DEC-11-ODEVA-A-D

DOS/BATCH

DEVICE DRIVER INFORMATION

FOR THE DOS/BATCH OPERATING SYSTEM

Monitor Version VØ9

August 1973

For additional copies, order No. DEC-11-ODEVA-A-D from Digital Equipment Corporation, Software Distribution Center, Maynard, Massachusetts Ø1754.

pdp11
SOFTWARE
DISTRIBUTION
CENTER

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 considered when documenting subsequent manuals.

Associated documents:

DOS/BATCH Monitor
Programmer's Manual, DEC-11-OMPMA-A-D

DOS/BATCH User's Guide, DEC-11-OBUGA-A-D

DOS/BATCH Assembler (MACRO-11)
Programmer's Manual, DEC-11-LASMA-A-D

DOS/BATCH FORTRAN Compiler and Object Time System
Programmer's Manual, DEC-11-LFRTA-A-D

DOS/BATCH System Manager's Guide, DEC-11-OSMGA-A-D

DOS/BATCH File Utility Package (PIP)
Programmer's Manual, DEC-11-UDEBA-A-D

DOS/BATCH Debugging Program (ODT-11R)
Programmer's Manual, DEC-11-UDEBA-A-D

DOS/BATCH Linker (LINK)
Programmer's Manual, DEC-11-ULKAA-A-D

DOS/BATCH Librarian (LIBR)
Programmer's Manual, DEC-11-ULBAA-A-D

DOS/BATCH Text Editor (EDIT-11)
Programmer's Manual, DEC-11-UEDAA-A-D

DOS/BATCH File Compare Program (FILCOM)
Programmer's Manual, DEC-11-UFCAA-A-D

DOS/BATCH File Dump Program (FILDMP)
Programmer's Manual, DEC-11-UFLDA-A-D

DOS/BATCH Verification Program (VERIFY)
Programmer's Manual, DEC-11-UVERA-A-D

DOS/BATCH Disk Initializer (DSKINT)
Programmer's Manual, DEC-11-UDKIA-A-D

PREFACE

This document provides general information about the DOS/BATCH device drivers which handle I/O transfers between the PDP-11 and its peripheral devices. A sample listing of the Line Printer Driver is provided in Appendix B.

CONTENTS

		Page
CHAPTER 1	INTRODUCTION	1-1
CHAPTER 2	DRIVER FORMAT	2-1
	2.1 STRUCTURE	2-1
	2.1.1 Driver Interface Table	2-1
	2.1.2 Setup Routines	2-1
	2.1.3 Interrupt Servicing	2-2
	2.1.4 Error Handling	2-2
	2.2 INTERFACE TO THE DRIVER	2-2
	2.2.1 Control Interface	2-2
	2.2.2 Interrupt Interface	2-3
CHAPTER 3	STAND-ALONE USE	3-1
	3.1 DRIVER ASSEMBLED WITH PROGRAM	3-1
	3.1.1 Setting Interrupt Vector	3-1
	3.1.2 Parameter Table for Driver Call	3-2
	3.1.3 Calling the Driver	3-3
	3.1.4 User Registers	3-3
	3.1.5 Returns From Driver	3-4
	3.1.6 Irrecoverable Errors	3-5
	3.1.7 General Comment	3-6
	3.2 DRIVERS ASSEMBLED SEPARATELY	3-6
	3.3 DEVICE-INDEPENDENT USAGE	3-8
APPENDIX A	I/O DRIVERS WITHIN THE DOS/BATCH OPERATING SYSTEM	A-1
	A.1 DRIVER STRUCTURE	A-1
	A.2 MONITOR CALLING	A-2
	A.3 DRIVER ROUTINES	A-4
	A.3.1 TRANSFER	A-4
	A.3.2 Interrupt Servicing	A-5
	A.3.3 OPEN	A-6
	A.3.4 CLOSE	A-7
	A.3.5 SPECIAL	A-7
	A.4 DRIVERS FOR TERMINALS	A-8
APPENDIX B	SAMPLE LINE PRINTER DRIVER LISTING	B-1

CHAPTER 1

USING DEVICE DRIVERS OUTSIDE DOS/BATCH

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/BATCH. 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/BATCH or who wish to run programs outside DOS/BATCH control.

All the subroutines associated with one peripheral device form an entity known as a driver. This manual 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, which are supplements to this manual. 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 this document and applicable supplements.

CHAPTER 2

DRIVER FORMAT

2.1 STRUCTURE

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

A detailed description of a driver is found in Appendix A. This section describes driver interfaces.

2.1.1 Driver Interface Table

The first section of each driver is 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 return control to the program only when the whole block has been transferred or when it is unable to continue because there is no more data available.

---

* See DOS/BATCH Assembler (MACRO) Programmer's Manual for information on PIC.

The driver can also contain subroutines by which the calling program can request (1) start-up or shut-down action, such as leader or trailer functions for a paper tape punch, or (2) 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 driver routine to service device interrupts is particularly dependent upon the device hardware provisions 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 principle link between a calling program and any driver subroutine is the first word of the driver table (link word). 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 link word. 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 table can also act as a busy indicator; if it is Ø, the driver is not currently performing a task, but if it 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 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 the Monitor, it can be similarly used in other applications. Since the easiest way to use the driver is to assemble it with the program that 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 Ø-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:

   Ø-2  Function (octally 2=WRITE, 4=READ)⁴
   8-1Ø Unit (if Device can consist of several,
          e.g., DECtape)
   11   Direction for DECtape travel (Ø = 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 it 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 -heir content, i.e., ASCII or Binary. Some devices (e.g., Card Reader) may be handled differently depending on the mode for these, Bit Ø must also be set to indicate ASCII=Ø, 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 #1ØØ-1Ø3 backward from DECtape 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     #54Ø4,-(RØ)         ;SET READ REV. UNIT 3
        MOV     #-1Ø24.,-(RØ)       ;4 BLOCKS REQUIRED
        MOV     PC,-(RØ)            ;GET AND STORE
        ADD     #BUFFER-.,@RØ       ;...BUFFER ADDRESS
        MOV     #1Ø3,-(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 Ø-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). It accesses them via addresses in memory locations 44_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 Ø-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 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 RØ 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 Ø-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 2ØØØ on a device having a maximum of 1ØØØ. 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/BATCH, 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 DOS/BATCH and contains some code 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 .RAD5Ø 'DT' to obtain a specifically packed format used internally by DOS/BATCH. If the user wishes to keep the name, for instance, for identification purposes as discussed in section 3.3, .RAD5Ø 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 follows:

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

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

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-XPTSA-A-D, Chapter 6, provides the capability to continue 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
           .
           .
           .
  PGREND:
          .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 programs 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 2.2.1, 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 RØ 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 be 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...
          .
          .
          .
IOSUB:    MOV       @SP,RØ              ;PICK UP DRIVER ADDR
          MOV       R5,R1               ;SET UP 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 cause a queue for the driver's services under DOS/BATCH. 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 DOS/BATCH 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 mag-tape.

A.1 DRIVER STRUCTURE

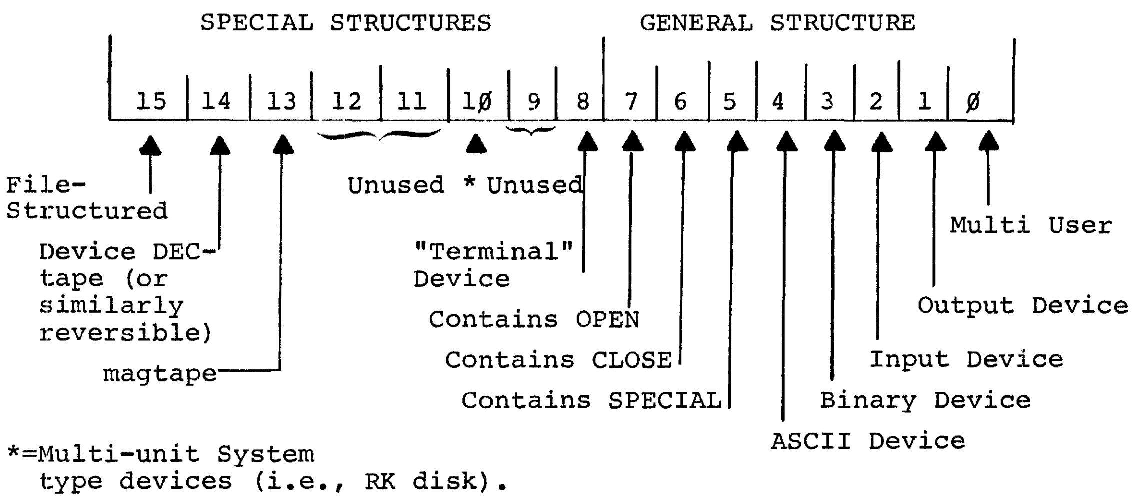
In order to provide a common interface to the monitor, all drivers must begin with a table of identifying information as follows:

DVR:

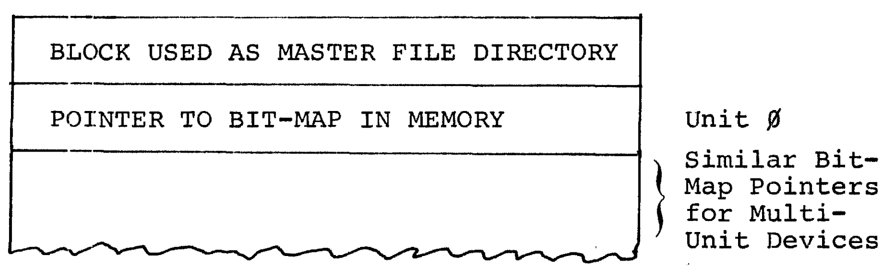
BUSY FLAG (initially Ø)	
FACILITY INDICATOR (expanded below)	
Offset to Interrupt Routine*	Standard Buffer Size in 16-word Units.
Offset to OPEN Routine*	Priority for Interrupt Service*
Offset to CLOSE Routine*	Offset to Transfer Routine*
Space	Offset to Special Functions*
DEVICE NAME (Packed Radix-5Ø)	

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:



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

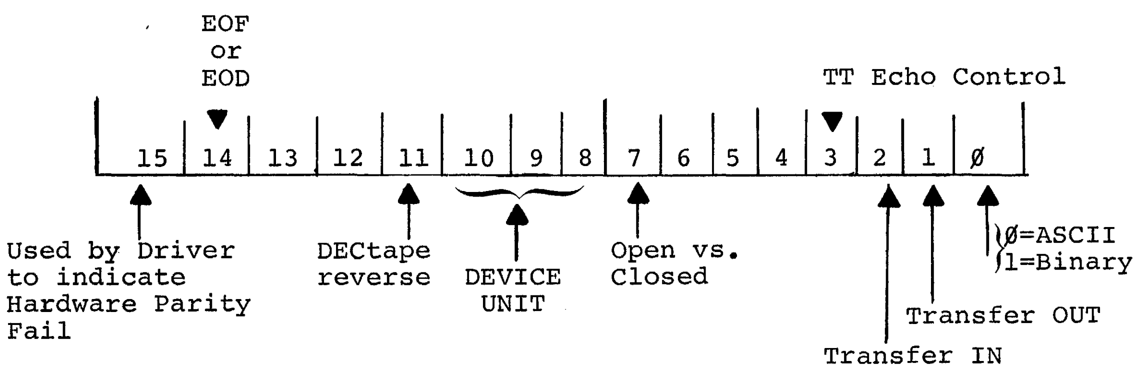
When a Monitor I/O processing routine needs to call the driver, it first sets up the parameters for the driver operation in relevant words of the appropriate DDB[1], as follows:

[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.

XYZ:

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

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 HSR 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.

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,Ø (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 thier 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 goind 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)+ to effect this. A multi-user driver, however, must allow for the possibility that it may be recalled to performe some new task waiting in a queue. This is shown 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.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     @1-(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 it 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 each 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, ie.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 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.

APPENDIX B

SAMPLE LINE PRINTER DRIVER LISTING

The following is a sample listing of a DOS/BATCH Device Driver. The actual driver is the LP11 Line Printer Driver (for device name LP:).

```
                ;       DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01
                ;       COPYRIGHT, 1973
                ;
                ;       DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY
                ;       FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT
                ;       WHICH IS NOT SUPPLIED BY DIGITAL EQUIPMENT CORPORATION.
                ;
                ;       VERSION NUMBER: V13.01
                ;
                ;       DATED:          MARCH 5, 1973
                ;
                ;       DEVICE DRIVER FOR THE LP11/LS11 LINE PRINTER(S)
                ;
                ;       DRIVER PARAMETERIZATION SYMBOLS
                ;               LP11, LS11, WIDTH, SPACES, SPREAD
                ;

                        .IF     NDF,LPTYP
                LPTYP   =       0
                        .ENDC
                        .IF     EQ,LPTYP
                        .TITLE  DV.LP0
        000001  LP11    =       1
        000012  SKIP2   =       12
                        .IFF
                        .IF     EQ,<LPTYP-1>
                        .TITLE  DV.LP1
                LS11    =       1
                SPREAD  =       1
                SKIP2   =       13
                        .IFF
                        .MERROR ;UNSUPPORTED LINE PRINTER
                        .ENDC
                        .ENDC

                        .IFNDF  WIDTH
                WIDTH   =       80.                     ; 80. COLUMN PRINTER DEFAULT
                        .ENDC

        000000  R0      =       %0
        000001  R1      =       %1
        000002  R2      =       %2
        000003  R3      =       %3
        000004  R4      =       %4
        000005  R5      =       %5
        000006  SP      =       %6
        000007  PC      =       %7

        000402  A002    =       402                     ; DIAGNOSTIC MESSAGE CODE

        000044  S.RSAV  =       44                      ; REGISTER SAVE (MONITOR SUPPORT
```

```
                                .GLOBL  LP
                                .IDENT  /13.01/

                        ;       DOS-11 DEVICE DRIVER'S STANDARDIZED INTERFACE

 000000 000000 LP:      .WORD   0                    ; USER'S DDB POINTER
                        .IFDF   LS11&SPREAD
                        .BYTE   362                  ; FACILITIES INDICATOR
                        .ENDC
                        .IFNDF  LS11&SPREAD
 000002    322          .BYTE   322                  ; FACILITIES INDICATOR
                        .ENDC
 000003    000          .BYTE   0                    ; SPECIAL STRUCTURES, NONE
 000004    003          .BYTE   <<WIDTH+37>/40>      ; STANDARD BUFFER SIZE
 000005    110          .BYTE   LP.INT-LP            ; INTERRUPT ENTRY OFFSET
 000006    200          .BYTE   200                  ; INTERRUPT PRIORITY 4
 000007    035          .BYTE   LP.OPN-LP            ; OPEN ENTRY OFFSET
 000010    060          .BYTE   LP.TRN-LP            ; TRAN ENTRY OFFSET
 000011    036          .BYTE   LP.CLS-LP            ; CLOSE ENTRY OFFSET
                        .IF     EQ,LPTYP
 000012    000          .BYTE   0
                        .IFF
                        .BYTE   LP.SPC-LP            ; SPECIAL ENTRY OFFSET
                        .ENDC
 000013    000          .BYTE   0                    ; SPARE
 000014 046600 LP.NAM:  .RAD50  /LP/                 ; DEVICE DRIVER'S NAME

        000200 LP.TRP   =       200                  ; INTERRUPT VECTOR'S ADDRESS
        177514 LP.CSR   =       177514               ; COMMAND/STATUS REGISTER
        177516 LP.DBR   =       177516               ; DATA BUFFER REGISTER

 000016 000120 LP.SIZ:  .WORD   WIDTH                ; THIS WORD IS SET BY THE INITIA
 000020 000133 UPPCAS:  .WORD   133                  ; SET TO THE HIGHER PRINT LIMIT
 000022 000000 OVPRNT:  .WORD   0                    ; SET TO TRUE WHEN OVER PRINTING
 000024 000000 LP.LIN:  .WORD   0                    ; ALREADY SENT (CHARACTERS)
 000026 000000 LP.BKS:  .WORD   0                    ; BLANK POSITIONS COUNTER
 000030 000000 LP.TCT:  .WORD   0                    ; TRANSFER CHARACTER COUNT
 000032 000000 LP.BAD:  .WORD   0                    ; BUFFER ADDRESS POINTER

 000034        LP.TOF:                               ; COMMAND DEVICE TO TOP-OF-FORM
                        .IFDF   LS11
                        .BYTE   21                   ; COMMAND DEVICE TO ON-LINE
                        .ENDC
 000034    015          .BYTE   15,14                ; CR, FF
 000035    014
                        .EVEN
                        .IFDF   LS11&SPREAD
               LP.FLG:  .WORD   0                    ; CHARACTER ELONGATION FLAG
                        .ENDC

        000040 LP.LOW   =       40                   ; PRINTABILITY, LOWER LIMIT
```

```
                              ;       OPEN PROCESSOR
  000036                      LP.OPN:
                              ;       CLOSE PROCESSOR
  000036                      LP.CLS:
  000036 004767                       JSR     PC,LP.STS          ; SIMULATE INTERRUPT
         000402
  000042 062701                       ADD     #LP.IOF-.,R1       ; R1 = PC (BY LP.STS)
         177772
  000046 010167                       MOV     R1,LP.BAD          ; INTERNAL BUFFER'S ADDRESS
         177760
                                      .IFDF   LS11
                                      MOV     #-3,LP.TCT         ; INITIALIZE TRANSFER COUNT
                                      .ENDC
                                      .IFNDF  LS11
  000052 010267                       MOV     R2,LP.TCT          ; R2 = -2 (BY LP.STS)
         177752
                                      .ENDC
                                      .IFDF   LS11&SPREAD
                                      CLR     LP.FLG             ; INITIALIZE ELONGATION FLAG
                                      .ENDC
  000056 000414                       BR      LP.INT             ; DISPATCH INTERNAL BUFFER

                                      .IFDF   LS11&SPREAD

                              ;       SPECIAL PROCESSOR
                              LP.SPC:
                                      MOV     2(R0),R1           ; R1 = FUNCTION BLOCK'S ADDRESS
                                      CMPB    #1,(R1)            ; LINE ELONGATION FUNCTION ?
                                      BNE     LP.S00             ; NO, IGNORE
                                      MOV     2(R1),LP.FLG       ; ENABLE/DISABLE ELONGATION
                              LP.S00: JMP     @14(R0)            ; EXIT VIA COMPLETION RETURN
                                      .ENDC

                              ;       TRAN PROCESSOR
  000060                      LP.TRN:
  000060 004767                       JSR     PC,LP.STS          ; SIMULATE AN INTERRUPT
         000430
  000064 016700                       MOV     LP,R0              ; R0 = USER'S DDB ADDRESS
         177710
  000070 016067                       MOV     6(R0),LP.BAD       ; RETAIN BUFFER'S ADDRESS
         000006
         177734
  000076 016067                       MOV     10(R0),LP.TCT      ; RETAIN DDB'S BYTE COUNT
         000010
         177724
  000104 006367                       ASL     LP.TCT             ;
         177720
```

```

                           ;       INTERRUPT PROCESSOR (VIA INTERRUPT VECTOR AT 200)
  000114                   LP.INT:
  000114 042737                    BIC     #100,@#LP.CSR    ; DISABLE INTERRUPT
         000100
         177514
  000116 002002                    BGE     LP.I0            ; SEGREGATE ERRORS
  000120 000167                    JMP     LP.ERR           ; ENTER ERROR PROCESSOR
         000052
  000124 005767    LP.I0:          TST     LP.TCT           ; ANY CHARACTERS REMAINING ?
         177700
  000130 001452                    BEQ     LP.DONE          ; NO, LINE COMPLETED
  000132 010446                    MOV     R4,-(SP)         ; SAVE REGISTERS
  000134 010346                    MOV     R3,-(SP)         ;
  000136 010246                    MOV     R2,-(SP)         ;
  000140 010146                    MOV     R1,-(SP)         ;
  000142 016704                    MOV     LP.BKS,R4        ; R4 = BLANK COUNTER
         177060
  000146 016703                    MOV     LP.LIN,R3        ; R3 = PRINT POSITION
         177052
  000152 016702                    MOV     LP.BAD,R2        ; R2 = BUFFER POINTER (ADDRESS)
         177054
  000156 112201    LP.I00:         MOVB    (R2)+,R1         ; *** ACCESS CHARACTER ***
  000160 001425                    BEQ     LP.DNP           ; NULL (0) IGNORED
  000162 120127    LP.I01:         CMPB    R1,#LP.LOW       ; PRINTABILITY CHECK
         000040
  000166 002442                    BLT     LP.I10           ; EXCEEDS LOWER LIMIT
                                   .IFDF   SPACES
                                   BGT     LP.I02           ; VALID CHARACTER, SO FAR
                                   INC     R4               ; BLANK (40) ISOLATED, COUNT
                                   BR      LP.TRT           ; ACCESS NEXT CHARACTER
                                   .ENDC
  000170 120167    LP.I02:         CMPB    R1,UPPCAS        ; PRINTABILITY CHECK
         177024
  000174 002110                    BGE     LP.I18           ; EXCEEDS UPPER LIMIT
  000176 005203    LP.I03:         INC     R3               ; PRINTER'S WIDTH EXCEEDED ?
  000200 003016                    BGT     LP.DNP           ; YES, DO NOT PRINT
  000202 032737    LP.I04:         BIT     #100200,@#LP.CSR; ACCESS ERROR/READY STATUS
         100200
         177514
  000210 100530                    BMI     LP.I22           ; ERROR INDICATION
  000212 001517                    BEQ     LP.I20           ; NOT READY INDICATION
  000214 005304                    DEC     R4               ; DECREMENT BLANK COUNTER
  000216 100404                    BMI     LP.I05           ; NOT PROCESSING BLANKS
  000220 112737                    MOVB    #40,@#LP.DBR     ; BLANK/HTAB EXPANSION PERFORMED
         000040
         177516
  000226 000763                    BR      LP.I03           ; CONTINUE PENDING COMPLETION
  000230 110137    LP.I05:         MOVB    R1,@#LP.DBR      ; *** PRINT CHARACTER ***
         177516
  000234 005004    LP.I06:         CLR     R4               ; INSURE NO BLANKS PENDING
  000236           LP.DNP:
  000236 005267    LP.TRT:         INC     LP.TCT           ; INCREMENT BUFFER'S CHARACTER
         177560
                                                            ; COUNTER, ANY MORE ?
  000242 001345                    BNE     LP.I00           ; YES
```

```
                               ;
                               ;                LINE COMPLETED
                               ;
       000244 105737                   TSTB     @#LP.CSR          ; DEVICE BUSY ?
              177514
       000250 100102                   BPL      LP.I21            ; YES
       000252 004567 LP.DNE:   JSR      R5,LP.SET         ; RESTORE TEMPORARIES
              040255
       000256 013746 LP.DON:   MOV      @#$.RSAV,-(SP)    ; SAVE REGISTERS
              000044
       000262 004536                   JSR      R5,@(SP)+         ;
       000264 016700                   MOV      LP,R0             ; R0 = USER'S DDB ADDRESS
              177516
       000270 000170                   JMP      @14(R0)           ; EXIT VIA COMPLETION RETURN
              000014

       000274 120127 LP.I10:   CMPB     R1,#11            ; HORIZONTAL TAB (11) ?
              000011
       000300 001010                   BNE      LP.I13            ; NO
                               ;
                               ;                HORIZONTAL TAB SIMULATION VIA BLANKS
                               ;
       000302 016746                   MOV      LP.SIZ,-(SP)      ;      PRINTER'S MAX WIDTH
              177510
                                       .IFDF    LS11&SPREAD
                                       TST      LP.FLG            ; ELONGATION ?
                                       BEQ      LP.I11            ; NO
                                       ASR      (SP)              ;   (PRINTER'S WIDTH)/2
                                       .ENDC
       000306 060316 LP.I11:   ADD      R3,(SP)           ;   - PRINT POSITION
                                       .IFDF    LS11&SPREAD
                                       BGE      LP.I12            ; NOT EXCEEDED PRINTER'S WIDTH
                                       CLR      LP.TCT            ; ELONGATION LINE TERMINATION
                                       BR       LP.DNE            ; EXIT
                                       .ENDC
       000310 060416 LP.I12:   ADD      R4,(SP)           ;   + BLANK COUNTER
       000312 052716                   BIS      #177770,(SP)      ;   ( MODULO 8 ) - 8
              177774
       000316 152604                   SUB      (SP)+,R4          ;   + BLANK COUNTER
                                                                  ; = BLANK COUNTER
       000320 000740                   BR       LP.TRT            ; ACCESS NEXT CHARACTER
```

```
000322 120127 LP.I13: CMPB    R1,#15            ; CARRIAGE-RETURN (15) ?
       000015
000326 003013         BGT     LP.I14            ; NO, ABOVE
000330 001014         BNE     LP.I15            ; NO, BELOW
000332 005767         TST     OVPRNT            ; PRINT THE CARRIAGE-RETURN ?
       177454
000336 001020         BNE     LP.I16            ; YES
000340 016703         MOV     LP.SIZ,R3         ; R3 = -( PRINTER'S WIDTH)
       177452
000344 005403         NEG     R3                ;
                      .IFDF   LS11&SPREAD
                      TST     LP.FLG            ; ELONGATION ENABLED ?
                      BEQ     LP.IXX            ; NO
                      ASR     R3                ; HALVE PRINTER'S WIDTH
                      MOV     R3,LP.FLG         ; RE-INITIALIZE THE FLAG
                      .ENDC
000346        LP.IXX:
000346 000732         BR      LP.I06            ; SUPPRESS CARRIAGE-RETURN
000350        LP.I14: .IFDF   LS11&SPREAD
                      TST     LP.FLG
                      BEQ     LP.IYY
                      CMPB    R1,#16
                      BEQ     LP.I04
              LP.IYY:
                      .ENDC
000350 120127         CMPB    R1,#22
       000022
000354 001015         BNE     LP.I17            ; NO
000356 012701         MOV     #SKIP2,R1         ; SUBSTITUTE APPROPRIATE CHAR
       000012
000362 120127 LP.I15: CMPB    R1,#12            ; LINEFEED (12) ?
       000012
000366 002410         BLT     LP.I17            ; NO, BELOW
000370 001403         BEQ     LP.I16            ; YES
000372 120127         CMPB    R1,#13            ; VERTICAL TAB (13) ?
       000013
000376 001717         BEQ     LP.ONP            ; YES, IGNORE IT !
                                                ; NO,  FORMFEED (14) ISOLATED
000400        LP.I16:
000400 016703         MOV     LP.SIZ,R3         ; R3 = -( PRINTER'S WIDTH )
       177412
000404 005403         NEG     R3                ;
                      .IFDF   LS11&SPREAD
                      TST     LP.FLG            ; ELONGATION ENABLED ?
                      BEQ     LP.I04            ; NO, PRINT CHARACTER
                      ASR     R3                ; HALVE PRINTER'S WIDTH
                      MOV     R3,LP.FLG         ; RE-INITIALIZE THE FLAG
                      .ENDC
000406 000673         BR      LP.I04            ; PRINT THE CHARACTER
```

```
000410 012701 LP.I17: MOV     #40,R1           ; UNPRINTABLE, BLANK SUBSTITUTIO
       000040
000414 000673         BR      LP.I03           ; PRINT A BLANK
000416 120127 LP.I18: CMPB    R1,#172          ; LOWER CASE ALPHABET ?
       000172
000422 003003         BGT     LP.I19           ; EXCEEDS
              ;
              ;               LOWER CASE TO UPPER CASE CONVERSION PERFORMED
              ;
000424 042701         BIC     #40,R1           ; CONVERSION PERFORMED
       000040
000430 000662         BR      LP.I03           ; PRINT CHARACTER
000432 120127 LP.I19: CMPB    R1,#177          ; RUBOUT (177) ?
       000177
000436 001677         BEQ     LP.UNP           ; YES, IGNORED
000440 126727         CMPB    UPPCAS,#137      ; UPPER CASE PERMITTED ?
       177354
       000137
000446 101253         BHI     LP.I03           ; YES, PRINT CHARACTER
000450 000757         BR      LP.I17           ; UNPRINTABLE, BLANK SUBSTITUTIO

000452 005303 LP.I20: DEC     R3               ; BACKUP PRINT POSITION
000454 005302         DEC     R2               ; BACKUP BUFFER POSITION
000456 004567 LP.I21: JSR     R5,LP.SET        ; RESTORE TEMPORARIES
       000052
000462 052737         BIS     #100,@#LP.CSR    ; ENABLE INTERRUPT
       000100
       177514
000470 000002         RTI                      ; EXIT FROM INTERRUPT

000472 005303 LP.I22: DEC     R3               ; BACKUP PRINT POSITION
000474 005302         DEC     R2               ; BACKUP BUFFER POSITION
000476 016746 LP.ERR: MOV     LP.NAM,-(SP)     ; DEVICE DRIVER'S MNEMONIC
       177312
000502 012746         MOV     #A002,-(SP)      ; MESSAGE CODE
       000402
000506 000004         IOT
000510 000167         JMP     LP.INT           ; TRY AGAIN
       177374
```

```
                        ;
                        ;                     INTERRUPT SIMULATOR
                        ;
  000514 012601 LP.STS: MOV       (SP)+,R1              ; RETURN PC
  000516 011646         MOV       (SP),-(SP)            ; OLD PC
  000520 005002         CLR       R2                    ; ADDRESS PS (-2)
  000522 014266         MOV       -(R2),2(SP)           ; OLD STATUS
         000002
  000526 013712         MOV       @#LP.TRP+2,(R2)       ; NEW STATUS
         000202
  000532 010107         MOV       R1,PC                 ; RETURN

  00534  010467 LP.SET: MOV       R4,LP.BKS             ; RESTORE TEMPORARIES
         177265
  00540  010367         MOV       R3,LP.LIN             ;
         177264
  00544  010267         MOV       R2,LP.BAD             ;
         177262
  00550  016604         MOV       10(SP),R4             ; RESTORE REGISTER 4
         000010
  00554  012666         MOV       (SP)+,6(SP)           ; RETAIN RETURN ADDRESS
         000006
  00560  012601         MOV       (SP)+,R1              ; RESTORE REGISTERS
  00562  012602         MOV       (SP)+,R2              ;
  00564  012603         MOV       (SP)+,R3              ;
  00566  000205         RTS       R5                    ; EXIT SUBROUTINE
         000001'        .END
```


SYMBOL TABLE

```
A002   = 000402        LP        000000RG      LPTYP  = 000000
LP.BAD   000032R       LP.BKS    000026R       LP.CLS   000036R
LP.CSR=  177514        LP.DBR=   177516        LP.DNE   000252R
LP.ONP   000236R       LP.DUN    000256R       LP.ERR   000476R
LP.INT   000110R       LP.IXX    000346R       LP.I0    000124R
LP.I00   000156R       LP.I01    000102R       LP.I02   000170R
LP.I03   000176R       LP.I04    000202R       LP.I05   000230R
LP.I06   000234R       LP.I10    000274R       LP.I11   000306R
LP.I12   000310R       LP.I13    000322R       LP.I14   000350R
LP.I15   000362R       LP.I16    000400R       LP.I17   000410R
LP.I18   000416R       LP.I19    000432R       LP.I20   000452R
LP.I21   000456R       LP.I22    000472R       LP.LIN   000024R
LP.LOW=  000040        LP.NAM    000014R       LP.OPN   000036R
LP.SET   000534R       LP.SIZ    000016R       LP.STS   000514R
LP.TCT   000030R       LP.TUF    000034R       LP.TRN   000060R
LP.TRP=  000200        LP.TRT    000236R       LP11   = 000001
OVPRNT   000022R       SKIP2  =  000012        S.RSAV=  000044
UPPCAS   000020R       WIDTH  =  000120
. ABS.   000000     000
         000570     001
ERRORS DETECTED:  0
FREE CORE:  49511. WORDS
,LP:/CR<DT:LP0NEW,V01
```

```
CROSS REFERENCE TABLE    S-1

AØØ2      1-49#   7-25
LP        2- 2#   2- 7#   2-16    2-18    2-19    2-2Ø    3-34    5- 9
LPTYP     1-18    1-21    2-21
LP.BAD    2-39#   3- 8#   3-35@   4-15    8-13@
LP.BKS    2-37#   4-13    8-11@
LP.CLS    2-26    3- 5#
LP.CSR    2-3Ø#   4- 4#   4-29    5- 4    7-19@
LP.DBR    2-31#   4-34#   4-35@
LP.DNE    5- 6#
LP.DNP    4-17    4-28    4-38#   6-3Ø    7-11
LP.DON    4- 8    5- 7#
LP.ERR    4- 6    7-24#
LP.INT    2-16    3-16    4- 3#   7-27
LP.IXX    6-14#
LP.IØ     4- 5    4- 7#
LP.IØØ    4-16#   4-41
LP.IØ1    4-18#
LP.IØ2    4-25#
LP.IØ3    4-27#   4-35    7- 2    7- 9    7-13
LP.IØ4    4-29#   6-41
LP.IØ5    4-33    4-36#
LP.IØ6    4-37#   6-15
LP.I1Ø    4-19    5-12#
LP.I11    5-23#
LP.I12    5-29#
LP.I13    5-13    6- 1#
LP.I14    6- 2    6-16#
LP.I15    6- 3    6-25#
LP.I16    6- 5    6-28    6-32#
LP.I17    6-24    6-27    7- 1#   7-14
LP.I18    4-26    7- 3#
LP.I19    7- 4    7-1Ø#
LP.I2Ø    4-31    7-16#
LP.I21    5- 5    7-18#
LP.I22    4-36    7-22#
LP.LIN    2-36#   4-14    8-12@
LP.LOW    2-32#   4-16
LP.NAM    2-27#   7-24
LP.OPN    2-18    3- 3#
LP.SET    5- 6    7-18    8-11#
LP.SIZ    2-33#   5-17    6- 5    6-33
LP.STS    3- 6    3-33    8- 4#
LP.TCT    2-38#   3-13@   3-36@   3-37@   4- 7    4-39@
LP.TOF    2-41#   3- 7
LP.TRN    2-19    3-32#
LP.TRP    2-29#   8- 8
LP.TRT    4-39#   5-33
LP11      1-23#
LS11      2- 8    2-11    2-42    2-47    3- 9    3-12    3-15    3-2Ø    5-18
          5-24    6- 8    6-13    6-35
OVPRNT    2-35#   6- 4
PC        1-47#   3- 6@   3-33@   8- 9@
RØ        1-4Ø#   3-34@   3-35    3-36    5- 9@   5-1Ø
R1        1-41#   3- 7@   3- 8    4-12    4-16@   4-18    4-25    4-36    5-12
          6- 1    6-23    6-25@   6-26    6-29    7- 1@   7- 3    7- 8@   7-1Ø
          8- 4@   8- 9    8-15@
R2        1-42#   3-13    4-11    4-15@   4-16    7-17@   7-23@   8- 6@   8- 7
          8- 8@   8-13    8-17@
```

CROSS REFERENCE TABLE S-2

R3	1-43#	4-10	4-14@	4-27@	5-23	6- 6@	6- 7@	6-33@	6-34@
	7-16@	7-22@	8-12	8-18@					
R4	1-44#	4- 9	4-13@	4-32@	4-37@	5-29	5-31@	8-11	8-14@
R5	1-45#	5- 6@	5- 8@	7-18@	8-19@				
SKIP2	1-24#	6-25							
SP	1-46#	4- 9@	4-10@	4-11@	4-12@	5- 7@	5- 8	5-17@	5-23@
	5-29@	5-30@	5-31	7-24@	7-25@	8- 4	8- 5@	8- 7@	8-14
	8-15@	8-16	8-17	8-18					
SPACES	4-20								
SPREAD	2- 8	2-11	2-47	3-15	3-20	5-18	5-24	6- 8	6-16
	5-35								
S.RSAV	1-51#	5- 7							
UPPCAS	2-34#	4-25	7-12						
WIDTH	1-36	2-15	2-33						
.	3- 7								

CROSS REFERENCE TABLE C-1

	55296
. ABS.	55296

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 included in each basic kit of system software for new computers. To assure that the monthly DIGITAL Software News is sent to the appropriate software contact 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. If no Software Specialist is available, please send a Software Performance Report form with details of the problems to:

Digital Equipment Corporation
Software Information Service
Software Engineering and Services
Maynard, Massachusetts 01754

These forms, which are provided in the software kit, should be fully completed and accompanied by terminal 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
Software Engineering and Services
Maynard, Massachusetts 01754

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 document.

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

____

____

____

____

____

How can this document be improved?

____

____

____

____

____

How does this document compare with other technical documents you have read?

____

____

____

____

____

Job Title_______________________ 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 Service
Software Engineering and Services
Maynard, Massachusetts 01754