DEC-11-ODDPA-A-D

P D P - 1 1

D E V I C E   D R I V E R   P A C K A G E

FOR MONITOR VERSION VØØ8A

October 1972

SOFTWARE SUPPORT CATEGORY

The software described in this document is supported by Digital Equipment Corporation under Category I, as defined on page ii of this document.

For additional copies, order No. DEC-11-ODDPA-A-D from Digital Equipment Corporation, Software Distribution Center, Maynard, Massachusetts 01754.

pdp11 SOFTWARE DISTRIBUTION CENTER

First Printing, October 1972

Your attention is invited to the last two pages of this document. The "How to Obtain Software Information" page tells you how to keep up-to-date with DEC's software. The "Reader's Comments" page, when filled in and mailed, is beneficial to both you and DEC; all comments received are acknowledged and are considered when documenting subsequent manuals.

This document is for information purposes and is subject to change without notice. DEC assumes no responsibility for the use or reliability of its software on equipment which is not supplied by DEC.

NEW AND CHANGED INFORMATION

This manual documents the software as of version VØ8 It has been revised to include all new and changed material since version VØ4. Such material is indicated by vertical bars in the outside margin. Whole new pages are not so marked but are dated in the lower outside corner.

SOFTWARE SUPPORT CATEGORIES

Digital Equipment Corporation (DEC) makes available four categories of software. These categories reflect the types of support a customer may expect from DEC for a specified software product. DEC reserves the right to change the category of a software product at any time. The four categories are as follows:

CATEGORY I
Software Products Supported at no Charge

This classification includes current versions of monitors, programming languages, and support programs provided by DEC. DEC will provide installation (when applicable), advisory, and remedial support at no charge. These services are limited to original purchasers of DEC computer systems who have the requisite DEC equipment and software products.

At the option of DEC, a software product may be recategorized from Category I to Category II for a particular customer if the software product has been modified by the customer or a third party.

CATEGORY II
Software Products that Receive Support for a Fee

This category includes prior versions of Category I programs and all other programs available from DEC for which support is given. Programming assistance (additional support), as available, will be provided on these DEC programs and non-DEC programs when used in conjunction with these DEC programs and equipment supplied by DEC.

CATEGORY III
Pre-Release Software

DEC may elect to release certain software products to customers in order to facilitate final testing and/or customer familiarization. In this event, DEC will limit the use of such pre-release software to internal, non-competitive applications. Category III software is only supported by DEC where this support is consistent with evaluation of the software product. While DEC will be grateful for the reporting of any criticism and suggestions pertaining to a pre-release, there exists no commitment to respond to these reports.

CATEGORY IV
Non-Supported Software

This category includes all programs for which no support is given

P R E F A C E

Within this document, Chapter 1 provides an introduction to device drivers in general; Chapter 2 outlines the established driver structure conventions and the driver's interface to a program using the driver's services; Chapter 3 illustrates methods by which standalone programs can communicate requests for service to the driver and access the results of such requests. Subsequent sections document each of the individual drivers. Each such section is preceded by a title page on red paper.

CONTENTS

CHAPTER 1

USING DEVICE DRIVERS OUTSIDE DOS

1.0 INTRODUCTION

Subroutines to handle I/O transfers between a PDP-11 and each of its peripheral devices are developed as required for use within the Disk Operating System (DOS). These subroutines are made available within an I/O Utilities Package for the benefit of PDP-11 users who have configurations unable to support DOS or who wish to run programs outside DOS control.

All the subroutines associated with one peripheral device form an entity known as a Driver. The Device Driver Package provides a general description of a driver and shows how it can be used in a stand-alone environment. The unique properties of each driver are discussed in separate documents issued as supplements to the Device Driver Package. The I/O Utilities Package for any system is determined by the peripherals of that system. Thus, the full documentation for a particular package consists of the Device Driver Package and applicable supplements.

CHAPTER 2

DRIVER FORMAT

2.1 STRUCTURE

The basic principle of all drivers under the DOS Monitor is that they must present a common interface to the routines using them in order to provide for device-independent operation. The subroutines are structured to meet this end. Moreover, the driver can be loaded anywhere in memory under Monitor control. Its code is always position-independent.

The detailed description of a driver is found in Appendix A. This section is concerned with driver interfaces.

2.1.1 Driver Interface Table

The first section of each driver consists of a table which contains, in a standard format, information on the nature and capabilities of the device it represents and entry points to each of its subroutines. The calling program can use this table as required, regardless of the device being called.

2.1.2 Setup Routines

Each driver is expected to handle its device under the PDP-11 interrupt system. When called by a program, therefore, a driver subroutine merely initiates the action required by setting the device hardware registers appropriately. It returns to the calling program by a standard subroutine exit.

The main setup routine prepares for a data transfer to or from the device, using parameters supplied by the calling program. Normally, blocks of data will be moved at each transfer. The driver will only return control to the program when the whole block has been transferred or when it is unable to continue because there is no more data available.

The driver can also contain subroutines by which the calling program can request start-up or shut-down action, such as leader or trailer code for a paper tape punch, or some special function provided by the device hardware (or a software simulation of that for some similar device), e.g., rewind of a magnetic tape or DECtape.

2.1.3 Interrupt Servicing

The nature of the driver routine to service device interrupts is particularly dependent upon the extent of the hardware provisions of the device for controlling transfers. In general, the driver determines the cause of the interrupt and checks whether the last action was performed correctly or was prevented by some error condition. If more device action is needed to satisfy the program request, the driver again initiates that action and takes a normal interrupt exit. If the program request has been fully met, control is returned to the program at an address supplied at the time of the request.

2.1.4 Error Handling

Device errors can be handled in two ways. There are some errors for which recovery can be programmed; the driver will, if appropriate, attempt this itself (as in the case of parity or timing failure on a bulk-storage device) or will recall the program with the error condition flagged (as at the end of a physical paper tape). Other errors normally require external action, perhaps by an operator. The driver calls a common error handler based on location 34 (IOT call) with supporting information on the processor stack to handle such errors.

2.2 INTERFACE TO THE DRIVER

2.2.1 Control Interface

The principal link between a calling program and any driver subroutine is the first word of the driver table. In order to provide the control parameters for a device operation, the calling program prepares a list in a standardized form and places a pointer to the list in the driver link. The called driver uses the pointer to access the parameters. If the driver need return status information, it can place it in the list area via the link-word.

The first word of the driver can also act as a busy indicator in that while it remains 0 the driver is not currently performing some task, whereas when the first word contains a list-pointer the driver can be assumed to be busy. Since most drivers support only one job at a time, the link-word state is significant.

2.2.2 Interrupt Interface

Although the driver expects to use the interrupt system, it does not itself ensure that its interrupt vector in the memory area below 400_8 has been set up correctly; the Monitor under DOS takes care of this. However, the Driver Table contains the information required to initialize the appropriate vector.

CHAPTER 3

STAND-ALONE USE

Because each driver is designed for operation within the device-independent framework of DOS Monitor, it can be similarly used in other applications. Since the easiest way to use the driver is to assemble it with the program which requires it, this method will be described first. Other possible methods will be discussed later.

3.1 DRIVER ASSEMBLED WITH PROGRAM

3.1.1 Setting Interrupt Vector

As noted in paragraph 2.2.2, the calling program must initialize the device transfer vector within memory locations 0-377. The address of the driver's interrupt entry point can be identified on the source listing by the symbolic name which appears as the content of the Driver Table Byte, DRIVER+5. The priority level at which the driver expects to process the interrupt is at byte DRIVER+6. For a program which can use position-dependent code, the setup sequence might be:

```
MOV     #DVRINT, VECTOR         ;SET INT. ADDRESS
MOVB    DRIVER+6, VECTOR+2      ;SET PRIORITY
CLRB    VECTOR+3                ;CLEAR UPPER STATUS BYTE
```

(where the Driver Table shows at DRIVER+5: .BYTE DVRINT-DRIVER).

If the program must be position-independent, it can take advantage of the fact that the Interrupt Entry address is stored as an offset from the start of the driver, as illustrated above. In this case, a sample sequence might be:

```
MOV     PC,R1                   ;GET DRIVER START
ADD     #DRIVER-.,R1
MOV     #VECTOR,R2              ;...& VECTOR ADDRESSED
CLR     @R2                     ;SET INT. ADDRESS
MOVB    5(R1),@R2               ;...AS START ADDRESS+OFFSET
ADD     R1,(R2)+
CLR     @R2                     ;SET PRIORITY
MOVB    6(R1),@R2
```

3.1.2 Parameter Table for Driver Call

For any call to the driver, the program must provide a list of control arguments mentioned in paragraph 2.2.1. This list must adhere to the following format[1]:

```
[SPECIAL FUNCTION POINTER]²
[BLOCK NO.]³
STARTING MEMORY ADDRESS FOR TRANSFER
NO. OF WORDS to be transferred (2's complement)
STATUS CONTROL showing in Bits:

   0-2:   Function (octally 2=WRITE, 4=READ)⁴
   8-10:  Unit (if Device can consist of several,
            e.g., DECtape)
   11:    Direction for DECtape travel (0 = Forward)

ADDRESS for RETURN ON COMPLETION
[RESERVED FOR DRIVER USE]⁵
```

The list can be assembled in the required format if its content will not vary. The driver can return information in this area as described in a later paragraph; however, this will not corrupt the program data and it is cleared by the driver before it begins its next operation.

On the other hand, most programs will probably use the same list area for several tasks or even for different drivers. In this case, the program must contain the necessary routine to set up the list for each task before making the driver call, perhaps as illustrated in the next paragraph. It must be noted, however, that the driver may refer to the list again when it is recalled by an interrupt or to return information to the calling program. Therefore, the list must not be changed until any driver has completed a function requested; for concurrent operations, different list areas must be provided.

---

[1] In some cases, it can be further extended as discussed in later paragraphs.

[2] Required only if Driver is being called for Special Function; addresses a Special Function Block.

[3] Required only if the Device is bulk storage (e.g., Disk or DECtape).

[4] Most devices transfer words regardless of their content, i.e., ASCII or Binary. Some devices (e.g., Card Reader) may be handled differently depending on the mode for these, Bit 0 must also be set to indicate ASCII=0, Binary=1. In these cases, the driver always produces or accepts ASCII even though the device itself uses some other code.

[5] This word may be omitted if the device is bulk storage (see below).

3.1.3 Calling the Driver

To enable the driver to access the parameter list, the program must set the first word of the driver to an address six bytes less than that of the word containing MEMORY START ADDRESS. It can then directly call the driver subroutine required by a normal JSR PC,xxxx call.

As an example, the following position-independent code might appear in a program which wishes to read Blocks #100-103 backward from DEC-tape unit 3 into a buffer starting at address BUFFER:

```
        MOV     PC,RØ               ;GET TABLE ADDRESS
        ADD     #TABLE+12-.,RØ
        MOV     PC,@RØ              ;GET AND STORE...
        ADD     #RETURN-.,@RØ       ;...RETURN ADDRESS
        MOV     #5404,-(RØ)         ;SET READ REV. UNIT 3
        MOV     #-1024.,-(RØ)       ;4 BLOCKS REQUIRED
        MOV     PC,-(RØ)            ;GET AND STORE
        ADD     #BUFFER-.,@RØ       ;...BUFFER ADDRESS
        MOV     #103,-(RØ)          ;START BLOCK
        CMP     -(RØ),-(RØ)         ;SUBTRACT 4 FROM POINTER
        MOV     RØ,DT               ;SET DRIVER LINK
        JSR     PC,DT.TFR           ;GOTO TRANSFER ROUTINE
  WAIT:  .                          ;RETURNS HERE WHEN
         .                          ;...TRANSFER UNDER WAY
         .                          ;RETURNS HERE WHEN
         .                          ;...TRANSFER COMPLETE
 TABLE: .WORD Ø                     ;LIST AREA SET
        .WORD Ø                     ;...BY ABOVE SEQUENCE
        .WORD Ø
        .WORD Ø
        .WORD Ø
```

3.1.4 User Registers

During its setup operations for the function requested, the driver assumes that Processor Registers 0-5 are available for its use. If their contents are of value, the program must save them before the driver is called.

While servicing intermediate interrupts, the driver may need to save or restore its registers. It expects to have two subroutines available for the purpose (provided by the Monitor under DOS). It accesses them via addresses in memory locations 44_8 (S.RSAV) for saves and 46_8 (S.RRES for restores) using the sequence:

```
        MOV     @#44,-(SP)          ;OR 'MOV    @#46,-(SP)
        JSR     R5,@(SP)+
```

It must also ensure that their start addresses are set into the correct locations (44_8 and 46_8).

At its final interrupt, the driver saves the contents of Registers 0-5 before returning control to the calling program completion return.

3.1.5 Returns From Driver

As shown in the example in paragraph 3.1.3, the driver returns control to the calling program immediately after the JSR as soon as it has set the device in motion. The program can wait or carry out alternative operations until the driver signals completion by returning at the address specified (i.e., RETURN above). Prior to this, the program must not attempt to access the data being read in, nor refill a buffer being written out.

The program routine beginning at address RETURN varies according to the device being used. In general, the driver has given control to the routine for one of two reasons; namely, the function has been satisfactorily performed, or it cannot be carried out due to some hardware failure with which the driver is unable to cope, though the program may be able to do so. In the latter case, the driver uses the STATUS word in the program list to show the cause:

Bit 15 = 1	indicates that a device parity or timing failure occurred and the driver has not been able to overcome this, perhaps after several attempts.
Bit 14 = 1	shows that the end of the available data has been reached.

The driver places in R0 the content of its first word as a pointer to the list concerned.

In addition, the driver can have transferred only some of the data requested. In this case, it will show in the RESERVED word of the program list a negative count of the words not transferred in addition to setting Bit 14 of the STATUS word. As mentioned in the note in paragraph 3.1.2, this applies only to non-bulk storage devices. The drivers for DECtape or Disks[1] always endeavor to complete the full transfer, even beyond a parity failure, or they take more drastic action (see paragraph 3.1.6).

---

[1]This includes RF11 Disk; although this is basically word-oriented, it is assumed to be subdivided into 64-word blocks.

It is thus the responsibility of the program RETURN routine to check the information supplied by the driver in order to verify that the transfer was satisfactory and to handle the error situations appropriately.

In addition, the routine must contain a sequence to take care of the Processor Stack, Registers, etc. As noted earlier, the driver takes the completion return address after an interrupt and has saved Registers 0-5 on the stack above the Interrupt Return Address and Status. The program routine should, therefore, contain some sequence to restore the processor to its state prior to such interrupt, e.g., using the same Restore subroutine illustrated earlier:

```
MOV     @#46,-(SP)          ;CALL REGISTER RESTORE
JSR     R5,@(SP)+
 :
 :
RTI                         ;RETURN TO INTERRUPTED PROG.
```

3.1.6 Irrecoverable Errors

All hardware errors other than those noted in the previous paragraph are more serious in that they cannot normally be overcome by the program or by the driver on its behalf. Some of these could be due to an operator fault, such as neglecting to turn a paper tape reader to on or to set the correct unit number on a DECtape transport. Once the operator has rectified the problem, the program could continue. Other errors, however, will require hardware repair or even software repair, e.g., if the program asks for Block 2000 on a device having a maximum of 1000. In general, all these errors will result in the driver placing identifying information on the processor stack and calling IOT to produce a trap through location 34_8.

Under DOS, the Monitor provides a routine to print a teleprinter message when this occurs. In a stand-alone environment, the program using the driver must itself contain the routine to handle the trap (unless the user wishes to modify the driver error exits before assembly). The handler format will depend upon the program. Should it wish to take advantage of the information supplied by the driver, the format is as follows:

(SP):	Return Address	Stored by IOT Call
2 (SP):	Return Status	
4 (SP):	Error No. Code	generally unique to driver
5 (SP):	Error Type Code:	1 = Recoverable after Operator Action 3 = No recovery
6 (SP):	Additional Information	such as content of Driver, Control Register, Driver Identity, etc.

As a rule, the driver will expect a return following the IOT call in the case of errors in Type 1 but will contain no provision following a return from Type 3.

3.1.7 General Comment

The source language of each driver has been written for use with particular,.
which will not be accepted by the Paper Tape Software PAL-11R, in particular, .TITLE, .GLOBL, and Conditional Assembly directives. Such statements should be deleted before the source is used. Similarly, an entry in the driver table gives the device name as .RAD50 'DT' to obtain a specially packed format used internally by DOS. If the user wishes to keep the name, for instance, for identification purposes as discussed in section 3.3, .RAD50 might easily be changed to .ASCII without detrimental effect, or it might be replaced with .WORD Ø.

3.2 DRIVERS ASSEMBLED SEPARATELY

Rather than assemble the driver with every program requiring its availability, the user may wish to hold it in binary form and attach it to the program only when loaded. This is readily possible; the only requirement is that the start address of the driver should be known or be determinable by the program.

The example in paragraph 3.1.2 showed that the Interrupt Servicing routine can be accessed through an offset stored in the Driver Table. The same technique can be used to call the setup subroutines, as these also have corresponding offsets in the Table, as as follows:

DRIVER+7	Open[1]
+1Ø	Transfer
+11	Close[1]
+12	Special Functions[1]

[1] If the routine is not provided, these are 0.

The problem is the start address. There is the obvious solution of assembling the driver at a fixed location so that each program using it can immediately reference the location chosen. This ceases to be convenient when the program has to avoid the area occupied by the driver. A more general method is to relocate the driver as dictated by the program using it, thus taking advantage of the position-independent nature of the driver. The Absolute Loader, described in the Paper Tape Software Handbook (DEC-11- Chapter 6, provides the capability of continuing a load from the point at which it ended. Using this facility to enter the driver immediately following the program, the program might contain the following code to call the subroutine to perform the transfer illustrated in paragraph 3.1.3:

```
          MOV     PC,R1               ;GET DRIVER START ADDRESS
          ADD     #PRGEND-.,R1
          MOV     PC,RØ               ;GET TABLE ADDRESS
          ADD     #TABLE+12-.,RØ      ;AND SET UP AS SHOWN
           .                          ;...IN SECTION 3.1.3
           .
           .
          CMP     -(RØ),-(RØ)         ;FINAL POINTER ADJUSTMENT
          MOV     RØ,@R1              ;STORE IN DRIVER LINK
          CLR     -(SP)               ;GET BYTE SHOWING...
          MOVB    1Ø(R1),@SP          ;...TRANSFER OFFSET
          ADD     (SP)+,R1            ;COMPUTE ADDRESS
          JSR     PC,@R1              ;GO TO DRIVER
           .
           .
           ,
 PRGEND:
          .END
```

This technique can be extended to cover situations in which several drivers are used by the same program, provided that it takes account of the size of each driver (known because of prior assembly) and the drivers themselves are always loaded in the same order.

For example, to access the second driver, the above sequence would be modified to:

```
          MOV     PC,R1               ;GET DRIVER 1 ADDRESS
          ADD     #PRGEND-.,R1
          ADD     #DVR1SZ,R1          ;STEP TO DRIVER 2
           .
           .
           .
 DVR1SZ=n
 PRGEND:
          .END
```

An alternative method may be to use the Relocatable Assembler PAL-11S in association with the Linker program LINK-11S, both of which are available through the DECUS Library. The start address of each driver is identified as a global. Any calling program need merely include a corresponding .GLOBL statement, e.g., .GLOBL DT.

3.3 DEVICE-INDEPENDENT USAGE

As mentioned earlier, the drivers are assigned for use in a device-independent environment, i.e., one in which a calling program need not know in advance which driver has been associated with a table for a particular execution run. One application of this type might be to allow line printer output to be diverted to some other output medium because the line printer is not currently available. Another might be to provide a general program to analyze data samples although these on one occasion might come directly from an Analog-to-Digital converter and on another be stored on a DECtape because the sampling rate was too high to allow immediate evaluation.

Programs of this type should be written to use all the facilities that any one device might offer, but not necessarily all of them. For instance, the program should ask for start-up procedures because it may sometime use a paper tape punch which provides them, even though it may normally use DECtape which does not. As noted in paragraph 2.1.1, the driver table contains an indication of its capabilities to handle this situation. The program can thus examine the appropriate item before calling the driver to perform some action. As an example, the code to request start-up procedures might be (assuming R0 already set to List Address):

```
        MOV     #DVRADD,R1          ;GET DRIVER ADDRESS
        TSTB    2(R1)               ;BIT 7 SHOWS...
        BPL     NOOPEN              ;...OPEN ROUTINE PRESENT
        MOV     RØ,@R1              ;STORE TABLE ADDRESS
        CLRB    -(SP)               ;BUILD ADDRESS
        MOVB    7(R1),@SP           ;...OF THIS ROUTINE
        ADD     (SP)+,R1
        JSR     PC,CR1              ;...AND GO TO IT
                                    ;FOLLOWED POSSIBLY BY
                                    ;WAIT AND COMPLETION
                                    ;PROCESSING
NOOPEN:                             ;RETURN TO COMMON OPERATION
```

Similarly, the indicators show whether the device is capable of performing input or output, or both; whether it can handle ASCII or binary data; whether it is a bulk storage device capable of supporting a directory structure or is a terminal-type device requiring special treatment, and the like. Other table entries show the device name as identification and how many words it might normally expect to transfer at a time (in 16-word units). All of the information can readily be examined by the calling program, thus enabling the use of a common call sequence for any I/O operation, as for example

```
         MOV    #DVRADR,R5          ;SET DRIVER START
         JSR    R5,IOSUB            ;CALL SET UP SUB
         BR     WAIT                ;SKIP TABLE FOLLOWING ON RETURN
         .WORD  1Ø                  ;TRANSFER REQUIRED
         .WORD  1Ø3                 ;BLOCK NO.
         .WORD  BUFFER              ;BUFFER ADDRESS
         .WORD  -256                ;WORD COUNT
         .WORD  4Ø4                 ;READ FROM UNIT 1
         .WORD  RETURN              ;EXIT ON COMPLETION
         .WORD  Ø                   ;RESERVED
WAIT:                               ;CONTINUE HERE...
         .                          ;WHILE TRANSFER IN PROGRESS
         .
         .
         .
IOSUB:   MOV    @SP,RØ              ;PICK UP DRIVER ADDR
         MOV    R5,R1               ;SET POINTER TO LIST
         TST    (R1)+               ;BUMP TO COLLECT CONTENT
         .                          ;ROUTINE CHECKS ON DEVICE
         .                          ;..CAPABILITY USING R1
         .                          ;..TO ACCESS LIST AND
         .                          ;..RØ THE DRIVER TABLE
         .                          ;IF O.K...
         MOV    @R1,R1              ;GET ROUTINE OFFSET
         ADD    RØ,R1
         CLR    -(SP)               ;USE IT TO BUILD
         MOVB   @R1,@SP             ;...ENTRY POINT
         ADD    RØ,@SP
         JSR    PC,@(SP)+           ;CALL DRIVER
         RTS    R5                  ;EXIT TO CALLER
```

The calling program, or a subroutine of the type just illustrated, may also wish to take advantage of a feature mentioned earlier: the fact that when a driver is in use its first word will be non-zero. The driver itself does not clear this word except in special cases shown in the description for the driver concerned. If the program itself always ensures that it is set to zero between driver tasks, this word forms a suitable driver-busy flag. Under DOS, the program parameter list is extended to allow additional words to provide linkage between lists as a queue of which the list indicated in the driver first word is the first link.

The preceding paragraphs are intended to indicate possible ways of incorporating the drivers available into the type of environment for which they were designed. The user will probably find others. However, he should carefully read the more detailed description of the driver structure in Appendix A, and the individual driver specifications before determining the final form of his program.

A word of warning is appropriate here. Although most drivers set up an operation and then wait for an interrupt to produce a completion state, there are some cases in which the driver can finish its required task without an interrupt, e.g., "opening" a paper tape reader involves only a check on its status. Moreover, where "Special Functions" are concerned, the driver routine may determine from the code specified that the function is not applicable to its device and, therefore, will have nothing to do. In such cases, the driver clears the intermediate return address from the processor stack and immediately takes the completion return. Special problems can arise, however, if the driver concerned is servicing several tasks, any of which can causes a queue for the driver's services under DOS. To overcome these problems, the driver expects to be able to refer to flags outside the scope of the list so far described. This can mean that a program using such a driver may also need to extend the list range to cover such possibilities. Particular care should be exercised in such cases.

APPENDIX A

I-O DRIVERS WITHIN THE DISK OPERATING SYSTEM

The principal function of an I/O driver is to satisfy a Monitor processing routine's requirement for the transfer of a block of data in a standard format to or from the device it services. This will involve both setting up the device hardware registers to cause the transfer and its control under the interrupt scheme of PDP-11, making allowance for peculiar device characteristics (e.g., conversion to or from ASCII if some special code is used).

It may also include routines for handling device start-up or shut-down such as punching leader or trailer, and for making available to the user certain special features of the device, such as rewind of magtape.

A.1 Driver Structure

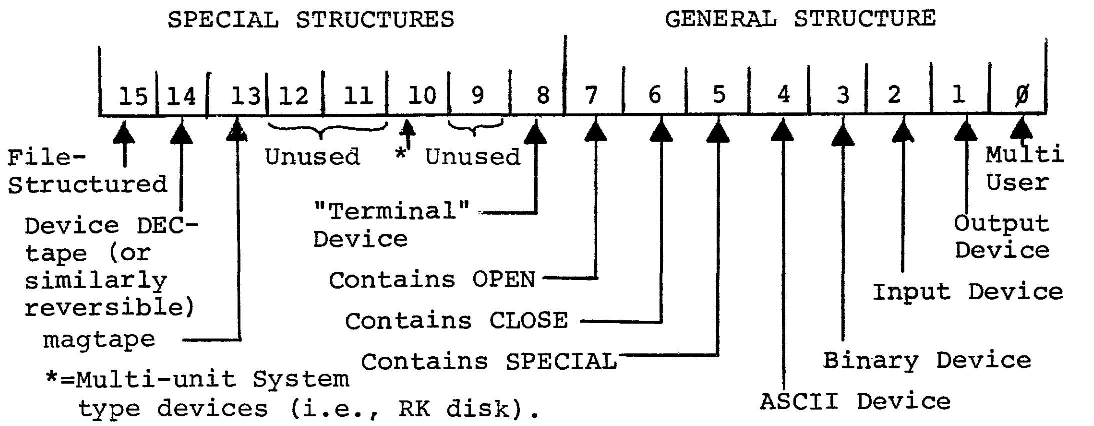
In order to provide a common interface to the monitor, all drivers must begin with a table of identifying information as follows:

DVR:

BUSY FLAG (initially Ø)	
FACILITY INDICATOR (expanded below)	
Offset to Interrupt Routine*	Standard Buffer Size in 16-word Units.
Offset to OPEN Routine *	Priority for Interrupt Service
Offset to CLOSE Routine *	Offset to Transfer Routine *
Space	Offset to Special Functions*
DEVICE NAME (Packed Radix-5Ø)	

Offsets marked * will enable calling routine to indicate routine required. They will be considered to be an unsigned value to be added to the start address of the driver. This may mean that with a 256-word maximum, the instruction referenced by the offset will be JMP or BR (routine).

Bits in the Facility Indicator Word define the device for monitor reference:



The table should be extended as follows if the device is file-structured:

BLOCK USED AS MASTER FILE DIRECTORY	
POINTER TO BIT-MAP IN MEMORY	Unit Ø
	Similar Bit-Map Pointers for Multi-unit Devices

The driver routines to set up the transfer and control it under interrupt, and possibly for OPEN, CLOSE, and SPECIAL, follow the table. Their detailed operation will be described later.

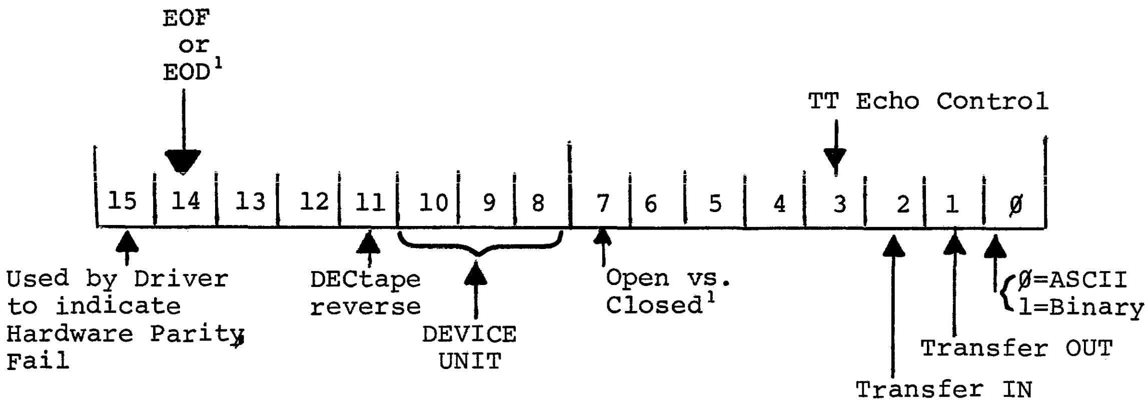
A.2 Monitor Calling

When a Monitor I/O processing routine needs to call the driver, it first sets up the parameters for the driver operation in relevant words of the appropriate DDB[1], as follows:

XYZ:	-	(User Call Address)
	SPECIAL FUNCTION CODE	(User Line Address)
	DEVICE BLOCK NUMBER	
	MEMORY START ADDRESS	
	WORD COUNT (2's Complement)	
	TRANSFER FUNCTIONS (expanded below)	
	COMPLETION RETURN ADDRESS	
	(DRIVER WORD-COUNT RETURN) Set to Zero	

[1]Dataset Data Block - in full, a 16-word table which provides the main source of communication between the Monitor drivers and a particular set of data being processed on behalf of a using program.

The relevant content of the Transfer Function word is as follows:



Provided that the Facility Indicator in the Driver Table described above shows that the driver is able to satisfy the request, both from the point of view of direction and mode and of the service required, the Monitor routine places in Register 1 the relative byte address of the entry in the Driver Table containing the offset to the routine to be used (e.g., for the Transfer routine, this would be 1Ø). It then calls the Driver Queue Manager, using JSR PC,S.CDB.

The Driver Queue Manager assures that the driver is free to accept the request, by reference to the Busy Flag (Word Ø of the driver table). If this contains Ø, the Queue Manager inserts the address of the DDB from Register Ø and jumps to the start of the routine in the driver using Register 1 content to evaluate the address required. If the driver is already occupied, the new request is placed in a queue linking the appropriate DDB's for datasets waiting for the driver's services. It is taken from the queue when the driver completes its current task. (This is done by a recall to the Queue Manager from the routine just serviced, using JSR PC,S.CDQ.)

On entry to the Driver Routine, therefore, the address following the Monitor routine call remains as the "top" element of the processor stack. It can be used by the driver in order to make an immediate return to the Monitor (having initiated the function requested), using RTS PC. It should also be noted that the Monitor routine will have saved register contents if it needs them after the device action. The driver may thus freely use the registers for its own operations.

[1] Note that bits 7 and 14 are undefined in DOS Monitors which precede VØØ8.

When the driver has completely satisfied the Monitor request, it should return control to the Monitor using the address set into the DDB. On such return, Register Ø must be set to contain the address of the DDB just serviced and since the return will normally follow an interrupt, Registers Ø-5 at the interrupt must be stored on top of the stack.

A.3 Driver Routines

A.3.1 TRANSFER

The sole purpose of the TRANSFER routine is to set the device in motion. As indicated above, the information needed to load the hardware registers is available in the DDB, whose address is contained in the first word of the driver. Conversion of the stored values is, of course, the function of the routine. It must also enable the interrupt; however, it need not take any action to set the interrupt vectors as these will have been preset by the Monitor when the driver is brought into core. Having then given the device GO, an immediate return to the calling processor should be made by RTS PC.

A.3.2 Interrupt Servicing

The form of this routine depends upon the nature of the device. In most drivers it will fall into two parts, one for handling the termination of a normal transfer and the other to deal with reported error conditions.

For devices which are word or byte-oriented, the routine must provide for individual word or byte transfers, with appropriate treatment of certain characters (e.g., TAB or Null) and for their conversion between ASCII or binary and any special device coding scheme, until either the word count in the DDB is satisfied or an error prevents this. On these devices, the most likely cause for such error is the detection of the end of the physical medium; its treatment will vary according to whether the device is providing input or accepting output. The calling program will usually need to take action in the former case and the driver should merely indicate the error by returning the unexpired portion of the word count in DDB Word 7 on exit to the Monitor. Output End of Data, however, will, in general, require operator action. To obtain this, the driver should call the Error Diagnostic Print routine within the Monitor by:

```
        MOV     DEVNAM,-(SP)        ;SHOW DEVICE NAME
        MOV     #4Ø2,0(SP)          ;SHOW DEVICE NOT READY
        IOT                         ;CALL ERROR DIAGNOSTIC PRINT ROUTINE
```

On the assumption that the operator will reset the device for further output and request continuation, the driver must follow the above sequence with a Branch or Jump to produce the desired resumption of the transfer.

Normal transfer handling on blocked devices (or those like RF11 Disk which are treated as such) is probably simpler since the hardware takes care of individual words or bytes and the interrupt only occurs on completion. Errors may arise from many more causes, and their handling is, as a result, much more complex and device dependent. In general, those which indicate definite hardware malfunctions must lead to the situation in which the operator must be informed by diagnostic message and the only recourse after rectification will be to start the program over.

At the other end of the scale there are errors which the driver itself can attempt to overcome by restarting the transfer - device parity failure on input is a common example. If a retrial, or several, still does not enable a satisfactory conclusion, the driver should normally allow programmed recovery and merely indicate the error by Bit 15 of DDB word 5. Nevertheless, because the program may wish to process the data despite the error, the driver should attempt to transfer the whole block requested if this has not already been effected. Between these two extremes, the remaining forms of error must be processed according to the type of recovery deemed desirable.

Whether the routine uses processor registers for its operation or not will naturally depend on considerations of the core space saved against the time taken to save the user's content. However, on completion (or error return to the Monitor), as indicated in an earlier paragraph, the calling routine expects the top of the stack to contain the contents of Registers Ø-5 and Register Ø to be set to the address of the DDB just serviced. The driver must, therefore, provide for this.

A.3.3 OPEN

This routine need be provided only for those devices for which some hardware initialization by the user is required. It should not normally appear in drivers for devices used in a file-oriented manner. Its presence must be indicated by the appropriate bit (Bit 7) in the driver table Facility Indicator.

The routine itself may vary according to the transfer direction of the device. For output devices, the probable action required is the transmission of appropriate data, e.g., CR/LF at a keyboard terminal, form-feed at a printer, or null characters as punched leader code, and for this a return interrupt is expected. The OPEN routine should then be somewhat similar to that for TRANSFER in that it merely sets the device going and makes an interim return via RTS PC, waiting until completion of the whole transmission before taking the final return address in the DDB.

On the other hand, an input OPEN will likely consist of just a check on the readiness of the device to provide data when requested. In this case, the desired function can be effected without any interrupt wait. The routine should, therefore, take the completion return immediately. Nevertheless, it must ensure that the saved PC value on top of the stack from the call to S.CDB is appropriately removed before exit. In the case of drivers which can only service one dataset at a time (i.e., Bit Ø of their Facility Pattern word is set to Ø) and can never, therefore, be queued; it will be sufficient to use TST (SP)+ toeffect this. A multi-user driver, however, must allow for the possibility that it may be recalled to perform some new task waiting in a queue. This is shwon by the byte at DDB-3 being non-zero. In this case, the intermediate return to the routine originally requesting the new task has already been made directly by S.CDB. The address now on top of the stack is the return to the routine, whose task the driver has just completed and which has called S.CDQ to dequeue the driver. This return must be taken when the first routine has performed its Completion Return processing. Moreover, this first routine expects to exit as from an interrupt. When a driver is recalled from a queue, it must simulate this interrupt. A possible sequence might be:

```
            MOV    DRIVER, RØ        ;PICK UP DDB ADDRESS
            MOV    (SP)+,R5          ;SAVE INTERIM RETURN
            TSTB   -3(RØ)            ;COME FROM QUEUE?
            BEQ    EXIT
            MOV    @#177776,-(SP)    ;IF SO, STORE STATUS
            MOV    R5,-(SP)          ;...& RETURN
            SUB    #14,SP            ;DUMMY SAVE REGS
    EXIT:   JMP    @14(RØ)
```

A.3.4 CLOSE

As with OPEN, this routine should provide for the possibility of some form of hardware shut down such as the punching of trailer code and is not necessary for file-structured devices. Moreover, it is likely to be a requirement for output devices only. If it is provided, Driver Table Facility Indicator (Bit 6) must be set.

Again, the probable form is initialization of the hardware action required, with immediate return via RTS PC and eventual completion return via the DDB-stored address.

A.3.5 SPECIAL

This routine may be included if either the device itself contains the hardware to perform some special function or there is a need for software simulation of such hardware on other devices, e.g., tape rewind. It should not be provided otherwise. Its presence must be indicated by Bit 5 of the Facility Indicator.

The function itself is stored by the Monitor as a code in the DDB as shown earlier. When called, the driver routine must determine whether such function is appropriate in its case. If not, the completion return should be taken immediately with prior stack clearance, as discussed under OPEN. For a recognized function, the necessary routine must be provided. Again, its exit method will depend upon the necessity for an interrupt wait or otherwise.

A.4 Drivers for Terminals

The rate of input from terminal devices is normally dictated externally by the operator, rather than being program-driven; moreover, for both input and output, the amount of data to be transferred on each occasion may be a varying value, i.e., a line rather than a block of standard size. Furthermore, there may be problems with the conflict between echo of input during output. As a result, drivers for such devices will demand special treatment.

Normal output operation, i.e., .WRITE by the program, is handled by the Monitor Processor. On recognizing that the device being used is a terminal, as shown by Bit 8 of the facility indicator, this routine always causes a driver transfer at the end of the user line, even though the internal buffer has not been filled. The driver, however, is given the whole of a standard buffer, padded as necessary with nulls. Provided the driver can ignore these, the effect is that of just a line of output.

Input control on the other hand, must remain driver responsibility. Overcoming the rate problem will, in most cases, require circular buffering within the driver until demanded by the Monitor. At this point, transfer of data already in should occur. If this is sufficient to fill the monitor buffer, the driver can await the next request before further transfer onward. If insufficient, it should operate as any other device and use subsequent interrupts to continue to satisfy the Monitor request. It must, nevertheless, stop any transfer at the end of a line in normal operation. In order to allow the Monitor to continue, the driver must simulate the filling of the buffer by null padding (of no consequence, since terminals are by nature character-based). (Normal operation, of course, means response to user .READ's and is indicated by the size of the buffer to be filled, namely the driver standard. Should the user be requesting .TRAN's, the buffer size will vary from the standard in all likelihood and the driver may then assume he requires operation as a normal device -- complete buffer fill-up before return.)

Where input echo is a further complexity, there will doubtless be other requirements. If the echo is made immediately after the input, it may be desirable to have a second buffer to cater for the likely situation that the echo will not exactly match its origin. On the other hand, if the echo is held for any length of time, perhaps to provide correct relations between program-driven output and the echo, the second buffer could be too expensive. A larger input buffer and routines to allow for several outputs to one input character while sitting on that character might be more convenient. The conflict between such echo and program-driven output will require controlled switching within the driver input and output handlers.

DEC-11-ODDCA-A-D

PDP-11

RC11 DISK DRIVER

October 1972

SUPPLEMENT TO:

PDP-11 DEVICE DRIVER PACKAGE

DEC-11-ODDPA-A-D

MONITOR VERSION VØØ8

NOTE

This document is for information purposes only and is subject to change without notice. DEC assumes no responsibility for the use or reliability of its software on equipment which is not supplied by DEC.

NEW AND CHANGED INFORMATION

This manual documents the software as of Monitor version VØ8. It has been revised to include all new and changed material since Monitor version VØ4. Such material is indicated by vertical bars in the outside margin. Whole new pages are not so marked but are dated in the lower outside corner.

pdp11 SOFTWARE DISTRIBUTION CENTER

RC11 DISK DRIVER

The RC11 Disk Driver provides the software interface between the RC11 Control and the Monitor in the Disk Operating System on PDP-11. It consists of routines to initiate block transfers of data to or from the disk and to handle interrupts arising from completion or through failure.

It does not include OPEN & CLOSE processors. As a file-structured device, these will be unnecessary owing to the form of the Monitor file-management system. SPECIAL FUNCTION processing is also omitted. If it is found necessary to simulate the hardware function of a similar device, the necessary routine could be added later.

This driver is part of the permanently resident Monitor when the RC11 is the system disk for DOS; it can nevertheless be used when the RC11 is just another device on a system based on a different type of disk.

The driver is in two parts: 1) a table providing the interface between the driver and the Monitor, and 2) the routines to service the calls for disk operations.

1. Driver Table

The Driver Table (DC) occupies the first nine words of the driver. It complies with the standards specified for all Monitor-driver interfacing in general, and for file-structured devices in particular. The descriptive elements of the table are set up as follows:

a)	Facilities available: = 100037	Multi-dataset handling on a single unit. Input & output in ASCII or binary. File-structured with no limit to the number of files that may be in creation at one time.
b)	Standard buffer size:	64
c)	Interrupt vector address:	21Ø
d)	Interrupt servicing priority	5
e)	Device name	DC
f)	Directory start block:	1
g)	No. of bit map pointers:	1

2. Service Routines

The driver contains two routines: Set-up Transfer and Service Interrupt.

2.1 Setup Transfer (DC.TFR)

This routine first initializes a counter which is used to control the number of retries in the event of parity or timing failure. Using the address of the DDB for the dataset it is servicing (as supplied by the calling routine in the first word of the driver table), it then collects control data from the DDB and transmits it to the hardware registers for the RC11, beginning at 377440.

Two of the items involved require special processing before outward transmission; the rest are moved directly.

1. For compatibility with RF11 based DOS systems, the disk is handled in blocks of 64-words which are assumed to be continuous across whole RC11 disk surface[1]. The block number passed to the driver must be converted to the 32-word sector and drive structure of the hardware.

2. The function bits contained in the DDB automatically produce the required transfer operation. To them, however, must be added the INT ENB & GO bits (combined value 101_8) needed to set the RC11 Control. Register correctly for the transfer operation to begin.

On completion of the setup, control is returned to the calling Monitor routine via the interim return address stored on top of the stack by the calling sequence.

2.2 Interrupt Service (DC.INT)

The RC11 Control causes a priority-5 interrupt either on satisfactory completion of the transfer or because an error has been detected. Having saved the processor registers on the stack, the servicing routine must determine which of these events has occurred by examination of bit 15 of the Control Status Register. On transfer completion, it collects the address of the DDB it is servicing from the first word of the driver table and uses it to return to the completion address set in the DDB. At this exit, RØ is set to the DDB address, as required by the established convention.

[1]Although the user may manually set disk drive numbers without regard for sequence, the DOS Monitor will assume that a strict ascending order has been established, i.e., Units Ø and 1 on a two-drive system. Drive units out of sequence will be ignored.

An error may be one of the several types as indicated by further bits of the Control Status or Extended Status registers. The servicing routine, however, is concerned with only two categories:

(1) Errors which can be handled internally

Data Synchronization or Block Parity failures may be eliminated on a second or later attempt. For the sake of simplicity, a retry is initiated by restarting the transfer from the beginning again rather than from the point at which the error was detected. If finally the eighth attempt produces no satisfactory result, the processing routine sets Bit 15 of word DDB+12 to show the failure. When a block-parity error is its cause, the data may still be of some value to the user program and so is passed on. However there may still be some words yet to be transferred beyond the failing block. The routine therefore attempts to resume from this point. If this is successful, it then takes the normal completion exit. Further failure, however, is treated as fatal (see below). Such treatment is immediate in the case of a repeated data sync error, since then no data can have yet been transferred.

(2) Errors which must be rectified by the operator when recovery is possible

All other failures cause an exit to the Error diagnostic print routine, with DSK ERROR F026 as the message and the contents of the Control Status register as evidence. Write lock-out or non-resident disk may be the result of an operator fault. The operator may be able to correct this and resume program execution by the appropriate keyboard command. Such action will probably be impossible in the case of a non-existent memory error, and other errors classified as 'HARD' in the RC11 Specification or after persistent parity or timing failures.

(3) VØØØB Program Listing

A complete assembly listing of the driver follows.

```
                        ;COPYRIGHT 1971, DIGITAL EQUIPMENT CORP., MAYNARD, MASS.

                        ;VERSION NUMBER:        V000B
                        ;
                                .TITLE  DV.DC
                        ;DISK DRIVER (RC11)      VERSION 1
                        ;       IF RC11 IS THE SYSTEM DISK, A SHORT FORM OF THIS
                        ;       DRIVER MAY BE OBTAINED BY INCLUDING A DEFINITION
                        ;       FOR 'SYSDV'. FOR A SYSTEM BASED ON A DIFFERENT
                        ;       DISK, THIS DRIVER MAY BE ASSEMBLED WITH A
                        ;       DEFINITION AND RC11 MAY THEN BE TREATED AS JUST
                        ;       ANOTHER DEVICE.
                        ;
                        ;       THIS VERSION CONTAINS SET-UP & TRANSFER
                        ;       ROUTINES ONLY.
                        ;
             000000     R0=%0
             000001     R1=%1
             000002     R2=%2
             000003     R3=%3
             000004     R4=%4
             000005     R5=%5
             000006     SP=%6
             000007     PC=%7
                                .GLOBL  DC
                        ;TABLE OF STANDARDS AND POINTERS
       00000 000000     DC:     .WORD   0               ;CURRENT DDB ADDRESS (0 IF IDLE)
       00002    037     DCFLGS: .BYTE   37              ;STANDARD FACILITY INDICATOR
       00003    200             .BYTE   200             ;(NORMAL & FILE-BASED)
       00004    004             .BYTE   4               ;STANDARD BUFFER SIZE/16
       00005    070             .BYTE   DC.INT-DC       ;T.V. CONTENT
       00006    240             .BYTE   240             ;PRIORITY FOR T.V.
       00007    000             .BYTE   0               ;DESPATCH TABLE
       00010    022             .BYTE   DC.TFR-DC       ;SHOWS TFR RTN ONLY
       00011    000             .BYTE   0
       00012    000             .BYTE   0
       00013    000             .BYTE   0               ;SPARE
       00014 014570     DC.NAM: .RAD50  /DC/
       00016 000001             .WORD   DC.DIR          ;MFD BLOCK
       00020 000000             .WORD   0               ;REQUIRED FOR BIT MAP INFO
```

```
                    ;TRANSFER INITIATE
  000022 011767 DC.TFR: MOV     @PC,DC.RTC      ;ZERO RETRY COUNT
         000102
  000026 016700         MOV     DC,R0           ;GET DDB ADDRESS
         177746
  000032 022020 DC.RPT: CMP     (R0)+,(R0)+     ;BUMP POINTER TO BLOCK NO.
  000034 012702         MOV     #DC.DCS-4,R2    ;SET HWR POINTER
         177442
  000040 012012         MOV     (R0)+,@R2       ;MOVE IN BLOCK NO. ...
  000042 006312         ASL     @R2             ;...MODIFIED
  000044 062702         ADD     #10,R2          ;STEP TO MEMORY STORE
         000010
  000050 012012         MOV     (R0)+,@R2       ;MOVE IN ADDR REGD. ...
   00052 012042         MOV     (R0)+,-(R2)     ;& WORD COUNT
   00054 012001         MOV     (R0)+,R1        ;GET FUNCTION
   00056 151701         BISB    @PC,R1          ;ADD INT ENB & GO
   00060 042701         BIC     #177470,R1      ;REMOVE OTHER GARBAGE (******)
         177470
   00064 010142         MOV     R1,-(R2)        ;SEND TO CONTROL
   00066 000207         RTS     PC              ;RETURN TO MONITOR FOR NOW
                    ;(******) - CARE!!!! USED AS LITERAL BY PREVIOUS INSTRUCTION
```

```
1                       ;INTERRUPT SERVICE
2  000070 013746 DC.INT: MOV     @#V.RSAV,-(SP)
          000044
3  000074 004536         JSR     R5,@(SP)+
4  000076 016700         MOV     DC,R0           ;GET DCB ADDRESS
          177676
5  000102 012701         MOV     #DC.DCS-2,R1    ;GET PTR TO H/W REGS.
          177444
6  000106 012103         MOV     (R1)+,R3        ;SAVE ERROR STATUS REGS.
7  000110 100411         BMI     DC.AGN          ;IF DATA LATE SET TRY AGAIN
8  000112 012102         MOV     (R1)+,R2        ;ANY ERRORS SEEN?
9  000114 100402         BMI     DC.ERR          ;YES - GO FIND CAUSE
10 00116 000170 DC.XIT: JMP     @14(R0)         ;RETURN MONITOR
          000014
11                      ;ERROR ROUTINE:
12 00122 006302 DC.ERR: ASL     R2              ;CHECK ERROR CAUSE ...
13 00124 100024         BPL     DC.OFF          ;FOR DATA FAILURE ...
14 00126 006327         ASL     (PC)+           ;... ALREADY RETRIED 8 TIMES?
15 00130 000000 DC.RTC: .WORD   0
16 00132 103406         BCS     DC.PER          ;IF SO FORCE CONTINUE
17 00134 004767 DC.AGN: JSR     PC,DC.RPT       ;OTHERWISE TRY AGAIN
          177672
18 00140 013705 DC.REC: MOV     @#V.XIT,R5
          000042
19 00144 000165         JMP     4(R5)
          000004
20 00150 006303 DC.PER: ASL     R3              ;IF DATA SYNC ERR, NOW ...
21 00152 100011         BPL     DC.OFF          ;... TREAT AS FATAL
22 00154 052760         BIS     #100000,12(R0)  ;RETURN PARITY FAIL FLAG
          100000
          000012
23 00162 005711         TST     @R1             ;ALREADY AT BLOCK END?
24 00164 001754         BEQ     DC.XIT          ;IF SO EXIT NOW
25 00166 011767         MOV     @PC,DC.RTC      ;NO MORE REPEAT TRIES NOW
          177736
26 00172 005241         INC     -(R1)           ;CONTINUE DISK TRANSFER
27 00174 000761         BR      DC.REC          ;... VIA COMMON EXIT
28                      ;ERROR IS NOT IMMEDIATELY RECOVERABLE:
29 00176 005067 DC.OFF: CLR     DC              ;FREE DISK FOR EDP
          177576
30 00202 014146         MOV     -(R1),-(SP)     ;DISK STATUS IS EVIDENCE
31 00204 012746         MOV     #DC.ENO,-(SP)   ;SET UP ERROR NO.
          001426
32 00210 032767         BIT     #SDRVR,DCFLGS   ;SYSTEM DRIVER?
          010000
          177564
33 00216 001401         BEQ     DC.SND          ;NO - BRANCH
34 00220 005016         CLR     @SP             ;CODE TO FORCE A HALT
35 00222 000004 DC.SND: IOT                     ;GO TO DIAG. PRT.
```

```
               ;DEFINITIONS:
        177446 DC.DCS=177446
        000001 DC.DIR=1
        001426 DC.ENC=1426
        000044 V.RSAV=44
        000042 V.XIT=42
        010000 SDRVR=10000
        000001'        .END
```

```
DV.DC    MACRO VØØ4-14 13-SEP-72 Ø2:5Ø PAGE 4+
SYMBOL TABLE

DC          ØØØØØØRG      DCFLGS   ØØØØØ2R     DC.AGN   ØØØ134R
DC.DCS= 177446        DC.DIR=  ØØØØØ1      DC.ENC=  ØØ1426
DC.ERR    ØØØ122R       DC.INT   ØØØØ7ØR     DC.NAM   ØØØØ14R
DC.OFF    ØØØ176R       DC.PER   ØØØ15ØR     DC.REC   ØØØ14ØR
DC.RPT    ØØØØ32R       DC.RTC   ØØØ13ØR     DC.SND   ØØØ222R
DC.TFR    ØØØØ22R       DC.XIT   ØØØ116R     PC       =%ØØØØØ7
RØ      =%ØØØØØØ        R1       =%ØØØØØ1    R2       =%ØØØØØ2
R3      =%ØØØØØ3        R4       =%ØØØØØ4    R5       =%ØØØØØ5
SDRVR = Ø1ØØØØ          SP       =%ØØØØØ6    V.RSAV=  ØØØØ44
V.XIT = ØØØØ42
. ABS.    ØØØØØØ     ØØØ
          ØØØ224     ØØ1

ERRORS DETECTED:  Ø
FREE CORE:  19413. WORDS
.LP:<DT:DC
```

(4) V000A Program Listing

```
                   ;COPYRIGHT 1971, DIGITAL EQUIPMENT CORP., MAYNARD, MASS.

                   ;VERSION NUMBER:         V000A
                   ;
                           .TITLE  DC
                   ;DISK DRIVER (RC11)       VERSION 1
                   ;       IF RC11 IS THE SYSTEM DISK, A SHORT FORM OF THIS
                   ;       DRIVER MAY BE OBTAINED BY INCLUDING A DEFINITION
                   ;       FOR 'SYSDV'. FOR A SYSTEM BASED ON A DIFFERENT
                   ;       DISK, THIS DRIVER MAY BE ASSEMBLED WITH A
                   ;       DEFINITION AND RC11 MAY THEN BE TREATED AS JUST
                   ;       ANOTHER DEVICE.
                   ;
                   ;       THIS VERSION CONTAINS SET-UP & TRANSFER
                   ;       ROUTINES ONLY.
                   ;
            000000 R0=%0
            000001 R1=%1
            000002 R2=%2
            000003 R3=%3
            000004 R4=%4
            000005 R5=%5
            000006 SP=%6
            000007 PC=%7
                           .GLOBL  DC,S.RSAV,S.XIT
                   ;TABLE OF STANDARDS AND POINTERS
     000000 000000 DC:     .WORD   0                   ;CURRENT DDB ADDRESS (0 IF IDLE)
     000002    037         .BYTE   37                  ;STANDARD FACILITY INDICATOR
     000003    200         .BYTE   200                 ;(NORMAL & FILE-BASED)
     000004    004         .BYTE   4                   ;STANDARD BUFFER SIZE/16
     000005    070         .BYTE   DC.INT-DC           ;T.V. CONTENT
     000006    240         .BYTE   240                 ;PRIORITY FOR T.V.
     000007    000         .BYTE   0                   ;DESPATCH TABLE
     000010    022         .BYTE   DC.TFR-DC           ;SHOWS TFR RTN ONLY
     000011    000         .BYTE   0
     000012    000         .BYTE   0
     000013    000         .BYTE   0                   ;SPARE
     000014 014570 DC.NAM: .RAD50  'DC'
     000016 000001         .WORD   DC.DIR              ;MFD BLOCK
     000020 000000         .WORD   0                   ;REQUIRED FOR BIT MAP INFO

                   ;TRANSFER INITIATE
     000022 011767 DC.TFR: MOV     @PC,DC.RTC          ;ZERO RETRY COUNT
            000102
     000026 016700         MOV     DC,R0               ;GET DDB ADDRESS
            177746
     000032 022020 DC.RPT: CMP     (R0)+,(R0)+         ;BUMP POINTER TO BLOCK NO.
     000034 012702         MOV     #DC.DCS-4,R2        ;SET HWR POINTER
            177442
     000040 012012         MOV     (R0)+,@R2           ;MOVE IN BLOCK NO. ...
     000042 006312         ASL     @R2                 ;...MODIFIED
     000044 062702         ADD     #10,R2              ;STEP TO MEMORY STORE
            000010
     000050 012012         MOV     (R0)+,@R2           ;MOVE IN ADDR REQD. ...
     000052 012042         MOV     (R0)+,-(R2)         ;& WORD COUNT
     000054 012001         MOV     (R0)+,R1            ;GET FUNCTION
     000056 151701         BISB    @PC,R1              ;ADD INT ENB & GO
     000060 042701         BIC     #177470,R1          ;REMOVE OTHER GARBAGE (******)
            177470
     000064 010142         MOV     R1,-(R2)            ;SEND TO CONTROL
     000066 000207         RTS     PC                  ;RETURN TO MONITOR FOR NOW
                   ;(******) - CARE!!!! USED AS LITERAL BY PREVIOUS INSTRUCTION
```

```
                        ;INTERRUPT SERVICE
                        DC.INT: .IFDF   SYSDV
                                JSR     R5,S.RSAV           ;GO SAVE REGISTERS
                                .ENDC
                                .IFNDF  SYSDV
000070 013746                   MOV     @#V.RSAV,-(SP)
       000044
000074 004536                   JSR     R5,@(SP)+
                                .ENDC
000076 016700                   MOV     DC,R0               ;GET DDB ADDRESS
       177676
000102 012701                   MOV     #DC.DCS-2,R1        ;GET PTR TO H/W REGS.
       177444
000106 012103                   MOV     (R1)+,R3            ;SAVE ERROR STATUS REGS.
000110 100411                   BMI     DC.AGN              ;IF DATA LATE SET TRY AGAIN
000112 012102                   MOV     (R1)+,R2            ;ANY ERRORS SEEN?
000114 100402                   BMI     DC.ERR              ;YES - GO FIND CAUSE
000116 000170           DC.XIT: JMP     @14(R0)             ;RETURN MONITOR
       000014
                        ;ERROR ROUTINE:
000122 006302           DC.ERR: ASL     R2                  ;CHECK ERROR CAUSE ...
000124 100024                   BPL     DC.OFF              ;FOR DATA FAILURE ...
000126 006327                   ASL     (PC)+               ;.... ALREADY RETRIED 8 TIMES?
000130 000000           DC.RTC: .WORD   0
000132 103406                   BCS     DC.PER              ;IF SO FORCE CONTINUE
000134 004767           DC.AGN: JSR     PC,DC.RPT           ;OTHERWISE TRY AGAIN
       177672
                        DC.REC: .IFDF   SYSDV
                                JMP     S.XIT+4             ;TAKE COMMON EXIT
                                .ENDC
                                .IFNDF  SYSDV
000140 013705                   MOV     @#V.XIT,R5
       000042
000144 000165                   JMP     4(R5)
       000004
                                .ENDC
000150 006303           DC.PER: ASL     R3                  ;IF DATA SYNC ERR. NOW ...
000152 100011                   BPL     DC.OFF              ;... TREAT AS FATAL
000154 052760                   BIS     #100000,12(R0)      ;RETURN PARITY FAIL FLAG
       100000
       000012
000162 005711                   TST     @R1                 ;ALREADY AT BLOCK END?
000164 001754                   BEQ     DC.XIT              ;IF SO EXIT NOW
000166 011767                   MOV     @PC,DC.RTC          ;NO MORE REPEAT TRIES NOW
       177736
000172 005241                   INC     -(R1)               ;CONTINUE DISK TRANSFER
000174 000761                   BR      DC.REC              ;... VIA COMMON EXIT
                        ;ERROR IS NOT IMMEDIATELY RECOVERABLE:
                        DC.OFF: .IFDF   SYSDV
                                CLR     DC                  ;FREE DISK FOR EDP
                                .ENDC
000176 014146                   MOV     -(R1),-(SP)         ;DISK STATUS IS EVIDENCE
000200 012746                   MOV     #DC.ENO,-(SP)       ;SET UP ERROR NO.
       001426
000204 000004                   IOT                         ;GO TO DIAG. PRT.
```

```
                ;DEFINITIONS:
        177446 DC.DCS=177446
        ØØØØØ1 DC.DIR=1
        ØØ1426 DC.ENO=1426
        ØØØØ44 V.RSAV=44
        ØØØØ42 V.XIT=42
        ØØØØØ1          .END


ØØØØØØ ERRORS
```

```
DC           ØØØØØØRG      DC.AGN     ØØØ134R      DC.DCS = 177446
DC.DIR =  ØØØØØ1          DC.ENO = ØØ1426         DC.ERR   ØØØ122R
DC.INT     ØØØØ7ØR        DC.NAM   ØØØØ14R        DC.OFF   ØØØ176R
DC.PER     ØØØ15ØR        DC.REC   ØØØ14ØR        DC.RPT   ØØØØ32R
DC.RTC     ØØØ13ØR        DC.TFR   ØØØØ22R        DC.XIT   ØØØ116R
PC        =%ØØØØØ7        RØ      =%ØØØØØØ        R1      =%ØØØØØ1
R2        =%ØØØØØ2        R3      =%ØØØØØ3        R4      =%ØØØØØ4
R5        =%ØØØØØ5        SP      =%ØØØØØ6        S.RSAV = ****** G
S.XIT     = ****** G      V.RSAV = ØØØØ44         V.XIT   = ØØØØ42
.         = ØØØ2Ø6R
```

DEC-11-ODDFA-A-D

P D P - 1 1

R F 1 1 D I S K D R I V E R

October 1972

SUPPLEMENT TO:

PDP-11 DEVICE DRIVER PACKAGE

DEC-11-ODDPA-A-D

MONITOR VERSION VØØ8

NOTE

This document is for information purposes only and is subject to change without notice. DEC assumes no responsibility for the use or reliability of its software on equipment which is not supplied by DEC.

NEW AND CHANGED INFORMATION

This manual documents the software as of Monitor version VØ8. It has been revised to include all new and changed material since Monitor version VØ4. Such material is indicated by vertical bars in the outside margin. Whole new pages are not so marked but are dated in the lower outside corner.

pdp11 SOFTWARE DISTRIBUTION CENTER

RF11 DISK DRIVER

The RF11 Disk Driver consists of routines to initiate block transfers of data to or from the disk and to handle interrupts arising from completion or through failure.

It does not include OPEN & CLOSE processors. As a file-structured device, these will be unnecessary owing to the form of the Monitor file-management system. SPECIAL FUNCTION processing is also omitted. If it is found necessary to simulate the hardware function of a similar device, the necessary routine could be added later.

This driver is part of the permanently resident Monitor when the RF11 is the system disk. It may also be used when RF11 is merely another device in a system based on a different type of disk.

The driver is in two parts: 1) a table providing the interface between the driver and the Monitor, and 2) the routines to service the calls for disk operations.

1. Driver Table

The Driver Table (DF) occupies the first nine words of the driver. It complies with the standards specified for all Monitor-driver interfacing in general, and for file-structured devices in particular. The descriptive elements of the table are set up as follows:

a)	Facilities available: =100037	Multi-dataset handling on a single unit. Input and output in ASCII or binary. File-structured with no limit to the number of files that may be in creation at one time.
b)	Standard buffer size:	64
c)	Interrupt vector address:	204
d)	Interrupt servicing priority	5
e)	Device name	DF
f)	Directory start block:	1
g)	No. of bit map pointers:	1

2. Service Routines

The driver contains two routines: Setup Transfer and Service Interrupt.

2.1 Set-up Transfer (DF.TFR)

This routine first initializes a counter which is used to control the number of retries in the event of parity or timing failure. Using the address of the DDB for the dataset it is servicing (as supplied by the calling routine in the first word of the driver table), it then collects control data from the DDB and transmits it to the hardware registers for the RF11, beginning at 377460.

Two of the items involved require special processing before outward transmission; the rest are moved directly.

1. The driver block number set into the DDB must be converted to meet the platter and word structure of RF11. All the platters currently under one control are considered as a single continuous surface. As a result, the most significant bits of the block number represent the appropriate platter number and the remainder the word starting the block. The required conversion is therefore merely multiplication of the block number by 64 across 21 bits.

2. The function bits contained in the DDB automatically produce the required transfer operation. To them, however, must be added the INT ENB & GO bits (combined value 101_8) needed to set the RF11 Control Register. correctly for the transfer operation to begin.

On completion of the setup, control is returned to the calling Monitor routine via the interim return address stored on top of the stack by the calling sequence.

2.2 Interrupt Service (DF.INT)

The RF11 control causes a priority-5 interrupt either on satisfactory completion of the transfer or because an error has been detected. Having saved the processor registers on the stack, the servicing routine must determine which of these events has occurred by examination of bit 15 of the Control Status Register. On transfer completion, it collects the address of the DDB it is servicing from the first word of the driver table and uses it to return to the completion address set in the DDB. At this exit, RØ is set to the DDB address, as required by the established convention.

An error may be one of the several types as indicated by further bits of the Control Status or Extended Status registers. The servicing routine, however, is concerned with only two categories:

(1) Errors which can be handled internally

Parity or timing failures may be eliminated on a second or later attempt. For the sake of simplicity, a retry is initiated by restarting the transfer from the beginning again rather than from the point at which the error was detected. If finally the eighth attempt produces no satisfactory result, the processing routine sets Bit 15 of Word DDB+12 to show the failure. It then checks if any words still remain to be transferred beyond the failing one. If so, it attempts to resume the transfer from this point. If this is successful, it then takes the normal completion exit. Further failure, however, is treated as fatal.

(2) Errors which must be rectified by the operator (when recovery is possible)

All other failures cause an exit to the Error diagnostic print routine, with DSK ERROR F026 as the message and the contents of the Control Status register as evidence. Write lock-out or non-resident disk may be the result of an operator fault. The operator may be able to correct this and resume program execution by the appropriate keyboard command. Such action will probably be impossible in the case of a non-existent memory error, and other errors classified as 'HARD' in the RF11 Specification or after persistent parity or timing failures.

(3) Program Listings

A complete assembly listing of the driver monitor VØ8-Ø2 follows.

```
                         ;COPYRIGHT 1971, DIGITAL EQUIPMENT CORP., MAYNARD, MASS.

                         ;VERSION NUMBER:        V003B

                                   .TITLE  DV.DF
                         ;DISK DRIVER (RF11)      VERSION 1
                         ;         RESIDENT MONITOR ROUTINE FOR SYSTEM USAGE
                         ;                   CONTAINS SET UP & TRANSFER ROUTINES ONLY
                  000000 R0=%0
                  000001 R1=%1
                  000002 R2=%2
                  000003 R3=%3
                  000004 R4=%4
                  000005 R5=%5
                  000006 SP=%6
                  000007 PC=%7
                                   .GLOBL  DF
                         ;TABLE OF STANDARDS AND POINTERS
    00000  000000 DF:      .WORD   0                 ;CURRENT DDB ADDRESS (0 IF IDLE)
    00002     037 DFFLGS:  .BYTE   37                ;STANDARD FACILITY INDICATOR
    00003     200          .BYTE   200               ;(NORMAL & FILE-BASED)
    00004     004          .BYTE   4                 ;STANDARD BUFFER SIZE/16
    00005     102          .BYTE   DF.INT-DF         ;T.V. CONTENT
    00006     240          .BYTE   240               ;PRIORITY FOR T.V.
    00007     000          .BYTE   0                 ;DESPATCH TABLE
    00010     022          .BYTE   DF.TFR-DF         ;SHOWS TFR RTN ONLY
    00011     000          .BYTE   0
    00012     000          .BYTE   0
    00013     000          .BYTE   0                 ;SPARE
    00014  014760 DF.NAM:  .RAD50  /DF/
    00016  000001          .WORD   DF.DIR            ;MFD BLOCK
    00020  000000          .WORD   0                 ;REQUIRED FOR BIT MAP INFO
```



```
                         ;TRANSFER INITIATE
   000022 011767 DF.TFR:  MOV     @PC,DF.RTC        ;ZERO RETRY COUNT
          000112
   000026 111737 DF.RPT:  MOVB    @PC,@#DF.DCS+1    ;CLEAR DISK IN CASE OF ERROR
          177461
   000032 016700          MOV     DF,R0             ;GET DDB ADDRESS
          177742
   000036 022020          CMP     (R0)+,(R0)+       ;BUMP POINTER TO BLOCK NO.
   000040 012702          MOV     #DF.DCS+12,R2     ;SET HWR POINTER
          177472
   000044 111703          MOVB    @PC,R3            ;SET UP BLOCK CONVERSION
   000046 012004          MOV     (R0)+,R4          ;GET BLOCK NUMBER   (******)
   000050 006304          ASL     R4                ;CONVERT TO WORDS
    00052 106103          ROLB    R3
    00054 103375          BCC     .-4
    00056 010342          MOV     R3,-(R2)          ;SET UP DISK ADDRESS & EXT.
    00060 010442          MOV     R4,-(R2)
    00062 012042          MOV     (R0)+,-(R2)       ;MOVE IN WORD COUNT ...
    00064 012042          MOV     (R0)+,-(R2)       ;& MEMORY ADDRESS
    00066 012001          MOV     (R0)+,R1          ;GET FUNCTION
    00070 151701          BISB    @PC,R1            ;ADD INT ENB & GO
    00072 042701          BIC     #177470,R1        ;REMOVE OTHER GARBAGE (******)
          177470
    00076 010142          MOV     R1,-(R2)          ;SEND TO CONTROL
    00100 000207          RTS     PC                ;RETURN TO MONITOR FOR NEW
                         ;(******) - CARE!!!! USED AS LITERAL BY PREVIOUS INSTRUCTION
```

```
                         ;INTERRUPT SERVICE
  000102 013746 DF.INT: MOV     @#V.RSAV,-(SP)
         000044
  000106 004536          JSR     R5,@(SP)+
  000110 012701          MOV     #DF.DCS,R1        ;ERROR CAUSE INTERRUPT?
         177460
  000114 012102          MOV     (R1)+,R2
  000116 100404          BMI     DF.ERR            ;YES - GO FIND CAUSE
  000120 016700          MOV     DF,R0             ;GET DDB ADDRESS
         177654
  000124 016007 DF.XIT: MOV     14(R0),PC         ;RETURN MONITOR
         000014
                         ;ERROR ROUTINE:
  00130 032702 DF.ERR: BIT     #11000,R2         ;PARITY OR MISSED?
         011000
  00134 001423          BEQ     DF.OFF
  00136 006327 DF.AGN: ASL     #0                ;YES - RETRIED 8 TIMES?
         000000
         000140'DF.RTC=.-2
  00142 103406          BCS     DF.PER            ;IF SO FORCE CONTINUE
  00144 004767          JSR     PC,DF.RPT         ;OTHERWISE TRY AGAIN
         177656
  00150 013705 DF.REC: MOV     @#V.XIT,R5
         000042
  00154 000165          JMP     4(R5)
         000004
  00160 052760 DF.PER: BIS     #100000,12(R0)   ;RETURN PARITY FAIL FLAG
         100000
         000012
  00166 005711          TST     @R1               ;ALREADY AT BLOCK END?
  00170 001755          BEQ     DF.XIT            ;IF SO EXIT NOW
  00172 005767          TST     DF.RTC            ;OTHERWISE CHECK IF 2ND TIME
         177742
  00176 001402          BEQ     DF.OFF            ;IF SO NO POINT IN MORE
  00200 005241          INC     -(R1)             ;CONTINUE DISK TRANSFER
  00202 000762          BR      DF.REC            ;... VIA COMMON EXIT
                         ;ERROR IS NOT IMMEDIATELY RECOVERABLE:
  00204 005067 DF.OFF: CLR     DF                ;FREE DISK FOR FOP
         177570
  00210 014146          MOV     -(R1),-(SP)       ;DISK STATUS IS EVIDENCE
  00212 012746          MOV     #DF.ENO,-(SP)     ;SET UP ERROR NO.
         001426
  00216 032767          BIT     #SDRVR,DFFLGS     ;SYSTEM DRIVER?
         010000
         177556
  00224 001401          BEQ     DF.SND            ;NO - BRANCH
  00226 005016          CLR     @SP               ;CODE TO FORCE A HALT
  00230 000004 DF.SND: IOT                       ;GO TO DIAG. PRT.
                         ;DEFINITIONS:
         177460 DF.DCS=177460
         000001 DF.DIR=1
         001426 DF.ENO=1426
         000042 V.XIT=42
         000044 V.RSAV=44
         010000 SDRVR=10000
         000001'        .END
```

```
SYMBOL TABLE

DF         000000RG        DFFLGS   000002R         DF.AGN   000136R
DF.DCS=  177460            DF.DIR=  000001          DF.END=  001426
DF.ERR   000130R           DF.INT   000102R         DF.NAM   000014R
DF.OFF   000204R           DF.PER   000160R         DF.REC   000150R
DF.RPT   000026R           DF.RTC=  000140R         DF.SND   000230R
DF.TFR   000022R           DF.XIT   000124R         PC       =%000007
R0       =%000000          R1       =%000001        R2       =%000002
R3       =%000003          R4       =%000004        R5       =%000005
SDRVR  = 010000            SP       =%000006        V.RSAV=  000044
V.XIT  = 000042
. ABS.   000000      000
         000232      001

ERRORS DETECTED:  0
FREE CORE:  19413. WORDS
,LP:<DT:DF
```

A listing of the DF: Driver for Monitor VØ4 release follows:

```
                 ;COPYRIGHT 1971, DIGITAL EQUIPMENT CORP., MAYNARD, MASS.

                 ;VERSION NUMBER:        V003B

                          .TITLE  DF
                 ;DISK DRIVER (RF11)      VERSION 1
                 ;        RESIDENT MONITOR ROUTINE FOR SYSTEM USAGE
                 ;                 CONTAINS SET UP & TRANSFER ROUTINES ONLY
         000000  R0=%0
         000001  R1=%1
         000002  R2=%2
         000003  R3=%3
         000004  R4=%4
         000005  R5=%5
         000006  SP=%6
         000007  PC=%7
                          .GLOBL  DF,S.RSAV,S.XIT
                 ;TABLE OF STANDARDS AND POINTERS
000000  000000  DF:       .WORD   0               ;CURRENT DDB ADDRESS (0 IF IDLE)
000002     037            .BYTE   37              ;STANDARD FACILITY INDICATOR
000003     200            .BYTE   200             ;(NORMAL & FILE-BASED)
000004     004            .BYTE   4               ;STANDARD BUFFER SIZE/16
000005     102            .BYTE   DF.INT-DF       ;T.V. CONTENT
000006     240            .BYTE   240             ;PRIORITY FOR T.V.
000007     000            .BYTE   0               ;DESPATCH TABLE
000010     022            .BYTE   DF.TFR-DF       ;SHOWS TFR RTN ONLY
000011     000            .BYTE   0
000012     000            .BYTE   0
000013     000            .BYTE   0               ;SPARE
000014  014760  DF.NAM:   .RAD50  'DF'
000016  000001            .WORD   DF.DIR          ;MFD BLOCK
000020  000000            .WORD   0               ;REQUIRED FOR BIT MAP INFO


                 ;TRANSFER INITIATE
000022  011767  DF.TFR:  MOV     @PC,DF.RTC       ;ZERO RETRY COUNT
        000112
000026  111737  DF.RPT:  MOVB    @PC,@#DF.DCS+1   ;CLEAR DISK IN CASE OF ERROR
        177461
000032  016700           MOV     DF,R0            ;GET DDB ADDRESS
        177742
000036  022020           CMP     (R0)+,(R0)+      ;BUMP POINTER TO BLOCK NO.
000040  012702           MOV     #DF.DCS+12,R2    ;SET HWR POINTER
        177472
000044  111703           MOVB    @PC,R3           ;SET UP BLOCK CONVERSION
000046  012004           MOV     (R0)+,R4         ;GET BLOCK NUMBER  (******)
000050  006304           ASL     R4               ;CONVERT TO WORDS
000052  106103           ROLB    R3
000054  103375           BCC     .-4
000056  010342           MOV     R3,-(R2)         ;SET UP DISK ADDRESS & EXT.
000060  010442           MOV     R4,-(R2)
000062  012042           MOV     (R0)+,-(R2)      ;MOVE IN WORD COUNT ...
000064  012042           MOV     (R0)+,-(R2)      ;& MEMORY ADDRESS
000066  012001           MOV     (R0)+,R1         ;GET FUNCTION
000070  151701           BISB    @PC,R1           ;ADD INT ENB & GO
000072  042701           BIC     #177470,R1       ;REMOVE OTHER GARBAGE (******)
        177470
000076  010142           MOV     R1,-(R2)         ;SEND TO CONTROL
000100  000207           RTS     PC               ;RETURN TO MONITOR FOR NOW
                 ;(******) = CARE!!!! USED AS LITERAL BY PREVIOUS INSTRUCTION
```

```
                ;INTERRUPT SERVICE
                DF.INT: .IFDF   SYSDV
                        JSR     R5,S.RSAV       ;GO SAVE REGISTERS
                        .ENDC
                        .IFNDF  SYSDV
000102 013746           MOV     @#V.RSAV,-(SP)
       000044
000106 004536           JSR     R5,@(SP)+
                        .ENDC
000110 012701           MOV     #DF.DCS,R1      ;ERROR CAUSE INTERRUPT?
       177460
000114 012102           MOV     (R1)+,R2
000116 100404           BMI     DF.ERR          ;YES - GO FIND CAUSE
000120 016700           MOV     DF,R0           ;GET DDB ADDRESS
       177654
000124 016007 DF.XIT: MOV       14(R0),PC       ;RETURN MONITOR
       000014
              ;ERROR ROUTINE:
000130 032702 DF.ERR: BIT       #11000,R2       ;PARITY OR MISSED?
       011000
000134 001423           BEQ     DF.OFF
000136 006327 DF.AGN: ASL       #0              ;YES - RETRIED 8 TIMES?
       000000
       000140 DF.RTC=.-2
000142 103406           BCS     DF.PER          ;IF SO FORCE CONTINUE
000144 004767           JSR     PC,DF.RPT       ;OTHERWISE TRY AGAIN
       177656
              DF.REC: .IFDF     SYSDV
                        JMP     S.XIT+4         ;TAKE COMMON EXIT
                        .ENDC
                        .IFNDF  SYSDV
000150 013705           MOV     @#V.XIT,R5
       000042
000154 000165           JMP     4(R5)
       000004
                        .ENDC
000160 052760 DF.PER: BIS       #100000,12(R0)  ;RETURN PARITY FAIL FLAG
       100000
       000012
000166 005711           TST     @R1             ;ALREADY AT BLOCK END?
000170 001755           BEQ     DF.XIT          ;IF SO EXIT NOW
000172 005767           TST     DF.RTC          ;OTHERWISE CHECK IF 2ND TIME
       177742
000176 001402           BEQ     DF.OFF          ;IF SO NO POINT IN MORE
000200 005241           INC     -(R1)           ;CONTINUE DISK TRANSFER
000202 000762           BR      DF.REC          ;... VIA COMMON EXIT
              ;ERROR IS NOT IMMEDIATELY RECOVERABLE:
              DF.OFF: .IFDF     SYSDV
                        CLR     DF              ;FREE DISK FOR EDP
                        .ENDC
000204 014146           MOV     -(R1),-(SP)     ;DISK STATUS IS EVIDENCE
000206 012746           MOV     #DF.ENO,-(SP)   ;SET UP ERROR NO.
       001426
000212 000004           IOT                     ;GO TO DIAG. PRT.
              ;DEFINITIONS:
       177460 DF.DCS=177460
```

```
000001 DF.DIR=1
001426 DF.END=1426
000042 V.XIT=42
000044 V.RSAV=44
000001          .END
```

```
000000 ERRORS
```

```
DF          000000RG     DF.AGN     000136R      DF.DCS = 177460
DF.DIR  = 000001        DF.END = 001426        DF.ERR   000130R
DF.INT    000102R      DF.NAM     000014R      DF.OFF   000204R
DF.PER    000160R      DF.REC     000150R      DF.RPT   000026R
DF.RTC  = 000140R      DF.TFR     000022R      DF.XIT   000124R
PC       =%000007      R0       =%000000      R1       =%000001
R2       =%000002      R3       =%000003      R4       =%000004
R5       =%000005      SP       =%000006      S.RSAV = ****** G
S.XIT   = ****** G     V.RSAV = 000044        V.XIT  = 000042
.        = 000214R
```

DEC-11-ODDKA-A-D

PDP-11

RK11 DISK DRIVER

October 1972

SUPPLEMENT TO:

PDP-11 DEVICE DRIVER PACKAGE

DEC-11-ODDPA-A-D

MONITOR VERSION VØØ8

NOTE

This document is for information purposes only and is subject to change without notice. DEC assumes no responsibility for the use or reliability of its software on equipment which is not supplied by DEC.

NEW AND CHANGED INFORMATION

This manual documents the software as of Monitor version VØ8. It has been revised to include all new and changed material since Monitor version VØ4. Such material is indicated by vertical bars in the outside margin. Whole new pages are not so marked but are dated in the lower outside corner.

pdp11 SOFTWARE DISTRIBUTION CENTER

RK11 DISK DRIVER

The RK11 Disk Driver consists of routines which initiate block transfers of data to or from a disk cartridge and which handle interrupts arising from normal completion or errors.

Special functions, OPEN and CLOSE processing, are not necessary and thus are not supported. Advance seeks are not supported in this initial release for several reasons, among which are:

- The majority of the DOS installations which utilize the RK have only one unit, so the extra code in the driver (approximately $25\emptyset_{1\emptyset}$ words) would be detrimental in most cases.

- No DOS system programs do their I/O in a manner which would reap huge benefits by seeking ahead.

- The Monitor would have to be altered to inform the RK driver before a Bus Init is issued.

The driver should be assembled at each installation where low density drives are present. If low density drives are present, proceed as follows:

(a) If <u>all</u> drives are low density, then define LOWDEN at assembly time.

(b) If there is a mixture of high and low density drives, then define MIXED at assembly time and define CONFIG as follows:

> Imagine CONFIG as an 8-bit field, the rightmost bit of which corresponds to unit Ø. If a bit in a given position is one (1), then that particular drive is low density. For example, CONFIG=12 (8) [ØØØØ1Ø1Ø(2) indicates that units 1 and 3 are low density.

LOWDEN and MIXED should not be simultaneously defined. If they are, MIXED is ignored, i.e., the assembly proceeds as if LOWDEN is defined and MIXED is undefined. If MIXED is defined, but CONFIG is not, an assembly error will result, viz., a "U" flag on the line labeled DENIND.

The default assembly condition, where no parameters are defined, is that all units are high density.

1. Driver Table

This driver contains the driver table required for the Monitor interface. The elements are:

(a)	Facilities indicator=1Ø2Ø37	Multi-dataset handling on a single unit. ASCII and Binary input and output. File-structured. Multiunit. No limit to the number of files which may be simultaneously created.
(b)	Standard buffer size	$256_{1\emptyset}$ (words)
(c)	Interrupt Vector Address	22Ø
(d)	Interrupt servicing priority	5
(e)	Device name	DK
(f)	Directory start block	1
(g)	Number of bit map pointers	8

2. The Transfer Routine

The retry counter is cleared; the unit number, block number, memory address, word count, and function (read or write) are obtained from the DDB, the address of which is in register zero at entrance. If the block number exceeds 4799, then output an error message. Otherwise:

(1) convert the block number to a disk address,

(2) set I.D.E. (bit 6) and GO (bit Ø) in the function word, and

(3) send to the controller and return to the caller via RTS PC.

3. The Interrupt Processing Routine

This routine is entered at level 5. The registers are saved on the stack, and pertinent RK controller registers are obtained in case this is an error. If it is not an error, and the last function issued was not a drive reset (see below), the completion return (@(DDB+14) is taken. If it is an error situation, then an attempt to re-try will be made if the error was one of the following:

(1) any "soft" error,

(2) seek incomplete,

(3) read timing error,

(4) data late, or

(5) seek error

All other error conditions result in a fatal error message. In addition, if the word count is not zero after eight re-tries, a fatal error message is issued. Otherwise, a parity error is returned.

NOTE

> Errors (2), (3), (4), and (5) above are among the "hard" errors. A control reset must be issued in order to continue. Additionally, a drive reset must be issued in order to continue after a seek incomplete. Thus, if the last function issued was a drive reset, the retry logic is called.

4. VØØ2 Program Listing

A listing follows, conditionalized for all drives being high density.

```
DV.RKH  MACRO VØØ4-14 13-SEP-72 Ø2:52 PAGE 1






                    .IFDF   LOWDEN
                    .TITLE  DV.RKL
                    .ENDC
                    .IFNDF  LOWDEN
                    .IFNDF  CONFIG
                    .TITLE  DV.RKH
                    .ENDC
                    .IFDF   CONFIG
                    .IFNZ   CONFIG&1
                    .TITLE  DV.RKL
                    .IFF
                    .TITLE  DV.RKH
                    .ENDC
                    .ENDC
                    .ENDC



                    ;DISK DRIVER (RK11) VERSION      VØØ5A
                    ;                                ØØ2
                    ;COPYRIGHT DIGITAL EQUIPMENT CORPORATION
                    ;MAYNARD, MASSACHUSETTS    JUNE 1971
```

```
         ØØØØØØ RØ=%Ø
         ØØØØØ1 R1=%1
         ØØØØØ2 R2=%2
         ØØØØØ3 R3=%3
         ØØØØØ4 R4=%4
         ØØØØØ5 R5=%5
         ØØØØØ6 R6=%6
         ØØØØØ7 PC=%7
         1774ØØ RKDS=1774ØØ
         1774Ø2 RKER=1774Ø2
         1774Ø4 RKCS=1774Ø4
         1774Ø6 RKWC=1774Ø6
         17741Ø RKBA=17741Ø
         177412 RKDA=177412
         ØØØØØ1 RKDIR=1
         ØØØØ44 V.RSAV=44
         ØØØØ42 V.XIT=42
         177776 PS=177776
         Ø1ØØØØ SDRVR=1ØØØØ


                        .GLOBL  DK

                ;STANDARD DOS INTERFACE TABLE

   ØØØØØ ØØØØØØ DK:     .WORD   Ø                  ;Ø IF IDLE, DDB PTR OTHERWISE
   ØØØØ2 1Ø2Ø37 DKFLGS: .WORD   1Ø2Ø37             ;FACILITIES WORD
   ØØØØ4    Ø2Ø         .BYTE   2Ø                 ;STD BUFFER SIZE IS 256. WORDS
   ØØØØ5    172         .BYTE   DKINT-DK           ;OFFSET TO INTERRUPT HANDLER
   ØØØØ6    24Ø         .BYTE   24Ø                ;PRIORITY LEVEL 5
   ØØØØ7    ØØØ         .BYTE   Ø                  ;NO OPEN ROUTINE
   ØØØ1Ø    Ø4Ø         .BYTE   DKSTRT-DK          ;OFFSET TO TRANSFER HANDLER
   ØØØ11    ØØØ         .BYTE   Ø                  ;NO CLOSE ROUTINE
   ØØØ12    ØØØ         .BYTE   Ø                  ;NO SPECIAL FUNCTIONS
   ØØØ13    ØØØ         .BYTE   Ø                  ;CURRENTLY UNUSED
   ØØØ14 Ø1527Ø DKNAM:  .RAD5Ø  /DK/               ;DEVICE NAME
   ØØØ16 ØØØØØ1         .WORD   RKDIR              ;FIRST MFD BLOCK
   ØØØ2Ø ØØØØØØ         .WORD   Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø    ;BIT MAP POINTERS
   ØØØ22 ØØØØØØ
   ØØØ24 ØØØØØØ
   ØØØ26 ØØØØØØ
   ØØØ3Ø ØØØØØØ
   ØØØ32 ØØØØØØ
   ØØØ34 ØØØØØØ
   ØØØ36 ØØØØØØ

   ØØØ4Ø Ø11767 DKSTRT: MOV     @PC,DKREPT         ;CLEAR RETRY INDICATOR
         ØØØ174
   ØØØ44 116ØØ1 DKRTRY: MOVB    13(RØ),R1          ;GET UNIT IN R1(13=15)
         ØØØØ13
   ØØØ5Ø Ø427Ø1         BIC     #17777Ø,R1
         17777Ø
                        .IFDF   MIXED
                        .IFNDF  LOWDEN
                        MOV     R1,R4              ;SAVE UNIT FOR LATER USE
                        .ENDC
                        .ENDC
```

```
00054 006201          ASR     R1              ;LEFT-JUSTIFY UNIT
00056 006001          ROR     R1
00060 006001          ROR     R1
00062 006001          ROR     R1              ;UNIT NOW AS DESIRED
00064 022020          CMP     (R0)+,(R0)+     ;POINTER DDB+BLOCK
00066 012002          MOV     (R0)+,R2
                      .IFDF   MIXED
                      .IFNDF  LOWDEN
                      MOV     (PC)+,R3        ;GET DENSITY PATTERN
                      .WORD   CONFIG
                      ASL     R3              ;MOVE APPROP. TO UNIT
                      DEC     R4
                      BGE     .-4
                      BCC     .+4             ;IF LOW DENSITY ...
                      ASL     R2              ;ADJUST BLOCK NO.
                      .ENDC
                      .ENDC
                      .IFDF   LOWDEN
                      ASL     R2
                      .ENDC
```

```
000070 020227           CMP     R2,#4800.         ;IS BLOCK WITHIN BOUNDS?
       011300
000074 103410           BLO     DKIN20            ;YES - BRANCH
000076 014046           MOV     -(R0),-(R6)       ;OUTPUT ILLEGAL BLOCK NUMBER
000100 012746           MOV     #1435,-(R6)       ;AND F035
       001435
000104 000501           BR      DKER20            ;... AFTER SYSDV CHK
000106 060201 DKIN10:   ADD     R2,R1             ;ADD IN VALID QUOTIENT
000110 006202           ASR     R2                ;ADJ REMAINDER FOR DIV BY 12
000112 006202           ASR     R2
000114 060402           ADD     R4,R2
00116 010204 DKIN20:    MOV     R2,R4             ;DIVIDE BY 16 - SAVE REMAINDER
00120 042704            BIC     #177760,R4
      177760
00124 040402            BIC     R4,R2             ;EXTRACT QUOTIENT ...
00126 001367            BNE     DKIN10            ;... IF ANY BUILD RESULT
00130 020427            CMP     R4,#12.           ;CHECK REMAINDER
      000014
00134 002402            BLT     .+6               ;IF BETWEEN 12 & 15 ...
00136 062704            ADD     #4,R4             ;... CAUSE SURFACE INCR.
      000004
00142 060401            ADD     R4,R1             ;PUT SECTOR INTO REST
00144 012704            MOV     #RKDA,R4
      177412
00150 010114            MOV     R1,@R4            ;SET UP DISK ADDRESS
00152 012044            MOV     (R0)+,-(R4)       ;SET UP MEMORY ADDRESS
00154 012044            MOV     (R0)+,-(R4)       ;SET UP WORD COUNT
00156 012001            MOV     (R0)+,R1          ;PUT IN THE FUNCTION
00160 151701            BISB    @PC,R1            ;SET I.D.F. AND GO BITS
00162 042701            BIC     #177460,R1        ;CLEAR GARBAGE -******-
      177460
00166 010144            MOV     R1,-(R4)          ;SEND FUNCTION TO CONTROL
00170 000207            RTS     PC
                  ;     -******- USED AS LITERAL BY THE PREVIOUS INSTRUCTION
```

```
                        ;
                        ;       INTERRUPT PROCESSOR

  000172 013746 DKINT:  MOV     @#V.RSAV,-(R6)
         000044
  000176 004536         JSR     R5,@(R6)+
  000200 016700         MOV     DK,R0           ;GET THE DDB
         177574
  000204 012705         MOV     #RKDS,R5
         177400
  000210 012501         MOV     (R5)+,R1        ;SAVE RKDS AND RKER FOR LATER
  000212 012502         MOV     (R5)+,R2
   00214 011504         MOV     @R5,R4          ;SAVE RKCS
   00216 100405         BMI     DKERP           ;YES - BRANCH
   00220 032704         BIT     #10,R4          ;WAS LAST FCN A DRIVE RESET?
         000010
   00224 001004         BNE     DKER00          ;YES - BRANCH
   00226 000170 DKXIT:  JMP     @14(R0)         ;EXIT
         000014
```

```
                         ;ERROR PROCESSOR:
  000232 006304 DKERP:   ASL     R4                    ;HARD ERROR?
  000234 100440          BMI     DKHER                 ;YES - BRANCH
  000236 006127 DKER00:  ROL     (PC)+                 ;TRIED 8 TIMES?
  000240 000000 DKREPT:  .WORD   0
  000242 103027          BCC     DKER25                ;IF NOT TRY SOME MORE
  000244 052760 DKER10:  BIS     #100000,12(R0)        ;SET FAILURE FLAG
         100000
         000012
  000252 005737          TST     @#RKWC                ;HAS WORD COUNT REACHED 0?
         177406
  000256 001763          BEQ     DKXIT                 ;YES - GO EXIT
   00260 010246 DKER15:  MOV     R2,-(R6)              ;OUTPUT RKER
   00262 012746          MOV     #1427,-(R6)           ;AND F027
         001427
   00266 132760          BITB    #7,13(R0)             ;IS THIS UNIT 0?
         000007
         000013
   00274 001005          BNE     DKER20                ;NO - BRANCH
   00276 032767          BIT     #SDRVR,DKFLGS         ;SYSTEM DRIVER?
         010000
         177476
   00304 001401          BEQ     DKER20                ;NO - BRANCH
   00306 005016          CLR     @R6                   ;SET CODE TO HALT
   00310 005067 DKER20:  CLR     DK                    ;FREE DRIVER
         177464
   00314 000004          IOT                           ;OUTPUT MESSAGE
   00316 012667          MOV     (R6)+,DK              ;IF COME BACK RESET FLAG
         177456
   00322 004767 DKER25:  JSR     PC,DKRTRY             ;RE-INIT TFR
         177516
   00326 013705 DKER30:  MOV     @#V.XIT,R5
         000042
   00332 000165          JMP     4(R5)
         000004
```

```
ØØØ336 Ø12715 DKHER:  MOV     #1,@R5          ;CLEAR THE CONTROL
       ØØØØØ1
ØØØ342 1Ø5715 DKHRØØ: TSTB    @R5             ;DONE YET?
ØØØ344 1ØØ376         BPL     DKHRØØ          ;NO - LOOP
ØØØ346 Ø327Ø1         BIT     #2ØØ,R1         ;IS IT DRIVE NOT READY
       ØØØ2ØØ
ØØØ352 ØØ1416         BEQ     DKRDY           ;IF YES : AØØ2
ØØØ354 Ø327Ø1         BIT     #1ØØØ,R1        ;IS IT SEEK INCOMPLETE?
       ØØ1ØØØ
ØØØ36Ø ØØ14Ø5         BEQ     DKHRØ5          ;NO - BRANCH
ØØØ362 Ø1Ø165         MOV     R1,4(R5)        ;REPLACE DRIVE #
       ØØØØØ4
ØØØ366 Ø12715         MOV     #115,@R5        ;SET UP FOR DRIVE RESET
       ØØØ115
ØØ372 ØØØ755          BR      DKER3Ø          ;TAKE INTERIM EXIT
ØØ374 Ø327Ø2 DKHRØ5:  BIT     #114ØØ,R2       ;CAN WE POSSIBLY GO ON?
       Ø114ØØ
ØØ4ØØ ØØ1316          BNE     DKERØØ          ;YES - BRANCH
ØØ4Ø2 Ø327Ø2          BIT     #2ØØØØ,R2       ;IS IT WRITE LOCK OUT?
       Ø2ØØØØ
ØØ4Ø6 ØØ1724          BEQ     DKER15          ;NO - BRANCH
ØØ41Ø Ø1Ø246 DKRDY:   MOV     R2,-(R6)        ;SAVE BUSY FLAG
ØØ412 Ø16746          MOV     DKNAM,-(R6)     ;OUTPUT NAME
       177376
ØØ416 Ø12746          MOV     #4Ø2,-(R6)      ;AND AØØ2
       ØØØ4Ø2
ØØ422 ØØØ732          BR      DKER2Ø          ;... & GO PRINT
       ØØØØØ1'        .END
```

```
DV.DKH  MACRO V004-14 13-SEP-72 02:52 PAGE 6+
SYMBOL TABLE

DK        000000RG        DKERR     000232R         DKER00    000236R
DKER10    000244R         DKER15    000260R         DKER20    000310R
DKER25    000322R         DKER30    000326R         DKFLGS    000002R
DKHER     000336R         DKHR00    000342R         DKHR05    000374R
DKINT     000172R         DKIN10    000106R         DKIN20    000116R
DKNAM     000014R         DKRDY     000410R         DKREPT    000240R
DKRTRY    000044R         DKSTRT    000040R         DKXIT     000226R
PC      =%000007          PS      = 177776          RKBA    = 177410
RKCS    = 177404          RKDA    = 177412          RKDIR   = 000001
RKDS    = 177400          RKER    = 177402          RKWC    = 177406
R0      =%000000          R1      =%000001          R2      =%000002
R3      =%000003          R4      =%000004          R5      =%000005
R6      =%000006          SDRVR   = 010000          V.RSAV= 000044
V.XIT = 000242
. ABS.    000000     000
          000424     001

ERRORS DETECTED:  0
FREE CORE:  19347. WORDS
,LP:<DT:DK
```

5. VØØ1A Program Listing

```
                                        .TITLE  DK


                                        ;DISK DRIVER (RK11) VERSION V005A
                          ;                                  001
                                        ;COPYRIGHT DIGITAL EQUIPMENT CORPORATION
        000000 R0=%0                    ;MAYNARD, MASSACHUSETTS  October 1971
        000001 R1=%1
        000002 R2=%2
        000003 R3=%3
        000004 R4=%4
        000005 R5=%5
        000006 R6=%6
        000007 PC=%7
        177400 RKDS=177400
        177402 RKER=177402
        177404 RKCS=177404
        177406 RKWC=177406
        177410 RKBA=177410
        177412 RKDA=177412
        000001 RKDIR=1
        000044 V.RSAV=44
        000042 V.XIT=42
        177776 PS=177776


                          .GLOBL  DK

                  ;STANDARD DOS INTERFACE TABLE

000000 000000 DK:         .WORD   0                   ;0 IF IDLE, DDB PTR OTHERWISE
000002 102037             .WORD   102037              ;FACILITES WORD
000004    020             .BYTE   20                  ;STD BUFFER SIZE IS 256. WORDS
000005    172             .BYTE   DKINT-DK            ;OFFSET TO INTERRUPT HANDLER
000006    240             .BYTE   240                 ;PRIORITY LEVEL 5
000007    000             .BYTE   0                   ;NO OPEN ROUTINE
000010    040             .BYTE   DKSTRT-DK           ;OFFSET TO TRANSFER HANDLER
000011    000             .BYTE   0                   ;NO CLOSE ROUTINE
000012    000             .BYTE   0                   ;NO SPECIAL FUNCTIONS
000013    000             .BYTE   0                   ;CURRENTLY UNUSED
000014 015270 DKNAM:      .RAD50  /DK/                ;DEVICE NAME
000016 000001             .WORD   RKDIR               ;FIRST MFD BLOCK
000020 000000             .WORD   0,0,0,0,0,0,0,0     ;BIT MAP POINTERS
000022 000000
000024 000000
000026 000000
000030 000000
000032 000000
000034 000000
000036 000000

000040 011767 DKSTRT:     MOV     @PC,DKREPT          ;CLEAR RETRY INDICATOR
       000174
000044 116001 DKRTRY:     MOVB    13(R0),R1           ;GET UNIT IN R1(13-15)
       000013
000050 042701             BIC     #177770,R1
       177770
                          .IFDF   MIXED
                          .IFNDF  LOWDEN
                          MOV     R1,R4               ;SAVE UNIT FOR LATER USE
                          .ENDC
                          .ENDC
```

```
000054 006201            ASR     R1              ;LEFT-JUSTIFY UNIT
000056 006001            ROR     R1
000060 006001            ROR     R1
000062 006001            ROR     R1              ;UNIT NOW AS DESIRED
000064 022020            CMP     (R0)+,(R0)+     ;POINTER DDB+BLOCK
000066 012002            MOV     (R0)+,R2
                         .IFDF   MIXED
                         .IFNDF  LOWDEN
                         MOV     (PC)+,R3        ;GET DENSITY PATTERN
                         .WORD   CONFIG
                         ASL     R3              ;MOVE APPROP. TO UNIT
                         DEC     R4
                         BGE     .-4
                         BCC     .+4             ;IF LOW DENSITY ...
                         ASL     R2              ;ADJUST BLOCK NO.
                         .ENDC
                         .ENDC
                         .IFDF   LOWDEN
                         ASL     R2
                         .ENDC


000070 020227            CMP     R2,#4800.       ;IS BLOCK WITHIN BOUNDS?
       011300
000074 103410            BLO     DKIN20          ;YES - BRANCH
000076 014046            MOV     -(R0),-(R6)     ;OUTPUT ILLEGAL BLOCK NUMBER
000100 012746            MOV     #1435,-(R6)     ;AND F035
       001435
000104 000470            BR      DKER20          ;... AFTER SYSDV CHK
000106 060201 DKIN10:    ADD     R2,R1           ;ADD IN VALID QUOTIENT
000110 006202            ASR     R2              ;ADJ REMAINDER FOR DIV BY 12
000112 006202            ASR     R2
000114 060402            ADD     R4,R2
000116 010204 DKIN20:    MOV     R2,R4           ;DIVIDE BY 16 - SAVE REMAINDER
000120 042704            BIC     #177760,R4
       177760
000124 040402            BIC     R4,R2           ;EXTRACT QUOTIENT ...
000126 001367            BNE     DKIN10          ;... IF ANY BUILD RESULT
000130 020427            CMP     R4,#12.         ;CHECK REMAINDER
       000014
000134 002402            BLT     .+6             ;IF BETWEEN 12 & 15 ...
000136 062704            ADD     #4,R4           ;... CAUSE SURFACE INCR.
       000004
000142 060401            ADD     R4,R1           ;PUT SECTOR INTO REST
000144 012704            MOV     #RKDA,R4
       177412
000150 010114            MOV     R1,@R4          ;SET UP DISK ADDRESS
000152 012044            MOV     (R0)+,-(R4)     ;SET UP MEMORY ADDRESS
000154 012044            MOV     (R0)+,-(R4)     ;SET UP WORD COUNT
000156 012001            MOV     (R0)+,R1        ;PUT IN THE FUNCTION
000160 151701            BISB    @PC,R1          ;SET I.D.E. AND GO BITS
000162 042701            BIC     #177460,R1      ;CLEAR GARBAGE -******-
       177460
000166 010144            MOV     R1,-(R4)        ;SEND FUNCTION TO CONTROL
000170 000207            RTS     PC
                ;        -******- USED AS LITERAL BY THE PREVIOUS INSTRUCTION
```

```
                        .GLOBL  S.RSAV,S.XIT
                ;
                ;       INTERRUPT PROCESSOR

                DKINT:  .IFDF   SYSDV
                        JSR     R5,S.RSAV               ;SAVE REGISTERS
                        .ENDC
                        .IFNDF  SYSDV
000172 013746           MOV     @#V.RSAV,-(R6)
       000044
000176 004536           JSR     R5,@(R6)+
                        .ENDC
000200 016700           MOV     DK,R0                   ;GET THE DDB
       177574
000204 012705           MOV     #RKDS,R5
       177400
000210 012501           MOV     (R5)+,R1                ;SAVE RKDS AND RKER FOR LATER
000212 012502           MOV     (R5)+,R2
000214 011504           MOV     @R5,R4                  ;SAVE RKCS
000216 100405           BMI     DKERP                   ;YES - BRANCH
000220 032704           BIT     #10,R4                  ;WAS LAST FCN A DRIVE RESET?
       000010
000224 001004           BNE     DKER00                  ;YES - BRANCH
000226 000170   DKXIT:  JMP     @14(R0)                 ;EXIT
       000014

                ;ERROR PROCESSOR:
000232 006304   DKERP:  ASL     R4                      ;HARD ERROR?
000234 100425           BMI     DKHER                   ;YES - BRANCH
000236 006127   DKER00: ROL     (PC)+                   ;TRIED 8 TIMES?
000240 000000   DKREPT: .WORD   0
000242 103014           BCC     DKER25                  ;IF NOT TRY SOME MORE
000244 052760   DKER10: BIS     #100000,12(R0)          ;SET FAILURE FLAG
       100000
       000012
000252 005737           TST     @#RKWC                  ;HAS WORD COUNT REACHED 0?
       177406
000256 001763           BEQ     DKXIT                   ;YES - GO EXIT
000260 010246   DKER15: MOV     R2,-(R6)                ;OUTPUT RKER
000262 012746           MOV     #1427,-(R6)             ;AND F027
       001427
                        .IFDF   SYSDV
                        BITB    #7,13(R0)               ;WAS THIS UNIT 0?
                        BNE     DKER20                  ;NO - BRANCH
                        CLR     @R6
                        .ENDC
                DKER20: .IFDF   SYSDV
                        CLR     DK                      ;CLEAR BUSY FLAG
                        .ENDC
000266 000004           IOT                             ;OUTPUT MESSAGE
000270 012667           MOV     (R6)+,DK                ;IF COME BACK RESET FLAG
       177504
000274 004767   DKER25: JSR     PC,DKRTRY               ;RE-INIT TFR
       177544
                DKER30: .IFDF   SYSDV
                        JMP     S.XIT+4                 ;... & TAKE INTERIM EXIT
                        .ENDC
                        .IFNDF  SYSDV
000300 013705           MOV     @#V.XIT,R5
       000042
000304 000165           JMP     4(R5)
       000004
                        .ENDC
```

```
000310 012715 DKHER:  MOV     #1,@R5          ;CLEAR THE CONTROL
       000001
000314 105715 DKHR00: TSTB    @R5             ;DONE YET?
000316 100376         BPL     DKHR00          ;NO - LOOP
000320 032701         BIT     #1000,R1        ;IS IT SEEK INCOMPLETE?
       001000
000324 001405         BEQ     DKHR05          ;NO - BRANCH
000326 010165         MOV     R1,4(R5)        ;REPLACE DRIVE #
       000004
000332 012715         MOV     #115,@R5        ;SET UP FOR DRIVE RESET
       000115
000336 000760         BR      DKER30          ;TAKE INTERIM EXIT
000340 032702 DKHR05: BIT     #11400,R2       ;CAN WE POSSIBLY GO ON?
       011400
000344 001334         BNE     DKER00          ;YES - BRANCH
000346 032702         BIT     #20000,R2       ;IS IT WRITE LOCK OUT?
       020000
000352 001742         BEQ     DKER15          ;NO - BRANCH
000354 010046         MOV     R0,-(R6)        ;SAVE BUSY FLAG
000356 016746         MOV     DKNAM,-(R6)     ;OUTPUT NAME
       177432
000362 012746         MOV     #402,-(R6)      ;AND A002
       000402
000366 000737         BR      DKER20          ;... & GO PRINT
       000001         .END


000000 ERRORS


DK         000000RG     DKERP     000232R      DKER00     000236R
DKER10     000244R      DKER15    000260R      DKER20     000266R
DKER25     000274R      DKER30    000300R      DKHER      000310R
DKHR00     000314R      DKHR05    000340R      DKINT      000172R
DKIN10     000106R      DKIN20    000116R      DKNAM      000014R
DKREPT     000240R      DKRTRY    000044R      DKSTRT     000040R
DKXIT      000226R      PC       =%000007      PS       = 177776
RKBA     = 177410       RKCS     = 177404      RKDA     = 177412
RKDIR    = 000001       RKDS     = 177400      RKER     = 177402
RKWC     = 177406       R0       =%000000      R1       =%000001
R2       =%000002       R3       =%000003      R4       =%000004
R5       =%000005       R6       =%000006      S.RSAV   = ****** G
S.XIT    = ****** G     V.RSAV   = 000044      V.XIT    = 000042
.        = 000370R
```

DEC-11-ODDTA-A-D

PDP-11

TC11 DECTAPE DRIVER

October 1972

SUPPLEMENT TO:

PDP-11 DEVICE DRIVER PACKAGE

DEC-11-ODDPA-A-D

MONITOR VERSION VØØ8

NOTE

This document is for information purposes only and is subject to change without notice. DEC assumes no responsibility for the use or reliability of its software on equipment which is not supplied by DEC.

NEW AND CHANGED INFORMATION

This manual documents the software as of Monitor version VØ8. It has been revised to include all new and changed material since Monitor version VØ4. Such material is indicated by vertical bars in the outside margin. Whole new pages are not so marked but are dated in the lower outside corner.

pdp11 SOFTWARE DISTRIBUTION CENTER

DRIVER for TC11 DECtape Control

The principal function of the TC11 Driver is to transfer data between the hardware control and a memory area specified by a calling Monitor routine on behalf of a user program. The number of words transferred, the DECtape transport, the absolute starting block on the tape, and the direction of tape travel in each case are all determined by the calling routine.

As required by the standard Monitor-driver interface for all devices and, as DECtape will be handled as such, for file-structured devices in particular, the first part of the driver consists of two consecutive tables:

a) Table of descriptors and pointers to routines included.
b) File-structured usage data

All data transfers utilize the normal read/write capability of the PDP-11 NPR facility. The driver contains a setup sequence to initiate a search for the requisite start block and routines then to handle interrupts for continuation of such search and, if this is successful, the subsequent data transfer specified.

As a file-structured device, the opening and closing of files are the responsibility of the Monitor file management routines. There are therefore no OPEN or CLOSE routines.

Also, no routine to handle SPECIAL FUNCTIONS is currently provided. This could be added later if it is found desirable to simulate the normal operation of some similar device, e.g., rewind as for Magnetic Tape.

1. Initial Tables

Relevant entries for this driver are as follows:

WORD Ø: = Ø initially-set to address of DDB for dataset being serviced when busy, by calling routine.

WORD 1: = Facility Pattern = 140037 signifying:

a) File-structured Device
b) DECtape (or similar reversible medium)

c) Capable of Input or Output in either ASCII or Binary on more than one dataset at a time.

WORD 2: = a) Standard Buffer Size = 16 X 16-word units (i.e., 1 standard DECtape block).

b) Offset to Interrupt Service routine.

WORD 3: = a) Priority for Interrupt Service = 7

b) Ø [No OPEN routine included]

WORD 4: = a) Offset to TRANSFER Set-up routine

b) Ø [No CLOSE routine included]

WORD 5: = Ø [No SPEC FUNC routine present]

WORD 6: = Name 'DT' in RADIX 50 format.

WORD 7: = Start Block of Directory Structure = 100

WORDS 1Ø-17: = Reserved for pointers to in-core Bit Maps for each of 8 transports supportable by TC11.

2. Processing Routines

2.1 Transfer Setup

A Monitor routine effectively calls for transfer setup by JSR PC XXXX where XXXX is the start address evaluated from the offset in WORD 4 of the table. The address of the DDB containing relevant parameters will be stored in WORD Ø of the table.

The setup routine will first set a counter for number of returns to be made in the event of parity or timing failures in tape operations (8-9). Using the given DDB address, it then extracts the following information and actions it as shown:

(i) Block No. (DDB+4) - two copies are stored internally as controls during Start Block search as detailed below.

(ii) Word Count & Memory Address (DDB+6 & 10) - these are stored immediately in the TC11 WC & BA registers for use as soon as the Start Block has been found.

(iii) Function (DDB+12) - the requirement for Read or Write is converted from the standard Monitor specification (4 or 2) into the corresponding DECtape value (4 or 14) and stored internally until completion of block search.

(iv) Tape Unit & Motion (DDB+13). The bits showing these are associated with the DECtape Search function [3] and are set into the TC11 Control Register to initiate the search for the start block.

The setup routines also sets two switches appropriately:

a) In any transfer, two types of interrupt may occur; the first at each block encountered during the search for the start specified; the second thereafter arising when the transfer has been completed. The switch is initially set for the first type.

b) The tape is started in the eventual transfer direction. Turn-around, however, may be necessary if the tape is badly positioned. The second switch is set initially to reflect the start direction in order to provide adequate control during such turn-around.

The driver then sets the TC11 Control Register for the search, and restores control to the calling Monitor routine, via RTS PC, to await its first interrupt.

As permitted by the General Driver Spec, the setup routine makes full use of the processor registers, without saving or restoring their original content.

2.2 Interrupt Servicing - Search Mode

Provided that a tape block-mark is encountered without error, the search interrupt servicing routine compares the number found (from TC11 Data Register) with one copy of that for the required block, stored internally by SETUP. If the comparison shows that current tape-motion will eventually lead to the required block, the routine exits immediately and waits for a subsequent interrupt to show that the transfer may begin.

If tape-motion is in the wrong direction, the routine resets the TC11 Control register to produce tape turn-around on exit. A second turn-around will now be essential for a transfer in the require direction. The routine therefore modifies, appropriately, by 2 the copy of the block number required used in the comparison. This factor is provided so the tape is sufficiently positioned beyond the block required to ensure that it will be up to speed at the right point after the second turn. For example, in order to transfer Block 100 forward, the first turn will seek Block 76 in reverse.

An equal comparison might then result after a single turn-around. The block number found is, therefore, checked against the second, unmodified, stored value. If not equal, a turn-around has occurred: the TC11 is reset for the second time and the first stored number is restored to its original value. When both stored values and the block found are all equal, the correct tape travel is assumed and the transfer is effected by moving the stored function into the TC11 control (byte only to avoid hardware delay imposition). The interrupt switch is changed to show that the operation is now in Transfer Mode.

In the event of an error in Search Mode, the TC11 Test Register is examined. If this shows that the cause is "End Zone Reached", the turn-around procedure is again effected, since such a condition is initially the same as being, for example, at Block 102 when 100 is wanted forwards. All other hardware-reported errors are treated as discussed in a subsequent paragraph.

Another type of error may occur but this can only be detected by software, i.e., a failure to find the block either because its number on the tape is corrupted or the one required is outside the range of the tape. For both situations the tape might rock endlessly owing to the turn-around algorithm. The search interrupt processor therefore counts the number of times a turn is effected. It gives up at the sixth attempt and requests printing of an FØ16 message with the failing Block Number as evidence.

To avoid unnecessary time wastage in the storage and retrieval of their contents, the normal search interrupt processing does not use processor registers.

2.3 Interrupt Servicing - Transfer Mode

The normal cause of an interrupt in transfer mode is the satisfactory completion of the whole of the data transfer specified. The driver must then recall the monitor routine which requested the transfer. Because this routine may have surrendered control to the user program during the period of the search and transfer operations, the driver must assume such is the case and save all register contents before setting RØ to the DDB address from its WORD Ø and taking the completion return set into DDB+14.

The interrupt may also occur if an error is determined by examination of the TC11 Test Register. In Transfer Mode, two types of errors specifically processed are Party or Timing Failure. Following either of these, the servicing routine restarts the whole process over from the original block search until at least 8 attempts to produce a satisfactory transfer have been made. If these all fail, the routine returns a flag indicating the error in Bit 15 of the relevant DDB+12.

It checks, however, whether the failure occurred at an intermediate block of a transfer involving several blocks. If such is the case, it endeavors to provide a satisfactory transfer of the remaining blocks. It then recalls the monitor at the completion return address.

Of the other types of error, transfer mode servicing also handles Non-existent Memory and End Zone. Both of these conditions are assumed to be the result of a programming error and cause printing of a fatal error message FØ15 with User Call Address as evidence.

2.4 Recoverable Errors

In both Search and Transfer modes, for errors not especially noted, a general routine is used to request printing of a diagnostic message requesting operator action. SEL and ILO errors are assumed to indicate a "Device Not Ready" state for which the device name (DT) is supporting evidence for the message 'AØØ2'. For the rest, and Mark Track Errors in particular, which might be resolved by changing tapes -- the message 'AØØ3' is printed with the TC11 Test Register content as evidence. For all these errors, the operator might request program resumption by a Monitor "Continue" command. The driver restarts the whole search and transfer process if this occurs.

3. Implementation

a. Comments on the driver listing show general methods of implementation. It should be noted, however, that in several instances, in-line code is modified. In particular, the two switches mentioned under "Setup" are variable Branch Instructions and the internal storage of data has already been indicated. This means first that the driver is not reentrant - an unlikely requirement when one control may only service the transport at a time, even though eight may be attached to it. In the second place, the driver, as written is not immediately usable in a ROM.

b. The priority level for interrupt servicing should also be mentioned. The hardware level is 6; the initial software level, however, is set at 7. This is to ensure that there will be no delay due to any other interrupt in the critical case in which the required block number has been found and a change of function from Search to Read or Write must occur within 400 msecs. The interrupt routines themselves lower the level to 6, if the critical case is not being actioned. This will mean that other interrupts may be delayed up to 50 msecs. in the worst case, the critical one.

c. A further minor point of interest is that the tape is always stopped at the end of each transfer (or when an error occurs to prevent this) in order to maintain correct tape positioning. A program STOP request is issued to effect this in all cases, even though the hardware may be set up to provide for it. However, resetting the TC11 Status Register for this purpose can remove error conditions. The content of this register is, therefore, examined (or is saved for later examination) before the STOP command is given.

4. PROGRAM LISTINGS

4.1 VØ2 Program Listing

```
;COPYRIGHT 1971, DIGITAL EQUIPMENT CORP., MAYNARD, MASS.

;VERSION NUMBER:        V02

        .TITLE  DV.DT
;
        .GLOBL  DT
;DECTAPE DRIVER          VERSION 1        23 JULY 70
;       PRESENTLY CONTAINS ONLY ROUTINE FOR TRANSFER
;
;STANDARD DRIVER TABLE:
DT:     .WORD   0                 ;BUSY FLAG (DDB ADDR WHEN BUSY)
        .BYTE   37,300            ;FACILITY INDICATOR
        .BYTE   16.               ;STD BUFF SIZE/16.
        .BYTE   DT.INT-DT         ;POINTER TO INT SVCE
        .BYTE   340               ;INT SVCE PRIORITY
        .BYTE   0                 ;DESPATCH TABLE ....
        .BYTE   DT.TFR-DT         ;...FOR TRANSFER ONLY!
        .BYTE   0
        .BYTE   0
        .BYTE   0                 ;SPARE
DT.NAM: .RAD50  'DT'
        .WORD   DT.DIR            ;FIXED MFD BLOCK
        .WORD   0,0,0,0,0,0,0,0 ;POINTERS FOR BIT MAP ACCESS
;REGISTER ASSIGNMENTS:
R0=%0
R1=%1
R2=%2
R3=%3
R4=%4
R5=%5
SP=%6
PC=%7
;SET UP TRANSFER:
DT.TFR: MOV     @PC,DT.RTC        ;SET RETRY COUNT
DT.PR1: MOV     DT,R0             ;GET ADDRESS OF DDB ...
        MOV     #DT.CBA,R1        ;... & OF HWR BLOCK
        CLR     @R1
        CMP     (R0)+,(R0)+       ;SKIP USER LINE IN DDB
        MOV     (R0)+,DT.BRQ      ;SAVE BLOCK NO FOR LATER
        MOV     (R0)+,@R1         ;SET READY MEMORY ADDR ...
        MOV     (R0)+,-(R1)       ;... & WORD COUNT
DT.PR2: CLRB    DT.INT            ;SET INT'RUPT SW. TO SRCH
        MOV     DT.BRQ,DT.BCK     ;SET BLK CTRL FOR SRCH
        MOV     #100,R3           ;USED IN NEXT SEQUENCE
        MOV     R3,DT.TAC         ;SET TURN AROUND COUNT
        MOV     @R0,-(SP)         ;GET UNIT, DIRECTION & FUNC
        BIC     #170341,@SP       ;CLEAR POSS. GARBAGE
        BIS     R3,@SP            ;ADD IN INT ENB BIT
        BITB    @SP,@PC           ;WRITE REQD?
        BEQ     .+6               ;(READ O.K. ALRDY)*****
        ADD     #12,@SP           ;IF SO GET DECTAPE EQUIV.
        MOVB    @SP,DT.FRQ        ;SAVE FUNC FOR LATER
        MOVB    @PC,@SP           ;RESET FUNC TO SRCH (INT ENB)
        ASL     R3                ;(NOW CONTAINS 200)*****
        BIT     @SP,#4000         ;TRAVEL FORWARD?
        BNE     .+4
        INC     R3                ;IF SO R3 NOW 201 & SO ...
        MOVB    R3,DT.SSW         ;MAKING BPL OR BMI AS REQD
        MOV     (SP)+,-(R1)       ;SET DECTAPE CONTROL
        RTS     PC                ;RETURN TO CALLER FOR NOW
;***** CARE USED AS LITERAL BY PREVIOUS INSTRUCTION!!
```

```
;INTERRUPT SERVICE (A) - SEARCH IN PRORESS:
DT.SIP: TST     @#DT.CCM          ;CHECK STATUS
        BMI     DT.SER            ;IF ERROR GO INVESTIGATE
        CMP     @#DT.CDT,DT.BRQ   ;CHECK BLOCK FOUND
        BEQ     DT.BFD            ;IF ONE REQD, GO ACTION
        BMI     DT.SXT            ;GET TO BLOCK THIS WAY?
DT.SSW=.-1                        ;(BPL IF TRAVEL BACKWARD)
DT.TA1: BICB    #40,@#177776      ;DROP PRIORITY
        ASRB    #0                ;HOW MANY TURNS?
DT.TAC=.-2
        BCS     DT.BER            ;IF 6 CAN'T FIND BLOCK
        MOV     #4000,-(SP)       ;OTHERWISE MUST TURN AROUND
        MOV     #2,-(SP)          ;ASSUME TRAVEL NOW FWD
        RORB    DT.SSW            ;CHECK DIRECTION
        BCS     DT.TA2            ;IF FWD OMIT NEXT
        NEG     2(SP)             ;IF BWD, REVERSE EVERYTHING
        NEG     @SP
DT.TA2: SUB     (SP)+,DT.BRQ      ;ALLOW 2 BLKS FOR 2ND TURN
        ADD     (SP)+,@#DT.CCM    ;SWITCH STATUS
        ROLB    DT.SSW            ;RESET DIR SW (C BIT REVERSES)
DT.SXT: INCB    @#DT.CCM          ;CONTINUE SEARCH
        RTI                       ;WAIT NEXT BLOCK
;BLOCK FOUND - CHECK TRAVEL CORRECT:
DT.BFD: CMP     #0,#0             ;TRAVEL AS ORIGINALLY STORED?
DT.BRQ=.-4
DT.BCK=.-2
        BNE     DT.TA1            ;IF NOT MUST TURN AGAIN
        INCB    DT.INT            ;RESET INT'RUPT SW FOR TFR
        MOVB    #0,@#DT.CCM       ;MOVE IN CORRECT FUNC
DT.FRQ=.-4
        BR      DT.SXT            ;... & GO SET UNDERWAY
;INTERRUPT SERVICE (B) - TRANSFER COMPLETE (?):
DT.INT: BR      .+2               ;INTERRUPT SWITCH ....
        BR      DT.SIP            ;FOR SRCH COMES HERE!
        BICB    #40,@#177776      ;DROP PRIORITY
        MOV     @#V.RSAV,-(SP)    ;ON TRANSFER COMPLETE ...
        JSR     R5,@(SP)+         ;SAVE USER REGISTERS
        MOV     DT,R0             ;GET DDB ADDR
        MOV     #DT.CCM,R1        ;GET STATUS ADDR
        MOV     #10,R3            ;SET MAGIC CONSTANT
        TST     @R1               ;ERROR CAUSE INT'RUPT?
        BMI     DT.TER            ;IF SO GO & SEE WHY
        MOVB    R3,@R1            ;OTHERWISE STOP TAPE ...
DT.TXT: MOV     14(R0),PC         ;... & TAKE COMPLETE RETN
;SEARCH ERROR - DETERMINE CAUSE:
DT.SER: TST     @#DT.TST          ;IN END ZONE?
        BMI     DT.TA1            ;O.K. MEANS TURN AROUND
        BICB    #40,@#177776      ;DROP PRIORITY
        MOV     @#V.RSAV,-(SP)    ;SAVE ALL USER REGS,
        JSR     R5,@(SP)+
        MOV     #DT.TST,R1        ;GET DECTAPE STATUS
DT.EXT: MOV     @R1,-(SP)         ;SET UP TO TELL USER
        MOV     #DT.IRE,-(SP)
        BIT     #14000,(R1)+      ;.... ASSUMING H-W FAILURE
        BEQ     DT.STP            ;.... IF SEL OR ILO
        MOV     #DT.NRE,@SP       ;DIAGNOSE TAPE FAULT DIFF.
        MOV     DT.NAM,2(SP)      ;... AS NOT READY
DT.STP: MOVB    #10,@R1           ;STOP TAPE IN CASE
        IOT                       ;GO TO DIAG PRINT
DT.RXT: JSR     PC,DT.PRI         ;ON RECOVERY, SET UP RETRY
        MOV     @#V.RRES,R5       ;RESTORE USER REGS
        JSR     R5,@R5
        RTI                       ;... & HOPE FOR BETTER THINGS!
;BLOCK NOT FOUND IN SEARCH:
DT.BER: MOV     DT.BCK,-(SP)      ;GIVE BLOCK NO. AS EVIDENCE
        MOV     #DT.BRE,-(SP)
        MOV     #DT.CCM,R1        ;GET CONTROL ADDRESS
        BR      DT.STP
```

```
;TRANSFER ERROR:
DT.TER: BIT     #34000,-(R1)     ;TAPE FAILURE/OPERATOR FAULT?
        BNE     DT.EXT           ;IF SO PRINT & WAIT RECOVERY
        BIT     #100400,(R1)+    ;END ZONE/N.E.M?
        BNE     DT.FER           ;IF SO TREAT AS FATAL
;RECOVERABLE ERRORS (TIMING OR PARITY):
        ASL     #0               ;RETRIED 8-9 TIMES ALRDY?
DT.RTC=.-2
        BCC     DT.RXT           ;IF NOT TRY AGAIN ....
        BIS     #100000,12(R0)   ;OTHERWISE SIGNAL ERROR
        MOVB    R3,(R1)+         ;STOP TAPE IN CASE
        MOV     1(R1),R2         ;...BUT CHK ALL WORDS DONE!
        BEQ     DT.TXT           ;IF SO THAT'S IT!
        ADD     R3,R0            ;GO TO WORD COUNT IN DDB
        SUB     (R0)+,R2         ;... & USE TO DETERMINE ...
        SWAB    R2               ;... NO. OF BLOCKS DONE
        BITB    R3,(R1)+         ;CHECK PRESENT TRAVEL
        BEQ     .+4              ;ADJUST NO. ACCORDINGLY
        NEG     R2
        ADD     R2,DT.BRQ        ;MODIFY SEARCH START BLOCK
        CLR     DT.RTC           ;... & RETRY COUNT
        JSR     PC,DT.PR2        ;GO SET UP NEW START
        BR      DT.RXT+4         ;... & WAIT RESULTS!
;FATAL ERRORS - END ZONE OR NON-EXISTENT MEMORY:
DT.FER: MOV     @R0,-(SP)        ;GIVE CALL AS EVIDENCE
        MOV     #DT.FRE,-(SP)    ;PRINT DIAGNOSIS
        BR      DT.STP
;MISCELLANEOUSDEFINITIONS:
V.RSAV=44
V.RRES=46
DT.DIR=100
DT.TST=177340
DT.CCM=177342
DT.CBA=177346
DT.CDT=177350
DT.NRE=402
DT.IRE=404
DT.FRE=1415
DT.BRE=1416
        .END
```

4.2 VØØ1A Program Listing

A complete assembly listing of the driver follows.

```
                    ;COPYRIGHT 1971, DIGITAL EQUIPMENT CORP., MAYNARD, MASS.

                    ;VERSION NUMBER:         V001A

                            .TITLE  DT
                    ;
                            .GLOBL  DT
                    ;DECTAPE DRIVER          VERSION 1        23 JULY 70
                    ;       PRESENTLY CONTAINS ONLY ROUTINE FOR TRANSFER
                    ;
                    ;STANDARD DRIVER TABLE:
000000 000000 DT:           .WORD   0                  ;BUSY FLAG (DDB ADDR WHEN BUSY)
000002    037               .BYTE   37,300             ;FACILITY INDICATOR
000003    300
000004    020               .BYTE   16.                ;STD BUFF SIZE/16.
000005    310               .BYTE   DT.INT-DT          ;POINTER TO INT SVCE
000006    340               .BYTE   340                ;INT SVCE PRIORITY
000007    000               .BYTE   0                  ;DESPATCH TABLE ....
000010    040               .BYTE   DT.TFR-DT          ;...FOR TRANSFER ONLY!
000011    000               .BYTE   0
000012    000               .BYTE   0
000013    000               .BYTE   0                  ;SPARE
000014 016040 DT.NAM:       .RAD50  'DT'
000016 000170               .WORD   DT.DIR             ;FIXED MFD BLOCK
000020 000000               .WORD   0,0,0,0,0,0,0,0    ;POINTERS FOR BIT MAP ACCESS
000022 000000
000024 000000
000026 000000
000030 000000
000032 000000
000034 000000
000036 000000
                    ;REGISTER ASSIGNMENTS:
       000000 R0=%0
       000001 R1=%1
       000002 R2=%2
       000003 R3=%3
       000004 R4=%4
       000005 R5=%5
       000006 SP=%6
       000007 PC=%7

                    ;SET UP TRANSFER:
000040 011767 DT.TFR: MOV   @PC,DT.RTC         ;SET RETRY COUNT
       000444
000044 016700 DT.PR1: MOV   DT,R0              ;GET ADDRESS OF DDB ...
       177730
000050 012701         MOV   #DT.CBA,R1         ;... & OF HWR BLOCK
       177346
000054 005011         CLR   @R1
000056 022020         CMP   (R0)+,(R0)+        ;SKIP USER LINE IN DDB
000060 012067         MOV   (R0)+,DT.BRQ       ;SAVE BLOCK NO FOR LATER
       000202
000064 012011         MOV   (R0)+,@R1          ;SET READY MEMORY ADDR ...
000066 012041         MOV   (R0)+,-(R1)        ;... & WORD COUNT
000070 105067 DT.PR2: CLRB  DT.INT             ;SET INT'RUPT SW. TO SRCH
       000214
000074 016767         MOV   DT.BRQ,DT.BCK      ;SET BLK CTRL FOR SRCH
       000156
       000166
000102 012703         MOV   #100,R3            ;USED IN NEXT SEQUENCE
       000100
```

```
000106 010367           MOV     R3,DT.TAC          ;SET TURN AROUND COUNT
       000100
000112 011046           MOV     @R0,-(SP)          ;GET UNIT, DIRECTION & FUNC
000114 042716           BIC     #170341,@SP        ;CLEAR POSS. GARBAGE
       170341
000120 050316           BIS     R3,@SP             ;ADD IN INT ENB BIT
000122 131617           BITB    @SP,@PC            ;WRITE REQD?
000124 001402           BEQ     .+6                ;(READ O.K. ALRDY)*****
000126 062716           ADD     #12,@SP            ;IF SO GET DECTAPE EQUIV.
       000012
000132 111667           MOVB    @SP,DT.FRG         ;SAVE FUNC FOR LATER
       000144
000136 111716           MOVB    @PC,@SP            ;RESET FUNC TO SRCH (INT ENB)
000140 006303           ASL     R3                 ;(NOW CONTAINS 200)*****
000142 031627           BIT     @SP,#4000          ;TRAVEL FORWARD?
       004000
000146 001001           BNE     .+4
000150 005203           INC     R3                 ;IF SO R3 NOW 201 & SO ...
000152 110367           MOVB    R3,DT.SSW          ;MAKING BPL OR BMI AS REQD
       000023
000156 012641           MOV     (SP)+,-(R1)        ;SET DECTAPE CONTROL
000160 000207           RTS     PC                 ;RETURN TO CALLER FOR NOW
                ;***** CARE USED AS LITERAL BY PREVIOUS INSTRUCTION!!!

                ;INTERRUPT SERVICE (A) - SEARCH IN PRORESS:
000162 005737   DT.SIP: TST     @#DT.CCM           ;CHECK STATUS
       177342
000166 100473           BMI     DT.SER             ;IF ERROR GO INVESTIGATE
000170 023767           CMP     @#DT.CDT,DT.BRQ    ;CHECK BLOCK FOUND
       177350
       000070
000176 001432           BEQ     DT.BFD             ;IF ONE REQD, GO ACTION
000200 100426           BMI     DT.SXT             ;GET TO BLOCK THIS WAY?
       000201   DT.SSW=.-1                         ;(BPL IF TRAVEL BACKWARD)
000202 142737   DT.TA1: BICB    #40,@#177776       ;DROP PRIORITY
       000040
       177776
000210 106227           ASRB    #0                 ;HOW MANY TURNS?
       000000
       000212   DT.TAC=.-2
000214 103517           BCS     DT.BER             ;IF 6 CAN'T FIND BLOCK
000216 012746           MOV     #4000,-(SP)        ;OTHERWISE MUST TURN AROUND
       004000
000222 012746           MOV     #2,-(SP)           ;ASSUME TRAVEL NOW FWD
       000002
000226 106067           RORB    DT.SSW             ;CHECK DIRECTION
       177747
000232 103403           BCS     DT.TA2             ;IF FWD OMIT NEXT
000234 005466           NEG     2(SP)              ;IF BWD, REVERSE EVERYTHING
       000002
000240 005416           NEG     @SP
000242 162667   DT.TA2: SUB     (SP)+,DT.BRQ       ;ALLOW 2 BLKS FOR 2ND TURN
       000020
000246 062637           ADD     (SP)+,@#DT.CCM     ;SWITCH STATUS
       177342
000252 106167           ROLB    DT.SSW             ;RESET DIR SW (C BIT REVERSES)
       177723
000256 105237   DT.SXT: INCB    @#DT.CCM           ;CONTINUE SEARCH
       177342
000262 000002           RTI                        ;WAIT NEXT BLOCK
```

```
                ;BLOCK FOUND - CHECK TRAVEL CORRECT:
000264 022727 DT.BFD: CMP     #0,#0           ;TRAVEL AS ORIGINALLY STORED?
       000000
       000000
       000266 DT.BRO=.-4
       000270 DT.BCK=.-2
000272 001343         BNE     DT.TA1          ;IF NOT MUST TURN AGAIN
000274 105267         INCB    DT.INT          ;RESET INT'RUPT SW FOR TFR
       000010
000300 112737         MOVB    #0,@#DT.CCM     ;MOVE IN CORRECT FUNC
       000000
       177342
       000302 DT.FRO=.-4
000306 000763         BR      DT.SXT          ;... & GO SET UNDERWAY
                ;INTERRUPT SERVICE (B) - TRANSFER COMPLETE (?):
000310 000400 DT.INT: BR      .+2             ;INTERRUPT SWITCH ....
000312 000723         BR      DT.SIP          ;FOR SRCH COMES HERE!
000314 142737         BICB    #40,@#177776    ;DROP PRIORITY
       000040
       177776
000322 013746         MOV     @#V.RSAV,-(SP)  ;ON TRANSFER COMPLETE ...
       000044
000326 004536         JSR     R5,@(SP)+       ;SAVE USER REGISTERS
000330 016700         MOV     DT,R0           ;GET DDB ADDR
       177444
000334 012701         MOV     #DT.CCM,R1      ;GET STATUS ADDR
       177342
000340 012703         MOV     #10,R3          ;SET MAGIC CONSTANT
       000010
000344 005711         TST     @R1             ;ERROR CAUSE INT'RUPT?
000346 100451         BMI     DT.TER          ;IF SO GO & SEE WHY
000350 110311         MOVB    R3,@R1          ;OTHERWISE STOP TAPE ...
000352 016007 DT.TXT: MOV     14(R0),PC       ;... & TAKE COMPLETE RETN
       000014

                ;SEARCH ERROR - DETERMINE CAUSE:
000356 005737 DT.SER: TST     @#DT.TST        ;IN END ZONE?
       177340
000362 100707         BMI     DT.TA1          ;O.K. MEANS TURN AROUND
000364 142737         BICB    #40,@#177776    ;DROP PRIORITY
       000040
       177776
000372 013746         MOV     @#V.RSAV,-(SP)  ;SAVE ALL USER REGS.
       000044
000376 004536         JSR     R5,@(SP)+
000400 012701         MOV     #DT.TST,R1      ;GET DECTAPE STATUS
       177340
000404 011146 DT.EXT: MOV     @R1,-(SP)       ;SET UP TO TELL USER
000406 012746         MOV     #DT.IRE,-(SP)
       000404
000412 032721         BIT     #14000,(R1)+    ;.... ASSUMING H-W FAILURE
       014000
000416 001405         BEQ     DT.STP          ;.... IF SEL OR ILO
000420 012716         MOV     #DT.NRE,@SP     ;DIAGNOSE TAPE FAULT DIFF.
       000402
000424 016766         MOV     DT.NAM,2(SP)    ;... AS NOT READY
       177364
       000002
000432 112711 DT.STP: MOVB    #10,@R1         ;STOP TAPE IN CASE
       000010
```

```
000436 000004           IOT                          ;GO TO DIAG PRINT
000440 004767 DT.RXT:   JSR     PC,DT.PR1            ;ON RECOVERY, SET UP RETRY
       177400
000444 013705           MOV     @#V.RRES,R5          ;RESTORE USER REGS
       000046
000450 004515           JSR     R5,@R5
000452 000002           RTI                          ;... & HOPE FOR BETTER THINGS!
              ;BLOCK NOT FOUND IN SEARCH:
000454 016746 DT.BER:   MOV     DT.BCK,-(SP)         ;GIVE BLOCK NO. AS EVIDENCE
       177610
000460 012746           MOV     #DT.BRE,-(SP)
       001416
000464 012701           MOV     #DT.CCM,R1           ;GET CONTROL ADDRESS
       177342
000470 000760           BR      DT.STP


              ;TRANSFER ERROR:
000472 032741 DT.TER:   BIT     #34000,-(R1)         ;TAPE FAILURE/OPERATOR FAULT?
       034000
000476 001342           BNE     DT.EXT               ;IF SO PRINT & WAIT RECOVERY
000500 032721           BIT     #100400,(R1)+        ;END ZONE/N.E.M?
       100400
000504 001027           BNE     DT.FER               ;IF SO TREAT AS FATAL
              ;RECOVERABLE ERRORS (TIMING OR PARITY):
000506 006327           ASL     #0                   ;RETRIED 8-9 TIMES ALRDY?
       000000
       000510 DT.RTC=.-2
000512 103352           BCC     DT.RXT               ;IF NOT TRY AGAIN ....
000514 052760           BIS     #100000,12(R0)       ;OTHERWISE SIGNAL ERROR
       100000
       000012
000522 110321           MOVB    R3,(R1)+             ;STOP TAPE IN CASE
000524 016102           MOV     1(R1),R2             ;...BUT CHK ALL WORDS DONE!
       000001
000530 001710           BEQ     DT.TXT               ;IF SO THAT'S IT!
000532 060300           ADD     R3,R0                ;GO TO WORD COUNT IN DDB
000534 162002           SUB     (R0)+,R2             ;... & USE TO DETERMINE ...
000536 000302           SWAB    R2                   ;... NO. OF BLOCKS DONE
000540 130321           BITB    R3,(R1)+             ;CHECK PRESENT TRAVEL
000542 001401           BEQ     .+4                  ;ADJUST NO. ACCORDINGLY
000544 005402           NEG     R2
000546 060267           ADD     R2,DT.BRQ            ;MODIFY SEARCH START BLOCK
       177514
000552 005067           CLR     DT.RTC               ;... & RETRY COUNT
       177732
000556 004767           JSR     PC,DT.PR2            ;GO SET UP NEW START
       177306
000562 000730           BR      DT.RXT+4             ;... & WAIT RESULTS!
              ;FATAL ERRORS - END ZONE OR NON-EXISTENT MEMORY:
000564 011046 DT.FER:   MOV     @R0,-(SP)            ;GIVE CALL AS EVIDENCE
000566 012746           MOV     #DT.FRE,-(SP)        ;PRINT DIAGNOSIS
       001415
000572 000717           BR      DT.STP
```

```
                      ;MISCELLANEOUSDEFINITIONS:
               000044 V.RSAV=44
               000046 V.RRES=46
               000100 DT.DIR=100
               177340 DT.TST=177340
               177342 DT.CCM=177342
               177346 DT.CBA=177346
               177350 DT.CDT=177350
               000402 DT.NRE=402
               000404 DT.IRE=404
               001415 DT.FRE=1415
               001416 DT.BRE=1416
               000001         .END
```

```
000000 ERRORS
```

```
DT          000000RG     DT.BCK =  000270R     DT.BER     000454R
DT.BFD      000264R      DT.BRE =  001416      DT.BRG  =  000266R
DT.CBA =  177346         DT.CCM =  177342      DT.CDT  =  177350
DT.DIR =  000100         DT.EXT    000404R     DT.FER     000564R
DT.FRE =  001415         DT.FRQ =  000302R     DT.INT     000310R
DT.IRE =  000404         DT.NAM    000014R     DT.NRE  =  000402
DT.PR1    000044R        DT.PR2    000070R     DT.RTC  =  000510R
DT.RXT    000440R        DT.SER    000356R     DT.SIF     000162R
DT.SSW =  000201R        DT.STP    000432R     DT.SXT     000256R
DT.TAC =  000212R        DT.TA1    000202R     DT.TA2     000242R
DT.TER    000472R        DT.TFR    000040R     DT.TST  =  177340
DT.TXT    000352R        PC      =%000007      R0      =%000000
R1      =%000001         R2      =%000002      R3      =%000003
R4      =%000004         R5      =%000005      SP      =%000006
V.RRES =  000046         V.RSAV =  000044      .       =  000574R
```

DEC-11-ODMTA-A-D

PDP-11

TM11/TU10 MAGTAPE DRIVER

October 1972

SUPPLEMENT TO:

PDP-11 DEVICE DRIVER PACKAGE

DEC-11-ODDPA-A-D

MONITOR VERSION VØØ8

NOTE

This document is for information purposes only and is subject to change without notice. DEC assumes no responsibility for the use or reliability of its software on equipment which is not supplied by DEC.

NEW AND CHANGED INFORMATION

This manual documents the software as of Monitor version VØ8. It has been revised to include all new and changed material since Monitor version VØ4. Such material is indicated by vertical bars in the outside margin. Whole new pages are not so marked but are dated in the lower outside corner.

pdp11
SOFTWARE
DISTRIBUTION
CENTER

CONTENTS

DRIVER FOR TM11/TU10 MAGTAPE CONTROL

1.0 INTRODUCTION

The TM11/TU10 Magtape driver provides the interface between the DOS Monitor transfer routines and the TM11 Magtape control unit. It supports the operation of both 7 and 9-track TU10 Magtape units. In addition to supporting DOS Monitor OPEN/CLOSE, READ/WRITE, and TRAN processing, this driver provides several functions to enable user control of special device features.

2.0 TAPE FORMAT

Although Magtape is not considered a file-structured device, certain structure and label processing features have been implemented to enable creation and retrieval of multiple files on a Magtape.

2.1 Files

A file is a collection of sequential records bounded by end-of-file (EOF) records or by the bottom-of-tape (BOT) marker and an end-of-file record. In nonfile-structured TRAN processing, each record of a file is 256 decimal words long except for the first record, which is the file label and which is seven words* long.

2.2 Logical End-of-Tape

In order to accomplish label searching, it is necessary to know when the last file of a tape has been passed. This is accomplished through the CLOSE request, which writes a logical end-of-tape (EOT) marker, i.e., a null file (three end-of-file records with no intervening data records).

A tape which has no files on it must be initialized by having at least one end-of-file record written on it in order to be used with OPEN/CLOSE processing.

*Six words for monitor release. VØØ4A

The last file on a tape is the one which was last opened for output. Any files which were on the tape following that file are not recoverable. New files which are added to the tape write over the old LEOT and write a new LEOT after the last record.

2.3 End-of-Tape Marker

Access is allowed beyond the end-of-tape (EOT) marker for all operations except WRITE. Attempts to write beyond the EOT marker are rejected and EOF/EOM status is set.

2.4 File Label Record

Each file created by OPEN processing has as its label (first record) a 7-word record of the following form:

```
LABEL+Ø    FILE (WORD)
LABEL+2    NAME (WORD)
LABEL+4    EXTENSION (WORD)
LABEL+6    UIC (DEFAULT TO LOGIN UIC IF NOT SPECIFIED) (WORD)
LABEL+8    PROTECT CODE (DEFAULT TO 233) IF NOT SPECIFIED (BYTE)
LABEL+9    UNUSED (BYTE)
LABEL+1Ø   DATE CREATED (WORD)
LABEL+12   UNUSED (WORD)
```

This is also the form of the user's filename block.

2.5 7/9 Track Bit Storage Patterns

The following is a short description of the bit patterns stored on magnetic tape by DEC's TM11 interface. The TM11 interfaces the 7 and 9-track TU1Ø drive to the PDP-11.

Figure 1 depicts the results of a normal write on 7-track tape. Bits 6, 7, 14 and 15 are dropped. The density may be 2ØØ, 556 or 8ØØ BPI. For this type of write operation, a "character" is six bits of an 8-bit computer byte. The output from one computer word (minus 4 bits) is stored in 2 characters.

Figure 2 illustrates the 7-track "CORE DUMP" mode of transfer. This mode is written at 8ØØ BPI only and channels "DATA5" and DATA6" are set to zero. The result is that 4 bits equal one character and 4 characters contain all the bits of one computer word (as shown).

DATA1	Bit Ø	Bit 8
DATA2	Bit 1	Bit 9
DATA3	Bit 2	Bit 1Ø
DATA4	Bit 3	Bit 11
DATA5	Bit 4	Bit 12
DATA6	Bit 5	Bit 13
LATERAL PARITY CHANNEL		

MOTION

Figure 1 7-Track Magtape PDP-11

Bit Ø = Least Significant Bit

Bits 6, 7, 14 & 15 are dropped

6 Bits = 1 character
(i.e.: Bits Ø-5 or 8-13)

The above is a graphic representation of 7-track Magtape after a normal write.

Density: 2ØØ BPI
556 BPI
8ØØ BPI

DATA1	Bit Ø	Bit 4	Bit 8	Bit 12
DATA2	Bit 1	Bit 5	Bit 9	Bit 13
DATA3	Bit 2	Bit 6	Bit 1Ø	Bit 14
DATA4	Bit 3	Bit 7	Bit 11	Bit 15
DATA5	Ø	Ø	Ø	Ø
DATA6	Ø	Ø	Ø	Ø
LATERAL PARITY CHANNEL				

MOTION

Figure 2 7-Track Magtape PDP-11

Bit Ø = Least Significant Bit

4 Bits = 1 Character
(i.e.: Bits Ø-3 or 4-7 or 8-11)

The above is a graphic representation of 7-track magtape after a "CORE DUMP" transfer. DATA5 and DATA6 channels are set = Ø for this mode.

For 9-track tape units, all 16 bits are transfered as shown in Figure 3. One computer byte (8 bits) is equal to one "character" and two characters contain one computer word. Recording density on the 9-track units is 8ØØ BPI only.

A record may be 2 to 32767_{10} words in length. The end of record is marked as follows:

1. 9-track: 3 blank "characters",
   CRC "character",
   3 blank "characters",
   LPC "character"
2. 7-track: 3 blank "characters"
   LPC "character

Finally, an EOF for 9-track is a 23_8 plus an LPC of 23_8 and for 7-track an EOF is a 17_8 plus an LPC of 17_8.

CRC - Cyclical redundency check

LPC - Longitudinal parity check

EOF - End-of-file

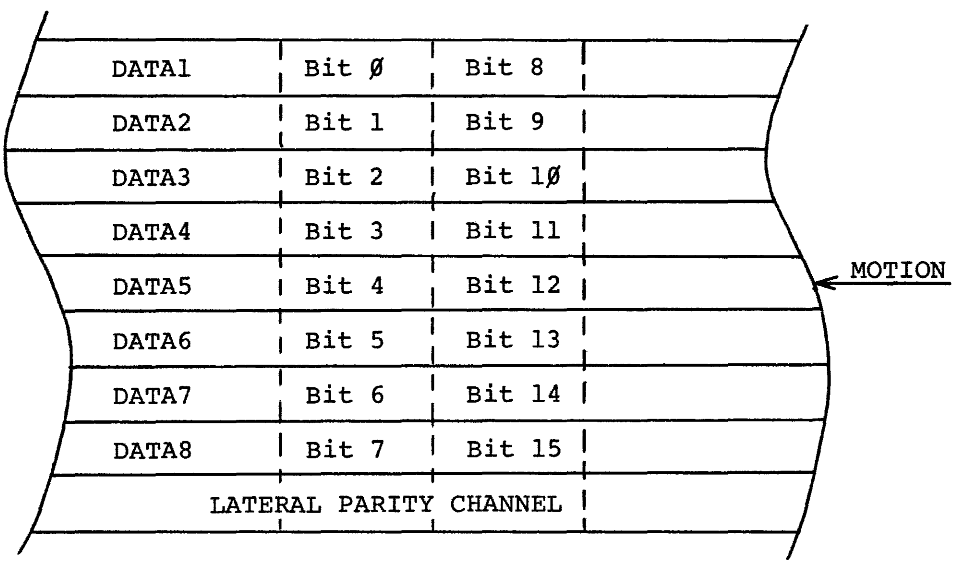


Figure 3 9-Track Magtape PDP-11

Bit Ø = Least Significant Bit

8 Bits = 1 Byte = 1 Character

The above is a graphic representation of 9-track magtape after a write operation.

Density: 8ØØ BPI Only

3.0 OPERATION

An OPEN or CLOSE request causes the Magtape to be rewound.

3.1 Standard Monitor Functions

3.1.1 OPEN

In general, an OPEN performs the following:

a) The driver rewinds the Magtape;

b) the driver checks if the device is already open and if so takes the user error exit;

c) if the OPEN is for output processing, the driver then checks that the write lock bit is off. If the write lock bit is on, the driver issues an action Monitor request to insert the file protect ring before continuing;

d) the driver then reads the first record of each file, comparing the filename, extension and UIC of the label with a merged version of the user filename block and any overriding assignment until it finds a match or until the logical end-of-tape is read.

If an error occurs while reading a file label, an action error message is printed. If the operator elects to continue processing, the label is read as though no error had occurred. When OPEN processing has been successfully completed, the device is set open and control returns to the user.

3.1.1.1 OPENI

This request requires that the file be found during file search. If LEOT is encountered, the user error exit is taken.

3.1.1.2 OPENE

If the file is found, the driver skips to the end-of-file. If the file is not found (i.e., LEOT read during file search), the file label is written over the LEOT.

3.1.1.3 OPENO

If the file is not found, the file label is written over the LEOT. If the file is found, an action diagnostic is issued. If the operator removes the current tape and readies a new one, the entire search procedure recurs. If the operator continues to operate without replacing the tape, OPEN behaves as if the driver just wrote the file label.

3.1.1.4 OPENC

Same as OPENE except that if file is found, does not skip to the end-of-file.

3.1.1.5 OPENU

This request is not allowed.

3.1.2 CLOSE

If the last operation to the device was a WRITE, CLOSE writes the logical end-of-tape and rewinds the tape. If the last operation was not a WRITE, CLOSE rewinds the tape. In either case, CLOSE clears the OPEN status.

3.1.3 READ/WRITE

These requests are buffered through the Monitor and allow all normal modes of character transmission (e.g., formatted ASCII, unformatted binary). EOF/EOM is flagged when an EOF record is read, or during output when the EOT marker is sensed.

Unlike most other devices, Magtape flags parity errors on WRITE operations.

3.1.4 BLOCK

This request is not allowed.

3.1.5 TRAN

This request allows sequential processing of records from 2 to 32767 words in length. On output all requested words are written. On input the requested number of words is read or all words in a record are read, whichever is less. Where the number of words requested is less than the number of words in a record, an error is flagged (see Section 3.2.2). Where the number of words requested is greater than the number of words in a record, a residue word count is returned. In the latter case, the Monitor may flag EOF/EOM; however, this will be erroneous unless the residue word count equals the requested word count (which case will occur only when an EOF is read).

If a record is short by an odd number of bytes, it is padded with one null character and the word count is set to (NUM BYTES READ+1)/2 before short record checking is done. Thus, the user can determine the size of a record only to the nearest rounded word.

3.2 Special Functions

These functions are provided for use in TRAN processing or outside the scope of OPEN/CLOSE processing. However, they are not restricted to these areas and care must be exercised in their use.

3.2.1 Special Function Block

The Magtape driver requires a special function block to perform the special function requests. The following is the calling sequence for Magtape special functions and the special function block format:

```
        MOV #SFBLK,-(SP)        ;ADDR of special function block
        MOV #LNKBLK,-(SP)       ;ADDR of link block
        EMT 12                  ;Special function EMT
         .
         .
         .
      SFBLK:  .BYTE    Special function code
              .BYTE    Words to follow must be 3 or larger
              .WORD    Tape unit status (returned by driver)
              .WORD    User specified count or control information
              .WORD    Residue count (returned by driver)
```

3.2.2 OFFLINE (Rewind and Unload) - function Code 1

This request causes the Magtape to be rewound to the beginning-of-tape (BOT) marker and SELECT REMOTE status to go off. If the last command to the driver for this device was a WRITE, an EOF is written before rewinding. Thus, this function could cause data to be lost if it is issued before a CLOSE during READ/WRITE processing.

3.2.3 WRITE END-OF-FILE - function Code 2

This request writes an end-of-file record on Magtape. It may cause data to be lost as described under OFFLINE.

3.2.4 REWIND - function Code 3

The REWIND request performs the same function as OFFLINE except that the SELECT REMOTE status does not go off.

3.2.5 SKIP RECORD(S) - function Code 4

Skips forward over the requested number of records (SFBLK+4) until either the SKIP count is exhausted or until an EOF record is encountered, in which case the EOF is spaced over and counted, but the operation terminates and a residue count (SFBLK+6) is returned (if any).

3.2.6 BACKSPACE RECORD(S) - function Code 5

This request skips backwards over the requested number of records until either the SKIP count is exhausted or until an EOF or the BOT marker is encountered. If an EOF is encountered it is spaced over and counted, but the operation terminates and a residue count is returned (if any). If the BOT marker is encountered, it is not skipped or counted, and a residue count is returned.

3.2.7 SET DENSITY AND PARITY - function Code 6

This request is ignored for 9-track tapes; it sets density and parity as follows for 7-track tapes:

DENSITY (SFBLK+5)	PARITY (SFBLK+4)
Ø = 2ØØ BPI	Ø = ODD
1 = 556 BPI	1 = EVEN
2 = 8ØØ BPI	
3 = 8ØØ BPI Dump Mode	

The default density and parity are 8ØØ BPI Dump Mode, ODD. In this mode, one byte from core is represented as two bytes on 7-track Magtape. Changing from this default causes one byte from core to be represented by one byte on tape with a loss of the two high order bits (6-7) of the byte.

3.2.8 TAPE UNIT STATUS - function Code 7

This request returns the current status of the tape unit in SFBLK+2 in the following form:

Bits	Content
Ø - 2	Last command was: Ø = OFFLINE 1 = READ 2 = WRITE 3 = WRITE EOF 4 = REWIND 5 = SKIP RECORD 6 = BACKSPACE RECORD
3 - 6	Unused.
7	1 = TAPE AFTER EOF (BEFORE EOF IF LAST COMMAND WAS BACKSPACE)
8	1 = TAPE AT BOT MARKER
9	1 = TAPE AFTER EOT MARKER
1Ø	1 = WRITE LOCK ON
11	PARITY: Ø = ODD 1 = EVEN (DEFAULT = ODD)
12	Ø = 9 TRACK 1 = 7 TRACK
13 - 14	DENSITY: Ø = 2ØØ BPI 1 = 556 BPI 2 = 8ØØ BPI 3 = 8ØØ BPI DUMP MODE
15	1 = LAST COMMAND CAUSED ERROR

Tape unit status is returned in SFBLK+2 for all special functions.

3.3 Error Processing

In most circumstances, the device driver attempts recovery from error conditions by retrying the operation several times, and failing to complete the operation either returns to the user with the error flag set or issues a fatal diagnostic.

3.3.1 Cyclical Redundancy/Parity Error

On input operations, the driver attempts to reread 15 times and if error persists, returns control to the user with error flag set.

On output operations, the driver attempts to rewrite 15 times with an extended record gap and if error persists issues an action diagnostic before returning to the user with the error flag set.

On other operations, the condition is not relevant and is ignored.

3.3.2 Record Length Error

On input the driver returns to the user with the error flag (bit 15 of TRNBLK+6) set (see DOS Programmers Manual). The condition is not possible on write operations.

If the number of words requested in an input TRAN is less than the physical record size on magtape, bit 15 of the Function/Status Word is turned on, the number of words requested are transferred, and the driver returns normally. The remaining information in the record is "lost" in the sense that it can only be read by back-spacing and re-TRANing with a larger request. The next TRAN will get the next physical record.

Record length errors can be differentiated from other (e.g., parity) errors only by inspecting the hardware registers.

3.3.3 Bad Tape Error

This error is treated as described in Section 3.3.1.

3.3.4 BUS Grant Late

Driver checks status word of device* to detect BUS Grant Late errors and issues a fatal diagnostic.

3.3.5 Non-existent Memory

The driver issues a fatal diagnostic.

3.3.6 Illegal Command

The driver issues a fatal diagnostic.

3.3.7 OFFLINE

Whenever the driver detects a device not-ready condition, it issues an action diagnostic before processing the command.

3.3.8 WRITE LOCK

If the last command given is a WRITE or WRITE EOF and the WRITE LOCK is on, the driver issues an action diagnostic before processing the command.

3.4 Diagnostics Issued

A002 - DEVICE NOT READY OR FILE PROTECT RING NEEDED (see 3.1.1, 3.3.7, 3.3.8).

A006 - UNRECOVERABLE WRITE ERROR AFTER 15 RETRIES (see 3.3.1).

A007 - LABEL FOUND DURING OPENO (see 3.1.1.3).

A010 - UNRECOVERABLE READ ERROR AFTER 15 RETRIES DURING OPEN (see 3.1.1)

F012 - NO USER ERROR RETURN SPECIFIED IN FILE NAME BLOCK DURING OPEN.

F032 - FATAL ERROR ON MAG TAPE (see 3.3, 3.3.4-3.3.6).

F033 - BAD SPECIAL FUNCTION BLOCK FORMAT (see 3.2.1).

---

*Release V004A does not check status word but does 15 attempts and gives fatal error.

4.0 CHARACTER CONVERSIONS BY THE DEVICE DRIVER

It has been suggested that it would be desirable to have the device driver convert data from ASCII to other coding schemes or vice versa. Although this presents no great implementation problem, there are two reasons why it is not being done:

1) The tables necessary to perform these conversions would be large.

2) The user can maintain his own tables and do conversions more flexibly than the driver.

4.1 Prototype Conversion Routine

The following is an example of a conversion routine which a user might use to do coding scheme conversions:

```
;
;CONVERT FROM CODING SCHEME A TO CODING SCHEME B
;
CONVAB: MOV     #RECADR,R1      ;ADDR OF BYTES IN SCHEME A
        MOV     RECLEN,R2       ;NUMBER OF BYTES TO CONVERT
;
NEXT;   CLR     RØ              ;GET BYTE IN SCHEME A
        BISB    @R1,RØ
        ADD     #CONAB,RØ       ;ADD ADDR OF A TO B TABLE
        MOVB    @RØ,(R1)+       ;REPLACE SCHEME A BYTE WITH
                                ;SCHEME B BYTE
        DEC     R2              ;DECREMENT BYTE COUNT
        BNE     NEXT            ;BRANCH IF NOT FINISHED
        .
        .
        .
```

NOTE

Conversion table CONAB contains bytes in coding scheme B ordered such that the numeric value of A byte in coding scheme A is the index into CONAB of the corresponding byte in coding scheme B.

5.0 PROCESSING NON-PDP-11 CREATED FILES

This feature is not yet available.

6.0 MAGTAPE DRIVER LISTING VØØ6A

```
DV.MT    MACRO VØØ4-14 13-SEP-72 Ø3:ØØ PAGE 1


                 ;COPYRIGHT:-    DIGITAL EQUIPMENT CORP.,MAYNARD,MASS.
                 ;               1971,1972
                 ;
                 ;VERSION NO:-   VØØ6A.ØØØ
                 ;
                 ;
                         .IFDF   DVRRWD
                         .TITLE  DV.QT
                         .GLOBL  QT
                         .ENDC
                         .IFNDF  DVRRWD
                         .TITLE  DV.MT
                         .GLOBL MT
                         .ENDC
                 ;
         ØØØØØØ'         .CSECT
         ØØØØØØ  RØ=%Ø   ;DDB PTR
         ØØØØØ1  R1=%1   ;LCMMD PTR
         ØØØØØ2  R2=%2   ;CMMD REG
         ØØØØØ3  R3=%3   ;SP FUNC BLOCK PTR
         ØØØØØ4  R4=%4   ;
         ØØØØØ5  R5=%5   ;SCRATCH
         ØØØØØ6  SP=%6
         ØØØØØ7  PC=%7
                 ;
                 ;
                 ;
         17252Ø  MTS=17252Ø        ;TM11 STATUS
         172522  MTC=172522        ;TM11 COMMAND
         172524  MTBRC=172524      ;TM11 BYTE/RECORD COUNTER
         172526  MTCMA=172526      ;TM11 CORE MEMORY ADDRESS
         17253Ø  MTD=17253Ø        ;TM11 DATA BUFFER
         172532  MTRD=172532       ;TM11 READ LINES
                 ;
         177776  PS=177776         ;PROCESSOR STATUS
                 ;
                 ;
                 ;MTS BITS
                 ;
         1ØØØØØ  ILC=1ØØØØØ
         Ø4ØØØØ  EOF=4ØØØØ
         Ø2ØØØØ  CRE=2ØØØØ
         Ø1ØØØØ  PAE=1ØØØØ
         ØØ4ØØØ  BGL=4ØØØ
         ØØ2ØØØ  EOT=2ØØØ
         ØØ1ØØØ  RLE=1ØØØ
         ØØØ4ØØ  BTE=4ØØ
         ØØØ2ØØ  NXM=2ØØ
         ØØØ1ØØ  SELR=1ØØ
         ØØØØ4Ø  BOT=4Ø
         ØØØØ2Ø  CH79=2Ø
         ØØØØ1Ø  SDWN=1Ø
         ØØØØØ4  WRL=4
         ØØØØØ2  RWS=2
         ØØØØØ1  TUR=1
         ØØØ14Ø  DENB=14Ø
         ØØØØ1Ø  PARB=1Ø
```

```
                     ;
                     ;MTC BITS
                     ;
              100000 ERR=100000
              060000 DEN=60000
              010000 PCWR=10000
              004000 PAR=4000
              003400 UNIT=3400
              000200 CUR=200
              000100 INT=100
              000060 ADEX=60
              000016 CMMD=16
              000001 GOB=1
                     ;
                     ;MTRD BIT
                     ;
              010000 GAPSDN=10000
                     ;
                     ;COMMANDS
                     ;
              000000 RWU=0
              000001 READ=1
              000002 WRITE=2
              000003 EOFM=3
              000004 RWD=4
              000005 SKPR=5
              000006 BSPR=6
                     ;
                     ;
                     ;
                     ;THIS IS THE DEVICE DRIVER FOR THE TM11/TU10
                     ;
                                .IFDF   DVRRWD
                     QT:        .ENDC
       00000  000000 MT:        .WORD   0                       ;BUSY INDICATOR
       00002     177            .BYTE   177                     ;ALL GENERAL STRUCTURE EXCEPT OP
       00003     040            .BYTE   40                      ;SPECIAL STRUCTURE = MAG TAPE
       00004     020            .BYTE   20                      ;BUFFER SIZE = 512 BYTES
       00005     360            .BYTE   INTJ-MT                 ;INTERRUPT HANDLER
       00006     240            .BYTE   240                     ;PRIO FOR INTERRUPT SERVICE
       00007     000            .BYTE   0                       ;NO OPEN ENTRY
       00010     374            .BYTE   TRANS-MT                ;TRANSFER ENTRY
       0011      370            .BYTE   CLOSJ-MT                        ;CLOSE ENTRY
       0012      364            .BYTE   SPECJ-MT                        ;SPECIAL FUNCTION ENTRY
       0013      000            .BYTE   0
                                .IFDF   DVRRWD
                     MT.NAM:    .RAD50  /CT/
                                .ENDC
                                .IFNDF  DVRRWD
       0014   052140 MT.NAM:    .RAD50  /MT /
                                .ENDC
                     ;
       0016      000 OPNFLG:    .BYTE   0,0,0,0,0,0,0,0 ;SET BY OPEN ROUTINE,CLEARED BY
       0017      000
       0020      000
       0021      000
       0022      000
```

```
     0023      000
     0024      000
     0025      000
     0026      377 LCMMD:  .BYTE   -1,-1,-1,-1,-1,-1,-1,-1,  ;1 BYTE FOR EACH DEVIC
     0027      377
     0030      377
     0031      377
     0032      377
     0033      377
     0034      377
     0035      377
     0036      000
     0037      000         .BYTE   0,0,0,0,0,0,0,0 ;DEN/PAR FOR EACH DEVICE
     0040      000
     0041      000
     0042      000
     0043      000
     0044      000
     0045      000
     0046      000
     0047      000 INTENB: .BYTE   0
     0050   000000 INTRET: .WORD   0                 ;ADDR FOR RET FROM INT HANDLER
     0052      000 TCMMD:  .BYTE   0                 ;LAST CMMD SAVE LOCN FOR ERR REC
     0053      000 ERRSW:  .BYTE   0                 ;SET BY ERR RECOVERY IF NOT RECO
     0054   000000 RETRY:  .WORD   0                 ;RETRY COUNT FOR ERROR RECOVERY
     0056   177761 TRYCNT: .WORD   -15.              ;INITIAL RETRY COUNT
     0060   000000 LASTAT: .WORD   0                 ;ADDR IN LCMMD VECTOR FOR INT HA
     0062   000000 CMA:    .WORD   0
     0064   000000 BRC:    .WORD   0
                   ;
     0066   016001 INIT:   MOV     12(R0),R1         ;GET UNIT NUM
            000012
     0072   042701         BIC     #174377,R1                ;CLR EXTRA BITS
            174377
     0076   010137         MOV     R1,@#MTC
            172522
     0102   000301         SWAB    R1
     0104   060701         ADD     PC,R1
     0106   062701         ADD     #LCMMD-.,R1       ;ADDR IN LAST CMMD VECTOR
            177720
     0112   012604         MOV     (SP)+,R4          ;SAVE RETURN ADDR
     0114   012605         MOV     (SP)+,R5          ;SAVE ORIGINAL PC
     0116   013746         MOV     @#PS,-(SP)        ;SIMULATE INT CALL
            177776
     0122   010546         MOV     R5,-(SP)
     0124   013746         MOV     @#44,-(SP)        ;SAVE REGS
            000044
     0130   004536         JSR     R5,@(SP)+
     0132   010446         MOV     R4,-(SP)          ;RESET CALLING PC
     0134   010167         MOV     R1,LASTAT         ;SAVE FOR INT HANDLER
            177720
     0140   004767         JSR     PC,READY          ;CHECK IF UNIT READY
            000162
     0144   105711         TSTB    (R1)              ;CHECK IF DEVICE INITIALIZED
     0146   100004         BPL     INITX             ;BRANCH IF IS
     0150   112761         MOVB    #DENB,10(R1)      ;SET DEFAULT DEN=800,PAR=ODD
            000140
```

```
          000010
0156 105011               CLRB    (R1)            ;CLEAR NON INIT STATE
0160 000207  INITX:       RTS     PC
                  ;
0162 105067  STMCOM:      CLRB    ERRSW           ;CLEAR ERROR SWITCH
          177665
0166 105767               TSTB    INTENB          ;BRANCH IF INT RET
          177655
0172 001005               BNE     MTEXIT
0174 105760               TSTB    -3(R0)          ;BRANCH IF CALLED FROM QUEUE
          177775
0200 001002               BNE     MTEXIT
0202 062706               ADD     #20,SP          ;REMOVE PC,PS AND REGS
          000020
0206 105067  MTEXIT:      CLRB    INTENB
          177635
0212 000170               JMP     @14(R0)         ;COMPLETION EXIT
          000014
                  ;
                  ;
                  ;
                  ;
0216 012667  GO:          MOV     (SP)+,INTRET    ;SAVE INT RETURN ADDR
          177626
0222 016737  GOA:         MOV     CMA,@#MTCMA
          177634
          172526
0230 016737               MOV     BRC,@#MTBRC
          177630
          172524
0236 121127               CMPB    (R1),#WRITE     ;CHECK IF THIS IS A WRITE
          000002
0242 001007               BNE     GO2             ;BRANCH IF NOT
0244 032737  GO1:         BIT     #WRL,@#MTS      ;CHECK IF WRITE LOCK ON
          000004
          172520
0252 001403               BEQ     GO2             ;BRANCH IF NOT
0254 004767               JSR     PC,READY1       ;ISSUE ACTION MSG
          000034
0260 000771               BR      GO1             ;GO TEST IF LOCK STILL ON
0262 004767  GO2:         JSR     PC,READY        ;CHECK IF DEVICE READY
          000240
0266 156137               BISB    10(R1),@#MTC+1  ;SET DEN AND PAR
          000010
          172523
0274 052702               BIS     #INT+GOB,R2             ;SET INT ENB AND GO BITS
          000101
0300 110237               MOVB    R2,@#MTC        ;ISSUE INSTRUCTION
          172522
0304 013746  GO3:         MOV     @#46,-(SP)      ;RESTORE REGS
          000046
0310 004536               JSR     R5,@(SP)+
0312 000002               RTI                     ;RETURN TO INTERRUPT
                  ;
                  ;
0314 016746  READY1:      MOV     MT.NAM,-(SP)    ;ISSUE ACTION DIAG -
          177474
```

```
Ø32Ø Ø12746          MOV     #422,-(SP)        ;DEVICE NOT READY
     ØØØ422
Ø324 ØØØØØ4          IOT
Ø326 Ø32737 READY:   BIT     #SELR,@#MTS       ;TEST IF DEVICE READY
     ØØØ1ØØ
     17252Ø
Ø334 ØØ1767          BEQ     READY1            ;BRANCH IF NOT
Ø336 Ø32737          BIT     #GAPSDN,@#MTRD
     Ø1ØØØØ
     172532
Ø344 ØØ1ØØ4          BNE     READYX
Ø346 Ø32737          BIT     #TUR+RWS+SDWN,@#MTS
     ØØØ213
     17252Ø
Ø354 ØØ1757          BEQ     READY1
Ø356 ØØØ2Ø7 READYX:  RTS     PC                ;RETURN TO CALLER
            ;
Ø36Ø ØØØ167 INTJ:    JMP     INTH
     ØØØ734
Ø364 ØØØ167 SPECJ:   JMP     SPEC
     ØØØ222
Ø37Ø ØØØ167 CLOSJ:   JMP     CLOSE
     ØØØ67Ø
            ;
            ;
Ø374 ØØ4767 TRANS:   JSR     PC,INIT           ;INIT CHECK DEVICE
     177466
Ø4ØØ Ø16Ø67          MOV     6(RØ),CMA         ;SET BUFF ADDR
     ØØØØØ6
     177454
Ø4Ø6 Ø16Ø67          MOV     1Ø(RØ),BRC        ;SET WORD COUNT
     ØØØØ1Ø
     17745Ø
Ø414 ØØ6367          ASL     BRC               ;CVT TO BYTE COUNT
     177444
Ø42Ø Ø16ØØ2          MOV     12(RØ),R2
     ØØØØ12
Ø424 Ø427Ø2          BIC     #177713,R2        ;CLR ALL BUT READ AND ADDR EXT B
     177713
Ø43Ø Ø327Ø2          BIT     #4,R2             ;CHECK INPUT/OUTPUT
     ØØØØØ4
Ø434 ØØ14Ø5          BEQ     TRANØ             ;BRANCH IF OUTPUT
Ø436 112711          MOVB    #READ,(R1)        ;SET LAST CMMD=READ
     ØØØØØ1
Ø442 1627Ø2          SUB     #2,R2             ;SET UP READ CMMD
     ØØØØØ2
Ø446 ØØØ414          BR      TRAN2
Ø45Ø Ø32737 TRANØ:   BIT     #EOT,@#MTS        ;CHECK IF AT EOT
     ØØ2ØØØ
     17252Ø
Ø456 ØØ14Ø4          BEQ     TRAN1             ;BRANCH IF NOT
Ø46Ø Ø16Ø6Ø          MOV     1Ø(RØ),16(RØ)     ;RETURN WORD COUNT
     ØØØØ1Ø
     ØØØØ16
Ø466 ØØØ635 TRAN7:   BR      STMCOM            ;REJECT CMMD - EXIT
Ø47Ø 112711 TRAN1:   MOVB    #WRITE,(R1)       ;SET LAST CMMD = WRITE
     ØØØØØ2
```

```
0474 062702          ADD     #4,R2             ;SET UP WRITE CMND
     000004
0500 004767 TRAN2:   JSR     PC,GO             ;GO INITIATE I/O
     177512
0504 013702          MOV     @#MTBRC,R2
     172524
0510 121127          CMPB    (R1),#READ        ;COME HERE AFTER INT ERR CHK
     000001
0514 001006          BNE     TRAN6             ;BRANCH IF NOT READ
0516 032737          BIT     #EOF,@#MTS        ;IF EOF, SET INIT CNT IN RESIDUE
     040000
     172520
0524 001402          BEQ     TRAN6
0526 016702          MOV     BRC,R2
     177332
0532 006202 TRAN6:   ASR     R2        ;CHECK IF ODD BYTES SHORT REC
0534 103004          BCC     TRAN3             ;BRANCH IF NOT
0536 013705          MOV     @#MTCMA,R5        ;PUT NULL IN NEXT BUFF POS
     172526
0542 105015          CLRB    (R5)
0544 005202          INC     R2                ;ROUND UP WORD COUNT
0546 001402 TRAN3:   BEQ     .+6               ;BRANCH IF NO RESIDUE
0550 052702          BIS     #100000,R2        ;INSURE NEG WORD COUNT
     100000
0554 032737          BIT     #RLE,@#MTS        ;CHECK IF LENGTH ERROR
     001000
     172520
0562 001402          BEQ     TRAN4             ;BRANCH IF NOT
0564 105267          INCB    ERRSW
     177263
0570 010260 TRAN4:   MOV     R2,16(R0)         ;RETURN RESIDUE WORD COUNT
     000016
0574 105767 TRANX:   TSTB    ERRSW             ;BRANCH IF NO ERRORS
     177253
0600 001403          BEQ     TRAN5
0602 052760          BIS     #100000,12(R0)    ;SET ERR BIT
     100000
     000012
0610 000726 TRAN5:   BR      TRAN7             ;TAKE DONE EXIT
            ;
0612 004767 SPEC:    JSR     PC,INIT           ;INIT CHECK DEVICE
     177250
0616 016003          MOV     2(R0),R3          ;GET FUNC BLOCK ADDR
     000002
0622 111305          MOVB    (R3),R5 ;GET FUNC BYTE
0624 162705          SUB     #SPFST,R5
     000001
0630 100767          BMI     TRAN5             ;BRANCH OUT IF NOT
0632 022705          CMP     #SPLST,R5         ;SUPPORTED FUNCTION
     000006
0636 103764          BLO     TRAN5
0640 122763          CMPB    #3,1(R3)          ;CHECK IF VALID FUNC BLOCK
     000003
     000001
0646 101014          BHI     ABORT             ;ABORT IF NOT
0650 006305          ASL     R5
0652 060705          ADD     PC,R5
```

```
0654 062705          ADD     #SPECT-.,R5
     000006
0660 000115          JMP     @R5                ;GO TO PROPER SP FUNC ROUTINE
            ;
            ;
     000001 SPFST=1
     000006 SPLST=6
            ;
0662 000412 SPECT:   BR      OFFLIN
0664 000452          BR      WEOF
0666 000417          BR      RWND
0670 000476          BR      SKP
0672 000502          BR      BSP
0674 000453          BR      PARDEN
0676 000530          BR      TUSTAT
            ;
            ;
0700 010346 ABORT:   MOV     R3,-(SP)           ;ISSUE SP FUNC BLOCK BAD ABORT
0702 012746          MOV     #1433,-(SP)        ;WITH ADDR OF SP FUNC BLOCK
     001433
0706 000004          IOT
            ;
0710 004767 OFFLIN:  JSR     PC,EOFCK           ;WRITE EOF IF NECESSARY
     000052
0714 105011          CLRB    (R1)               ;SET LAST CMMD=OFFLINE
0716 112737          MOVB    #1,@#MTC           ;ISSUE DISABLED RWU
     000001
     172522
0724 000515          BR      TUSTAT             ;GET STAT AND EXIT
0726 004767 RWND:    JSR     PC,EOFCK           ;ISSUE WRITE EOF IF NECESSARY
     000034
0732 004767          JSR     PC,RWNDC           ;ISSUE DISABLED RWD
     000002
0736 000512          BR      TUSTAT             ;GET STATUS AND EXIT
0740 112711 RWNDC:   MOVB    #RWD,(R1)          ;SET LAST CMMD=RWD
     000004
                     .IFDF   DVRRWD
                     BR      RWNDX    ;BYPASS REWIND
                     .ENDC
0744 032737          BIT     #BOT+RWS,@#MTS  ;IS IT REWOUND
     000042
     172520
0752 001004          BNE     RWNDX    ;YES, BRANCH
0754 012702          MOV     #16,R2             ;ISSUE RWD
     000016
0760 000167          JMP     GO
     177232
0764 000207 RWNDX:   RTS     PC
            ;
0766 121127 EOFCK:   CMPB    (R1),#WRITE        ;IF LAST CMMD WAS WRITE
     000002
0772 001401          BEQ     EOFCK1             ;BRANCH
0774 000207          RTS     PC                 ;ELSE RETURN
0776 012702 EOFCK1:  MOV     #6,R2              ;SET CMMD=WRITE EOF
     000006
1002 012667          MOV     (SP)+,INTRET       ;SET INT RET ADDR
     177042
```

```
1006 000167          JMP     GO1             ;GO EXECUTE
     177232
                ;
1012 112711 WEOF:    MOVB    #EOFM,(R1)      ;SET LAST CMMD
     000003
1016 004767          JSR     PC,EOFCK1       ;GO EXECUTE WRITE EOF
     177754
1022 000456          BR      TUSTAT  ;AT INT RET, GET STAT AND EXIT
                ;
                ;
1024 032737 PARDEN:  BIT     #CH79,@#MTS     ;IF 9 TRACK TAPE
     000020
     172520
1032 001452          BEQ     TUSTAT          ;BRANCH (IGNORE NEW SETTINGS)
1034 016305          MOV     4(R3),R5        ;GET NEW DEN/PAR
     000004
1040 042705          BIC     #176376,R5      ;CLR EXTRA BITS
     176376
1044 106205          ASRB    R5              ;SET INTO PROPER POSITION
1046 106005          RORB    R5
1050 106005          RORB    R5
1052 006005          ROR     R5
1054 006005          ROR     R5
1056 006005          ROR     R5
1060 110561          MOVB    R5,10(R1)       ;SET NEW DEN/PAR
     000010
1064 000435          BR      TUSTAT
                ;
1066 112711 SKP:     MOVB    #SKPR,(R1)      ;SET LAST CMMD=SKP
     000005
1072 012702          MOV     #8.,R2          ;SET CMMD
     000010
1076 000414          BR      SKPBSP          ;GO SET COUNT AND EXEC
1100 112711 BSP:     MOVB    #BSPR,(R1)      ;SET LAST CMMD=BSP
     000006
1104 012702          MOV     #10.,R2         ;SET CMMD
     000012
1110 032737          BIT     #BOT+RWS,@#MTS  ;TEST IF AT BOT
     000042
     172520
1116 001404          BEQ     SKPBSP          ;BRANCH IF NOT
1120 016363          MOV     4(R3),6(R3)     ;REJECT CMMD RETURN REC COUNT
     000004
     000006
1126 000414          BR      TUSTAT          ;SET STATUS AND EXIT
1130 016367 SKPBSP:  MOV     4(R3),BRC       ;SET RECORD COUNT
     000004
     176726
1136 005467          NEG     BRC
     176722
1142 004767          JSR     PC,GO           ;GO EXECUTE CMMD
     177050
1146 013763          MOV     @#MTBRC,6(R3)   ;SET RESIDUE REC COUNT
     172524
     000006
1154 005463          NEG     6(R3)
     000006
```

```
                        ;
     1160 116102 TUSTAT: MOVB    10(R1),R2          ;GET DEN/PAR
          000010
     1164 111101         MOVB    (R1),R1 ;GET LAST CMMD
     1166 000302         SWAB    R2
     1170 050201         BIS     R2,R1              ;SET DEN AND PAR
     1172 013702         MOV     @#MTS,R2
          172520
     1176 032702         BIT     #CRE+PAE+RLE,R2 ;TEST STATUS FOR ERROR
          031000
     1202 001402         BEQ     STAT1              ;BRANCH IF NONE
     1204 052701         BIS     #100000,R1         ;SET ERR BIT
          100000
     1210 032702 STAT1:  BIT     #EOT,R2            ;TEST IF EOT
          002000
     1214 001402         BEQ     STAT2              ;BRANCH IF NOT
     1216 052701         BIS     #1000,R1           ;SET EOT BIT
          001000
     1222 032702 STAT2:  BIT     #BOT+RWS,R2        ;TEST IF AT BOT
          000042
     1226 001402         BEQ     STAT3              ;BRANCH IF NOT
     1230 052701         BIS     #400,R1            ;SET BOT BIT
          000400
     1234 032702 STAT3:  BIT     #EOF,R2            ;TEST IF EOF
          040000
     1240 001402         BEQ     STAT4              ;BRANCH IF NOT
     1242 052701         BIS     #200,R1            ;SET EOF BIT
          000200
     1246 042702 STAT4:  BIC     #177753,R2         ;CLEAR ALL BUT WRL AND 79CH BITS
          177753
     1252 000302         SWAB    R2
     1254 050201         BIS     R2,R1              ;SET WRL AND 7,9 TRACK
     1256 010163         MOV     R1,2(R3)           ;RETURN STATUS
          000002
     1262 000414         BR      COMJ               ;EXIT
                        ;
                        ;
                        ;
     1264 004767 CLOSE:  JSR     PC,INIT            ;INIT CHECK ON DEVICE
          176576
     1270 105061         CLRB    -10(R1)            ;CLEAR OPEN FLAG
          177770
     1274 004767         JSR     PC,EOFCK           ;IF LAST CMMD WAS WRITE, WRITE 3
          177466
     1300 004767         JSR     PC,EOFCK
          177462
     1304 004767         JSR     PC,EOFCK
          177456
                         .IFDF   DVRRWD
                        ;
                        ;        ISSUE 2 BSPS INSTEAD OF RWD IF OUTPUT
                        ;        OR SKIP TO END OF FILE IF INPUT
                        ;
                         CMPB    (R1),#WRITE        ;IF LAST CMMD WAS WRITE
                         BNE     CLOSE1
                         MOV     #10.,R2            ;ISSUE 2 BSPS
                         MOV     #-1,BRC
```

```
                         MOVB    #BSPR,(R1)      ;THIS IS SOLELY FOR INT ERR CHK
                         JSR     PC,GO
                         MOV     #10.,R2
                         BR      CLOSE2
                 ;
                 CLOSE1: BIT     #EOF,@#MTS      ;SKIP TO EOF UNLESS ALREADY THER
                         BNE     CLOSE3
                         MOV     #8.,R2
                         CLR     BRC
                         MOV     #SKPR,(R1)
                 CLOSE2: JSR     PC,GO
                         .ENDC
1310 004767      CLOSE3: JSR     PC,RWNDC        ;ISSUE DISABLED RWD
     177424
1314 000167      COMJ:   JMP     STMCOM
     176642
                 ;
                 ;
                 ;
                 ;
1320 013746      INTH:   MOV     @#44,-(SP)      ;SAVE REGS
     000044
1324 004536              JSR     R5,@(SP)+
1326 016700              MOV     MT,R0           ;GET CCR ADDR
     176446
1332 016701              MOV     LASTAT,R1       ;GET LCMMD VECTOR ADDR
     176522
1336 016003              MOV     2(R0),R3        ;GET SP FNC BLOCK ADDR
     000002
1342 012705              MOV     #MTC,R5         ;ADDR OF CMMD REG
     172522
1346 042715              BIC     #100,(R5)       ;DISABLE DEVICE INTERRUPT
     000100
1352 013702              MOV     @#MTS,R2        ;GET STATUS OF DEVICE
     172520
1356 105267              INCB    INTENB          ;SET INT FLAG
     176465
1362 032702              BIT     #ILC+NXM,R2     ;CHECK ILLEGAL CMMD, NONEXIST CO
     100200
1366 001404              BEQ     INT1            ;BRANCH IF NOT
1370 010246      INTF:   MOV     R2,-(SP)        ;DISPLAY STATUS AND DIAGNOSE
1372 012746              MOV     #1432,-(SP)     ;FATAL ERROR-MAG TAPE
     001432
1376 000004              IOT                     ;ABORT
                 ;
1400 105767      INT1:   TSTB    TCMMD           ;CHECK IF THIS WAS A RETRY
     176446
1404 001433              BEQ     INT3            ;BRANCH IF NOT
1406 100007              BPL     INT2            ;BRANCH IF WAS NOT RSP OF RETRY
1410 005002              CLR     R2
1412 156702              BISB    TCMMD,R2        ;GET CMMD
     176434
1416 105467              NEGB    TCMMD           ;SET-NOT RSP
     176430
1422 000167              JMP     GOA             ;GO TRY AGAIN
     176574
                 ;
```

```
398 1426 Ø327Ø2 INT2:    BIT     #BGL+BTE+CRE+PAE,R2;TEST IF ERROR THIS TIME
         Ø344ØØ
399 1432 ØØ1463          BEQ     INT7              ;BRANCH IF NOT
4ØØ 1434 ØØ5267          INC     RETRY
         176414
4Ø1 1440 ØØ1Ø36          BNE     INT6              ;BRANCH IF TO TRY AGAIN
4Ø2 1442 Ø327Ø2          BIT     #BGL,R2           ;IF ERR=BUS GRANT LATE
         ØØ4ØØØ
4Ø3 1446 ØØ135Ø          BNE     INTF              ;IS FATAL
4Ø4 145Ø 1Ø5267          INCB    ERRSW             ;SET ERROR FLAG
         176377
4Ø5 1454 Ø32715          BIT     #4,(R5) ;IF WRITE OR WEOF, ISSUE ACTION DIAG
         ØØØØØ4
4Ø6 146Ø ØØ145Ø          BEQ     INT7
4Ø7 1462 Ø1Ø246          MOV     R2,-(SP)
4Ø8 1464 Ø12746          MOV     #412,-(SP)
         ØØØ412
4Ø9 147Ø ØØØØØ4          IOT
41Ø 1472 ØØØ443          BR      INT7              ;GO TO SPECIFIC ROUTINE
411                      ;
412 1474 Ø327Ø2 INT3:    BIT     #BGL+BTE+PAE+CRE,R2;CHECK IF ERROR
         Ø344ØØ
413 15ØØ ØØ144Ø          BEQ     INT7              ;BRANCH IF NOT
414 15Ø2 122711          CMPB    #SKPR,(R1)        ;BRANCH IF SKIP OR BSP
         ØØØØØ5
415 15Ø6 ØØ1424          BEQ     INT4
416 151Ø 122711          CMPB    #BSPR,(R1)
         ØØØØØ6
417 1514 ØØ1421          BEQ     INT4
418 1516 Ø16767          MOV     TRYCNT,RETRY      ;SET RETRY COUNT
         176334
         17633Ø
419 1524 Ø32715          BIT     #12,(R5)          ;IF CMMD IS WRITE
         ØØØØ12
42Ø 153Ø ØØ1ØØ2          BNE     INT6
421 1532 Ø52715          BIS     #1Ø,(R5)          ;TRY WRITE WITH LONG GAP
         ØØØØ1Ø
422 1536 111567 INT6:    MOVB    (R5),TCMMD        ;SAVE CMMD
         17631Ø
423 1542 1127Ø2          MOVB    #1Ø.,R2           ;SET UP BSP
         ØØØØ12
424 1546 Ø12737          MOV     #-1,@#MTBRC       ;COUNT OF 1
         177777
         172524
425 1554 ØØØ167          JMP     GO2               ;GO EXECUTE
         1765Ø2
426                      ;
427 156Ø Ø327Ø2 INT4:    BIT     #EOF+BOT,R2       ;IF EOF OR BOT
         Ø4ØØ4Ø
428 1564 ØØ1ØØ6          BNE     INT7              ;SKIP OR BSP IS DONE
429 1566 ØØ5737          TST     @#MTBRC           ;IF COUNT EXHAUSTED
         172524
43Ø 1572 ØØ14Ø3          BEQ     INT7              ;IS DONE
431 1574 1115Ø2          MOVB    (R5),R2 ;ELSE SET UP CMMD
432 1576 ØØØ167          JMP     GO2               ;CONTINUE SKIP OR BSP
         17646Ø
433                      ;
```

```
434 1602 105067 INT7:   CLRB    TCMMD           ;CLEAR RETRY INDICATORS
         176244
435 1606 105067         CLRB    RETRY
         176242
436 1612 016707         MOV     INTRET,PC       ;GO TO SPECIFIC ROUTINE
         176232
437                ;
438                ;
439      000001'        .END
```


SYMBOL TABLE

```
ABORT    ØØØ7ØØR          ADEX   = ØØØØ6Ø          BGL    = ØØ4ØØØ
BOT    = ØØØØ4Ø           BPC      ØØØØ64R         BSP      ØØ11ØØR
BSPR   = ØØØØØ6           BTE    = ØØØ4ØØ          CH79   = ØØØØ2Ø
CLOSE    ØØ1264R          CLOSE3   ØØ131ØR         CLOSJ    ØØØ37ØR
CMA      ØØØØ62R          CMMD   = ØØØØ16          COMJ     ØØ1314R
CRE    = Ø2ØØØØ           CUR    = ØØØ2ØØ          DEN    = Ø6ØØØØ
DENB   = ØØØ14Ø           EOF    = Ø4ØØØØ          EOFCK    ØØØ766R
EOFCK1   ØØØ776R          EOFM   = ØØØØØ3          EOT    = ØØ2ØØØ
ERR    = 1ØØØØØ           ERRSW    ØØØØ53R         GAPSDN = Ø1ØØØØ
GO       ØØØ216R          GOA      ØØØ222R         GOB    = ØØØØØ1
GO1      ØØØ244R          GO2      ØØØ262R         GO3      ØØØ3Ø4R
ILC    = 1ØØØØØ           INIT     ØØØ266R         INITX    ØØ16ØØR
INT    = ØØØ1ØØ           INTENB   ØØØØ47R         INTE     ØØ137ØR
INTH     ØØ132ØR          INTJ     ØØØ362R         INTRET   ØØØØ5ØR
INT1     ØØ14ØØR          INT2     ØØ1426R         INT3     ØØ1474R
INT4     ØØ156ØR          INT6     ØØ1536R         INT7     ØØ16Ø2R
LASTAT   ØØØØ6ØR          LCMMD    ØØØØ26R         MT       ØØØØØØRG
MTBRC  = 172524           MTC    = 172522          MTCMA  = 172526
MTD    = 17253Ø           MTEXIT   ØØØ226R         MTRD   = 172532
MTS    = 17252Ø           MT.NAM   ØØØØ14R         NXM    = ØØØ2ØØ
OFFLIN   ØØØ71ØR          OPNFLG   ØØØØ16R         PAE    = Ø1ØØØØ
PAR    = ØØ4ØØØ           PARB   = ØØØØ1Ø          PARDEN   ØØ1Ø24R
PC     =%ØØØØØ7           POWR   = Ø1ØØØØ          PS     = 177776
READ   = ØØØØØ1           READY    ØØØ326R         READYX   ØØØ356R
READY1   ØØØ314R          RETRY    ØØØØ54R         RLE    = ØØ1ØØØ
RWD    = ØØØØØ4           RWND     ØØØ726R         RWNDC    ØØØ74ØR
RWNDX    ØØØ764R          RWS    = ØØØØØ2          RWU    = ØØØØØØ
RØ     =%ØØØØØØ           R1     =%ØØØØØ1          R2     =%ØØØØØ2
R3     =%ØØØØØ3           R4     =%ØØØØØ4          R5     =%ØØØØØ5
SDWN   = ØØØØ1Ø           SELR   = ØØØ1ØØ          SIMCOM   ØØØ162R
SKP      ØØ1266R          SKPBSP   ØØ113ØR         SKPR   = ØØØØØ5
SP     =%ØØØØØ6           SPEC     ØØØ612R         SPECJ    ØØØ364R
SPECT    ØØØ662R          SPFST  = ØØØØØ1          SPLST  = ØØØØØ6
STAT1    ØØ121ØR          STAT2    ØØ1222R         STAT3    ØØ1234R
STAT4    ØØ1246R          TCMMD    ØØØØ52R         TRANØ    ØØØ45ØR
TRANS    ØØØ374R          TRANX    ØØØ574R         TRAN1    ØØØ47ØR
TRAN2    ØØØ5ØØR          TRAN3    ØØØ546R         TRAN4    ØØØ57ØR
TRAN5    ØØØ61ØR          TRAN6    ØØØ532R         TRAN7    ØØØ466R
TRYCNT   ØØØØ56R          TUR    = ØØØØØ1          TUSTAT   ØØ116ØR
UNIT   = ØØ34ØØ           WEOF     ØØ1Ø12R         WRITE  = ØØØØØ2
WRL    = ØØØØØ4
. ABS.   ØØØØØØ     ØØØ
         ØØ1616     ØØ1

ERRORS DETECTED:  Ø
FREE CORE:  19Ø19. WORDS
,LP:<DT:MT
```

6.1 MAGTAPE DRIVER LISTING VØØØ4A

```
        ;COPYRIGHT:-    DIGITAL EQUIPMENT CORP.,MAYNARD,MASS.
        ;               1971
        ;
        ;VERSION NO:-   V004A
        ;
                .TITLE MT
                .GLOBL MT
000000          .CSECT
000000 R0=%0    ;DDB PTR
000001 R1=%1    ;LCMMD PTR
000002 R2=%2    ;CMMD REG
000003 R3=%3    ;SP FUNC BLOCK PTR
000004 R4=%4    ;
000005 R5=%5    ;SCRATCH
000006 SP=%6
000007 PC=%7
        ;
        ;
        ;
172520 MTS=172520       ;TM11 STATUS
172522 MTC=172522       ;TM11 COMMAND
172524 MTBRC=172524     ;TM11 BYTE/RECORD COUNTER
172526 MTCMA=172526     ;TM11 CORE MEMORY ADDRESS
172530 MTD=172530       ;TM11 DATA BUFFER
172532 MTRD=172532      ;TM11 READ LINES
        ;
177776 PS=177776        ;PROCESSOR STATUS
        ;
        ;
        ;MTS BITS
        ;
100000 ILC=100000
040000 EOF=40000
020000 CRE=20000
010000 PAE=10000
004000 BGL=4000
002000 EOT=2000
001000 RLE=1000
000400 BTE=400
000200 NXM=200
000100 SELR=100
000040 BOT=40
000020 CH79=20
000010 SDWN=10
000004 WRL=4
000002 RWS=2
000001 TUR=1
000140 DENB=140
000010 PARB=10
        ;
        ;MTC BITS
        ;
100000 ERR=100000
060000 DEN=60000
```

```
       010000 POWR=10000
       004000 PAR=4000
       003400 UNIT=3400
       000200 CUR=200
       000100 INT=100
       000060 ADEX=60
       000016 CMMD=16
       000001 GOB=1
              ;
              ;MTRD BIT
              ;
       010000 GAPSDN=10000
              ;
              ;COMMANDS
              ;
       000000 RWU=0
       000001 READ=1
       000002 WRITE=2
       000003 EOFM=3
       000004 RWD=4
       000005 SKPR=5
       000006 BSPR=6
              ;
              ;
              ;
              ;THIS IS THE DEVICE DRIVER FOR THE TM11/TU10
              ;
000000 000000 MT:      .WORD   0                 ;BUSY INDICATOR
000002    177          .BYTE   177               ;ALL GENERAL STRUCTURE EXCEPT OP
000003    040          .BYTE   40                ;SPECIAL STRUCTURE = MAG TAPE
000004    020          .BYTE   20                ;BUFFER SIZE = 512 BYTES
000005    360          .BYTE   INTJ-MT           ;INTERRUPT HANDLER
000006    240          .BYTE   240               ;PRIO FOR INTERRUPT SERVICE
000007    000          .BYTE   0                 ;NO OPEN ENTRY
000010    374          .BYTE   TRANS-MT          ;TRANSFER ENTRY
000011    370          .BYTE   CLOSJ-MT                  ;CLOSE ENTRY
000012    364          .BYTE   SPECJ-MT                  ;SPECIAL FUNCTION ENTRY
000013    000          .BYTE   0
000014 052140 MT.NAM:  .RAD50  /MT /
              ;
000016    000 OPNFLG:  .BYTE   0,0,0,0,0,0,0,0   ;SET BY OPEN ROUTINE,CLEARED BY
000017    000
000020    000
000021    000
000022    000
000023    000
000024    000
000025    000
000026    377 LCMMD:   .BYTE   -1,-1,-1,-1,-1,-1,-1,-1,   ;1 BYTE FOR EACH DEVIC
000027    377
000030    377
000031    377
000032    377
000033    377
000034    377
000035    377
000036    000
000037    000          .BYTE   0,0,0,0,0,0,0,0   ;DEN/PAR FOR EACH DEVICE
```

```
000040      000
000041      000
000042      000
000043      000
000044      000
000045      000
000046      000
000047      000 INTENB: .BYTE    0
000050 000000 INTRET: .WORD    0                  ;ADDR FOR RET FROM INT HANDLER
000052      000 TCMMD:  .BYTE    0                  ;LAST CMMD SAVE LOCN FOR ERR REC
000053      000 ERRSW:  .BYTE    0                  ;SET BY ERR RECOVERY IF NOT RECO
000054 000000 RETRY:  .WORD    0                  ;RETRY COUNT FOR ERROR RECOVERY
000056 177761 TRYCNT: .WORD    -15.               ;INITIAL RETRY COUNT
000060 000000 LASTAT: .WORD    0                  ;ADDR IN LCMMD VECTOR FOR INT HA
000062 000000 CMA:    .WORD    0
000064 000000 BRC:    .WORD    0
              ;
000066 016001 INIT:   MOV      12(R0),R1          ;GET UNIT NUM
       000012
000072 042701         BIC      #174377,R1                 ;CLR EXTRA BITS
       174377
000076 010137         MOV      R1,@#MTC
       172522
000102 000301         SWAB     R1
000104 060701         ADD      PC,R1
000106 062701         ADD      #LCMMD-.,R1        ;ADDR IN LAST CMMD VECTOR
       177720
000112 012604         MOV      (SP)+,R4           ;SAVE RETURN ADDR
000114 012605         MOV      (SP)+,R5           ;SAVE ORIGINAL PC
000116 013746         MOV      @#PS,-(SP)         ;SIMULATE INT CALL
       177776
000122 010546         MOV      R5,-(SP)
000124 013746         MOV      @#44,-(SP)         ;SAVE REGS
       000044
000130 004536         JSR      R5,@(SP)+
000132 010446         MOV      R4,-(SP)           ;RESET CALLING PC
000134 010167         MOV      R1,LASTAT          ;SAVE FOR INT HANDLER
       177720
000140 004767         JSR      PC,READY           ;CHECK IF UNIT READY
       000162
000144 105711         TSTB     (R1)               ;CHECK IF DEVICE INITIALIZED
000146 100004         BPL      INITX              ;BRANCH IF IS
000150 112761         MOVB     #DENB,10(R1)       ;SET DEFAULT DEN=800,PAR=ODD
       000140
       000010
000156 105011         CLRB     (R1)               ;CLEAR NON INIT STATE
000160 000207 INITX:  RTS      PC
              ;
000162 105067 SIMCOM: CLRB     ERRSW              ;CLEAR ERROR SWITCH
       177665
000166 105767         TSTB     INTENB             ;BRANCH IF INT RET
       177655
000172 001005         BNE      MTEXIT
000174 105760         TSTB     -3(R0)             ;BRANCH IF CALLED FROM QUEUE
       177775
000200 001002         BNE      MTEXIT
```

```
000202 062706           ADD     #20,SP            ;REMOVE PC,PS AND REGS
       000020
000206 105067 MTEXIT:   CLRB    INTENB
       177635
000212 000170           JMP     @14(R0)           ;COMPLETION EXIT
       000014
              ;
              ;
              ;
              ;
000216 012667 GO:       MOV     (SP)+,INTRET      ;SAVE INT RETURN ADDR
       177626
000222 016737 GOA:      MOV     CMA,@#MTCMA
       177634
       172526
000230 016737           MOV     BRC,@#MTBRC
       177630
       172524
000236 121127           CMPB    (R1),#WRITE       ;CHECK IF THIS IS A WRITE
       000002
000242 001007           BNE     GO2               ;BRANCH IF NOT
000244 032737 GO1:      BIT     #WRL,@#MTS        ;CHECK IF WRITE LOCK ON
       000004
       172520
000252 001403           BEQ     GO2               ;BRANCH IF NOT
000254 004767           JSR     PC,READY1         ;ISSUE ACTION MSG
       000034
000260 000771           BR      GO1               ;GO TEST IF LOCK STILL ON
000262 004767 GO2:      JSR     PC,READY          ;CHECK IF DEVICE READY
       000040
000266 156137           BISB    10(R1),@#MTC+1    ;SET DEN AND PAR
       000010
       172523
000274 052702           BIS     #INT+GOB,R2               ;SET INT ENB AND GO BITS
       000101
000300 110237           MOVB    R2,@#MTC          ;ISSUE INSTRUCTION
       172522
000304 013746 GO3:      MOV     @#46,-(SP)        ;RESTORE REGS
       000046
000310 004536           JSR     R5,@(SP)+
000312 000002           RTI                       ;RETURN TO INTERRUPT
              ;
              ;
000314 016746 READY1:   MOV     MT.NAM,-(SP)      ;ISSUE ACTION DIAG -
       177474
000320 012746           MOV     #402,-(SP)        ;DEVICE NOT READY
       000402
000324 000004           IOT
000326 032737 READY:    BIT     #SELR,@#MTS       ;TEST IF DEVICE READY
       000100
       172520
000334 001767           BEQ     READY1            ;BRANCH IF NOT
000336 032737           BIT     #TUR+RWS+SDWN,@#MTS
       000013
       172520
000344 001004           BNE     READYX
```

```
000346 032737          BIT     #GAPSDN,@#MTRD
       010000
       172532
000354 001757          BEQ     READY1
000356 000207 READYX:  RTS     PC                ;RETURN TO CALLER
              ;
000360 000167 INTJ:    JMP     INTH
       000734
000364 000167 SPECJ:   JMP     SPEC
       000222
000370 000167 CLOSJ:   JMP     CLOSE
       000670
              ;
              ;
000374 004767 TRANS:   JSR     PC,INIT           ;INIT CHECK DEVICE
       177466
000400 016067          MOV     6(R0),CMA         ;SET BUFF ADDR
       000006
       177454
000406 016067          MOV     10(R0),BRC        ;SET WORD COUNT
       000010
       177450
000414 006367          ASL     BRC               ;CVT TO BYTE COUNT
       177444
000420 016002          MOV     12(R0),R2
       000012
000424 042702          BIC     #177713,R2        ;CLR ALL BUT READ AND ADDR EXT B
       177713
000430 032702          BIT     #4,R2             ;CHECK INPUT/OUTPUT
       000004
000434 001405          BEQ     TRANO             ;BRANCH IF OUTPUT
000436 112711          MOVB    #READ,(R1)        ;SET LAST CMMDSREAD
       000001
000442 162702          SUB     #2,R2             ;SET UP READ CMMD
       000002
000446 000414          BR      TRAN2
000450 032737 TRANO:   BIT     #EOT,@#MTS        ;CHECK IF AT EOT
       002000
       172520
000456 001404          BEQ     TRAN1             ;BRANCH IF NOT
000460 016060          MOV     10(R0),16(R0)     ;RETURN WORD COUNT
       000010
       000016
000466 000635 TRAN7:   BR      SIMCOM            ;REJECT CMMD - EXIT
000470 112711 TRAN1:   MOVB    #WRITE,(R1)       ;SET LAST CMMD S WRITE
       000002
000474 062702          ADD     #4,R2             ;SET UP WRITE CMMD
       000004
000500 004767 TRAN2:   JSR     PC,GO             ;GO INITIATE I/O
       177512
000504 013702          MOV     @#MTBRC,R2
       172524
000510 121127          CMPB    (R1),#READ        ;COME HERE AFTER INT ERR CHK
       000001
000514 001006          BNE     TRAN6             ;BRANCH IF NOT READ
000516 032737          BIT     #EOF,@#MTS        ;IF EOF, SET INIT CNT IN RESIDUE
       040000
       172520
```

```
000524 001402           BEQ     TRAN6
000526 016702           MOV     BRC,R2
       177332
000532 006202 TRAN6:    ASR     R2       ;CHECK IF ODD BYTES SHORT REC
000534 103004           BCC     TRAN3            ;BRANCH IF NOT
000536 013705           MOV     @#MTCMA,R5       ;PUT NULL IN NEXT BUFF POS
       172526
000542 105015           CLRB    (R5)
000544 005202           INC     R2               ;ROUND UP WORD COUNT
000546 001402 TRAN3:    BEQ     .+6              ;BRANCH IF NO RESIDUE
000550 052702           BIS     #100000,R2       ;INSURE NEG WORD COUNT
       100000
000554 032737           BIT     #RLE,@#MTS       ;CHECK IF LENGTH ERROR
       001000
       172520
000562 001402           BEQ     TRAN4            ;BRANCH IF NOT
000564 105267           INCB    ERRSW
       177263
000570 010260 TRAN4:    MOV     R2,16(R0)        ;RETURN RESIDUE WORD COUNT
       000016
000574 105767 TRANX:    TSTB    ERRSW            ;BRANCH IF NO ERRORS
       177253
000600 001403           BEQ     TRAN5
000602 052760           BIS     #100000,12(R0)   ;SET ERR BIT
       100000
       000012
000610 000726 TRAN5:    BR      TRAN7            ;TAKE DONE EXIT
              ;
000612 004767 SPEC:     JSR     PC,INIT          ;INIT CHECK DEVICE
       177250
000616 016003           MOV     2(R0),R3         ;GET FUNC BLOCK ADDR
       000002
000622 111305           MOVB    (R3),R5 ;GET FUNC BYTE
000624 162705           SUB     #SPFST,R5
       000001
000630 100767           BMI     TRAN5            ;BRANCH OUT IF NOT
000632 022705           CMP     #SPLST,R5        ;SUPPORTED FUNCTION
       000006
000636 103764           BLO     TRAN5
000640 122763           CMPB    #3,1(R3)         ;CHECK IF VALID FUNC BLOCK
       000003
       000001
000646 101014           BHI     ABORT            ;ABORT IF NOT
000650 006305           ASL     R5
000652 060705           ADD     PC,R5
000654 062705           ADD     #SPECT-.,R5
       000006
000660 000115           JMP     @R5              ;GO TO PROPER SP FUNC ROUTINE
              ;
              ;
       000001 SPFST=1
       000006 SPLST=6
              ;
000662 000412 SPECT:    BR      OFFLIN
000664 000452           BR      WEOF
000666 000417           BR      RWND
```

```
000670 000476            BR      SKP
000672 000502            BR      BSP
000674 000453            BR      PARDEN
000676 000530            BR      TUSTAT
                  ;
                  ;
000700 010346 ABORT:     MOV     R3,-(SP)          ;ISSUE SP FUNC BLOCK BAD ABORT
000702 012746            MOV     #1433,-(SP)       ;WITH ADDR OF SP FUNC BLOCK
       001433
000706 000004            IOT
                  ;
000710 004767 OFFLIN:    JSR     PC,EOFCK          ;WRITE EOF IF NECESSARY
       000052
000714 105011            CLRB    (R1)              ;SET LAST CMMD=OFFLINE
000716 112737            MOVB    #1,@#MTC          ;ISSUE DISABLED RWV
       000001
       172522
000724 000515            BR      TUSTAT            ;GET STAT AND EXIT
000726 004767 RWND:      JSR     PC,EOFCK          ;ISSUE WRITE EOF IF NECESSARY
       000034
000732 004767            JSR     PC,RWNDC          ;ISSUE DISABLED RWD
       000002
000736 000510            BR      TUSTAT            ;GET STATUS AND EXIT
000740 112711 RWNDC:     MOVB    #RWD,(R1)         ;SET LAST CMMD=RWD
       000004
000744 032737            BIT     #BOT+RWS,@#MTS    ;IS IT REWOUND
       000042
       172520
000752 001004            BNE     RWNDX    ;YES, BRANCH
000754 012702            MOV     #16,R2            ;ISSUE RWD
       000016
000760 000167            JMP     GO
       177232
000764 000207 RWNDX:     RTS     PC
                  ;
000766 121127 EOFCK:     CMPB    (R1),#WRITE       ;IF LAST CMMD WAS WRITE
       000002
000772 001401            BEQ     EOFCK1            ;BRANCH
000774 000207            RTS     PC                ;ELSE RETURN
000776 012702 EOFCK1:    MOV     #6,R2             ;SET CMMD=WRITE EOF
       000006
001002 012667            MOV     (SP)+,INTRET      ;SET INT RET ADDR
       177042
001006 000167            JMP     GO1               ;GO EXECUTE
       177232
                  ;
001012 112711 WEOF:      MOVB    #EOFM,(R1)        ;SET LAST CMMD
       000003
001016 004767            JSR     PC,EOFCK1         ;GO EXECUTE WRITE EOF
       177754
001022 000456            BR      TUSTAT   ;AT INT RET, GET STAT AND EXIT
                  ;
                  ;
001024 032737 PARDEN:    BIT     #CH79,@#MTS       ;IF 9 TRACK TAPE
       000020
       172520
```

```
001032 001452           BEQ     TUSTAT          ;BRANCH (IGNORE NEW SETTINGS)
001034 016305           MOV     4(R3),R5        ;GET NEW DEN/PAR
       000004
001040 042705           BIC     #176376,R5      ;CLR EXTRA BITS
       176376
001044 106205           ASRB    R5              ;SET INTO PROPER POSITION
001046 106005           RORB    R5
001050 106005           RORB    R5
001052 006005           ROR     R5
001054 006005           ROR     R5
001056 006005           ROR     R5
001060 110561           MOVB    R5,10(R1)       ;SET NEW DEN/PAR
       000010
001064 000435           BR      TUSTAT
                 ;
001066 112711    SKP:   MOVB    #SKPR,(R1)      ;SET LAST CMMD=SKP
       000005
001072 012702           MOV     #8.,R2          ;SET CMMD
       000010
001076 000414           BR      SKPBSP          ;GO SET COUNT AND EXEC
001100 112711    BSP:   MOVB    #BSPR,(R1)      ;SET LAST CMMD=BSP
       000006
001104 012702           MOV     #10.,R2         ;SET CMMD
       000012
001110 032737           BIT     #BOT+RWS,@#MTS  ;TEST IF AT BOT
       000042
       172520
001116 001404           BEQ     SKPBSP          ;BRANCH IF NOT
001120 016363           MOV     4(R3),6(R3)     ;REJECT CMMD RETURN REC COUNT
       000004
       000006
001126 000414           BR      TUSTAT          ;SET STATUS AND EXIT
001130 016367    SKPBSP: MOV    4(R3),BRC       ;SET RECORD COUNT
       000004
       176726
001136 005467           NEG     BRC
       176722
001142 004767           JSR     PC,GO           ;GO EXECUTE CMMD
       177050
001146 013763           MOV     @#MTBRC,6(R3)   ;SET RESIDUE REC COUNT
       172524
       000006
001154 005463           NEG     6(R3)
       000006
                 ;
001160 116102    TUSTAT: MOVB   10(R1),R2       ;GET DEN/PAR
       000010
001164 111101           MOVB    (R1),R1 ;GET LAST CMMD
001166 000302           SWAB    R2
001170 050201           BIS     R2,R1           ;SET DEN AND PAR
001172 013702           MOV     @#MTS,R2
       172520
001176 032702           BIT     #CRE+PAE+RLE,R2 ;TEST STATUS FOR ERROR
       031000
001202 001402           BEQ     STAT1           ;BRANCH IF NONE
001204 052701           BIS     #100000,R1      ;SET ERR BIT
       100000
```

```
001210 032702 STAT1:  BIT     #EOT,R2         ;TEST IF EOT
       002000
001214 001402         BEQ     STAT2           ;BRANCH IF NOT
001216 052701         BIS     #1000,R1        ;SET EOT BIT
       001000
001222 032702 STAT2:  BIT     #BOT+RWS,R2     ;TEST IF AT BOT
       000042
001226 001402         BEQ     STAT3           ;BRANCH IF NOT
001230 052701         BIS     #400,R1         ;SET BOT BIT
       000400
001234 032702 STAT3:  BIT     #EOF,R2         ;TEST IF EOF
       040000
001240 001402         BEQ     STAT4           ;BRANCH IF NOT
001242 052701         BIS     #200,R1         ;SET EOF BIT
       000200
001246 042702 STAT4:  BIC     #177753,R2      ;CLEAR ALL BUT WRL AND 79CH BITS
       177753
001252 000302         SWAB    R2
001254 050201         BIS     R2,R1           ;SET WRL AND 7,9 TRACK
001256 010163         MOV     R1,2(R3)        ;RETURN STATUS
       000002
001262 000414         BR      COMJ            ;EXIT
              ;
              ;
              ;
001264 004767 CLOSE:  JSR     PC,INIT         ;INIT CHECK ON DEVICE
       176576
001270 105061         CLRB    -10(R1)         ;CLEAR OPEN FLAG
       177770
001274 004767         JSR     PC,EOFCK        ;IF LAST CMMD WAS WRITE, WRITE 3
       177466
001300 004767         JSR     PC,EOFCK
       177462
001304 004767         JSR     PC,EOFCK
       177456
001310 004767         JSR     PC,RWNDC        ;ISSUE DISABLED RWD
       177424
001314 000167 COMJ:   JMP     SIMCOM
       176642
              ;
              ;
              ;
              ;
001320 013746 INTH:   MOV     @#44,-(SP)      ;SAVE REGS
       000044
001324 004536         JSR     R5,@(SP)+
001326 016700         MOV     MT,R0           ;GET DDB ADDR
       176446
001332 016701         MOV     LASTAT,R1       ;GET LCMMD VECTOR ADDR
       176522
001336 016003         MOV     2(R0),R3        ;GET SP FNC BLOCK ADDR
       000002
001342 012705         MOV     #MTC,R5         ;ADDR OF CMMD REG
       172522
001346 042715         BIC     #100,(R5)       ;DISABLE DEVICE INTERRUPT
       000100
```

```
001352 013702           MOV     @#MTS,R2        ;GET STATUS OF DEVICE
       172520
001356 105267           INCB    INTENB          ;SET INT FLAG
       176465
001362 032702           BIT     #ILC+NXM,R2     ;CHECK ILLEGAL CMMD, NONEXIST CO
       100200
001366 001404           BEQ     INT1            ;BRANCH IF NOT
001370 010246 INTF:     MOV     R2,-(SP)        ;DISPLAY STATUS AND DIAGNOSE
001372 012746           MOV     #1432,-(SP)     ;FATAL ERROR-MAG TAPE
       001432
001376 000004           IOT                     ;ABORT
              ;
001400 105767 INT1:     TSTB    TCMMD           ;CHECK IF THIS WAS A RETRY
       176446
001404 001433           BEQ     INT3            ;BRANCH IF NOT
001406 100007           BPL     INT2            ;BRANCH IF WAS NOT BSP OF RETRY
001410 005002           CLR     R2
001412 156702           BISB    TCMMD,R2        ;GET CMMD
       176434
001416 105467           NEGB    TCMMD           ;SET-NOT BSP
       176430
001422 000167           JMP     GOA             ;GO TRY AGAIN
       176574
              ;
001426 032702 INT2:     BIT     #BGL+BTE+CRE+PAE,R2;TEST IF ERROR THIS TIME
       034400
001432 001463           BEQ     INT7            ;BRANCH IF NOT
001434 005267           INC     RETRY
       176414
001440 001036           BNE     INT6            ;BRANCH IF TO TRY AGAIN
001442 032702           BIT     #BGL,R2         ;IF ERR=BUS GRANT LATE
       004000
001446 001350           BNE     INTF            ;IS FATAL
001450 105267           INCB    ERRSW           ;SET ERROR FLAG
       176377
001454 032715           BIT     #4,(R5) ;IF WRITE OR WEOF, ISSUE ACTION DIAG
       000004
001460 001450           BEQ     INT7
001462 010246           MOV     R2,-(SP)
001464 012746           MOV     #406,-(SP)
       000406
001470 000004           IOT
001472 000443           BR      INT7            ;GO TO SPECIFIC ROUTINE
              ;
001474 032702 INT3:     BIT     #BGL+BTE+PAR+CRE,R2;CHECK IF ERROR
       030400
001500 001440           BEQ     INT7            ;BRANCH IF NOT
001502 122711           CMPB    #SKPR,(R1)      ;BRANCH IF SKIP OR BSP
       000005
001506 001424           BEQ     INT4
001510 122711           CMPB    #BSPR,(R1)
       000006
001514 001421           BEQ     INT4
001516 016767           MOV     TRYCNT,RETRY    ;SET RETRY COUNT
       176334
       176330
```

```
001524 032715           BIT     #12,(R5)         ;IF CMMD IS WRITE
       000012
001530 001002           BNE     INT6
001532 052715           BIS     #10,(R5)         ;TRY WRITE WITH LONG GAP
       000010
001536 111567 INT6:     MOVB    (R5),TCMMD       ;SAVE CMMD
       176310
001542 112702           MOVB    #10.,R2          ;SET UP BSP
       000012
001546 012737           MOV     #-1,@#MTBRC      ;COUNT OF 1
       177777
       172524
001554 000167           JMP     GO2              ;GO EXECUTE
       176502
              ;
001560 032702 INT4:     BIT     #EOF+BOT,R2      ;IF EOF OR BOT
       040040
001564 001006           BNE     INT7             ;SKIP OR BSP IS DONE
001566 005737           TST     @#MTBRC          ;IF COUNT EXHAUSTED
       172524
001572 001403           BEQ     INT7             ;IS DONE
001574 111502           MOVB    (R5),R2 ;ELSE SET UP CMMD
001576 000167           JMP     GO2              ;CONTINUE SKIP OR BSP
       176460
              ;
001602 105067 INT7:     CLRB    TCMMD            ;CLEAR RETRY INDICATORS
       176244
001606 105067           CLRB    RETRY
       176242
001612 016707           MOV     INTRET,PC        ;GO TO SPECIFIC ROUTINE
       176232
              ;
              ;
       000001           .END
```

000000 ERRORS

```
ABORT      000700R    ADEX   =  000060     BGL     =  004000
BOT     =  000040     BRC       000064R    BSP        001100R
BSPR    =  000006     BTE    =  000400     CH79    =  000020
CLOSE      001264R    CLOSJ     000370R    CMA        000062R
CMMD    =  000016     COMJ      001314R    CRE     =  020000
CUR     =  000200     DEN    =  060000     DENB    =  000140
EOF     =  040000     EOFCK     000766R    EOFCK1     000776R
EOFM    =  000003     EOT    =  002000     ERR     =  100000
ERRSW      000053R    GAPSDN =  010000     GO         000216R
GOA        000222R    GOB    =  000001     GO1        000244R
GO2        000262R    GO3       000304R    ILC     =  100000
INIT       000066R    INITX     000160R    INT     =  000100
INTENB     000047R    INTF      001370R    INTH       001320R
INTJ       000360R    INTRET    000050R    INT1       001400R
INT2       001426R    INT3      001474R    INT4       001560R
INT6       001536R    INT7      001602R    LASTAT     000060R
LCMMD      000026R    MT        000000RG   MTBRC   =  172524
MTC     =  172522     MTCMA  =  172526     MTD     =  172530
MTEXIT     000206R    MTRD   =  172532     MTS     =  172520
MT.NAM     000014R    NXM    =  000200     OFFLIN     000710R
OPNFLG     000016R    PAE    =  010000     PAR     =  004000
PARB    =  000010     PARDEN    001024R    PC      =%000007
POWR    =  010000     PS     =  177776     READ    =  000001
READY      000326R    READYX    000356R    READY1     000314R
RETRY      000054R    RLE    =  001000     RWD     =  000004
RWND       000726R    RWNDC     000740R    RWNDX      000764R
RWS     =  000002     RWU    =  000000     R0      =%000000
R1      =%000001      R2     =%000002      R3      =%000003
R4      =%000004      R5     =%000005      SDWN    =  000010
SELR    =  000100     SIMCOM    000162R    SKP        001066R
SKPBSP     001130R    SKPR   =  000005     SP      =%000006
SPEC       000612R    SPECJ     000364R    SPECT      000662R
SPFST   =  000001     SPLST  =  000006     STAT1      001210R
STAT2      001222R    STAT3     001234R    STAT4      001246R
TCMMD      000052R    TRAN0     000450R    TRANS      000374R
TRANX      000574R    TRAN1     000470R    TRAN2      000500R
TRAN3      000546R    TRAN4     000570R    TRAN5      000610R
TRAN6      000532R    TRAN7     000466R    TRYCNT     000056R
TUR     =  000001     TUSTAT    001160R    UNIT    =  003400
WEOF       001012R    WRITE  =  000002     WRL     =  000004
.       =  001616R
```

DEC-11-ODLPA-A-D

P D P - 1 1

L P 1 1   L I N E   P R I N T E R   D R I V E R

October 1972

SUPPLEMENT TO:

PDP-11 DEVICE DRIVER PACKAGE

DEC-11-ODDPA-A-D

MONITOR VERSION VØØ8

NEW AND CHANGED INFORMATION

This manual documents the software as of Monitor version VØ8. It has been revised to include all new and changed material since Monitor version VØ4. Such material is indicated by vertical bars in the outside margin. Whole new pages are not so marked but are dated in the lower outside corner.

pdp11 SOFTWARE DISTRIBUTION CENTER

LP11 LINE PRINTER DRIVER

The line printer driver provides the basic, device specific functions for the PDP-11 Line Printer (LP11) or the Centronics 101A. The driver accepts a block of any specified length (48-word standard) and feeds it to the printer. The block may contain any number of lines (line feed characters) or pages (form feed characters) to be printed in a single call to the driver.

The line printer driver consists of two sections: the fixed driver table and the driver code. The driver table gives the following information:

- Line printer facilities:

  Single user
  Output only
  ASCII only
  Non-file structured

- Standard buffer size of 48 words

- Entry points to the various line printer function routines.

The detailed description of the functions of the line printer driver is given in the following flow chart. The following special points should be noted:

1. Both the OPEN and CLOSE functions cause a skip to head of form (a form feed is printed) on the printer.
2. The transfer (and interrupt) function(s) transfer as many characters as possible to the line printer with the line printer interrupt temporarily disabled. This transfer terminates when one of two conditions is reached:

   a. The line printer starts a physical operation (because its buffer is full, or because a line terminator character was transferred); or
   b. The transfer count is exhausted.

3. Special character handling: NUL's, DEL's and VT's are deleted; AUX-ON is transmitted as LF (for LP11) or as VT (for Centronics); CR is transmitted (if necessary) before LF, VT, or FF; TABs are transmitted as 1-8 SPACEs (depending on current line position); all other characters are passed without change.

4. Trailing SPACEs (and TABs) on a line are not printed.

The maximum characters per line is an assembly parameter, which may be specified by statements:

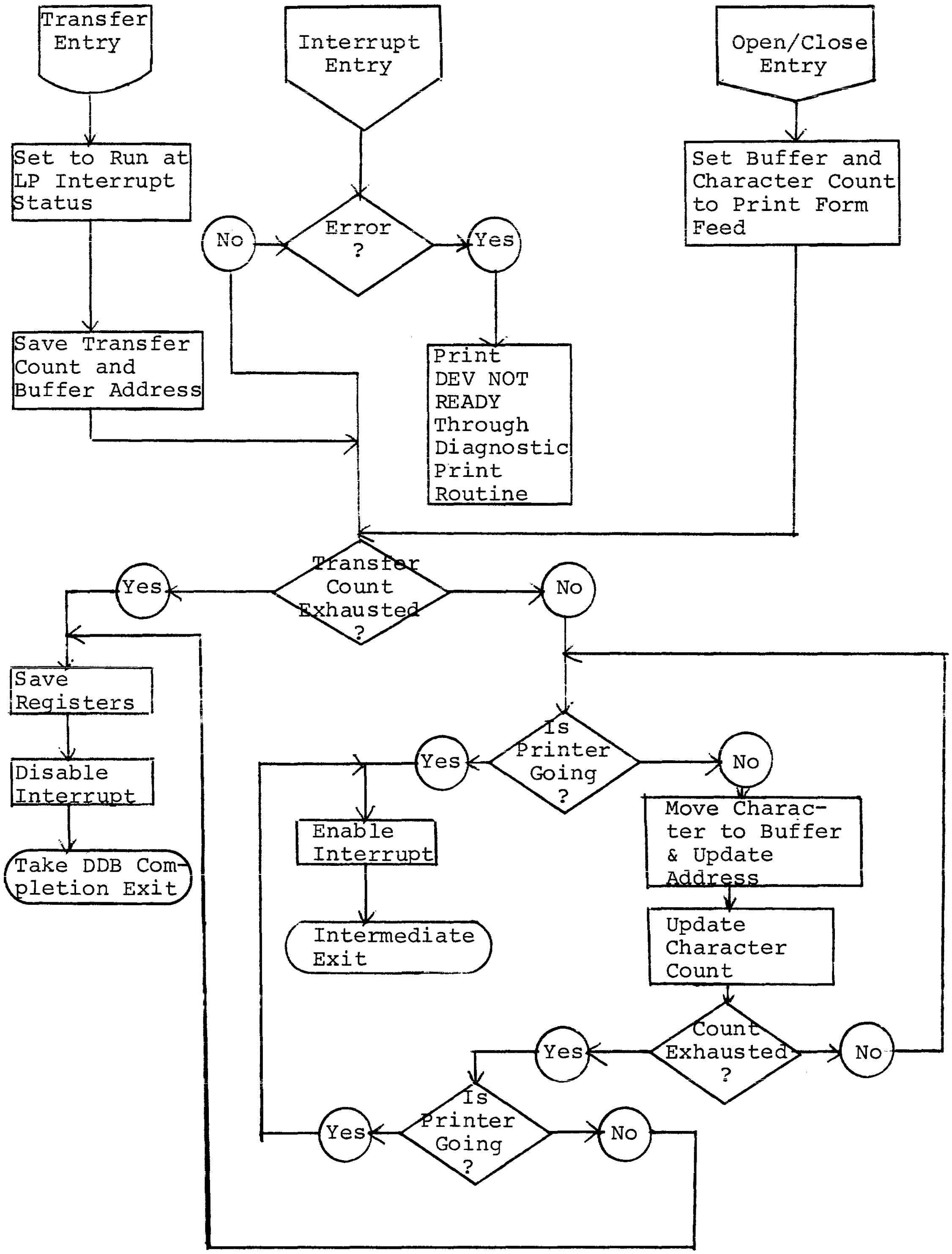
```
LP11=80 or LP11=132
```

If not specified, LP11=80 by default. Furthermore, the Centronics line printer version of this driver is produced by an assembly parameter, specified as:

```
CENT=132
```

If specified, CENT causes code unique to the Centronics printer to be assembled and overrides any LP11 parameter specification.

A flow chart and listings of the driver follow.


Transfer Entry
Interrupt Entry
Open/Close Entry
Set to Run at LP Interrupt Status
Set Buffer and Character Count to Print Form Feed
No
Error ?
Yes
Save Transfer Count and Buffer Address
Print DEV NOT READY Through Diagnostic Print Routine
Yes
Transfer Count Exhausted ?
No
Save Registers
Disable Interrupt
Take DDB Completion Exit
Yes
Is Printer Going ?
No
Enable Interrupt
Move Character to Buffer & Update Address
Intermediate Exit
Update Character Count
Yes
Count Exhausted ?
No
Yes
Is Printer Going ?
No

A listing of the V007A driver for use under DOS Monitor release V08-02 follows;

```
DV.LP    MACRO V004-14 13-SEP-72 03:11 PAGE 1


                      ;       COPYRIGHT 1972 DIGITAL EQUIPMENT CORPORATION

                      ;VERSION NUMBER          V007A    ,003


                              .TITLE  DV.LP
                              .IFNDF  CENT
                              .IFNDF  LP11
                      LP11=80.        ;DEFAULT TO 80. COLUMN LP-11
                              .ENDC
                              .ENDC
                              .IFDF   LP11
              000120 LP.SIZ=LP11      ;NUMBER OF PRINT POSITIONS
              000012 LP.SK2=12        ;LF -- TRANSLATION OF "SKIP-CH-2" (22)
                              .ENDC
                              .IFDF   CENT
                      LP.SIZ=CENT     ;NUMBER OF PRINT POSITIONS
                      LP.SK2=13       ;VT -- TRANSLATION OF "SKIP-CH-2" (22)
                              .ENDC
                              .GLOBL LP
                      ;DRIVER FOR LP11 AND CENTRONICS 101
                      ;CHANGE LINES LP.NAM TO LP.SK2 FOR
                              ;DEVICE DEPENDENT CHARACTERISTICS
 00000 000000 LP:     .WORD   0                 ;CURRENT DDB PTR
 00002    322         .BYTE   322               ;FACILITIES INDICATOR
 00003    000         .BYTE   0
 00004    003         .BYTE   3                 ;NO. BLF UNITS/BUFFER
                                                ;(ALLOWS 96 BYTES/TRAN)
 00005    102         .BYTE   LP.INT-LP         ;INTERRUPT ENTRY
 00006    200         .BYTE   200               ;INT STATUS (PRI=4)
 00007    030         .BYTE LP.OPN-LP           ;OPEN ENTRY
 00010    052         .BYTE   LP.TRN-LP         ;TRAN ENTRY
 00011    030         .BYTE   LP.CLS-LP         ;CLOSE ENTRY
 00012    000         .BYTE   0,0               ;SPECIAL,SPARE
 00013    000
 00014 046600 LP.NAM: .RAD50  /LP/              ;DEVICE NAME
        177514 LP.CSR=177514                    ;CSR ADDR
        000200 LP.TRP=200                       ;TRAP VEC ADDR
                                                ;FOR LP11, USE 12(LF)
                                                ;FOR CENTRONICS, USE 13(VT)
 00016 000000 LP.LIN: .WORD   0                 ;# CHARS SENT FOR THIS LINE
 00020 000000 LP.BKS: .WORD   0                 ;BLANK COUNTER
 00022 000000 BTCT:   .WORD   0                 ;TRAN CHAR COUNT (COMPL)
 00024 000000 BUFAD:  .WORD   0                 ;BUF PTR
                                                ;REGISTER DEFINITIONS
        000000 R0=%0
        000001 R1=%1
        000002 R2=%2
        000003 R3=%3
        000004 R4=%4
        000005 R5=%5
        000006 SP=%6
        000007 PC=%7
 00026    015 LP.FRM: .BYTE   15,14             ;CR,FF
 00027    014
              ;OPEN AND CLOSE
 00030        LP.OPN:LP.CLS:
```

```
00030 004767          JSR     PC,LP.STS       ;SIM INT
      000364
00034 062701          ADD     #LP.FRM-.,R1    ;R1=PC BY LP.STS
      177772
00040 010167          MOV     R1,BUFAD        ;ADDR OF CR,FF
      177760
00044 010267          MOV     R2,BTCT         ;R2=-2 BY LP.STS
      177752
00050 000414          BR      LP.INT

             ;TRAN
00052        LP.TRN:
00052 004767          JSR     PC,LP.STS       ;SIM INT
      000342
00056 016700          MOV     LP,R0           ;DDB ADDR
      177716
00062 016067          MOV     6(R0),BUFAD     ;SAVE BUF ADDR
      000006
      177734
00070 016067          MOV     10(R0),BTCT     ;DDB WORD COUNT
      000010
      177724
00076 006367          ASL     BTCT            ;CHANGE TO BYTES
      177720
             ;INTERRUPT
00102        LP.INT:
00102 042737 LP.DIS:  BIC     #100,@#LP.CSR   ;DISABLE
      000100
      177514
00110 005737          TST     @#LP.CSR        ;ERROR TEST
      177514
00114 100533          BMI     LP.ERR

             ;NO ERROR,SO CONTINUE
00116 010246          MOV     R2,-(SP)
00120 010146          MOV     R1,-(SP)
00122 016746          MOV     LP.LIN,-(SP)
      177670
00126 005767          TST     BTCT            ;CURRENT BYTE RESIDUE
      177670
00132 001476          BEQ     LP.DNE          ;DONE (NO MORE)
00134 016702          MOV     BUFAD,R2        ;CURRENT BUF ADDR
      177664
00140        LP.LOP:                          ;MAIN LOOP
00140 112201          MOVB    (R2)+,R1        ;NEXT CHAR
00142 001464          BEQ     LP.DNP          ;SKIP IF NULL
00144        LP.LPB:
             ;NOTE: THE NEXT FOUR INTRUCTIONS MAY BE DELETED,
             ;        IF TRAILING SPACE SUPPRESSION IS NOT NEEDED --
             ;        TRAILING TABS WILL STILL BE SUPPRESSED, HOWEVER...
00144 120127          CMPB    R1,#40          ;BLANK?
      000040
00150 001003          BNE     .+10
00152 005267          INC     LP.BKS          ;JUST INC BLANK COUNT AND MOVE O
      177642
00156 000456          BR      LP.TRT
00160 120127          CMPB    R1,#13          ;VT?
```

```
       000213
00164  001453          BEQ     LP.DNP
00166  122127          CMPB    R1,#22          ;CHANNEL 2?
       000022
00172  001002          BNE     .+6             ;NO
00174  112701          MOVB    #LP.SK2,R1      ;YES, TRANSLATE
       000012
00200  122127          CMPB    R1,#177         ;RUBOUT?
       000177
00204  001443          BEQ     LP.DNP
0206   105737          TSTB    @#LP.CSR        ;CHECK DEVICE READY
       177514
0212   100061          BPL     LP.STI          ;QUIT IF NOT
               ;DEVICE CAN ACCEPT ANOTHER CHAR
               ;PROCEED TO CHECK
0214   122127          CMPB    R1,#11          ;TAB?
       000011
0220   001507          BEQ     LP.PTB  ;YES, GO SIMULATE IT
0222   103415          BLO     LP.CLC          ;NOT A TERMINATOR EITHER
0224   122127          CMPB    R1,#15          ;CR,FF,VT,LF?
       000015
0230   101012          BHI     LP.CLC          ;NO
0232   001407          BEQ     LP.RSC          ;TREAT CR SPECIALLY
               ;GUARANTEE CR BEFORE TERMINATORS
0234           LP.TRM:
0234   005716          TST     @SP             ;AT BEGINNING OF LINE?
0236   001422          BEQ     LP.RSB          ;YES
                                               ;NO, FORCE CR
0240   005302          DEC     R2
0242   005367          DEC     BTCT
       177554
0246   112701          MOVB    #15,R1
       000015
0252           LP.RSC:                 ;RESET COUNTS
0252   005016          CLR     @SP             ;NEW LINE
0254   000413          BR      LP.RSB

0256           LP.CLC:                         ;CHECK LINE OVERFLOW
0256   021627          CMP     @SP,#LP.STZ
       000120
0262   002014          BGE     LP.DNP          ;SKIP IF FULL
0264   005216          INC     @SP             ;ELSE COUNT LINE CHARS
0266   005367          DEC     LP.BKS          ;CHECK BLANK COUNT
       177526
0272   100404          BMI     LP.RSB          ;GO AHEAD IF NO BLANK COUNT
0274   112737          MOVB    #40,@#LP.CSR+2  ;OTHERWISE, PUT OUT A BLANK
       000040
       177516
0302   000720          BR      LP.LPB          ;AND TRY AGAIN

0304   005067  LP.RSB: CLR     LP.BKS          ;SHOW NO BLANKS WAITING FOR PRIN
       177510

0310   110137  LP.TBF: MOVB    R1,@#LP.CSR+2   ;OUTPUT CHAR
       177516
0314           LP.DNP:
0314   005267  LP.TRT: INC     BTCT            ;UPDATE BUF RESIDUE
```

```
          177502
     0320 001307            BNE     LP.LOP          ;MORE?
                                                    ;DONE FOR NOW
     0322 105737            TSTB    @#LP.CSR        ;DEV BUSY?
          177514
     0326 100014            BPL     LP.STJ          ;YES

     0330          LP.DNE:                          ;NO, SO NO INTERRUPT
     0330 012667            MOV     (SP)+,LP.LIN    ;RESTORE TEMPORARIES
          177462
     0334 012601            MOV     (SP)+,R1
     0336 012602            MOV     (SP)+,R2
     0340 013746            MOV     @#S.RSV,-(SP)
          000244
     0344 004536            JSR     R5,@(SP)+       ;SAVE REGS

     0346 016700            MOV     LP,R0           ;DDB PTR
          177426
     0352 000170            JMP     @14(R0)         ;DDB COMPLETE - EXIT
          000214
                   ;
     0356 005302 LP.STI:    DEC     R2              ;SET TO RESCAN BYTE
     0360 012667 LP.STJ:    MOV     (SP)+,LP.LIN    ;SAVE COUNTS FOR INTERRUPT
          177432
     0364 010267            MOV     R2,BUFAD
          177434
     0370 012601            MOV     (SP)+,R1        ;RESTORE REGS
     0372 012602            MOV     (SP)+,R2
     0374 052737            BIS     #100,@#LP.CSR   ;ALLOW INTERRUPT
          000100
          177514
     0402 000002            RTI                     ;RETURN TO USER
     0404          LP.ERR:                          ;ERROR ON DEVICE
     0404 016746            MOV     LP.NAM,-(SP)    ;SHOW DEVC NAME
          177404
     0410 012746            MOV     #402,-(SP)      ;GIVE 1-2 ERR CODE
          000402
     0414 000004            IOT
     0416 000631            BR      LP.DIS          ;TRY AGAIN
                   ;
                   ;INTERRUPT SIMULATOR
     0420 012601 LP.STS:    MOV     (SP)+,R1        ;RTN PC
     0422 011646            MOV     (SP),-(SP)      ;OLD PC
     0424 005002            CLR     R2              ;ADDRESS PS (-2)
     0426 014266            MOV     -(R2),2(SP)     ;OLD STATUS
          000002
     0432 013712            MOV     @#LP.TRP+2,(R2) ;NEW STATUS
          000202
     0436 010107            MOV     R1,PC           ;RETURN
                   ;
                   ;TAB SIMULATOR
     0440          LP.PTR:
     0440 011646            MOV     @SP,-(SP)       ;CURRENT LINE POSITION
     0442 066716            ADD     LP.BKS,@SP      ;+ACCUMULATED BLANKS
          177352
     0446 052716            BIS     #177770,@SP     ;MOD(X,8)-8
          177770
```

```
    0452 162667            SUB      (SP)+,LP.BKS     ;ADD 1 TO 8 BLANKS
         177342
    0456 000716            BR       LP.TRT           ;MOVE ON

         177776 S.STAT=177776                        ;PS ADDR
         000044 S.RSV=44                             ;ADDR OF REGSAVE ROUTINE
         000001            .END     LP
```


SYMBOL TABLE

```
BTCT     000022R         BUFAD    000024R         LP       000000RG
LP.BKS   000020R         LP.CLO   000256R         LP.CLS   000030R
LP.CSR=  177514          LP.DIS   000102R         LP.DNF   000330R
LP.DNP   000314R         LP.ERR   000404R         LP.FRM   000026R
LP.INT   000102R         LP.LIN   000016R         LP.LOP   000140R
LP.LPB   000144R         LP.NAM   000014R         LP.OPN   000030R
LP.PTB   000440R         LP.RSB   000304R         LP.RSC   000252R
LP.SIZ=  000120          LP.SK2=  000012          LP.STI   000356R
LP.STJ   000360R         LP.STS   000420R         LP.TBF   000310R
LP.TRM   000234R         LP.TRN   000052R         LP.TRP=  000200
LP.TRT   000314R         LP11   = 000120          PC     =%000007
R0     =%000000          R1     =%000001          R2     =%000002
R3     =%000003          R4     =%000004          R5     =%000005
SP     =%000006          S.RSV  = 000044          S.STAT=  177776

. ABS.   000000     000
         000460     001

ERRORS DETECTED:  0
FREE CORE:  19357. WORDS
,LP:<DT:LP
```

A listing of the V001A driver for use under DOS Monitor release V04A follows:

```
                ;COPYRIGHT 1971, DIGITAL EQUIPMENT CORP., MAYNARD, MASS.

                ;VERSION NUMBER:        V001A

                ; LINE PRINTER DRIVER (LP)
        000120 PRSIZE=80.               ; NUMBER OF COLUMNS FOR THIS PRINTER
                        .TITLE  LP
                        .GLOBL  LP
        000000 R0=%0
        000001 R1=%1
        000002 R2=%2
        000003 R3=%3
        000004 R4=%4
        000005 R5=%5
        000006 SP=%6
        000007 PC=%7
                ;  PREAMBLE (FIXED)
000000 000000 LP:       .WORD   0               ; CURRENT DCB OR 0
000002    322           .BYTE   LP.BP           ; FACILITIES INDICATOR
000003    000           .BYTE   0
000004    003           .BYTE   3               ; STD. BUF. / 16
                                                ;   80 CHAR. MAX.
000005    110           .BYTE   LP.INT-LP       ; INTERRUPT ADDRESS
000006    200           .BYTE   200             ; STATUS (PRIORITY 4)
000007    030           .BYTE   LP.OPN-LP       ; OPEN ENTRY
000010    056           .BYTE   LP.TFR-LP       ; TRANSFER (OUT)
000011    030           .BYTE   LP.CLS-LP       ; CLOSE
000012    000           .BYTE   0               ; SPECIAL
000013    000           .BYTE   0               ; SPARE
000014 046600 LP.NAM:   .RAD50  'LP'
                ; MISC. STORES:
000016    014 LP.FRM:   .BYTE   14,0            ; USED FOR OPEN & CLOSE
000017    000
000020 000000 LP.LIN:   .WORD   0
000022 000000 BTCT:     .WORD   0               ; INTERNAL COUNT
000024 000000 BUFAD:    .WORD   0               ; BUFFER POINTER
000026 000000 LP.TAB:   .WORD   0               ; TABULATION COUNT
                ; OPEN & CLOSE ROUTINES:
000030 004567 LP.OPN:   JSR     R5,LP.STS       ; SIMULATE INTERRUPT
       000340
000034 012767           MOV     #1,BTCT         ; SET FOR FF PRINT
       000001
       177750
000042 010767           MOV     PC,BUFAD        ; FROM HEADER
       177756
000046 062767           ADD     #LP.FRM-.,BUFAD
       177750
       177750
000054 000415           BR      LP.INT
                ; TRANSFER ROUTINE:
000056 016700 LP.TFR:   MOV     LP,R0           ; PICK UP DDB
       177716
000062 004567           JSR     R5,LP.STS       ; RUN AT LP STATUS
       000306
000066 016067           MOV     6(R0),BUFAD     ; SAVE BUFFER ADDRESS
       000006
       177730
```

```
000074 016004          MOV     10(R0),R4          ; PRESERVE DDB W.C.
       000010
000100 006304          ASL     R4                 ; CHARACTER COUNT
000102 005404          NEG     R4                 ; MAKE POSITIVE
000104 010467          MOV     R4,BTCT
       177712


000110 042737 LP.INT:  BIC     #100,@#LP.CSR      ; DISABLE INTERRUPT
       000100
       177514
000116 005737          TST     @#LP.CSR           ; CHECK FOR ERROR
       177514
000122 100516          BMI     LP.ERR             ; YES
000124 010146          MOV     R1,-(SP)           ;QUICK SAVE
000126 010246          MOV     R2,-(SP)                   ;  REGS.
000130 016701          MOV     BTCT,R1            ; GET CURRENT BYTE COUNT
       177666
000134 001474          BEQ     LP.DNE             ; NO MORE
000136 016702          MOV     BUFAD,R2           ; GET CURRENT BUF LOC.
       177662
000142 105737 LP.LOP:  TSTB    @#LP.CSR                   ; IS PRINTER GOING
       177514
000146 100055          BPL     LP.STI             ;  YES
000150 121227          CMPB    (R2),#11           ; TAB ?
       000011
000154 001523          BEQ     LP.PTB
000156 121227          CMPB    (R2),#15           ; CARRIAGE RETURN ...
       000015
000162 001416          BEQ     LP.RSC             ; ... RESET COUNTS
000164 105712          TSTB    @R2                ; IGNORE NULL ...
000166 001537          BEQ     LP.DNP
000170 121227          CMPB    @R2,#13            ; VERTICAL TAB ...
       000013
000174 001534          BEQ     LP.DNP
000176 121227          CMPB    @R2,#177           ; ... & RUBOUT
       000177
000202 001531          BEQ     LP.DNP
000204 121227          CMPB    @R2,#12            ; IF LINE TERMINATOR ...
       000012
000210 001403          BEQ     LP.RSC
000212 121227          CMPB    @R2,#14
       000014
000216 001007          BNE     LP.CLO
000220 012767 LP.RSC:  MOV     #9.,LP.TAB         ; ... RESET COUNTS
       000011
       177600
000226 012767          MOV     #PRSIZE+1,LP.LIN
       000121
       177564
000234 000403          BR      LP.TBF             ;... & OMIT NEXT
000236 005767 LP.CLO:  TST     LP.LIN             ; OTHERWISE CHECK LINE OFLO
       177556
000242 001511          BEQ     LP.DNP             ; IGNORE CHAR IF FULL
```

```
000244 112237 LP.TBF: MOVB    (R2)+,@#LP.BUF   ; PRINT CHAR
       177516
000250 005367         DEC     LP.LIN           ; COUNT CHARS. DESPATCHED
       177544
000254 005367         DEC     LP.TAB           ; UPDATE TAB COUNT
       177546
000260 001003         BNE     LP.TRT
000262 012767         MOV     #8.,LP.TAB       ; RESET TAB COUNT
       000010
       177536
000270 005301 LP.TRT: DEC     R1               ; UPDATE COUNT
000272 001323         BNE     LP.LOP   ; MORE
000274 105737         TSTB    @#LP.CSR         ; PRINTER GOING
       177514
000300 100412         BMI     LP.DNE           ; NO, SO NO INTERRUPT
000302 010167 LP.STI: MOV     R1,BTCT          ; SET UP FOR NEXT TIME
       177514
000306 010267         MOV     R2,BUFAD
       177512
000312 052737 LP.TWC: BIS     #100,@#LP.CSR    ; ENABLE INTERRUPT
       000100
       177514
000320 012602         MOV     (SP)+,R2         ; RESTORE REGS
000322 012601         MOV     (SP)+,R1
000324 000002         RTI                      ; THROUGH INTERRUPT


000326 012602 LP.DNE: MOV     (SP)+,R2         ; RESTORE REGS
000330 012601         MOV     (SP)+,R1
000332 013767         MOV     @#LP.SAV,.+10    ; SAVE ALL REGS
       000044
       000002
000340 004537         JSR     R5,@#0
       000000
000344 005037         CLR     @#LP.CSR         ; DISABLE INTERRUPT
       177514
000350 016700 LP.IGN: MOV     LP,R0
       177424
000354 000170         JMP     @14(R0)          ; COMPLETION RETURN
       000014
000360 016746 LP.ERR: MOV     LP.NAM,-(SP)     ; ON ERROR SHOW NAME
       177430
000364 012746         MOV     #402,-(SP)       ; 1-2 ERR MSG
       000402
000370 000004         IOT
000372 000646         BR      LP.INT
```

```
                       ; SUBROUTINE FOR INTERRUPT SIMULATION:
000374  011604 LP.STS: MOV     (SP),R4            ; SIMULATE INTERRUPT
000376  016603         MOV     2(SP),R3
        000002
000402  013766         MOV     @#S.STAT,2(SP)     ; FROM  JSR  PC,XXXX
        177776
        000002
000410  010316         MOV     R3,(SP)
000412  010446         MOV     R4,-(SP)
000414  013737         MOV     @#LP.STV,@#S.STAT  ; RUN UNDER LP STATUS
        000202
        177776
000422  000205         RTS     R5
000424  005767 LP.PTB: TST     LP.TAB             ; AT NEW TAB ALREADY?
        177376
000430  001413         BEQ     LP.EVN
000432  112737 LP.MTB: MOVB    #40,@#LP.BUF       ; SPACE FOR TABS
        000040
        177516
000440  005767         TST     LP.LIN             ; LINE OVERFLOW
        177354
000444  001410         BEQ     LP.DNP             ;  YES, IGNORE REST
000446  005367         DEC     LP.LIN             ; INCLUDE IN LINE COUNT
        177346
000452  005367         DEC     LP.TAB             ;  DONE ?
        177350
000456  001231         BNE     LP.LOP
000460  012767 LP.EVN: MOV     #8.,LP.TAB         ; RESET
        000010
        177340
000466  005202 LP.DNP: INC     R2
000470  000677         BR      LP.TRT
        177514 LP.CSR=177514
        177516 LP.BUF=177516
        000322 LP.BP=322
        000044 LP.SAV=44
        177776 S.STAT=177776
        000202 LP.STV=202
        000030 LP.CLS=LP.OPN
        000001         .END


000000 ERRORS


 BTCT      000022R      BUFAD      000024R      LP        000000RG
 LP.BP   = 000322       LP.BUF  =  177516       LP.CLO    000236R
 LP.CLS  = 000030R      LP.CSR  =  177514       LP.DNE    000326R
 LP.DNP    000466R      LP.ERR     000360R      LP.EVN    000460R
 LP.FRM    000016R      LP.IGN     000350R      LP.INT    000110R
 LP.LIN    000020R      LP.LOP     000142R      LP.MTB    000432R
 LP.NAM    000014R      LP.OPN     000030R      LP.PTB    000424R
 LP.RSC    000220R      LP.SAV  =  000044       LP.STI    000302R
 LP.STS    000374R      LP.STV  =  000202       LP.TAB    000026R
 LP.TBF    000244R      LP.TFR     000056R      LP.TRT    000270R
 LP.TWC    000312R      PC        =%000007      PRSIZE  = 000120
 R0       =%000000      R1        =%000001      R2       =%000002
 R3       =%000003      R4        =%000004      R5       =%000005
 SP       =%000006      S.STAT  =  177776       .        = 000472R
```

DEC-11-ODCRA-A-D

P D P - 1 1

C R 1 1 / C M 1 1   C A R D   R E A D E R   D R I V E R

October 1972

SUPPLEMENT TO:

PDP-11 DEVICE DRIVER PACKAGE

DEC-11-ODDPA-A-D

MONITOR VERSION VØØ8

NOTE

This document is for information purposes only and is subject to change without notice. DEC assumes no responsibility for the use or reliability of its software on equipment which is not supplied by DEC.

NEW AND CHANGED INFORMATION

This manual documents the software as of Monitor version VØ8. It has been revised to include all new and changed material since Monitor version VØ4. Such material is indicated by vertical bars in the outside margin. Whole new pages are not so marked but are dated in the lower outside corner.

pdp11 SOFTWARE DISTRIBUTION CENTER

CONTENTS

CR11/CM11 CARD READER DRIVER

The card-reader driver performs device-dependent I/O functions for the PDP-11 CR11 Card Reader Control within the Disk Operating System (DOS). At each Monitor request on behalf of a running program, the driver, in its basic version, reads a single card, which may be punched in either 026 or 029 Hollerith notation as indicated by specially coded cards in the input deck. The resultant data is stored in a specified area as a line of up to 80 ASCII characters terminated by a carriage-return/line-feed.

By conditional assembly of its source, however, the driver may be produced in various versions to include the following additional features:

- Restriction of input conversions to one type of punch.
- Automatic deletion of card-columns 73-80 and of trailing spaces from preceding columns.
- Reading of cards punched in a binary format with data passed to the user, packed 4 columns to 3 words.
- Provision of similar facilities for the 40-column Mark Sense Reader under CM11 control.

All cards are read under the PDP-11 interrupt system. The driver, therefore, contains the routines needed, firstly, to initiate a card transfer and, secondly, to service the interrupt as each column is read and supply the required conversion of its content until the end of the card is seen. An OPEN function is also included to enable a using program to ensure that the reader is on-line before issuing its first read. CLOSE and Special Functions processing is unnecessary and is not provided.

1.0 BASIC DRIVER (ASCII ONLY) - CR11

The driver is in two parts: the Driver Table and the Service Routines.

1.1 Driver Table

The table occupies the first seven words of the driver in the standard format for I/O drivers under DOS. It includes the following particular information:

- Capabilities: Single user
  Input in ASCII only
  Non-file structured

- Standard buffer size = 96 bytes
- Interrupt servicing at priority level 6
- Device Name: CR

1.2 SERVICE ROUTINES

1.2.1 OPEN

The OPEN routine first checks the Control Status of the reader. If for any reason this is off-line, printing of an AØØ2 error message (device not ready) is requested. If a return is made, the check and message are repeated until an on-line state is detected. The routine then prepares the driver to accept 029 punching by default and returns control to the calling Monitor routine.

1.2.2 TRANSFER

Using the starting address set into its first word by the calling routine, the driver's TRANSFER processor accesses the DDB for the dataset requiring the card input to extract and store internally pointers to the start and end of the buffer area for the data. The first word of the buffer is then cleared as an indicator that the first column is yet to be read. The routine returns to the Monitor with the Reader Control set to INT ENB and GO.

1.2.3 Interrupt Service

At each interrupt, a check is first made for error or card-done conditions. If neither is seen, the column data just accessed is used to compute an index into a table of associated parity-ASCII characters (see Appendix A), the relevant character is extracted and stored as a byte in the buffer provided. The next buffer byte is set to a positive non-zero value to show that a valid read is under way. An interrupt return is then taken.

For card column 1, however, checks are also made for a card with any of the following special codings in that colum:

- 12-2-4-8[1] This indicates that the cards which follow are to be read as 026 punch codes, and on recognition of this an internally stored offset is modified to use the appropriate section of the table of ASCII values.
- 12-0-2-4-6-8[1] This indicates that the following cards contain 029 and cause similar offset modification.

[1]These codes are 12-11-8-9 and 12-0-7-9 in Version ØØ5A Monitor VØØ4 release.

12-11-0-1-6-7-8-9[1] This indicates the end of the card file, and a card so coded must be present ("Hopper Empty" is merely deemed a "Device not ready" state to allow usage of very large decks). When this card is seen, the next buffer byte is set negative to show EOF. Since no data will now be forthcoming, the appropriate word is set in the dataset DDB to show this.

When any of the three cases is seen, the Reader Control Status is reset to EJECT before the interrupt exit is taken, thereby causing the remainder of the card to be ignored.

The rest of a card is similarly ejected if, during the processing of any column, the buffer is filled. In normal READ operations for which the Monitor provides a standard-sized buffer of 96 bytes, this cannot occur. This is not necessarily the case if the user program has requested TRAN. If this program also supplies short buffers, this can mean the possible loss of card data, intentionally or otherwise.

When a card-done condition is detected, the Reader interrupt is disabled. The underway state, shown as noted earlier in the next buffer byte, is then checked. If no data has yet been processed because the card just read merely contained a control code, a new card transfer is requested by recalling the TRANSFER routine. Otherwise, the unused portion of the buffer provided is cleared and the parity-ASCII values for RETURN and LINE FEED are inserted to follow the last data read (in the short buffer, these will overwrite the last two columns processed). As required by the general driver specification, the service routine then saves all user registers on the processor stack and takes the supplied completion return with Register 0 set to the address of the DDB just serviced.

It should be noted that this process allows the reading of only one card at each request, regardless of the size of the buffer provided. Because a card-read (once it has begun) must continue to completion, any attempt to fill the unused buffer space must necessitate the internal storage of any overflow, if possible loss of data is to be avoided. In keeping the size of the driver to a reasonable limit, the provision of such internal storage is not considered desirable. For the READ form of I/O, the buffer supplied by the Monitor must be

[1]This code is 12-11-0-1 in Version ØØ5A Monitor VØØ4 release.

excessive, as space is allocated in 16-word units; the null padding, however, is not passed on to the user program. On the other hand, it can be seen that no advantage is gained by a program defining a buffer larger than 82 bytes when using the device-dependent TRAN.

1.2.4 Error Handling

The detection of any error condition is taken to mean a "Device not ready" state, leading to the printing of an AØØ2 message with the reader interrupt disabled. If the operator requests resumption by a CONTINUE command at the keyboard, the error processor will recall the TRANSFER routine to repeat the read and exit to await a fresh interrupt. This allows the operator to rectify the error, if possible: the card causing the error should be replaced as the first to be read after the resumption.

NOTE

> A "Hopper Empty" condition is detected before the last card has been processed. It is, therefore, essential that the EOF card for a deck be followed by at least one more card (can be blank). Should this be omitted, normal completion can be effected by re-insertion of the EOF card followed by a blank card.

1.3 ALTERNATIVE DRIVERS FOR ASCII ONLY USAGE

As has been shown in the previous section, the standard driver accepts cards punched in either 026 or 029 codes when so directed by control cards, or the driver assumes 029 by default. Unless the user program then requests input by TRAN with short buffer sizes, 80-character lines are the norm. To provide other versions of the driver more suited to the needs of a particular installation, the following conditional assembly parameters have been included in the source language. If these are defined when the source is processed (DEFALT = Ø is sufficient definition), the driver will operate as indicated.

1.3.1 DEFALT

This forces the driver to assume 026 card codes as the default. Control cards as defined, however, will still override this assumption. The effect on the driver length is negligible - one word.

1.3.2 ONLY26/ONLY29

If the user has only one type of punch, he can restrict the driver accordingly by the definition of the relevant one of these parameters.

In this case, control cards will have no effect and will be ignored if present. Because the driver then needs only half of its conversion tables and certain checks can be eliminated, the driver size is reduced by some 45 words.

1.3.3 BLANKS

By common practice, card columns 73-80 are often used only for control information, e.g., sequence numbering, which need not be processed by the using program (initial value of Blank suppress is off). Moreover, quite a number of columns before these may contain nothing but blanks (translated into spaces in ASCII). Although cards of this type will be accepted by systems programs such as Assembler or Editor without error, the burden on lines always 80 characters long can be excessive, especially if, as one example, the only means for listing the assembly of a card source is a teleprinter.

The parameter BLANKS has been included to enable the user optionally to remove this burden, provided that he is also prepared to accept an increase of some 18 words in driver size (initial value of Blank suppress is off). The driver in this case will still continue to transfer 80 characters as its normal operation. If, however, the card deck is preceded by a control card punched 12-11-0-7-8-9 in column 1, or at any point contains a card so punched, columns 73-80 in all subsequent cards will be ignored and the CR/LF terminating the line each card represents will be set immediately after the last non-blank data column. The automatic deletion will remain until the user program requests an OPEN for a fresh deck.

NOTE

> DEFALT, ONLY26, and ONLY29 are of course mutually exclusive. BLANKS, on the other hand, may be defined alone or with any one of the other three.

2.0 READING OF BINARY CARDS

Some users may wish to have the additional facility of reading cards directly as 12 bits per column rather than as ASCII characters, perhaps for one of the following situations:

- Linking or loading of card programs produced by cross-assemblers or linkers developed on other computers.
- Processing of binary data output by other computers.

- Reading of cards using character codes other than those associated with 026 or 029 punches[1].

A further conditional assembly parameter, BINARY, has been included in the driver source to meet this requirement.

2.1 BINARY FORMAT

The driver, assembled with this parameter defined, still continues to function exactly as described earlier whenever the using program requests ASCII input. If, however, a binary transfer is called, the processing Monitor routine will inform the driver of this by setting to 1 bit Ø of the status word of the DDB for the dataset concerned (DDB+12). On recognition of this, the driver accepts each column as 12 data bits and passes it to the program in a packed form, four columns taking three words, in accordance with the following pattern:

	Row 12	Row 9
c.c. 4	11 ------------word 3--------------- 0	
c.c. 3	7 ----word 2-------0	15 word 3 12
c.c. 2	3 word 1 0	15 ----- word 2 ----- 8
c.c. 1	15 ------------word 1--------------- 4	

This format, which is compatible with that used by IBM 1130 and 1800 Series, has been chosen because it alone provides for all the facilities listed above and, moreover, is compatible with the device-independent philosophy of DOS. It may nevertheless mean that the user who needs to process each column on a word basis must include in his program the routine to unpack again the data from the driver (a possible algorithm is offered in Appendix C).

The main effect of the inclusion of the binary capability in the driver is a substantial increase in its size, hence the reason for this not being made a feature of the standard version. Apart from the coding changes need to cater for the different processing (which are outlined in subsequent paragraphs), the assumption made in the case of ASCII data, that null padding in oversize buffers can be safely overlooked, no longer applies. Each buffer word must always be considered

[1]An alternative in this case might be to change the driver's comversion tables to satisfy the different codes, provided that these use the same pattern - null, 0, 11 or 12 in association with one punch in range 1-7, perhaps with a punch in 8 or 9.

as valid data. In order that the driver may cope with this situation (the Monitor can only supply a buffer made up of 16-word units), it must now contain its own internal buffer to hold any residue from a card used to fill the remaining Monitor area. Allowing that such residue can be stored already packed in its final form, the internal buffer must be at least 60 words long. Together with the additional coding, the driver increases from its 200 words, in the basic ASCII-only form, to 380. Against this, however, the presence of the internal buffer also means that the driver can supply valid binary data into user buffers of any length for a program issuing TRAN requests.

It should be noted, too, that the format used does not in itself provide any checking upon the read accuracy of each card. All 80 columns are assumed to contain actual data. Programs READing in unformatted binary modes or using TRAN must make their own checks if these are important, in just the same way as with other drivers. On the other hand, the Monitor processing formatted binary READs will expect the data to conform to its normal standards for each request.

```
Byte Ø:     =1
Byte 1:     =Ø
Bytes 2-3:  Number of bytes to be read including bytes
            Ø-3 but not the final checksum.
Bytes 4-N:  DATA BYTES
Byte N+1:   Checksum of Bytes Ø-N
```

If program developed to produce binary cards in such format also punch one card for each READ, the data checksum can serve as card checksum as well (in this case, nulls following the checksum will be ignored).

2.2 CODING CHANGES FOR BINARY OPERATIONS

The changes in the driver's operations brought about by the definition of BINARY are as follows:

2.2.1 DRIVER Table

Capabilities as indicated in Section 1.1 are extended to include binary input.

2.2.2 OPEN

The OPEN processor still first checks the on-line state of the reader and takes appropriate action as described earlier if it is not ready. It now, however, anticipates the fact that after its exit, the driver will be recalled immediately to fill the Monitor buffer against the program's first READ. At this time the Monitor will be unable to direct the driver on the mode of reading. The OPEN routine therefore sets a switch to cause a return to be made without a card transfer, when this recall occurs. It also means that the Monitor will give the user program 96 bytes of null (equivalent to leader on a paper tape) for its first unformatted binary operations. (Incidentally, the switch is set to perform a proper read when the driver is loaded into memory; if, therefore, the program does not request an OPEN but starts by a READ, the correct result will occur.)

2.3 TRANSFER

As with all drivers, the card reader driver must contain only position-independent code. To control its internal buffer, however, it needs absolute pointers. The first actual read causes execution of some once-only code to establish these. Again, a switch effects this. A further switch is then set according to the mode in which the data is to be handled. As mentioned earlier, if this is ASCII, the code for the standard version of the driver is followed, both during the TRANSFER and INTERRUPT service functions.

For binary transfers, any data remaining from a previous read is passed to the Monitor buffer immediately. If this is sufficient to satisfy the Monitor's requirements, an immediately completion return is taken. (Since this would normally follow an interrupt and the Monitor will expect this, the driver must in this case simulate the appropriate conditions, i.e., leave an interrupt exit on the stack, supposedly preceded by saved registers.) A new card read is initiated in the same way as ASCII if more data is needed. In addition, the second byte in the Monitor buffer not yet filled is cleared as a switch for use by the packing algorithm which handles odd and even card columns differently (see Appendix B).

2.4 INTERRUPT SERVICING

The packing of binary data is accomplished as each column is read. At the beginning of each card a check is again made for EOF. Unlike the ASCII case, the coding of a single column cannot provide a unique identification. The same pattern (12-11-0-1-6-7-8-9[1]) is therefore looked for in each of the first eight columns before the end-of-data is signalled and the remainder of the card ejected. (The same card can still be used for either data form; the ASCII processor merely uses the first column punched.) No other control cards are expected in binary mode.

[1]This was code 12-11-0-1 in Version ØØ5A Monitor VØØ4A release.

After the entire card has been read as indicated by a card-done condition, the TRANSFER routine is recalled to continue its process of transferring the data into the Monitor buffer. Since an interrupt has no occurred, the return to the Monitor on completion is by normal means.

2.5 ERROR HANDLING

Any error condition is again considered a "device not ready" and is handled accordingly. Because a repeat of the TRANSFER routine as a way of resuming read operations on return would perhaps lose data already passed to the Monitor from a previous card, a failure in binary mode leads only to that part of the TRANSFER operations which causes a new card read.

3.0 CM11 MARK SENSE READER DRIVER

The CM11 Control is expected to use only 40-column cards. (The normal CR11 driver with or without definition of the special assembly parameters will function without change if 80-column cards are used.) To provide the following particular benefits in view of the smaller amount of data available at each card read, one more parameter for conditional assembly has been included - MARKS:

- Standard Buffer size = 64 bytes rather than 96.
- Internal buffer for binary operation is reduced from 60 to 30 words.
- If BLANKS has been defined, automatic deletion of trailing spaces will follow recognition of the relevant control card but not of the last 8 columns.

4.0 DETAILED IMPLEMENTATION

Comments on the listing which follow illustrate the general form of the driver. Further explanation of some of the more obscure techniques used is given in Appendix B. Other appendices summarize the ASCII/Hollerith equivalences, the procedures for obtaining the various versions of the driver, and the comparative sizes of each.

```
;COPYRIGHT 1971,1972, DIGITAL EQUIPMENT CORP., MAYNARD, MASS.
;VERSION NUMBER:         V006A
;
;CARD READER DRIVER (CR)
;
;   A)    FOR ASCII INPUT, AT EACH TRANSFER REQUEST
;         ONE CARD WILL BE READ, UP TO 80 CHARACTERS,
;         FOLLOWED BY CR-LF, WILL BE PASSED TO THE
;         CALLING ROUTINE AS SPECIFIED BY THE WORD
;         COUNT GIVEN, (IF THIS IS > 41, REMAINING
;         BYTES WILL BE CLEARED.
;         ALL ERRORS (INCLUDING 'HOPPER EMPTY' UPON
;         AN 'OPEN' CALL) WILL BE TREATED AS 'DEVICE
;         NOT READY', USER CAN RESUME OPERATION BY
;         RECTIFICATION OF ERROR OR REFILL OF HOPPER
;         AND ENTRY OF 'CO' COMMAND AT KEYBOARD.
;         THE END OF A FILE WILL BE DETERMINED BY
;         RECOGNITION OF A TERMINAL CONTROL CARD:-
;                 12-11-0-1-6-7-8-9 PUNCHED IN C.C. 1
;
;                 XXXXXXXXXXXXXXXXXXXXXXXXX
;                 X                       X
;                 X          NOTES        X
;                 X                       X
;                 XXXXXXXXXXXXXXXXXXXXXXXXX
;
;   1)    THIS DRIVER CAN BE ASSEMBLED FOR USE
;         IN CONNECTION WITH EITHER '026' OR '029'
;         PUNCHES OR BOTH AS INDICATED BY PARAMETER
;         SPECIFICATION AT START OF SOURCE INPUT
;         AS FOLLOWS:-
;         A)  "ONLY26=0"   -  READ ONLY '026' CODES.
;         B)  "ONLY29=0"   -  READ ONLY '029' CODES
;         C)  "DEFALT=0"   -  READ BOTH TYPES OF CODE
;                             WITH '026' AS DEFAULT
;         D)    NIL        -  READ BOTH TYPES OF CODE
;                             WITH '029' AS DEFAULT
;
;         IN CASES (C) & (D), DRIVER WILL USE DEFAULT
;         UNLESS DIRECTED OTHERWISE BY ENTRY OF A
;         CONTROL CARD PUNCHED IN C.C. 1:-
;
;                 12-0-2-4-6-8   =   '029' CODES FOLLOW
;                 12-2-4-8       =   '026' CODES FOLLOW
;
;   2)    IF PARAMETER "BLANKS" IS DEFINED, C.C. 73-80
;         & TRAILING SPACES BEFORE THESE WILL BE DISCARDED,
;         WITH 'CR-LF' FOLLOWING LAST VALID DATA, PROVIDED
;         THAT CARD FILE IS PRECEDED BY CTL CARD WITH
;         12-11-0-7-8-9 PUNCHED IN C.C. 1. IN THIS CASE
;         HOWEVER, IF THE USER BUFFER IS </= 82 BYTES,
;         ONLY TRAILING SPACE REMOVAL WILL BE EFFECTED.
```

```
        ;  B)   IF THE PARAMETER "BINARY" IS DEFINED AT ASSEMBLY,
        ;       XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
        ;
        ;       THIS VERSION WILL ALSO ALLOW READING OF CARDS
        ;       IN BINARY FORMAT, AS EACH CARD IS READ, 12 BITS FROM
        ;       ALL 80 COLS WILL BE ACCESSED & STORED IN AN INTERNAL
        ;       BUFFER IN A PACKED FORM, I.E. 4 COLS = 3 WORDS AS
        ;       FOLLOWS:-
        ;               C.C.1 > WORD 1, BITS 15-4
        ;               C.C.2 > WORD 1, BITS 3-0; WORD 2, BITS 15-8
        ;               C.C.3 > WORD 2, BITS 7-0; WORD 3, BITS 15-12
        ;               C.C.4 > WORD 3, BITS 11-0
        ;
        ;       THE PACKED FORM WILL BE TRANSFERRED TO THE USER
        ;       BUFFER UNTIL THIS IS FILLED, ANY DATA THEN REMAINING
        ;       IN THE INTERNAL BUFFER BEING RETAINED UNTIL THE
        ;       NEXT READ REQUEST.
        ;
        ;       TREATMENT OF ASCII READING WILL STILL FOLLOW PATTERN
        ;       DESCRIBED IN THE PREVIOUS PARAGRAPH WITHOUT EXCEPTION.
        ;
        ;       THE ONLY CONTROL CARD WHICH WILL HAVE ANY EFFECT
        ;       IN BINARY READING WILL BE THAT INDICATING E.O.F. (IN
        ;       THIS CASE 12-11-0-1-6-7-8-9 PUNCHING MUST APPEAR IN AT
        ;       LEAST C.C. 1 THRU 8).
        ;
        ;       N.B. WHEN ASSEMBLED FOR USAGE IN BOTH MODES, AN 'OPEN'
        ;       CALL WILL NOT CAUSE READING OF A CARD TO ALLOW THIS
        ;       TO BE TRANSLATED AS REQUIRED BY THE READ MODE
        ;       SPECIFIED BY THE USER.
        ;
        ;  C)   DRIVER CAN ALSO BE USED FOR 80-COLUMN MARK
        ;       SENSE READER. FOR 40 -COLUMN READER, ECONOMIES
        ;       IN BUFFER SIZE CAN BE OBTAINED BY DEFINITION
        ;       AT ASSEMLBY OF PARAMETER "MARKS". THIS WILL
        ;       ALSO PREVENT AUTOMATIC REMOVAL OF COLS 33-40
        ;       IN 'BLANKS-SUPPRESS' MODE OF USAGE.
```

```
                      ;
                      ;
                      ;                   XXXXXXXXXXXXXXXXXXXXXXXXX
                      ;                   X                       X
                      ;                   X         NOTE          X
                      ;                   X                       X
                      ;                   XXXXXXXXXXXXXXXXXXXXXXXXX
                      ;
                      ;           PARAMETER DEFINITIONS CAN BE MADE DURING PASS 1
                      ;           OF THE ASSEMBLY ONLY IF REQD AS DESCRIBED IN
                      ;           PAL-11R MANUAL, SECTION 9-2, E.G.
                      ;
                      ;                   *CR.LP:,/PA:2<KB:/PA:1,DF:CR
                      ;           FOLLOWED BY:-
                      ;                   ONLY29=0
                      ;                   BINARY=0
                      ;                   BLANKS=0
                      ;                   ^C
                      ;                   .END<CR><LF>
                      ;
                      ;                   XXXXXXXXXXXXXXXXXXXXXXXX
                      ;
                                  .TITLE DV.CR
                                  .GLOBL CR
                      ;
             000000 R0=%0
             000001 R1=%1
             000002 R2=%2
             000003 R3=%3
             000004 R4=%4
             000005 R5=%5
             000006 SP=%6
             000007 PC=%7
                      ;INTERFACE TABLE:
      00000 000000 CR:    .WORD 0                 ;CURRENT DDB OR 0 IF IDLE
                          .IFNDF   BINARY
      00002    224        .BYTE    224,0          ;FACILITIES: ASCII INPUT,OPEN IN
      00003    000
                          .ENDC
                          .IFDF    BINARY
                          .BYTE    234,0          ;ALLOW BINARY IF REQD.
                          .ENDC
                          .IFNDF   MARKS
      00004    003        .BYTE    3              ;STD BUFFER SIZE = 96 BYTES
                          .ENDC
                          .IFDF    MARKS
                          .BYTE    2              ;(64 IF 40-COL MARK SENSE)
                          .ENDC
      00005    106        .BYTE    CR.INT-CR,300  ;INTERUPT SVCE AT PRL 6
      00006    300
      00007    022        .BYTE    CR.OPN-CR      ;OFFSET TO OPEN
      00010    044        .BYTE    CR.TFR-CR      ;OFFSET TO TRANSFER
      00011    000        .BYTE    0,0,0          ;(NO CLOSE OR SPEC. FUNC.)
      00012    000
      00013    000
      00014 012620 CR.NAM: .RAD50  'CR'
```

```
                           ;OPEN PROCESSOR:
  000016 004767 CR.ONR: JSR      PC,CR.NRY
         000514
  000022 032737 CR.OPN: BIT      #400,@#CR.CSR    ;CARDS IN HOPPER?
         000400
         177160
  000030 001372         BNE      CR.ONR           ;IF NOT TELL USER TO READY
                        .IFNDF   ONLY26&ONLY29
                        .IFNDF   DEFALT           ;FOR DUAL-PUNCH DRIVER ...
  000032 005027         CLR      (PC)+            ;... SET CONV. TABLE OFFSET
                        .ENDC                     ;... FOR DEFAULT PUNCH
                        .IFDF    DEFALT           ;... AS APPROPRIATE
                        MOV      #126,(PC)+       ;OFFSET TO 026 CONVERSION TABLE
                        .ENDC
  00034 000000  CR.TOS: .WORD    0
                        .ENDC
                        .IFDF    BLANKS           ;IF BLANK-SUPPRESS VERSION ...
                        CLRB     CR.ZSW           ;... FORCE SUPPRESS OFF
                        .ENDC
                        .IFNDF   BINARY
  00036 005726          TST      (SP)+            ;IGNORE INTERIM RETURN
  00040 000167          JMP      CR.DXT           ;... & TAKE COMPLETION
         000422
                        .ENDC
                        .IFDF    BINARY           ;FOR BINARY VERSION ...
                        CLRB     CR.TFR           ;... FORCE NO OPEN READ
                        BR       CR.ODN
                CR.DXT: INCB     CR.TFR           ;... BY MAKING COME HERE
                        BR       CR.ODN           ;... BEFORE EXIT
                        .ENDC
```

```
                        ;SUBSIDIARY ROUTINES:
                        ;   A)  RESTART AFTER ERROR:
  000044                CR.AGN: .IFDF   BINARY          ;IN BINARY VERSION ...
                                TSTB    CR.ISW          ;... CHECK IF BINARY READ
                                BEQ     CR.IER          ;IF NOT CAN JUST START OVER
                                BR      CR.ERD          ;ELSE LEAVE USER BUFFER ALONE
                        ;   B)  INITIALISE INTERNAL BUFFER POINTERS:
                        CR.ISP: MOV     PC,-(SP)        ;GET BUFFER START
                                ADD     #CR.BUF-.,@SP
                                MOV     @SP,(PC)+
                        CR.IBS: .WORD   0
                                ADD     #CR.BSZ,@SP     ;NOW GET END
                                MOV     @SP,(PC)+       ;STORE AS CONTROL
                        CR.IBE: .WORD   0
                                MOV     (SP)+,(PC)+     ;... & AS INIT. PTR
                        CR.IBP: .WORD   0
                                INCB    @PC             ;MUSTN'T COME HERE AGAIN!
                                .ENDC
```

```
                        ;TRANSFER SET-UP PROCESSOR:
 000044                 CR.TFR: .IFDF   BINARY          ;FOR BINARY VERSION ...
                                BR      .+4             ;... SWITCH TABLE FORCES ...
                                BR      CR.CXT          ;... CORRECT INIT.
                                BR      CR.ISP
                                .ENDC
 000044 016700                  MOV     CR,R0           ;GET DCR ADDRESS
        177730
 000050 062700                  ADD     #6,R0           ;... & MOVE TO BUFFER STORE
        000006
 000054 012046                  MOV     (R0)+,-(SP)     ;GET BUFFER POINTER
 00056 011646                   MOV     @SP,-(SP)       ;... & BUILD BUFF END
 00060 161016                   SUB     @R0,@SP
 00062 162016                   SUB     (R0)+,@SP
 00064 012627                   MOV     (SP)+,(PC)+     ;SAVE RESULT ...
 00066 000000           CR.UBE: .WORD   0
                                .IFDF   BINARY          ;IN BINARY VERSION ...
                                MOVB    @R0,R0          ;... CHECK MODE
                                BICB    #376,R0
                                MOVB    R0,CR.ISW       ;IF ASCII CLEAR SWITCH
                                BEQ     CR.RDC          ;IF BINARY WANTED ...
                                MOV     (SP)+,R0        ;... SET PTRS & SWITCH
                                MOV     CR.IBP,R1
                                CLR     R2              ;SET INTERRUPT FLAG
                        CR.BIN: CMP     R1,CR.IBE       ;INTERNAL BUFF EMPTY?
                                BNE     CR.BLP
                        CR.ERD: MOV     CR.IBS,CR.IBP   ;IF SO RESET INTERNAL PTR
                                MOV     R0,-(SP)        ;SAVE USER BUFF PTR ...
                        CR.RDC: .ENDC
 00070 011627                   MOV     @SP,(PC)+
 00072 000000           CR.UBP: .WORD   0
 00074 005036                   CLR     @(SP)+          ;ZERO UNDERWAY FLAG
 00076 012737                   MOV     #101,@#CR.CSR   ;ENABLE INT & GO FOR CD READ
       000101
       177160
 00104 000207                   RTS     PC              ;RETURN USER FOR NOW

                                .IFDF   BINARY          ;WITH BINARY DATA ...
                        CR.BLP: SWAB    @R1             ;... COMPLETE CONVERSION
                                MOV     (R1)+,(R0)+     ;... & GIVE TO USER
                                CMP     R0,CR.UBE       ;USER BUFFER FULL?
                                BNE     CR.BIN          ;IF NOT GET NEXT WORD
                                MOV     R1,CR.IBP       ;OTHERWISE SAVE INT PTR
                                TST     R2              ;COME HERE ON INTERRUPT?
                                BNE     CR.ODN          ;IF SO MODE SW. SET
                                MOV     @SP,-(SP)       ;ELSE MUST SIMULATE ...
                                MOV     -(R2),2(SP)     ;... STORE PC & PS
                                SUB     #16,SP          ;... & DUMMY SAVE REGS.
                        CR.ODN: TST     (SP)+           ;IGNORE RETURN PC
                                JMP     CR.DXT          ;... & TAKE COMPLETION EXIT
                                .ENDC
```

```
                          ;INTERRUPT SERVICE ROUTINES:
                          ;  A)  CHECK FOR ERROR & COLLECT INPUT:
ØØØ1Ø6 Ø1ØØ46  CR.INT:  MOV     RØ,-(SP)          ;SAVE USER RØ
ØØØ11Ø Ø167ØØ           MOV     CR.UBP,RØ         ;GET USER BUFF PTR ...
       177756
ØØØ114 Ø13746           MOV     @#CR.CSR,-(SP)    ;... & READER STATUS
       17716Ø
ØØØ12Ø ØØ6326           ASL     (SP)+             ;CHECK FOR SPECIAL CASES
ØØØ122 1Ø3ØØ2           BCC     .+6
ØØØ124 ØØØ167           JMP     CR.ERR            ;GO RETRY IF ERROR
       ØØØ326
ØØØ13Ø 1ØØ523           BMI     CR.DUN            ;CLEAN UP IF DONE
ØØØ132 Ø1Ø146           MOV     R1,-(SP)          ;NOW SAVE USER R1
                        .IFDF   BINARY            ;IN BINARY VERSION ...
               CR.ISW:  BR      .+4               ;... USE APPROPRIATE CONVERSION
                        BR      CR.ASC
                          ;  B)  BINARY CONVERT & STORE:
                        MOV     CR.IBP,R1         ;GET INT BUFF PTR
                        MOV     @#CR.DB1,-(SP)    ;... & INPUT
                        MOV     @SP,-(SP)         ;... 2 COPIES FOR LATER
                        COMB    1(RØ)             ;ODD COLUMN?
                        BPL     CR.BST
                        ASL     @SP               ;IF SO SHIFT INPUT TO HIGH
                        ASL     @SP
                        ASL     @SP
                        ASL     @SP
                        CLRB    (R1)+             ;MAKE NXT INSTR = MOVB
               CR.BST:  BISB    1(SP),-1(R1)      ;SET HIGH BYTE AS REQD,
                        MOVB    (SP)+,(R1)+       ;THEN LOW BYTE
                        MOV     R1,CR.IBP         ;SAVE PTR
                        CMP     (SP)+,#7417       ;NOW LOOK FOR EOF CARD
                        BNE     CR.BXT
                        DECB    @RØ               ;... PUNCHED 12-11-Ø-1
                        BPL     CR.BXT            ;... IN CC 1 THRU 8
                        ASLB    @RØ
                        BMI     CR.CXT            ;IF NOT END TRY NEXT TIME
                        JMP     CR.EOF            ;OTHERWISE IGNORE REST OF CARD
                        .ENDC
```

```
                        ;  C)  ASCII CONVERT & STORE:
                        ;      IF AT COL 1 WE MUST READ THE BINARY BUFFER, CHECK FOR
                        ;      ONE OF SEVERAL CONTROL CARDS, AND CONVERT BACK TO ASCII.
                        ;
000134 105710  CR.ASC:  TSTB    @R0                     ;CHECK FOR COL 1
000136 001403           BEQ     CR.C1
000140 113701           MOVB    @#CR.DB2,R1             ;GET ASCII VERSION
       177164
000144 000451           BR      CR.CVT                  ;. . . AND GO CONVERT
000146 013701  CR.C1:   MOV     @#CR.DB1,R1             ;GET CHARACTER IN BINARY FORM
       177162
000152 020127           CMP     R1,#EOF                 ;CHECK FOR EOF CARD
       007417
000156 001002           BNE     .+6
000160 000167           JMP     CR.EOF
       000310
                        .IFDF   BLANKS                  ;FOR BLANK SUPPRESS ...
                        CMP     R1,#BSUP                ;... LOOK FOR SUPPRESS ON
                        BNE     .+6
                        JMP     CR.ZON
                        .ENDC
                        .IFNDF  ONLY26&ONLY29;FOR DUAL PUNCH DRIVER ...
000164 020127           CMP     R1,#SET29               ;... CHECK IF 029 CTL
       005252
000170 001002           BNE     .+6
000172 000167           JMP     CR.029
       000332
000176 020127           CMP     R1,#SET26               ;... OR 026 CTL
       004242
000202 001002           BNE     .+6
000204 000167           JMP     CR.026
       000310
                        .ENDC

                        ;       NOW WE KNOW THAT WE HAVE A NORMAL ASCII CHARACTER BUT
                        ;       IT IS IN BINARY MODE, CONVERT AS FOLLOWS:
                        ;               BITS    GO TO
                        ;               11-9    7-5
                        ;               1       3
                        ;               0       4
                        ;               8-2     2-0  NOTE THAT THIS ONE WONT FIT,
                        ;                               SEE PERIPHERALS MANUAL FOR THE
                        ;                               CORRECT ALGORITHM.
                        ;                               THIS ALGORITHM WORKS FOR
                        ;                               NON-MULTI-PUNCHED CARDS.
                        ;                               (IE, ALL LEGAL CHARACTER CODES)

000210 010246           MOV     R2,-(SP)                ;SAVE USER REGISTER 2
000212 010102           MOV     R1,R2
000214 042701           BIC     #170774,R1              ;CLEAR ALL BUT 11-9, 0, 1
       170774
000220 006201           ASR     R1                      ;OLD BIT 0 => C BIT IN STATUS
000222 103002           BCC     .+6
000224 052701           BIS     #200,R1                 ;WILL BE BIT 4 IN FINAL WORD
       000200
000230 006201           ASR     R1                      ;OLD BIT 1 => C BIT
000232 103002           BCC     .+6
```

```
   00234 052701            BIS     #40,R1          ;WILL BE BIT 3
         000040
   00240 006201            ASR     R1
   00242 006201            ASR     R1              ;11-9 NOW IN 7-5, 3-4 ALSO SET
   00244 032702            BIT     #774,R2         ;CHECK 8-2
         000774
   00250 001405            BEQ     CR.CVD          ;ALL DONE IF ZERO
   00252 006202            ASR     R2
   00254 000302            SWAB    R2              ;OLD 8-2 NOW IN 15-9
   00256 005201  CR.CVL:   INC     R1              ;ALL THIS JUST TO BE COMPATABLE
   00260 006302            ASL     R2              ;WITH THE 10
   00262 103375            BCC     CR.CVL
   00264 012602  CR.CVD:   MOV     (SP)+,R2        ;RESTORE USER R2
   00266 110101            MOVB    R1,R1           ;SIGN EXTEND IF BIT 7 ON.
   00270 012746  CR.CVT:   MOV     #104,-(SP)      ;SET INDEX FOR SPECIAL CHARACTER
         000104
   00274 120127            CMPB    R1,#240         ;TEST IF RPG SPECIAL
         000240
   00300 001422            BEQ     CR.RPG
   00302 005216            INC     (SP)
   00304 120127            CMPB    R1,#140         ;TEST IF RPG SPECIAL
         000140
   00310 001416            BEQ     CR.RPG
   00312 005726            TST     (SP)+
   00314 105701            TSTB    R1
   00316 100002            BPL     .+6             ;CONVERT CODES >200 ...
   00320 062701            ADD     #340,R1         ;... TO RANGE >140
         000340
   00324 010146            MOV     R1,-(SP)        ;CONVERT CARD CODE ...
   00326 162701  CR.SUB:   SUB     #40,R1          ;FOR EACH 40 IN CODE ...
         000040
   00332 100403            BMI     CR.STO          ;... ADD 21 & STRIP 40
   00334 162716            SUB     #17,@SP         ;... TO GET TABLE INDEX
         000017
   00340 000772            BR      CR.SUB
   00342         CR.STO:   .IFNDF  ONLY26&ONLY29
   00342 066716            ADD     CR.TOS,@SP      ;PICK APPROP. TABLE
         177466
                           .ENDC
   00346 060716  CR.RPG:   ADD     PC,@SP          ;COMPUTE ADDR OF BYTE REQD
   00350 062716            ADD     #CR.TBL-.,@SP
         000202
   00354 113620            MOVB    @(SP)+,(R0)+    ;... & STORE IN BUFFER
   00356 020067            CMP     R0,CR.UBE       ;BUFFER FULL?
         177504
   00362 001452            BEQ     CR.EXT
   00364 111710  CR.BXT:   MOVB    @PC,@R0         ;IF NOT SET UNDERWAY FLAG
   00366 010067  CR.CXT:   MOV     R0,CR.UBP       ;SAVE NEW POINTER
         177500
   00372 012601            MOV     (SP)+,R1        ;RESTORE USER REGS.
   00374 012600  CR.IXT:   MOV     (SP)+,R0
   00376 000002            RTI                     ;... & EXIT
```

```
                      ;  D)  CARD COMPLETED:
000400 105037 CR.DUN: CLRB    @#CR.CSR          ;STOP INTERRUPTS
       177160
000404 105710         TSTB    @R0               ;IF NO PROCESSING YET ...
000406 001427         BEQ     CR.RPT            ;... CONTINUE
000410 012600         MOV     (SP)+,R0          ;OTHERWISE RESTORE USER R0
000412 013746         MOV     @#CR.RSV,-(SP)    ;... & NOW SAVE ALL
       000044
000416 004536         JSR     R5,@(SP)+
000420 016700         MOV     CR.UBP,R0         ;SET USER BUFF PTR
       177446
                      .IFDF   BINARY            ;FOR BINARY VERSION ...
                      TSTB    CR.TSW            ;... CHECK IF BINARY READ
                      BNE     CR.BDN            ;IF SO ACTION ACCRODINGLY
                      .ENDC
 00424 016701         MOV     CR.UBE,R1         ;FOR ASCII, SET END PTR
       177436
                      .IFDF   BLANKS            ;... & PERHAPS CHECK SUPPRESS
              CR.ZSW: BR      .+2               ;SWITCH ON?
                      BR      CR.ADN            ;IF NOT NO SUPPRESSION
                      .IFNDF  MARKS
                      TSTB    @R0               ;TEST IF END OF FILE
                      BMI     CR.DXT            ;SKIP NEXT CALCULATION IF EOF
                      CMP     R0,R1             ;IF BUFFER FULL OMIT NEXT
                      BEQ     .+6
                      SUB     #8.,R0            ;OTHERWISE LOSE CC 73-80
                      .ENDC
                      CMPB    -(R0),#240        ;THEN TRAILING SPACES
                      BEQ     .-4
                      TSTB    (R0)+             ;ADJUST PTR WHEN DONE
              CR.ADN: .ENDC
 00430 105041         CLRB    -(R1)             ;CLEAR REST OF BUFFER
 00432 020100         CMP     R1,R0
 00434 101375         BHI     .-4
 00436 112721         MOVB    #215,(R1)+        ;MOVE IN CARRIAGE RETURN
       000215
 00442 112721         MOVB    #012,(R1)+        ;MOVE IN LINE FEED
       000012
 00446 016700 CR.DXT: MOV     CR,R0             ;GET DDB ADDRESS
       177326
 00452 000172         JMP     @14(R0)           ;TAKE COMPLETION EXIT
       000014
                      .IFDF   BINARY
              CR.BDN: MOV     CR.IBS,R1         ;FOR BINARY, INIT INT PTR
                      MOVB    @R0,R2            ;EXIT IF EOF SEEN
                      BMI     CR.DXT            ;ALSO SETS INTERRUPT FLAG
                      JSR     PC,CR.BIN         ;ELSE GO MOVE DATA TO USER
                      MOV     @#CR.SXT,R5       ;IF COME BACK, MORE READ REQD.
                      JMP     4(R5)             ;SO TAKE SYSTEM EXIT
                      .ENDC
```

```
                          ;SPECIAL CASE PROCESSING:
                          ;  A)  ERROR ROUTINE:
ØØØ456 1Ø5Ø37  CR.ERR: CLRB     @#CR.CSR          ;STOP INTERRUPTS
       17716Ø
ØØØ462 ØØ4767          JSR      PC,CR.NRY         ;INFORM OPERATOR
       ØØØØ5Ø
ØØØ466 ØØ4767  CR.RPT: JSR      PC,CR.AGN         ;IF RETURN TRY AGAIN
       177352
ØØØ472 ØØØ74Ø          BR       CR.IXT            ;... & EXIT FOR NOW
                          ;  B)  END OF FILE CARD SEEN:
ØØØ474 Ø167Ø1  CR.EOF: MOV      CR,R1             ;GET DDB ADDRESS
       1773ØØ
                       .IFNDF   BINARY            ;FOR SIMPLE VERSION ...
ØØ5ØØ Ø16161           MOV      1Ø(R1),16(R1)     ;... NO DATA READ ON EOF
       ØØØØ1Ø
       ØØØØ16
                       .ENDC
                       .IFDF    BINARY            ;MAYBE SOME IF BINARY ...
                       ADD      #16,R1            ;SO MOVE TO UNUSED COUNT STORE
                       MOV      RØ,@R1            ;... & COMPUTE VALUE REQD
                       SUB      CR.UBF,@R1
                       ASR      @R1               ;... AS WORDS!
                       .ENDC
ØØ5Ø6 ØØ5112           COM      @R2               ;SET FLAG
ØØ51Ø 152737  CR.EXT: BISB      #2,@#CR.CSR       ;ALLOW REST OF CARD THRU
       ØØØØØ2
       17716Ø
ØØ516 ØØØ723           BR       CR.XT
                          ;  C)  CONTROL CARD SEEN:
                       .IFNDF   ONLY26&ONLY29     ;FOR DUAL PUNCH DRIVER ...
ØØ52Ø Ø12767  CR.Ø26: MOV       #1Ø6,CR.TOS       ;... SET TABLE OFFSET ...
       ØØØ1Ø6
       1773Ø6
ØØ526 ØØØ77Ø           BR       CR.EXT            ;... & IGNORE REST OF CTL CARD
ØØ53Ø ØØ5Ø67  CR.Ø29: CLR       CR.TOS
       1773ØØ
ØØ534 ØØØ765           BR       CR.EXT
                       .ENDC
                       .IFDF    BLANKS            ;IN SUPPRESS VERSION
              CR.ZON: MOVB      #1,CR.ZSW         ;...SET SUPPRESS ON
                       BR       CR.EXT            ;AGAIN IGNORE REST OF CARD
                       .ENDC
```

```
                        ;READER NOT READY SUBROUTINE:
  ØØØ536 Ø16746 CR.NRY: MOV     CR.NAM,-(SP)    ;IDENTIFY DEVICE
         177252
  ØØØ542 Ø12746         MOV     #4Ø2,-(SP)      ;GIVE NOT READY CODE
         ØØØ4Ø2
  ØØØ546 ØØØØØ4         IOT                     ;... & CALL EDP
  ØØØ55Ø ØØØ2Ø7         RTS     PC              ;TRY AGAIN IF COME BACK
                        ;
                        ;MISCELLANEOUS DEFINITIONS:
         17716Ø CR.CSR=17716Ø
         177162 CR.DB1=177162
         177164 CR.DB2=177164
         ØØØØ42 CR.SXT=42
         ØØØØ44 CR.RSV=44
         ØØ7417 EOF=ØØ7417                      ;12-11-Ø-1-6-7-8-9 PUNCH
         ØØ4242 SET26=ØØ4242                    ;12-2-4-8 PUNCH
         ØØ5252 SET29=ØØ5252                    ;12-Ø-2-4-6-8 PUNCH
         ØØ7ØØ7 BSUP=ØØ7ØØ7                     ;12-11-Ø-7-8-9 PUNCH
```

```
000552                    CR.TBL:
                          ;PARITY ASCII CONVERSION TABLE FOR 029 PUNCH
                                  .IFNDF  ONLY26
000552         240                .BYTE 240                ;SPACE
000553         261                .BYTE 261                ;1
000554         262                .BYTE 262                ;;2
000555         063                .BYTE 63                 ;3
000556         264                .BYTE 264                ;4
000557         065                .BYTE 65                 ;5
 00560         066                .BYTE 66                 ;6
 00561         267                .BYTE 267                ;7
 00562         270                .BYTE 270                ;8
 00563         240                .BYTE 240                ;EMPTY
 00564         072                .BYTE 72                 ;:
 00565         243                .BYTE 243                ;#
 00566         300                .BYTE 300                ;@
 00567         047                .BYTE 47                 ;'
 00570         275                .BYTE 275                ;=
 00571         042                .BYTE 42                 ;"
 00572         071                .BYTE 71                 ;9
                          ;
 00573         060                .BYTE 60                 ;0
 00574         257                .BYTE 257                ;/
 00575         123                .BYTE 123                ;S
 00576         324                .BYTE 324                ;T
 00577         125                .BYTE 125                ;U
 00600         126                .BYTE 126                ;V
 00601         327                .BYTE 327                ;W
 00602         330                .BYTE 330                ;X
 00603         131                .BYTE 131                ;Y
 00604         240                .BYTE 240                ;EMPTY
 00605         134                .BYTE 134                ;\
 00606         254                .BYTE 254                ;,
 00607         245                .BYTE 245                ;%
 00610         137                .BYTE 137                ;_
 00611         276                .BYTE 276                ;>
 00612         077                .BYTE 77                 ;?
 00613         132                .BYTE 132                ;Z
                          ;
```

```
000614    055        .BYTE 55         ;-
000615    312        .BYTE 312        ;J
000616    113        .BYTE 113        ;K
000617    314        .BYTE 314        ;L
000620    115        .BYTE 115        ;M
000621    116        .BYTE 116        ;N
000622    317        .BYTE 317        ;O
000623    120        .BYTE 120        ;P
000624    321        .BYTE 321        ;Q
 00625    240        .BYTE 240        ;EMPTY
 00626    335        .BYTE 335        ;]
 00627    044        .BYTE 44         ;$
 00630    252        .BYTE 252        ;*
 00631    251        .BYTE 251        ;)
 00632    273        .BYTE 273        ;;
 00633    336        .BYTE 336        ;^
 00634    322        .BYTE 322        ;R
                 ;
 00635    246        .BYTE 246        ;&
 00636    101        .BYTE 101        ;A
 00637    102        .BYTE 102        ;B
 00640    303        .BYTE 303        ;C
 00641    104        .BYTE 104        ;D
 00642    305        .BYTE 305        ;E
 00643    306        .BYTE 306        ;F
 00644    107        .BYTE 107        ;G
 00645    110        .BYTE 110        ;H
 00646    240        .BYTE 240        ;EMPTY
 00647    333        .BYTE 333        ;[
 00650    056        .BYTE 56         ;.
 00651    074        .BYTE 74         ;<
 00652    050        .BYTE 50         ;(
 00653    053        .BYTE 53         ;+
 00654    041        .BYTE 41         ;!
 00655    311        .BYTE 311        ;I
 00656    173        .BYTE 173        ;LEFT CURLY BRACKET
 00657    175        .BYTE 175        ;RIGHT CURLY BRACKET
                     .ENDC
```

```
                         ;PARITY ASCII CONVERSION TABLE FOR 026 PUNCH:
                                 .IFNDF  ONLY29
 000660     240                  .BYTE 240                ;SPACE
 000661     261                  .BYTE 261                ;1
 000662     262                  .BYTE 262                ;2
 000663     063                  .BYTE 63                 ;3
 000664     264                  .BYTE 264                ;4
 000665     065                  .BYTE 65                 ;5
 000666     066                  .BYTE 66                 ;6
 00667      267                  .BYTE 267                ;7
 00670      270                  .BYTE 270                ;8
 00671      240                  .BYTE 240                ;EMPTY
 00672      137                  .BYTE 137                ;[
 00673      275                  .BYTE 275                ;=
 00674      300                  .BYTE 300                ;@
 00675      336                  .BYTE 336                ;^
 00676      047                  .BYTE 47                 ;'
 00677      134                  .BYTE 134                ;\
 00700      271                  .BYTE 71                 ;9
                         ;
 00701      060                  .BYTE 60                 ;0
 00702      257                  .BYTE 257                ;/
 00703      123                  .BYTE 123                ;S
 00704      324                  .BYTE 324                ;T
 00705      125                  .BYTE 125                ;U
 00706      126                  .BYTE 126                ;V
 00707      327                  .BYTE 327                ;W
 00710      330                  .BYTE 330                ;X
 00711      131                  .BYTE 131                ;Y
 00712      240                  .BYTE 240                ;EMPTY
 00713      273                  .BYTE 273                ;;
 00714      254                  .BYTE 254                ;,
 00715      050                  .BYTE 50                 ;(
 00716      042                  .BYTE 42                 ;"
 00717      243                  .BYTE 243                ;#
 00720      245                  .BYTE 245                ;%
 00721      132                  .BYTE 132                ;Z
                         ;
```

```
000722     055          .BYTE 55          ;-
000723     312          .BYTE 312         ;J
000724     113          .BYTE 113         ;K
000725     314          .BYTE 314         ;L
000726     115          .BYTE 115         ;M
000727     116          .BYTE 116         ;N
000730     317          .BYTE 317         ;O
000731     120          .BYTE 120         ;P
000732     321          .BYTE 321         ;Q
 00733     240          .BYTE 240         ;EMPTY
 00734     072          .BYTE 72          ;:
 00735     044          .BYTE 44          ;$
 00736     252          .BYTE 252         ;*
 00737     333          .BYTE 333         ;[
 00740     276          .BYTE 276         ;>
 00741     246          .BYTE 246         ;&
 00742     322          .BYTE 322         ;R
                 ;
 00743     053          .BYTE 53          ;+
 00744     101          .BYTE 101         ;A
 00745     102          .BYTE 102         ;B
 00746     303          .BYTE 303         ;C
 00747     104          .BYTE 104         ;D
 00750     305          .BYTE 305         ;E
 00751     306          .BYTE 306         ;F
 00752     107          .BYTE 107         ;G
 00753     110          .BYTE 110         ;H
 00754     240          .BYTE 240         ;EMPTY
 00755     077          .BYTE 77          ;?
 00756     056          .BYTE 56          ;.
 00757     251          .BYTE 251         ;)
 00760     335          .BYTE 335         ;]
 00761     074          .BYTE 74          ;<
 00762     241          .BYTE 41          ;!
 00763     311          .BYTE 311         ;I
 00764     173          .BYTE 173         ;LEFT CURLY BRACKET
 00765     175          .BYTE 175         ;RIGHT CURLY BRACKET
                        .ENDC
```

```
                        ;INTERNAL BUFFER FOR BINARY STORAGE:
                                .IFDF   BINARY
                        CR.BUF: .IFNDF  MARKS
                        CR.BSZ=120.
                                .ENDC
                                .IFDF   MARKS
                        CR.BSZ=60.
                                .ENDC
                        .=.+CR.BSZ
                                .ENDC
        000001'                 .END
```

```
CV.CR    MACRO V004-14 13-SEP-72 02:57 PAGE 16+
SYMBOL TABLE

BSUF   = 007007          CR       000000RG         CR.AGN  000044R
CR.ASC   000134R         CR.BXT   000364R          CR.CSR= 177160
CR.CVD   000264R         CR.CVL   000256R          CR.CVT  000270R
CR.CXT   000366R         CR.C1    000146R          CR.DB1= 177162
CR.DB2=  177164          CR.DUN   000400R          CR.DXT  000446R
CR.EOF   000474R         CR.ERR   000456R          CR.EXT  000510R
CR.INT   000106R         CR.IXT   000374R          CR.NAM  000014R
CR.NRY   000536R         CR.ONR   000016R          CR.OPN  000022R
CR.RPG   000346R         CR.RPT   000466R          CR.RSV= 000044
CR.STO   000342R         CR.SLR   000326R          CR.SXT= 000042
CR.TBL   000552R         CR.TFR   000044R          CR.TOS  000034R
CR.LBF   000066R         CR.URP   000072R          CR.026  000520R
CR.029   000530R         EOF    = 007417           PC     =%000007
R0     =%000000          R1     =%000001           R2     =%000002
R3     =%000003          R4     =%000004           R5     =%000005
SET26  = 004242          SET29  = 005252           SP     =%000006

. ABS.   000000     000
         000766     001

ERRORS DETECTED:  0
FREE CORE:  19337. WORDS
,LP:<DT:CR
```

```
;COPYRIGHT 1971, DIGITAL EQUIPMENT COPR., MAYNARD, MASS.

;VERSION NUMBER:          V005A
;                         000
;
;CARD READER DRIVER (CR)
;
;  A)    FOR ASCII INPUT. AT EACH TRANSFER REQUEST
;        ONE CARD WILL BE READ. UP TO 80 CHARACTERS,
;        FOLLOWED BY CR-LF, WILL BE PASSED TO THE
;        CALLING ROUTINE AS SPECIFIED BY THE WORD
;        COUNT GIVEN. (IF THIS IS > 41, REMAINING
;        BYTES WILL BE CLEARED.
;        ALL ERRORS (INCLUDING 'HOPPER EMPTY' UPON
;        AN 'OPEN' CALL) WILL BE TREATED AS 'DEVICE
;        NOT READY'. USER CAN RESUME OPERATION BY
;        RECTIFICATION OF ERROR OR REFILL OF HOPPER
;        AND ENTRY OF 'CO' COMMAND AT KEYBOARD.
;        THE END OF A FILE WILL BE DETERMINED BY
;        RECOGNITION OF A TERMINAL CONTROL CARD:-
;                12-11-0-1 PUNCHED IN C.C. 1
;
;                XXXXXXXXXXXXXXXXXXXXXXXX
;                X                      X
;                X         NOTES        X
;                X                      X
;                XXXXXXXXXXXXXXXXXXXXXXXX
;
;  1)    THIS DRIVER CAN BE ASSEMBLED FOR USE
;        IN CONNECTION WITH EITHER '026' OR '029'
;        PUNCHES OR BOTH AS INDICATED BY PARAMETER
;        SPECIFICATION AT START OF SOURCE INPUT
;        AS FOLLOWS:-
;
;        A)  "ONLY26=0"   -  READ ONLY '026' CODES.
;        B)  "ONLY29=0"   -  READ ONLY '029' CODES
;        C)  "DEFALT=0"   -  READ BOTH TYPES OF CODE
;                            WITH '026' AS DEFAULT
;        D)    NIL        -  READ BOTH TYPES OF CODE
;                            WITH '029' AS DEFAULT
;
;        IN CASES (C) & (D), DRIVER WILL USE DEFAULT
;        UNLESS DIRECTED OTHERWISE BY ENTRY OF A
;        CONTROL CARD PUNCHED IN C.C. 1:-
;
;                12-0-7-9   =  '029' CODES FOLLOW
;                12-11-8-9  =  '026' CODES FOLLOW
;
;  2)    IF PARAMETER "BLANKS" IS DEFINED, C.C. 73-80
;        & TRAILING SPACES BEFORE THESE WILL BE DISCARDED,
;        WITH 'CR-LF' FOLLOWING LAST VALID DATA, PROVIDED
;        THAT CARD FILE IS PRECEDED BY CTL CARD WITH
;        12-11-0-7-8-9 PUNCHED IN C.C. 1. IN THIS CASE
;        HOWEVER, IF THE USER BUFFER IS </= 82 BYTES,
;        ONLY TRAILING SPACE REMOVAL WILL BE EFFECTED.
```

```
;  B)   IF THE PARAMETER "BINARY" IS DEFINED AT ASSEMBLY,
;       XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
;
;       THIS VERSION WILL ALSO ALLOW READING OF CARDS
;       IN BINARY FORMAT. AS EACH CARD IS READ, 12 BITS FROM
;       ALL 8Ø COLS WILL BE ACCESSED & STORED IN AN INTERNAL
;       BUFFER IN A PACKED FORM, I.E. 4 COLS = 3 WORDS AS
;       FOLLOWS:-
;               C.C.1 > WORD 1, BITS 15-4
;               C.C.2 > WORD 1, BITS 3-Ø; WORD 2, BITS 15-8
;               C.C.3 > WORD 2, BITS 7-Ø; WORD 3, BITS 15-12
;               C.C.4 > WORD 3, BITS 11-Ø
;
;       THE PACKED FORM WILL BE TRANSFERRED TO THE USER
;       BUFFER UNTIL THIS IS FILLED, ANY DATA THEN REMAINING
;       IN THE INTERNAL BUFFER BEING RETAINED UNTIL THE
;       NEXT READ REQUEST.
;
;       TREATMENT OF ASCII READING WILL STILL FOLLOW PATTERN
;       DESCRIBED IN THE PREVIOUS PARAGRAPH WITHOUT EXCEPTION.
;
;       THE ONLY CONTROL CARD WHICH WILL HAVE ANY EFFECT
;       IN BINARY READING WILL BE THAT INDICATING E.O.F. (IN
;       THIS CASE 12-11-Ø-1 PUNCHING MUST APPEAR IN AT LEAST
;       C.C. 1 THRU 8).
;
;       N.B. WHEN ASSEMBLED FOR USAGE IN BOTH MODES, AN 'OPEN'
;       CALL WILL NOT CAUSE READING OF A CARD TO ALLOW THIS
;       TO BE TRANSLATED AS REQUIRED BY THE READ MODE
;       SPECIFIED BY THE USER.
;
;  C)   DRIVER CAN ALSO BE USED FOR 8Ø-COLUMN MARK
;       SENSE READER. FOR 4Ø -COLUMN READER, ECONOMIES
;       IN BUFFER SIZE CAN BE OBTAINED BY DEFINITION
;       AT ASSEMLBY OF PARAMETER "MARKS". THIS WILL
;       ALSO PREVENT AUTOMATIC REMOVAL OF COLS 33-4Ø
;       IN 'BLANKS-SUPPRESS' MODE OF USAGE.

;               XXXXXXXXXXXXXXXXXXXXXXXXX
;               X                       X
;               X         NOTE          X
;               X                       X
;               XXXXXXXXXXXXXXXXXXXXXXXXX
;
;       PARAMETER DEFINITIONS CAN BE MADE DURING PASS 1
;       OF THE ASSEMBLY ONLY IF REQD AS DESCRIBED IN
;       PAL-11R MANUAL, SECTION 9-2, E.G.
;
;               #CR,LP:,/PA:2<KB:/PA:1,DF:CR
;       FOLLOWED BY:-
;               ONLY29=Ø
;               BINARY=Ø
;               BLANKS=Ø
;               ^C
;               .END<CR><LF>
;
;               XXXXXXXXXXXXXXXXXXXXXXXX
;
```

```
                              .TITLE CR
                              .GLOBL CR
                      ;
        000000 R0=%0
        000001 R1=%1
        000002 R2=%2
        000003 R3=%3
        000004 R4=%4
        000005 R5=%5
        000006 SP=%6
        000007 PC=%7
                      ;INTERFACE TABLE:
000000 000000 CR:         .WORD 0                     ;CURRENT DDB OR 0 IF IDLE
                          .IFNDF   BINARY
000002    224             .BYTE    224,0              ;FACILITIES: ASCII INPUT,OPEN IN
000003    000
                          .ENDC
                          .IFDF    BINARY
                          .BYTE    234,0              ;ALLOW BINARY IF REQD.
                          .ENDC
                          .IFNDF   MARKS
000004    003             .BYTE    3                  ;STD BUFFER SIZE = 96 BYTES
                          .ENDC
                          .IFDF    MARKS
                          .BYTE    2                  ;(64 IF 40-COL MARK SENSE)
                          .ENDC
000005    104             .BYTE    CR.INT-CR,300      ;INTERUPT SVCE AT PRL 6
000006    300
000007    022             .BYTE    CR.OPN-CR          ;OFFSET TO OPEN
000010    042             .BYTE    CR.TFR-CR          ;OFFSET TO TRANSFER
000011    000             .BYTE    0,0,0              ;(NO CLOSE OR SPEC. FUNC.)
000012    000
000013    000
000014 012620 CR.NAM: .RAD50   'CR'
                      ;OPEN PROCESSOR:
000016 004767 CR.ONR: JSR      PC,CR.NRY
       000356
000022 032737 CR.OPN: BIT      #400,@#CR.CSR          ;CARDS IN HOPPER?
       000400
       177160
000030 001372         BNE      CR.ONR                 ;IF NOT TELL USER TO READY
                      .IFNDF   ONLY26&ONLY29
                      .IFNDF   DEFALT                 ;FOR DUAL-PUNCH DRIVER ...
000032 005027         CLR      (PC)+                  ;... SET CONV. TABLE OFFSET
                      .ENDC                           ;... FOR DEFAULT PUNCH
                      .IFDF    DEFALT                 ;... AS APPROPRIATE
                      MOV      #104,(PC)+
                      .ENDC
000034 000000 CR.TOS: .WORD    0
                      .ENDC
                      .IFDF    BLANKS                 ;IF BLANK-SUPPRESS VERSION ...
                      CLRB     CR.ZSW                 ;... FORCE SUPPRESS OFF
                      .ENDC
                      .IFNDF   BINARY
000036 005726         TST      (SP)+                  ;IGNORE INTERIM RETURN
000040 000523         BR       CR.DXT                 ;... & TAKE COMPLETION
                      .ENDC
                      .IFDF    BINARY                 ;FOR BINARY VERSION ...
                      CLRB     CR.TFR                 ;... FORCE NO OPEN READ
                      BR       CR.ODN
              CR.OXT: INCB     CR.TFR                 ;... BY MAKING COME HERE
                      BR       CR.ODN                 ;... BEFORE EXIT
                      .ENDC
```

```
                        ;SUBSIDIARY ROUTINES:
                        ;  A)  RESTART AFTER ERROR:
                        CR.AGN: .IFDF   BINARY          ;IN BINARY VERSION ...
                                TSTB    CR.ISW          ;... CHECK IF BINARY READ
                                BEQ     CR.TFR          ;IF NOT CAN JUST START OVER
                                BR      CR.ERD          ;ELSE LEAVE USER BUFFER ALONE
                        ;  B)  INITIALISE INTERNAL BUFFER POINTERS:
                        CR.ISP: MOV     PC,-(SP)        ;GET BUFFER START
                                ADD     #CR.BUF-.,@SP
                                MOV     @SP,(PC)+
                        CR.IBS: .WORD   0
                                ADD     #CR.BSZ,@SP     ;NOW GET END
                                MOV     @SP,(PC)+       ;STORE AS CONTROL
                        CR.IBE: .WORD   0
                                MOV     (SP)+,(PC)+     ;... & AS INIT. PTR
                        CR.IBP: .WORD   0
                                INCB    @PC             ;MUSTN'T COME HERE AGAIN!
                                .ENDC

                        ;TRANSFER SET-UP PROCESSOR:
                        CR.TFR: .IFDF   BINARY          ;FOR BINARY VERSION ...
                                BR      .+4             ;... SWITCH TABLE FORCES ...
                                BR      CR.OXT          ;... CORRECT INIT.
                                BR      CR.ISP
                                .ENDC
000042 016700                   MOV     CR,R0           ;GET DDB ADDRESS
       177732
000046 062700                   ADD     #6,R0           ;... & MOVE TO BUFFER STORE
       000006
000052 012046                   MOV     (R0)+,-(SP)     ;GET BUFFER POINTER
000054 011646                   MOV     @SP,-(SP)       ;... & BUILD BUFF END
000056 161016                   SUB     @R0,@SP
000060 162016                   SUB     (R0)+,@SP
000062 012627                   MOV     (SP)+,(PC)+     ;SAVE RESULT ...
000064 000000           CR.UBE: .WORD   0
                                .IFDF   BINARY          ;IN BINARY VERSION ...
                                MOVB    @R0,R0          ;... CHECK MODE
                                BICB    #376,R0
                                MOVB    R0,CR.ISW       ;IF ASCII CLEAR SWITCH
                                BEQ     CR.RDC          ;IF BINARY WANTED ...
                                MOV     (SP)+,R0        ;... SET PTRS & SWITCH
                                MOV     CR.IBP,R1
                                CLR     R2              ;SET INTERRUPT FLAG
                        CR.BIN: CMP     R1,CR.IBE       ;INTERNAL BUFF EMPTY?
                                BNE     CR.BLP
                        CR.ERD: MOV     CR.IBS,CR.IBP   ;IF SO RESET INTERNAL PTR
                                MOV     R0,-(SP)        ;SAVE USER BUFF PTR ...
                        CR.RDC: .ENDC
000066 011627                   MOV     @SP,(PC)+
000070 000000           CR.UBP: .WORD   0
000072 005036                   CLR     @(SP)+          ;ZERO UNDERWAY FLAG
000074 012737                   MOV     #101,@#CR.CSR   ;ENABLE INT & GO FOR CD READ
       000101
       177160
000102 000207                   RTS     PC              ;RETURN USER FOR NOW
```

```
            .IFDF   BINARY          ;WITH BINARY DATA ...
    CR.BLP: SWAB    @R1             ;... COMPLETE CONVERSION
            MOV     (R1)+,(R0)+     ;... & GIVE TO USER
            CMP     R0,CR.UBE       ;USER BUFFER FULL?
            BNE     CR.BIN          ;IF NOT GET NEXT WORD
            MOV     R1,CR.IBP       ;OTHERWISE SAVE INT PTR
            TST     R2              ;COME HERE ON INTERRUPT?
            BNE     CR.ODN          ;IF SO MODE SW. SET
            MOV     @SP,-(SP)       ;ELSE MUST SIMULATE ...
            MOV     -(R2),2(SP)     ;... STORE PC & PS
            SUB     #16,SP          ;... & DUMMY SAVE REGS.
    CR.ODN: TST     (SP)+           ;IGNORE RETURN PC
            BR      CR.DXT          ;... & TAKE COMPLETION EXIT
            .ENDC
```

```
                    ;INTERRUPT SERVICE ROUTINES:
                    ;   A)  CHECK FOR ERROR & COLLECT INPUT:
000104 010046       CR.INT: MOV     R0,-(SP)        ;SAVE USER R0
000106 016700               MOV     CR.UBP,R0       ;GET USER BUFF PTR ...
       177756
000112 013746               MOV     @#CR.CSR,-(SP)  ;... & READER STATUS
       177160
000116 006326               ASL     (SP)+           ;CHECK FOR SPECIAL CASES
000120 103477               BCS     CR.ERR          ;GO RETRY IF ERROR
000122 100447               BMI     CR.DUN          ;CLEAN UP IF DONE
000124 010146               MOV     R1,-(SP)        ;NOW SAVE USER R1
                            .IFDF   BINARY          ;IN BINARY VERSION ...
                    CR.ISW: BR      .+4             ;... USE APPROPRIATE CONVERSION
                            BR      CR.ASC
                    ;   B)  BINARY CONVERT & STORE:
                            MOV     CR.IBP,R1       ;GET INT BUFF PTR
                            MOV     @#CR.DB1,-(SP)  ;... & INPUT
                            MOV     @SP,-(SP)       ;... 2 COPIES FOR LATER
                            COMB    1(R0)           ;ODD COLUMN?
                            BPL     CR.BST
                            ASL     @SP             ;IF SO SHIFT INPUT TO HIGH
                            ASL     @SP
                            ASL     @SP
                            ASL     @SP
                            CLRB    (R1)+           ;MAKE NXT INSTR = MOVB
                    CR.BST: BISB    1(SP),-1(R1)    ;SET HIGH BYTE AS REQD.
                            MOVB    (SP)+,(R1)+     ;THEN LOW BYTE
                            MOV     R1,CR.IBP       ;SAVE PTR
                            CMP     (SP)+,#7400     ;NOW LOOK FOR EOF CARD
                            BNE     CR.BXT
                            DECB    @R0             ;..., PUNCHED 12-11-0-1
                            BPL     CR.BXT          ;... IN CC 1 THRU 8
                            ASLB    @R0
                            BPL     CR.EOF          ;IF END IGNORE REST OF CARD
                            BR      CR.CXT          ;OTHERWISE TRY NEXT TIME
                            .ENDC
```

```
                 ;  C)  ASCII CONVERT & STORE:
000126 113701 CR.ASC: MOVB     @#CR.DB2,R1        ;GET COMPACTED INPUT
       177164
000132 100002         BPL      .+6                ;CONVERT CODES >200 ...
000134 062701         ADD      #340,R1            ;... TO RANGE >140
       000340
000140 105710         TSTB     @R0                ;IF FIRST C.C. ...
000142 001011         BNE      CR.CVT
000144 020127         CMP      R1,#301            ;... LOOK FOR E.O.F.
       000301
000150 001472         BEQ      CR.EOF
                      .IFDF    BLANKS             ;FOR BLANK SUPPRESS ...
                      CMPB     R1,#337            ;... LOOK FOR SUPPRESS ON
                      BEQ      CR.ZON
                      .ENDC
                      .IFNDF   ONLY26&ONLY29      ;FOR DUAL PUNCH DRIVER ...
000152 020127         CMP      R1,#227            ;... CHECK IF 029 CTL
       000227
000156 001505         BEQ      CR.029
000160 020127         CMP      R1,#270            ;... OR 026 CTL
       000270
000164 001476         BEQ      CR.026
                      .ENDC
000166 010146 CR.CVT: MOV      R1,-(SP)           ;CONVERT CARD CODE ...
000170 162701         SUB      #40,R1             ;FOR EACH 40 IN CODE ...
       000040
000174 100403         BMI      CR.STO             ;... ADD 21 & STRIP 40
000176 162716         SUB      #17,@SP            ;... TO GET TABLE INDEX
       000017
000202 000772         BR       CR.CVT+2
              CR.STO: .IFNDF   ONLY26&ONLY29
000204 066716         ADD      CR.TOS,@SP         ;PICK APPROP. TABLE
       177624
                      .ENDC
000210 060716         ADD      PC,@SP             ;COMPUTE ADDR OF BYTE REQD
000212 062716         ADD      #CR.TBL-.,@SP
       000202
000216 113620         MOVB     @(SP)+,(R0)+       ;... & STORE IN BUFFER
000220 020067         CMP      R0,CR.UBE          ;BUFFER FULL?
       177640
000224 001452         BEQ      CR.EXT
000226 151710 CR.BXT: BISB     @PC,@R0            ;IF NOT SET UNDERWAY FLAG
000230 010067 CR.CXT: MOV      R0,CR.UBP          ;SAVE NEW POINTER
       177634
000234 012601         MOV      (SP)+,R1           ;RESTORE USER REGS.
000236 012600 CR.IXT: MOV      (SP)+,R0
000240 000002         RTI                         ;... & EXIT
```

```
                        ;  D)   CARD COMPLETED:
000242 105037   CR.DUN: CLRB    @#CR.CSR                ;STOP INTERRUPTS
       177160
000246 105710           TSTB    @R0                     ;IF NO PROCESSING YET ...
000250 001427           BEQ     CR.RPT                  ;... CONTINUE
000252 012600           MOV     (SP)+,R0                ;OTHERWISE RESTORE USER R0
000254 013746           MOV     @#CR.RSV,-(SP)          ;... & NOW SAVE ALL
       000044
000260 004536           JSR     R5,@(SP)+
000262 016700           MOV     CR.UBP,R0               ;SET USER BUFF PTR
       177602
                        .IFDF   BINARY                  ;FOR BINARY VERSION ...
                        TSTB    CR.ISW                  ;... CHECK IF BINARY READ
                        BNE     CR.BDN                  ;IF SO ACTION ACCRODINGLY
                        .ENDC
000266 016701           MOV     CR.UBE,R1               ;FOR ASCII, SET END PTR
       177572
                        .IFDF   BLANKS                  ;... & PERHAPS CHECK SUPPRESS
                CR.ZSW: BR      .+4                     ;SWITCH ON?
                        BR      CR.ADN                  ;IF NOT NO SUPPRESSION
                        .IFNDF  MARKS
                        TSTB    @R0                     ;TEST IF END OF FILE
                        BMI     CR.DXT                  ;SKIP NEXT CALCULATION IF EOF
                        CMP     R0,R1                   ;IF BUFFER FULL OMIT NEXT
                        BEQ     .+6
                        SUB     #8.,R0                  ;OTHERWISE LOSE CC 73-80
                        .ENDC
                        CMPB    -(R0),#240              ;THEN TRAILING SPACES
                        BEQ     .-4
                        TSTB    (R0)+                   ;ADJUST PTR WHEN DONE
                CR.ADN: .ENDC
000272 105041           CLRB    -(R1)                   ;CLEAR REST OF BUFFER
000274 020100           CMP     R1,R0
000276 101375           BHI     .-4
000300 112721           MOVB    #215,(R1)+              ;MOVE IN CARRIAGE RETURN
       000215
000304 112721           MOVB    #012,(R1)+              ;MOVE IN LINE FEED
       000012
000310 016700   CR.DXT: MOV     CR,R0                   ;GET DDB ADDRESS
       177464
000314 000170           JMP     @14(R0)                 ;TAKE COMPLETION EXIT
       000014
                        .IFDF   BINARY
                CR.BDN: MOV     CR.IBS,R1               ;FOR BINARY, INIT INT PTR
                        MOVB    @R0,R2                  ;EXIT IF EOF SEEN
                        BMI     CR.DXT                  ;ALSO SETS INTERRUPT FLAG
                        JSR     PC,CR.BIN               ;ELSE GO MOVE DATA TO USER
                        MOV     @#CR.SXT,R5             ;IF COME BACK, MORE READ REQD.
                        JMP     4(R5)                   ;SO TAKE SYSTEM EXIT
                        .ENDC
```

```
                ;SPECIAL CASE PROCESSING:
                ;   A)   ERROR ROUTINE:
000320 105037 CR.ERR: CLRB     @#CR.CSR          ;STOP INTERRUPTS
       177160
000324 004767         JSR      PC,CR.NRY         ;INFORM OPERATOR
       000050
000330 004767 CR.RPT: JSR      PC,CR.AGN         ;IF RETURN TRY AGAIN
       177506
000334 000740         BR       CR.IXT            ;... & EXIT FOR NOW
                ;   B)   END OF FILE CARD SEEN:
000336 016701 CR.EOF: MOV      CR,R1             ;GET DDB ADDRESS
       177436
                      .IFNDF   BINARY            ;FOR SIMPLE VERSION ...
000342 016161         MOV      10(R1),16(R1)     ;... NO DATA READ ON EOF
       000010
       000016
                      .ENDC
                      .IFDF    BINARY            ;MAYBE SOME IF BINARY ...
                      ADD      #16,R1            ;SO MOVE TO UNUSED COUNT STORE
                      MOV      R0,@R1            ;... & COMPUTE VALUE REQD
                      SUB      CR.UBE,@R1
                      ASR      @R1               ;... AS WORDS!
                      .ENDC
000350 005110         COM      @R0               ;SET FLAG
000352 152737 CR.EXT: BISB     #2,@#CR.CSR       ;ALLOW REST OF CARD THRU
       000002
       177160
000360 000723         BR       CR.CXT
                ;   C)   CONTROL CARD SEEN:
                      .IFNDF   ONLY26&ONLY29     ;FOR DUAL PUNCH DRIVER ...
000362 012767 CR.026: MOV      #104,CR.TOS       ;... SET TABLE OFFSET ...
       000104
       177444
000370 000770         BR       CR.EXT            ;... & IGNORE REST OF CTL CARD
000372 005067 CR.029: CLR      CR.TOS
       177436
000376 000765         BR       CR.EXT
                      .ENDC
                      .IFDF    BLANKS            ;IN SUPPRESS VERSION
              CR.ZON: MOVB     #1,CR.ZSW         ;...SET SUPPRESS ON
                      BR       CR.EXT            ;AGAIN IGNORE REST OF CARD
                      .ENDC


              ;READER NOT READY SUBROUTINE:
000400 016746 CR.NRY: MOV      CR.NAM,-(SP)      ;IDENTIFY DEVICE
       177410
000404 012746         MOV      #402,-(SP)        ;GIVE ONT READY CODE
       000402
000410 000004         IOT                        ;... & CALL EDP
000412 000207         RTS      PC                ;TRY AGAIN IF COME BACK
              ;
              ;MISCELLANEOUS DEFINITIONS:
       177160 CR.CSR=177160
       177162 CR.DB1=177162
       177164 CR.DB2=177164
       000042 CR.SXT=42
       000044 CR.RSV=44
```

```
                    CR.TBL:
                    ;PARITY ASCII CONVERSION TABLE FOR 029 PUNCH
                              .IFNDF  ONLY26
000414    240                 .BYTE 240              ;SPACE
000415    261                 .BYTE 261              ;1
000416    262                 .BYTE 262              ;;2
000417    063                 .BYTE 63               ;3
000420    264                 .BYTE 264              ;4
000421    065                 .BYTE 65               ;5
000422    066                 .BYTE 66               ;6
000423    267                 .BYTE 267              ;7
000424    270                 .BYTE 270              ;8
000425    240                 .BYTE 240              ;EMPTY
000426    072                 .BYTE 72               ;:
000427    243                 .BYTE 243              ;#
000430    300                 .BYTE 300              ;@
000431    047                 .BYTE 47               ;'
000432    275                 .BYTE 275              ;=
000433    042                 .BYTE 42               ;"
000434    071                 .BYTE 71               ;9
                    ;
000435    060                 .BYTE 60               ;0
000436    257                 .BYTE 257              ;/
000437    123                 .BYTE 123              ;S
000440    324                 .BYTE 324              ;T
000441    125                 .BYTE 125              ;U
000442    126                 .BYTE 126              ;V
000443    327                 .BYTE 327              ;W
000444    330                 .BYTE 330              ;X
000445    131                 .BYTE 131              ;Y
000446    240                 .BYTE 240              ;EMPTY
000447    335                 .BYTE 335              ;]
000450    254                 .BYTE 254              ;,
000451    245                 .BYTE 245              ;%
000452    137                 .BYTE 137              ;_
000453    276                 .BYTE 276              ;>
000454    077                 .BYTE 77               ;?
000455    132                 .BYTE 132              ;Z
                    ;

000456    055                 .BYTE 55               ;-
000457    312                 .BYTE 312              ;J
000460    113                 .BYTE 113              ;K
000461    314                 .BYTE 314              ;L
000462    115                 .BYTE 115              ;M
000463    116                 .BYTE 116              ;N
000464    317                 .BYTE 317              ;O
000465    120                 .BYTE 120              ;P
000466    321                 .BYTE 321              ;Q
000467    240                 .BYTE 240              ;EMPTY
000470    041                 .BYTE 41               ;!
000471    044                 .BYTE 44               ;$
000472    252                 .BYTE 252              ;*
000473    251                 .BYTE 251              ;)
000474    273                 .BYTE 273              ;;
000475    134                 .BYTE 134              ;\
000476    322                 .BYTE 322              ;R
                    ;
```

```
000477  246         .BYTE 246          ;&
000500  101         .BYTE 101          ;A
000501  102         .BYTE 102          ;B
000502  303         .BYTE 303          ;C
000503  104         .BYTE 104          ;D
000504  305         .BYTE 305          ;E
000505  306         .BYTE 306          ;F
000506  107         .BYTE 107          ;G
000507  110         .BYTE 110          ;H
000510  240         .BYTE 240          ;EMPTY
000511  333         .BYTE 333          ;[
000512  056         .BYTE 56           ;.
000513  074         .BYTE 74           ;<
000514  050         .BYTE 50           ;(
000515  053         .BYTE 53           ;+
000516  336         .BYTE 336          ;^
000517  311         .BYTE 311          ;I
                    .ENDC


               ;PARITY ASCII CONVERSION TABLE FOR 026 PUNCH:
                    .IFNDF   ONLY29
000520  240         .BYTE 240          ;SPACE
000521  261         .BYTE 261          ;1
000522  262         .BYTE 262          ;2
000523  063         .BYTE 63           ;3
000524  264         .BYTE 264          ;4
000525  065         .BYTE 65           ;5
000526  066         .BYTE 66           ;6
000527  267         .BYTE 267          ;7
000530  270         .BYTE 270          ;8
000531  240         .BYTE 240          ;EMPTY
000532  137         .BYTE 137          ;_
000533  275         .BYTE 275          ;=
000534  300         .BYTE 300          ;@
000535  336         .BYTE 336          ;^
000536  047         .BYTE 47           ;'
000537  134         .BYTE 134          ;\
000540  071         .BYTE 71           ;9
               ;
000541  060         .BYTE 60           ;0
000542  257         .BYTE 257          ;/
000543  123         .BYTE 123          ;S
000544  324         .BYTE 324          ;T
000545  125         .BYTE 125          ;U
000546  126         .BYTE 126          ;V
000547  327         .BYTE 327          ;W
000550  330         .BYTE 330          ;X
000551  131         .BYTE 131          ;Y
000552  240         .BYTE 240          ;EMPTY
000553  273         .BYTE 273          ;;
000554  254         .BYTE 254          ;,
000555  050         .BYTE 50           ;(
000556  042         .BYTE 42           ;"
000557  243         .BYTE 243          ;#
000560  245         .BYTE 245          ;%
000561  132         .BYTE 132          ;Z
               ;
```

```
000562      055           .BYTE 55                ;-
000563      312           .BYTE 312               ;J
000564      113           .BYTE 113               ;K
000565      314           .BYTE 314               ;L
000566      115           .BYTE 115               ;M
000567      116           .BYTE 116               ;N
000570      317           .BYTE 317               ;O
000571      120           .BYTE 120               ;P
000572      321           .BYTE 321               ;Q
000573      240           .BYTE 240               ;EMPTY
000574      072           .BYTE 72                ;:
000575      044           .BYTE 44                ;$
000576      252           .BYTE 252               ;*
000577      333           .BYTE 333               ;[
000600      276           .BYTE 276               ;>
000601      246           .BYTE 246               ;&
000602      322           .BYTE 322               ;R
                    ;
000603      053           .BYTE 53                ;+
000604      101           .BYTE 101               ;A
000605      102           .BYTE 102               ;B
000606      303           .BYTE 303               ;C
000607      104           .BYTE 104               ;D
000610      305           .BYTE 305               ;E
000611      306           .BYTE 306               ;F
000612      107           .BYTE 107               ;G
000613      110           .BYTE 110               ;H
000614      240           .BYTE 240               ;EMPTY
000615      077           .BYTE 77                ;?
000616      056           .BYTE 56                ;.
000617      251           .BYTE 251               ;)
000620      335           .BYTE 335               ;]
000621      074           .BYTE 74                ;<
000622      041           .BYTE 41                ;!
000623      311           .BYTE 311               ;I
                          .ENDC


                    ;INTERNAL BUFFER FOR BINARY STORAGE:
                          .IFDF   BINARY
                    CR.BUF: .IFNDF  MARKS
                    CR.BSZ=120.
                          .ENDC
                          .IFDF   MARKS
                    CR.BSZ=60.
                          .ENDC
                    .=.+CR.BSZ
                          .ENDC
            000001        .END
```

```
000000 ERRORS




CR          000000RG      CR.AGN    000042R       CR.ASC    000126R
CR.BXT      000226R       CR.CSR =  177160        CR.CVT    000166R
CR.CXT      000230R       CR.DB1 =  177162        CR.DB2 =  177164
CR.DUN      000242R       CR.DXT    000310R       CR.EOF    000336R
CR.ERR      000320R       CR.EXT    000352R       CR.INT    000104R
CR.IXT      000236R       CR.NAM    000014R       CR.NRY    000400R
CR.ONR      000016R       CR.OPN    000022R       CR.RPT    000330R
CR.RSV =    000044        CR.STO    000204R       CR.SXT =  000042
CR.TBL      000414R       CR.TFR    000042R       CR.TOS    000034R
CR.UBE      000064R       CR.UBP    000070R       CR.026    000362R
CR.029      000372R       PC       =%000007       R0       =%000000
R1         =%000001       R2       =%000002       R3       =%000003
R4         =%000004       R5       =%000005       SP       =%000006
.          = 000624R
```

APPENDIX A

CHARACTER CODES

A.1 CARD CODES

CARD CODES
(ANSI X3.26–1970)

Zone / Digit	12	11	0		12 0	12 11	11 0		12 9	11 9	0 9	9	12 0 9	12 11 9
	&	–	0	space	{	|	}							
1	A	J	/	1	a	j	~		SOH	DC1				
2	B	K	S	2	b	k	s		STX	DC2		SYN		
3	C	L	T	3	c	l	t		ETX	DC3				
4	D	M	U	4	d	m	u							
5	E	N	V	5	e	n	v		HT		LF			
6	F	O	W	6	f	o	w			BS	ETB			
7	G	P	X	7	g	p	x		DEL		ESC	EOT		
8	H	Q	Y	8	h	q	y			CAN				
9	I	R	Z	9	i	r	z							
8-1				grave						EM			NUL	DLE
8-2	[	]	\	:										
8-3	.	$	,	#					VT					
8-4	<	*	%	@					FF	FS		DC4		
8-5	(	)	_	'					CR	GS	ENQ	NAK		
8-6	+	;	>	=					SO	RS	ACK			
8-7	!	^	?	"					SI	US	BEL	SUB		

NOTES

To determine the card punch for a particular character, locate the character in the table and read the corresponding zone punch and then digit punch. For example, the card punch for a % is 0-8-4.

To obtain the character corresponding to a particular card punch, locate the junction of the zone punch and the digit punch. For example, the character corresponding to the card punch 12-11-9 is r.

Slots that do not contain characters represent card punches for which there are no ASCII equivalents.

A.2 PDP-11 PUNCHED CARD CODES

A.2 PDP-11 PUNCHED CARD CODES

CHARACTER	Parity ASCII	DECØ29	DECØ26	CHARACTER	Parity ASCII	DECØ29	DECØ26
{	173	12 Ø	12 Ø				
}	175	11 Ø	11 Ø				
SPACE	24Ø	NONE	NONE	@	3ØØ	8 4	8 4
!	Ø41	12 8 7	12 8 7	A	1Ø1	12 1	12 1
"	Ø42	8 7	Ø 8 5	B	1Ø2	12 2	12 2
#	243	8 3	Ø 8 6	C	3Ø3	12 3	12 3
$	Ø44	11 8 3	11 8 3	D	1Ø4	12 4	12 4
%	245	Ø 8 4	Ø 8 7	E	3Ø5	12 5	12 5
&	246	12	11 8 7	F	3Ø6	12 6	12 6
'	Ø47	8 5	8 6	G	1Ø7	12 7	12 7
(	Ø5Ø	12 8 5	Ø 8 4	H	11Ø	12 8	12 8
)	251	11 8 5	12 8 4	I	311	12 9	12 9
*	252	11 8 4	11 8 4	J	312	11 1	11 1
+	Ø53	12 8 6	12	K	113	11 2	11 2
,	254	Ø 8 3	Ø 8 3	L	314	11 3	11 3
-	Ø55	11	11	M	115	11 4	11 4
.	Ø56	12 8 3	12 8 3	N	116	11 5	11 5
/	257	Ø 1	Ø 1	O	317	11 6	11 6
Ø	Ø6Ø	Ø	Ø	P	12Ø	11 7	11 7
1	261	1	1	Q	321	11 8	11 8
2	262	2	2	R	322	11 9	11 9
3	Ø63	3	3	S	123	Ø 2	Ø 2
4	264	4	4	T	324	Ø 3	Ø 3
5	Ø65	5	5	U	125	Ø 4	Ø 4
6	Ø66	6	6	V	126	Ø 5	Ø 5
7	267	7	7	W	327	Ø 6	Ø 6
8	27Ø	8	8	X	33Ø	Ø 7	Ø 7
9	Ø71	9	9	Y	131	Ø 8	Ø 8
:	Ø72	8 2	11 8 2	Z	132	Ø 9	Ø 9
;	273	11 8 6	Ø 8 2	[	333	12 8 2	11 8 5
<	Ø74	12 8 4	12 8 6	\	134	Ø 8 2	8 7
=	275	8 6	8 3	]	335	11 8 2	12 8 5
>	276	Ø 8 6	11 8 6	↑ or ^	336	11 8 7	8 5
?	Ø77	Ø 8 7	12 8 2	← or _	137	Ø 8 5	8 2

APPENDIX B

ALGORITHMS USED IN CR11/CM11 CARD READER DRIVER

B.1 HOLLERITH TO ASCII CONVERSION

Examination of the valid Hollerith character codes listed at Appendix A shows that in any one character there can be only one punch, if any, in control zones 12, 11, and 0. When translated by the CR11 Control into byte form, as follows:

12	11	0	9	8			
7	6	5	4	3	2	1	0

Octal digit corr. Zones 1-7)

it can be seen that all characters must fall into one of the octal ranges: 0-37, 40-77; 100-137, 200-237. Moreover, within each range, the values are in fact restricted to the first seventeen. Basically, it is therefore possible to establish a table in four sections, each corresponding to one of these ranges, or two like tables if both 026 and 029 punches are considered.

Further, if the bytes so formed are transferred from the CR11 buffer into a register, values in the last range produce negative results by sign extension. If 340_8 is added to these, their range now becomes 140-177; a natural progression from the other three. Using a second register as a form of counter, a relative index to the required ASCII equivalent within its appropriate table section can be thus established simply by adding 21_8 to the low-order 5 bits of the Hollerith code for each time 40_8 can be successfully subtracted from the high-order three bits. (In practice, this is accomplished by subtracting 17_8 from one register containing the final index, to remove the 40 while adding 21, while reducing the counter register.) To the index must then be added the appropriate offset into the correct table for the punch concerned (0 for 029 and 104_8 for 026 if both tables are present). The address of the ASCII value required is merely the index added to a computed absolute table base.

B.2 BINARY PACKING

Basically, the packed format in which binary data is passed to a user program can first be considered as a problem of packing two 12-bit words representing card columns into three 8-bit bytes as:

Byte 0	Byte 1	Byte 2
cc 1 (12-5)	cc 1 (6-9)¦cc 2 (12-1)	cc 2 (2-9)

or, in other words, the first column is shifted to the high-order position over the first two bytes while the second column remains in the low-order position in which it was read over the last two. Using a simple flip-flop type of switch, the algorithm distinguishes between columns 1 and 2 to accomplish this during the appropriate interrupts. Columns 3 and 4 require similar treatment. Owing to the byte addressing scheme of PDP-11, however, the result so obtain means that the bytes within each word are misplaced. A simple byte-swap when the word is passed to the user corrects this.

B.3 SWITCHING

The version of the driver which allows binary processing requires several switches as noted in the main text. Because the Monitor will never allow the driver to be called to perform more than one operation at a time, there is no need for the driver to be restricted to reentrant code. As a result, the general form of switching used is of the form:

```
SWITCH:    BR  .+2
           BR  PROC.B
PROC.A:
```

When the low-order byte of SWITCH is cleared, the effective instruction at that point then cecomes BR .+2 and the branch to process B is taken. If on the other hand that byte is then incremented, the instruction becomes BR .+4 and process A is entered.

A variation of this technique has been used at the start of the TRANSFER routine in the binary-type driver firstly to allow OPEN to stop a first read as described in the main text, and secondly to cause execution of the once-only code needed to initialize the internal buffer pointers. This extends the single fixed branch to a table:

```
CR.TFR:    BR  .+4       (Allows READ w/o to proceed)
           BR  CR.OXT    (Ignore first read after OPEN)
           BR  CR.ISP    (Initialize buffer pointers)
```

If OPEN is called, the switch byte is cleared causing the first transfer call to branch to CR.OXT (BR .+2). The routine at CR.OXT merely increments it back to the BR .+4 state and exits. The next entry at CR.TFR (or the first if OPEN is not called) takes branch to CR.ISP. The last instruction of this routine, which also immediately precedes CR.TFR, executes INCB @PC, hence finally setting the switch byte for

BR .+6 leading to all successive calls beginning normal execution immediately, until either the driver is re-initialized by a new OPEN or is removed from core and brought in afresh.

APPENDIX C

UNPACKING BINARY DATA FROM THE CR11/CM11 CARD READER DRIVER

A SUGGESTED ALGORITHM

1. Each four card-columns of data passed to a user program in binary format appear in memory as follows, when the byte-addressing scheme of PDP-11 is considered:

cc 1 6-9 ¦ cc 2 12-1	cc 1 12-5	cc 3 12-5	cc 2 2-9	cc 4 2-9	cc 3 6-9 ¦ cc 4 12-1
BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5

At first sight, a simple algorithm to restore the original 12-bit card column images from this format might seem a problem. If, however, the bytes in each word are first switched, the format now shows a more logical sequence:

cc 1 12-5	cc 1 6-9 ¦ cc 2 12-1	cc 2 2-9	cc 3 12-5	cc 3 6-9 ¦ cc 4 12-1	cc 4 2-9
BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5

and the solution reduces merely to one of splitting two like sets of three bytes into two 12-bit words. The first step, therefore, is to perform the necessary swap across all words transferred. Assuming the data has been read into a line buffer as defined under DOS, the appropriate code might be:

```
MOV     #LINE+4,RØ       ;GET BYTE COUNT FROM LINE HDR
MOV     (RØ)+,R1         ;... & BUMP POINTER TO FIRST DATA
ADD     RØ,R1            ;USE CNT TO SET LINE END
MOV     RØ,R2            ;SAVE START POINTER
SWAB    (R2)+            ;... & DO BYTE SWITCH
CMP     R2,R1
BLO     .-4
```

2. The column 1 image is simply obtained by taking byte 0 as the high-order part and byte 1 as the low-order part of a word which is then shifted until the required 12 high-order bits are right-justified. Column 2 image is similarly extracted from byte 1 and byte 2 except that no shift is needed. Using a simple flip-flop type switch to differentiate between odd and even columns, the necessary sequence might be:

```
        CLR     R2              ;INIT. FLIP-FLOP
A:      MOVB    (RØ)+,-(SP)     ;GET FIRST BYTE
        SWAB    @SP             ;... INTO HIGH ORDER SPOT
        MOVB    (RØ)+,@SP       ;... & 2ND INTO LOW
        COM     R2              ;FIRST COLUMN?
        BPL     B
        ASR     @SP             ;IF SO, RIGHT JUSTIFY
        ASR     @SP
        ASR     @SP
        ASR     @SP
        DEC     RØ              ;STAY AT SECOND BYTE
B:      BIC     #17ØØØØ,@SP     ;REMOVE GARBAGE
        .......                 ;REQD. IMAGE NOW ON STACK TOP
        .......                 ;... & CAN BE PROCESSED AS NECESSARY
        CMP     RØ,R1           ;END OF BUFFER?
        BLO     A               ;IF NOT, GET NEXT IMAGE
```

NOTES

a) The stack is used rather than a register to build the image, as this avoids the problem of possible sign extension in the operation at A+4. After processing, the image should of course be removed before proceeding to the next.

b) For mere storage of images in another buffer, a further register might be used as a pointer, e.g., MOV #BUFFER,R3 and all references to SP can then be changed to @R3 until the one at B, which should become (R3)+ to step to next word.

APPENDIX D

PREPARATION AND USAGE OF CR11/CM11 CARD READ DRIVER

D.1 PREPARATION

It has been shown that, by defining the relevant conditional parameters at assembly time, the user can tailor the card-reader driver to meet the particular needs of his installation. To allow him to do this, the driver is supplied as a source tape. The following paragraphs illustrate special points to be observed in preparing this tape for usage. It is assumed that the user is already familiar with the general operating procedures of the PAL-11R Assembler and Link-11 Linker.

D.1.1 Assembly

In order to enter the definitions for the required optional parameters, the user should specify that the keyboard will be used to supply input on Pass 1 only. Thus, assuming a disk-to-disk assembly with line-printer listings, the response to PAL-11R request for command input might be:

```
#CR.OBJ,LP:,LP:<KB:/PA:1,DF:CR.PAL
```

After this, the user can begin to type in the definitions he needs, e.g.:

```
BLANKS = Ø
BINARY = Ø
```

and then terminate by calling the Monitor to signal the end of this particular input as:

```
↑C            (CTRL/C
.END          (Two carriage-returns are essential after .END)
```

Thereafter, the assembly proceeds in the normal way.

For easy reference, the possible parameters and their effects are summarized below:

Parameter	Driver Version	Size
Nil	ASCII only - 026 or 029 (029 default)	200
DEFALT	ASCII only - 026 or 029 (026 default)	+1

Parameter	Driver Version	Size
ONLY26	ASCII only - 026 punch only	-51
ONLY29	ASCII only - 029 punch only	-51
BLANKS	Auto-deletion of cc 73-80 & trailing spaces by Special Card Control	+18
BINARY	Adds binary to other capabilities.	+161
MARKS	Restricted driver for 40-col CM11	0 on Basic -4 on BLANKS -30 on BINARY

NOTE

Signed size values should be added to the basic size for the ASCII-only version, e.g., driver assembled with ONLY29, BINARY, and BLANKS defined is 328 words long. Also first four mutually exclusive.)

D.1.2 Linking

The output file from PAL-11R should then be linked to paper tape, using Link-11. It must be origined at location Ø if it is to be used within DOS, i.e., the Command String in this case might be:

```
#PP:,LP:<CR/B:Ø/E
```

D.1.3 Inclusion in DOS Monitor Library

The DOS Monitor has already been set up to recognize the Card Reader Driver. Therefore, the user need merely incorporate the linked module into the Monitor Library, either by making the paper tape part of the input while building the system on disk by SYSLOD or by including it in the Monitor Library on a DECtape using MODS. The associated descriptions for these programs give details of method.

D.2 USAGE

In general, a card file consists merely of the data cards followed by an EOF card, then a blank. It has been shown that control cards may also be included to force the driver to function in a special way. The format for each of these is summarized below:

EOF	12-11-0-1-6-7-8-9[1]	in c.c. 1 for ASCII, in c.c. 1-8 for Binary	
029 codes	12-0-2-4-6-8[2]	in c.c. 1	If also in c.c. 80, card is symmetrical & can be used any way up.
026 codes	12-2-4-8[3]	in c.c. 1	
Suppress Blanks	12-11-0-7-8-9	in c.c. 1	

[1]12-11-0-1 for Version ØØ5A Monitor Release 4A.
[2]12-0-7-9 for Version ØØ5A Monitor Release 4A.
[3]12-11-8-9 for Version ØØ5A Monitor Release 4A.

After placing the card file in the hopper, the operator should ensure that power has been turned on and that both the MOTOR START & READ START buttons have been pressed (both indicators green). The read will then respond to any program request for input from device CR.

If an error occurs at any time, the Monitor message "AØØ2 12Ø6Ø" will indicate this. The operator should rectify the error if possible, replace the last card read with the remainder of the deck in the hopper, re-enable the reader and type CO to resume.

P D P - 1 1

PC11 HIGH-SPEED PAPER TAPE READER DRIVER

October 1972

SUPPLEMENT TO:

PDP-11 DEVICE DRIVER PACKAGE

DEC-11-ODDPA-A-D

MONITOR VERSION VØØ8

NOTE

This document is for information purposes only and is subject to change without notice. DEC assumes no responsibility for the use or reliability of its software on equipment which is not supplied by DEC.

NEW AND CHANGED INFORMATION

This manual documents the software as of Monitor version VØ8. It has been revised to include all new and changed material since Monitor version VØ4. Such material is indicated by vertical bars in the outside margin. Whole new pages are not so marked but are dated in the lower outside corner.

SOFTWARE DISTRIBUTION CENTER

PC11 HIGH-SPEED PAPER TAPE READER DRIVER

The paper tape reader driver provides the device dependent I/O functions for the PDP-11 paper tape reader. To allow the common I/O processor to be device independent, the paper tape reader driver is a block processor. Any size block may be processed by the driver, but to provide the most efficient operation the standard buffer size is 32 words. The driver code is position independent.

1.1 DESCRIPTION

The paper tape reader driver consists of two sections: the standard driver header and the driver body.

The driver header gives the following information about the paper tape driver:

1. Capabilities
    a. Single user
    b. Input only device
    c. ASCII and BINARY both may be handled
    d. Non-file structured
2. 32 word standard buffer size
3. Interrupt entry address and priority (4)
4. Dispatch table containing entry addresses for:
    a. Open
    b. Transfer
5. Internal word count and buffer address

The driver body contains the code to perform the three paper tape reader functions: opening, reading (transfer), and interrupt servicing.

1.2 OPEN

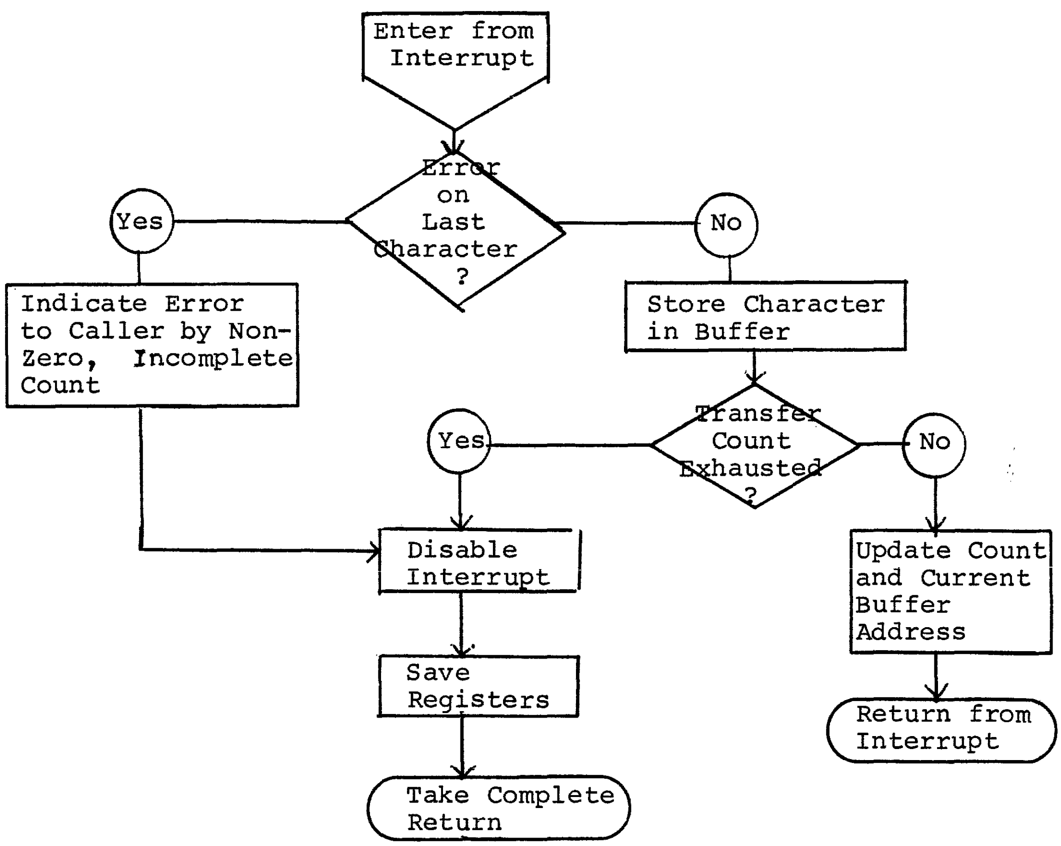
The OPEN function for the paper tape reader exists to give the user a means to ensure the reader is ready for operation (i.e., contains tape, is turned on, etc.). The OPEN routine tests the tape reader status register for an error indication. If such exists, an A002 message (Device Not Ready) is printed to the operator. The check is repeated following a return from the Diagnostic Print routine indicating that the operator has requested continuation. Because no interrupt is necessary to make this check, the routine merely removes the interim return address stored on the top of the processor stack by the calling sequence and takes the completion exit immediately (since this driver is for single-use only, there can be no queue for its services, hence it need take no action to cater for a queue situation).

1.3 TRANSFER

The TRANSFER entry initializes the driver and initiates the read of the first character. Initialization consists of storing the byte count (2 * Word Count) and buffer address from the calling DDB into the driver header positions reserved for them, and enabling the reader interrupt.

1.4 INTERRUPT SERVICE

Interrupt servicing is the heart of the paper tape reader driver. The following flow chart gives a detailed explanation of this function.



It should be particularly noted that an error during interrupt servicing signifying "Reader Off" or "Out of Tape" is considered an "End of Data" and is treated accordingly.

1.5 Program Listing

A complete assembly listing of the driver follows.

```
DV.PR    MACRO V004-14 13-SEP-72 03:09 PAGE 1


                    ;COPYRIGHT 1971, DIGITAL EQUIPMENT CORP., MAYNARD, MASS.

                    ;VERSION NUMBER:          V001A

                    ; PAPER TAPE READER DRIVER (PR)
                            .TITLE  DV.PR
                            .GLOBL  PR
             000000 R0=%0
             000001 R1=%1
             000002 R2=%2
             000003 R3=%3
             000004 R4=%4
             000005 R5=%5
             000006 SP=%6
             000007 PC=%7
                    ;  PREAMBLE
       00000 000000 PR:     .WORD    0                  ;DCURRENT DDCB OR 0
       00002    234         .BYTE    PR.BP              ; FACILITIES INDICATOR
       00003    000         .BYTE    0
       00004    002         .BYTE    2                  ; STANDARD BUFFER SIZE / 16.
       00005    056         .BYTE    PR.INT-PR          ; INTERRUPT ADDRESS
       00006    200         .BYTE    200                ; PRIORITY 4 INTERRUPT
       00007    170         .BYTE    PR.OPN-PR          ; DISPATCH OPEN
       00010    022         .BYTE    PR.TFR-PR          ; TRANSFER (IN)
       00011    000         .BYTE    0                  ; CLOSE
       00012    000         .BYTE    0                  ; SPECIAL FUNCTIONS
       00013    000         .BYTE    0        ; DUMMY
       00014 063320 PR.NAM: .RAD50   !PR!
       00016 000000 INTCNT: .WORD    0                  ; INTERNAL COUNT
       00020 000000 STOADD: .WORD    0                  ; STORE NEXT ADDRESS
                    ; MAIN DRIVER
                    ;  BEGIN TRANSFER
       00022 016700 PR.TFR: MOV      PR,R0              ; GET DDB
             177752
       00026 016004         MOV      10(R0),R4          ; PRESERVE USER COUNT
             000010
       00032 006304         ASL      R4                 ; BYTE COUNT
       00034 010467         MOV      R4,INTCNT
             177756
       00040 016067         MOV      6(R0),STOADD       ; SAVE BUFFER ADDRESS
             000006
             177752
       00046 052737         BIS      #101,@#PR.CSR      ; ENABLE INTERRUPT
             000101
             177550
       00054 000207         RTS      PC                 ;RETURN
                    ; THE PR IS DRIVEN BY THE FOLLOWING INTERRUPT ROUTINE
                    ;
       00056 005737 PR.INT: TST      @#PR.CSR           ; TEST FOR ERROR
             177550
       00062 100414         BMI      PR.ERR             ; YES
       00064 113777         MOVB     @#PR.BUF,@STOADD ; STORE CHARACTER
             177552
             177726
       00072 005267         INC      STOADD             ; UPDATE
             177722
       00076 005267         INC      INTCNT             ;  POINTERS
```

DEVICE NAME - RAD 50

SPACE

```
            177714
47 00102 001404           BEQ     PR.DNE
48 00104 052737           BIS     #101,@#PR.CSR      ; ENABLE
            000101
            177552
49 00112 000002           RTI                        ;  AND RETURN
50 00114          PR.ERR:
51 00114 013746 PR.DNE: MOV     @#PR.SAV,-(SP)     ; SET LP JSR
            000044
52 00120 004536           JSR     R5,@(SP)+
53 00122 105037 PR.DIS: CLRB    @#PR.CSR           ; DISABLE INTERRUPT
            177550
54 00126 016700           MOV     PR,R0              ; DDB ADDRESS
            177646
55 00132 016701           MOV     INTCNT,R1          ; REMAINING COUNT
            177660
56 00136 001405           BEQ     PR.FRT             ; NONE
57 00140 162701           SUB     #6,R1              ;  ROUNDED TO WORDS (AND TEAR)
            000006
58 00144 006201           ASR     R1
59 00146 010160           MOV     R1,16(R0)          ; RETURN RESULT TO CALLER
            000016
60 00152 000170 PR.FRT: JMP     @14(R0)            ; COMPLETION RETURN
            000014
61                ; OPEN ROUTINE:
62 00156 016746 PR.OPR: MOV     PR.NAM,-(SP)       ; ADDITIONAL INFO
            177632
63 00162 012746           MOV     #402,-(SP)         ;NOT READY - 1,2 ERR MSG
            000402
64 00166 000004           IOT
65 00170 005737 PR.OPN: TST     @#PR.CSR           ; TAPE READY
            177550
66 00174 100770           BMI     PR.OPR             ; NO
67 00176 005726           TST     (SP)+              ;CLEAR CALL FROM STACK
68 00200 016700           MOV     PR,R0              ;GET DDB ADDRESS
            177574
69 00204 000762           BR      PR.FRT             ; .... & TAKE COMPLETE RETN
70                ;
71          177552 PR.BUF=177552
72          177550 PR.CSR=177550
73          000234 PR.BP=234
74          000044 PR.SAV=44
75
76          000001'         .END
```

DV.PR   MACRO V004-14 13-SEP-72 03:09 PAGE 1+
SYMBOL TABLE

```
INTCNT   000016R          PC      =%000007        PR        000000RG
PR.BP  = 000234           PR.BUF= 177552          PR.CSR= 177550
PR.DIS   000122R          PR.DNE   000114R        PR.ERR   000114R
PR.FRT   000152R          PR.INT   000056R        PR.NAM   000014R
PR.OPN   000170R          PR.OPR   000156R        PR.SAV= 000044
PR.TER   000022R          R0      =%000000        R1      =%000001
R2      =%000002          R3      =%000003        R4      =%000004
R5      =%000005          SP      =%000006        STOADD   000020R

 ABS.   000000     000
        000206     001

ERRORS DETECTED:  0
FREE CORE:  19435. WORDS
,LP:<DT:PR
```

DEC-11-ODPPA-A-D

PDP-11

PCØ5 HIGH-SPEED PAPER TAPE PUNCH DRIVER

October 1972

SUPPLEMENT TO:

PDP-11 DEVICE DRIVER PACKAGE

DEC-11-ODDPA-A-D

MONITOR VERSION VØØ8

NOTE

This document **is** for information purposes only and is subject to change without notice. DEC assumes no responsibility for the use or reliability of its software on equipment which is not supplied by DEC.

NEW AND CHANGED INFORMATION

This manual documents the software as of Monitor version VØ8. It has been revised to include all new and changed material since Monitor version VØ4. Such material is indicated by vertical bars in the outside margin. Whole new pages are not so marked but are dated in the lower outside corner.

SOFTWARE
DISTRIBUTION
CENTER

PCØ5 HIGH-SPEED PAPER TAPE PUNCH DRIVER

The paper tape punch driver supplies the basic device dependent operating functions for the PDP-11 paper tape punch. To facilitate the device dependent operation of the I/O common routines, the paper tape punch driver processes blocks of data to be punched. The driver will process any size block (as given in the DDB) but for efficient operation a default (standard) block size of 32 words has been chosen.

The paper tape reader driver provides open, close, transfer, and interrupt servicing functions. The open and close functions cause the paper tape punch to punch two fanfolds of blank leader and trailer tape respectively. The transfer function causes the punching of the given block of data. Since the PDP-11 paper tape punch punches one character at a time, the interrupt servicing function provides the actual control of the punch for each of the other functions.

2.1 DESCRIPTION

The paper tape punch driver consists of two distinct parts: the standard driver table and the driver body.

The driver table contains the following information:

1. Facilities indicator - The facilities provided by the paper tape punch driver are:
   a) Single User
   b) Output only
   c) ASCII or Binary format
   d) Non-file Structured

2. 32 word standard buffer size

3. Run at priority 4

4. Internal information
   a) Trailer Indicator
   b) Internal byte count
   c) Internal (byte) buffer address

The code for the paper tape driver is organized as follows. The open, close, and transfer routines perform their initialization processes and control is transferred to the interrupt service routine for actual control of the data transfer. The initialization processes consist of setting the internal byte count, the beginning buffer address, and the trailer indicator (Ø implies open/close in process, 1 otherwise). The interrupt servicing routine is then called. Leader/trailer punching and actual transfer punching differ only in that the internal buffer address always points to a zero in the former case, and this pointer is incremented through the block in the later case. Upon <u>total</u> completion of the requested operation, the DDB completion return is taken; the DDB intermediate return occurs immediately upon initiation of the punching of the initial byte. At each interrupt the detection of an error (Punch Out of Tape) results in a request for an A002 message at the console (Device Not Ready). If a return from the Diagnostic Print routine occurs, indicating an operator request to continue, the function is again resumed.

2.2 Program Listing

A complete assembly listing of the driver follows.

```
DV.PP   MACRO V004-14 13-SEP-72 03:10 PAGE 1

                     ;COPYRIGHT 1971,1972, DIGITAL EQUIPMENT CORP., MAYNARD, MASS.

                     ;VERSION NUMBER:        V005A.005

                              .TITLE  DV.PP
                              .GLOBL  PP
            000000            R0=%0
            000001            R1=%1
            000002            R2=%2
            000003            R3=%3
            000004            R4=%4
            000005            R5=%5
            000006            SP=%6
            000007            PC=%7
                     ; PAPER TAPE PUNCH DRIVER (PP)
                     ;  PREAMBLE
                     ;
   00000 000000 PP:           .WORD   0                  ; CURRENT DDB OR 0
   00002    332               .BYTE   PP,PP              ; FACILITIES
   00003    000               .BYTE   0
   00004    002               .BYTE   2                  ; 32 WORD STD BUFFER
   00005    074               .BYTE   PP.INT-PP          ; TRANSFER ADDRESS
   00006    200               .BYTE   200                ; STATUS
   00007    206               .BYTE   PP.OPN-PP          ; RELATIVE ADDRESSES FOR OPEN
   00010    024               .BYTE   PP.TFR-PP          ; TRANSFER
   00011    206               .BYTE   PP.CLS-PP          ; CLOSE
   00012    000               .BYTE   0,0                ; SPF & SPARE
   00013    000
   00014 063200 PP.NAM: .RAD50  /PP/
   00016 000001 PP.TRL: .WORD   1                  ; TRAILER INDICATOR = 0
   00020 000000 PPCT:   .WORD   0                  ; INTERNAL COUNT
   00022 000000 PPBPT:  .WORD   0                  ; CURRENT BUFFER POINTER
```

```
32                 ;
33                 ; DRIVER BODY
34 00024 016700 PP.TFR: MOV     PP,R0              ; GET CURRENT DDB
         177750
35 00030 016067         MOV     6(R0),PPFPT        ; GET BUFFER POINTER
         000006
         177764
36 00036 016004         MOV     12(R0),R4          ; PRESERVE WORD COUNT
         000012
37 00042 006304         ASL     R4                 ; CONVERT TO BYTES
38 00044 010467         MOV     R4,PPCT            ;  AND SAVE
         177750
39 00050 112767         MOVB    #1,PP.TRL                   ; RESET TO TFR
         000001
         177740
40 00056 011646 PP.UEN: MOV     (SP),-(SP)         ; SIMULATE INTERRUPT
41 00060 013766         MOV     @#ST.ATS,2(SP)     ;  FROM JSR PC,XXX
         177776
         000002
42 00066 013737         MOV     @#PP.VCT,@#ST.ATS           ; RUN UNDER PUNCH STATUS
         000076
         177776
43 00074 005737 PP.INT: TST     @#PP.CSR           ; PUNCH OUT OF PAPER OR OFF
         177554
44 00100 100434         BMI     PP.ERR             ;  YES
45 00102 005767         TST     PPCT
         177712
46 00106 001416         BEQ     PP.DNE             ; ALREADY FINISHED
47 00110 005267         INC     PPCT               ; COUNT THIS ONE
         177704
48 00114 117737         MOVB    @PPFPT,@#PP.BRG    ; MOVE CHARACTER TO PUNCH
         177702
         177556
49 00122 105767         TSTB    PP.TRL             ; TRAILER OR NO
         177670
50 00126 001402         BEQ     PP.NOT             ; TRAILER
51 00130 005267         INC     PPFPT              ; NEXT ADDRESS OF BUF.
         177666
52 00134 052737 PP.NOT: BIS     #100,@#PP.CSR      ; ENABLE INTERRUPT
         000100
         177554
53 00142 000002         RTI                        ; RETURN
54 00144 013767 PP.DNE: MOV     @#PP.SAV,.+10      ; SAVE REGS FOR RETURN
         000244
         000002
55 00152 004537         JSR     R5,@#0
         000000
56 00156 005037         CLR     @#PP.CSR           ; DISABLE INTERRUPT
         177554
57 00162 016700 PP.IGN: MOV     PP,R0              ; CURRENT DDB
         177612
58 00166 000170         JMP     @14(R0)            ; COMPLETION RETURN
         000014
59 00172 012746 PP.ERR: MOV     #63200,-(SP)       ;SHOW DEVICE NAME
         063200
60 00176 012746         MOV     #402,-(SP)         ; PRINT 1-2 ERR MSG
         000402
61 00202 000004         IOT                        ;  NOT READY
62 00204 000733         BR      PP.INT
63 00206        PP.OPN:
64 00206 105267 PP.CLS: CLRB    PP.TRL             ; INDICATE TRAILER OPERATION
         177604
```

```
65 00212 010746             MOV      PC,-(SP)
66 00214 062716             ADD      #PP.TRL-.,@SP
         177602
67 00220 012667             MOV      (SP)+,PPFPT      ; SET BUFFER ADDRESS
         177576
68 00224 012767             MOV      #177524,PPCT     ; Z FOLDS TRAILER
         177524
         177566
69 00232 000711             BR       PP.UEN           ; NORMAL FROM HERE ON
```

```
1        177776 ST.ATS=177776
2        000076 PP.VCT=76
3        177554 PP.CSR=177554
4        177556 PP.BRG=177556
5        000044 PP.SAV=44
6        000332 PP.BP=332
7        000162'PP.SPF=PP.IGN
8        000001'        .END
```

```
DV.PP    MACRO V004-14 13-SEP-72 03:10 PAGE 2+
SYMBOL TABLE

PC      =%000007            PP       000000RG          PPCT     000020R
PPFPT    000022R            PP.BP  = 000332            PP.BRG=  177556
PP.CLS   000206R            PP.CSR=  177554            PP.DNE   000144R
PP.ERR   000172R            PP.IGN   000162R           PP.INT   000074R
PP.NAM   000014R            PP.NOT   000134R           PP.OPN   000206R
PP.SAV=  000044             PP.SPF=  000162R           PP.TFR   000024R
PP.TRL   000016R            PP.UEN   000056R           PP.VCT=  000076
R0      =%000000            R1      =%000001           R2      =%000002
R3      =%000003            R4      =%000004           R5      =%000005
SP      =%000006            ST.ATS=  177776
. ABS.   000000      000
         000234      001

ERRORS DETECTED:  0
FREE CORE:  19413. WORDS
,LP:<DT:PP
```

HOW TO OBTAIN SOFTWARE INFORMATION

Announcements for new and revised software, as well as programming notes, software problems, and documentation corrections are published by Software Information Service in the following newsletters.

Digital Software News for the PDP-8 and PDP-12
Digital Software News for the PDP-11
Digital Software News for 18-bit Computers

These newsletters contain information applicable to software available from Digital's Software Distribution Center. Articles in Digital Software News update the cumulative Software Performance Summary which is contained in each basic kit of system software for new computers. To assure that the monthly Digital Software News is sent to the appropriate software contract at your installation, please check with the Software Specialist or Sales Engineer at your nearest Digital office.

Questions or problems concerning Digital's software should be reported to the Software Specialist. In cases where no Software Specialist is available, please send a Software Performance Report form with details of the problems to:

Digital Equipment Corporation
Software Information Service
Programming Department
Maynard, Massachusetts 01754

These forms, which are provided in the software kit, should be fully filled out and accompanied by Teletype output as well as listings or tapes of the user program to facilitate a complete investigation. An answer will be sent to the individual and appropriate topics of general interest will be printed in the newsletter.

Orders for new and revised software manuals, additional Software Performance Report forms, and software price lists should be directed to the nearest Digital Field office or representative. USA customers may order directly from the Software Distribution Center in Maynard. When ordering, include the code number and a brief description of the software requested.

Digital Equipment Computer Users Society (DECUS) maintains a user library and publishes a catalog of programs as well as the DECUSCOPE magazine for its members and non-members who request it. For further information, please write to:

Digital Equipment Corporation
DECUS
Programming Department
Maynard, Massachusetts 01754

PDP-11 Device Driver Package
DEC-11-ODDPA-A-B
November 1972

READER'S COMMENTS

Digital Equipment Corporation maintains a continuous effort to improve the quality and usefulness of its publications. To do this effectively we need user feedback -- your critical evaluation of this manual.

Please comment on this manual's completeness, accuracy, organization, usability, and readability.

Did you find errors in this manual? If so, specify by page.

How can this manual be improved?

Other comments?

Please state your position. ______________________ Date: __________

Name: ______________________ Organization: ______________________

Street: ______________________ Department: ______________________

City: ______________ State: ______________ Zip or Country __________

Fold Here

Do Not Tear - Fold Here and Staple

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

digital

Digital Equipment Corporation
Software Information Services
Programming Department
Maynard, Massachusetts 01754