PART 5

DOS/BATCH DEVICE DRIVERS

PART 5

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 device driver. This part 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 made available as part of 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 this document and applicable supplements.

PART 5

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).[1]

A detailed description of a driver is found in Chapter 5-4. 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. See Section 5-4.1 for a detailed description of the table.

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 required action by setting the device hardware registers appropriately. It returns to the calling program by a standard subroutine exit.

The main setup routines prepares for a data transfer to or from the device, using parameters supplied by the calling program. Normally, blocks of data are moved at each transfer. The driver returns 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.

[1]See Part 6 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, if appropriate, attempts this itself (as in the case of parity or timing failure on a bulk-storage device) or recalls 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 DOS/BATCH error handler via an IOT call with supporting information on the processor stack.

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 (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. The driver can place return status information (if needed) 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 listpointer, 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 $4ØØ_8$ has been set up correctly; the Monitor takes care of this. However, the driver table contains the information required to initialize the appropriate vector.

PART 5

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 Section 5-2.2.2, 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 contents 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 reference 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 Byte (at DRIVER+5) shows the follwing instruction:

```
        .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 Section 5-2.2.1. 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 units, 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 since its content will not vary. The driver can return information in this area; this will not corrupt the program data.

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. The driver may refer to the list again when it is recalled by an interrupt or when returning 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 sections.
[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 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.

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 the MEMORY START ADDRESS. It can then directly call the required driver subroutine by a normal JSR PC,xxxx call, where xxxx is the address of the driver subroutine.

As an example, the following position-independent code might appear in a program which wishes to read Blocks #1ØØ-103 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              ;GO TO 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
            .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 and 46_8.

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

The driver must also ensure that the 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 Section 5-3.1.3, 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; either 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 contents of its first word as a pointer to the parameter table (see Section 5-3.1.2).

Possibly, the driver has transferred only some of the data requested. In this case, it shows 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 Section 5-3.1.2, 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 Section 5-3.1.6).

[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 saves 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 PROGRAM
```

3.1.6 Irrecoverable Errors

All hardware errors other than those noted in the previous section 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 not turning on a paper tape reader or not setting the correct unit number on a DECtape transport. Once the operator has rectified the problem, the program could continue. Other errors, however, 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 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 depends upon the program. The following format takes advantage of the information supplied by the driver:

	(SP):	Return Address	
2	(SP):	Return Status	Stored by IOT call
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 expects a return following the IOT call in the case of recoverable errors but contains no provision for an IOT call following a return from irrecoverable errors.

3.1.7 General Comment

The source language of each driver has been written for use with DOS/BATCH and contains some code which is not 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 5-3.3, .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. The only requirement is that the start address of the driver should be known or be determinable by the program.

The example in Section 5-3.1.2 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 routines, as these also have corresponding offsets in the Table, as follows:

DRIVER+7	Open[1]
+10	Transfer
+11	Close[1]
+12	Special Functions[1]

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), 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 Section 5-3.1.3.

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

```
          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 5-3.1.3
           .
           .
          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 that 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            ;SET TO DRIVER 2
           .
           .
           .
DVR1SZ=n
PRGEND:
          .END
```

An alternative method may be to use the MACRO Assembler in association with the Linker program LINK, 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 .GLOBAL statement, e.g., .GLOBL DT.

3.3 DEVICE-INDEPENDENT USAGE

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 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 for each device. 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 5-2.2.1, 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,@R1              ;...AND GO TO IT
                                          ;FOLLOWED POSSIBLY BY
                                          ;WAIT AND COMPLETION
                                          ;PROCESSING
                                          ;RETURN TO COMMON OPERATION

NOOPEN:
```

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. Other table entries show the device name as identification and the number of words the device might normally expect to transfer at a time (in 16-word units). All of the information can be readily examined by the calling program, thus enabling the use of a common call sequence for any I/O operation, as illustrated in the example on the following page.

```
         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 take advantage of a feature mentioned earlier; the fact that when a driver is in use, its first word is 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 the first word of the driver is set to zero between driver tasks, then 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 in which the list indicated in the driver's first word is the first link.

The preceding paragraphs indicate possible ways of incorporating the available drivers into the type of environment for which they were designed. The user should carefully read the more detailed description of the driver structure in Chapter 5-4, 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, 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.

PART 5

CHAPTER 4

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 involves setting up the device hardware registers to cause the transfer and gaining control under the interrupt scheme of PDP-11, making allowance for peculiar device characters (e.g., conversion to or from ASCII if some special code is used).

The I/O driver must 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.

4.1 DRIVER STRUCTURE

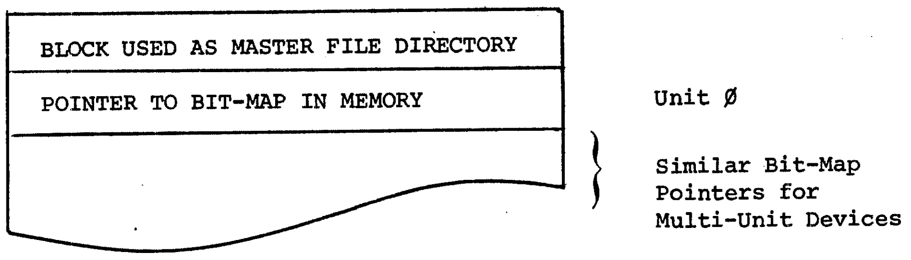
In order to provide a common interface to the Monitor, all drivers must begin with a table of identifying information as follows:

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-50)	

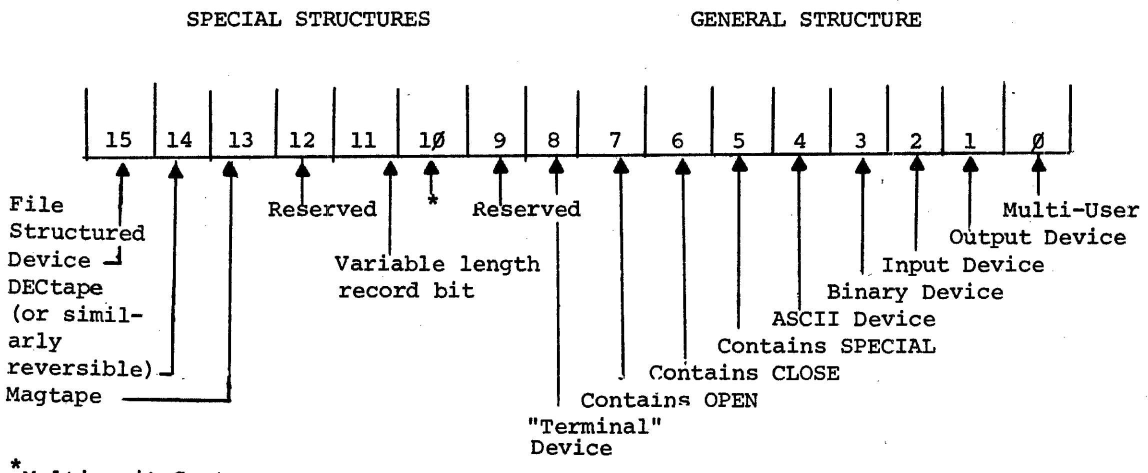
Offsets marked * enable the calling routine to indicate the routine required. The offsets are considered to be an unsigned value to be added to the start address of the driver. This may mean that with a 256-word maximum, the instruction referenced by the offset is a JMP or BR (routine).

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 that set up the transfer and control under the interrupt follow the table.

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



*Multi-unit System type devices (i.e., RK disk).

4.2 MONITOR CALLING

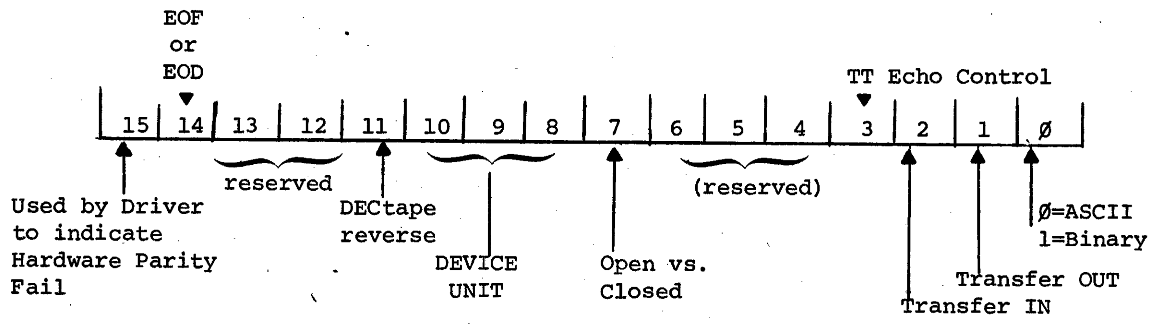
When a Monitor I/O processing routine needs to call the driver, it first sets up the parameters for the driver operation in relevant words of the appropriate DDB[1], as illustrated in the following table.

[1]Dataset Data Block - 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.

DDB:

—	(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, according to the direction and mode and 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). The Monitor routine then calls the Driver Queue Manager, using a JSR PC, S.CDB instruction.

The Driver Queue Manager refers to the Busy Flag (Word Ø of the driver table) to assure that the driver is free to accept the request. If the Busy Flag 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 saves 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 completly 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.

4.3 DRIVER ROUTINES

4.3.1 TRANSFER

The sole purpose of the TRANSFER routine is to set the device in motion. 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 the function of the routine. It must also enable the interrupt; however, it need not set the interrupt vectors as these are preset by the Monitor when the driver is brought into core. After the TRANSFER routine has activated the device, the routine returns to the calling processor by an RTS PC instruction.

4.3.2 Interrupt Servicing

The form of this routine depends upon the nature of the device. In most drivers it falls 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 case for such error is the detection of the end of the physical medium; the treatment for the error varies according to whether the device is providing input or accepting output. The calling program usually needs 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 requires 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,-(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 resume the transfer.

Normal transfer handling on blocked devices (or those like RF11 Disk which are treated as such) is simpler since the hardware takes care of individual words or bytes and the interrupt only occurs on completion.

Errors that indicate definite hardware malfunctions must generate diagnostic messages to the Operator, The only recourse is to start the program over, after the malfunction has been corrected.

There are some errors which the driver can attempt to overcome by restarting the transfer. Device parity failure on input is a common example. If one or more retries are unsuccessful, the driver should normally allow programmed recovery and indicate the error by Bit 15 of DDB word 5. Nevertheless, because the program may try to process the data despite the error, the driver should attempt to transfer the whole block requested if this has not already been effected. The remaining forms of errors must be processed according to the type of recovery deemed desirable.

Whether the routine uses processor registers for its operation depends on considerations of the core space saved against the time taken to save the user's contents. However, on completion (or error return to the Monitor), 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.

4.3.3 OPEN

This routine need be provided only for those devices that require some hardware initialization. It should not normally appear in drivers for devices used in a file oriented manner. The presence of the routine must be indicated by Bit 7 in the driver table Facility Indicator.

The OPEN routine 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 the TRANSFER routine in that it 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.

An input OPEN may 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 service only one dataset at a time (i.e., Bit Ø of their Facility Pattern word is set to Ø) and can never be queued, a TST (SP)+ instruction can effect this. However, a multi-user driver must allow for the possibility that it may be recalled to perform some new task waiting in a queue. This condition exists if the byte at DDB-3 is non-zero. In this case, the driver must simulate the interrupt expected by the completion process. This is accomplished by inserting a PS word on the stack above the return address supplied by the JSR of the Open request. A possible sequence for the interrupt simulation is illustrated below.

```
          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       @10(RØ)
```

4.3.4 CLOSE

The CLOSE routine is like the OPEN routine, in that it should provide for the possibility of some form of hardware shut-down, such as the punching of trailer code and that 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.

4.3.5 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. 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. Its exit method depends upon the necessity for an interrupt wait.

4.4 DRIVERS FOR TERMINALS

The rate of input from terminal devices normally reflects the typing skill of the operator. For both input and output, the amount of data to be transferred on each occasion may be a varying length, i.e., a line rather than a block of standard size. Furthermore, echoing input may conflict with interrupting output. As a result, drivers for such devices 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 the suppression of trailing nulls.

Input control remains the driver's responsibility since overcoming the rate problem requires circular buffering within the driver. This circular buffering feature allows the user type-ahead facilities. A subsequent input request may then be satisfied by data already in core. If the data is sufficient to fill the Monitor buffer, the driver awaits the next request before further transfer. If this is insufficient, the driver should operate as any other device and use subsequent interrupts to satisfy the Monitor's requests. Since the driver must 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. If the user requests .TRAN's which are not line oriented, the buffer size varies from the standard and the driver assumes the program requires a complete buffer before return.

PART 5

CHAPTER 5

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=Ø if LP11
                LPTYP   =       0               ;LPTYP=1 if LS11
                        .ENDC                   ;DEFAULT IS Ø
                        .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
 00002    322              .BYTE   322               ; FACILITIES INDICATOR
                           .ENDC
 00003    000              .BYTE   0                 ; SPECIAL STRUCTURES, NONE
 00004    003              .BYTE   <<WIDTH+37>/40>   ; STANDARD BUFFER SIZE
 00005    110              .BYTE   LP.INT-LP         ; INTERRUPT ENTRY OFFSET
 00006    200              .BYTE   200               ; INTERRUPT PRIORITY 4
 00007    036              .BYTE   LP.OPN-LP         ; OPEN ENTRY OFFSET
 00010    060              .BYTE   LP.TRN-LP         ; TRAN ENTRY OFFSET
 00011    036              .BYTE   LP.CLS-LP         ; CLOSE ENTRY OFFSET
                           .IF     EQ,LPTYP
 00012    000              .BYTE   0
                           .IFF
                           .BYTE   LP.SPC-LP         ; SPECIAL ENTRY OFFSET
                           .ENDC
 00013    000              .BYTE   0                 ; SPARE
 00014 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

 00016 000120 LP.SIZ:      .WORD   WIDTH             ; THIS WORD IS SET BY THE INITIA
 00020 000133 UPPCAS:      .WORD   133               ; SET TO THE HIGHER PRINT LIMIT
 00022 000000 OVPRNT:      .WORD   0                 ; SET TO TRUE WHEN OVER PRINTING
 00024 000000 LP.LIN:      .WORD   0                 ; ALREADY SENT (CHARACTERS)
 00026 000000 LP.BKS:      .WORD   0                 ; BLANK POSITIONS COUNTER
 00030 000000 LP.TCT:      .WORD   0                 ; TRANSFER CHARACTER COUNT
 00032 000000 LP.BAD:      .WORD   0                 ; BUFFER ADDRESS POINTER

 00034        LP.TOF:                                ; COMMAND DEVICE TO TOP-OF-FORM
                           .IFDF   LS11
                           .BYTE   21                ; COMMAND DEVICE TO ON-LINE
                           .ENDC
 00034    015              .BYTE   15,14             ; CR, FF
 00035    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
       000454
000042 062701                 ADD     #LP.TOF-.,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
 00052 010267                 MOV     R2,LP.TCT          ; R2 = -2 (BY LP.STS)
       177752
                              .ENDC
                              .IFDF   LS11&SPREAD
                              CLR     LP.FLG             ; INITIALIZE ELONGATION FLAG
                              .ENDC
 00056 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
 00060                LP.TRN:
 00060 004767                 JSR     PC,LP.STS          ; SIMULATE AN INTERRUPT
       000432
 00064 016700                 MOV     LP,R0              ; R0 = USER'S DDB ADDRESS
       177710
 00070 016067                 MOV     6(R0),LP.BAD       ; RETAIN BUFFER'S ADDRESS
       000006
       177734
 00076 016067                 MOV     10(R0),LP.TCT      ; RETAIN DDB'S BYTE COUNT
       000010
       177724
 00104 006367                 ASL     LP.TCT             ;
       177720
```

```
                      ;       INTERRUPT PROCESSOR (VIA INTERRUPT VECTOR AT 200)
  000110              LP.INT:
  000110 042737               BIC     #100,@#LP.CSR    ; DISABLE INTERRUPT
         000100
         177514
  000116 002002               BGE     LP.I0            ; SEGREGATE ERRORS
  000120 000167               JMP     LP.ERR           ; ENTER ERROR PROCESSOR
         000354
  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
   00134 010346               MOV     R3,-(SP)         ;
   00136 010246               MOV     R2,-(SP)         ;
   00140 010146               MOV     R1,-(SP)         ;
   00142 016704               MOV     LP.BKS,R4        ; R4 = BLANK COUNTER
         177660
   00146 016703               MOV     LP.LIN,R3        ; R3 = PRINT POSITION
         177652
   00152 016702               MOV     LP.BAD,R2        ; R2 = BUFFER POINTER (ADDRESS)
         177654
   00156 112201 LP.I00: MOVB    (R2)+,R1         ; *** ACCESS CHARACTER ***
   00160 001426               BEQ     LP.DNP           ; NULL (0) IGNORED
   00162 120127 LP.I01: CMPB    R1,#LP.LOW       ; PRINTABILITY CHECK
         000040
   00166 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
   00170 120167 LP.I02: CMPB    R1,UPPCAS        ; PRINTABILITY CHECK
         177624
   00174 002111               BGE     LP.I18           ; EXCEEDS UPPER LIMIT
   00176 005203 LP.I03: INC     R3               ; PRINTER'S WIDTH EXCEEDED ?
   00200 003016               BGT     LP.DNP           ; YES, DO NOT PRINT
   00202 032737 LP.I04: BIT     #100200,@#LP.CSR; ACCESS ERROR/READY STATUS
         100200
         177514
   00210 100531               BMI     LP.I22           ; ERROR INDICATION
   00212 001520               BEQ     LP.I20           ; NOT READY INDICATION
   00214 005304               DEC     R4               ; DECREMENT BLANK COUNTER
   00216 100404               BMI     LP.I05           ; NOT PROCESSING BLANKS
   00220 112737               MOVB    #40,@#LP.DBR     ; BLANK/HTAB EXPANSION PERFORMED
         000040
         177516
   00226 000763               BR      LP.I03           ; CONTINUE PENDING COMPLETION
   00230 110137 LP.I05: MOVB    R1,@#LP.DBR      ; *** PRINT CHARACTER ***
         177516
   00234 005004 LP.I06: CLR     R4               ; INSURE NO BLANKS PENDING
   00236        LP.DNP:
   00236 005267 LP.TRT: INC     LP.TCT           ; INCREMENT BUFFER'S CHARACTER
         177566
                                                       ; COUNTER, ANY MORE ?
   00242 001345               BNE     LP.I00           ; YES
```

```
                     ;
                     ;                  LINE COMPLETED
                     ;
  000244 105737                TSTB     @#LP.CSR          ; DEVICE BUSY ?
         177514
  000250 100103                BPL      LP.I21            ; YES
  000252 004567 LP.DNE: JSR    R5,LP.SET          ; RESTORE TEMPORARIES
         000260
  000256 013746 LP.DON: MOV    @#S.RSAV,-(SP)     ; SAVE REGISTERS
         000044
  000262 004536                JSR      R5,@(SP)+         ;
  000264 016700                MOV      LP,R0             ; R0 = USER'S DDB ADDRESS
         177510
   00270 000170                JMP      @14(R0)           ; EXIT VIA COMPLETION RETURN
         000014

   00274 120127 LP.I10: CMPB   R1,#11             ; HORIZONTAL TAB (11) ?
         000011
   00300 001010                BNE      LP.I13            ; NO
                     ;
                     ;                  HORIZONTAL TAB SIMULATION VIA BLANKS
                     ;
   00302 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
   00306 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
   00310 060416 LP.I12: ADD    R4,(SP)            ;    + BLANK COUNTER
   00312 052716                BIS      #177770,(SP)      ;    ( MODULO 8 ) - 8
         177770
   00316 162604                SUB      (SP)+,R4          ;    + BLANK COUNTER
                                                          ; = BLANK COUNTER
   00320 000746                BR       LP.TRT            ; ACCESS NEXT CHARACTER

```

```
000322 120127 LP.I13: CMPB      R1,#15            ; CARRIAGE-RETURN (15) ?
       000015
000326 003010         BGT       LP.I14            ; NO, ABOVE
000330 001014         BNE       LP.I15            ; NO, BELOW
000332 005767         TST       OVPRNT            ; PRINT THE CARRIAGE-RETURN ?
       177464
000336 001021         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
00346          LP.IXX:
00346  000732         BR        LP.I06            ; SUPPRESS CARRIAGE-RETURN
00350          LP.I14: .IFDF    LS11&SPREAD
                      TST       LP.FLG
                      BEQ       LP.IYY
                      CMPB      R1,#16
                      BEQ       LP.I04
               LP.IYY:
                      .ENDC
00350  120127         CMPB      R1,#22
       000022
00354  001016         BNE       LP.I17            ; NO
00356  012701         MOV       #SKIP2,R1         ; SUBSTITUTE APPROPRIATE CHAR
       000012
00362  120127  LP.I15: CMPB     R1,#12            ; LINEFEED (12) ?
       000012
00366  002411         BLT       LP.I17            ; NO, BELOW
00370  001404         BEQ       LP.I16            ; YES
00372  120127         CMPB      R1,#13            ; VERTICAL TAB (13) ?
       000013
00376  001717         BEQ       LP.DNP            ; YES, IGNORE IT !
00400  000400         BR        LP.I16            ; NO,  FORMFEED (14) ISOLATED
00402          LP.I16:
00402  016703         MOV       LP.SIZ,R3         ; R3 = -( PRINTER'S WIDTH )
       177410
00406  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
00410  000674         BR        LP.I04            ; PRINT THE CHARACTER
```

```
000412 012701 LP.I17: MOV     #40,R1             ; UNPRINTABLE, BLANK SUBSTITUTIO
       000040
000416 000667         BR      LP.I03             ; PRINT A BLANK
000420 120127 LP.I18: CMPB    R1,#172            ; LOWER CASE ALPHABET ?
       000172
000424 003003         BGT     LP.I19             ; EXCEEDS
              ;
              ;               LOWER CASE TO UPPER CASE CONVERSION PERFORMED
              ;
000426 042701         BIC     #40,R1             ; CONVERSION PERFORMED
       000040
000432 000661         BR      LP.I03             ; PRINT CHARACTER
 00434 120127 LP.I19: CMPB    R1,#177            ; RUBOUT (177) ?
       000177
 00440 001676         BEQ     LP.DNP             ; YES, IGNORED
 00442 126727         CMPB    UPPCAS,#137        ; UPPER CASE PERMITTED ?
       177352
       000137
 00450 101252         BHI     LP.I03             ; YES, PRINT CHARACTER
 00452 000757         BR      LP.I17             ; UNPRINTABLE, BLANK SUBSTITUTIO

 00454 005303 LP.I20: DEC     R3                 ; BACKUP PRINT POSITION
 00456 005302         DEC     R2                 ; BACKUP BUFFER POSITION
 00460 004567 LP.I21: JSR     R5,LP.SET          ; RESTORE TEMPORARIES
       000052
 00464 052737         BIS     #100,@#LP.CSR      ; ENABLE INTERRUPT
       000100
       177514
 00472 000002         RTI                        ; EXIT FROM INTERRUPT

 00474 005303 LP.I22: DEC     R3                 ; BACKUP PRINT POSITION
 00476 005302         DEC     R2                 ; BACKUP BUFFER POSITION
 00500 016746 LP.ERR: MOV     LP.NAM,-(SP)       ; DEVICE DRIVER'S MNEMONIC
       177310
 00504 012746         MOV     #A002,-(SP)        ; MESSAGE CODE
       000402
 00510 000004         IOT
 00512 000167         JMP     LP.INT             ; TRY AGAIN
       177372
```

```
                ;
                ;                  INTERRUPT SIMULATOR
                ;
000516 012601 LP.STS: MOV          (SP)+,R1          ; RETURN PC
000520 011646         MOV          (SP),-(SP)        ; OLD PC
000522 005002         CLR          R2                ; ADDRESS PS (=2)
000524 014266         MOV          -(R2),2(SP)       ; OLD STATUS
       000002
000530 013712         MOV          @#LP.TRP+2,(R2)   ; NEW STATUS
       000202
000534 010107         MOV          R1,PC             ; RETURN

00536  010467 LP.SET: MOV          R4,LP.BKS         ; RESTORE TEMPORARIES
       177264
00542  010367         MOV          R3,LP.LIN         ;
       177256
00546  010267         MOV          R2,LP.BAD         ;
       177260
00552  016604         MOV          10(SP),R4         ; RESTORE REGISTER 4
       000010
00556  012666         MOV          (SP)+,6(SP)       ; RETAIN RETURN ADDRESS
       000006
00562  012601         MOV          (SP)+,R1          ; RESTORE REGISTERS
00564  012602         MOV          (SP)+,R2          ;
00566  012603         MOV          (SP)+,R3          ;
00570  000205         RTS          R5                ; EXIT SUBROUTINE
       000001'        .END
```

```
SYMBOL TABLE

AØØ2   = ØØØ4Ø2          BFSHFT= Ø1ØØØØ          BLANK = ØØØØ4Ø
BSLSH  = ØØØ134          CR    = ØØØØ15          DDBADR= ØØØØØ6
DDBBLK= ØØØØØ4           DDBCNT= ØØØØ1Ø          DDBCRT= ØØØØ14
DDBDVA= 177776           DDBSTS= ØØØØ12          DDBULA= ØØØØØ2
DDBUNT= ØØØØ13           DITBFS= ØØØØØ4          DITBMP= ØØØØ16
DITBSY= ØØØØØØ           DITFAC= ØØØØØ2          DITINT= ØØØØØ5
DITMFD= ØØØØ14           DITNAM= ØØØØ12          DITOPN= ØØØØØ7
DITPRI= ØØØØØ6           DITSPF= ØØØØ11          DITXFR= ØØØØ1Ø
EMTINT= ØØØØØ6           EMTRET= ØØØØ14          EMTVAL= 104ØØØ
EMTVEC= ØØØØ3Ø           FTCOM = ØØØØØ1          FTDOS = ØØØØØ1
FTMUO = ØØØØØ1           FTRPG = ØØØØØ1          FTRPO3= ØØØØØ1
FØØ1   = ØØ14Ø1          FØØ2   = ØØ14Ø2         FØØ3   = ØØ14Ø3
FØØ5   = ØØ14Ø5          FØØ7   = ØØ14Ø7         FØ11   = ØØ1411
FØ12   = ØØ1412          FØ17   = ØØ1417         FØ24   = ØØ1424
FØ42   = ØØ1442          FØ5Ø   = ØØ145Ø         FØ52   = ØØ1452
KSBSIZ= ØØØ4ØØ           LF     = ØØØØ12         LP       ØØØØØØRG
LPTYP = ØØØØØØ           LP.BAD   ØØØØ32R        LP.BKS   ØØØØ26R
LP.CLS   ØØØØ36R         LP.CSR= 177514          LP.DBR= 177516
LP.DNE   ØØØ252R         LP.DNP   ØØØ236R        LP.DON   ØØØ256R
LP.ERR   ØØØ5ØØR         LP.INT   ØØØ11ØR        LP.IXX   ØØØ346R
LP.IØ    ØØØ124R         LP.IØØ   ØØØ156R        LP.IØ1   ØØØ162R
LP.IØ2   ØØØ17ØR         LP.IØ3   ØØØ176R        LP.IØ4   ØØØ2Ø2R
LP.IØ5   ØØØ23ØR         LP.IØ6   ØØØ234R        LP.I1Ø   ØØØ274R
LP.I11   ØØØ3Ø6R         LP.I12   ØØØ31ØR        LP.I13   ØØØ322R
LP.I14   ØØØ35ØR         LP.I15   ØØØ362R        LP.I16   ØØØ4Ø2R
LP.I17   ØØØ412R         LP.I18   ØØØ42ØR        LP.I19   ØØØ434R
LP.I2Ø   ØØØ454R         LP.I21   ØØØ46ØR        LP.I22   ØØØ474R
LP.LIN   ØØØØ24R         LP.LOW= ØØØØ4Ø          LP.NAM   ØØØØ14R
LP.OPN   ØØØØ36R         LP.SET   ØØØ536R        LP.SIZ   ØØØØ16R
LP.STS   ØØØ516R         LP.TCT   ØØØØ3ØR        LP.TOF   ØØØØ34R
LP.TRN   ØØØØ6ØR         LP.TRP= ØØØ2ØØ          LP.TRT   ØØØ236R
LP11   = ØØØØØ1          MSBSIZ= ØØ1ØØØ          OVLØØ6= ØØØØØ2
OVLØ16= ØØØØØ6           OVPRNT   ØØØØ22R        OV1Ø61= ØØØØ12
OV2Ø61= ØØØØ12           PATSIZ= ØØØØ3Ø          PRI4   = ØØØ2ØØ
PRI7   = ØØØ34Ø          PS     = 177776         PSPRIO= 177437
RPBIT = ØØ4ØØØ           RPO2SZ= ØØØØ2Ø          RUBOUT= ØØØ177
SKIP2 = ØØØØ12           SMBSIZ= ØØØØ4Ø          STMASK= 10707Ø
S.RSAV= ØØØØ44           TABCH = ØØØØ11          UPPCAS   ØØØØ2ØR
V.CDB = ØØØØ5Ø           V.CDQ = ØØØØ52          V.GTB = ØØØØ54
V.RLB = ØØØØ56           V.RRES= ØØØØ46          V.RSAV= ØØØØ44
V.SVT = ØØØØ4Ø           V.XIT = ØØØØ42          WIDTH = ØØØ12Ø
XFTCOM= ØØØØØØ           XFTDOS= ØØØØØØ          XFTMUO= ØØØØØØ
XFTRPG= ØØØØØØ           $$PASS= ØØØØØØ
. ABS.   ØØØØØØ      ØØØ
         ØØØ572      ØØ1
ERRORS DETECTED:  Ø
FREE CORE:  15Ø39. WORDS
,LP:LPØ/CRF<SY:PRAMTR/NL,SYSMAC,FEATSW,DK1:LPØ[2ØØ,2ØØ]/LI/LI:ME
```

```
CROSS REFERENCE TABLE   S-1

AØØ2       1- 83#   4- 49#  1Ø- 25
BFSHFT     1-1Ø2#
BLANK      1-123#
BSLSH      1-124#
CR         1-12Ø#
DDBADR     1- 54#
DDBBLK     1- 53#
DDBCNT     1- 55#
DDBCRT     1- 58#
DDBDVA     1- 51#   1-1Ø6#
DDBSTS     1- 56#
DDBULA     1- 52#   1-1Ø5#
DDBUNT     1- 57#
DITBFS     1- 36#
DITBMP     1- 45#
DITBSY     1- 34#
DITFAC     1- 35#
DITINT     1- 38#
DITMFD     1- 44#
DITNAM     1- 43#
DITOPN     1- 4Ø#
DITPRI     1- 39#
DITSPF     1- 42#
DITXFR     1- 41#
EMTINT     1- 63#
EMTRET     1- 62#
EMTVAL     1- 61#
EMTVEC     1- 64#
FTCOM      3-  6#
FTDOS      3- 12#
FTMUO      3-  8#
FTRPG      3- 1Ø#
FTRPO3     3- 16#
FØØ1       1- 81#
FØØ2       1- 8Ø#
FØØ3       1- 85#
FØØ5       1- 82#
FØØ7       1- 88#
FØ11       1- 86#
FØ12       1- 87#
FØ17       1- 89#
FØ24       1- 91#
FØ42       1- 92#
FØ5Ø       1- 84#
FØ52       1- 9Ø#
KSBSIZ     2-16Ø#
LF         1-121#
LP         5-  2#   5-  7#   5- 16    5- 18    5- 19    5- 2Ø    6- 34
           8-  9
LPTYP      4- 18    4- 21    5- 21
LP.BAD     5- 39#   6-  8@   6- 35@   7- 15   11- 13@
LP.BKS     5- 37#   7- 13   11- 11@
LP.CLS     5- 2Ø    6-  5#
LP.CSR     5- 3Ø#   7-  4@   7- 29    8-  4   1Ø- 19@
LP.DBR     5- 31#   7- 34@   7- 36@
LP.DNE     8-  6#
LP.DNP     7- 17    7- 28    7- 38#   9- 3Ø   1Ø- 11
LP.DON     7-  8    8-  7#
```

```
CROSS REFERENCE TABLE   S-2

LP.ERR   7-  6   10- 24#
LP.INT   5- 16    6- 18    7-  3#  10- 27
LP.IXX   9- 14#
LP.I0    7-  5    7-  7#
LP.I00   7- 16#   7- 41
LP.I01   7- 18#
LP.I02   7- 25#
LP.I03   7- 27#   7- 35   10-  2   10-  9   10- 13
LP.I04   7- 29#   9- 41
LP.I05   7- 33    7- 36#
LP.I06   7- 37#   9- 15
LP.I10   7- 19    8- 12#
LP.I11   8- 23#
LP.I12   8- 29#
LP.I13   8- 13    9-  1#
LP.I14   9-  2    9- 16#
LP.I15   9-  3    9- 26#
LP.I16   9-  5    9- 28    9- 31    9- 32#
LP.I17   9- 24    9- 27   10-  1#  10- 14
LP.I18   7- 26   10-  3#
LP.I19  10-  4   10- 10#
LP.I20   7- 31   10- 16#
LP.I21   8-  5   10- 18#
LP.I22   7- 30   10- 22#
LP.LIN   5- 36#   7- 14   11- 12@
LP.LOW   5- 51#   7- 18
LP.NAM   5- 27#  10- 24
LP.OPN   5- 18    6-  3#
LP.SET   8-  6   10- 18   11- 11#
LP.SIZ   5- 33#   8- 17    9-  6    9- 33
LP.STS   6-  6    6- 33   11-  4#
LP.TCT   5- 38#   6- 13@   6- 36@   6- 37@   7-  7    7- 39@
LP.TOF   5- 41#   6-  7
LP.TRN   5- 19    6- 32#
LP.TRP   5- 29#  11-  8
LP.TRT   7- 39#   8- 33
LP11     4- 23#
LS11     5-  8    5- 11    5- 42    5- 47    6-  9    6- 12    6- 15
         6- 20    8- 18    8- 24    9-  8    9- 16    9- 35
MSBSIZ   2-159#
OVL006   1-108#
OVL016   1-107#
OVPRNT   5- 35#   9-  4
OV1061   1-109#
OV2061   1-110#
PATSIZ   1-116#
PC       1- 14#   4- 47#   6-  6@   6- 33@  11-  9@
PRI4     1- 74#
PRI7     1- 75#
PS       1- 72#
PSPRIO   1- 73#
RPBIT    1-103#
RP02SZ   1-104#
RUBOUT   1-125#
R0       1-  7#   4- 40#   6- 34@   6- 35    6- 36    8-  9@   8- 10
R1       1-  8#   4- 41#   6-  7@   6-  8    7- 12    7- 16@   7- 18
         7- 25    7- 36    8- 12    9-  1    9- 23    9- 25@   9- 26
         9- 29   10-  1@  10-  3   10-  8@  10- 10   11-  4@  11-  9
```

```
            11- 16@
R2           1-  9#    4- 42#    6- 13     7- 11     7- 15@    7- 16    10- 17@
            10- 23@   11-  6@   11-  7    11-  8@   11- 13    11- 17@
R3           1- 10#    4- 43#    7- 10     7- 14@    7- 27@    8- 23     9-  6@
             9-  7@    9- 33@    9- 34@   10- 16@   10- 22@   11- 12    11- 18@
R4           1- 11#    4- 44#    7-  9     7- 13@    7- 32@    7- 37@    8- 29
             8- 31@   11- 11    11- 14@
R5           1- 12#    4- 45#    8-  6@    8-  8@   10- 18@   11- 19@
SKIP2        4- 24#    9- 25
SMBSIZ       2-158#    2-159     2-160
SP           1- 13#    4- 46#    7-  9@    7- 10@    7- 11@    7- 12@    8-  7@
             8-  8     8- 17@    8- 23@    8- 29@    8- 30@    8- 31    10- 24@
            10- 25@   11-  4    11-  5@   11-  7@   11- 14    11- 15@   11- 16
            11- 17    11- 18
SPACES       7- 20
SPREAD       5-  8     5- 11     5- 47     6- 15     6- 20     8- 18     8- 24
             9-  8     9- 16     9- 35
STMASK       1- 93#
S.RSAV       4- 51#    8-  7
TABCH        1-122#
UPPCAS       5- 34#    7- 25    10- 12
V.CDB        1- 25#
V.CDW        1- 26#
V.GTB        1- 27#
V.RLB        1- 28#
V.RRES       1- 24#
V.RSAV       1- 23#
V.SVT        1- 21#
V.XIT        1- 22#
WIDTH        4- 36     5- 15     5- 33
XFTCOM       3-  7#
XFTDOS       3- 13#
XFTMUD       3-  9#
XFTRPG       3- 11#
$$PASS       2-177#
             6-  7
```

```
CROSS REFERENCE TABLE   M=1

AERROR   2= 31#
CALL     2= 73#
CALL5    2= 69#
CHKPNT   2= 96#
DIAL     2=324#
ERROR    2= 18#
FERROR   2= 28#
FREMSB   2=251#
GETBUF   2=124#
IERROR   2= 39#
IOTERR   2= 24#
MODEND   2=232#
MODSTA   2=179#
MOVMSB   2=273#
MOVSEG   2=297#
OUTMSB   2=259#
OVLNAM   2=222#
POP      2= 63#
PUSH     2= 57#
QDRVR    2=146#
RELBUF   2=138#
RESREG   2=107#
RETEMT   2=280#
RETURN   2= 77#
SAVREG   2=102#
SERROR   2= 35#
SETABS   2=165#
STDEV    2=333#
SWPCAL   2= 84#
WERROR   2= 43#




CROSS REFERENCE TABLE   C=1
'
         55520
. ABS.   55520
```