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

digital™

DIGITAL EQUIPMENT CORPORATION
Maynard. Massachusetts 01754-2571

First Printing, November 1985

CONTENTS

FIGURES

TABLES

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

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.

- **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.

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 postitional 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 postitional device for the Printer Port.

- **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.

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.

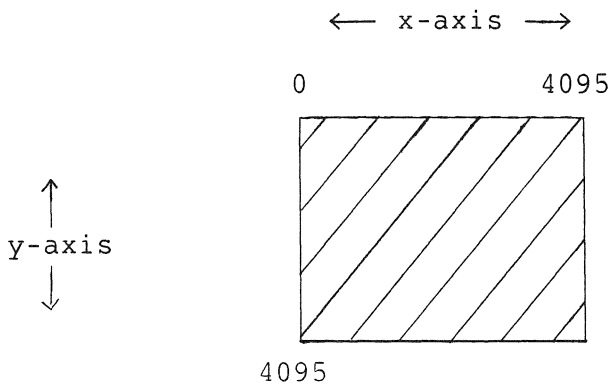

Figure 2-1: Square Input Device Area

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.

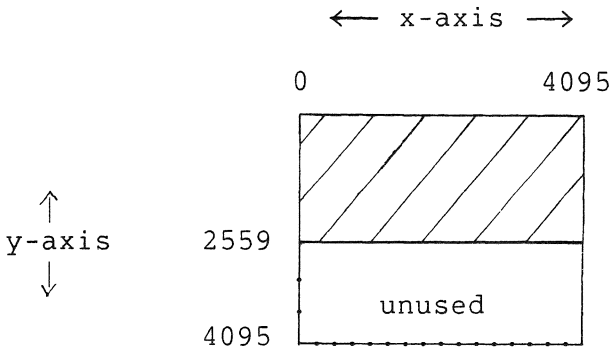

Figure 2-2: Summagraphics MM961 Input Device Area

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.

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.

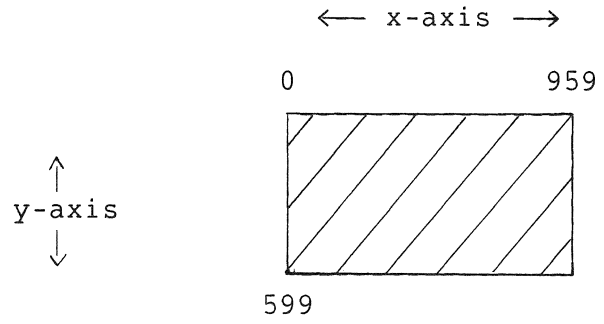

Figure 2-3: Coordinate Units of Terminal Screen

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.

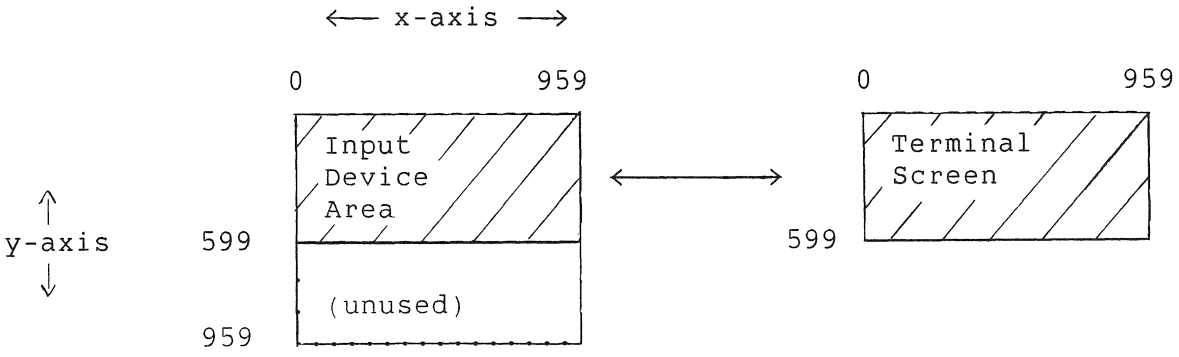

Figure 2-4: Simple Coordinate Units Mapping

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.

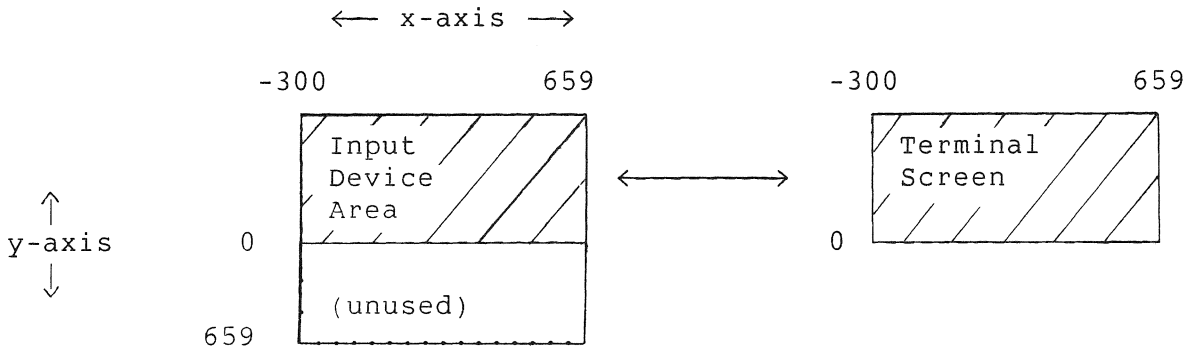

Figure 2-5: Alternative Coordinate Units Mapping

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.

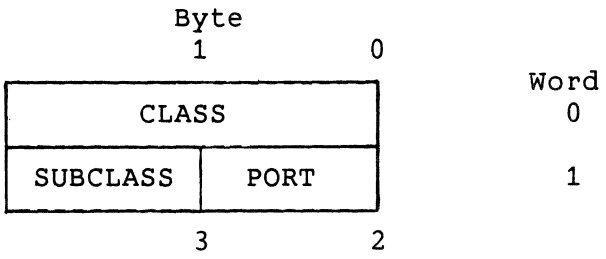

Figure 2-6: Device Identification Parameter

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.

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.

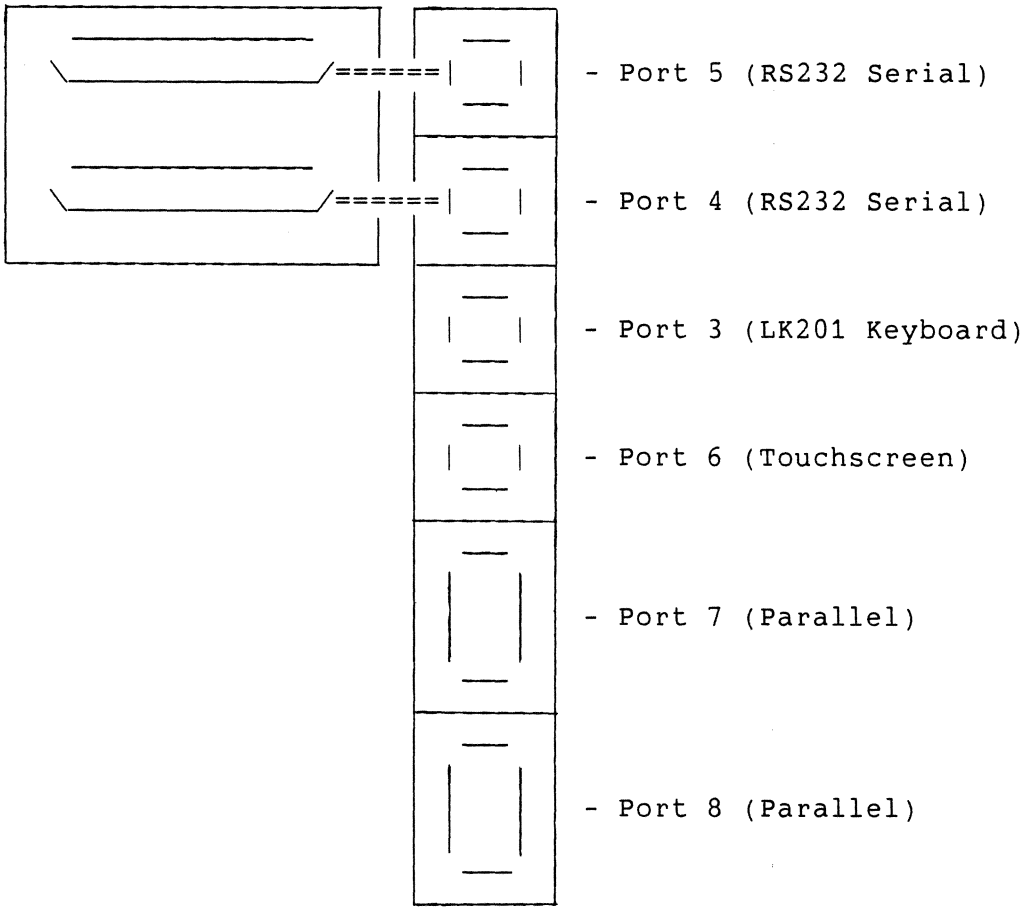

Figure 2-7: DECtouch Ports

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

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

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

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.

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.

```
        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.

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.

```
                .ROOT BASIC2-RMSROT-USER,RMSALL
(1)     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.

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.

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.

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).

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.

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.

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)

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.

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.

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: • 36 = No device connected. • 37 = Only the LK201 is connected. • 44 = Printer Port connected, LK201 disabled. • 45 = Printer Port connected, LK201 connected.

Word	Description
0	• 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.

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.

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.

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.

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:

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).

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.

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.

- **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.

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

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.

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

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.

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.

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.

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.

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 -521 -522 -523 -524	Invalid devid parameter. See Tables 2-2 and 2-3.

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)

Status Returns

+1	Success.
+2	Function not supported.
-500	One or more data values are out of range.
-520 -521 -522 -523 -524	Invalid devid parameter. See Tables 2-2 and 2-3.

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

Status Returns

+1	Success.
+2	Function not supported.
-500	One or more data values are out of range.
-520 -521 -522 -523 -524	Invalid devid parameter. See Tables 2-2 and 2-3.

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.

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.

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 - xminimum)/2) + x_minimum
y = ((y_maximum - yminimum)/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.

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	

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

Status Returns

+1	Success.
-500	One or more data values are out of range.
-520 -521 -522 -523 -524	Invalid devid parameter. See Table 2-2 and 2-3.

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 intrepreted 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.

Table 3-13: Data Values for RPMOD Characteristic

Value	Description
0	Absolute mode (default)
1	Relative mode
2	Device physical

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.

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 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.

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 specifed 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.

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 tranmit 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                               ; 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 ------------------------------------------------------------
```

```
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
        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
        INTEGER*2   status,xint,yint,button
c
c Declare variables for use as parameters in CGL calls:
c
        REAL        xreal,yreal
c
c Declare flag variable to control looping:
c
        LOGICAL         flag
c
c Initialize CGL core and invoke new frame:
c
        call CGL( GIC )                 !INITIALIZE_CORE
        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
```

```
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
```

```
                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                                                 }
```

```
{                .END                                                    }
{ ------------------------------------------------------------------     }
{       To run this program at DCL level:                                }
{ ------------------------------------------------------------------     }
{                                                                        }
{       $ INSTALL [ZZSYS]CGLFPU                                          }
{       $ INSTALL [ZZSYS]PASRES                                          }
{       $ INSTALL LB:[1,5]PDL                                            }
{                                                                        }
{                                                                        }
{ ------------------------------------------------------------------     }

program mouse;

{ This program uses a positional device to draw on                       }
{ the screen. The program loops indefinitely until aborted               }
{ or until it encounters a positional device error.                      }
{ The graphics are generated using the CORE Graphics Library (CGL).      }

%include 'lb:[1,5]CGLDEFS.PAS/NOLIST'

{ This instruction provides the file CGL.FTN                             }
{ that declares a set of integer constants                               }
{ corresponding to the names of the CGL instructions.                    }

{ Declarations -- external entry points for the PDL routines:            }

[external(ATTPD)]
procedure ATTACH_POSITIONAL_DEVICE( VAR status : integer ); SEQ11;

[external(REDRPT)]
procedure READ_POSITIONAL_REPORT( VAR status,
                                      xint,
                                      yint,
                                      button : integer ); SEQ11;

{ Declarations -- integer variables, for use as parameters in calls.     }
{                 real variables, for use as parameters in CGL calls.    }
{                 boolean variable, to control looping.                  }

var
    status,xint,yint,button     : integer;
    xreal,yreal                 : real;
    flag                        : boolean;

label
    900;

    begin

        { Initialize CGL core and invoke new frame: }
```

```
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;
```

```
                { 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
```

```
!       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$EFN:3       ; for Core Graphics Library
!       GBLDEF  = PD$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:
```

```
        !
        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
```

```
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
```

4.4 MACRO-11

```
        .TITLE  Example - POSITIONAL DEVICE EXAMPLE
        .IDENT  /V1.00/
        .ENABL  LC

        .MCALL  DIR$,QIOW$,EXIT$S

;+
; 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:
;
```

```
; 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 XCOOR
;    -----------------------------------------------------#R5 +6
;                  ADDRESS OF YCOOR
;    -----------------------------------------------------#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
```

```
        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 occured.  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$s                          ; 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,buf1l,,sd.gds>
Track:  QIOW$   IO.WSD,5,1,,,,<buf2,buf2l,,sd.gds>
Lines:  QIOW$   IO.WSD,5,1,,,,<buf3,buf3l,,sd.gds>

;+
;
; These are the data buffers for the TI: QIO's
;
;-

Buf1:   .BYTE   1,1                     ; Initialize
        .WORD   -1                      ; All subsystems
```

```
        .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

Buf1l   =.-Buf1


Buf2:   .BYTE   2,29.                   ; Set_Position
XMOVE:  .WORD   0                       ; X
YMOVE:  .WORD   0                       ; Y
Buf2l   =.-Buf2


Buf3:   .BYTE   2,25.                   ; Draw_Lines
Xdata:  .WORD   0                       ; X
Ydata:  .WORD   0                       ; Y
Buf3l   =.-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
```

```
        .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.

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 autmatically 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

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

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.

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.

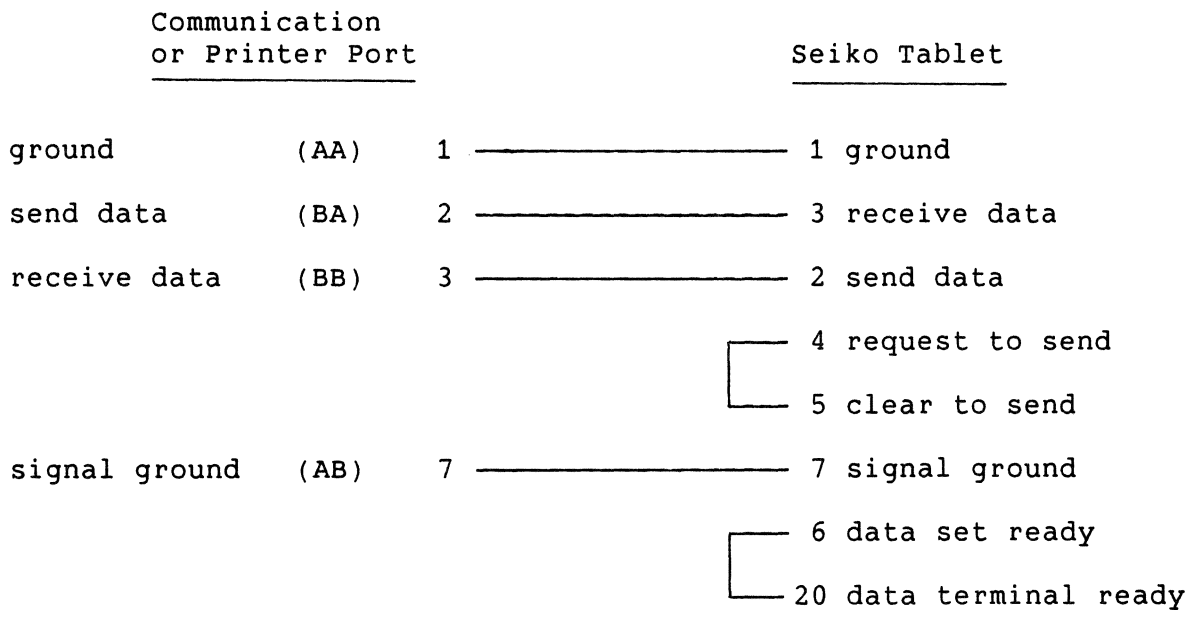

Figure A-1: Wiring for the Seiko Tablet

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

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.

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

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

A.8 SUMMAGRAPHICS 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.

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)

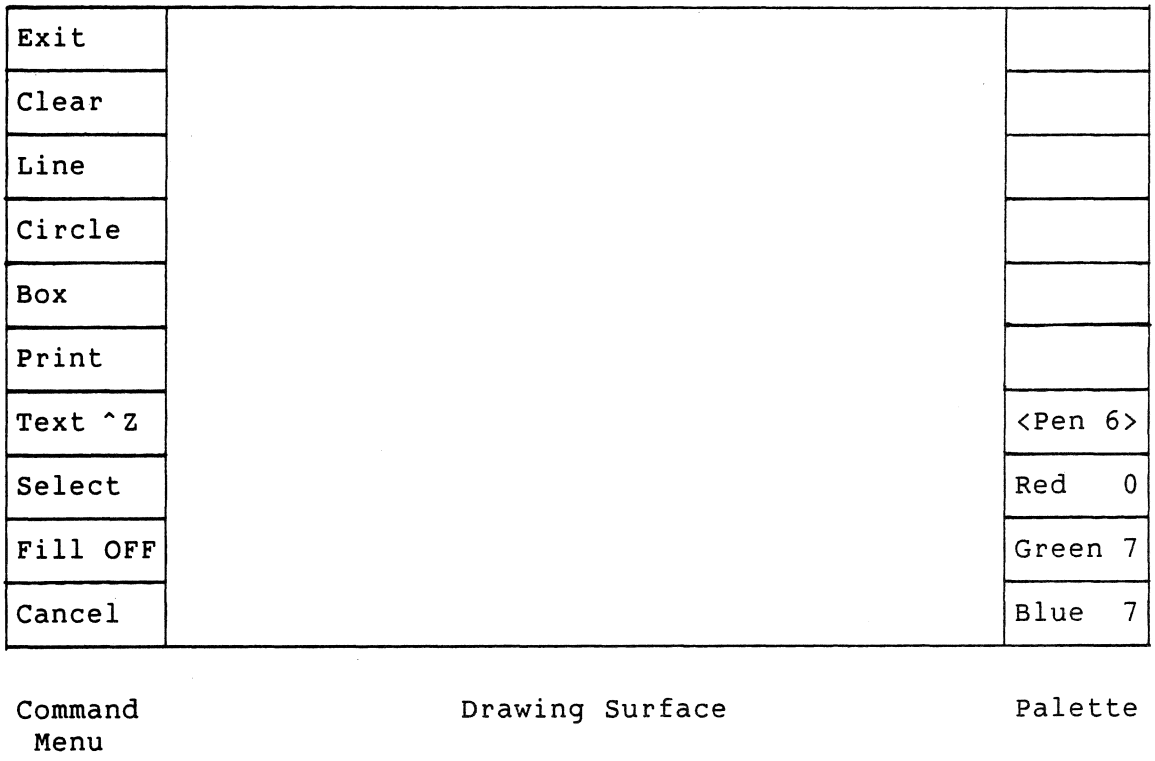

Figure B-1: Screen Display for Sketchpad Application

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.

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.

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.

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: • 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.

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