VAX/VMS CMU PTY Driver Reference Manual

Carnegie Mellon University

Computer Science Research Facility

Dale Moore

11 March 1988

Carnegie
Mellon

Preface

This document describes the use of the VAX/VMS Pseudo Terminal Driver as implemented Carnegie Mellon University Compter Science Department. Addtional informatin about VMS device drivers and the VMS operating system interface can be found in *VAX/VMS I/O User's Guide*, and in *VAX/VMS System Service Reference Manual*.

1. Pty Driver Features and Capabilities

Pseudo terminals (or Ptys) are virtual terminals in the system. Unlike normal terminals where the I/O is actually done with a physical device, Pty's I/O has no interface to any physical device. In this way they are much like mailboxes.

1.1 Concept of Control and Terminal Device

A single Pty consists of two devices, a control and a terminal device. The terminal device is the device that acts like a terminal.

The control device on VMS is named ''PYAx:''. For example, a pty control device could be named ''PYA1:''. The terminal device portion of the pty is name ''TZAx:''. The terminal device ''TZA1:'' is the mate to the control device ''PYA1:''. Similarly, the control device ''PYA5:'' is the mate to the terminal device ''TZA5:''.

1.1.1 Similarities Between Ptys and Terminals

The terminal portion of a pty behaves very much like a regular VAX/VMS terminal. The terminal portion of a pty has

- Type-ahead
- Specifiable or default line terminators
- Special operating modes, such as NOECHO and PASSALL
- American National Standard escape sequence detection
- Terminal/mailbox interaction
- Terminal control characters such as Control-S and Control-Q for starting and stopping output, Control-O for discarding output, and all other special characters that are handled by the VAX/VMS terminal driver.
- Limited full-duplex operation(simultaneously active read and write requests)

1.1.2 Differences Between Ptys and Terminals

The difference between a VAX/VMS terminal and the terminal portion of a pty is where the input comes from and where the output goes to. On a VAX/VMS terminal, the input comes from an actual terminal and the output goes to an actual terminal. On a pty the input comes from the control device and the output goes to the control device. In order to simulate someone typing at the terminal device ''TZA5:'' we must write to the control device ''PYA5:''. In order to read what has been typed out to the terminal device ''TZA5:'' we must read from the control device ''PYA5:''. Ptys are like terminals that only software can access.

There is no buffering of the input on the control device. It is possible to fill up the terminal device's typeahead buffer by writing large amounts of data to the control device.

Ptys are virtual devices. When allocating a pty, they behave much like mailboxes or network devices. To allocate a pty, simply allocate the first one ('PYA0:'). The allocating routine will be create a new pty and assign it to you. The pty will be deallocated when no process is referencing the device.

1.1.3 Uses of Ptys

There is no device from DEC which suffices for remote logins when using non-DECnet protocols. The remote terminal driver expects the remote node to do the local line editing. This includes delete, control-o, control-s, control-q, and much other stuff. Ptys handle the local line editing locally. When the line editing is done locally, the remote machine has to know much less about VMS and its terminal handling conventions.

2. Installing Ptys

In VAX/VMS V4, several pieces of software support the terminal interface. The hardware independent portion is SYS$SYSTEM:TTDRIVER.EXE. This version of ptys does not require any changes or patches to this code. There are other pieces of software for particular pieces of hardware. The code in SYS$SYSTEM:DZDRIVER.EXE is the device driver required for DZ11's and DZ32's. The code in SYS$SYSTEM:YCDRIVER.EXE is the device driver required for DMF32's. The pty drivers consist of two device drivers, TZDRIVER.EXE for the terminal portion of the pty's, and PYDRIVER.EXE for the control portion.

2.1 Compiling Sources

There are several pieces of source code that make up the pty drivers. PYDRIVER.MAR is the source for the control portion. TZDRIVER.MAR is the source for the terminal portion.

The commands for compiling and linking the devices are:

```
$ ! Compile the drivers
$ MACRO /LIST /OBJECT TZDRIVER.MAR+SYS$LIBRARY:LIB/LIBRARY
$ MACRO /LIST /OBJECT PYDRIVER.MAR+SYS$LIBRARY:LIB/LIBRARY
$ ! Link the drivers
$ LINK /SHARE /MAP /FULL /CROSS -
      TZDRIVER,SYS$SYSTEM:SYS.STB/SELECTIVE,SYS$INPUT:/OPTION
BASE=0
$ LINK /SHARE /MAP /FULL /CROSS -
      PYDRIVER,SYS$SYSTEM:SYS.STB/SELECTIVE,SYS$INPUT:/OPTION
BASE=0
```

3. Device Information

User processes can obtain terminal and control device information by using the $GETDVI, $GETCHN and $GETDEV system services (see *VAX/VMS System Services Reference Manual*).

It is recommended that new programs make use of the $GETDVI and $GETDVIW system services.

3.1 Pty Terminal Device Dependent Information

The information returned about a terminal device is in the same format as information returned about a regular VAX/VMS terminal. By only looking at the information returned from $GETCHN and $GETDEV system services, it is impossible to tell the difference between a pty terminal device and an physical terminal device.

3.2 Pty Control Device Dependent Information

When applied to a pty control device, $GETCHN return information in the format as show in Figure 3-1.

```
 31        24 23        16 15         8 7          0
-----------------------------------------------------
|              Device Characteristics               |
-----------------------------------------------------
|       Buffer Size        |    Type    |    Class  |
-----------------------------------------------------
|           Unused         |     PT Unit Number     |
-----------------------------------------------------
```

Figure 3-1: Pty Control Device Dependent Information

The first longword contains device-independent data. The second and third longwords contain device-dependent data.

3.2.1 Device Characteristics

The characteristics of the pty control device (PYA) can be found by either using the first longword returned by the $GETDEV and $GETCHN system services, or by using the item code DVI$_DEVCHAR to the $GETDVI system service.

A pty control device has the following characteristics.

- DEV$M_AVL - On line and available
- DEV$M_IDV - Capable of input
- DEV$M_ODV - Capable of output
- DEV$M_REC - Record oriented

A pty control device does **not** have the following characteristics.

- DEV$M_CCL - Carriage Control
- DEV$M_TRM - Terminal Device

3.2.2 Buffer Size

Also returned from $GETDVI by using item code DVI$_DEVBUFSIZ.

3.2.3 Device Type

The device type is DT$_PY. Since a pty is a nonstandard device, you will probably not find the correct macros, literals or constants for this device type. This field should have a value of 0.

Also returned from $GETDVI by using item code DVI$_DEVTYPE.

3.2.4 Device Class

The device class is DC$_PY. Since a pty seems to be such an odd creature, this field should be FF in hex, or 177 in octal.

Also returned from $GETDVI by using item code DVI$_DEVCLASS.

3.2.5 PT Unit Number

The unit number is the unit number of the associated pty terminal device.

The unit number is also returned as the low sixteen bits in DVI$_DEVDEPEND by $GETDVI.

4. Pty Function Codes

The function codes for the terminal device portion of a pty are exactly the same as those for regular VAX/VMS terminals. For more information on VAX/VMS terminal I/O see *VAX/VMS I/O User's Guide.*

The basic function of the control portion of a pty are read, write and set mode or characteristics.

A user does not need to have assigned the pty terminal device in order to do I/O operations on the pty control device. If the terminal device has type ahead enabled, sending the right characters at the control device will send a message to OPCOM to start running SYS$SYSTEM:LOGINOUT.EXE on the terminal device.

4.1 Read

The basic purpose of a pty control device read is to transfer data from the output buffer of the pty terminal device to a user specified buffer. There are three read functions which a user can apply to a pty control device.

- IO$_READVBLK - Read virtual block
- IO$_READLBLK - Read logical block
- IO$_READPBLK - Read physical block

A read is complete if either of the below conditions occur:

- The user specified buffer is full
- At least one character is available from the pty terminal device

The read function codes can take the following device/function dependent arguments:

- P1 = The starting virtual address of the buffer that is to receive the data
- P2 = The size of the buffer that is to receive the data in bytes
- P3, P4, P5, P6 = ignored

4.2 Write

The basic purpose of a pty control device write is to transfer data from the user specified buffer to the typeahead buffer of the pty terminal device. There are three write functions which a user can apply to a pty control device.

- IO$_WRITEVBLK - write virtual block
- IO$_WRITELBLK - write logical block
- IO$_WRITEPBLK - write physical block

The write function codes can take the following device/function dependent arguments:

- P1 = The starting virtual address of the buffer that is to be written to the pty terminal device
- P2 = The number of bytes that are to be sent
- P3, P4, P5, P6 = Ignored

4.3 Set Mode and Set Characteristics

The Set Mode function affects the mode and temporary characteristics of the associated pty control device. Set Mode is a logical I/O function and requires no privilege. A single function code is provided:

- IO$_SETMODE

The Set Charateristics function affects the permanent characteristics of the associated pty control device. Set Characteristics is a physical I/O function and requires the privilege necessary to perform physical I/O. A single function code is provided:

- IO$_SETCHAR

These functions take the following device/function dependent arguments if no function modifiers are specified:

- P1 = address of characteristics buffer
- P2, P3, P4, P5, P6 = ignored

The P1 argument points to a quadword block, as shown in Figure 4-1.

```
 31         24 23         16 15          8 7           0
-----------------------------------------------------
|       Buffer Size        |    Type    |    Class   |
-----------------------------------------------------
|                       Unused                       |
-----------------------------------------------------
```

Figure 4-1: Set Mode Characteristics Buffer

4.3.1 Function Modifiers

Function Modifiers to the control device currently do not affect the state or accessibility of the terminal device.

You cannot change the mode or characteristics of the terminal device by adding function modifiers to the I/O routines for the control device. An earlier version of the pty drivers supported this feature. This feature has not yet been added to this version.

4.4 Sense Mode and Sense Characterisitics

The two function codes to sense the mode of the pty control device are:

- IO$_SENSEMODE
- IO$_SENSECHAR

The IO$_SENSEMODE function returns the process-associated, that is temporary, characteristics of the pty control device. The IO$_SENSECHAR function returns the permanent characteristics of the pty control device. The IO$_SENSEMODE function is a logical function and requires no privilege. The IO$_SENSECHAR function is a physical function and requires the privilege necessary to perform physical I/O.

These function codes take the following device/function dependent.

- P1 = address of the quadword characteristics buffer
- P2, P3, P4, P5, P6 = ignored

The P1 argument points to a quadword block which is the same format as the Set Mode Characteristics Buffer, shown in Figure 4-1

4.5 I/O Status Block

The I/O status block formats for read, write, set mode and sense mode I/O functions are shown in Figures 4-2 and 4-3.

```
 31        24 23        16 15          8 7          0
-----------------------------------------------------
|       Byte Count         |         Status         |
-----------------------------------------------------
|                        Unused                     |
-----------------------------------------------------
```

Figure 4-2: IOSB Contents - Read and Write Function

```
 31        24 23        16 15          8 7          0
-----------------------------------------------------
|         Unused           |         Status         |
-----------------------------------------------------
|                        Unused                     |
-----------------------------------------------------
```

Figure 4-3: IOSB Contents - Set and Sense Mode and Characteristics Functions

The status indicates the succes or failure of the specified operation. Below are possible values for the status field:

- SS$_ABORT - The operation was canceled by the Cancel I/O on Channel ($CANCEL) system service. Applicable only if the driver was actively involved in an operation.
- SS$_NORMAL - Successful Completion
- SS$_ACCVIO - The specified buffer is not accessible to the specified process

5. Possible Improvements and Bugs

It may be useful for the associated mailbox of the PY device to receive a message every time the STOP, START, ABORT, RESUME, XOFF, or XON routine is called. This would permit programs that control the PY device to determine when to slow down the data rate, or abort the output in the buffers which have already been read.

It is not currently possible to affect the TZ device characteristics by any function on the PY device. It would be usefull to be able to modify the terminal device characteristic by performing some sort of I/O on the control device.

Table of Contents

List of Figures