file is being transferred, it is displayed at your screen.

The HELP Command

The HELP command displays a help screen that describes the FILEXFER commands and how to use the COPY and NAME commands. You can see the help screen by typing **HELP** at the asterisk.

The NAME Command

The NAME command allows you to configure UR1, UP1, and UL1 with file names, without ending your CP/M operating session and returning to the CP/M menus. This command allows you to assign names to devices (See “Statistics on Devices” for more information on devices.) For example, suppose you are sending the output from a file to a device that has been set equal to UL1. If you use the NAME command to name UL1 “MYFILE.UL1,” then all output will be sent to the file called “MYFILE.UL1.”

Rename the UR1 device as USERTEST.UR1. At the asterisk, type:

```
*NAME UR1=[USERFILES]USERTEST.UR1
```

The UR1 device now corresponds to the P/OS file called USERTEST.UR1. You can check this by returning to the CP/M Main Menu and selecting “Configure I/O devices.”

SUBMIT

The SUBMIT command runs SUBMIT.COM, which allows you to execute previously generated sequences of CP/M commands stored in a file of the type .SUB. SUBMIT, therefore, permits you to accomplish very complex tasks in one command sequence.

The .SUB file, sometimes referred to as a command file, can include both self-contained commands and commands that must be completed by keyboard input. The latter capability allows you to specify which files, drives, or devices are to be involved at the time you run the command file.

Before you use SUBMIT, you must create a file with the file type .SUB, which contains the sequence of CP/M commands to execute. When you invoke SUBMIT, it looks for the .SUB file and creates on Drive A a second file, \$\$\$SUB, into which it writes the commands contained in your .SUB file. It then executes those commands one at a time, deleting the \$\$\$SUB file when it has finished. This means that there must be some free space on the diskette in Drive A and that the diskette cannot be write-protected.

Examples Using SUBMIT

SUBMIT has two forms—one in which it executes command files containing only complete commands, the other in which it executes command files that require you to supply certain processing information when you issue the SUBMIT command.

Command Files

This section contains two examples of using the SUBMIT command to execute command files that contain complete commands.

Example 1: First create a small text file using ED, and name the file MY.TXT. (You can type any text you want in the file.) Then use ED to create the file SHOW.SUB, which contains the following CP/M commands:

```
DIR *.TXT
PIP B:=A:*.TXT
DIR A:
DIR B:
```


Now execute SUBMIT. (Make sure there is an initialized diskette in Drive B.)
Type:

```
A>SUBMIT SHOW <RETURN>
```

CP/M creates a new file called \$\$\$SUB, which contains the four commands in SHOW.SUB. This process is shown below. (Note that if your diskette contains any other files with a .TXT extension, they too will appear wherever the MY.TXT file appears.)

```
A>ED MY.TXT
NEW FILE
: *I
1: THE QUICK BROWN FOX JUMPED OVER THE LAZY DOG.
2:
: *E

A>ED SHOW.SUB
NEW FILE
: *I
1: DIR *.TXT
2: PIP B:=A:*.TXT
3: DIR A:
4: DIR B:
5:
: *E

A>SUBMIT SHOW

A>DIR *.TXT
A: MY      TXT
A>PIP B:=A:*.TXT

COPYING -
MY.TXT

A>DIR A:
A: COSTS  BAK : ASM      COM : DDT      COM : DUMP    ASM
A: DUMP   COM : ED     COM : LOAD    COM : PIP     COM
A: STAT   COM : SUBMIT COM : XSUB    COM : COSTS  JAN
A: MY     BAK : MY     TXT : SHOW   BAK : SHOW   SUB
A: $$$    SUB : SYSGEN COM : FILEXFER COM
A>DIR B:
B: COSTS  JAN : MY     TXT
A>
```

Example 2: Create the following file, called SUBEX.SUB:

```
STAT
ED SAMPLE.TXT
TYPE SAMPLE.TXT
STAT
```

Now type:

```
SUBMIT SUBEX <RETURN>
```

When SUBEX.SUB executes, STAT displays the amount of free space left on the diskette in the active drive. Next, ED is invoked and you must create a new file, SAMPLE.TXT. (To create the new file, follow the ED procedures described in this chapter. After you create the file, you must exit it so that SUBEX.SUB can continue processing.)

When you exit ED, TYPE displays the contents of SAMPLE.TXT. Then STAT displays the amount of free space left on the diskette in the active drive. Finally, the A> prompt is redisplayed. The complete sequence is shown below.

```
A>ED SUBEX.SUB
NEW FILE
: *I
1: STAT
2: ED SAMPLE.TXT
3: TYPE SAMPLE.TXT
4: STAT
5:
: *E

A>SUBMIT SUBEX

A>STAT
A: R/W, Space: 310k

A>ED SAMPLE.TXT
NEW FILE
: *I
1: THE QUICK BROWN FOX JUMPED OVER THE LAZY DOG.
2:
: *E

A>TYPE SAMPLE.TXT
THE QUICK BROWN FOX JUMPED OVER THE LAZY DOG.

A>STAT
A: R/W, Space: 308k

A> █
```

Command Files that Require Additional User Input

You can execute command files (those containing commands that require additional input when you issue the SUBMIT command) by typing the SUBMIT command in the following format:

```
"SUBMIT name par1 par2 parn"
```

where "parn" is user input information (command parameters) that SUBMIT will accept. You can specify drive name, file name, file type, and PIP parameters as SUBMIT parameters.

The parameter strings will be inserted wherever the placeholder symbols \$1, \$2, \$3, and so on, occur in the command file. The first parameter string to be typed after the name of the SUBMIT file will be inserted in place of all occurrences of the \$1 placeholder symbol, the second string in place of all occurrences of the \$2 symbol, and so on.

NOTE: The \$ character is used only as part of a placeholder symbol in your .SUB file. Therefore, if you want an actual \$ character to be placed in your file, you must type it twice (\$\$).

Parameter strings must be separated by a space and no other punctuation.

Example: Use ED to create the following file, called TRANSFER.SUB:

```
PIP $2: = $1:$3.* <RETURN>
STAT $2:$3.* <RETURN>
```

The symbols \$1, \$2, and \$3 are placeholders for which SUBMIT will substitute arguments that you will provide when you invoke the command, and the colons and asterisks are literal additions that SUBMIT will append to those arguments—thus:

```
SUBMIT TRANSFER B A MY <RETURN>
```

With the aid of TRANSFER.SUB, SUBMIT will substitute B for \$1, A for \$2, and MY for \$3. In addition it will append a colon to B and A, separate the first two parameters in the first line with =, and append .* to MY. The result will be equivalent to having issued the following commands:


```
PIP A: = B:MY.* <RETURN>
STAT A:MY.* <RETURN>
```

NOTE: Format your .SUB files along the patterns just illustrated so that the user invoking SUBMIT is not required to include a period or colon within a parameter even to separate a file name from a file type.

For example:

```
A> SUBMIT TASK TEST.DOC <RETURN>
```

would result in an error because the period between TEST and DOC would be interpreted as a parameter separator rather than as a character in a single parameter.

To abort the execution of a command file, press either BS or  or type CTRL/H.

TYPE

The TYPE command displays the contents of a file containing printable characters on the current console device. (Note that if your console device is TTY: and you wish to have a printed record of a file you are about to display with TYPE, typing CTRL/P will cause console displays to be echoed on a printer connected to the printer port.) The basic form of the command is:

```
A>TYPE X.Y<RETURN>
```

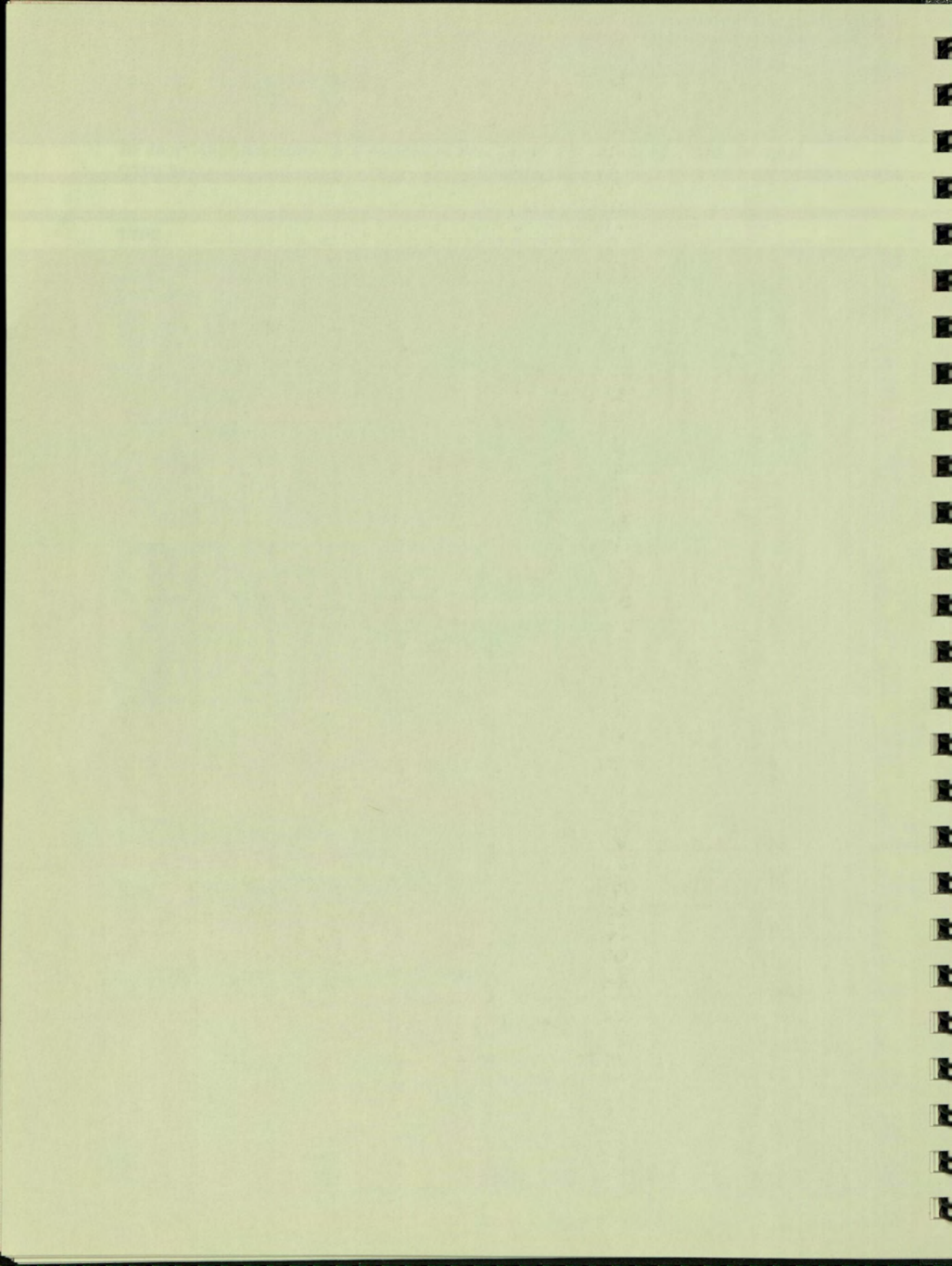
where “X” and “Y” represent an unambiguous file name and file type. You can also specify a drive name if the file you want to see is not on the current diskette:

```
A>TYPE B:X.Y<RETURN>
```

Press HOLD SCREEN to temporarily stop the display; press it again to continue. Press any other key to abort the display.

4

What to Do in Case
of Problems



Chapter 4

What to Do in Case of Problems

WHAT PROBLEMS ARE

For a computer user, problems are any behavior on the part of the machine that does not conform to the user's expectations. Problems occur because an electronic component has failed, because an essential connection has been broken, because a program has undetected errors (called "bugs"), because a file copy has become corrupted, or because the user has asked for inappropriate or impossible operations to be performed.

From this follows the first rule of troubleshooting: when a problem occurs, double-check to make sure you didn't cause it. If an operation seems to fail, reread the documentation covering that operation to make sure you haven't omitted an essential argument or skipped a command. Then try the operation again, carefully checking each step to make sure you invoke all required commands, arguments, and references.

Once you satisfy yourself that your command or commands are not at fault, think about the symptoms and try to deduce a possible explanation. If, for example, you suddenly begin having trouble with an applications program that seemed to work properly in the past, recopy the program from your master diskette and try again. Flexible diskettes are fragile both physically and magnetically. A crease in the diskette's cover or a dent or a fingerprint on the diskette surface is all it takes to convert a reliable data storage device into a source of frustration and anxiety.

HOW TO GET HELP

In general, problems with your Professional or with CP/M can be broken into two categories:

1. Problems that you can find and eliminate on your own.
2. Problems that require assistance from one or more of the vendors from whom you acquired your hardware and software system components.

Where you go for assistance will depend on whether the problem is with software or hardware.

- If you encounter software problems, contact the vendor who sold you the applications program with which you are having trouble.
- If you encounter hardware problems, contact your Digital representative or your Professional vendor.

It will be up to you to isolate which of these sources you should contact. This discrimination usually is not as complicated as it might seem. Clearly, if a new applications program cannot be copied or loaded without error, you have received it either in a corrupted state or in the wrong format. In either case, return it to the vendor for replacement.

If a new applications program loads and runs properly up to a point and then fails, make sure that it isn't expecting a device that you don't have (a printer, for example)—or that it doesn't require you to redefine a logical device. If none of these explanations fit, there are two possible offenders—a "bug" in the applications program or incompatibility between your version of CP/M and the applications program. In either case, your best bet is to contact the applications program vendor and see what he or she recommends. The chances are good that other users have already had the same problem. If that is the case, the vendor probably can tell you immediately how to fix it.

The odds are small that you will uncover an undetected bug in CP/M itself. CP/M is one of the most widely used operating systems in the world today. As a result, most of its unpredictable quirks have been found and corrected. Nevertheless, if CP/M behaves consistently in ways contrary to the definitions and examples in this book—and if these peculiarities cannot be corrected by recopying your system diskette—contact your Digital representative or the vendor from whom you purchased your CP/M application.

Before you contact a software vendor on any shortcoming, however, you should first confirm the reliability of your hardware. Your Professional contains an internal self-test program that runs every time the Professional power switch is turned on. If you receive any error message when you turn your system on, refer to Chapter 3 of the *Professional 300 Series Owner's Manual* for instructions on how to proceed.

If you cannot install your CP/M application on the hard disk, use your Self-Starting System Test diskette to diagnose the problem. (Chapter 3 of the *Professional 300 Series Owner's Manual* describes this procedure.) If your system fails these diagnostics, contact your Professional vendor or Digital representative and have the problem corrected before drawing any conclusions about the reliability of any piece of software.

SOME COMMON TROUBLE SYMPTOMS AND THEIR CURES

Not all correctable troubles with your system are caused by improperly understood system or applications program commands. Some originate in easily correctable switch, connector, keyboard, or monitor conditions. The following table identifies some trouble sources that you may encounter at one time or another—particularly if your Professional is shared by several people.

<i>Symptom</i>	<i>Possible Cause</i>	<i>Cure</i>
No response from monitor (no characters on screen, no cursor)	Professional power switch is off	Turn power switch on
P/OS Main Menu appears on screen at powerup, but pressing the keys on the keyboard has no effect	Keyboard is disconnected	Connect keyboard
Nothing visible on screen after 45 seconds of warmup	Screen brightness is set too low	Increase screen brightness

CP/M does not run and you receive an "I/O error attempting to mount disk foreign ..." error message when you select "Start CP/M" from the CP/M Main Menu

(1) No diskette in specified drive,
(2) drive door is not closed,
(3) or unreadable or non-system diskette is installed

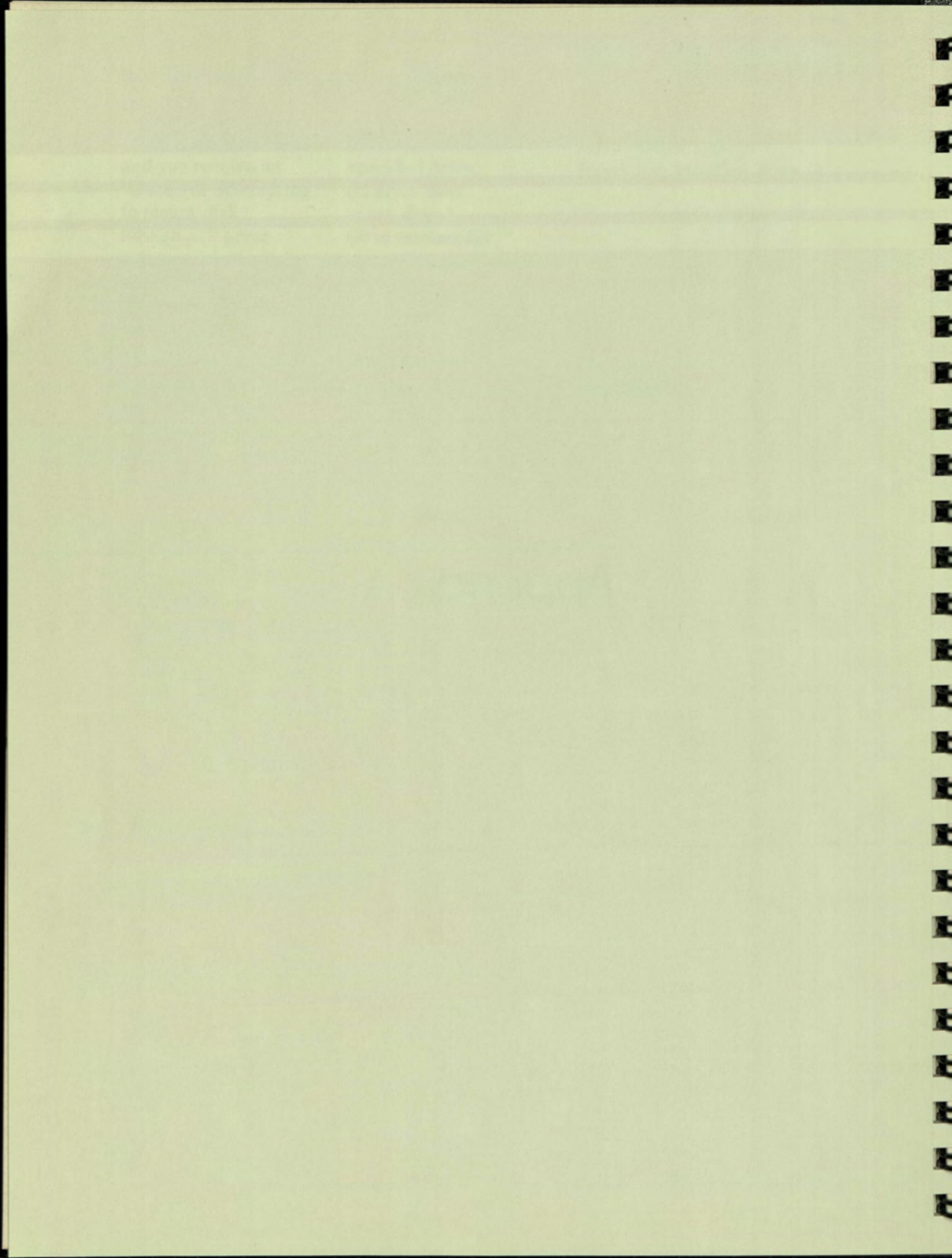
Install CP/M system diskette in specified drive

CP/M or applications program suddenly seems to go dead; keyboard operations do not echo on terminal

HOLD SCREEN function or CTRL/S has been activated

Press a letter or number key

Appendix A



Appendix A

How to Compress a Virtual Diskette

INTRODUCTION

The amount of space used by each virtual diskette expands to accommodate the files that the virtual diskette contains. The space used by a virtual diskette does not decrease, however, if you delete all or parts of that virtual diskette's files.

Because virtual diskettes do not decrease in size when you delete files they contain, you could waste valuable space on your hard disk by keeping virtual diskettes that have had files deleted. You therefore should compress your virtual diskettes from time to time. The following procedure describes how to compress virtual diskettes so that you can free unused space.

If, when you do this procedure, you receive a message that your hard disk is full, you will have to back up some of your files onto diskettes before you can complete the procedure. The section below, titled "What to Do if Your Hard Disk is Full," tells you how to proceed if this happens.

PROCEDURE

1. Initialize a new virtual diskette. (Chapter 1 describes how to do this.)
2. Configure the following two virtual diskettes: the virtual diskette you want to compress and the virtual diskette that you just initialized. (Chapter 1 describes the procedure for configuring virtual diskettes.)
3. Start a CP/M operating session. (Select "Start CP/M" from the CP/M Main Menu, as described in Chapter 1.)

4. At the A> prompt, use PIP to copy all the files from the virtual diskette you are compressing to the new virtual diskette. (PIP H:*. * = G:*. *[OVR], for example, copies the files from the old virtual diskette, G, to the new virtual diskette, H.) You now have two virtual diskettes that contain the same files. The new virtual diskette, however, uses only enough space to accommodate the files contained on it.
5. Press SET-UP and RESUME to end the CP/M operating session. The CP/M Main Menu is redisplayed.
6. From the CP/M Main Menu press MAIN SCREEN. The P/OS Main Menu is displayed.
7. Move the pointer to "File services," and press DO. The File Services Menu is displayed.
8. Move the pointer to "Delete file," and press DO. The File Selection Menu is displayed.
9. Move the pointer to the name of **the virtual diskette you are compressing**. DO NOT delete the new virtual diskette that you just created.
10. Press DO. The virtual diskette that you selected is deleted.

NOTE: You now have a new virtual diskette that contains all of the files that the old virtual diskette contained. The old virtual diskette has been deleted. The new diskette is only large enough to hold the files contained on it. It does not contain any of the empty space that resulted when files were deleted from the old virtual diskette.

WHAT TO DO IF YOUR HARD DISK IS FULL

If your hard disk does not have enough space left on it to hold both the virtual diskette that you are compressing and the new virtual diskette that you create in the process, you will receive an error message when you do the procedure described above. If this happens, follow the steps below to complete virtual diskette compression.

1. Insert a diskette into Drive B and use PIP to back up the files from the virtual diskette you want to compress. For example, PIP B:*. * = E:*. *[OVR] copies all the files from Drive E (the virtual diskette) to the diskette in Drive B. (Note that you either will have to use a diskette that has enough free space left to contain all the files from the virtual diskette, or you will have to use more than one diskette to copy all the files. If you have to use more than one diskette, use PIP to

copy individual files. For example, `PIP B:NEWFILE.EXT=E:OLD-FILE.EXT` copies `OLDFILE.EXT` from the virtual diskette on Drive E to `NEWFILE.EXT` on the diskette on Drive B. Use the `STAT` command to determine how much space the files on the virtual diskette occupy and how much space is available on the diskette in Drive B.)

2. When all the files have been copied, press **SET-UP** and **RESUME** to end the CP/M operating session. The CP/M Main Menu is redisplayed.
3. From the CP/M Main Menu press **MAIN SCREEN**. The P/OS Main Menu is displayed.
4. Move the pointer to "File services," and press **DO**. The File Services Menu is displayed.
5. Move the pointer to "Delete file," and press **DO**. The File Selection Menu is displayed.
6. Move the pointer to the name of the virtual diskette you are compressing.
7. Press **DO**. The virtual diskette has now been deleted, but its files are still contained on the diskette (or diskettes) onto which you copied them. If you wish to create a new virtual diskette and copy the files onto it, follow the remaining steps in this procedure.
8. Return to the CP/M Main Menu and initialize a new virtual diskette. (This procedure is described in Chapter 1.)
9. Configure the new virtual diskette according to the procedure described in Chapter 1.
10. Start a CP/M operating session.
11. At the `A>` prompt, use `PIP` to copy all the files from the back-up diskette to the new virtual diskette. If all the files are contained on the diskette in Drive B and the new virtual diskette is on Drive E, for example, type: `PIP E:*. * = B:*. *[OVR]`. If the files are contained on more than one diskette, use this format:

`PIP E:NEWFILE.EXT=B:OLDFILE.EXT.`

First paragraph of faint text.

Second paragraph of faint text.

Third paragraph of faint text.

Fourth paragraph of faint text.

Fifth paragraph of faint text.

Sixth paragraph of faint text.

Seventh paragraph of faint text.

Eighth paragraph of faint text.

Ninth paragraph of faint text.

Tenth paragraph of faint text.

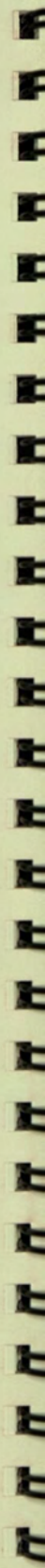
Eleventh paragraph of faint text.

Twelfth paragraph of faint text.

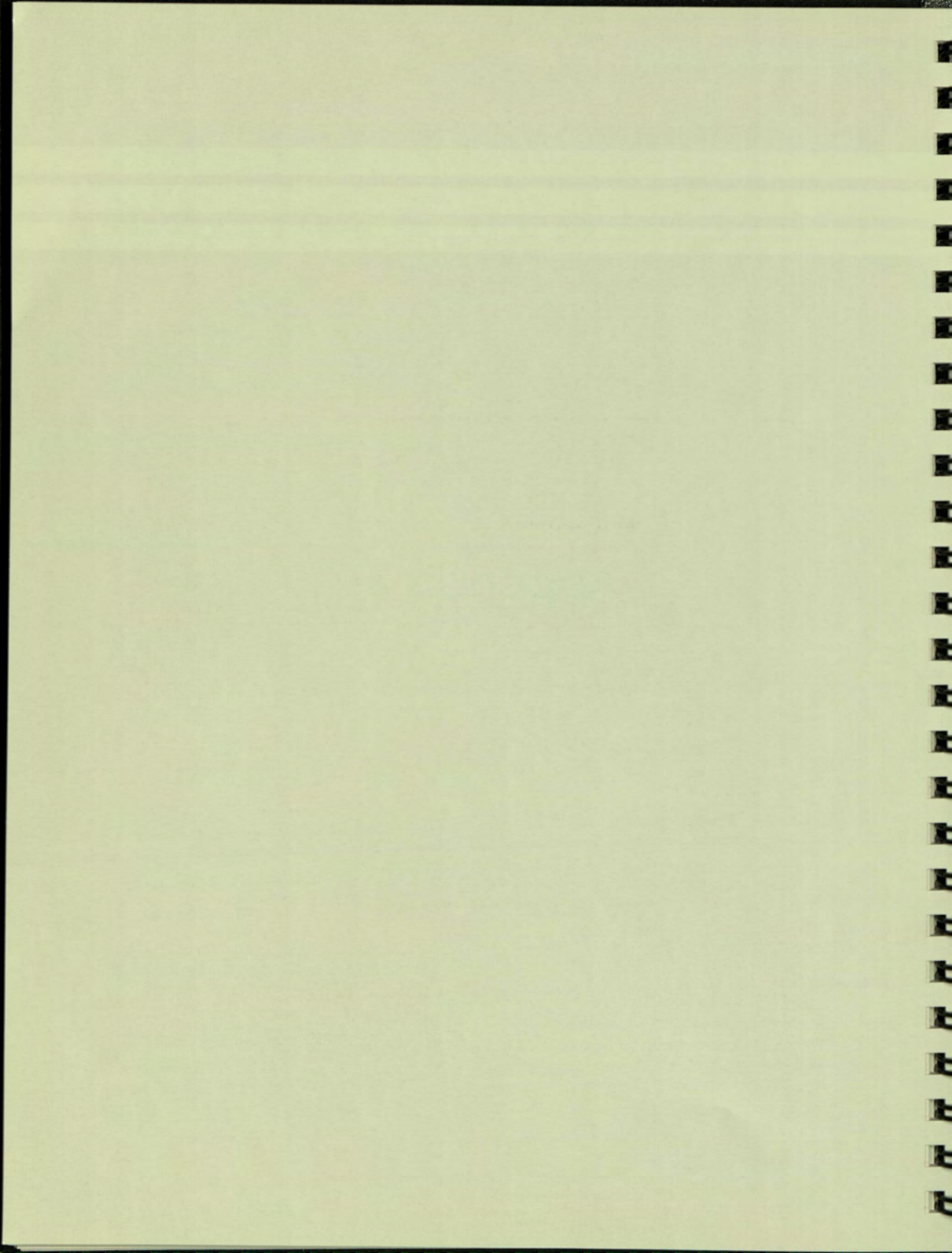
Thirteenth paragraph of faint text.

Fourteenth paragraph of faint text.

Fifteenth paragraph of faint text.



Appendix B



Appendix B

Error Messages

Occasionally when you use CP/M you will receive error messages. Depending on the type of error message you receive you can take one of the following actions:

1. You can always press **SET-UP** and **RESUME** to return to the CP/M Main Menu. This is not usually the most efficient action to take, but it works for all error messages. In just a few cases, it is the only way for you to correct the error condition.
2. For most messages you can type **CTRL/C** to return to the **A >** prompt and continue processing.
3. For some messages, you can type any key on the keyboard to return to the **A >** prompt to continue processing.

The list below contains the error messages that you are most likely to see in the course of CP/M processing. Included in the list is a brief description of the meaning of each error message as well as some instructions on how to recover from the error condition. The list does not contain all CP/M error messages. For convenience, the error messages are presented in alphabetical order.

Bdos Err On drv: Select

This message indicates that you specified a drive name other than A, B, E, F, G, or H. You can type any key to return to the **A >** prompt unless you tried to change the currently active drive. If you tried only to change the currently active drive, then you must press **SET-UP** and **RESUME** to recover from this error condition. (For example, if you typed **A > C:**, then you must press **SET-UP** and **RESUME** to recover from this condition.)

BREAK 'n' AT

:

This message occurs when you are using the ED command. It indicates that you issued a command to which ED could not respond. ED displays the first character that it cannot interpret on the line at which the problem occurred. The ED prompt is automatically redisplayed. You can enter the ED command that you want at the ED prompt to continue your editing session.

? Cannot create UP1 file, RMS error code = nnn

This message occurs when the diskette is full or when you specify an invalid file name for UP1. Press CTRL/C to return to the A> prompt.

? Cannot OPEN UR1 file, RMS error code nnn

This message means that the file name that you have specified for UR1 is invalid or does not exist. Press CTRL/C to return to the A> prompt.

CANNOT WRITE: dev:

This message indicates that you tried to write to a read-only device. For example, you tried to output data to PTR:. The A> prompt is automatically redisplayed after this message.

CP/M cancelled on user request

Press RESUME to return to CP/M Main Menu

This message appears when you press the SET-UP key. Press RESUME to return to the CP/M Main Menu.

Diskette is not a bootable CP/M system diskette

Press RESUME to return to CP/M Main Menu

This message appears if you try to start a CP/M operating session with any diskette other than your system diskette inserted in Drive A. Press RESUME, insert your system diskette in Drive A, and start the operating session again.

DISKETTE OR DIRECTORY FULL

This message occurs under the following circumstances:

- You typed ED RETURN without specifying a file name and file type.
- There is no room left on the diskette to accept data being written from the current ED buffer.

- There is no room left in the diskette directory area to enter another file specification. (CP/M allows a maximum of 128 entries in a diskette directory and 256 entries in a virtual diskette directory.)

Before you start an editing session, make sure that the current diskette has room for the text you plan to add or create. (Type `STAT drv: <RETURN>` to find out how much free space is left on a given diskette.) Remember that both the `.BAK` and the `.TXT` files must be stored on the same diskette.

When you receive the "DISKETTE OR DIRECTORY FULL" message, the `A>` prompt is automatically redisplayed.

DISKETTE WRITE ERROR: filename

This message occurs when you are using PIP. It indicates that the current copying operation requires more space than is available on the destination diskette. The transfer is automatically aborted, and the `A>` prompt redisplayed.

drv:name.typ?

This message indicates that you specified a nonexistent file name and/or file type or that you specified an ambiguous file name. The `A>` prompt is automatically redisplayed.

? Error creating UL1: file, RMS status = nnn/nnn

This message is displayed when the diskette is full or when you specify an invalid file name. Press `CTRL/C` to return to the `A>` prompt.

? Error in EXTEND for virtual diskette file, RMS error code = nnn/nnn

This message indicates that the diskette is full. Type `CTRL/C` to return to the `A>` prompt.

Error On Line 001 No 'SUB' File Present

This message indicates that the command file name specified in the `SUBMIT` command cannot be found. The `A>` prompt is automatically redisplayed.

? Error OPENing virtual diskette file, RMS error code = nnn/nnn

This message indicates that you have specified an invalid file name for the virtual diskette or that the file that you have named for the virtual diskette does not exist. Press `CTRL/C` to return to the `A>` prompt.

FILE EXISTS

This message is displayed when you use the REN command to specify an already existing file name as the new name for a file. When this happens, CP/M aborts the rename operation and redisplay the A> prompt.

```

** FILE IS READ/ONLY **
      : *

```

This message occurs if you try to use ED to access a file that was set (by STAT) to read-only status. Type CTRL/C to return to the A> prompt.

```

? Forced LPT output failed, I/O status = 375
  Device not ready

```

This message occurs if you attempt to send data to the LPT: device when the LPT: device is off-line.

INVALID ASSIGNMENT

This message indicates that, while using STAT, you tried to make an assignment prohibited by the assignment table (as shown by STAT VAL:). The A> prompt is automatically redisplayed.

INVALID FILE INDICATOR

This message indicates that you made a typing error while using STAT or tried to set files to a state other than R/W or R/O. The A> prompt is automatically redisplayed.

INVALID FORMAT: string

This message indicates that you mistyped a logical or physical name (string). For example, *string* would appear as LIST: if you typed PIP LIST:name.typ instead of PIP LST:name.typ. The A> prompt is automatically redisplayed after this message.

```

? I/O error attempting to mount disk foreign, I/O status code = 375
  Device not ready

```

Bdos Err on drv: Select

This message indicates that either there is no diskette in the requested drive or the drive door is not closed. Correct the problem and press any key to continue.

NO FILE

This message indicates that the specified file(s) could not be found on the selected diskette. The A> is automatically redisplayed on the line beneath the message.

NO FILE: = drv:NAME.TYP

This message occurs when you use the PIP command. It indicates that the specified file could not be found on the specified diskette. The A> prompt is automatically redisplayed.

? No file name for virtual diskette

Bdos Err On E: Select

This message indicates that you have requested a virtual diskette on a drive that has not been configured for the current operating session. Type any key to return to the A> prompt.

QUIT NOT FOUND: = drv:name.typ[Qxxx]

This message indicates that the Q parameter has been invoked, but the string (xxx) specifying the quitting point could not be found. The A> prompt is automatically redisplayed.

RX-50 read error, I/O status = nnn

Followed by:

Parity Error on Device

or

File Processor Device Read Error

or

Device not ready

These messages indicate that an unreadable diskette has been encountered in the named drive. You can type any key to return to the A> prompt unless you tried to change the currently active drive to the drive that contains the damaged diskette. If you tried only to change the currently active drive, then you must press **SET-UP** and **RESUME** to recover from this error condition. (For example, if you typed A>B:, and Drive B contained a damaged diskette, then you must press **SET-UP** and **RESUME** to recover from this condition.)

? RX-50 write error, I/O status = 364

Write attempt to locked unit

Bdos Err On B: Bad Sector

This message means that you tried to write to a diskette that is protected by a write-protect tab. Type **CTRL/C** to return to the A> prompt. (You will have to remove the write-protect tab from the diskette before you can write to it.)

START NOT FOUND: = drv:name.typ[Sxxx]

This message occurs when you are using PIP. It indicates that the S parameter has been invoked, but the string (xxx) specifying the starting point could not be found. The A > prompt is automatically redisplayed.

? The CP/M softcard must be present to run CP/M

This message indicates that you have not yet installed your Z-80 board or that the Z-80 board is not correctly installed. Press SET-UP and RESUME to end the CP/M operating session. Return to the P/OS Main Menu, and turn the system off before you install your Z-80 board.

? Time-out waiting for results of Z-80 self-test

This message occurs if your Z-80 board has a hardware problem. Press SET-UP and RESUME to end the CP/M operating session. Contact your vendor.

Z-80 halted at PC = nnnn. Registers at HALT:

AF = nnnn BC = nnnn DE = nnnn H = nnnn

A'F' = nnnn B'C' = nnnn D'E' = nnnn H'L' = nnnn

IX = nnnn IY = nnnn IR = nnnn SP = nnnn

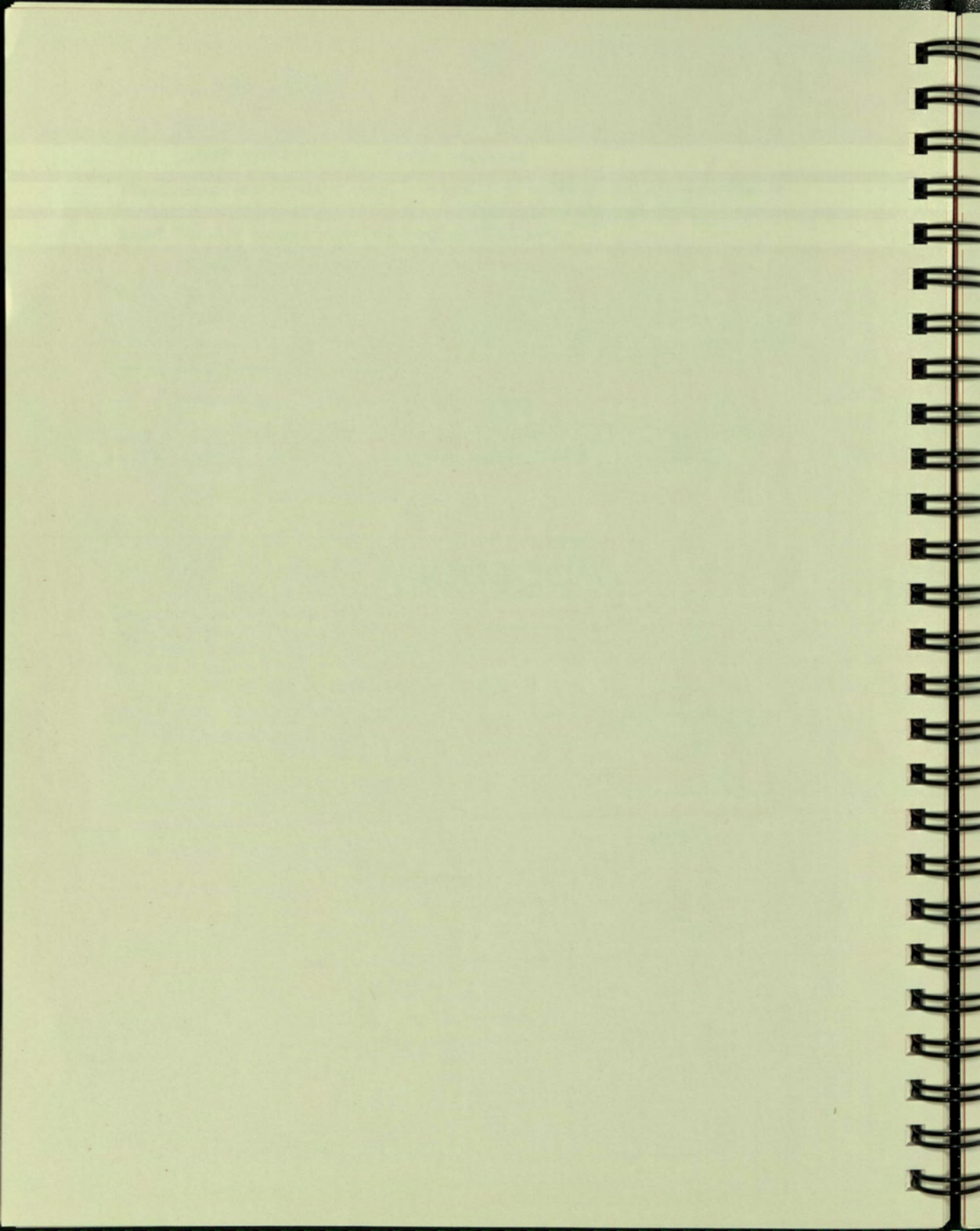
Press RESUME to return to CP/M Main Menu

This message occurs if the program has executed a HALT instruction. Press RESUME to return to the CP/M Main Menu.

? Z-80 self-test has failed, error status code = nnn

This message occurs if your Z-80 board has a hardware problem. Press SET-UP and RESUME to end the CP/M operating session. Contact your vendor.

Appendix C



Appendix C

System Calls

Assembly Language Programmers Only

The Professional's CP/M includes two additional system calls for assembly language programmers. These system calls are added at the end of the BIOS function table. The first system call retrieves system date and time information. The second system call allows assembly language programmers to design programs in which users can specify P/OS file names for the UR1, UP1, and UL1 devices. (The NAME command in the FILEXFER program performs this task. With the system call, assembly language programmers can incorporate the NAME command into other programs.)

RETRIEVING SYSTEM INFORMATION

The following parameters describe the call to retrieve system information. This call is added at BIOS + 36H.

Enter routine:

Register C—Request code

1—Date/Time

2—System serial information

Register HL—Address of buffer to store response

Exit routine:

HL—Pointer to buffer containing:

Request 1-8 bytes

- 1) Year-1900
- 2) Month
- 3) Day
- 4) Hour
- 5) Minute
- 6) Second
- 7) Tick
- 8) Ticks/Second

Request 2-6 bytes of system serial number

Low order byte first

CALL TO SPECIFY P/OS FILE NAMES

The following parameters describe the call to specify P/OS file names for the UR1, UP1, and UL1 devices. The call is added at BIOS + 39H.

Enter routine:

Register C—Request code

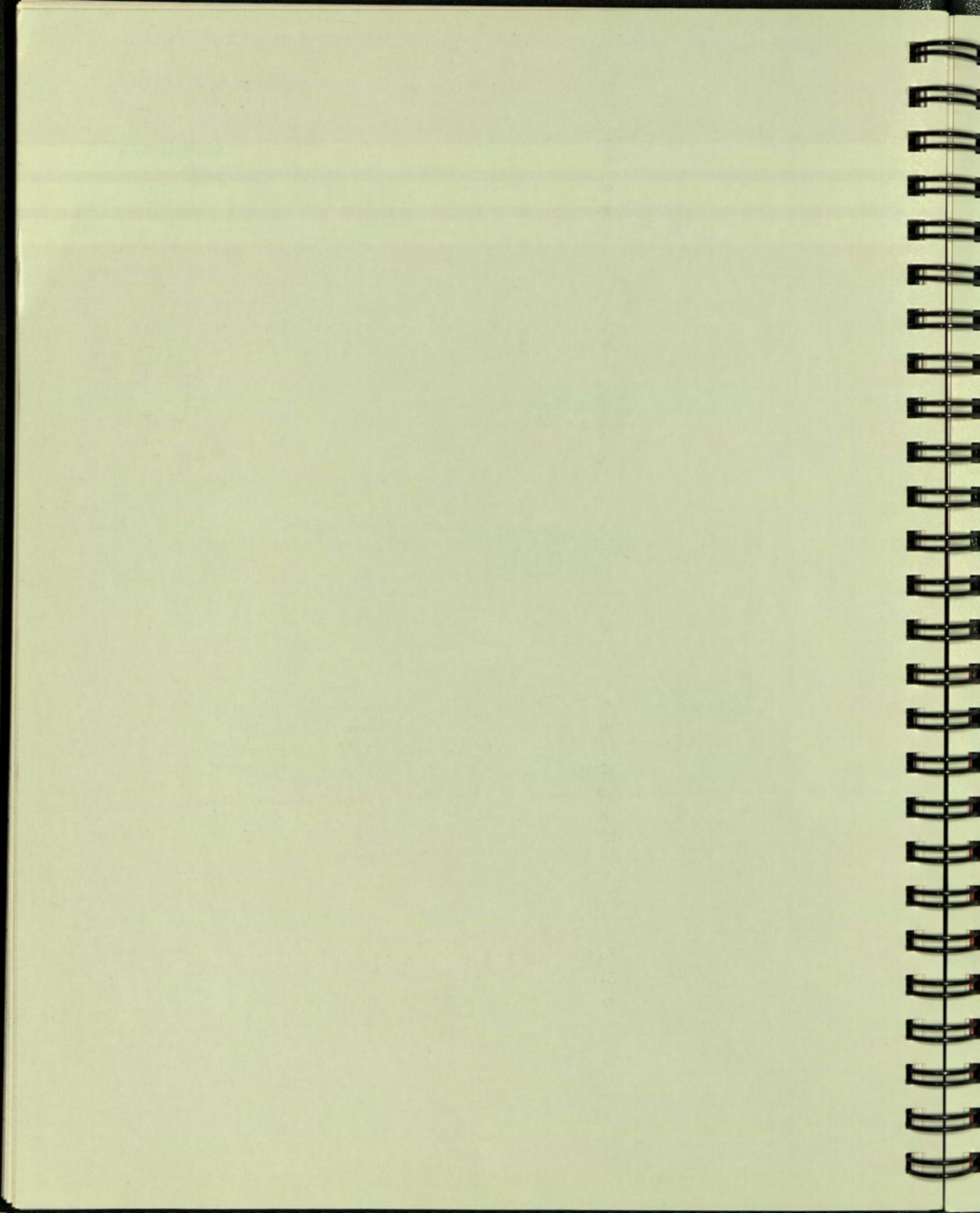
- 1—Specify UR1 file name
- 2—Specify UP1 file name
- 3—Specify UL1 file name

Register HL—Pointer to buffer. First byte of buffer is the length of the file specification string that follows. (Note that the string must be a valid P/OS file specification.)

Byte 1—number of bytes in specification

Bytes 2 through n + 1—bytes of the specification itself

Index



Index

- <X> key, 23 to 24, 28
- ADDTNL OPTIONS key, 27
- Ambiguous file references, 35
- Application diskette, ii
 - copying, 16 to 17
- Applications programs, 46, 83 to 85
- Applying a write-protect tab, 22
- Arguments
 - PIP, 62 to 64
- Arrow keys, 26
- ASM command, 50
- ASM.COM, 50
- Assemble command, 50
- Assembly language programs
 - information about, ii
- Assignments
 - device, 69
- Auxiliary keypad keys, 25 to 26

- Backing up diskettes, 16 to 18, 31
- Backing up files, 44, 53
- BACKSPACE key, 23 to 24
- BAK files, 53
- BAT, 69
- BREAK key, 27
- BS key, 23 to 24, 27, 28
- Buffers
 - text, 38
 - using ED with, 58
- Bytes, 30, 43 to 44

- CANCEL key, 27
- Changing file names, 42 to 43, 50, 65
- Character pointer
 - ED, 39 to 40, 54
 - moving, 56 to 57
- Checking directory, 15
- COM files, 49
- Command level, 21
- Command mode
 - PIP, 60 to 61
- Commands
 - advanced ED, 59
 - ASM, 50
 - CP/M, 1, 21, 36 to 37, 49
 - DDT, 50
 - DIR, 15, 32 to 33, 34, 35, 36, 50, 51
 - DUMP, 50
 - ED, 36 to 37, 50, 52
 - ERA, 3 to 11, 36, 45, 50, 59 to 60
 - FILEXFER, 36, 50, 72 to 76
 - for ending ED, 41
 - issuing, 21, 32
 - LOAD, 50
 - PIP, 36, 44, 50, 60
 - REN, 36, 42 to 43, 50, 65
 - resident, 36, 49
 - SAVE, 36, 50
 - STAT, 36, 43, 50, 65
 - SUBMIT, 51, 76
 - SYSGEN, 51

INDEX

- transient, 36, 49
- TYPE, 36, 42, 51, 80
- USER, 36, 51
- using, 37, 49 to 50
- XSUB, 51
- Commands used with FILEXFER, 72 to 76
- COMPOSE CHARACTER key, 23
- Compressing virtual diskettes, 8, 89
- CON:, 68 to 69
- Configuring I/O devices, 8 to 11
- Configuring virtual diskettes, 11 to 13
- Control Keys, 24 to 25
 - CTRL/C, 24
 - CTRL/E, 24
 - CTRL/H, 24
 - CTRL/J, 24
 - CTRL/M, 24
 - CTRL/P, 24
 - CTRL/R, 24
 - CTRL/S, 24
 - CTRL/U, 25, 29
 - CTRL/X, 25, 28
 - CTRL/Z, 25
- COPY
 - with FILEXFER, 73 to 75
- Copying diskettes, 16 to 18, 22, 31
- Copying files, 36, 44 to 45, 50, 60 to 65
- Copying virtual diskettes, 31
- Correcting typing errors, 28 to 29
- CP, 39 to 40, 54
 - moving, 56 to 57
- CP/M
 - Boot Screen, 14
 - commands, 1, 21, 36 to 37, 49
 - default drive, 33
 - definition of, 1
 - error messages, 29
 - file names, 32
 - file references, 35
 - installing, 2
 - prompt, 14, 21
- Creating files, 37, 50, 52
- CRT:, 68 to 69
- CTRL/C, 24, 30
- CTRL/E, 24
- CTRL/H, 24
- CTRL/J, 24
- CTRL/M, 24
- CTRL/P, 24
- CTRL/R, 24
- CTRL/S, 24
- CTRL/U, 25, 29
- CTRL/X, 25, 28
- CTRL/Z, 25
- DDT command, 50
- DDT.COM, 50
- Debugging
 - information about, ii
- Debugging command, 50
- Default drive, 33
- DELETE key, 23, 28
- Deleting files, 45, 50, 59 to 60
- Device assignments, 69
- Devices
 - configuring, 8 to 11
 - logical, 68 to 69
 - physical, 68 to 69
- DIR command, 15, 32 to 33, 34, 35, 36, 50, 51
- DIR examples, 51
- Directories
 - displaying, 32 to 33, 50, 51 to 52
- Directory
 - checking, 15
- Diskettes, 30
 - application, ii
 - backing up, 16 to 18, 31
 - copying, 16 to 18, 22, 31
 - initializing, 6
 - inserting, 31
 - storing information on, 30 to 31
 - System, ii, 2
 - using, 31
 - virtual, 7, 30, 31
 - write protecting, 22
- Display commands
 - ED, 55
- DO key, 27
- Drives
 - changing currently active, 33 to 35
- DUMP command, 50
- DUMP.COM, 50
- ECHO
 - with FILEXFER, 75
- ED
 - advanced editing, 59
 - display commands, 55
 - ending an editing session, 41, 58
 - examples, 52
 - getting text into the buffer, 54 to 55
 - inserting text, 38
 - moving text from buffers, 58
 - prompt, 38, 52
 - routine editing, 54
 - sample session, 37
 - text buffer, 38, 53
 - text deletion, 56
 - the CP, 39 to 40
- ED and large documents, 52
- ED buffer
 - getting text into, 54 to 55
- ED character pointer, 39 to 40, 54
 - moving, 56 to 57

- ED command, 36 to 37, 50, 52
- ED CP, 54
- ED.COM, 50
- Editing files, 37, 50, 52
- Editing keypad keys, 25 to 26
- Editors
 - line versus screen, 52
- Ending an ED session, 58
- Ending an operating session, 15
- ENTER key, 23
- ERA command, 36, 45, 50, 59 to 60
- ERA examples, 60
- Erasing errors, 23 to 24, 28 to 29
- Erasing files, 50, 59 to 60
- Error conditions, 30
- Error messages, 95
 - CP/M, 29
 - responding to, 95
- Errors
 - correcting, 28 to 29
- ESC key, 27
- Examining files, 51
- Examples
 - DIR, 51
 - ED, 52
 - ERA, 60
 - PIP, 60 to 62, 64 to 65
 - REN, 65
 - STAT, 66 to 68
 - STAT on devices, 68 to 72
 - SUBMIT, 76 to 78
- EXIT
 - with FILEXFER, 73
- EXIT key, 4, 27
- Extent, 44

- F5 key, 27
- F17 key, 27
- F18 key, 27
- F19 key, 27
- F20 key, 27
- File access attributes, 66
- File extensions, 32
- File names
 - changing, 42 to 43, 50, 65
 - creating, 32
- File types, 32
- Files, 30
 - .BAK, 53
 - .COM, 49
 - .SUB, 76
 - ambiguous, 35
 - as virtual diskettes, 12
 - backing up, 44, 53
 - copying, 36, 44 to 45, 50, 60 to 65
 - creating, 37, 50, 52
 - deleting, 45, 50, 59 to 60
 - displaying, 42, 51
 - editing, 37, 50, 52
 - erasing, 50, 59 to 60
 - getting statistics on, 43 to 44
 - inserting text in, 38
 - renaming, 42 to 43, 50, 65
 - searching for, 35, 50
 - transfer between CP/M and P/OS, 9, 50, 72 to 76
 - transferring, 36, 44 to 45, 50, 60 to 65
 - unambiguous, 35
 - URL:, UPL:, and ULI: as, 10
 - valid names for, 32
- FILEXFER
 - COPY with, 73 to 75
 - ECHO with, 75
 - EXIT with, 73
 - HELP with, 75
 - NAME with, 75 to 76
- FILEXFER command, 36, 50, 72 to 76
- FILEXFER commands, 72 to 76
- FILEXFER.COM, 50, 72 to 76
- FIND key, 26
- Function keys, 25, 27

- HELP
 - with FILEXFER, 75
- Help
 - getting, 83, 84
- HELP key, 3, 27
- HOLD light, 23
- HOLD SCREEN key, 23, 27

- I/O devices
 - configuring, 8 to 11
- Initializing CP/M diskettes, 6
- Initializing virtual diskettes, 8
- INSERT HERE key, 26
- Inserting diskettes, 31
- Inserting text in a file, 38
- Installing CP/M application, 2
- INTERRUPT key, 27
- Issuing commands, 21, 32

- Keys
 - <X> 23 to 24, 28
 - ADDTNL OPTIONS, 27
 - arrow, 26
 - auxiliary keypad, 25 to 26
 - BACKSPACE, 23 to 24
 - BREAK, 27
 - BS, 23 to 24, 27, 28
 - CANCEL, 27
 - COMPOSE CHARACTER, 23
 - Control, 24 to 25
 - CTRL, 24 to 25

INDEX

- CTRL/C, 30
- DELETE, 23, 28
- DO, 27
- editing keypad, 25 to 26
- ENTER, 23
- ESC, 27
- EXIT, 4, 27
- F5, 27
- F17, 27
- F18, 27
- F19, 27
- F20, 27
- FIND, 26
- function, 25, 27
- HELP, 3, 27
- HOLD SCREEN, 23, 27
- INSERT HERE, 26
- INTERRUPT, 27
- LF, 23, 27
- LINE FEED, 23
- MAIN SCREEN, 4, 27
- NEXT SCREEN, 3, 26
- PF1, 26
- PF2, 26
- PF3, 26
- PF4, 26
- PREV SCREEN, 3, 26
- PRINT SCREEN, 27
- REMOVE, 26
- RESUME, 3, 27, 30
- RETURN, 23
- SELECT, 26
- SET-UP, 22, 27, 30
- Keys to which CP/M responds, 22 to 25
- LF key, 23, 27
- LINE FEED key, 23
- Line input to SUBMIT, 57
- Line-oriented editors, 52
- LOAD command, 50
- LOAD.COM, 50
- Logical devices, 68 to 69
- LPT., 69
- LST., 68 to 69
- MAIN SCREEN key, 4, 27
- Menus
 - CP/M I/O Device Config., 9
 - CP/M Main, 2, 5
 - CP/M Main Help, 3
 - CP/M Virtual Diskette Config., 12
 - Initialize CP/M Diskette, 6
 - using, 2
- Messages
 - CP/M error, 29
- Moving text with ED, 58
- NAME
 - with FILEXFER, 75 to 76
- Naming files, 32
- NEXT SCREEN key, 3, 26
- Operating session
 - ending, 15
 - starting, 13 to 14
- Operating system
 - definition of, 1
- Parameter strings
 - with SUBMIT, 78 to 79
- Parameters
 - PIP, 62 to 64
- PF1 key, 26
- PF2 key, 26
- PF3 key, 26
- PF4 key, 26
- Physical devices, 68 to 69
- PIP arguments, 62 to 64
- PIP command, 36, 44, 50, 60
- PIP command mode, 60 to 61
- PIP examples, 60 to 62, 64 to 65
- PIP parameters, 62 to 64
- PIP program mode, 62
- PIP.COM, 50
- PREV SCREEN key, 3, 26
- PRINT SCREEN key, 27
- PRO-CP/M-80 APPLICATION DISKETTE, 2
- PRO-CP/M-80 SYSTEM DISKETTE, 2
- Problems
 - cures, 85 to 86
 - definition, 83
 - symptoms, 85 to 86
 - what to do about, 83
- Program mode
 - PIP, 62
- Programs
 - applications, 46, 83 to 85
- Prompt
 - CP/M, 14, 21
 - ED, 38, 52
- Protecting diskettes, 22
- PTP., 68 to 69
- PTR., 68 to 69
- PUN., 68 to 69
- RDR., 68 to 69
- Record, 44
- Referring to files, 35
- REMOVE key, 26
- REN command, 36, 42 to 43, 50, 65
- REN examples, 65
- Renaming files, 42 to 43, 50, 65

- Reset I/O device table, 11
- Reset virtual diskette table, 13
- Resident commands, 36, 49
- RESUME key, 3, 27, 30
- RETURN
 - function of, 50
- RETURN key, 23
- Routine editing with ED, 54

- SAVE command, 36, 50
- Save I/O device table, 11
- Save virtual diskette table, 13
- Screen-oriented editors, 52
- Searching for files, 35, 50
- Sectors, 30
- SELECT key, 26
- SET-UP key, 22, 27, 30
- Start CP/M, 13 to 14
- Starting a CP/M operating session, 13 to 14
- STAT
 - command, 36, 43, 50, 65
 - examples, 66 to 68
 - file access attributes, 66
 - on devices, 68 to 72
 - statistics on files, 43 to 44, 66
 - write-protecting with, 71 to 72
- string, 54
- SUB files, 76
- SUBMIT
 - aborting execution, 80
 - additional user input with, 78 to 80
- SUBMIT command, 51, 76
- SUBMIT examples, 76 to 78
- SUBMIT parameter strings, 78 to 79
- SUBMIT.COM, 51
- SYSGEN command, 51
- SYSGEN.COM, 51
- System calls, 103
- System diskette, ii, 2
 - copying, 16 to 18

- Text
 - inserting in a file, 38
- Text buffer
 - ED, 38, 53

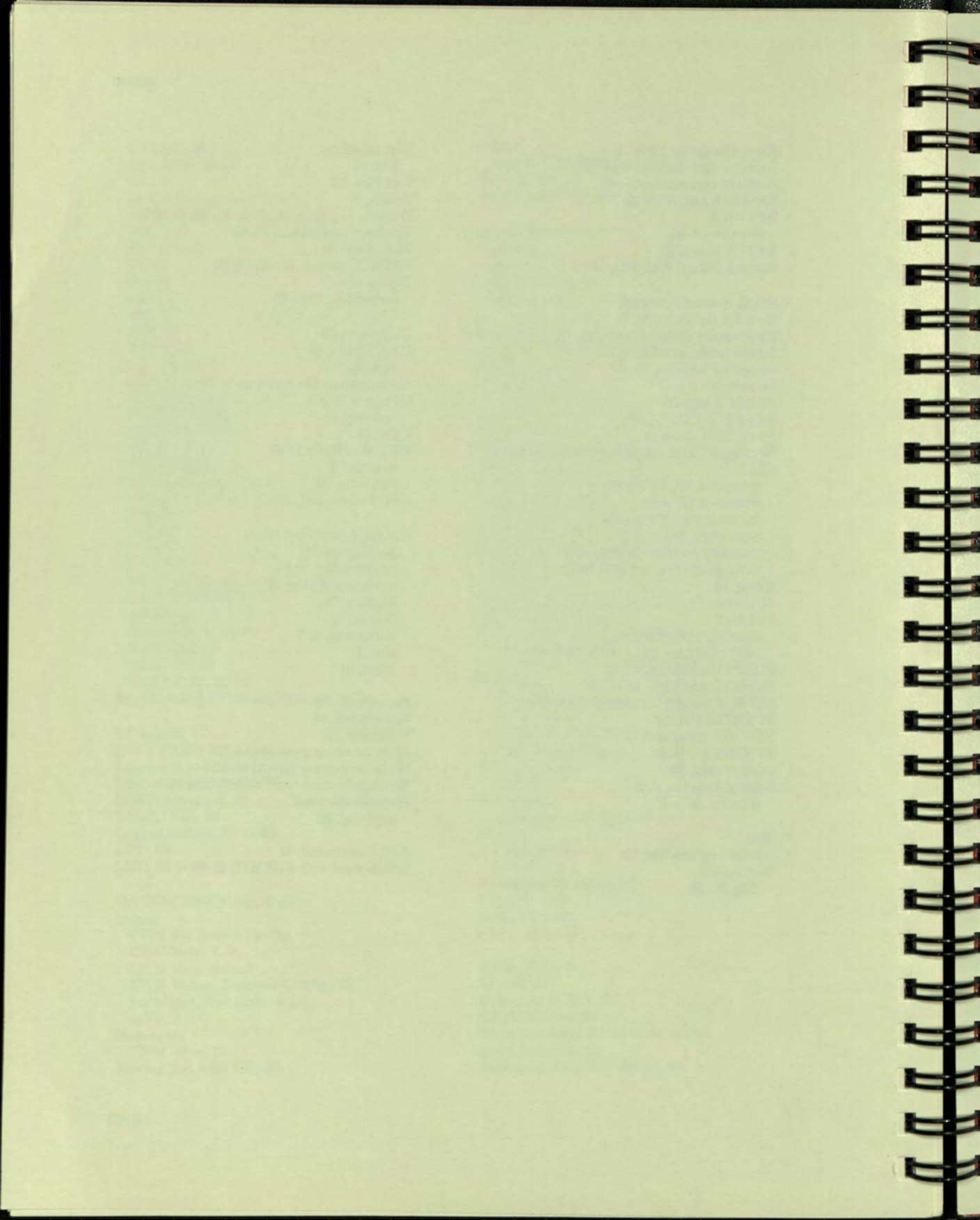
- Text deletion
 - ED, 56
- Text files, 37
- Tracks, 30
- Transferring files, 36, 44 to 45, 50, 60 to 65
- Transient commands, 36, 49
- TTY:, 68 to 69
- TYPE command, 36, 42, 51, 80
- Typing errors
 - correcting, 28 to 29

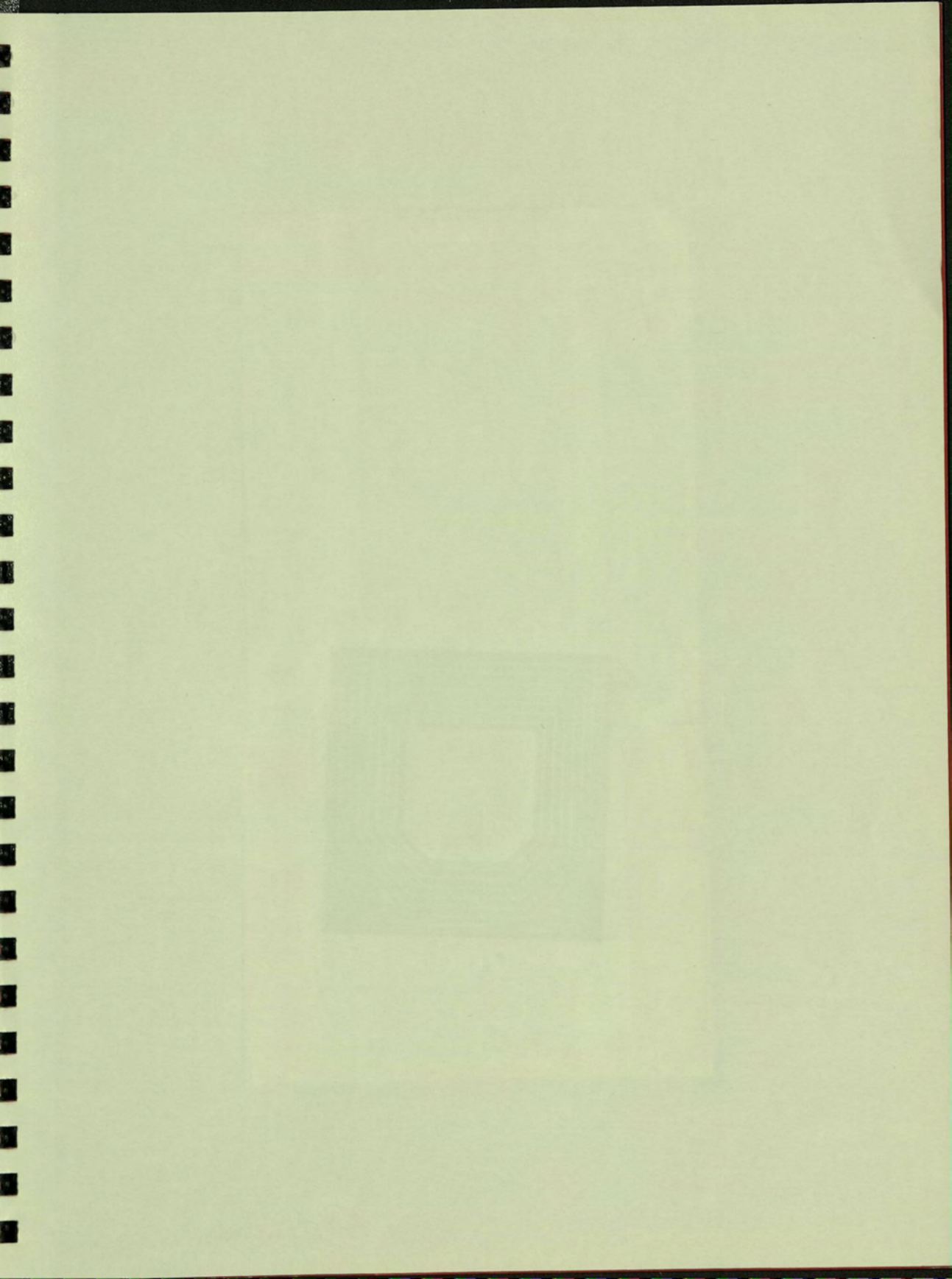
- UCL:, 68 to 69
- ULI:, 9 to 10, 69
 - as a file, 10
- Unambiguous file references, 35
- UP1:, 9 to 10, 69
 - as a file, 10
- UP2:, 69
- UR1:, 9 to 10, 68 to 69
 - as a file, 10
- UR2:, 68 to 69
- USER command, 36, 51

- Virtual diskettes, 7, 30, 31
 - backing up, 31
 - compressing, 8, 89
 - configuring, 11 to 13
 - copying, 31
 - files as, 12
 - initializing, 6, 8
 - size, 31
 - using, 31

- Warm boot, 30
- Warm start, 30
- Wildcards, 35
- Word processing programs, 52
- Write protecting diskettes, 22
- Write protection with STAT, 71 to 72
- Write-Protect tab
 - applying, 22

- XSUB command, 51
- XSUB used with SUBMIT, 51





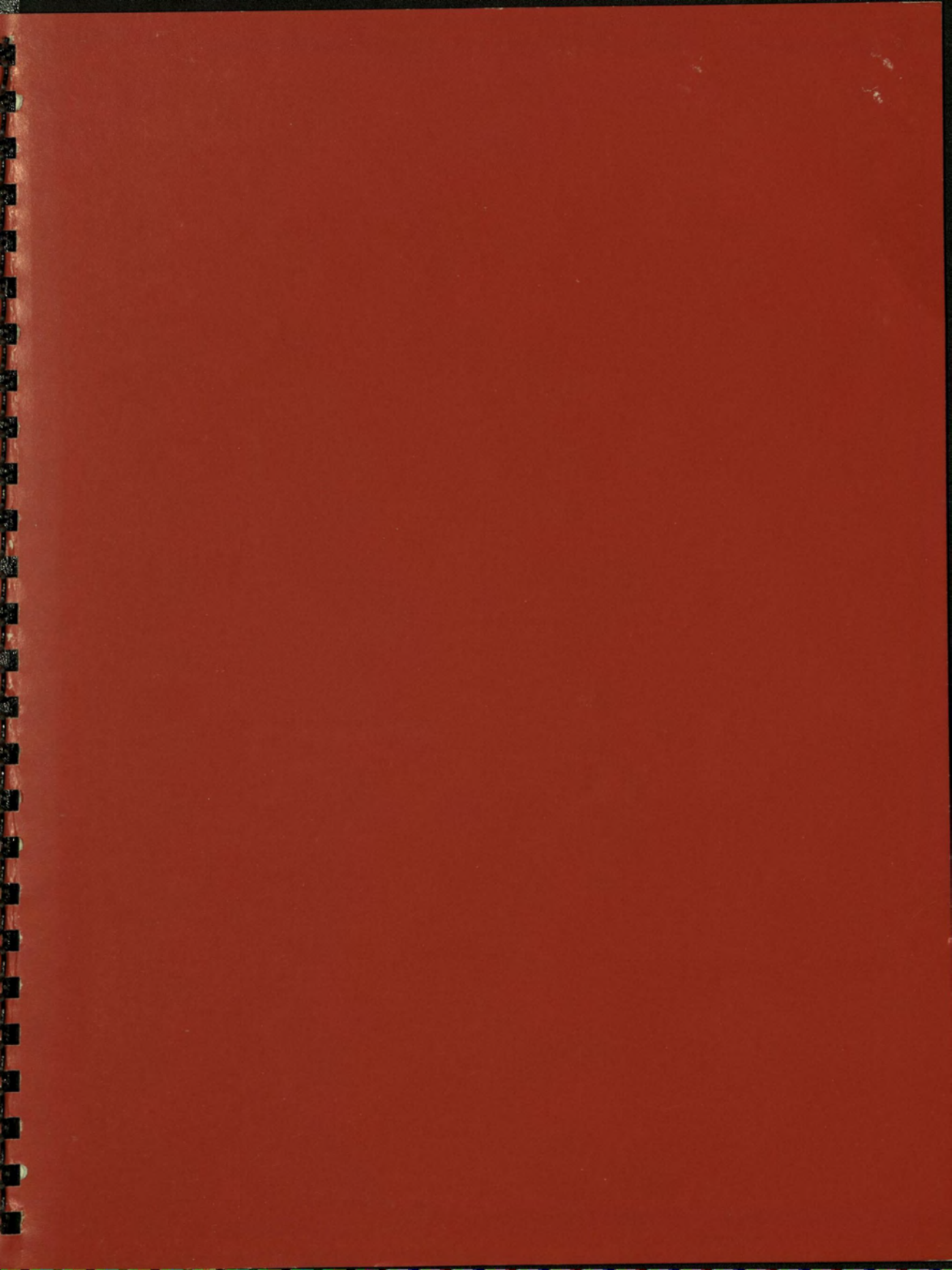
82705

AA PRO V449A

Professional 300 series -
CP/M-80 user's guide hard
disk system

SHREWSBURY LIBRARY
Digital Equipment Corporation
333 South Street SHR1-3/G18
Shrewsbury, MA 01545
(DTN) 237-3271

GAYLORD



digital



0211989

SHREWSBURY LIBRARY
DIGITAL EQUIPMENT CORPORATION
SHR1 3/G18
DTN 237-3400

Printed in U.S.A.
AA-V449A-TH



Professional™
300 series



Developer's Tool Kit

SHREWSBURY LIBRARY
DIGITAL EQUIPMENT CORPORATION
SHR1-3/G18
DTN: 237-3271

DECLIT may not be renewed. If you
need this for longer than one
month, please make a copy and send
the library copy back

DECLIT
AA
PRO
ED06A

digital
software

94 003/049/20

Positional Device Interface Programmer's Manual

Order No. AA-ED06A-TH

November 1985

This manual describes the Positional Device Interface (PDI), a kit that allows you to write applications that use devices such as mice, bitpads, and touch screens.

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

OPERATING SYSTEM: P/OS V3.0

digitalTM

DIGITAL EQUIPMENT CORPORATION
Maynard, Massachusetts 01754-2571

43624

First Printing, November 1985

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

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

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

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

Copyright © 1985 by Digital Equipment Corporation
All Rights Reserved

The following are trademarks of Digital Equipment Corporation:

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

CONTENTS

PREFACE	vii
CHAPTER 1	INTRODUCTION TO THE POSITIONAL DEVICE INTERFACE
1.1	COMPONENTS OF THE PDI KIT 1-2
1.1.1	The Device Driver 1-2
1.1.2	The Applications 1-2
1.1.3	The Positional Device Library (PDL) . . . 1-3
1.2	USING THE PDI KIT 1-3
1.2.1	Step 1: Install Applications 1-4
1.2.2	Step 2: Load Driver and Set Device Type . 1-4
1.2.3	Step 3: Connect Your Device 1-5
1.2.4	Step 4: Run the Test Application 1-5
1.2.5	Step 5: Run the Sketchpad Application . . 1-5
1.2.6	Step 6: Produce Your Own Application . . . 1-6
CHAPTER 2	DEVELOPING APPLICATIONS THAT USE POSITIONAL DEVICES
2.1	MAPPING DEVICE COORDINATE UNITS 2-1
2.1.1	Maintaining 1:1 Aspect Ratio 2-2
2.1.2	Mapping Coordinate Units to Terminal Screen 2-3
2.2	SPECIFYING DEVICE IDENTIFICATION 2-5
2.2.1	Device Class 2-5
2.2.2	Device Subclass 2-5
2.2.3	Port 2-6
2.2.4	Using Zero in the Device Identification . 2-9
2.3	HANDLING BUTTON DATA 2-10
2.4	CALLING THE LIBRARY ROUTINES 2-10
2.4.1	Using the PDL Global Entry Point 2-10
2.5	LINKING THE PROGRAM 2-11
2.5.1	Step 1: Modify Command File 2-11
2.5.2	Step 2: Modify Descriptor File (Optional) 2-12
2.6	RUNNING THE PROGRAM 2-13
2.6.1	If Program Uses Cluster Library 2-14
2.6.2	If Using Certain Graphics Software 2-14
CHAPTER 3	CALLING THE LIBRARY ROUTINES
3.1	ATTPD--ATTACH POSITIONAL DEVICE 3-2
3.2	CNMAST--CANCEL MOUSE AST 3-4
3.3	DETPD--DETACH POSITIONAL DEVICE 3-5
3.4	EVNCAN--CANCEL READ ON EVENT 3-6
3.5	GETDVC--GET DEVICE COORDINATES 3-7
3.6	GETNAM--GET DEVICE NAME 3-8
3.7	PDL--REQUEST PDL OPERATION 3-9

3.8	REDCNF--READ CONFIGURATION	3-11
3.9	REDEVN--READ ON EVENT	3-14
3.10	REDRPT--READ POSITIONAL REPORT	3-18
3.11	SETCHR--SET DEVICE CHARACTERISTICS	3-20
3.11.1	SETCHR BAUD--Set Baud Rate	3-23
3.11.2	SETCHR CLICK--Set Touchscreen Click	3-25
3.11.3	SETCHR DRATE--Set Device Data Rate	3-27
3.11.4	SETCHR IDA--Set Input Device Area	3-28
3.11.5	SETCHR IDAASP--Set Input Device Area/1:1 Aspect Ratio	3-30
3.11.6	SETCHR LNCHR--Set Serial Line Characteristics	3-32
3.11.7	SETCHR ORIG--Set Device Origin	3-34
3.11.8	SETCHR PORT--Set Device to Port	3-35
3.11.9	SETCHR PROD--Position a Relative-Oriented Device	3-36
3.11.10	SETCHR RESET--Reset Positional Device	3-37
3.11.11	SETCHR RESOL--Set Coordinate Resolution	3-38
3.11.12	SETCHR RPMOD--Set Report Mode	3-40
3.11.13	SETCHR SCALE--Set Touchscreen Scaling	3-42
3.12	SPMAST--SPECIFY MOUSE AST	3-43
3.13	WRTDEV--WRITE RAW DATA TO DEVICE	3-46

CHAPTER 4 SAMPLE PROGRAMS

4.1	FORTTRAN-77	4-1
4.2	PASCAL	4-4
4.3	BASIC-PLUS-2	4-7
4.4	MACRO-11	4-11

APPENDIX A DEVICES YOU CAN USE WITH THE PDI

A.1	SUMMAGRAPHS MM 961 AND MM 1201 DIGITIZERS	A-2
A.2	GTCO MICRO DIGI-PAD	A-3
A.3	SUMMAGRAPHS SUMMAMOUSE	A-3
A.4	MICROSOFT SERIAL MOUSE	A-4
A.5	DECTOUCH (VRTS1-A)	A-4
A.6	SEIKO DT-3100 TABLET	A-5
A.7	SUMMAGRAPHS BIT PAD ONE	A-6
A.8	SUMMAGRAPHS BIT PAD TWO	A-9

APPENDIX B USING THE SKETCHPAD DEMONSTRATION APPLICATION

B.1	THE SCREEN	B-2
B.2	THE COMMAND MENU	B-3
B.3	THE PALETTE	B-5

APPENDIX C

GLOSSARY

INDEX

FIGURES

2-1	Square Input Device Area	2-1
2-2	Summagraphics MM961 Input Device Area	2-2
2-3	Coordinate Units of Terminal Screen	2-3
2-4	Simple Coordinate Units Mapping	2-3
2-5	Alternative Coordinate Units Mapping	2-4
2-6	Device Identification Parameter	2-5
2-7	DECTouch Ports	2-7
2-8	Sample Command (.CMD) File	2-12
2-9	Sample Descriptor (.ODL) File	2-13
A-1	Wiring for the Seiko Tablet	A-5
B-1	Screen Display for Sketchpad Application	B-2

TABLES

2-1	Device Classes	2-6
2-2	Values for Devid Parameter When Using DECTouch Ports	2-7
2-3	Values for Devid Parameter When Using XK0: or TT2:	2-8
2-4	Combinations of Class, Subclass, and Port Using Zero	2-9
3-1	Format of GETDVC Buffer	3-7
3-2	Request Values for PDI Operations	3-10
3-3	Format of Configuration Data	3-11
3-4	Stack Values Upon AST Entry, REDEVN	3-17
3-5	Values for SETCHR Characteristics	3-22
3-6	Data Values for BAUD Characteristic (Serial Only)	3-24
3-7	Data Values for IDA Characteristic	3-29
3-8	Data Values for IDA Characteristic	3-30
3-9	Data Values for LNCHR Characteristic (Serial Only)	3-32
3-10	Data Values for ORIG Characteristic	3-34
3-11	Data Values for PROD Characteristic	3-36
3-12	Data Values for RESOL Characteristic	3-38
3-13	Data Values for RPMOD Characteristic	3-41
3-14	Stack Values Upon AST Entry, SPMASST	3-45
A-1	Bit Pad One Switch Settings for XK0: or TT2:	A-7
A-2	Bit Pad One Switch Settings for DECTouch Port	A-8
A-3	Bit Pad Two Switch Settings for Any Port	A-9
B-1	Sketchpad Commands from Command Menu	B-3

PREFACE

Document Objectives

After reading this manual, you will be able to use the Positional Device Interface kit to:

- Connect various positional devices to the Professional computer and install the software that drives them.
- Run several test programs to demonstrate the operation of a positional device.
- Write your own applications that use positional devices.

Intended Audience

You should be an experienced programmer who is familiar with the procedures described in the *Tool Kit User's Guide* and in your language-specific manuals for developing applications to run on the Professional computer.

Structure of This Document

This document contains the following chapters and appendixes:

Chapter 1, *Introduction to the Positional Device Interface*, provides general information about the kit. The chapter describes the components of the kit and gives a step-by-step description of how to install and test the required software.

Chapter 2, *Developing Applications that Use Positional Devices*, shows in detail how to write your application. Included are details on writing, linking, and running your program.

Chapter 3, *Calling the Library Routines*, is a reference chapter that describes each of the routines you can call from the PDI Library.

Chapter 4, *Sample Programs*, provides complete examples for each of the programming languages you can use.

Appendix A, *Devices You Can Use with the PDI*, presents basic information on how each device operates and describes how to connect each device to the Professional. The appendix also lists the names and addresses of the device manufacturers.

Appendix B, *Using the Sketchpad Demonstration Application*, shows how to use one of the demonstration applications that comes with the PDI kit.

Associated Documents

For general information on writing applications for the Professional computer, see the *Tool Kit User's Guide*.

For information on the DECTouch touch screen monitor, see the *PRO/DECTouch Software User's Guide* (and the *VRTS1-A Color/Touch Screen Monitor Installation/Owner's Guide*).

For specific information regarding the programming language you are using, refer to the following manuals:

- *BASIC-PLUS-2 Documentation Supplement* and the PDP-11 BASIC-PLUS-2 documentation set.
- *Professional Tool Kit FORTRAN-77 Installation Guide and Documentation Supplement* and the PDP-11 FORTRAN-77 documentation set.
- The Tool Kit PASCAL documentation set: *User's Guide*, *Language Reference Manual*, and *Installation Guide and Release Notes*.
- *PDP-11 MACRO-11 Language Reference Manual*

Acknowledgements and Disclaimer

Copyright of Atari Corporation:

ATARI

Registered trademarks of Summagraphics Corporation:

BIT PAD ONE
BIT PAD TWO
SUMMAMOUSE
MM (DIGITIZERS)

Registered trademarks of GTCO Corporation:

DIGI-PAD
MICRO DIGI-PAD

Registered trademark of The Microsoft Corporation:

MICROSOFT

This document describes several positional devices manufactured and distributed by independent vendors. These descriptions (in some cases including installation instructions) are provided for the reader's information and convenience only. They are based upon information provided by the vendors.

Digital Equipment Corporation assumes no responsibility for the accuracy of the descriptions, for any changes that vendors may make that affect the descriptions of their products, nor for the quality or performance of the products.

Further, Digital Equipment Corporation makes no representation that the use of these products with this software or with a Professional computer as described in this document will not infringe existing or future patent rights. Neither does Digital Equipment Corporation imply the grant of a license to make, use, or sell any of the products described herein.

Also, this document does not imply that the equipment as used with this software or with a Professional computer will meet any governmental regulations or laws under which the positional devices or connected equipment may fall.

Finally, by describing certain positional devices that can be used with this software or with a Professional computer, Digital Equipment Corporation does not imply that the described devices are the only positional devices that can be used.

For additional information on any of these devices, please contact the vendor directly.

Conventions and Terminology Used In This Document

The manual uses the following conventions and terminology:

Convention/Term	Meaning
[optional]	In a command line, square brackets indicate that the enclosed item is optional. In a file specification, square brackets are part of the required syntax.
UPPERCASE	Uppercase words and letters indicate that you should type the word or letter exactly as shown.
lowercase	Lowercase words and letters indicate that you should substitute a word or value of your own. Usually the lowercase word identifies the type of substitution required.
...	A horizontal ellipsis indicates that you can repeat the preceding item one or more times. For example: parameter [,parameter...]
.	A vertical ellipsis means that not all of the statements are shown.
red	Interactive input appears in red.
Tool Kit	This general term refers to the software you use to develop applications to run on a Professional computer.
Host Tool Kit	The Host Tool Kit is Tool Kit software that runs on a host computer, rather than on the Professional itself.
PRO/Tool Kit	The PRO/Tool Kit is the Tool Kit software that runs on the Professional computer.

Also, numeric values are decimal unless specified otherwise.

CHAPTER 1

INTRODUCTION TO THE POSITIONAL DEVICE INTERFACE

The Positional Device Interface (PDI) is software that enables you to write applications that use a mouse, digitizing tablet, touch screen, or other positional device. A positional device is hardware that you use for input. Its main feature is the ability to transmit information about location as input to the computer.

The PDI software operates with positional devices connected to the Professional's Communication Port or Printer Port. Additionally, the software supports devices connected to ports located on the DECTouch Touch Screen Monitor.

You can connect the following positional devices to the Communication Port, the Printer Port, or a DECTouch port:

- GTCO Digi-Pad 5
- GTCO Micro Digi-Pad
- Summagraphics MM 961
- Summagraphics MM 1201
- Summagraphics SummaMouse
- Microsoft Serial Mouse
- Seiko DT-3100 Tablet
- Summagraphics Bit Pad One and Bit Pad Two

In addition, PDI supports the following devices only when they are connected to ports located on the DECTouch monitor:

- LM200 Quadrature Mouse
- Atari(c)-compatible Joystick

Appendix A describes each of the devices that you can use with the PDI software.

The remainder of this chapter describes the components of PDI and presents an overview of how you use them.

1.1 COMPONENTS OF THE PDI KIT

PDI comes as a kit that consists of the following:

- A device driver
- Three applications
- A library of routines to access the driver

Note that the kit does not supply any positional devices. You must supply these yourself.

The following sections describe each component of the kit.

1.1.1 The Device Driver

The driver is called DTDRV.TSK. It comes with the operating system and is loaded by the PDI Setup Application.

1.1.2 The Applications

The PRODRIVERS diskette, distributed with P/OS, contains the following applications:

- **Positional Device Setup**

This application allows you to load the driver into memory and indicate which positional device you will connect. This application is described in Section 1.2.2.

COMPONENTS OF THE PDI KIT

- **Test the PDI**

This application simply attaches a positional device, then reads and displays data transmitted from that device. Section 1.2.4 describes Test the PDI.

- **PDI Sketchpad**

The Sketchpad is a sample application that allows you to use a positional device to draw simple pictures on the terminal screen. Sketchpad has a crosshair cursor that you can move across any of three screen areas: Command Menu, Drawing Area, and Color Palette. Appendix B describes the Sketchpad application.

1.1.3 The Positional Device Library (PDL)

The Positional Device Library (PDL) comes as a cluster library with the operating system, or as an object module with the Tool Kit. The library contains routines that call the PDI. These routines enable you to write applications that use positional devices.

You can call the library routines from the following Tool Kit programming languages:

- BASIC-PLUS-2
- FORTRAN-77
- PASCAL
- PDP-11 MACRO-11

For detailed, language-specific information, refer to the appropriate language manual listed in the Preface.

1.2 USING THE PDI KIT

To get started using the kit, perform the steps outlined in the following subsections.

USING THE PDI KIT

1.2.1 Step 1: Install Applications

Install the applications from the diskette PRODRIVERS, which comes with P/OS. Use the normal P/OS installation procedure.

1.2.2 Step 2: Load Driver and Set Device Type

To load the driver and set the device type, you must run the Positional Device Interface setup application. This application automatically loads the PDI driver if it has not previously been loaded. Once loaded, the PDI driver remains in memory until you reboot your Professional.

The system displays a *Positional Device Interface Setup* Menu. The options displayed on this menu are as follows:

- **Feature Selection**

This option provides the ability to set up your positional device configuration. For example, you can assign the current port (when DECTouch is not present), force the PDI system start up at boot time, change mouse scaling, and vary the PDI's button support. Note that if DECTouch is connected, you cannot assign the Communication Port or Printer Port as the current port.

- **DECTouch Port Setup**

With this option, you can assign different devices to the DECTouch ports.

- **Communication Port Setup**

When the Communication Port is enabled, you can use this option to assign different devices to the Communication Port. This option also allows you to set the default positional device for the Communication Port.

- **Printer Port Setup**

When the Printer Port is enabled, you can use this option to assign different devices to the Printer Port. This option also allows you to set the default positional device for the Printer Port.

USING THE PDI KIT

- **Reset PDI**

This option puts the PDI system into its default state.

Each submenu provides HELP.

To set the device type for the device you will be using, choose the appropriate option for your PDI configuration.

1.2.3 Step 3: Connect Your Device

You must connect your positional device to the desired port before running any application that uses a positional device. This allows the application to initialize the device with a startup sequence.

1.2.4 Step 4: Run the Test Application

The test application is designed to simply attach and read data from a positional device.

The application continuously updates a display showing:

- Coordinates (x and y) indicating the current location of the device
- Status of one button
- Device ID of the device reporting the data
- Status block returned from the driver

See Chapter 2 for descriptions of the status codes. A status code of 1 indicates success.

You can press any key to exit this application.

1.2.5 Step 5: Run the Sketchpad Application

Sketchpad is an application that illustrates some of the capabilities of positional devices when used with the Professional's graphics software. Appendix B describes how to use this application.

USING THE PDI KIT

1.2.6 Step 6: Produce Your Own Application

Using the library routines in the Positional Device Library, and either the PRO/Tool Kit or Host Tool Kit, you can write applications in any of the supported languages. This step is fully described in Chapter 2.

CHAPTER 2

DEVELOPING APPLICATIONS THAT USE POSITIONAL DEVICES

This chapter provides important information that you need to write an application that uses a positional device.

2.1 MAPPING DEVICE COORDINATE UNITS

During execution, your program reads data as input from a positional device. For each positional device, the PDI driver defines an area from which the device can send valid input. We call this area the *input device area*.

For example, the Microsoft Mouse has a square input device area that measures 4096 by 4096 units. That is, the driver reports movements of the mouse in 4096 equal units in each direction. Figure 2-1 illustrates this input device area.

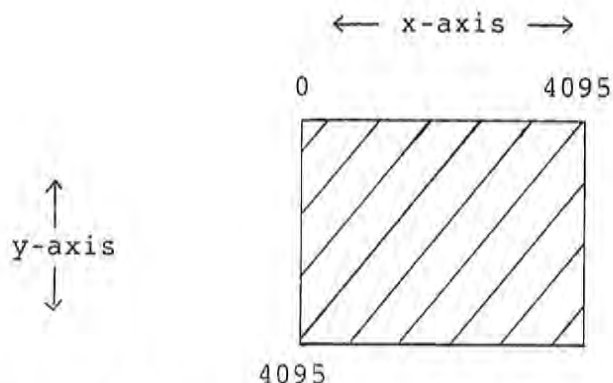


Figure 2-1: Square Input Device Area

MAPPING DEVICE COORDINATE UNITS

Not all input device areas are square, however. For example, PDI defines the input device area for the Summagraphics MM961 to be 4096 by 2560. This rectangle corresponds with the active area on the tablet. Figure 2-2 shows the input device area for the MM961. Points extending below +2559 on the y-axis are unused.

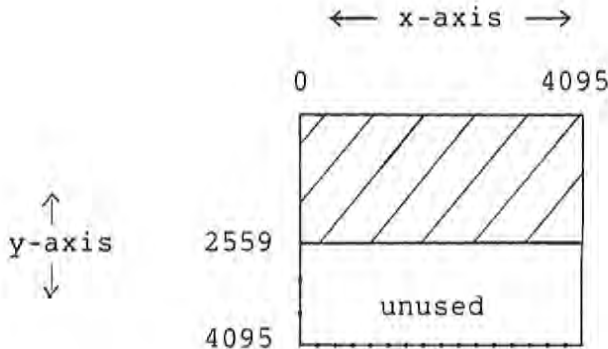


Figure 2-2: Summagraphics MM961 Input Device Area

2.1.1 Maintaining 1:1 Aspect Ratio

The aspect ratio of the input device area is a ratio between the size of the units on the x-axis and the size of the units on the y-axis. Maintaining an aspect ratio of 1:1 means that units on both axes are equal in size. Consequently, the positional device always reports the same number of units for the same distance moved in either direction.

If the movement of one inch along the x-axis equals ten units, then movement of one inch along the y-axis also equals ten units. The default settings for PDI guarantee that this is always true.

To maintain the 1:1 aspect ratio, PDI by default defines the length of the x axis to be 4096 units, and adjusts the length of the y-axis according to the proportions of the positional device's active area. If the input device area is square, then it measures 4096 by 4096. If the input device area is not square, then it measures 4096 by n , where n is calculated by PDI.

MAPPING DEVICE COORDINATE UNITS

2.1.2 Mapping Coordinate Units to Terminal Screen

Once your program has read input data from the input device area, it will likely transmit this data as output to the terminal screen. However, the coordinate units of the input device area and the terminal screen are different--the terminal screen measures 960 by 600 units. See Figure 2-3.

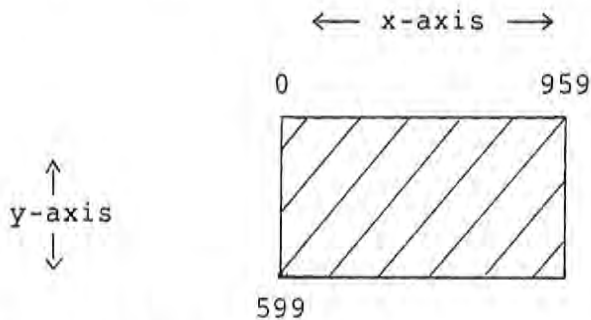


Figure 2-3: Coordinate Units of Terminal Screen

In order for your program to correctly map input data to the screen coordinates, you must change the coordinate units of the input device area. While doing so, you generally should maintain the 1:1 aspect ratio of the x- and y-coordinates.

Figure 2-4 illustrates how the coordinates of the terminal screen appear when mapped over a square input device area. Points below +599 in the input device area are unused.

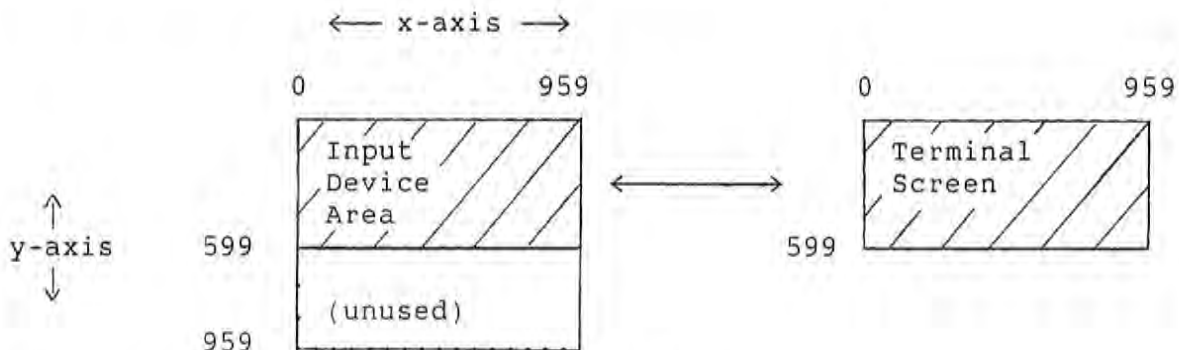


Figure 2-4: Simple Coordinate Units Mapping

MAPPING DEVICE COORDINATE UNITS

To perform the mapping in your program, call the routine SETCHR IDAASP. This routine changes the coordinates of the input device area. It allows you to specify the low and high values along the x-axis as well as the low value along the y-axis, while maintaining the 1:1 aspect ratio. PDI computes the high value of the y-axis.

For example, suppose that for the Microsoft Mouse you want to perform the terminal screen mapping. Pseudocode follows:

```

DECLARE    status(2),coordinate(3) INTEGER WORD ARRAY
           request
           INTEGER WORD

request = 9.                ! request IDAASP
coordinate(0) = 0           ! low bound of x-coordinate
coordinate(1) = 959        ! high bound of x-coordinate
coordinate(2) = 0           ! low bound of y-coordinate

CALL SETCHR (status,request,coordinate)
    
```

Because you can set the x- and y-coordinates separately, you do not have to set the same high and low bounds for each, as long as you maintain the 960:600 proportion of the terminal screen. For example, the following mapping would also work:

```

coordinate(0) = -300       ! low bound of x-coordinate
coordinate(1) = +659       ! high bound of x-coordinate
coordinate(2) = -300       ! low bound of y-coordinate

CALL SETCHR (status,request,coordinate)
    
```

Figure 2-5 shows this alternative mapping.

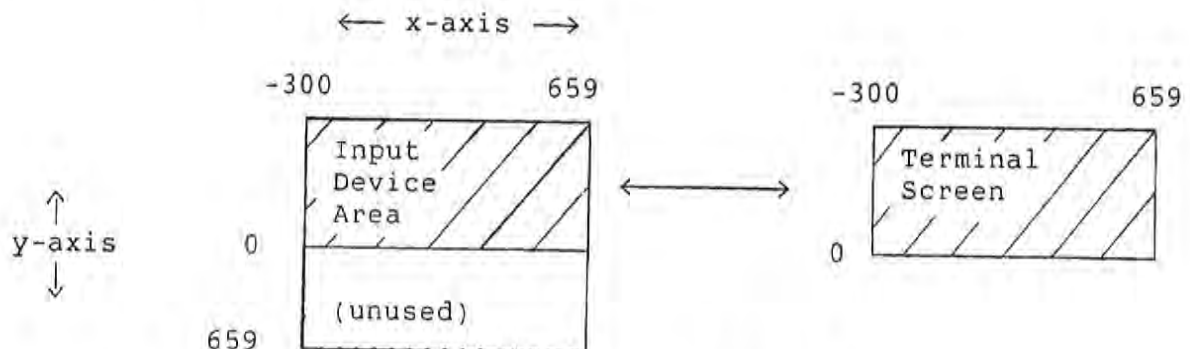


Figure 2-5: Alternative Coordinate Units Mapping

SPECIFYING DEVICE IDENTIFICATION

2.2 SPECIFYING DEVICE IDENTIFICATION

The device identification is an optional two-word parameter (devid) that appears in most of the routines. Its primary purpose is to allow you to specify the devices from which to read data. Also, the PDI returns the complete device identification value of an accessed device into the devid parameter, if supplied. You can obtain a list of all device identification values on the system by calling the REDCNF routine.

The devid parameter consists of three components: class, subclass, and port. Figure 2-6 illustrates these components.

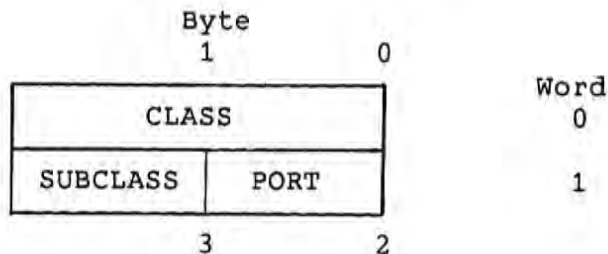


Figure 2-6: Device Identification Parameter

2.2.1 Device Class

Devices are grouped into classes. A device class is a set of similar devices, such as all mice, all keyboards, or all joysticks.

Use the device class alone to tell the PDI that you want reports from all connected devices of a similar type, such as all connected joysticks.

The first word of the devid parameter is a bit field indicating one of 16 possible device classes. Table 2-1 shows the classes and the decimal codes that provide the corresponding bit mask.

2.2.2 Device Subclass

The device subclass indicates a specific device; it further qualifies the class. For example, you can tell the PDI that you want reports from all connected GTCO Digi-Pad graphics tablets. Note that you must minimally use both the class and subclass to identify a specific device--a subclass alone is not unique.

SPECIFYING DEVICE IDENTIFICATION

Table 2-1: Device Classes

Class Code (Decimal)	Device Class
00	All device classes
01	Keyboard
02	Mouse
04	Graphics Tablet
08	Joystick
16	Touch screen

The subclass is an 8-bit integer located in the second word, high byte of the devid parameter. Tables 2-2 and 2-3 show the combinations of class, subclass, and port that you can use.

2.2.3 Port

The PDI assigns a port number to each connected device. If you know the port to which a device is connected, you can directly access that device by specifying the port number in the devid parameter. Determine which ports have devices connected by calling the REDCNF routine.

You can also use the port to distinguish between two devices having the same class and subclass. For example, if two identical joysticks are connected to the DECTouch monitor, distinguish between them using the port number.

Note that you cannot distinguish between the Communication Port and the Printer Port using the port number; the value is the same (2) for both ports. This is because you can only use one of these ports at a time.

The port is an 8-bit integer located in the second word, low byte of the devid parameter. Figure 2-7 illustrates the DECTouch ports and their port numbers.

SPECIFYING DEVICE IDENTIFICATION

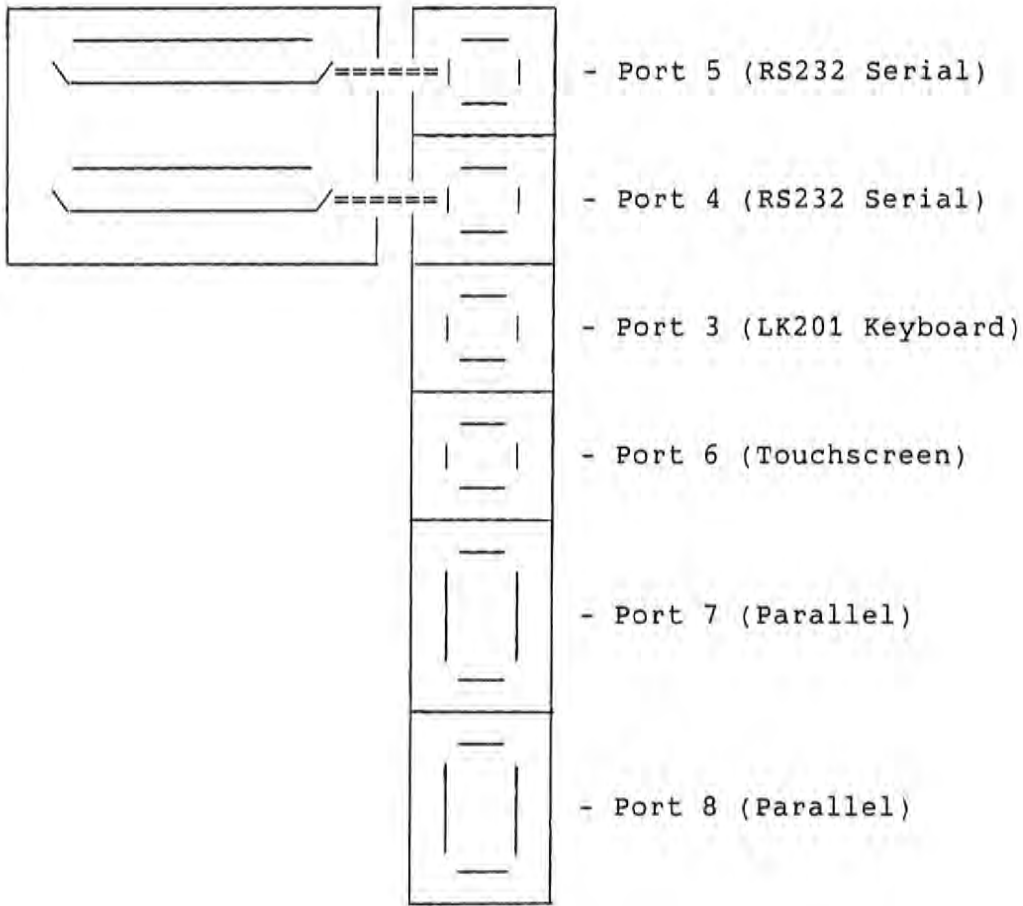


Figure 2-7: DECTouch Ports

Table 2-2 shows combinations of class, subclass, and port for each device that can be connected to a DECTouch port. All values shown are decimal.

Table 2-2: Values for Devid Parameter When Using DECTouch Ports

Class	Subclass	Port	Description
0	0	0	All connected devices (ATTPD and DETPD only)
1	1	3	LK201 keyboard

SPECIFYING DEVICE IDENTIFICATION

Class	Subclass	Port	Description
2	2	7 or 8	LM200 Quadrature Mouse
2	8	4 or 5	Summagraphics SummaMouse
2	4	4 or 5	Microsoft Mouse
4	1	4 or 5	Any device listed in Table 2-3.
8	1	7 or 8	Atari-compatible Joystick
16	1	6	DECtouch Touch Screen

Table 2-3 shows the combinations for each device that can be connected to the Communication Port or Printer Port. All values shown are decimal.

Table 2-3: Values for Devid Parameter When Using XK0: or TT2:

Class	Subclass	Port	Description
0	0	0	All connected devices (ATTPD and DETPD only)
1	1	3	LK201 keyboard
4	1	2	GTCO Digi-Pad 5
4	2	2	GTCO Micro Digi-Pad
4	4	2	Summagraphics MM 961
4	8	2	Summagraphics MM 1201
2	8	2	Summagraphics SummaMouse
2	4	2	Microsoft Mouse
4	16	2	Seiko DT-3100
4	32	2	Summagraphics Bit Pad One
4	64	2	Summagraphics Bit Pad Two

SPECIFYING DEVICE IDENTIFICATION

2.2.4 Using Zero in the Device Identification

You can specify zero in different combinations of class, subclass, and port. Table 2-4 shows how the PDI handles such combinations.

Table 2-4: Combinations of Class, Subclass, and Port Using Zero

Class	Subclass	Port	Result
c	s	p	This combination precisely locates a particular device on a particular port.
c	s	0	The PDI performs the operation on all devices matching the class/subclass combination.
c	0	0	The PDI performs the operation on all devices of the specified class.
0	0	p	The PDI performs the operation on whatever device is connected to the specified port.
0	0	0	For ATTPD, the PDI attaches all connected devices. For REDRPT, the PDI reads from the last attached device that reported a position. For REDEVN, the PDI reads from the first device that satisfies the request parameters. For all other operations, the PDI performs the operation on all attached devices.

Specifying zero for the devid parameter is equivalent to omitting the parameter altogether.

KEY

c = Valid class other than 0
s = Valid subclass other than 0
p = Valid port other than 0

HANDLING BUTTON DATA

2.3 HANDLING BUTTON DATA

A parameter on read operations allows you to obtain the status of up to 16 buttons on a positional device. A button can be either the kind of standard button you find on a mouse, or it can be some other switch-type mechanism. For example, with some bitpads you can order an optional stylus whose point retracts when depressed. This is considered a button.

The button status parameter is a one-word bit mask that reflects the current button status. Each bit set indicates that the corresponding button is down (switch closed). Each bit cleared indicates that the corresponding button is up (switch open).

2.4 CALLING THE LIBRARY ROUTINES

Your program must always attach the positional device before attempting to read data from it. Then, to read the input data from the device, your program must call either the REDRPT or REDEVN routine.

After all I/O operations with the device are complete, you can detach the device either explicitly by calling the DETPD routine, or implicitly by terminating your task. The operating system automatically detaches a device upon terminating the task that attached the device.

2.4.1 Using the PDL Global Entry Point

You can make any request to the library through the PDL global entry point, rather than calling each routine individually. The PDL entry point is similar to the CGL entry point used to make requests to the Core Graphics Library.

For some programs, it is advantageous to use the technique of making requests through a single global entry point. For clarity, you should always use one technique or another--not both--in the same program.

See Chapter 3, Section 3.7 for a detailed description of the PDL entry point.

LINKING THE PROGRAM

2.5 LINKING THE PROGRAM

Linking programs that use the PDL routines requires several steps, as described in the following sections.

2.5.1 Step 1: Modify Command File

You must modify the command (.CMD) file that you submit to PAB. Figure 2-8 shows a sample command file for a BASIC-PLUS-2 program. Numbers in parentheses in the left margin show the changed lines, and correspond to the numbered descriptions that follow:

1. UNITS option: Increase by 1 the number of LUNs (logical unit numbers) available to your program. The extra lun is for use by the positional device. The number of LUNs in the illustrated command file was increased from 18 to 19.
2. CLSTR option: Define the library PDL as a cluster library to be linked with your program. In Figure 2-8, PDL was added to the CLSTR option.

NOTE

Omit this step if you want to use the object library instead of the cluster library. Section 2.5.2 describes how to use the object library.

3. GBLDEF option: Assign the unused LUN that you added in the UNITS option to the symbol PD\$LUN. Your program uses this LUN for positional device I/O. Note that the value 23 (octal) in the GBLDEF option is equivalent to LUN number 19 (decimal).
4. GBLDEF option: Assign an unused event flag number (EFN) to the symbol PD\$EFN. Your program needs this event flag to synchronize all positional device operations. In the figure, the EFN 2 was not used for any other EFN, so we assigned it to PD\$EFN.

LINKING THE PROGRAM

```
SY:MOUSE/FP/CP=SY:MOUSE/MP
TASK=MOUSE
(1) UNITS = 19
    ASG = SY:5:6:7:8:9:10:11:12
    ASG = TI:13:15
    EXTTSK= 952
(2) CLSTR=PBFSML,PDL,CGLFPU,RMSRES,POSRES:RO
    ;
    ; DEFINE BUFFER SIZES
    EXTSCT = DM$BUF:4540 ; dynamic single choice menu
    EXTSCT = FL$BUF:4310 ; file selection/specification
    EXTSCT = HL$BUF:3500 ; help text/menu
    EXTSCT = MM$BUF:1000 ; multi-screen menu
    EXTSCT = MN$BUF:4540 ; static single choice menu
    ;
    ; DEFINE LUN ASSIGNMENTS
    GBLDEF = HL$LUN:21 ; help frame file
    GBLDEF = MN$LUN:20 ; menu frame file
    GBLDEF = MS$LUN:16 ; message frame file
    GBLDEF = TT$EFN:1 ; terminal I/O event flag
    GBLDEF = TT$LUN:15 ; terminal I/O
    GBLDEF = WC$LUN:22 ; directory searches for OLDFIL
    ; and NEWFIL routines and
    ; callable print services
    GBLDEF = G$LUN:17 ; for Core Graphics Library
    GBLDEF = G$EFN:3 ; for Core Graphics Library
(3) GBLDEF = PD$LUN:23 ; PDL device I/O
(4) GBLDEF = PD$EFN:2 ; PDL I/O event flag
//
```

Figure 2-8: Sample Command (.CMD) File

2.5.2 Step 2: Modify Descriptor File (Optional)

This step is optional and you should perform it only if your program does not need to run on future versions of P/OS.

NOTE

For most applications, we recommend that you omit this step so as to maintain compatibility with future P/OS releases.

LINKING THE PROGRAM

You perform this step in order to link your program with a PDL object module rather than a cluster library. Note that using a cluster library ensures that your program will not need to be relinked upon release of a new version of P/OS. The operating system always contains the most recent cluster libraries.

However, a drawback to using a cluster library is the loss of performance due to mapping and unmapping of the cluster library. By linking an object module directly into the application address space, you avoid this performance loss.

To link the object module with the application, first modify the build command file as described in Section 2.5.1, but do not add "PDL" to the CLSTR option as described in that Section.

Next, modify the overlay descriptor file to include PDLOBJ.OBJ as a segment in the application. Figure 2-9 shows a sample overlay descriptor file for use with a BASIC-PLUS-2 program. The line indicated by a (1) in the left margin shows the addition of "LB:[1,5]PDLOBJ", surrounded by hyphens.

```
(1)      .ROOT BASIC2-RMSROT-USER,RMSALL
        USER:  .FCTR SY:TEST-LB:[1,5]PDLOBJ-LIBR
        LIBR:   .FCTR LB:[1,5]PBFOTS/LB
        @LB:[1,5]PBFIC1
        @LB:[1,5]RMSRLX
        .END
```

Figure 2-9: Sample Descriptor (.ODL) File

PAB resolves references to the object library by searching in LB:[1,5] for PDLOBJ.OBJ.

2.6 RUNNING THE PROGRAM

In order to run your program successfully, you must load the appropriate driver. If you have not yet done so, load a driver as described in Section 1.2.2.

RUNNING THE PROGRAM

2.6.1 If Program Uses Cluster Library

If you are using the cluster library version of the PDI Library, you must install the cluster library on the Professional before running your program. In A .INS form of the application installation command file, insert the command:

```
INSTALL [ZZSYS]PDL.TSK/LIBRARY
```

In a .INB file, you must use the /CLUSTER qualifier on the INSTALL command:

```
INSTALL [ZZSYS]PDL.TSK/LIBRARY/CLUSTER
```

If you are executing your program from DCL instead, enter the following command:

```
INSTALL LB:[ZZSYS]PDL.TSK/READ
```

2.6.2 If Using Certain Graphics Software

Tool Kit graphics software that you might use likely has its own requirements. For example, for programs that call CGL routines, you must install [ZZSYS]CGLFPU on the target Professional before running the program. In the application installation command file insert the line:

```
INSTALL [ZZSYS]CGLFPU.TSK/LIBRARY
```

Or, from DCL, first enter the command:

```
$ INSTALL LB:[ZZSYS]CGLFPU/READ
```

See the *Core Graphics Library Manual* for details.

CHAPTER 3

CALLING THE LIBRARY ROUTINES

The standard PDP-11 R5 parameter passing mechanism is the calling method for all of the routines in the PDI Library. Upon entry into a routine, R5 contains the address of a parameter block.

Also, each value in a parameter block contains the address of the variable, rather than containing the actual value of the variable. Consequently, all of the subroutines indirectly refer to the values passed to the calling routines.

This chapter describes the PDI routines in alphabetical order.

ATTPD--ATTACH POSITIONAL DEVICE

3.1 ATTPD--ATTACH POSITIONAL DEVICE

Attach your task to the currently connected positional device.

Format

CALL ATTPD (status [,devid])

Where:

status A two-word decimal integer array whose first word receives the status return of the call. The received value can be one of the following decimal integers:

IS.SUC	+01	Call completed successfully.
IE.BAD	-01	Invalid format for parameter block. See MACRO-11 examples for correct format.
IE.ONP	-05	Invalid subfunction. You have not loaded the PDI driver.
IE.DAA	-08	Device already attached by another task. Other task must first detach.
IE.DUN	-09	Device not attachable. Communication Port or Printer Port is currently busy.
IE.FHE	-59	Fatal hardware error while performing operation. Often occurs when device is not physically attached.
IE.TMO	-75	Timeout error. the DTM did not acknowledge the request within two seconds.
(none)	-510	Attach failed due to device handler error.
(none)	-511	DECTouch driver not active.
(none)	-520	Invalid devid parameter. See Tables 2-2 and 2-3.
(none)	-521	
(none)	-522	
(none)	-523	
(none)	-524	

devid The address of a two-word device identification number. The default devid is 00. See Section 2.2 for details.

ATTPD--ATTACH POSITIONAL DEVICE

Description

This routine logically connects a positional device or group of devices to the interface. Once attached, a positional device is ready to transmit data. You can call this routine any number of times.

Under the following conditions, the system returns status -09, "Device not attachable," upon invoking ATTPD:

- You have attempted to attach the device via the Communication Port (XK0:), but a Communication Service such as file transfer or terminal emulation is active, or the Communication Port is otherwise attached.
- You have attempted to attach the device via the Printer Port, but a Print Service request is active, or the Printer Port (TT2:) is otherwise attached.

You might also receive status -09 if you incorrectly linked your program with the PDI Library.

If the Communication Port has a modem connection, but there is no activity on the line, the modem connection is broken. The PDI attach is successful.

3.2 CNMAST--CANCEL MOUSE AST

Disable an AST previously set by the SPMAST routine.

Format

CALL CNMAST (status [,devid])

Where:

status A two-word decimal integer array whose first word receives the status return of the call. The received value is always the following decimal integer:

IS.SUC +01 Call completed successfully.

devid The address of a two-word device identification number. The default devid is 00. See Section 2.2 for details.

Description

This routine disables an asynchronous system trap (AST) set by the SPMAST routine for the specified device(s).

DETPD--DETACH POSITIONAL DEVICE

3.3 DETPD--DETACH POSITIONAL DEVICE

Detach your task from the currently attached positional device.

Format

CALL DETPD (status [,devid])

Where:

status A two-word decimal integer array whose first word receives the status return of the call. The received value can be one of the following decimal integers:

IS.SUC	+01	Call completed successfully.
IE.BAD	-01	Invalid format for parameter block. See the MACRO-11 examples for correct format.
IE.DNA	-07	Device not attached. You cannot detach a device that is not attached.
IE.TMO	-75	Timeout error. the DTM did not acknowledge the request within two seconds.
(none)	-520	Invalid devid parameter. See Tables 2-2 and 2-3.
(none)	-521	
(none)	-522	
(none)	-523	
(none)	-524	

devid The address of a two-word device identification number. The default devid is 00. See Section 2.2 for details.

Description

This routine explicitly detaches one or more positional devices. That is, it logically disconnects your task from the device(s). You can also implicitly detach all positional devices by terminating your task.

NOTE

Detaching a device connected to the Printer Port or Communication Port allows normal operation of the port.

EVNCAN--CANCEL READ ON EVENT

3.4 EVNCAN--CANCEL READ ON EVENT

Cancel any pending Read on Event calls.

Format

CALL EVNCAN (status)

Where:

status A two-word decimal integer array whose first word receives the status return of the call. The received value is always the following decimal integer:

IS.SUC +01 Call completed successfully.

Description

This routine cancels any pending calls to the REDEVN routine. The Read on Event operation terminates immediately with IE.ABO as the status.

GETDVC--GET DEVICE COORDINATES

3.5 GETDVC--GET DEVICE COORDINATES

Get the input device area coordinates.

Format

CALL GETDVC (status, buff, devid)

Where:

status A two-word decimal integer array whose first word receives the status return of the call. The received value can be one of the following decimal integers:

IS.SUC +001 Call completed successfully.

(none) -520 Invalid devid parameter. See Tables 2-2 and 2-3.

buff A four-word integer array that receives the input device area coordinates. Table 3-1 shows the format of the array.

devid The address of a two-word device identification number. The default devid is 00. See Section 2.2 for details. The devid parameter must specify a unique device.

Description

This routine is the converse of IDA and IDAASP; rather than setting the input device area coordinates, it returns them.

Table 3-1: Format of GETDVC Buffer

Word	Value
0	x minimum (default = 0)
1	x maximum (default = 4095)
2	y minimum (default = 0)
3	y maximum (default = 4095)

GETNAM--GET DEVICE NAME

3.6 GETNAM--GET DEVICE NAME

Get the name of a device with the specified device ID.

Format

CALL GETNAM (status, buff, devid)

Where:

status A two-word decimal integer array whose first word receives the status return of the call. The received value can be one of the following decimal integers:

IS.SUC +01 Call completed successfully.

IE.BAD -01 Invalid format for parameter block.

buff A 16-byte ASCII string in which GETNAM returns the device name.

devid The address of a two-word device identification number. The default devid is 00. See Section 2.2 for details. This routine returns the name of the device you specify in the devid parameter. The devid parameter must specify a unique device.

Description

This routine returns a 16-byte ASCII name corresponding to the device ID. The name is left-justified. If the name is less than 16 bytes, it is terminated by a null byte (0). If the routine cannot find the device name, it returns the string value "No_Device".

3.7 PDL--REQUEST PDL OPERATION

Request a positional device library operation using the PDL global entry point.

Format

CALL PDL (reqnum, status, [p1,...])

Where:

reqnum A one-word decimal integer indicating the operation that you want to perform. Table 3-2 lists the valid values.

status A two-word decimal integer array whose first word receives the status return of the call. The received value can be one of the following decimal integers:

IS.SUC +01 Call completed successfully.

IE.BAD -01 Invalid format for parameter block.

p1,... Are the actual parameters required by the particular request. Each routine description in this chapter describes the parameters.

Description

PDL is a global entry point through which you can request any operation. It is similar in operation to the CGL entry point used in the Core Graphics Library.

PDL--REQUEST PDL OPERATION

Table 3-2: Request Values for PDI Operations

Request	Number (Decimal)	Description
ATTPD	01	Attach Positional Device
CNMAST	12	Cancel Mouse AST
DETPD	02	Detach Positional Device
EVNCAN	14	Cancel Read on Event
GETDVC	15	Get Input Device Area Coordinates
GETNAM	13	Get Device Name
REDCNF	05	Read Configuration
REDDEV	07	Read Raw Data from Device
REDEVN	04	Read on Event
REDRPT	03	Read Positional Report
SETCHR	08	Set Device Characteristics
SPMAST	11	Specify Mouse AST
WRTDEV	06	Write Raw Data to Device

NOTE

Request values 09 and 10 are reserved.

REDCNF--READ CONFIGURATION

3.8 REDCNF--READ CONFIGURATION

Get the configuration of a connected positional device.

Format

CALL REDCNF (status, buff)

Where:

status A two-word decimal integer array whose first word receives the status return of the call. The received value can be one of the following decimal integers:

IS.SUC	+01	Call completed successfully.
IE.BAD	-01	Invalid format for parameter block.
IE.TMO	-75	Timeout occurred. Either the device or the DECTouch module did not acknowledge the request within 2 seconds.

buff A 20-word data buffer to receive the configuration data. Table 3-3 shows the format of this data.

See Section 2.2 for information on device identification values.

Table 3-3: Format of Configuration Data

Word	Description
0	This word can have the following decimal values: <ul style="list-style-type: none">● 36 = No device connected.● 37 = Only the LK201 is connected.● 44 = Printer Port connected, LK201 disabled.● 45 = Printer Port connected, LK201 connected.

REDCNF--READ CONFIGURATION

Word	Description
0	<ul style="list-style-type: none">● 52 = Communication Port connected, LK201 disabled.● 53 = Communication Port connected, LK201 connected.● 60 = DECTouch present, LK201 disabled.*● 61 = DECTouch present, LK201 connected.
1	If DECTouch present: Control Module status (normally zero). If DECTouch not present: zero.
2	Reserved
3	Reserved
4	Port 2 device class ID--serial port (Communication or Printer), valid only if DECTouch not present.
5	Port 2 device subclass ID--serial port (Communication or Printer), valid only if DECTouch not present.
6	Port 3 device class ID--always LK201
7	Port 3 device subclass ID--always LK201
8	Port 4 device class ID
9	Port 4 device subclass ID
10	Port 5 device class ID
11	Port 5 device subclass ID
12	Port 6 device class ID
13	Port 6 device subclass ID
14	Port 7 device class ID

* REDCNF reports that the LK201 is disabled either when the cable is physically unconnected or when LK201 input is disabled from the PDI setup application.

REDCNF--READ CONFIGURATION

Word	Description
15	Port 7 device class ID
16	Port 8 device class ID
17	Port 8 device subclass ID
18	Reserved
19	Reserved

3.9 REDEVN--READ ON EVENT

Read report when a specified event occurs.

Format

CALL REDEVN (status, xcoor, ycoor, button, [devid], [butmsk],
[tmo], [evtflg], [astadd], [xinc], [yinc])

Where:

status A one-word decimal integer that receives the status return of the call. The received value can be one of the following decimal integers:

IS.SUC	+01	Success, button changed state
(none)	+02	Success, x increment satisfied
(none)	+03	Success, y increment satisfied
(none)	+04	Success, timeout occurred
IE.ABO	-15	Request terminated. This value is returned when you abort REDEVN by calling the EVNCAN routine.
IE.BAD	-01	Invalid format for parameter block. Use the correct R5 calling format.
IE.DNA	-07	Device not attached. You cannot read input data from a device that is not attached.
IE.FHE	-59	Fatal hardware error while performing operation. Often occurs when device is not physically attached.
(none)	-520	Invalid devid parameter. See Tables 2-2 and 2-3.
(none)	-521	
(none)	-522	
(none)	-523	
(none)	-524	

xcoor The x-coordinate value word, an integer value whose possible values depend on the current input device area and on the report mode (absolute, relative, or device physical, see Section 3.11.12). By default, the possible values of xcoor are between 0 and 4095, inclusive.

REDEVN--READ ON EVENT

- ycoor** The y-coordinate value word, an integer value whose possible values depend on the current input device area and on the report mode (absolute, relative, or device physical, see Section 3.11.12). By default, the possible values of ycoor are between 0 and 4095, inclusive.
- button** The button status, a one-word bit mask identifying up to 16 button states. For each bit, 0 is button up (switch open), and 1 is button down (switch closed).
- devid** The address of a two-word device identification used for input to and output from the call. The default devid is 00. See the description and Section 2.2 for details.
- butmsk** The button mask, a one-word integer specifying the button events that will trigger completion of the read operation.
- tmo** A one-word integer specifying a timeout value in seconds that will trigger completion of the read operation. Possible values are 0 to 255.
- evtflg** A one-word integer specifying the event flag to be set upon completion of the read operation. A value of -1 sets the event flag PD\$EFN, specified in the task build command file. A positive value indicates the explicit number of the flag you want to set. A value of zero sets no event flag.
- astadd** A one-word integer specifying the address of an AST routine that the system will call upon completion of the read operation. A value of 0 for this parameter specifies that the system will not attempt to call an AST routine.
- xinc** A one-word positive integer value specifying the amount of change in the x-coordinate value to trigger completion of the read operation. If the value is 0, a change in the x-coordinate value has no effect.
- yinc** A one-word positive integer value specifying the amount of change in the y-coordinate value to trigger completion of the read operation. If the value is 0, a change in the y-coordinate value has no effect.

REDEVN--READ ON EVENT

Description

This routine performs a conditional read operation. It returns a positional report to the calling task only when one of the events you specify occurs. The routine performs a logical OR operation on the specified events.

The x- and y-coordinate values reflect the current input device coordinates.

NOTE

The x- and y-coordinate values are affected by the IDA, ORIG, RESOL, and RPMOD characteristics, which you can set with the SETCHR routine. See Section 3.11 for details.

Button events occur for both down and up states for the buttons specified in the butmsk parameter.

To perform the conditional read asynchronously, you supply a value for the evtflg parameter.

When you specify a value for the astadd parameter, you must ensure that the following parameters in the call are contiguous:

- status (one word)
- xcoor (one word)
- ycoor (one word)
- button (one word)
- devid (two words)

These parameters must always appear in the order shown.

Upon entry to the AST routine, the user stack contains the values shown in Table 3-4.

The device identification (devid) identifies one of several devices that you can connect to a DECTouch port. If you specify a devid other than 00, the PDI performs the operation on the specified device. If you specify a devid of 00, the PDI performs the operation on any and all devices connected.

REDEVN--READ ON EVENT

In all cases, the routine always returns the device identification of the device reporting data into the devid parameter, if supplied. You should reset the devid to zero each time you call REDEVN when specifying an explicit devid of 00.

NOTE

You must remove the completion cause word (top word) from the stack prior to exiting the AST routine.

Table 3-4: Stack Values Upon AST Entry, REDEVN

Current Stack Pointer	Contents										
SP+10	Event flag mask word										
SP+06	PS of task prior to AST										
SP+04	PC of task prior to AST										
SP+02	Task's Directive Status Word										
SP+00	Cause of completion:										
	<table border="1"><thead><tr><th>Value</th><th>Cause</th></tr></thead><tbody><tr><td>01</td><td>Button changed state</td></tr><tr><td>02</td><td>X increment satisfied</td></tr><tr><td>03</td><td>Y increment satisfied</td></tr><tr><td>04</td><td>Timeout</td></tr></tbody></table>	Value	Cause	01	Button changed state	02	X increment satisfied	03	Y increment satisfied	04	Timeout
Value	Cause										
01	Button changed state										
02	X increment satisfied										
03	Y increment satisfied										
04	Timeout										

3.10 REDRPT--READ POSITIONAL REPORT

Read input data from the positional device.

Format

CALL REDRPT (status, xcoor, ycoor, button [,devid])

Where:

status A two-word decimal integer array whose first word receives the status return of the call. The received value can be one of the following decimal integers:

IS.SUC	+01	Call completed successfully.
IE.BAD	-01	Invalid format for parameter block. Use the correct R5 calling format.
IE.DNA	-07	Device not attached. You cannot read input data from a device that is not attached.
IE.FHE	-59	Fatal hardware error while performing operation. Often occurs when device is not physically attached.
(none)	-520	Invalid devid parameter. See Tables 2-2 and 2-3.
(none)	-521	
(none)	-522	
(none)	-523	
(none)	-524	

xcoor The x-coordinate value word, an integer value whose possible values depend on the current input device area and on the report mode (absolute, relative, or device physical, see Section 3.11.12). By default, the possible values of xcoor are between 0 and 4095, inclusive.

ycoor The y-coordinate value word, an integer value whose possible values depend on the current input device area and on the report mode (absolute, relative, or device physical, see Section 3.11.12). By default, the possible values of ycoor are between 0 and 4095, inclusive.

button The button status, a one-word bit mask identifying up to 16 button states. For each bit, 0 is button up (switch open), and 1 is button down (switch closed).

REDRPT--READ POSITIONAL REPORT

devid The address of a two-word device identification used for input to and output from the call. The default devid is 00. See the description and Section 2.2 for details.

Description

This library call returns a positional report to the calling task, reflecting the current position of the active positional device. The report contains one word each for x- and y-coordinate values, and one word for button status.

The x- and y-coordinate values reflect the current input device coordinates.

NOTE

The x- and y-coordinate values are affected by the IDA, ORIG, RESOL, and RPMOD characteristics, which you can set with the SETCHR routine. See Section 3.11 for details.

SETCHR--SET DEVICE CHARACTERISTICS

3.11 SETCHR--SET DEVICE CHARACTERISTICS

Set the characteristics of an attached positional device.

Format

CALL SETCHR (status, chr, data [,devid])

Where:

status A two-word decimal integer array whose first word receives the status return of the call. The received value can be one of the following decimal integers:

IS.SUC	+01	Call completed successfully.
(none)	-520	Invalid devid parameter. See Tables 2-2 and 2-3.
(none)	-521	
(none)	-522	
(none)	-523	
(none)	-524	

chr A one-word integer value specifying the characteristic to set. Table 3-5 shows the possible values for this parameter.

data The address of a data block containing the parameters for the particular characteristic you are setting. The description of each characteristic provides the format and size of the block.

devid The address of a two-word device identification used for input to and output from the call. The default devid is 00. See the description and Section 2.2 for details.

Description

There are two classes of device characteristics:

- **Device Independent**

Device-independent characteristics are applicable to any positional device. You use them mainly to set the format of the coordinate data returned to your application.

SETCHR--SET DEVICE CHARACTERISTICS

• Device Specific

Device-specific characteristics, code values 65 to 67 (decimal), can be altered for some types of positional devices, but not necessarily all of them. The PDI handles such characteristics differently for each device.

You should consult the documentation for the particular device to make sure that you can alter the desired characteristic.

If you attempt to set a device-specific characteristic that is unsupported for a device or device group, the PDI returns the conditional success code 2, indicating that it ignored the SETCHR call for that particular device.

The device identification (devid) identifies one of several devices that you can connect to a DECTouch port. If you specify a devid other than 00, the PDI performs the operation on the specified device. If you specify a devid of 00, the PDI performs the operation on any and all devices connected.

For device-specific characteristics, always specify a unique device ID in the devid parameter.

Table 3-5 lists the characteristics you can set. Sections following the table describe the characteristics in alphabetical order.

NOTE

We suggest you use the names shown for the characteristics if you are creating your own symbols.

SETCHR--SET DEVICE CHARACTERISTICS

Table 3-5: Values for SETCHR Characteristics

Decimal Value	Characteristic
1	RESET--Reset Positional Device
2	RPMOD--Set Report Mode
3	IDA--Set Input Device Area
4	RESOL--Set Coordinate Resolution
5	ORIG--Set Device Origin
6	PORT--Set Device to Port
8	PROD--Position Relative-Oriented Device
9	IDAASP--Set Input Device Area with 1:1 Aspect Ratio
65	BAUD--Set Baud Rate for Device
66	LNCHR--Set Serial Line Characteristics
67	DRATE--Set Device Data Rate
128	SCALE--Set Screen Scaling
129	CLICK--Set Touchscreen Click

SETCHR--SET DEVICE CHARACTERISTICS

3.11.1 SETCHR BAUD--Set Baud Rate

Characteristic Number: 65

Data Value: 1-Word Integer

Description

This characteristic is device-specific. You must specify a unique device ID in the devid parameter of the call.

The BAUD characteristic allows you to change the data transmission rate between the positional device and the port to which it is connected.

You can set the baud rate for serial ports only. The serial ports are DECTouch ports 4 and 5, and the Professional's Communication and Printer ports, both port number 2.

Table 3-6 shows the values for the data parameter in the SETCHR call and their associated baud rates.

Status Returns

- +1 Success.
- +2 Function not supported.
- 500 One or more data values are out of range.

SETCHR--SET DEVICE CHARACTERISTICS

Table 3-6: Data Values for BAUD Characteristic (Serial Only)

Data Value (Decimal)	Baud Rate
1	50
2	75
3	110
4	300
5	600
6	1200
7	2400
8	4800
9	9600

SETCHR--SET DEVICE CHARACTERISTICS

3.11.2 SETCHR CLICK--Set Touchscreen Click

Characteristic Number: 129

Data Value: 1 Byte

Description

This characteristic is device-specific. You must specify a unique device ID in the devid parameter of the call.

NOTE

This characteristic applies only to the DECTouch monitor. For other devices, the PDI ignores this characteristic and performs no operation.

Use CLICK to set or change the way the DECTouch monitor produces sounds during user operations. The data parameter is a byte containing bit fields that you can set or clear to specify how DECTouch produces the sounds.

The format of the eight bits of the data value follows:

xxcrfnnn

nnn Three bits indicating the volume of the output sound. The possible bit combinations are:

- 000--No change
- 001--Turn sound off
- 010--Low volume
- 011--Medium volume
- 100--High volume

- f If set, DECTouch produces sound on the first touch.
- r If set, DECTouch produces sound when the touch is released.
- c If set, DECTouch produces a beep sound. If clear, DECTouch produces a click tone.
- xx These two bits are reserved.

SETCHR--SET DEVICE CHARACTERISTICS

Status Returns

- +1 Success.
- +2 Function not supported.
- 500 One or more data values are out of range.
- 520 Invalid devid parameter. See Table 2-2.

SETCHR--SET DEVICE CHARACTERISTICS

3.11.3 SETCHR DRATE--Set Device Data Rate

Characteristic Number: 67

Data Value: 1-Word Integer

Description

This characteristic is device-specific. You must specify a unique device ID in the devid parameter of the call.

NOTE

This characteristic applies to devices connected to DECTouch ports 6, 7, and 8 only. For devices connected to any other ports, the PDI ignores this characteristic and performs no operation.

The DRATE characteristic sets the rate at which a device sends the x,y-coordinate pairs. The device data rate is independent of the baud rate; however, the baud rate can affect the ceiling at which coordinate pairs can be generated.

You specify the data rate in coordinate pairs generated per second. The actual data rate can vary from the value specified, but the PDI approximates as close as possible.

Status Returns

- +1 Success.
- +2 Function not supported.
- 500 One or more data values are out of range.
- 520 Invalid devid parameter. See Table 2-2.

SETCHR--SET DEVICE CHARACTERISTICS

3.11.4 SETCHR IDA--Set Input Device Area

Characteristic Number: 3

Data Value: 4-Word Integer Array

Description

This characteristic is device-independent.

The IDA characteristic sets the values of the x- and y-coordinates for the input device area.

To invoke IDA, you must have set the report mode of the target device or device group to absolute mode (see the description of the RPMOD characteristic).

In a SETCHR call with the IDA characteristic, you specify the minimum and maximum values for both the x- and y-coordinates. The PDI does not maintain a 1:1 aspect ratio of the input device area. (See Section 3.11.5.)

NOTE

This characteristic affects the x- and y-coordinate values returned by any read operation.

Table 3-7 shows the values for the data parameter in the SETCHR call with the IDA characteristic.

SETCHR--SET DEVICE CHARACTERISTICS

Table 3-7: Data Values for IDA Characteristic

Word	Value
0	x minimum (default = 0)
1	x maximum (default = 4095)
2	y minimum (default = 0)
3	y maximum (default = 4095)

Status Returns

- +1 Success.
- +2 Function not supported.
- 500 One or more data values are out of range.
- 520 Invalid devid parameter. See Tables 2-2
-521 and 2-3.
- 522
- 523
- 524

SETCHR--SET DEVICE CHARACTERISTICS

3.11.5 SETCHR IDAASP--Set Input Device Area/1:1 Aspect Ratio

Characteristic Number: 9

Data Value: 3-Word Integer Array

Description

This characteristic is device-independent.

The IDAASP characteristic sets the values of the x- and y-coordinates for the input device area, while maintaining a 1:1 aspect ratio.

To invoke IDAASP, you must have set the report mode of the target device or device group to absolute mode (see the description of the RPMOD characteristic).

In a SETCHR call with the IDAASP characteristic, you specify the minimum and maximum x-coordinate values, and the minimum y-coordinate value. The PDI calculates the y-coordinate maximum value in order to maintain a 1:1 aspect ratio of the input device area.

NOTE

This characteristic affects the x- and y-coordinate values returned by any read operation.

Table 3-8 shows the values for the data parameter in the SETCHR call with the IDAASP characteristic.

Table 3-8: Data Values for IDA Characteristic

Word	Value
0	x minimum (default = 0)
1	x maximum (default = 4095)
2	y minimum (default = 0)

SETCHR--SET DEVICE CHARACTERISTICS

Status Returns

- +1 Success.
- +2 Function not supported.
- 500 One or more data values are out of range.
- 520 Invalid devid parameter. See Tables 2-2
-521 and 2-3.
- 522
- 523
- 524

SETCHR--SET DEVICE CHARACTERISTICS

3.11.6 SETCHR LNCHR--Set Serial Line Characteristics

Characteristic Number: 66

Data Value: 4-Word Integer Array

Description

This characteristic is device-specific. You must specify a unique device ID in the devid parameter of the call.

The LNCHR characteristic modifies the transmission line characteristics between a positional device and the port to which it is connected. Serial line characteristics include character length, number of stop bits per character, and parity type.

After issuing the SETCHR call with this characteristic, the PDI first attempts to notify the device of the line change, and then it sets the port to the correct serial mode.

You can set the line characteristics for serial ports only. The serial ports are DECTouch ports 4 and 5, and the Professional's Communication and Printer ports, both port number 2.

Table 3-9 shows the values for the data parameter in the SETCHR call with the LNCHR characteristic.

Table 3-9: Data Values for LNCHR Characteristic (Serial Only)

Word	Line Characteristic	Possible Values
0	Bits per character	Integer between 5 and 9 (decimal), inclusive. Default is 8.
1	Number of stop bits	0 = 1.0 stop bit 1 = 1.5 stop bits (default) 2 = 2.0 stop bits
2	Parity enable	0 = disable (default) 1 = enable
3	Parity type	0 = even 1 = odd

SETPHR--SET DEVICE CHARACTERISTICS

Status Returns

- +1 Success.
- +2 Function not supported.
- 500 One or more data values are out of range.
- 520 Invalid devid parameter. See Tables 2-2
and 2-3.
- 521
- 522
- 523
- 524

SETCHR--SET DEVICE CHARACTERISTICS

3.11.7 SETCHR ORIG--Set Device Origin

Characteristic Number: 5

Data Value: 1-Word Integer

Description

This characteristic is device-independent.

The ORIG characteristic specifies which corner of the physical device space corresponds to the coordinate origin. The device origin affects both relative and absolute coordinates.

NOTE

This characteristic affects the x- and y-coordinate values returned by any read operation.

Table 3-10 shows the possible values for the data parameter in the SETCHR call with the ORIG characteristic.

Table 3-10: Data Values for ORIG Characteristic

Data Value (Decimal)	Origin
0	Bottom left
1	Top left (default)
2	Top right
3	Bottom right

Status Returns

- +1 Success.
- 500 One or more data values are out of range.
- 520 Invalid devid parameter. See Table 2-2.

SETCHR--SET DEVICE CHARACTERISTICS

3.11.8 SETCHR PORT--Set Device to Port

Characteristic Number: 6

Data Value: None, parameter is ignored

Description

This characteristic is device-dependent.

The PORT characteristic associates a class and subclass with a particular port. Use this call to assign a device number to a DECTouch port. Since the PDI driver handles different devices differently, you can call SETCHR PORT to cause the driver to handle a device connected to a particular port as the specified device.

Your application can change the port association only under the following conditions:

- The device of the desired class/subclass combination must not be attached.
- The desired port must not be attached.

For example, you can change the PORT characteristic before calling ATTPD for the desired class/subclass, or after issuing a DETPD for the desired class/subclass.

Also, the device ID must have a class set, and optionally can have the subclass set. The port must be specified, and its value must be in the range 2 through 8 (decimal).

Status Returns

- +1 Success.
- 500 One or more data values are out of range.
- 520 Invalid devid parameter. See Table 2-2.

SETCHR--SET DEVICE CHARACTERISTICS

3.11.9 SETCHR PROD--Position a Relative-Oriented Device

Characteristic Number: 8

Data Value: 3-Word Integer Array

Description

This characteristic is device-independent.

The PROD characteristic centers the device coordinates for a positional device. This is especially useful when used with a relative-oriented device such as a quadrature mouse or joystick. After invoking this characteristic, subsequent data received from the positional device causes the driver to increment and/or decrement the initial value.

You can also use PROD with a positional device that is not relative-oriented; however, as soon as the device begins transmitting data, the initialized coordinates are overwritten.

The formulas the driver uses for centering the coordinates are:

$$x = ((x_maximum - x_minimum)/2) + x_minimum$$
$$y = ((y_maximum - y_minimum)/2) + y_minimum$$

Table 3-11 shows the possible values for the data parameter in the SETCHR call with the PROD characteristic.

Table 3-11: Data Values for PROD Characteristic

Word	Contents
0	Center coordinates
1	Reserved
2	Reserved

Status Returns

+1 Success.

SETCHR--SET DEVICE CHARACTERISTICS

3.11.10 SETCHR RESET--Reset Positional Device

Characteristic Number: 1

Data Value: None, parameter is ignored

Description

This characteristic is device-independent.

The RESET characteristic forces a positional device to a known state. For all devices except the Touch Screen Monitor of DECTouch, RESET sets the normalization boundaries to 4096 by 4096. For the Touch Screen Monitor, RESET sets the normalization boundaries to 4096 by 2560. For all devices including the Touch Screen Monitor, RESET also sets the origin to the upper left corner.

Status Returns

+1	Success.
-520	Invalid devid parameter. See Table 2-2
-521	and 2-3.
-522	
-523	
-524	

SETCHR--SET DEVICE CHARACTERISTICS

3.11.11 SETCHR RESOL--Set Coordinate Resolution

Characteristic Number: 8

Data Value: 2-Word Integer Array

Description

This characteristic is device-independent.

The RESOL characteristic sets the resolution of relative coordinates returned by any read operation. The coordinate resolution is the number of coordinate units returned by the physical device for every inch of its movement.

NOTE

This characteristic affects the x- and y-coordinate values returned by any read operation.

To invoke RESOL, you must have set the report mode of the target device or device group to relative mode (see the description of the RPMOD characteristic).

As an example of setting the resolution, suppose you set a mouse device in relative mode to an x resolution of 10 units per inch. If the mouse moves 2 inches horizontally, the x value returned by the REDRPT routine is 20 (or -20, depending on the coordinate origin).

Table 3-12 shows the possible values for the data parameter in the SETCHR call with the RESOL characteristic.

Table 3-12: Data Values for RESOL Characteristic

Word	Value
0	Units per inch of x coordinate
1	Units per inch of y coordinate

SETPHR--SET DEVICE CHARACTERISTICS

Status Returns

- +1 Success.
- 500 One or more data values are out of range.
- 520 Invalid devid parameter. See Table 2-2
- 521 and 2-3.
- 522
- 523
- 524

SETCHR--SET DEVICE CHARACTERISTICS

3.11.12 SETCHR RPMOD--Set Report Mode

Characteristic Number: 2

Data Value: 1-Word Integer

Description

This characteristic is device-independent.

The RPMOD characteristic sets the report mode of a device or device group to either absolute mode, relative mode, or device physical mode:

- In *absolute mode*, all coordinate data returned by a REDRPT call are absolute values that range between the minimum and maximum normalization boundaries. (See Section 3.11.4 for details on the normalization boundaries.)
- In *relative mode*, all coordinate data returned in a REDRPT call are values relative to the last report block that was read. The resolution of relative coordinate data can also be altered. (See Section 3.11.11 for details on resolution.)
- In *device physical mode*, no coordinate conversion occurs. Your application receives coordinate data as physically interpreted by the positional device.

NOTE

This characteristic affects the x- and y-coordinate values returned by any read operation.

Table 3-13 shows the possible values for the data parameter in the SETCHR call with the RPMOD characteristic.

Status Returns

- +1 Success.
- 1 Bad parameters; report mode was not in the range 0-2.
- 520 Invalid devid parameter. See Tables 2-2 and 2-3.

SETCHR--SET DEVICE CHARACTERISTICS

Table 3-13: Data Values for RPMOD Characteristic

Value	Description
0	Absolute mode (default)
1	Relative mode
2	Device physical

SETCHR--SET DEVICE CHARACTERISTICS

3.11.13 SETCHR SCALE--Set Touchscreen Scaling

Characteristic Number: 128

Data Value: 8-Word Integer Array

Description

This SETCHR routine is used by the DECTouch alignment procedure.

SPMAST--SPECIFY MOUSE AST

3.12 SPMAST--SPECIFY MOUSE AST

Specify AST routine to execute upon occurrence of event.

Format

```
CALL SPMAST (status, astadd, [devid], [butmask],  
            [xinc], [yinc])
```

Where:

status A two-word decimal integer array whose first word receives the status return of the call. The received value can be one of the following decimal integers:

IS.SUC +01 Call completed successfully.

(none) -538 AST address is invalid. (Cannot be odd, for example.)

astadd A one-word integer containing the address of an AST service routine to be executed upon occurrence of the specified event.

devid The address of a two-word device identification. The default devid is 00. See Section 2.2 for details.

butmsk The button mask, a one-word integer specifying the button events that will trigger completion of the read operation.

xinc A one-word positive integer value specifying the amount of change in the x-coordinate value to trigger the AST. The value can be 0, and it must be within the input device area.

yinc A one-word positive integer value specifying the amount of change in the y-coordinate value to trigger completion of the read operation. The value can be 0, and it must be within the input device area.

SPMAST--SPECIFY MOUSE AST

Description

The purpose of SPMAST is to allow you to specify the address of an AST routine that will execute when the specified event(s) occur. An event can be a change in state of specified button(s), and/or a change in the position of at least the amount specified by the x or y increments.

While the AST executes, mouse ASTs are disabled; upon exiting, the ASTs are reenabled.

You can call this routine to specify ASTs for any and all connected devices. You can specify separate AST addresses and qualifiers for each device.

Subsequent calls to SPMAST for a particular device override any previously-specified AST for that device.

All of the data normally returned by read routines is returned on the user stack, which contains the data shown in Table 3-14. To handle ASTs, your programming language must be able to access the user stack.

NOTE

Your task must remove the eight words SP+00 to SP+12 from the stack before exiting the AST.

SPMAST--SPECIFY MOUSE AST

Table 3-14: Stack Values Upon AST Entry, SPMAST

Current Stack Pointer	Contents
SP+22	Event flag mask word
SP+20	PS of task prior to AST
SP+18	PC of task prior to AST
SP+16	Task's Directive Status Word (DSW)
SP+14	Reserved
SP+12	Reserved
SP+10	Device ID second word (subclass/unit)
SP+08	Device ID first word (class)
SP+06	Button data
SP+04	y-coordinate
SP+02	x-coordinate
SP+00	Cause code: 01 - Button changed state 02 - X increment satisfied 03 - Y increment satisfied

3.13 WRTDEV--WRITE RAW DATA TO DEVICE

Write data directly to a device without interpretation.

Format

CALL WRTDEV (status, len, buff, devid)

Where:

status A two-word decimal integer array whose first word receives the status return of the call. The received value can be one of the following decimal integers:

IS.SUC	+01	Call completed successfully.
IE.BAD	-01	Invalid format for parameter block.
IE.DNA	-07	Device not attached.
IE.FHE	-59	Fatal hardware error on device.
(none)	-520	Invalid devid parameter. See Tables 2-2 and 2-3.
(none)	-521	
(none)	-522	
(none)	-523	
(none)	-524	

len A one-word integer value specifying the number of characters to write to the device. This is the length of the data buffer in bytes.

buff A one-word integer containing the address of a buffer to be transmitted to the device.

devid The address of a two-word device identification number. The default devid is 00. See Section 2.2 for details.

Description

This routine writes raw data to a positional device. The device identification parameter must describe a single unique device. The buff parameter is a data buffer whose length is determined by the value of the len parameter.

You normally use WRTDEV to transmit data to an intelligent device, such as a graphics tablet. An intelligent device is one that can process a string of data.

CHAPTER 4

SAMPLE PROGRAMS

4.1 FORTRAN-77

```
c Program file MOUSE.FTN
c -----
c Command file, MOUSE.CMD:
c -----
c SY:MOUSE/FP/CP=SY:MOUSE/MP
c TASK=MOUSE
c UNITS = 19
c ASG = SY:5:6:7:8:9:10:11:12
c ASG = TI:13:15
c EXTTSK= 952
c CLSTR=PROF77,PDL,CGLFPU,RMSRES,POSRES:RO
c
c ;
c ; DEFINE BUFFER SIZES
c EXTSCT = DM$BUF:4540 ; dynamic single choice menu
c EXTSCT = FL$BUF:4310 ; file selection/specification
c EXTSCT = HL$BUF:3500 ; HELP text/menu
c EXTSCT = MM$BUF:1000 ; multiscreen menu
c EXTSCT = MN$BUF:4540 ; static single choice menu
c ;
c ; DEFINE LUN ASSIGNMENTS
c GBLDEF = HL$LUN:21 ; HELP frame file
c GBLDEF = MN$LUN:20 ; menu frame file
c GBLDEF = MS$LUN:16 ; message frame file
c GBLDEF = TT$EFN:1 ; terminal I/O event flag
c GBLDEF = TT$LUN:15 ; terminal I/O
c GBLDEF = WC$LUN:22 ; directory searches for OLDFIL
c ; and NEWFIL routines and
c ; callable print services
c ;
c GBLDEF = G$LUN:17 ; for Core Graphics Library
c GBLDEF = G$EFN:3 ; for Core Graphics Library
c GBLDEF = PD$LUN:23 ; PDL device I/O
c GBLDEF = PD$EFN:2 ; PDL I/O event flag
c //
c -----
```


FORTRAN-77

```

c Overlay Descriptor Language file, MOUSE.ODL:
c -----
c           .ROOT MOUSE-RMSROT-LIBR,RMSALL
c   LIBR:   .FCTR LB:[1,5]PROF77/LB
c   @LB:[1,5]PROF77
c   @LB:[1,5]RMSRLX
c           .END
c -----
c To run this program at DCL level:
c
c   $ INSTALL [ZZSYS]CGLFPU
c   $ INSTALL [ZZSYS]PROF77
c   $ INSTALL LB:[1,5]PDL
c
c -----
c
c   program mouse
c
c This program uses a positional device to draw on
c the screen. The program loops indefinitely until aborted
c or until it encounters a positional device error.
c The graphics are generated using the CORE Graphics Library (CGL).
c
c   include 'lb:[1,5]CGL.FTN'
c
c This instruction provides the file CGL.FTN
c that declares a set of integer constants
c corresponding to the names of the CGL instructions.
c
c Declare variables for use as parameters in PDL calls:
c
c   INTEGER*2   status,xint,yint,button
c
c Declare variables for use as parameters in CGL calls:
c
c   REAL        xreal,yreal
c
c Declare flag variable to control looping:
c
c   LOGICAL     flag
c
c Initialize CGL core and invoke new frame:
c
c   call CGL( GIC )           !INITIALIZE_CORE
c   call CGL( GNF )           !NEW_FRAME
c
c Guarantees that the graphics system is in
c a start state with default parameters
c established, and clears the screen.
c

```

FORTRAN-77

```

c Map the positional device coordinates to the
c screen and allow for the aspect ratio (decimal points
c are required since parameters are REAL):
c
      call CGL( GSW, 0.00, 4095.00, 0.00, 4095.00*0.625 ) !SET_WINDOW
c
c Specifies the edges of the window and resets
c the current position to the origin of the window. Notice
c that we specified REAL constants for parameters.
c
      call CGL( GSO, 1 )      !SET_ORIGIN
c
c Define our cursor symbol:
c
      call CGL( GSMKS, 2, 0 ) !SET_MARKER_SYMBOL
      call CGL( GSWM, 2 )     !SET_WRITING_MODE
c
c Specifies one of five standard symbols or
c a user-defined symbol as the current marker
c symbol, and specifies the exact manner in
c which CGL draws output primitives on the screen.
c
c Attach the positional device:
c
      call ATTPD( status )
      if ( status .NE. 1 ) goto 900
      flag = .FALSE.
c
c Now read coordinates of positional device and convert
c to real values. Notice that we do not care what device
c we get the input from, so we have omitted the devid from the
c parameter list.
c
c (Loop begins here.)
c
300   call REDRPT( status, xint, yint, button )
      if ( status .NE. 1 ) goto 900
      yreal = yint
      xreal = xint
      if ( flag .EQ. .TRUE. ) call CGL( GMKR2, 0, 0 ) !MARKER_REL_2
      flag = .FALSE.
c
c If user presses button, draw a line; if user does not press
c button, just echo the cursor:
c
      if ( button .EQ. 1 ) then
          call CGL( GSWM, 4 )           !SET_WRITING_MODE
          call CGL( GLA2, xreal, yreal ) !LINE_ABSOLUTE_2
          call CGL( GSWM, 2 )           !SET_WRITING_MODE
      else
          call CGL( GMKA2, xreal, yreal ) !MARKER_ABSOLUTE_2

```

FORTRAN-77

```

        flag = .TRUE.
    endif
c
c Go back to top of loop:
c
    goto 300
c
c Process error
c
900    print 990, 'Positional device error: ',status
990    format ( A,I )
    end

```

4.2 PASCAL

```

{ Program file MOUSE.PAS
{ -----
{ Command file, MOUSE.CMD:
{ -----
{ MOUSE/CP/FP,MOUSE/MA/-SP=MOUSE/MP
{ CLSTR=PASRES,PDL,CGLFPU,POSRES,RMSRES:RO
{ TASK = MOUSE
{ STACK = 30
{ UNITS = 47 ; Extra unit
{ GBLDEF = TT$EFN:7
{ GBLDEF = WC$LUN:45
{ GBLDEF = MS$LUN:44
{ GBLDEF = HL$LUN:43
{ GBLDEF = MN$LUN:42
{ GBLDEF = TT$LUN:41
{ GBLDEF = G$LUN:41
{ GBLDEF = PD$LUN:57 ; LUN for positional device
{ GBLDEF = PD$EFN:2 ; Event flag for positional device.
{ ASG = TT1:33
{ ASG = SY:36
{ ASG = LB:34:35:37
{ ;EXTSCT = MS$BUF:3100
{ ;EXTSCT = MN$BUF:4540
{ ;EXTSCT = DM$BUF:4540
{ ;EXTSCT = MM$BUF:1000
{ ;EXTSCT = HL$BUF:3400
{ //
{ -----
{ Overlay Descriptor Language File, MOUSE.ODL:
{ -----
{ .ROOT USER-PASLB-RMSROT
{ USER: .FCTR MOUSE
{ PASLB: .FCTR LB:[1,5]PASLIB/LB
{ @LB:[1,5]RMSRLX

```


PASCAL

```
initialize_core;
new_frame;

{ Guarantees that the graphics system is in      }
{ a start state with default parameters         }
{ established, and clears the screen.           }

{ Map the positional device coordinates to the   }
{ screen and allow for the aspect ratio         }

set_window( 0.00, 4095.00, 0.00, 4095.00*0.625 );

{ Specifies the edges of the window and resets   }
{ the current position to the origin of the     }
{ window. Note that we specified REAL constants }
{ for parameters.                               }

set_origin( 1 );

{ Define our cursor symbol. Note that the CGLDEFS.PAS }
{ procedure declaration of SET_MARKER_SYMBOL requires }
{ a CHAR value as the second parameter, NOT an integer. }

set_marker_symbol( 2, '0' );
set_writing_mode( 2 );

{ Specifies one of five standard symbols or      }
{ a user-defined symbol as the current marker   }
{ symbol, and specifies the exact manner in     }
{ which CGL draws output primitives on the screen. }

{ Attach the positional device:                  }

ATTACH_POSITIONAL_DEVICE( status );
if ( status <> 1 ) then goto 900;
flag := false;

{ Now read coordinates of positional device and convert }
{ to real values. Note that we do not care what device }
{ we get the input from, so we have omitted the devid  }
{ from the parameter list.                            }

while true do
  begin
    READ_POSITIONAL_REPORT( status, xint, yint, button );
    if ( status <> 1 ) then goto 900;
    yreal := yint;
    xreal := xint;
    if ( flag = true )
      then marker_rel_2( 0, 0 );
    flag := false;
  end;
```

PASCAL

```
{ If user presses button, draw a line; if user }
{ does not press button, just echo the cursor: }

if ( button = 1 )
  then begin
    set_writing_mode( 4 );
    line_abs_2( xreal, yreal );
    set_writing_mode( 2 );
  end
  else marker_abs_2( xreal, yreal );
flag := true;

  { Go back to top of loop.}
end;

{ Process error. }
900: writeln( 'Positional device error: ', status );

end.
```

4.3 BASIC-PLUS-2

```
10      ! Program file MOUSE.B2S
!
!-----
!      PAB Command file, MOUSE.CMD:
!-----
!      SY:MOUSE/FP/CP=SY:MOUSE/MP
!      TASK=MOUSE
!      UNITS = 19
!      ASG = SY:5:6:7:8:9:10:11:12
!      ASG = TI:13:15
!      EXTTSK= 952
!      CLSTR=PBFSML,PDL,CGLFPU,RMSRES,POSRES:RO
!
!
!      ; DEFINE BUFFER SIZES
!      EXTSCT = DM$BUF:4540 ; dynamic single choice menu
!      EXTSCT = FL$BUF:4310 ; file selection/specification
!      EXTSCT = HL$BUF:3500 ; HELP text/menu
!      EXTSCT = MM$BUF:1000 ; multiscreen menu
!      EXTSCT = MN$BUF:4540 ; static single choice menu
!
!
!      ; DEFINE LUN ASSIGNMENTS
!      GBLDEF = HL$LUN:21 ; HELP frame file
!      GBLDEF = MN$LUN:20 ; menu frame file
!      GBLDEF = MS$LUN:16 ; message frame file
!      GBLDEF = TT$EFN:1 ; terminal I/O event flag
!      GBLDEF = TT$LUN:15 ; terminal I/O
```


BASIC-PLUS-2

```

!       GBLDEF = WC$LUN:22      ; directory searches for OLDF
!                               ; and NEWFIL routines and
!                               ; callable print services
!
!       GBLDEF = G$LUN:17      ; for Core Graphics Library
!       GBLDEF = G$EFN:3       ; for Core Graphics Library
!       GBLDEF = PD$LUN:23     ; PDL device I/O
!       GBLDEF = PD$EFN:2     ; PDL I/O event flag
!       //
!-----
! Overlay Descriptor File, MOUSE.ODL:
!-----
!           .ROOT BASIC2-RMSROT-USER,RMSALL
!       USER: .FCTR SY:MOUSE-LIBR
!       LIBR: .FCTR LB:[1,5]PBFOTS/LB
!       @LB:[1,5]PBFIC5
!       @LB:[1,5]RMSRLX
!           .END
!-----
! To run this program at DCL level:
!-----
! $ INSTALL [ZZSYS]CGLFPU
! $ INSTALL LB:[1,5]PDL
! $ RUN MOUSE
!
!-----
! This program uses a positional device to draw on
! the screen. The program loops indefinitely until
! aborted or encountering a positional device error.
!
! The graphics are generated using the CORE Graphics
! Library (CGL).
!
! The next instruction provides the file CGL.B2S
! that declares a set of integer constants
! corresponding to the names of the CGL instructions.
!
20 %include 'lb:[1,5]CGL.B2S'
!
! Declarations of integer variables used with the PDL
! routines:
!
! Declare integer
!           xint,           ! x-coordinate returned           &
!           yint,           ! y-coordinate returned           &
!           bint,           ! button status returned         &
!           libstatus       ! PDL routine status word       &
!
! Declarations of real variables used with the CGL
! routines:

```

BASIC-PLUS-2

```

!
Declare real                                     &
        xreal,                                ! x-coordinate   &
        yreal                                 ! y-coordinate
!
! We use the real variables to convert the integer
! coordinates, because CGL requires the real values.
!
100 !
! Clear the screen
!
call CGL by ref (initialize_core)
call CGL by ref (new_frame)
!
! Guarantees that the graphics system is in
! a start state with default parameters
! established, and clears the screen.
!
110 !
! Map the positional device coordinates to the
! screen and allow for the aspect ratio.
!
call CGL by ref (set_window,0,4095.,0,4095.*.625)
120 !
! Specifies the edges of the window and resets
! the current position to the origin of the window.
!
! Set window origin to top-left to match
! positional device.
!
call CGL by ref (set_origin,1)
130 !
! Specifies which corner of the viewport
! corresponds to the origin of the window.
!
! Define our cursor symbol.
!
call CGL by ref (set_marker_symbol,2%,0%)
call CGL by ref (set_writing_mode,2%)
140 !
!
! Specifies one of five standard symbols or
! user-defined symbol as the current marker
! symbol, and specifies the exact manner in
! which CGL draws output primitives on the
! screen.
!
! Attach the positional device.
!
call ATTPD by ref (libstatus)
150 if libstatus <> 1 then goto 900

```

BASIC-PLUS-2

```

160   f% = 0%
300   !
      ! Loop Begins here.
      !
      ! Read coordinates of positional device and convert
      ! to real values. Notice that we do not care what
      ! device we get input from, so we have ommited the
      ! devid from the parameter list.
      !
      call REDRPT by ref (libstatus,xint,yint,bint)
310   if libstatus <> 1 then 900
320   xreal=xint
      yreal=yint
330   if f% = 1% then call CGL by ref (marker_rel_2,0,0)
      f% = 0%
350   !
      ! If button pressed, draw a line.
      !
      if bint = 1% then call CGL by ref (set_writing_mode,4%)
      call CGL by ref (set_writing_mode,4%)
      call CGL by ref (line_abs_2,xreal,yreal)
      call CGL by ref (set_writing_mode,2%)
      goto 300
360   !
      !
      ! Specifies the exact manner in which CGL
      ! draws the output primitives on the screen,
      ! changes the current position to the
      ! specified position and draws a connecting
      ! line.
      !
      ! If button not pressed, echo the cursor.
      !
      if bint = 0% then call CGL by ref (marker_abs_2,xreal,yreal)
      f% = 1%
      !
      ! Changes the current position to the
      ! specified position and draws a marker.
      !
390   goto 300
900   !
      ! Process error
      !
      print "Positional device error: ";libstatus
999   END

```


MACRO-11

4.4 MACRO-11

```
.TITLE Example - POSITIONAL DEVICE EXAMPLE
.IDENT /V1.00/
.ENABL LC

.MCALL DIR$,QIOW$,EXIT$$

;+
; This program demonstrates the Positional Device Interface
; using GIDIS to move a cursor around the screen and draw lines
; when the button is depressed. The program is an endless loop.
; You must press INTERRUPT-DO to exit.
;
; The following will assemble and build it.
;
; PMA TEST
; PAB TEST=TEST,[1,5]PDIOBJ
; /
; GBLDEF = PD$LUN:1
; GBLDEF = PD$EFN:2
; //
;-

;+
;
; This is the entry point. The SETUP data buffer will be sent to
; TI: to initialize GIDIS, clear the screen, set device coordinate
; system, and set the cursor to a continuous mode crosshair.
; Refer to the data definition of SETUP for the actual commands.
;
; We will also attempt to attach the positional device. If the
; status from the call is not a +1, we will exit the task.
;
;-

START: Dir$      #Setup          ; Clear the screen
        Mov      #Attach,R5     ; Get PDI attachment arguments
        Call     ATTPD          ; Attach to the PDI
        Cmp     #1,Status       ; Check the status
        Bne     Error          ; Not successful, branch to exit

;+
;
; We have now attached the positional device and setup the screen.
; It's time to go into our loop.
;
; Logic is:
;
```

MACRO-11

```

; DO forever
;
; Get device data
;
; Is Button          NO
; Set?              THEN Move Position
;                  Draw := False
;
;
;                  YES
; THEN Is Draw := False?
;
;                  NO
; THEN Move position
;                  Draw := true
;
;                  YES
; THEN Draw a line
;
; END DO
;
;
;
;-

```

```

;+
;
; Structure of parameter block in REDRPT call:
;
;           8   7
; 15 -----#R5 +0
;     UNUSED          |          NUMBER OF PARAMETERS
; -----#R5 +2
;           ADDRESS OF STATUS
; -----#R5 +4
;           ADDRESS OF XCOORD
; -----#R5 +6
;           ADDRESS OF YCOORD
; -----#R5 +8
;           ADDRESS OF BUTTON
; -----#R5 +10
;
;
;-

```

```

Loop:  Mov    #Read,R5          ; Load the REDRPT arguments
      Call  REDRPT            ; Call the library

      Cmp   #1,Status         ; Success?
      Bne   Error             ; Nope, exit

      Bit   #1,Button         ; Is the RIGHTHAND button set?
      Bne   10$              ; Yes, branch

```

MACRO-11

```

        Clr      Draw          ; Set drawing flag FALSE
        Br       20$          ; Go to common cursor tracking
                               ; code

10$:    Tst      Draw          ; Were we drawing last loop?
        Bne     30$          ; Yes, branch

        Mov     #1,Draw       ; Set Draw to TRUE

20$:    Mov     Xdata,Xmove    ; Load X address
        Mov     Ydata,Ymove    ; Load Y address
        Dir$   #Track         ; Move the current position
        Br     40$          ; Go to end of loop

30$:    Dir$   #Lines         ; Draw a line from last point

40$:    Br     Loop          ; Loop

```

```

;+
;
; At this point, and error has ocured. We will simply EXIT from
; the task. Note that if the positional device was ATTACHED it
; will be DETACHED automatically by the P/OS I/O rundown mechanism.
;
;-

```

```

Error:  Exit$$          ; Just exit on an error

```

```

;+
;
; These QIOW$'s are Write Special Data's (IO.WSD), and are in GIDIS
; format (SD.GDS).
;
; I'm simply going to use the default TI: LUN of 5
;
;-

```

```

Setup:  QIOW$   IO.WSD,5,1,,,,<buf1,buf11,,sd.gds>
Track:  QIOW$   IO.WSD,5,1,,,,<buf2,buf21,,sd.gds>
Lines:  QIOW$   IO.WSD,5,1,,,,<buf3,buf31,,sd.gds>

```

```

;+
;
; These are the data buffers for the TI: QIO's
;
;-

```

```

Buf1:   .BYTE   1,1          ; Initialize
        .WORD   -1          ; All subsystems

```


MACRO-11

```

.BYTE 0,6 ; New_Picture
.BYTE 6,5 ; Set_Output_Cursor
.WORD -1 ; Special Alphabet
.WORD 1 ; Tracking crosshair
.WORD 16.,16.,8.,8. ; Ignored for crosshair

.BYTE 1,72. ; Set_output_cursor_rendition
.WORD 0 ; CONTINUOUS (nonblinking)

.BYTE 2,12. ; Set_Output_IDS
.WORD 4096.,2560. ;

.BYTE 4,9. ; Set_GIDIS_Output_Space
.WORD 0,0,4095.,2559. ;

.BYTE 4,13. ; Set_Output_Viewport
.WORD 0,0,4095.,2559. ;

.BYTE 1,21. ; Set_Primary_Color
.WORD 7 ; Use Color Map entry 7

.BYTE 1,22. ; Set_Writing_Mode
.WORD 4 ; OVERLAY mode

Buf11 =.-Buf1

Buf2: .BYTE 2,29. ; Set_Position
XMOVE: .WORD 0 ; X
YMOVE: .WORD 0 ; Y
Buf21 =.-Buf2

Buf3: .BYTE 2,25. ; Draw_Lines
Xdata: .WORD 0 ; X
Ydata: .WORD 0 ; Y
Buf31 =.-Buf3

;+
;
; These are the argument blocks for ATTPD and REDRPT
; Notice that the devid parameter is NOT supplied, since
; I'll use any device that responds.
;
;-

Attach: .WORD 1 ; 1 parameter
        .WORD Status ; Status

Read: .WORD 4 ; 4 parameters
      .WORD Status ; Status

```

MACRO-11

```
.WORD Xdata ; X
.WORD Ydata ; Y
.WORD Button ; Button

Status: .WORD 0 ; Status word
Button: .WORD 0 ; Button

Draw: .WORD 0 ; Drawing flag

.END Start ; End of source
```


APPENDIX A

DEVICES YOU CAN USE WITH THE PDI

There are several ports through which you can connect a positional device to the Professional computer:

- The Communication Port
- The Printer Port
- Ports provided by the DECTouch (VRTS1-A) Color/Touch Screen Monitor

The Communication and Printer ports can support a number of serial input devices. Both ports function identically, with the exception of the cable needed for each. A cable for the Printer Port requires a 9-pin female connector, while a cable for the Communication Port requires a standard 25-pin female RS-232 type connector.

When not being used for positional device input, both of these ports may be used for their standard functions.

By connecting a DECTouch monitor to your Professional, you provide two additional serial ports as well as two parallel ports. We describe DECTouch later in this appendix.

The following sections describe each of the available devices. For specifications, installation instructions, and other detailed information about a particular device, refer to the documentation provided with that device.

Note that, as described in this appendix, some devices require modification to the initial switch settings to work properly with the Positional Device Interface.

SUMMAGRAPHICS MM 961 AND MM 1201 DIGITIZERS

A.1 SUMMAGRAPHICS MM 961 AND MM 1201 DIGITIZERS

The Summagraphics Corporation digitizers are tablet-type positional devices. A tablet is a surface that has a specified "active" area for digitizing. The Summagraphics tablet has a narrow groove etched on its plastic surface that defines the active area.

Summagraphics provides a mouse-like *cursor* and a pencil-like *stylus* that you can move over the tablet. This movement locates points defined within the active area, and sends the coordinates as digital information to the computer.

The MM 961 is a 6" x 9" tablet. The MM 1201 is a 12" x 12" version of the MM 961.

Both digitizers require a startup sequence after power-up, which the positional device driver automatically sends. To receive this data, the tablet must be connected to the desired port before running an application that uses the Positional Device Interface.

The MM 961 and MM 1201 require a power supply and null modem cable for correct operation. Modifications you must make to the standard jumper setting are to set the device to autobaud, 8-bit, no parity operation. Do this by setting the following jumper options:

- Jumper AC should be OUT. This selects no parity. On older models of the bit pads, this jumper was called the "8/9 bit" jumper, which also selected 8-bit bytes.
- Jumper AA should be OUT. This selects the automatic baud rate feature. On early models, this jumper was called the "BDR" jumper.
- Jumper AB should be IN. This selects 8-bit binary bytes. On early models, this option was automatically selected by the "8/9 bit" jumper.

The technical reference manual that comes with your bit pad should describe jumper modifications.

For ordering information, contact:

Summagraphics Corporation
35 Brentwood Avenue
P.O. Box 781
Fairfield, Connecticut 06430
(203) 384-1344

GTCO MICRO DIGI-PAD

A.2 GTCO MICRO DIGI-PAD

The Micro Digi-Pad is a compact electromagnetic positional device. Like the GTCO Digi-Pad 5, the Micro Digi-Pad uses a plastic tablet containing an array of conducting wires, through which a pulsing direct current travels. The current produces an electromagnetic field, which induces a signal in a coil contained in the cursor or stylus as it moves over the tablet.

The tablet has an active area consisting of a 6" x 6" unmarked square area on the tablet surface.

The Micro Digi-Pad needs no modification of the standard jumper settings. You should order the device to operate at 9600 baud (baud rate jumper IN). Also, the Micro Digi-Pad requires a power supply and null modem cable.

For ordering information, contact:

GTCO Corporation
1055 First Street
Rockville, Maryland 20850
(301)279-9550

A.3 SUMMAGRAPHICS SUMMAMOUSE

The Summagraphics SummaMouse is a mouse that reads optical information as it moves across a Mouse Pad. On the surface of the Mouse Pad run two sets of perpendicular stripes; these stripes absorb different wavelengths of light. The SummaMouse uses this optical information to translate movement over the stripes into digital data suitable for input to the computer.

The Summagraphics mouse needs no modification of the standard settings. You must order the SummaMouse in the standard "MM" data format, set with the standard automatic baud rate feature.

For ordering information, contact:

Summagraphics Corporation
35 Brentwood Avenue
P.O. Box 781
Fairfield, Connecticut 06430
(203) 384-1344

MICROSOFT SERIAL MOUSE

A.4 MICROSOFT SERIAL MOUSE

The Microsoft Serial Mouse is a mechanical positional device that detects a change in position by the movement of a metal ball over a hard surface. The mouse enclosure contains sensors that read the motion of the ball, and send this information to an on-board processor that digitizes the information. Once digitized, the information passes to the host computer.

The Microsoft Serial Mouse needs no modifications; it connects directly to the Professional's Communication Port. Note, however, that to connect the mouse to the Professional's Printer Port, you must connect pins 4 and 20 of the mouse to pin 5 (DTR) of the Printer Port.

You can ignore the instructions that Microsoft provides for using their mouse with the MS-DOS and PC-DOS operating systems.

For ordering information, contact:

Microsoft Corporation
10700 Northup Way
P.O. Box 97200
Bellevue, Washington 98009
(800) 426-9400

A.5 DECTOUCH (VRTS1-A)

The DECTouch (VRTS1-A) color monitor is a positional device whose main feature is a touch-sensitive screen. The screen uses resistive membrane technology, which provides extremely high resolution for individual touch points.

In addition to the touch screen, DECTouch also provides two parallel ports and two RS232 serial ports. To the parallel ports you can connect either Atari(c)-compatible joysticks or the DIGITAL LM200 Quadrature Mouse. To the two serial ports you can connect any of the supported serial devices.

The joystick and Quadrature Mouse require no modification.

For ordering information, contact your local DIGITAL sales office or sales representative. For detailed information on using, installing, or programming with the DECTouch monitor, refer to the DECTouch documents listed in the Preface.

SEIKO DT-3100 TABLET

A.6 SEIKO DT-3100 TABLET

The Seiko Tablet DT-3100 is a very high resolution tablet. Due to this high resolution capability, the device is ideal for use with Oriental character sets.

Figure A-1 shows how to connect the Seiko tablet to the Communication or Printer ports.

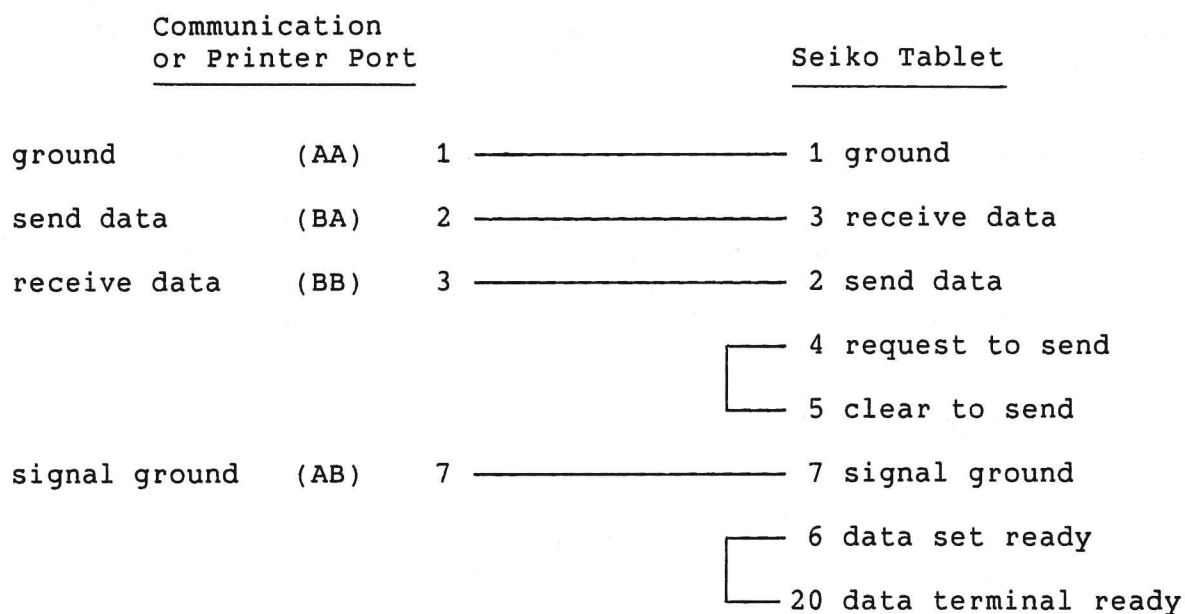


Figure A-1: Wiring for the Seiko Tablet

The connecting cable must be male to female. The male end connects to the Seiko Tablet. The female end connects to the Professional.

For ordering information, contact:

Seiko Instruments (USA), Inc.
19 Crosby Drive
Bedford, MA 01730
Tel: (617) 275-4092

SUMMAGRAPHICS BIT PAD ONE

A.7 SUMMAGRAPHICS BIT PAD ONE

The Bit Pad One is a tablet that works on the magnetostrictive principle. Below the surface of the tablet are special wires that deform in a known manner when current is pulsed nearby. A send wire in the tablet carries the current past these magnetostrictive wires, and the resulting deformation is received as strain waves in the coils of the Bit Pad One's cursor or stylus. A microprocessor formats the data and translates it into x and y coordinates.

To connect the Bit Pad One to the Professional, you must open the tablet by removing the metal bottom to gain access to the circuit board. Set the following connections, called "straps," as follows:

- **Pluggable Program Strap BA (Baud Rate)**

This item consists of three pins located on the circuit board, with a blue plug (Pluggable Strap) placed over two of the pins. The letters A and B are on the board on each side of the plug. Set the plug so that it covers the center pin and the pin next to the letter B.

- **POE Strap (Parity)**

This item consists of two points on the circuit board labeled POE. There should be no soldered connection (POE Strap) between these two points. If there is a connection, remove it. The factory setting omits the connection.

- **HCB Strap (Stop Bits)**

This item consists of two points on the circuit board labeled HCB. There should be no soldered connection (HCB Strap) between these two points. If there is a connection, remove it. The factory setting omits the connection.

In addition to the strap settings, you must set the switches in the three switch packs that are on the circuit board. These are labeled on the circuit board as SW1, SW2, and 7.

Table A-1 shows the switch settings when you are connecting the Bit Pad One to the Communication Port or the Printer Port. Table A-2 shows the settings when you are connecting the Bit Pad One to a DECTouch serial port.

SUMMAGRAPHICS BIT PAD ONE

Table A-1: Bit Pad One Switch Settings for XK0: or TT2:

	Switch Pack 1	Switch Pack 2	Switch Pack 7
Switch 1	*	off	off
Switch 2	*	off	on
Switch 3	*	off	off
Switch 4	*	off	off
Switch 5	*	off	off
Switch 6	off	off	off
Switch 7	on	x	off
Switch 8	off	x	off
Switch 9	on	x	off
Switch 10	x	x	off

KEY

* = You must not alter the factory setting
 x = Switch pack does not have the indicated switch

SUMMAGRAPHS BIT PAD ONE

Table A-2: Bit Pad One Switch Settings for DECtouch Port

	Switch Pack 1	Switch Pack 2	Switch Pack 7
Switch 1	*	on	off
Switch 2	*	off	off
Switch 3	*	on	on
Switch 4	*	on	off
Switch 5	*	on	off
Switch 6	off	off	off
Switch 7	on	x	off
Switch 8	off	x	off
Switch 9	on	x	off
Switch 10	x	x	off

KEY

* = You must not alter the factory setting
 x = Switch pack does not have the indicated switch

For ordering information, contact:

Summagraphics Corporation
 35 Brentwood Avenue
 P.O. Box 781
 Fairfield, Connecticut 06430
 (203) 384-1344

SUMMAGRAPHS BIT PAD TWO

A.8 SUMMAGRAPHS BIT PAD TWO

Table A-3 shows the switch settings when you are connecting the Bit Pad Two to any port.

Table A-3: Bit Pad Two Switch Settings for Any Port

	Switch Pack 1	Switch Pack 2	Switch Pack 7
Switch 1	off	off	off
Switch 2	off	off	off
Switch 3	on	off	on
Switch 4	off	off	off
Switch 5	off	off	off
Switch 6	on	off	on
Switch 7	off	on	on
Switch 8	off	off	off

APPENDIX B

USING THE SKETCHPAD DEMONSTRATION APPLICATION

The Sketchpad application included in the kit is a sample application that you can use to draw simple pictures on the screen.

NOTE

Before running the Sketchpad, you must first start the Positional Device Interface, as described in Section 1.2.

Note the following items regarding the positional devices you can use with Sketchpad:

- Some devices, when initialized, transmit coordinates that are below the viewport on the Sketchpad's screen display. Therefore, you might have to move your positional device around before the cursor appears on the screen. The OptoMouse, for instance, must be "pushed up" several times before the cursor appears.
- The driver recognizes only one button on any device. Therefore, the demo will respond only to the first button of a particular device. For devices that use a stylus instead of buttons, the driver responds to pressure on the stylus tip.

When you are finished with the demonstration, select EXIT to return to the applications menu.

THE SCREEN

B.1 THE SCREEN

Figure B-1 shows the Sketchpad display. It is divided into three areas:

1. Command Menu (left side of screen)
2. Drawing surface (center area)
3. Palette (right side of screen)

Exit		
Clear		
Line		
Circle		
Box		
Print		
Text ^Z		<Pen 6>
Select		Red 0
Fill OFF		Green 7
Cancel		Blue 7

Command
Menu

Drawing Surface

Palette

Figure B-1: Screen Display for Sketchpad Application

You can move freely among the three areas simply by moving your positional device (your finger in the case of DECTouch).

Before you select any of the commands on the command menu, Sketchpad places you in the default "Freehand Drawing" mode. By pressing and holding down the button on your device, you can draw lines on the drawing surface.

THE SCREEN

For various effects, you can select items from either the Command Menu or the Palette, as described in Sections B.2 and B.3.

B.2 THE COMMAND MENU

When you move your positional device into the menu area, the corresponding box "lights." Pressing the device button (or depressing the stylus) selects the function. Sketchpad confirms a selection by sounding the keyboard bell.

Only one function can be active at any time, except for the following:

- FILL can be active in combination with other functions.
- You can select CANCEL at any time.

Table B-1 describes each of the commands available from the Command Menu.

Table B-1: Sketchpad Commands from Command Menu

Command	Description
Exit	Terminates program execution and returns to the Main Menu.
Clear	Refreshes the screen, sets the current pen to 6, and sets Fill to OFF.
Line	Allows you to draw a straight line by marking its end points. When you are at the first point you want to select, press the device button. Then mark the second point in the same fashion. Sketchpad draws the line dynamically as you move to the second point. To terminate line mode, press the device button twice while on the second point; or, you could select the Cancel command.

THE COMMAND MENU

Command	Description
Circle	Allows you draw a circle by marking its center point, then marking the outside point of the the circle's radius. Begin the circle by selecting a center point and pressing the device button. Then mark the outside point of the radius in the same fashion. Sketchpad draws the circle dynamically as you move to the second point.
Box	Allows you draw a box by marking one corner, then a diagonally opposite corner. Begin the box by selecting a corner point and pressing the device button. Then mark a point that you want to appear as the diagonally opposite corner of the box. Sketchpad draws the box dynamically as you move to the second point.
Print	Dumps the image from the drawing surface to the printer connected to the Professional. The printer must be able to print graphic images (refer to the documentation that comes with your printer). The background does not appear in the printed copy, and all objects appear black. Neither the Command Menu nor the Palette are printed.
Text	Allows you to enter text from the keyboard. Once you have selected this command, mark a point on the drawing surface by moving your positional device to that point and pressing the button. Then you can enter text from the keyboard. To exit from the Text command you press the CTRL key and then the Z key on the keyboard.
Select	Allows you to select a point that defines a fill area when the Fill command is set to ON. If the Box, Circle, or Line commands are not in effect, Sketchpad fills areas to the selected point as you move your positional device over the drawing surface with the button depressed.

THE COMMAND MENU

Command	Description
Fill	Causes Sketchpad to fill in areas of the screen. Selecting Fill either toggles it to ON or OFF. When Fill is ON, other commands are affected as follows: <ul style="list-style-type: none">• Box--fills the box with current color as indicated by the Palette.• Circle--fills the circle with the current color as indicated by the Palette.• Line--fills the triangular area from the line to a point that you have last marked with the Select command.
Cancel	Cancels any selected command and turns Fill to OFF.

B.3 THE PALETTE

The top seven boxes on the right side of the screen are the colors available from Sketchpad's Palette. These colors correspond to the CGL writing index values zero through seven.

Sketchpad indicates the current color by displaying "<Pen n>" in the appropriate box in the palette area of the screen (see Figure B-1). You select a new color by moving the cursor to one of the palette boxes and then pressing the positional device button.

The lower three boxes on the right side of the screen are the red, green, and blue (RGB) settings for the current color. These settings indicate how much of each of the primary colors is mixed into the current color.

To alter the RGB settings, move the cursor to one of the RGB boxes and press the positional device button. The setting increases from zero to seven, and then resets to zero. Changing the RGB values affects the entire display, as the Sketchpad's CGL color map entry changes. See the *Core Graphics Library Manual* for further information.

THE PALETTE

To erase a portion of the display, set the color to zero by selecting the top box on the palette. This color is also the background color.

Although you can change the background color, it does not appear when you print an image from the screen.

APPENDIX C

GLOSSARY

aspect ratio

As used in this manual: the ratio between the size of the units on the x-axis and the size of the units on the y-axis.

button

A switch-type mechanism on a positional device. It can be the kind of standard button you find on a mouse, or it can be some other type of mechanism such as a retracting stylus tip.

cluster library

A structure that allows tasks to dynamically map memory-resident, shared libraries at run time. The advantage of using a cluster library is that it saves task virtual address space. A cluster library is also referred to as a *clustered resident library*.

DECTouch

A touch screen monitor.

device class

A set of similar devices, such as mice, keyboards, or joysticks.

device driver

A part of the operating system that interfaces hardware I/O controllers and their attached devices with the Executive.

device subclass

A value indicating a specific device in a device class.

driver

See device driver.

input device area

The area from which a positional device is able to transmit valid input.

positional device

Hardware used for input. The main feature of a positional device is its ability to transmit information about location to the computer.

Positional Device Interface (PDI)

Software that enables you to write applications that use a mouse, digitizing tablet, touch screen, or other positional device.

Positional Device Library (PDL)

A set of routines supplied with the PDI kit that perform operations for positional devices. PDL is the name of the library's global entry point.

Sketchpad

A sample application that allows you to use a positional device to draw simple pictures on the terminal screen.

task

The fundamental executable program unit.

task builder

A tool (sometimes called a *linker*) that converts an object module into a task image by relocating code and data and resolving external references.

task image

A file that contains a loadable task in the form of absolute binary instructions and data.

INDEX

- Alternative
 - coordinate unit mapping, 2-4
- Application
 - Positional Device Interface, 1-2
 - Sketchpad, 1-3
 - Test the PDI, 1-3
- Application development
 - mapping coordinate units, 2-1
 - writing the program, 2-1ff
- Aspect ratio
 - of input device area, 2-2
- Attach
 - positional device routine, 3-2
- ATTPD
 - routine, 3-2
- BASIC-PLUS-2
 - sample program, 4-7
- Button
 - status, 2-10
- Calling method
 - R5, 3-1
- Cancel
 - mouse AST, 3-4
- Cancel Read on Event (EVNCAN)
 - positional device routine, 3-6
- CLSTR option
 - modifying in .CMD file, 2-11
- Cluster library
 - installing, 2-14
 - vs. object module, 2-13
- CNMAST
 - routine, 3-4
- Command file
 - installation, 2-14
 - modifying, 2-11
 - sample, 2-12
- Components
 - of PDI kit, 1-2
- Connecting
 - positional device, 1-5
- Coordinate units mapping, 2-1, 2-3, 2-4
- Coordinates
 - of terminal screen, 2-3
- Core Graphics Library
 - installing, 2-14
- DECTouch
 - parallel ports, A-4
 - serial ports, A-4
 - VRTS1-A, A-4
- Descriptor File
 - modifying, 2-12
 - NOTE regarding, 2-12
- Detach
 - explicitly vs. implicitly, 2-10
 - positional device routine, 3-5
- DETPD
 - routine, 3-5
- Development
 - of PDI applications, 2-1ff
- Device
 - DECTouch, A-4
 - descriptions, A-1ff
 - GTCO Micro Digi-Pad, A-3
 - identification, 2-5
 - Microsoft Mouse, A-4
 - Summagraphics MM 961/1201, A-2
 - Summagraphics SummaMouse, A-3
- Device coordinates
 - mapping units, 2-1
- Device driver
 - loading, 1-2
- Device drivers
 - component of PDI kit, 1-2
 - loading, 2-13
- Device identification
 - See Devid parameter
- Devid parameter
 - possible values, 2-7, 2-8
- DIGITAL
 - DECTouch VRTS1-A monitor, A-4
- Event Flag Number
 - PD\$EFN, 2-11
- Example
 - BASIC-PLUS-2 program, 4-7
 - FORTTRAN-77 program, 4-1
 - MACRO-11 program, 4-11
 - PASCAL program, 4-4
- Executing

INDEX

- your program, 2-13
- FORTTRAN-77
 - sample program, 4-1
- GBLDEF option
 - modifying in .CMD file, 2-11
- Get Device Coordinates (GETDVC)
 - positional device routine, 3-7
- Get Device Name (GETNAM)
 - positional device routine, 3-8
- GTCO
 - Micro Digi-Pad, A-3
- Identification
 - see device
- IE.ABO
 - status, 3-14
- IE.BAD
 - status, 3-2, 3-5, 3-14, 3-18
- IE.DAA
 - status, 3-2
- IE.DNA
 - status, 3-5, 3-14, 3-18
- IE.DUN
 - status, 3-2
- IE.FHE
 - status, 3-2, 3-14, 3-18
- IE.ONP
 - status, 3-2
- IE.TMO
 - status, 3-2, 3-5
- Input device area
 - aspect ratio, 2-2
 - definition, 2-1
 - shape, 2-1
 - Summagraphics MM961, 2-2
- Interface
 - see Positional Device Interface
- IS.SUC
 - status, 3-2, 3-4, 3-5, 3-6, 3-7, 3-8, 3-9, 3-11, 3-14, 3-18, 3-20, 3-43, 3-46
- Joysticks
 - Atari(c)-compatible, A-4
- Kit component
 - applications, 1-2
 - device drivers, 1-2
- Languages
 - used with kit, 1-3
- Library routines
 - calling, 2-10ff
- Linking programs
 - description, 2-11
- Loading
 - device drivers, 2-13
- Logical Unit Number
 - PD\$LUN, 2-11
- LUN
 - See Logical Unit Number
- MACRO-11
 - sample program, 4-11
- Mapping
 - device coordinate units, 2-1
- Micro Digi-Pad
 - GTCO, A-3
- Microsoft
 - mouse, A-4
- MM 1201
 - Summagraphics, A-2
- MM 961
 - Summagraphics, A-2
- Mouse
 - Microsoft, A-4
- Object module
 - linking with, 2-13
 - vs. cluster library, 2-13
- Overlay Descriptor Language File
 - See Descriptor File
- P/OS
 - versions, 2-12
- Parameter
 - button status, 2-10
 - passing mechanism, 3-1
- PASCAL
 - sample program, 4-4
- PDI
 - see Positional Device Interface
- PDI Library
 - description, 1-3
- PDLOBJ.OBJ
 - in .ODL file, 2-13
- Performance
 - improvement in, 2-13
- Ports
 - Communication, A-1

INDEX

- DECTouch, A-1
- DECTouch (Figure), 2-7
- Printer, A-1
- Positional Device
 - connecting to DECTouch ports, 1-1
 - connecting to XK0: or MK0:, 1-1
 - definition, 1-1
 - supported by PDI, 1-1
- Positional Device Interface
 - Application, 1-2
 - introduction, 1-1
 - kit components, 1-2
- Programming languages
 - See Languages
- Programs
 - sample, 4-1ff
- Read Configuration (REDCNF)
 - positional device routine, 3-11
- Read on Event (REDEVN)
 - positional device routine, 3-14
- Read Positional Report (REDRPT)
 - positional device routine, 3-18
- Request PDL operation (PDL)
 - positional device routine, 3-9
 - routine
 - ATTACH (ATTPD), 3-2
 - Cancel mouse AST (CNMAST), 3-4
 - DETACH (DETPD), 3-5
- Routines
 - Cancel Read on Event (EVNCAN), 3-6
 - Get Device Coordinates (GETDVC), 3-7
 - Get Device Name (GETNAM), 3-8
 - Read Configuration (REDCNF), 3-11
 - Read on Event (REDEVN), 3-14
 - Read Positional Report (REDRPT), 3-18
 - Request PDL operation (PDL), 3-9
 - Set Device Characteristics (SETCHR), 3-20
 - Specify Mouse AST (SPMAST), 3-43
 - Write Raw Data to Device (WRTDEV), 3-46
- Running
 - your program, 2-13
- Sample
 - command file, 2-12
 - .ODL file, 2-13
- Sample programs, 4-1ff
- Set Device Characteristics (SETCHR)
 - positional device routine, 3-20
- Shape
 - of input device area, 2-1
- Simple coordinate unit mapping, 2-3
- Sketchpad
 - altering RGB, B-5
 - application, 1-3
 - Box command, B-4
 - Cancel command, B-5
 - Circle command, B-4
 - Clear command, B-3
 - directions for using, B-1
 - Exit command, B-3
 - Fill command, B-5
 - Line command, B-3
 - Palette, B-5
 - Print command, B-4
 - screen display, B-2
 - Select command, B-4
 - Text command, B-4
- Specify Mouse AST (SPMAST)
 - positional device routine, 3-43
- Status
 - return, 3-2, 3-4, 3-5, 3-6, 3-7, 3-8, 3-9, 3-11, 3-14, 3-18, 3-20, 3-43, 3-46
- Success
 - button changed state, 3-14
 - timeout occurred, 3-14
 - x increment satisfied, 3-14
 - y increment satisfied, 3-14
- Summagraphics
 - MM 1201 digitizer, A-2
 - MM 961 digitizer, A-2
 - MM961 input device area, 2-2
 - SummaMouse, A-3
- SummaMouse
 - Summagraphics, A-3
- Synchronizing
 - device operations, 2-11
- Target machine
 - P/OS version on, 1-4
- Terminal screen

INDEX

- coordinates, 2-3
- Test the PDI application, 1-3
- Unit Number
 - See Logical Unit Number
- Units
 - mapping, 2-3, 2-4
- UNITS option
 - modifying in .CMD file, 2-11
- Using PDI Kit
 - procedural description, 1-3ff
- Version of P/OS
 - see P/OS
- VRTS1-A
 - DECTouch, A-4
- Write Raw Data to Device (WRTDEV)
 - positional device routine, 3-46

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 _____

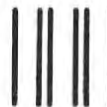
or

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 Workstations Publications
DECLIT AA PRO ED06A
Positional device interface programmer's manual 71

DECLIT AA PRO ED06A
Positional device interface
programmer's manual

SHREWSBURY LIBRARY
Digital Equipment Corporation
333 South Street SHR1-3/G18
Shrewsbury, MA 01545
(DTN) 237-3271

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

Cut Along Dotted Line

digital



097459