RT-11

Software Support Manual

DEC-11-ORPGA-A-D

pdp11

digital

RT-11

Software Support Manual

DEC-11-ORPGA-A-D

Order additional copies as directed on the Software Information page at the back of this document.

digital equipment corporation · maynard. massachusetts

First Printing November 1973

The HOW TO OBTAIN SOFTWARE INFORMATION page, located at the back of this document, explains the various services available to DIGITAL software users.

The postage prepaid READER's COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

PREFACE

The RT-11 Software Support manual covers the internal description of the RT-11 Software System including the core layout, and system device. Details on the EMT structure, I/O queueing system and monitor tables are also covered. It is assumed that the reader has already read the RT-11 System Reference Manual (DEC-11-ORUGA-A-D).

CONTENTS

FIGURES

1.0 OVERVIEW

RT-11 is designed to be a small, fast system useful for program development and applications programming. The system can operate in any PDP-11 family computer with between 8 and 28K words of core memory and a mass storage device (disk or DECtape). In addition, RT-11 can utilize a large number of PDP-11 peripheral devices in a device independent manner. RT-11 combines an easy to use, versatile set of monitor functions with a simple, fast directory structure to produce a system which is useful for both program development and applications jobs. The system supports a comprehensive set of programs to aid easy software development. These include a text editor, MACRO assembler, a linker, a peripheral interchange program, and a debugging tool (ODT).

The monitor itself provides the capability for performing event driven, real-time I/O. RT-11 is fully interrupt driven, with many facilities for using overlapped I/O and computation. These and other features make programming real-time applications very easy.

This manual attempts to give the advanced user more details than are provided in the RT-11 SYSTEM REFERENCE MANUAL (DEC-11-ORUGA-A-D).

1.1 System Concepts and Terminology.

The basic concepts necessary to use RT-11 effectively are defined in the RT-11 SYSTEM REFERENCE MANUAL. The user should be familiar with those concepts before proceeding to use this manual.

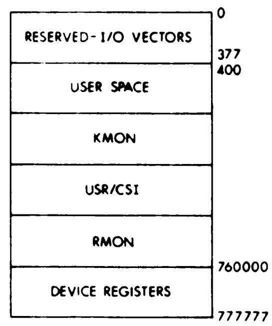
Abbreviations used in this document are:

Term	Meaning
KMON	Keyboard Monitor The console terminal interface to RT-11. KMON allows the user to run programs, assign device names, and generally control the system.
USR	User Service Routines The non-resident (swapping) part of RT-11. The USR performs file oriented operations in general.
RMON	Resident Monitor RT-11 has a 1,300(decimal) word resident monitor. This area contains the following services: EMT Dispatcher Keyboard (console) interrupt service Read/Write processor USR swap routines I/O queueing routines System device handler System I/O tables
CSI	Command String Interpreter The CSI is part of the USR. It accepts a string of characters from core or the console and performs specified file operations or constructs a table from the information supplied.
$CSW	Channel status word table. See Section 7.0 for a description of $CSW.
JSW	Job Status Word. Bytes 44 and 45. JSW contains information about the job now in core. For more information, refer to Section 2.2.1.

2.0 CORE LAYOUT

RT-11 operates properly in any configuration of between 8 and 28K (words) of core memory. No user intervention is required when programs are moved to a different size machine, i.e., programs developed in one environment can work in any size environment with no relinking necessary.

Following is a general diagram of the core layout in an RT-11 system.



The core area diagrammed is arranged as follows:

Core Area	Use
0-377	Reserved for I/O trap vectors, RT-11 system communication area, system core control block.
400-RMON	Space available for user programs.

The areas marked KMON and USR/CSI are the areas that these units would normally occupy when they are in core. The amount of core that a user program requires is determined by:

1. the initial size of the program

2. the amount of core the user program requests via a .SETTOP monitor request.

When the program is executed (via the KMON commands R, RUN or GET and START) the top of core is set to correspond with the size of the program. If the top of user core never goes past KMON, both KMON and USR/CSI are resident. If all of core (excluding RMON) is requested, no modules are resident, and swapping of USR is required. Programs doing a lot of file oriented operations would gain from having the USR resident, as no extraneous time doing swapping is required. (Swapping is the process of saving user core in system device scratch blocks, reading in and executing a USR function, and then restoring the user code.)

The KMON, USR, and RMON modules always occupy the upper 22000 bytes of core. This implies that larger core configurations automatically have more free core available.

The area marked 'Device registers' is the top 4K of core in any PDP-11. It is reserved for holding the status and control registers for peripheral devices.

2.1 'Floating' USR

Normally, the RT-11 USR operates in an area of core directly below RMON. However, in some programs it may be desirable to be able to have the USR at some alternate address. That can be accomplished by storing an address in the word at location 46. If this word is non-zero (and even!) the USR is loaded at the specified address, rather than at its normal address. Note that this happens only if USR swapping is required. If the USR is resident, it is not reloaded at the new address. On doing a ↑C or .EXIT back to the keyboard Monitor, the user program is restored, and RT-11 clears word 46 and returns to its normal core configuration.

2.2 Important Core Areas

Certain areas of core between 0 and 400 are reserved for use by RT-11. The system does not allow a program to destroy these locations when initiated by KMON, e.g., R, RUN, will not write over the areas. However, no protection is supplied via memory protect. Thus, programs should never alter the content of the indicated areas at run time.

Locations	Contents
0,2	Monitor restart. Executes .EXIT request and returns control to KMON.
4	Time out. Bus error trap - RT-11 sets this to point to location 6.
6	HALT instruction.
10	Reserved instruction trap. RT-11 sets it to point to location 12.
12	HALT instruction.

NOTE

The HALT instructions are set purposely. If a HALT occurs at either location 6 or location 12, the top of the stack will point to the address immediately following the instruction which was in error. For example, assume the following program segment was being executed:

```
Location      Instruction

1000          MOV  (R1)+,R2
1002          ADD  #6,R2
                   .
                   .
                   .
```

If R1 has an odd value when this code is executed, a HALT at location 6 occurs, because auto-increment mode with an odd value in a register is illegal. The top of the stack would contain 1002, thus indicating that the error occurred in the preceding instruction.

Locations	Contents
20,22	IOT trap vector. RT-11 uses IOT for reporting fatal monitor errors. IOT should not be used by user programs.
30,32	EMT trap vector and status.
40-57	RT-11 system communication area. Refer to Section 2.2.1 for details.
60,62	TTY input interrupt vector and status.
64,66	TTY output interrupt vector and status.
214,216	TC11 vector and status.
220,222	RK05 vector and status.

These areas are not replaced by RT-11. If they are destroyed by a program, the system must be re-bootstrapped, or the program must restore them. In addition, RT-11 uses 360-377 each time a program is run. Thus the user should not depend on those bytes containing useful information across an R, RUN or GET and START sequence.

2.2.1 Description of Bytes 40-57: RT-11 System Communication Area - The area in core from 40-57 holds certain words and bytes of information used by the system. These words are:

Byte Address	Contents
40,41	Program Start Address When a GET, R, RUN, or START command is given, the starting address of the program is contained in these bytes. The REENTER command uses the start address-2.
42,43	Initial value of stack pointer, SP (register 6). This is initialized by the LINKer.
44,45	Job Status Word. (JSW) JSW is used as a series of bits indicating various conditions relevant to monitor operation. An asterisk (*) beside a particular bit indicates that the bit must be set by the user program. The other bits are set or cleared by the monitor.

Bit #	Set to	Meaning
15	1	USR is resident. No swapping is needed.

Bit #	Set to	Meaning
14	-	Unused
13*	0	Program is not re-enterable.
	1	REENTER is legal.
12*		"Special" TT input mode. Monitor does not echo typed input. The only special characters acted on are CTRL/C and CTRL/O.
11.10		Unused
9		The job is an overlay job. This bit is set to 1 by the Linker.
8		Unused
7*	1	Halt if a hardware I/O error occurs.

The other bits are currently unused. Digital reserves the right to use the remaining bits for monitor purposes.

2.2.2 Restrictions - Because RT-11 does not utilize any memory protection, the following restrictions must be observed:

1. The "Important Core Areas", listed above should not, as a rule, be altered by user programs. RT-11 could malfunction if they are disturbed.

2. When swapping of the USR is required, a 2K piece of user code has to be saved on the system device to make room for the USR. Clearly, then, pieces of code which must always be resident should not be in the area the USR would be swapped into. (Small programs that do not require swapping do not have the problem.) Code which should not reside in the same locations as the USR are:

   a. I/O completion routines

   b. I/O buffers

   c. Arguments to USR requests.

   d. Device handlers.

3.0 RT-11 FILE STRUCTURE

The device directory begins with physical block 6 of any file structure device. The device directory is a series of directory segments which contain the names and lengths of the files contained on that device. Each directory segment is made up of two physical blocks. Thus, a single directory segment is 512 words in length. The directory area is variable in length, from 1 to 31(decimal) directory segments. PIP allows specification of the number of segments when the /Z switch is used. The default value is four directory segments.

Each directory segment contains a 5-word header block, leaving 507 (decimal) words for directory entries.

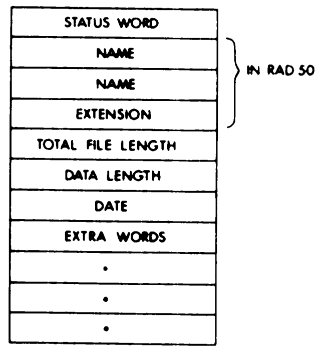
A directory segment has the following format

5 header words
file entries . . .

The header words contain the following information:

Word #	Contents
1	The number of segments available for entries. This number is specified in PIP when the device is zeroed, and must be in the range 1 N 31.
2	Pointer to the next logical directory segment number. The directory could, in certain cases, be a linked list. This word is the link word between logically contiguous segments. If this word equals 0, there are no more segments in the list. Refer to Section 3.1, Directory Segment Extensions, for more details on the link word.
3	The highest segment currently open. Each time a new segment is created, this number is incremented. This word is updated only in the first segment. It is unused in any but the first segment.
4	The number of extra bytes per directory entry. This number can be specified when the device is zeroed with PIP. Currently, RT-11 does not allow direct manipulation of information in the extra bytes.
5	Block number where files in this segment begin.

The remainder of the segment is filled with directory entries. An entry is in the following format:



Status Word

This word is broken down into two bytes of data.

Even byte: Used as channel identifier for ENTER and CLOSE operations. RT-11 maintains this byte. The identifier is the low order byte of the address of the $CSW entry for the channel. Thus, adding the address of $CSW to this byte points to the $CSW block for the channel.

Odd byte: Indicates the type of entry. Currently RT-11 recognizes the following file types:

Value	File Type
1	Tentative File. One which has been .ENTERed, but not .CLOSEd. Files of this type are deleted if not eventually .CLOSEd. PIP lists these as empty files.
2	An empty file. The name, extension, and date fields are not used. PIP lists an empty file as <UNUSED> followed by the length of the unused area.
4	A permanent entry. A tentative which has been .CLOSEd is a permanent file. The name of a permanent file is unique. There can be only one file with a given name and extension. If another exists before the .CLOSE is done, it is deleted by the monitor as part of the .CLOSE operation.

Value	File Type
10	End of segment marker. RT-11 uses this to determine when the end of the directory segment has been reached.

Name-Extension

These three words (in .RAD50) represent the symbolic name and extension assigned to a file.

Total File Length

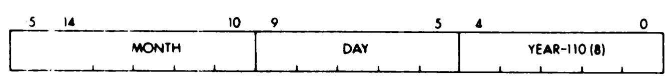
The number of blocks currently a part of the file. Attempts to read/write outside the limits of the file result in an End of File error.

Data Length

This word is not currently used in Release 1 of RT-11. Eventually, total file length and data length will be two distinct numbers. Total length will indicate the maximum number of blocks available for a .WRITE(W,C) operation. The data length will represent the greatest block actually written out. An attempt to read past the end of data block will cause an EOF. This will allow file extensions under RT-11. Currently, when RT-11 .CLOSEs a file, any space not actually written into is returned to an empty file which follows the tentative file being closed. Thus, in the first release, only total file length is considered.

Date

When a tentative file is created via .ENTER, the system date word is put into the creation date slot for the file. The date word is in the following format:

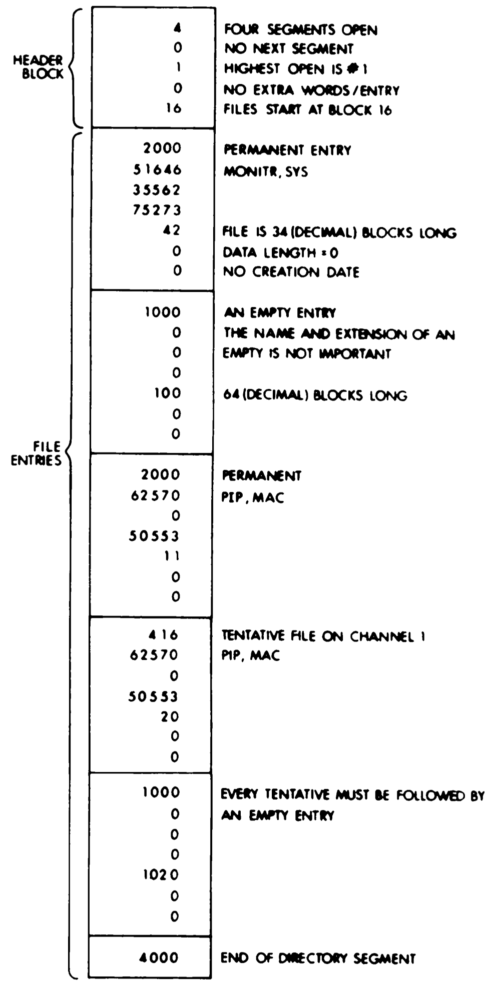


Extra Words

Described by the number of extra bytes per entry in the header words. RT-11 provides no direct facilities for manipulating this extra information. To manipulate extra words, the user program would have to perform direct operations on the RT-11 directory.

Example:

Following is a typical RT-11 directory segment:



When the tentative file PIP.MAC is .CLOSEd, the permanent file PIP.MAC is deleted.

To find the starting block of any particular file, add all file lengths before the desired file and the starting block number given in the fifth word of the segment header. For example, in the above directory, PIP.MAC will begin at block number 160 (octal).

3.1 Size and Number of RT-11 files

The maximum number of directory segments on any RT-11 device is 31(decimal). With no extra information words in a directory entry, this theoretically leaves room for a maximum of 31 x $\frac{512-5}{7+N}$ entries. If N equals 0, this indicates that the maximum is 2232(decimal) entries.

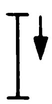
However, because of the way RT-11 performs directory expansions, the total number of entries allowed is 1/2 the theoretical maximum. Thus, with 31 segments available, there would be a maximum of 1116(decimal) directory entries. The equation to determine the maximum number of entries in any particular directory size is:

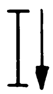
$$M \times \left[\frac{507}{2\ (7+N)}\right]$$

Here, M is the number of directory segments open, N is the number of extra words per directory entry.

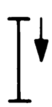
Directory Segment Extensions

RT-11 allows a maximum of 31(decimal) directory segments. This section details what occurs when a directory segment overflows. For illustrational purposes, the following symbols are used:

n  **To represent a directory segment with some number of directory entries. n is the segment number.**

n  **To represent a segment which is full; i.e. no more entries will fit in the segment.**

Systems start out with entries being put into segment #1:

1 

As entries are added, this segment fills:

1 

When this occurs, and an attempt is made to add another entry to the device, the system must open up another directory segment. Word #1 of each directory segment contains the maximum number of segments

available. When the last segment becomes full, and an attempt is made to enter another file, a fatal error,

?M-DIR OVFLO?

is generated.

If another segment is available, the following happens:

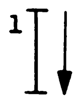
1. One half of the entries from the filled segment are put into the next available segment.

2. The shortened segment is re-written to the disk.

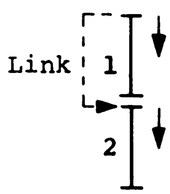
3. The directory segment links are set.

4. The file is entered in the newly created segment.

Thus, in the normal case, the segment appears as:

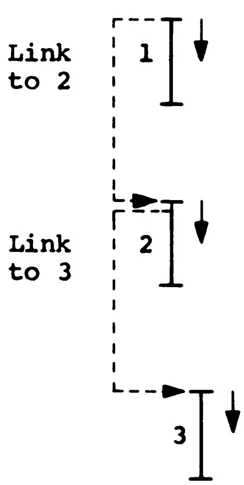


before extension

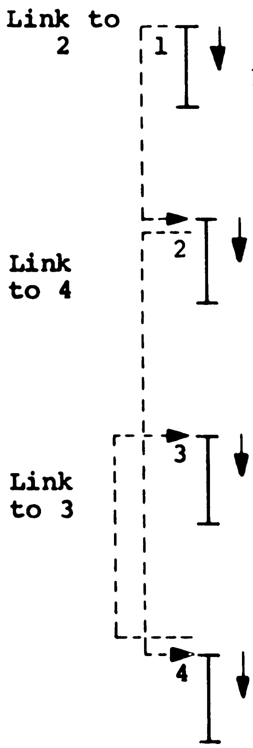


After extension. Half the entries are in the new segment, half in the old. Segment 1 is linked to segment 2.

If many more files are entered, they will fill up the second segment, and overflow into the third segment, if it is available. That would appear as:



In this case, the links between the segments are not strictly necessary, as the segments are contiguous. However, the links do become necessary if a large file is deleted from segment 2, and many small files are entered, it would be possible to overflow segment 2 again. If this occurred, and a fourth segment existed, the directory would appear:



In this case, segment 2 overflows into segment 4 and the links are used to link logical pieces rather than physical pieces.

4.0 DETAILED STRUCTURE OF THE SYSTEM DEVICE

The RT-11 system device holds all of the components of the system. The system device is used by RT-11 to store device handlers, and the system itself. The layout of the system device is:

Block #	Contents
0	Bootstrap
1	Reserved for RSX-11D compatibility
2	Bootstrap
3 to 5	Reserved for expansion
6 to (N*2)+5	Directory segments. N is the number of directory segments open
(N*2)+6 to end	File storage

All other system components, i.e., KMON, USR/CSI, RMON and device handlers, are files on the system device.

The files which are system components are:

File	Contains
MONITR.SYS	The current RT-11 monitor. Contains: bootstrap, swap blocks, KMON, USR/CSI, RMON.
LP.SYS	Line printer handler
DT.SYS	DECtape handler
TT.SYS	Console handler
RK.SYS	RK disk handler
PR.SYS	Highspeed reader handler
PP.SYS	Highspeed punch handler

Refer to Appendix A for listings of device handlers.

In general, files with the .SYS extension are parts of the monitor system. The bootstrap acts to record the block numbers of the relevant areas. Thus, RT-11 is extremely flexible with respect to the interchange and construction of systems.

4.1 Content of MONITR.SYS

This is the block layout of the RT-11 monitor file, MONITR.SYS:

Block # (Relative to start of MONITR.SYS)	Contents
0,1	Copy of system bootstrap. (Blocks 0 and 2 of the system device)
2-17	Swap blocks
20-24	KMON
25-34	USR, CSI
35-41	RMON

5.0 EMT CALLS

RT-11 MACRO supports all RT-11 monitor calls via standard system MACRO calls. However, in certain applications, it is necessary to call the monitor in a way which is not supported by SYSMAC.SML. This section describes the detailed breakdown of the RT-11 EMT structure. The listing of SYSMAC.SML will be a helpful reference when reading this section.

The EMT structure is:

15 8	7 4	3 0
EMT code: 104	Function code (16)	Channel # (subcode)

Bits 8 through 15 contain the op code 104, to designate the EMT instruction.

Bits 4 through 7 contain the code of the function to be performed. The codes currently assigned are:

Code	Function
0	Delete
1	Lookup
2	Enter
3	Unused
4	Rename
5	Savestatus
6	Reopen
7	Close
10	Read
11	Write
12	Wait
13	
14	Unused
15	
16	Miscellaneous. See below.
17	Reserved for internal RT-11 use. Should not be used by user programs.

Bits 0-3 contain the channel number (0-17).

Code 16 is used to handle those requests which do not require a channel indentifier. In that case, bits 0 through 3 give the subcode of the function to be performed.

Sub-Code (Bits 0-3)	Function
0	Get character from console (.TTYIN/.TTINR)
1	Output character to console (.TTYOUT/.TTOUTR)
2	Get device status (.DSTATUS)
3	Fetch device handler (.FETCH)
4	Call CSI - general mode(.CSIGEN)
5	Call CSI - special mode(.CSISPC)
6	Lock USR in core(.LOCK)
7	Unlock USR(.UNLOCK)
10	EXIT(.EXIT)
11	Print .ASCIZ string(.PRINT)
12	Software reset(.SRESET)
13	Expand I/O queue(.QSET)
14	Set top of core(.SETTOP)
15	Reset control O(.RCTRLO)
16	Unused
17	Hardware reset(.HRESET)

As an example of the use of the basic form of the EMT, the following loop will close files on any channel which may currently be in use:

```
          MOV #20, R1                ;16 possible files.
          MOV (PC)+, @(PC)+          ;put in a close channel 0 instruction.
          EMT 16*7+0                 ;
          LOOP
LOOP:     HALT                       ;this gets modified.
          INC LOOP                   ;bump the channel number.
          DEC R1
          BNE LOOP                   ;loop until done.
```

Note that this technique should not be used as a common practice. It should only be used where absolutely necessary, realizing that programs which use that technique may be incompatible with future versions of the monitor. It is strongly suggested that the macros defined in SYSMAC.SML be used.

The EMT can be written as:

```
EMT FC*20+CHAN
```

Where FC is the function code; CHAN is the desired channel number.

Error returns from this type of call are exactly the same as the returns from a MACRO call, i.e., the C bit is set in the processor status word, and byte 52 contains an error code.

Again, it is emphasized that this form of the monitor call should not be used in general, as the programs would malfunction if the EMT arguments were ever changed. The system macro calls, will, however, be correct.

6.0 I/O SYSTEM

I/O transfers in RT-11 are handled by the monitor, through routines known as device handlers. Device handlers are routines which are resident on the system mass storage device, and can be called into core at a location specified by the user (.FETCH handler request). Once a device handler is in core, the several available .READ/.WRITE requests are taken by the monitor, and translated into a call to the I/O device. To facilitate overlapped I/O and computation, all I/O requests to RT-11 are done through an I/O queue. This section details the structure of the I/O queueing system.

6.1 Queue Element

The RT-11 I/O queue is made up of a linked list of queue elements. A single element has the following structure:

Word #	Contents
1	Pointer to next element for this device. If 0, indicates no more entries for this device.
2	Pointer to CSW (Channel Status Word) area.

Word #	Contents
3	Physical device address where data is to be stored. (Block number).
4	Unit number of the device to be used.
5	Buffer address to commence transfer.
6	Word count. Greater than 0 indicates a read operation, less than 0 indicates a write operation. A value of 0 is generally taken to mean a block seek operation.
7	Completion Function. 0 = Wait for I/O transfer to complete. 1 = Return after queueing request. No action on completion. N = Return after queueing request. On completion, transfer to completion routine, at the address indicated by N.

Each I/O queue element is in this format. A monitor request, .QSET, is available to increase the number of slots available in the I/O queue. RT-11 maintains a one-element queue in the resident monitor. This is sufficient for any program which uses wait-mode I/O (.READW/.WRITW). However, for maximum throughput, .QSET should be used to create additional queue elements.

If an I/O operation is requested and no free queue elements exist, RT-11 must wait until an element is free to queue the request. This obviously slows up program execution. It will always be sufficient to allocate N new queue elements, where N is the total number of devices which will be used in a particular program. This produces a total of N+1 available elements, since the resident monitor element is added to the list of available elements.

6.2 RT-11 Device Handlers

The user should refer to the PDP-11 Peripherals and Interfacing Handbook for details regarding the operation of any particular peripheral.

6.2.1 Device Handler Format

6.2.1 Device Handler Format - A device handler is an RT-11 routine which is used to transfer data between physical devices and core memory. The handler is resident on the system device, and can be loaded dynamically by a user program. RT-11 currently supports the following devices:

TC11	DECtape
RK11	Disk
PC11	High-speed reader/punch
LP11	
LS11	Line printers
LV11	

Any Teletype-like device, i.e., ASR33, 35, LA30, VT05, etc.

Some users may wish to write and use their own device handlers for special peripherals. This can be done as long as the handler follows the general rules and format for all device handlers. This section details the exact rules and format to be followed.

As an example in this discussion, refer to Figure 1, which is a listing of the RT-11 PR handler. Appendix A contains listings of all the other RT-11 handlers.

The first five words of any device handler are header words.

The format is:

Word #	Contents
1	Address of first word of trap vector.
2	Offset from current PC to interrupt handler.
3	Processor status word to be used when interrupt occurs.
4,5	Zero.

6.2.2 Entry Conditions - The device handler is entered directly from the monitor I/O queue manager. Thus, the handler is entered with information about the transfer to be done. The fifth word of the header contains a pointer into the queue element to be processed. This word (called CQE, for Current Queue Element) points to the third word of the queue element. The third word of the queue element is the block number to be read or written. (Refer to Section 6.1 for the description of a queue element.) Referring to the example, location PRCQE contains the address of the third word in the queue element to be processed. It is generally advisable to put the pointer into a register, as that greatly facilitates picking up arguments to initiate the transfer.

6.2.3 Data Transfer - It is assumed that any device handler will work with the interrupt enabled. In this case, the handler will return control to the monitor before data transfer is complete. If the interrupt is not used, the transfer must be finished before returning to the monitor. In either case, to return control to the monitor after initiating the transfer, an RTS PC must be executed. Registers need not be preserved on entry to the handler.

6.2.4 Interrupt Handler - Once the transfer has been initiated, and control has passed back to the monitor, data interrupts will occur.

Information in the header of the handler causes the interrupt to be vectored to the location specified in the header. The code at the interrupt location should keep the transfer going, determine when the transfer is complete and detect errors.

When the transfer is done, control must be passed to the monitor's I/O queue manager which performs cleanup manipulation on the I/O queue.

```
                         .TITLE PR   V01-04  7/22/73

                         ;RT-11 HIGH SPEED PAPER TAPE READER (PC11) HANDLER
                         ;
                         ;DEC-11-ORTRA-A-LA
                         ;
                         ;ERIC PETERS/ROBERT BEAN
                         ;
                         ;APRIL 1973
                         ;
                         ;COPYRIGHT 1973
                         ;DIGITAL EQUIPMENT CORPORATION
                         ;MAYNARD,MASSACHUSETTS 01754

                         ;DEC ASSUMES NO RESPONSIBILITY FOR THE
                         ;USE OR RELIABILITY OF ITS SOFTWARE ON
                         ;EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.


        000000'          .CSECT PC11PR

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

                         ;PAPER TAPE READER CONTROL REGISTERS
        177550           PRS=177550             ;CONTROL REGISTER
        177552           PRB=177552             ;DATA REGISTER
        000070           PRVEC=70               ;READER VECTOR ADDR

        000001           PRGO=1                 ;READER ENABLE BIT
        000101           PINT=101               ;INTERRUPT ENABLE BIT AND GO BIT

                         ;TELETYPE REGISTERS
        177560           TKS=177560
        177562           TKB=177562
        177564           TPS=177564
        177566           TPB=177566

                         ;CONSTANTS FOR MONITOR COMMUNICATION
        000001           HDERR=1                ;HARD ERROR BIT
        000054           MONLOW=54              ;POINTER TO MONITOR BASE
        000270           OFFSET=270             ;POINTER TO Q MANAGER COMPLETION


                         .MCALL  .EXIT,.SRESET

        000203           CTRLC=203
```

Figure 1 Sample Handler

```
                        ;LOAD POINT

000000 000070           .WORD PRVEC         ;ADDR OF INTERRUPT VECTOR
000002 000120           .WORD PRINT-.       ;OFFSET TO INTERRUPT ENTRY
000004 000200           .WORD 200           ;PRIORITY 4
000006 000000 PRLQE:    .WORD 0             ;POINTER TO LAST Q ENTRY
000010 000000 PRCQE:    .WORD 0             ;POINTER TO CURRENT Q ENTRY

                        ;ENTRY POINT

 00012 016703 PR:       MOV PRCQE,R3        ;POINTER TO Q ENTRY IN R3
       177772
 00016 006363           ASL 6(R3)           ;WORD COUNT TO BYTE COUNT
       000006
 00022 103506           BCS PRERR           ;A WRITE REQUEST IS ILLEGAL TO P
 00024 001507           BEQ SEEK            ;A REQUEST FOR 0 BYTES IS A SEEK
 00026 005713           TST (R3)            ;IS THIS A REQUEST FOR BLOCK 0?
 00030 001030           BNE 5$              ;NO-JUST DO IT
                                            ;YES-WE NEED PROMPT CHARACTER
 00032 012702           MOV #TPS,R2         ;POINT R2 TO TPS
       177564
 00036 032712 4$:       BIT #100,(R2)       ;IS INTERRUPT BIT ON?
       000100
 00042 001375           BNE 4$              ;MONITOR WILL TURN IT OFF WHEN D
                                            ;THE ABOVE IS OUR WAY OF TESTING
                                            ;THE KEYBOARD IS IDLE. IT WORKS
                                            ;FOR RT-11 KEYBOARD I/O
 00044 112737           MOVB #'^,@#TPB      ;TYPE "^"
       000136
       177566
 00052 105712 3$:       TSTB (R2)           ;DONE?
 00054 100376           BPL 3$              ;NO-WAIT
 00056 012702           MOV #TKS,R2         ;POINT R2 TO KEYBOARD STATUS
       177560
 00062 011246           MOV (R2),-(SP)      ;SAVE INTERRUPT ENABLE BIT
 00064 012712           MOV #1,(R2)         ;REQUEST A CHAR
       000001
 00070 105712 1$:       TSTB (R2)           ;HAS HE TYPED ONE YET?
 00072 100376           BPL 1$              ;NO-TAPE IS NOT READY
 00074 122737           CMPB #CTRLC,@#TKB   ;IS IT ^C?
       000203
       177562
 00102 001002           BNE 2$              ;NO
 00104                  .SRESET             ;WE HAVE TO RESET TO DELETE THIS
 00106                  .EXIT               ;RETURN TO THE MONITOR
 00110 012612 2$:       MOV (SP)+,(R2)      ;RESTORE INTERRUPT BIT
 00112 012737 5$:       MOV #PINT,@#PRS     ;ENABLE THE READER INTERRUPT AND
       000101
       177550
 00120 000207           RTS PC

                        ;INTERRUPT SERVICE
 00122 010346 PRINT:    MOV R3,-(SP)        ;SAVE R3
 00124 016703           MOV PRCQE,R3        ;R3 POINTS TO Q ENTRY
       177660
 00130 062703           ADD #4,R3           ;POINT R3 TO BUFFER ADDRESS
       000004
```

Figure 1 Sample Handler (Cont.)

```
44 00134 005737          TST @#PRS                ;ANY ERRORS?
         177550
45 00140 100414          BMI PREOF                ;YES-TREAT AS EOF
46 00142 113773          MOVB @#PRB,@(R3)         ;PUT CHAR IN BUFFER
         177552
         000000
47 00150 005223          INC (R3)+                ;BUMP BUFFER POINTER
48 00152 005313          DEC (R3)                 ;DECREASE BYTE COUNT
49 00154 001416          BEQ PRDONE               ;IF ZERO,WE ARE DONE
50 00156 052737          BIS #PRGO,@#PRS          ;RE-ENABLE READER
         000001
         177550
51 00164 012603          MOV (SP)+,R3             ;RESTORE R3
52 00166 000002          RTI
53
54 00170 005213 PREO1:   INC (R3)
55 00172 105073 PREOF:   CLRB @(R3)          \    ;CLEAR REMAINDER OF BUFFER
         000000                              |
56 00176 005363          DEC 2(R3)           |
         000002                              >B
57 00202 001372          BNE PREO1           |    ;MORE
58 00204 052773          BIS #20000,@-6(R3)  |    ;SET EOF BIT IN CHANNEL STATUS
         020000                              /
         177772
```



```
1                ;OPERATION COMPLETE
2
3 000212 042737 PRDONE: BIC #PINT,@#PRS       ;TURN OFF THE READER INTERRUPT
         000101
         177550
4                                             ;IN CASE WE GET AN ERROR LATER
5 000220 010703         MOV PC,R3
6 000222 062703         ADD #PRCQE-.,R3       ;GET ADDR OF CURRENT Q ENTRY
         177566
7 000226 010046         MOV R0,-(SP)
8 000230 013700         MOV @#MONLOW,R0    \
         000054                            >A
9 000234 016007         MOV OFFSET(R0),PC  /  ;JUMP TO COMPLETION
         000270
10
11 00240 052753 PRERR:  BIS #HDERR,@-(R3)     ;SET HARD ERROR BIT
         000001
12 00244 005046 SEEK:   CLR -(SP)
13 00246 004767         JSR PC,1$             ;FAKE AN INTERRUPT
         000002
14 00252 000207         RTS PC
15 00254 010346 1$:     MOV R3,-(SP)          ;Q MGR EXPECTS R3 ON STACK
16 00256 000755         BR PRDONE             ;AND COMPLETE OPERATION
17
18       000001'        .END
```

Figure 1 Sample Handler (Cont.)

```
PR    V01-04  7/22/73     RT-11 MACRO VM01-01    23-OCT-73 PAGE 3+
SYMBOL TABLE

CTRLC = 000203             HDERR = 000001              MONLOW= 000054
OFFSET= 000270             PC    =%000007              PINT  = 000101
PR      000012R     002    PRB   = 177552              PRCQE   000010R     002
PRDONE  000212R     002    PREOF   000172R     002     PREO1   000170R     002
PRERR   000240R     002    PRGO  = 000001              PRINT   000122R     002
PRLQE   000006R     002    PRS   = 177550              PRVEC = 000070
R0    =%000000             R1    =%000001              R2    =%000002
R3    =%000003             R4    =%000004              R5    =%000005
SEEK    000244R     002    SP    =%000006              TKB   = 177562
TKS   = 177560             TPB   = 177566              TPS   = 177564

. ABS.  000000      000
        000000      001
PC11PR  000260      002
ERRORS DETECTED: 0
FREE CORE: 6905. WORDS
```

Figure 1 Sample Handler (Cont.)

The restrictions which apply to interrupt code are:

1. Any registers used in the code must be preserved on the stack.

2. A check must be made to determine if the transfer is complete. However, with non-file oriented devices, such as paper tape, line printer, etc., an interrupt occurs whenever a character has been processed. For these devices, the byte count, which is in the queue element, is used as a character count.

   Non-file structured input devices should be able to detect an end of file condition, and pass that on to the monitor.

   NOTE

   The queue element contains a word count, not a byte count. The initial entry to the handler should change the word count to a byte count if the device interrupts at each character. The transfer is complete when the byte count decrements to 0.

3. Check for occurrence of an error. If a hardware error occurred, the hard error bit in the channel status word should be set, and the transfer should be aborted. The address of the channel status word is in word 2 of the queue element. The error bit is bit 0 of the CSW. Generally, it is advisable to retry a certain number of times if an error occurs. RT-11 currently retries up to eight times before deciding an error has occurred. (Note that this is for file structure devices only.) It is desirable, in case an error occurs, to do a drive or control reset, where appropriate, to clear the error condition before a retry is initiated.

4. If the transfer is not complete, and no error has occurred, registers used should be restored, and an RTI executed.

   To pass an EOF (end of file) to the monitor, the 20000 bit in the CSW should be set. Refer to the code at B in Figure 1 for an example of setting the EOF bit.

5. If the transfer is complete, and no error occurred, the monitor I/O completion routine must be entered to terminate activity and/or enter a completion routine, if necessary. When return is made to the monitor, R3 and R0 must be pushed on the stack, in that order. The monitor expects that, and restores those registers for the user. Refer to the code at A in Figure 1 for the method of returning to the monitor completion routine.

6. Handlers should check for special error conditions which can happen on the initial entry to the handler. For example, trying to write on a read-only device should produce a hard error. The code at C in Figure 1 does this. It must be emphasized that the user handlers should interface to the system in substantially the same way as the handlers included here. These handlers are included as a guide and an example.

6.2.5 Inserting Device Handlers in the System - Once the handler has been written, and independently debugged independently, the monitor must be modified to recognize the new device . This is done by adding entries to the RT-11 tables (refer to Section 7.0).

NOTE

The addresses listed here are for the V01-15 monitor only. In future releases, the numbers may change.

Table to be changed	Contents	Addresses available (Addressed with PATCH)
$HSIZE	Size of handler (in bytes)	12410-12422 inclusive
$DVSIZ	Size of device, in 256-word blocks. If nonfile device, entry = 0	12444-12456 inclusive
$PNAME	Permanent name of the device. Should be 2 alphanumeric characters entered in .RAD50 notation.	12534-12546 inclusive
$STAT	Device status table. Refer to the paragraph on the format of $STAT table.	15402-15414 inclusive

There is a restriction on the permanent names that can be put into the system. Currently, the arrangement of $PNAME is:

TT
SY
RK
PR
PP
LP
DT
DK

The implication is that they are in reverse alphabetical order. Thus, any names added to this table must follow that order to ensure proper operation. Thus, the only names used should be from the set:

DJ,DI,DH,...AC,AB,AA

Also, note that the name must be entered in .RAD50. Since PATCH does not have a .RAD50 interpretation switch, the name must be entered to PATCH in its numerical form. Refer to Appendix E of the RT-11 System Reference Manual for detailed .RAD50 conversion tables.

As an example, assume a handler for a new type of disk is to be inserted in the system. First, the values of the table entries for this device are determined:

$HSIZE: 300	After assembly, the handler was found to take up 300 bytes.
$DUSIZ: 2000	The disk has 1024 (decimal) 256-word blocks for storage.
$PNAME: .RAD50 /DA/ or 14450	The name assigned will be DA. The .RAD50 value of DA is 14450.
$STAT: 100011	The device is file structured, is a read/write device, and uses the standard RT-11 file structure. The identifier selected is 11. The index is selected by the user. Refer to Section 7.0 for the format of the $STAT table.

Once these values have been decided, the steps for inserting the device handler are:

1. Assemble the handler, using either MACRO or ASEMBL.

2. Link the handler at 1000. The name of the handler should be whatever the $PNAME entry is, with the .SYS extension appended:

```
.R LINK
*DA.SYS=DA       where DA.OBJ is the handler
*                object module.  The default link
                 address=1000.
```

3. Run PATCH to modify the tables:

```
.R PATCH

PATCH V01-01

FILE NAME--
*MONITR.SYS/M
*16000;0R                              (/M is necessary!)
*0,12410/       0         300          $HSIZE
*0,12444/       0         2000         $DUSIZ
*0,12534/       0         14450        $PNAME
*0,15402/       0         100011       $STAT
*E                                     Exit to Monitor
.
```

In this case we were able to use the first available entries in each table. Future device additions will use the succeeding entries in each table.

At this point, the system should be re-bootstrapped, to make the modified monitor resident. The device DA will then be available for use.

7.0 RT-11 MONITOR TABLES

RT-11 maintains several tables for use in performing certain system functions. These tables are set up to make I/O transfers and file operations as simple as possible. The tables allow RT-11 to translate the user specified transfer into an internal call to the monitor I/O service. The tables used by RT-11, and their contents are:

$CSW	Channel status word area. The "master link" between user I/O calls and the actual I/O transfer.
$UNAM1 $UNAM2	Used to hold user assigned names from the ASSIGN keyboard monitor command.
$STAT	Status words for the RT-11 I/O devices.
$ENTRY	Pointer into the physical device handler to be used for the transfer.
$HSIZE	Table of the sizes (in bytes) of the physical device handlers.
$DVSIZ	Table of the size of various file structured devices. The entries are in number of 256-word blocks on the device.
$DVREC	Table indicating where each device handler is on the system device. This table is updated during each system bootstrap.
$PNAME	A list of the permanent names of devices in RT-11. The names are in RAD50 format. The names are: TT SY RK PR PP LP DT DK The names are assembled in reverse alphabetical order. This order is vital to system operation, and must not be altered.

The format of $CSW is as follows:

Word #		Contents
1		Channel status word. The breakdown of this word is:
	Bit 0	1 = Hardware error occurred on this channel.
	1-5	Index into the physical device table.
	6	1 = A .RENAME operation is in progress.

Word #		Contents
	7	1 = .CLOSE requires a directory rewrite.(.ENTER)
		0 = No rewrite is required (.LOOKUP).
	8-12	Contains the directory segment number where the open-file can be found for .CLOSE.
	13	1 = An EOF (end of file) was found on this channel.
	14	Unused.
	15	1 = A file is open on this channel.
2		Starting block number of file referenced by this channel. If the file was opened on a non-file structured device, this word = 0.
3		Length of file; if .ENTER, was done it contains the length of the hole (empty slot) assigned for use by this file. If a .LOOKUP was done, it contains the length of the file.
4		Actual data length. This word is currently unused. It is reserved for future use.
5		Even byte: Channel I/O count. When any I/O request to this channel is made, this byte is incremented. The .WAIT request waits for this byte to return to 0.
		Odd byte: Unit number of the device to be used. A number between 0 and 7.

There are 16(decimal) groups of these words, each group representing one of the 16 possible channels RT-11 can use. The first of the CSW areas can be found at RMON + 4, and extends for 240 bytes.

Example of CSW area:

1. A file is open (.ENTER) on channel 0 on device DT3.

```
100614
   200
    50
     0
  1400
```

The starting block of the hole allotted is 200. The size of the hole allotted is 50. Bit 7 equals 1 because a .CLOSE will require the directory to be rewritten. Bits 8-12 contain a 1, indicating that the file was opened in the first directory segment. Bits 1-5 contain a six, which indexes into the device tables, and in fact points to device DT. The unit number bits equal 3, indicating the file is open on device DT3.

2. A .LOOKUP was done on device DK:

```
        101016
            44
            10
             0
             0
```

```
Bits 1-5 = 7 which indicates DK
       7 = 0 No directory rewrite is required on .CLOSE
    8-12 = 2 File is on second directory segment
      15 = 1 Channel is active.
```

Format of $STAT Table

The $STAT table contains information on the status of the device it represents. The breakdown of the $STAT entry is:

Even byte: A device type identifier. Each physical device is assigned an identifying value. This value appears in the even byte of $STAT. The values currently assigned are:

- 0 = RK05 disk
- 1 = DECtape
- 2 = Cassette (not currently available)
- 3 = Line Printer
- 4 = TTY (ASR33, 35, LA30, VT05)
- 7 = High speed reader
- 10 = High speed punch.

Odd byte:

Bit 15:	1 - File structured device (disk, DECtape) 0 - Non-file structured (TTY, high speed reader, etc.)
Bit 14:	1 - Read only device (PR)
Bit 13:	1 - Write only device (LP, PP)
Bit 12:	1 - Special file structured device. This is to handle cases of a file structured device which does not use the RT-11 file structure. Not currently implemented.

APPENDIX A

HANDLERS


```
                        .TITLE  RK05  V01-01

                        ;RT-11 DISK (RK11) HANDLER
                        ;
                        ;DEC-11-ORTKA-A-LA
                        ;
                        ;ED FRIEDMAN
                        ;
                        ;MAY,1973
                        ;
                        ;COPYRIGHT 1973
                        ;DIGITAL EQUIPMENT CORPORATION
                        ;MAYNARD,MASSACHUSETTS 01754

                        ;DEC ASSUMES NO RESPONSIBILITY FOR THE
                        ;USE OR RELIABILITY OF ITS SOFTWARE ON
                        ;EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.


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

                ;GLOBAL DEFINITIONS
                        .GLOBL  SYSTAT,SYSIZE,SBLOCK
                        .GLOBL  SY,SYINT,SYINPR
                        .GLOBL  RKSYS,DTSYS

        000000 DTSYS=   0                       ;THIS IS RK HANDLER
        100000 SYSTAT=  100000                  ;RK05 STATUS WORD
        011262 SBLOCK=  11262                   ;SIZE IN BLOCKS
               ;RK CONTROL DEFINITIONS:
        177400          RKDS=   177400
        177402          RKER=   177402
        177404          RKCS=   177404
        177406          RKWC=   177406
        177410          RKBA=   177410
        177412          RKDA=   177412

        000010 RKCNT=10                         ;# ERROR RETRYS

 00000  000220 START:   .WORD   220             ;RK TRAP ADDRESS
 00002  000136          .WORD   RKINT-.         ;OFFSET FROM INTERRUPT ADDRESS
 00004  000000          .WORD   0               ;INTERRUPTS SERVICE AT LEVEL 0
 00006         RKSYS:
 00006         SY:
 00006  000000 RKLQE:   .WORD   0               ;LAST Q ENTRY ADDRESS
 00010  000000 RKCQE:   .WORD   0               ;CURRENT Q ENTRY ADDRESS
 00012  012727          MOV     #RKCNT,(PC)+    ;SET ERROR RETRIES
        000010
 00016  000000 RKTRY:   0
```

```
00020 016701 RKAGN: MOV     RKCQE,R1        ;GET Q PARAMETER POINTER
      177764
00024 012702        MOV     #14,R2          ;DE BLOCK TO RK DISK ADDRESS
      000014
00030 012103        MOV     (R1)+,R3        ;RAW BLOCK NUMBER
00032 000402        BR      2$
00034 062702 1$:    ADD     #20,R2
      000020
00040 162703 2$:    SUB     #14,R3
      000014
00044 100373        BPL     1$
00046 060203        ADD     R2,R3           ;R3 HAS DISK ADDRESS
00050 012102        MOV     (R1)+,R2        ;GET UNIT #
00052 000302        SWAB    R2              ;RK HAS IT IN BITS 15-13
      000004        .REPT   4
                    ROR     R2
                    .ENDR
00064 050203        BIS     R2,R3           ;PUT IN UNIT #
```

```

000066 012702           MOV     #RKDA,R2        ;NOW LOAD DSK
       177412
000072 010312           MOV     R3,(R2)         ;DISK ADD, AND UNIT SELECT
000074 012142           MOV     (R1)+,-(R2)     ;BUFFER ADDRESS
000076 012703           MOV     #103,R3         ;FUNCTION, GUESS AT BEING A WRIT
       000103
000102 011142           MOV     (R1),-(R2)      ;WORD COUNT
000104 001412           BEQ     4$              ;0 WORDS IS A SEEK
000106 100403           BMI     3$
000110 005412           NEG     (R2)            ;READ, MAKE WORD COUNT NEG.
 00112 062703           ADD     #2,R3           ;ALTER FUNCTION
       000002
 00116 032737 3$:       BIT     #100,@#RKDS     ;TEST ACCESS READY
       000100
       177400
 00124 001774           BEQ     3$
 00126 010342           MOV     R3,-(R2)        ;START TRANSFER
 00130 000207           RTS     PC
 00132 062703 4$:       ADD     #6,R3           ;MAKE IT A SEEK
       000006
 00136 000767           BR      3$

 00140        SYINT:
       000240 SYINPR=240
 00140 032737 RKINT:    BIT     #120000,@#RKCS  ;TEST FOR ERR AND SEEK CMPLT
       120000
       177404
 00146 100413           BMI     RKERR           ;ERROR
 00150 001035           BNE     RKOUT           ;SEEK COMPLETE, IGNORE INTERRUPT
 00152 005037 RKH:      CLR     @#-2            ;BACK TO 0
       177776
 00156 004317           JSR     R3,(PC)         ;R3 ON STACK AND GET CURRENT PC
 00160 062703           ADD     #RKCQE-.,R3     ;CURRENT Q TO R3
       177630
 00164 010046           MOV     R0,-(SP)        ;SAVE R0
 00166 013700           MOV     @#54,R0         ;NOW GO INTO RESIDENT Q COMPLETE
       000054
       000270           OFFSET=270              ;WORD IS 256 BYTES PAST $RMON
 00172 016007           MOV     OFFSET(R0),PC   ;INTO RESIDENT.
       000270

 00176 010146 RKERR:    MOV     R1,-(SP)        ;SAVE R1
 00200 012737           MOV     #15,@#RKCS      ;DO DRIVE RESET
       000015
       177404
 00206 105737 1$:       TSTB    @#RKCS
       177404
 00212 100375           BPL     1$              ;WAIT FOR READY
 00214 005037           CLR     @#-2
       177776
 00220 005367           DEC     RKTRY           ;DONE TRYING?
       177572
 00224 100410           BMI     RKOUT2
 00226 010246           MOV     R2,-(SP)        ;NO, SAVE REGISTERS AND GO AGAIN
 00230 010346           MOV     R3,-(SP)
 00232 004767           JSR     7,RKAGN
```

```
      177562
00236 012603            MOV     (SP)+,R3
00240 012602            MOV     (SP)+,R2
00242 012601            MOV     (SP)+,R1
00244 000002 RKOUT:     RTI
00246 016701 RKOUT2:    MOV     RKCQE,R1
      177536
00252 052751            BIS     #1,@-(R1)       ;TURN ON HARD ERROR
      000001
00256 012601            MOV     (SP)+,R1
00260 000734            BR      RKH             ;START NEXT TRANSFER
      000262 SYSIZE=.-START
      000001'           .END
```

```
RK05  V01-01     RT-11 MACRO VM01-01    23-OCT-73 PAGE 2+
SYMBOL TABLE

DTSYS = 000000 G          OFFSET= 000270            PC     =%000007
RKAGN   000020R           RKBA   = 177410           RKCNT  = 000010
RKCQE   000010R           RKCS   = 177404           RKDA   = 177412
RKDS  = 177400            RKER   = 177402           RKERR    000176R
RKH     000152R           RKINT    000140R          RKLQE    000006R
RKOUT   000244R           RKOUT2   000246R          RKSYS    000006RG
RKTRY   000016R           RKWC   = 177406           R0     =%000000
R1     =%000001           R2     =%000002           R3     =%000003
R4     =%000004           R5     =%000005           SBLOCK= 011262 G
SP     =%000006           START    000000R          SY       000006RG
SYINPR= 000240 G          SYINT    000140RG         SYSIZE= 000262 G
SYSTAT= 100000 G
. ABS.  000000      000
        000262      001
ERRORS DETECTED: 0
FREE CORE: 6941. WORDS
```

```
                        .TITLE PP  V01  5/29/73

                        ;RT-11 HIGH SPEED PAPER TAPE (PC11) PUNCH HANDLER
                        ;
                        ;DEC-11-ORTPA-A-LA
                        ;
                        ;ROBERT BEAN
                        ;
                        ;APRIL 1973
                        ;
                        ;COPYRIGHT 1973
                        ;DIGITAL EQUIPMENT CORPORATION
                        ;MAYNARD,MASSACHUSETTS 01754

                        ;DEC ASSUMES NO RESPONSIBILITY FOR THE
                        ;USE OR RELIABILITY OF ITS SOFTWARE ON
                        ;EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.


        000000'         .CSECT PC11PP

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

                        ;PAPER TAPE PUNCH CONTROL REGISTERS
        177554          PPS=177554              ;PUNCH CONTROL REGISTER
        177556          PPB=177556              ;PUNCH DATA BUFFER
        000074          PPVEC=74                ;PUNCH VECTOR ADDR

                        ;CONSTANTS FOR MONITOR COMMUNICATION
        000001          HDERR=1                 ;HARD ERROR BIT
        000054          MONLOW=54               ;MONITOR BASE ADDR
        000270          OFFSET=270              ;POINTER TO Q MANAGER COMPL ENTR
```

```
                        ;LOAD POINT
  000000 000074                 .WORD PPVEC           ;ADDR OF INTERRUPT VECTOR
  000002 000032                 .WORD PPINT-.         ;OFFSET TO INTERRUPT SERVICE
  000004 000200                 .WORD 200             ;PRIORITY 4
  000006 000000 PPLQE:          .WORD 0               ;POINTER TO LAST Q ENTRY
  000010 000000 PPCQE:          .WORD 0               ;POINTER TO CURRENT Q ENTRY

                        ;ENTRY POINT

  000012 016703 PP:             MOV PPCQE,R3          ;R3 POINTS TO CURRENT Q ENTRY
         177772
  000016 006363                 ASL 6(R3)             ;WORD COUNT TO BYTE COUNT
         000006
  000022 103024                 BCC PPREAD            ;A READ REQUEST IS ILLEGAL
  000024 052737                 BIS #100,@#PPS        ;CAUSES INTERRUPT,STARTING TRANS
         000100
         177554
  000032 000207                 RTS PC

                        ;INTERRUPT SERVICE

  000034 010346 PPINT:          MOV R3,-(SP)          ;SAVE R3
  000036 005737                 TST @#PPS             ;ERROR?
         177554
  000042 100421                 BMI PPERR             ;YES-PUNCH OUT OF PAPER
  000044 016703                 MOV PPCQE,R3          ;NO-R3 POINTS TO CURRENT Q ENTRY
         177740
  000050 062703                 ADD #6,R3             ;POINT R3 TO BYTE COUNT
         000006
  000054 005713                 TST (R3)              ;ANY MORE CHARS TO OUTPUT?
  000056 001415                 BEQ PPDONE            ;NO-TRANSFER DONE
  000060 005213                 INC (R3)              ;DECREMENT BYTE COUNT (IT IS NEG
  000062 115337                 MOVB @-(R3),@#PPB     ;PUNCH CHARACTER
         177556
  000066 005213                 INC (R3)              ;BUMP POINTER
  000070 012603                 MOV (SP)+,R3          ;RESTORE R3
  000072 000002                 RTI

  000074 005046 PPREAD:         CLR -(SP)
  000076 004767                 JSR PC,1$             ;FAKE AN INTERRUPT
         000002
  000102 000207                 RTS PC                ;RETURN TO Q MGR
  000104 010346 1$:             MOV R3,-(SP)

  000106 052753 PPERR:          BIS #HDERR,@-(R3)     ;SET HARD ERROR BIT
         000001

  000112 005037 PPDONE:         CLR @#PPS             ;CLEAR INTERRUPT ENABLE
         177554
  000116 010703                 MOV PC,R3
  000120 062703                 ADD #PPCQE-.,R3       ;ADDR OF NEXT Q ENTRY POINTER
         177670
  000124 010046                 MOV R0,-(SP)
  000126 013700                 MOV @#MONLOW,R0
         000054
  000132 016007                 MOV OFFSET(R0),PC     ;JUMP TO Q MANAGER
         000270
```

```
44
45        000001'          .END
```

```
PP  V01  5/29/73         RT-11 MACRO VM01-01   23-OCT-73 PAGE 2+
SYMBOL TABLE

MDERR = 000001          MONLOW= 000054              OFFSET= 000270
PC    =%000007          PP       000012R     002    PPB    = 177556
PPCQE   000010R   002   PPDONE   000112R     002    PPERR    000106R    002
PPINT   000034R   002   PPLQE    000006R     002    PPREAD   000074R    002
PPS   = 177554          PPVEC  = 000074             R0     =%000000
R1    =%000001          R2     =%000002             R3     =%000003
R4    =%000004          R5     =%000005             SP     =%000006

. ABS.  000000    000
        000000    001
PC11PP  000136    002
ERRORS DETECTED: 0
FREE CORE: 7020. WORDS
```

```
                            .TITLE TT  V01-01  18-JUN-73

                            ;RT-11 GENERAL TERMINAL HANDLER
                            ;
                            ;DEC-11-ORTTA-A-LA
                            ;
                            ;ERIC PETERS
                            ;
                            ;JUNE, 1973
                            ;
                            ;COPYRIGHT 1973
                            ;DIGITAL EQUIPMENT CORPORATION
                            ;MAYNARD MASSACHUSETTS 01754

                    ; DEC ASSUMES NO RESPONSIBILITY FOR THE
                    ; USE OR RELIABILITY OF ITS SOFTWARE ON
                    ; EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.

        000000'             .CSECT TT

        000000              R0=%0
        000001              R1=%1
        000002              R2=%2
        000003              R3=%3
        000004              R4=%4
        000005              R5=%5
        000006              SP=%6
        000007              PC=%7
        177776              PS=177776

                            ;TERMINAL VECTORS
        000060              TKVEC=60
        000062              TKPR=62
        000064              TPVEC=64
        000066              TPPR=66
        177564              TPS=177564
        177560              TKS=177560

                            ;CONSTANTS FOR MONITOR COMMUNICATION
        000001              HDERR=1             ;HARD ERROR BIT IN CSW
        000054              MONLOW=54           ;POINTER TO MONITOR BASE
        000270              OFFSET=270          ;POINTER TO Q MGR COMPLETION ENTRY
        000032              CTRLZ=32

                            .MCALL .TTINR,.TTOUTR,.TTYIN,.TTYOUT

```

```
                               ;LOAD POINT

  000000 000000                .WORD 0                         ;0 BECAUSE WE WANT FETCH TO LEAV
  000002 000000                .WORD 0
  000004 000000                .WORD 0
  000006 000000 TTLQE:         .WORD 0                         ;POINTER TO LAST Q ENTRY
  000010 000000 TTCQE:         .WORD 0                         ;POINTER TO CURRENT Q ENTRY

                               ;ENTRY POINT

  00012 016703  TT:            MOV TTCQE,R3                    ;R3 POINTS TO Q ENTRY
        177772
  00016 006363                 ASL 6(R3)                       ;WD CNT => BYTE CNT
        000006
  00022 103014                 BCC        TTREAD
  00024 013767                 MOV        @#TPVEC,TPVS1        ;SAVE INTERRUPT VECTOR ADDRESS
        000064
        000140
  00032 010703                 MOV        PC,R3
  00034 062703                 ADD        #TPINT-.,R3
        000072
  00040 010337                 MOV        R3,@#TPVEC           ;SET NEW INTERRUPT ADDR
        000064
  00044 012737                 MOV        #100,@#TPS           ;AND ENABLE INTERRUPT
        000100
        177564
  00052 000207                 RTS        PC

  00054 013767  TTREAD:        MOV        @#TKVEC,TKVS1        ;SAVE OLD VECTOR CONTENTS
        000060
        000154
  00062 005713                 TST        @R3                  ;BLOCK ZERO?
  00064 001004                 BNE        1$
  00066                        .TTYOUT #'*                     ;TYPE PROMPT CHARACTER
  00076 005046  1$:            CLR        -(SP)                ;SIMULATE AN INTERRUPT TO
  00100 004767                 JSR        PC,KBENT2            ;EMPTY THE MONITOR RING BUFFER
        000134
  00104 020367                 CMP        R3,TTCQE             ;WAS REQUEST SATISFIED?
        177700
  00110 001005                 BNE        2$                   ;YES, SIMPLY RETURN
  00112 010703                 MOV        PC,R3
  00114 062703                 ADD        #KBINT-.,R3          ;CALC ADDR OF KBINT
        000116
  00120 010337                 MOV        R3,@#TKVEC           ;AND SET INTERRUPT INTERCEPT
        000060
  00124 000207  2$:            RTS        PC
```

```
                         ; TELEPRINTER INTERRUPT ENTRY
000126 010346 TPINT:     MOV     R3,-(SP)
000130 010046            MOV     R0,-(SP)
000132 016703            MOV     TTCQE,R3
       177652
000136 062703            ADD     #6,R3                ;POINT TO BYTE COUNT
       000006
000142 005713            TST     @R3                  ;DONE?
000144 001413 TPENT2:    BEQ     TPDONE
000146 115300            MOVB    @-(R3),R0            ;GET NEXT CHAR FROM BUFFER
000150 001402            BEQ     1$                   ;CHAR IS NULL
 00152                   .TTOUTR                      ;TYPE CHAR IF POSSIBLE
 00154 103403            BCS     TPFULL               ;NO ROOM, WAIT AWHILE
 00156 005223 1$:        INC     (R3)+                ;BUMP BUFFER POINTER
 00160 005213            INC     @R3                  ;AND BYTE COUNT
 00162 000770            BR      TPENT2
 00164 012600 TPFULL:    MOV     (SP)+,R0             ;RESTORE REGS
 00166 012603            MOV     (SP)+,R3
 00170 000137            JMP     @(PC)+               ;AND PASS INTERRUPT TO MONITOR
 00172 000000 TPVS1:     .WORD   0                    ;POINTER TO MONITOR INTERRUPT EN
 00174 012600 TPDONE:    MOV     (SP)+,R0
 00176 011603            MOV     @SP,R3
 00200 005016            CLR     @SP                  ;CREATE A PSEUDO INTERRUPT
 00202 016746            MOV     TPVS1,-(SP)          ;THAT RETURNS TO MONITOR INTERRU
       177764
 00206 011637            MOV     @SP,@#TPVEC          ;RESTORE INTERRUPT VECTOR
       000064
 00212 004317 RTQMGR:    JSR     R3,(PC)              ;AND CALL Q MANAGER COMPLETION R
 00214 062703            ADD     #TTCQE-.,R3
       177574
 00220 010046            MOV     R0,-(SP)
 00222 013700            MOV     @#MONLOW,R0
       000054
 00226 016007            MOV     OFFSET(R0),PC
       000270

 00232 005046 KBINT:     CLR     -(SP)                ;LET MONITOR SERVICE CHARACTER I
 00234 004737            JSR     PC,@(PC)+            ;AND RETURN VIA RTI
 00236 000000 TKVS1:     .WORD   0                    ;KEEP MON INTERRUPT ADDR HERE
 00240 010346 KBENT2:    MOV     R3,-(SP)
 00242 010046            MOV     R0,-(SP)
 00244 016703            MOV     TTCQE,R3             ;POINT TO CURRENT QUEUE ENTRY
       177540
 00250 062703            ADD     #6,R3   ;POINT TO BYTE COUNT
       000006
 00254 005713            TST     @R3                  ;DONE?
 00256 001424 1$:        BEQ     KBDONE               ;YES
 00260                   .TTINR                       ;GET NEXT CHAR
 00262 103430            BCS     TKRTI                ;RETURN IF NONE AVAILABLE
 00264 120027            CMPB    R0,#CTRLZ            ;IS CHAR CONTROL-Z?
       000032
 00270 001404            BEQ     KBEOF
 00272 110053            MOVB    R0,@-(R3)            ;STUFF CHAR INTO BUFFER
 00274 005223            INC     (R3)+                ;BUMP POINTER
 00276 005313            DEC     @R3                  ;AND CHAR COUNT
 00300 000766            BR      1$                   ;AND TRY FOR ANOTHER
 00302 105053 KBEOF:     CLRB    @-(R3)               ;END FILE - CLEAR REST OF BUFFER
```

```
00304 005223          INC     (R3)+
00306 005313          DEC     @R3
00310 001374          BNE     KBEOF
00312 052773          BIS     #20000,@-8.(R3) ;SET EOF BIT IN CSW
      020000
      177770
00320                 .TTINR                  ;EAT CR/LF AFTER CTRL-Z
00322 103402          BCS     KBDONE          ;NOTHING THERE - FORGET IT
00324                 .TTYIN
00330 012600 KBDONE:  MOV     (SP)+,R0
00332 016737          MOV     TKVS1,@#TKVEC   ;RESTORE KEYBOARD INTERRUPT
      177700
      000060
00340 012603          MOV     (SP)+,R3
00342 000723          BR      RTQMGR
00344 012600 TKRTI:   MOV     (SP)+,R0
00346 012603          MOV     (SP)+,R3
00350 000002          RTI

      000001'         .END
```

```
TT  V01-01  18-JUN-73   RT-11 MACRO VM01-01   23-OCT-73 PAGE 3+
SYMBOL TABLE

CTRLZ = 000032                   HDERR = 000001                    KBDONE   000330R    002
KBENT2   000240R    002          KBEOF    000302R    002           KBINT    000232R    002
MONLOW= 000054                   OFFSET= 000270                    PC      =%000007
PS     = 177776                  RTQMGR   000212R    002           R0      =%000000
R1     =%000001                  R2      =%000002                  R3      =%000003
R4     =%000004                  R5      =%000005                  SP      =%000006
TKPR   = 000062                  TKRTI    000344R    002           TKS     = 177560
TKVEC  = 000060                  TKVS1    000236R    002           TPDONE   000174R    002
TPENT2   000144R    002          TPFULL   000164R    002           TPINT    000126R    002
TPPR   = 000066                  TPS     = 177564                  TPVEC  = 000064
TPVS1    000172R    002          TT       000012R    002           TTCQF    000010R    002
TTLQE    000006R    002          TTREAD   000054R    002
. ABS.   000000     000
         000000     001
TT       000352     002
ERRORS DETECTED: 0
FREE CORE: 6796. WORDS
```

```
                        .TITLE LP   V01-03      5/29/73
                        ;RT-11 LINE PRINTER (LP/LS11) HANDLER
                        ;
                        ;DEC-11-ORTLA-A-LA
                        ;
                        ;ROBERT BEAN/ERIC PETERS
                        ;
                        ;MARCH 1973
                        ;
                        ;COPYRIGHT 1973
                        ;DIGITAL EQUIPMENT CORPORATION
                        ;MAYNARD,MASSACHUSETTS 01754
                        ;DEC ASSUMES NO RESPONSIBILITY FOR THE
                        ;USE OR RELIABILITY OF ITS SOFTWARE ON
                        ;EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
        000000'         .CSECT LP11
        000000          R0=%0
        000003          R3=%3
        000005          R5=%5
        000006          SP=%6
        000007          PC=%7
                        ;LINE PRINTER CONTROL REGISTERS
        177514          LPS=177514              ;LINE PRINTER CONTROL REGISTER
        177516          LPB=177516              ;LINE PRINTER DATA BUFFER
        000200          LPVEC=200               ;LINE PRINTER VECTOR ADDR
                        ;CONSTANTS FOR MONITOR COMMUNICATION
        000001          HDERR=1                 ;HARD ERROR BIT
        000054          MONLOW=54               ;BASE ADDR OF MONITOR
        000270          OFFSET=270              ;POINTER TO Q MANAGER COMP ENTRY
                        ;ASCII CONSTANTS
        000015          CR=15
        000012          LF=12
        000014          FF=14
        000011          HT=11
        000120          COLSIZ=80.              ;204 FOR 132 COLS
                        .GLOBL COLCNT
```

```

                      ;LOAD POINT
000000 000200                 .WORD LPVEC           ;ADDR OF INTERRUPT VECTOR
000002 000032                 .WORD LPINT-.         ;OFFSET TO INTERRUPT SERVICE
000004 000200                 .WORD 200             ;PRIORITY 4
000006 000000 LPLQE:          .WORD 0               ;POINTER TO LAST Q ENTRY
000010 000000 LPCQE:          .WORD 0               ;POINTER TO CURREMT Q ENTRY

                      ;ENTRY POINT

00012  016703 LP:             MOV LPCQE,R3          ;R3 POINTS TO CURRENT Q ENTRY
       177772
00016  006363                 ASL 6(R3)             ;WORD COUNT TO BYTE COUNT
       000006
00022  103100                 BCC LPERR             ;A READ REQUEST IS ILLEGAL
00024  052737                 BIS #100,@#LPS        ;CAUSE AN INTERRUPT,STARTING TRA
       000100
       177514
00032  000207                 RTS PC

                      ;INTERRUPT SERVICE

00034  005737 LPINT:          TST @#LPS             ;ERROR CONDITION?
       177514
00040  100433                 BMI RET               ;YES-HANG TILL CORRECTED
00042  010346                 MOV R3,-(SP)          ;SAVE R3
00044  106327                 ASLB (PC)+            ;TAB IN PROGRESS?
00046  000000 TABFLG:         .WORD 0
00050  001056                 BNE TAB               ;BRANCH IF DOING TAB
00052  016703 IGNORE:         MOV LPCQE,R3          ;SET UP R3
       177732
00056  005723                 TST (R3)+             ;IS THIS BLOCK 0?
00060  001455                 BEQ BLK0              ;YES-OUTPUT AN INITAIL FORM FEED
00062  005723                 TST (R3)+             ;POINT R3 TO BUFFER ADDRESS
00064  113346                 MOVB @(R3)+,-(SP)     ;GET NEXT CHAR (IF ANY)
00066  005713                 TST (R3)              ;ANY MORE CHARS?
00070  001465                 BEQ LPDONE            ;NO:FINISHED
00072  005213                 INC (R3)              ;YES-DECEMENT COUNT (IT WAS NEGA
00074  005243                 INC -(R3)             ;BUMP BUFFER POINTER
00076  012603                 MOV (SP)+,R3          ;CHAR INTO R3
00100  120327                 CMPB R3,#40           ;PRINTING CHAR?
       000040
00104  103412                 BLO CHRTST            ;NO-GO TEST FOR SPECIAL CHAR.
00106  005327 PCHAR:          DEC (PC)+             ;ANY ROOM LEFT ON LINE?
00110  000120 COLCNT:         .WORD COLSIZ          ;# OF PRINTER COLUMNS LEFT
00112  002757                 BLT IGNORE            ;NO MORE ROOM ON LINE,DON'T PRIN
00114  106327                 ASLB (PC)+            ;UPDATE TAB COUNT
00116  000001 TABCNT:         .WORD 1
00120  001423                 BEQ RSTTAB            ;RESET TAB
00122  110337 PC1:            MOVB R3,@#LPB         ;PRINT THE CHAR
       177516
00126  012603                 MOV (SP)+,R3          ;RESTORE R3
00130  000002 RET:            RTI
```

```
1 000132 120327 CHRTST: CMPB R3,#HT             ;IS CHAR A TAB?
         000011
2 000136 001420         BEQ TABSET              ;YES-RESET TAB
3 000140 120327         CMPB R3,#CR             ;IS IT CR?
         000015
4 000144 001406         BEQ RSTC                ;YES-RESTORE COLUMN COUNT
5 000146 120327         CMPB R3,#LF             ;IS IT LF?
         000012
6 000152 001403         BEQ RSTC                ;YES-RESTORE COLUMN COUNT
7 000154 120327         CMPB R3,#FF             ;IS IT A FF?
         000014
8 000160 001334         BNE IGNORE              ;NO-CHAR IS NON-PRINTING
9 000162 012767 RSTC:   MOV #COLSIZ,COLCNT      ;RE-INITIALIZE COLUMN COUNTER
         000120
         177720
10 00170 012767 RSTTAB: MOV #1,TABCNT           ;RESET TAB COUNTER
         000001
         177720
11 00176 000751         BR PC1                  ;PRINT THE CHAR
12 00200 016767 TABSET: MOV TABCNT,TABFLG       ;SET UP TAB
         177712
         177640
13 00206 012703 TAB:    MOV #40,R3              ;PRINT SPACES
         000040
14 00212 000735         BR PCHAR
15
16 00214 005243 BLK0:   INC -(R3)               ;MAKE SURE WE ONLY COME HERE ONC
17 00216 012703         MOV #FF,R3              ;PRINT INITIAL FF
         000014
18 00222 000757         BR RSTC
19
20 00224 005046 LPERR:  CLR -(SP)
21 00226 004767         JSR PC,1$               ;FAKE AN INTERRUPT
         000002
22 00232 000207         RTS PC                  ;RETURN TO Q MGR
23 00234 052753 1$:     BIS #HDERR,@-(R3)       ;SET HARD ERROR BIT
         000001
24 00240 010346         MOV R3,-(SP)
25 00242 005746         TST -(SP)               ;NORMAL RETURN HAS CHAR ON STACK
26
27              ;OPERATION COMPLETE
28
29 00244 005037 LPDONE: CLR @#LPS               ;TURN OFF INTERRUPT
         177514
30 00250 010703         MOV PC,R3
31 00252 062703         ADD #LPCQE-.,R3         ;ADDR OF CQE IN R3
         177536
32 00256 010016         MOV R0,(SP)             ;PUT R0 ON STACK OVER LAST CHAR
33 00260 013700         MOV @#MONLOW,R0
         000054
34 00264 016007         MOV OFFSET(R0),PC       ;JUMP TO Q MANAGER
         000270
35
36       000001'        .END
```

SYMBOL TABLE

```
BLK0      000214R    002  CHRTST    000132R    002  COLCNT    000110RG   002
COLSIZ= 000120            CR     =  000015          FF     =  000014
HDERR  = 000001           HT     =  000011          IGNORE    000052R    002
LF     = 000012           LP        000012R    002  LPB    =  177516
LPCQE     000010R    002  LPDONE    000244R    002  LPERR     000224R    002
LPINT     000034R    002  LPLQE     000006R    002  LPS    =  177514
LPVEC  = 000200           MONLOW=   000054          OFFSET=   000270
PC     =%000007           PCHAR     000106R    002  PC1       000122R    002
RET       000130R    002  RSTC      000162R    002  RSTTAB    000170R    002
R0     =%000000           R3     =%000003           R5     =%000005
SP     =%000006           TAB       000206R    002  TABCNT    000116R    002
TABFLG    000046R    002  TABSET    000200R    002
. ABS.    000000     000
          000000     001
LP11      000270     002
ERRORS DETECTED: 0
FREE CORE: 6964. WORDS
```

```
                        .TITLE DT  V01-02  5/29/73

                        ;RT-11 DECTAPE (TC11) HANDLER
                        ;
                        ;DEC-11-ORTDA-A-LA
                        ;
                        ;ROBERT BEAN
                        ;
                        ;MARCH 1973
                        ;
                        ;COPYRIGHT 1973
                        ;DIGITAL EQUIPMENT CORPORATION
                        ;MAYNARD, MASSACHUSETTS 01754

                        ;DEC ASSUMES NO RESPONSIBILITY FOR THE
                        ;USE OR RELIABILITY OF ITS SOFTWARE ON
                        ;EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.


        000000'         .CSECT TC11



        000000          R0=%0
        000001          R1=%1
        000003          R3=%3
        000005          R5=%5
        000006          SP=%6
        000007          PC=%7
        177776          PS=177776

                ;DECTAPE CONTROL REGISTERS
        177350 TCDT=177350              ;DATA REGISTER
        177340 TCST=177340              ;CONTROL AND STATUS REGISTER
        177342 TCCM=177342              ;COMMAND REGISTER
        177344 TCWC=177344              ;WORD COUNT REGISTER
        177346 TCBA=177346              ;BUS ADDRESS REGISTER
        000214 TCVEC=214                ;TC11 INTERRUPT VECTOR

                ;CONSTANTS FOR MONITOR COMMUNICATION
        000001 HDERR=1                  ;HARD ERROR BIT
        000054 MONLOW=54                ;MONITOR BASE POINTER
        000270 OFFSET=270               ;POINTER TO Q MANAGER COMP ENTRY

                        .GLOBL DTSYS,RKSYS
                        .GLOBL SYINT,SYINPR
                        .GLOBL SBLOCK,SYSIZE,SYSTAT,SY

        001102          SBLOCK=1102             ;1102 OCTAL BLOCKS ON EMPTY DEVI
        100001          SYSTAT=100001           ;FILE STRUCTURED DEVICE,DECTAPE
        000000          RKSYS=0                 ;RK IS NOT RESIDENT
```

```

000000            BEG:
                  ;LOAD POINT
000000  000214            .WORD TCVEC             ;ADDRESS OF INTERRUPT VECTOR
000002  000026            .WORD DTINT-.           ;OFFSET TO INTERRUPT SERVICE
000004  000300            .WORD 300               ;PRIORITY 6
000006            DTSYS:
000006            SY:
000006  000000    DTLQE:  .WORD 0                 ;POINTER TO LAST Q ENTRY
 00010  000000    DTCQE:  .WORD 0                 ;POINTER TO CURRENT Q ENTRY

                  ;ENTRY POINT
 00012  012727            MOV #8.,(PC)+           ;INIT THE RETRY COUNT
        000010
 00016  000000    DTTRY:  .WORD 0                 ;RETRY COUNTER
 00020  005046            CLR -(SP)
 00022  004767            JSR PC,DTINT            ;PSEUDO-INTERRUPT ON STACK
        000002
 00026  000207            RTS PC                  ;CONTROL RETURNS HERE WHEN OPERA
                                                  ;HAS BEEN STARTED

                  ;INTERRUPT SERVICE

 00030            SYINT:
        000300    SYINPR=300
 00030  010346    DTINT:  MOV R3,-(SP)            ;SAVE REGS
 00032  010046            MOV R0,-(SP)
 00034  010146            MOV R1,-(SP)
 00036  016700            MOV DTCQE,R0            ;R0 POINTS TO Q ELEMENT
        177746
 00042  012701            MOV #TCCM,R1            ;R1 POINTS TO CONTROL REGISTER
        177342
 00046  012046            MOV (R0)+,-(SP)         ;DESIRED BLOCK # ONTO STACK
 00050  012003            MOV (R0)+,R3            ;UNIT # INTO R3
 00052  032711            BIT #100100,(R1)        ;ERROR BIT ON?
        100100
 00056  100454            BMI DTERR               ;YES
 00060  001503            BEQ RETRY               ;IF INTERRUPT IS OFF,WE ARE INIT
                                                  ;A REQUEST
 00062  032711            BIT #2,(R1)             ;SEARCHING?
        000002
 00066  001465            BEQ DTDONE              ;NO-A READ OR WRITE JUST COMPLET
 00070  023767            CMP @#TCDT,BWANT        ;COMPARE ACTUAL BLOCK TO DESIRED
        177350
        000210
 00076  001424            BEQ BLKFND              ;FOUND IT
```

```
000100 002407 DIRECT: BLT FORWARD         ;SEARCH IN THE FORWARD DIRECTION
000102 052703 REVERSE:BIS #4000,R3        ;SET REVERSE BIT
       004000
000106 1627[illegible]        SUB #2,(SP)         ;SEARCH FOR TWO BLOCKS BEFORE ON
       000002
                                          ;ACTUALLY DESIRED (TO ALLOW
                                          ;SPACE FOR THE TURN-AROUND
000112 000402         BR FORWARD          ;DON'T SET DELAY INHIBIT
000114 052703 FORW1:  BIS #10000,R3       ;TAPE IS ALREADY MOVING FORWARD
       010000
                                          ;SO INHIBIT HARDWARE DELAY
000120 052703 FORWARD:BIS #103,R3         ;INTERRUPT ENABLE,RNUM,AND GO
       000103
00124 012667          MOV (SP)+,BWANT     ;REMEMBER THE BLOCK WE ARE LOOKI
       000156
00130 010311 RETRNI:  MOV R3,(R1)         ;TELL CONTROLLER TO GO
00132 012601          MOV (SP)+,R1        ;RESTORE REGS
00134 012600          MOV (SP)+,R0
00136 012603          MOV (SP)+,R3
00140 000002          RTI

00142 032711 ENDZR:   BIT #4000,(R1)      ;WERE WE IN REVERSE?
       004000
00146 001755          BEQ REVERSE         ;NO-REVERSE TAPE
00150 032711 BLKFND:  BIT #4000,(R1)      ;WERE WE GOING FORWARD?
       004000
00154 001361          BNE FORWARD         ;NO-WE HAVE TO TURN AROUND

              ;INITIATE READ/WRITE REQUEST

00156 052703          BIS #10115,R3       ;ASSUME WRITE
       010115
00162 012037          MOV (R0)+,@#TCBA    ;CORE ADDRESS
       177346
00166 011016          MOV (R0),(SP)       ;WORD COUNT (OVER BLOCK #)
00170 100404          BMI 1$              ;WRITE WAS A GOOD GUESS
00172 001423          BEQ DTDONE          ;IF ZERO,SEEK
00174 005416          NEG (SP)            ;READ-NEGATE WORD COUNT
00176 042703          BIC #10,R3          ;SET READ FUNCTION
       000010
00202 012637 1$:      MOV (SP)+,@#TCWC    ;SET WORD COUNT
       177344
00206 000750          BR RETRNI
```

```
                           ;ERROR ROUTINE

   000210 032737 DTERR:   BIT #104000,@#TCS1      ;ENDZ ERROR?
          104000
          177340
   000216 100003          BPL NOTEZ               ;NOT ENDZ
   000220 032711          BIT #2,(R1)             ;WERE WE SEARCHING?
          000002
   000224 001346          BNE ENDZR               ;YES-REVERSE TAPE
   000226 005367 NOTEZ:   DEC DTTRY               ;MORE TRIES LEFT?
          177564
   000232 003016          BGT RETRY               ;YES
   00234 052770           BIS #HDERR,@-6(R0)      ;NO-SET HARD ERROR BIT
          000001
          177772

                           ;OPERATION FINISHED

   00242 112711 DTDONE:   MOVB #11,(R1)           ;STOP TAPE
          000011
   00246 005726           TST (SP)+               ;POP BLOCK #
   00250 012601           MOV (SP)+,R1            ;RESTORE R1
   00252 010703           MOV PC,R3
   00254 062703           ADD #DTCQE-.,R3         ;ADDR OF CQE IN R3
          177534
   00260 013700           MOV @#MONLOW,R0
          000054
   00264 016007           MOV OFFSET(R0),PC       ;JUMP TO Q MANAGER
          000270

                           ;RETRY CODE

   00270 105737 RETRY:    TSTB @#TCST             ;TAPE UP TO SPEED?
          177340
   00274 100707           BMI FORW1               ;YES-AVOID STOPPING TAPE
   00276 062767           ADD #4,BWANT            ;NO-IT TAKES 4 BLOCKS TO START A
          000004
          000002
   00304 022716           CMP (PC)+,(SP)          ;MAKE AN ATTEMPT TO START IN THE
   00306 000000 BWANT:    0                       ;RIGHT DIRECTION BASED ON LAST B
   00310 000673           BR DIRECT               ;DESIRED

          000312          SYSIZE=.-BEG            ;SIZE OF DT HANDLER

          000001'         .END
```

SYMBOL TABLE

```
BEG       000000R    002  BLKFND   000150R    002  BWANT    000306R    002
DIRECT    000100R    002  DTCQE    000010R    002  DTDONE   000242R    002
DTERR     000210R    002  DTINT    000030R    002  DTLQE    000006R    002
DTSYS     000006RG   002  DTTRY    000016R    002  ENDZR    000142R    002
FORWAR    000120R    002  FORW1    000114R    002  HUERR  = 000001
MONLOW=   000054          NOTEZ    000226R    002  OFFSET= 000270
PC      =%000007          PS     = 177776          RETRNI   000130R    002
RETRY     000270R    002  REVERS   000102R    002  RKSYS  = 000000 G
R0      =%000000          R1     =%000001          R3     =%000003
R5      =%000005          SBLOCK= 001102 G         SP     =%000006
SY        000006RG   002  SYINPR= 000300 G         SYINT    000030RG   002
SYSIZE=   000312 G        SYSTAT= 100001 G         TCBA   = 177346
TCCM    = 177342          TCDT   = 177350          TCST   = 177340
TCVEC   = 000214          TCWC   = 177344
. ABS.    000000     000
          000000     001
TC11      000312     002
ERRORS DETECTED: 0
FREE CORE: 6940. WORDS
```

APPENDIX B

DETAILED OPERATION OF BOOTSTRAP

Bootstrapping a system causes a fresh copy of that system to be installed in core. In the RT-11 boot, certain system device resident tables are also updated. Following is a detailed description of the bootstrap:

Action	Explanation
1. User executes hardware bootstrap.	This causes block 0 of the system device to be read into 0-777. Control then passes to location 0.
2. Second part of bootstrap is read.	The first part of the boot reads the second half into 1000-1777.
3. Determine how much core is available.	Boot sets a trap at location 4 and then starts addressing memory. When the trap is taken, illegal memory has been addressed.
4. Read in directory and find MONITR.SYS.	The entire directory is searched. If no such file is found, a HALT occurs, as no valid monitor was found.
5. Read the monitor into 8K.	The monitor contained in MONITR.SYS is used in 8K to perform the rest of the bootstrap.
6. LOOKUP the device handlers in system and store their record numbers in $DVREC.	Boot looks at $PNAME table to find the names of the devices in the system. The extension .SYS is appended. Thus, the PR handler is a file called PR.SYS. The location of the handler is then placed in $DVREC. If the LOOKUP fails, the device gets a 0 in the devices $DVREC entry. That implies that the device does not exist.
7. Write out table $DVREC.	The block of MONITR.SYS which holds $DVREC is re-written. That allows the USR to FETCH handlers without doing a LOOKUP each time.
8. Put pointers to monitor file blocks into RMON.	RMON references the monitor swap blocks directly. Thus the position of the swap blocks will vary as the placement of MONITR.SYS varies. The real position of the blocks is updated for each boot operation.
9. Update position dependent areas in RMON.	MONITR.SYS is initially linked for 8K. However, if more than 8K is available, RT-11 uses it. To do that, certain words must be updated to point to the

Action	Explanation
	actual areas of high core where they will be. Boot contains a list of all words to be updated.
10. Move the monitor to high core.	If more than 8K of core was found, the system is moved to the top of core.
11. Print bootstrap header.	"RT-11 V01-15" or the current version message.
12. Set up locations 4 and 10 to HALT on traps. Put an .EXIT EMT into locations 0 and 2.	
13. EXIT to keyboard Monitor.	

APPENDIX C

FIXING THE SIZE OF A SYSTEM

RT-11 is designed to automatically operate out of the top of the highest available 4K memory bank. However, it is possible to force the system to operate out of a specified 4K bank which is not necessarily the highest. For instance, RT-11 may be run in a 16K environment, even though the configuration actually has 28K of core. This can be done as follows:

Commands	Explanation
.R PATCH	Run RT-11 PATCH program.
PATCH V01-01	
FILE NAME--	
*MONITR.SYS/M	Specifying MONITR.SYS/M indicates it is a monitor file.
*316/ 405 0	Change location 316 from a 405 to a 0 (HALT).
*E	E causes an exit to the
.R PIP	monitor. Now run PIP to
*A=MONITR.SYS/U	update the bootstrap
*SY:/O	and re-boot the system.

When the bootstrap is performed, the computer halts. The halt allows the user to enter the desired size in the switch register. The switch settings are:

Switch Register	Core Size Used
0	8K
20000	12K
40000	16K
60000	20K
100000	24K
120000	28K

When the switches are set properly, press the CONTinue switch and the bootstrap will be executed. If the CONTinue switch is pressed immediately following the halt, without changing the switch settings, a normal core determination is done. To change the bootstrap back to its original (non-halting) form, execute the same commands as above, but change the 0 at 316 back to a 405.

This procedure allows the user to have 'protected' core areas, as RT-11 never accesses memory outside the bounds within which it runs.

INDEX

HOW TO OBTAIN SOFTWARE INFORMATION

SOFTWARE NEWSLETTERS, MAILING LIST

The Software Communications Group, located at corporate headquarters in Maynard, publishes newsletters and Software Performance Summaries (SPS) for the various Digital products. Newsletters are published monthly, and contain announcements of new and revised software, programming notes, software problems and solutions, and documentation corrections. Software Performance Summaries are a collection of existing problems and solutions for a given software system, and are published periodically. For information on the distribution of these documents and how to get on the software newsletter mailing list, write to:

Software Communications
P. O. Box F
Maynard, Massachusetts 01754

SOFTWARE PROBLEMS

Questions or problems relating to Digital's software should be reported to a Software Support Specialist. A specialist is located in each Digital Sales Office in the United States. In Europe, software problem reporting centers are in the following cities.

Reading, England	Milan, Italy
Paris, France	Solna, Sweden
The Hague, Holland	Geneva, Switzerland
Tel Aviv, Israel	Munich, West Germany

Software Problem Report (SPR) forms are available from the specialists or from the Software Distribution Centers cited below.

PROGRAMS AND MANUALS

Software and manuals should be ordered by title and order number. In the United States, send orders to the nearest distribution center.

Digital Equipment Corporation
Software Distribution Center
146 Main Street
Maynard, Massachusetts 01754

Digital Equipment Corporation
Software Distribution Center
1400 Terra Bella
Mountain View, California 94043

Outside of the United States, orders should be directed to the nearest Digital Field Sales Office or representative.

USERS SOCIETY

DECUS, Digital Equipment Computer Users Society, maintains a user exchange center for user-written programs and technical application information. A catalog of existing programs is available. The society publishes a periodical, DECUSCOPE, and holds technical seminars in the United States, Canada, Europe, and Australia. For information on the society and membership application forms, write to:

DECUS
Digital Equipment Corporation
146 Main Street
Maynard, Massachusetts 01754

DECUS
Digital Equipment, S.A.
81 Route de l'Aire
1211 Geneva 26
Switzerland

READER'S COMMENTS

NOTE: This form is for document comments only. Problems with software should be reported on a Software Problem Report (SPR) form (see the HOW TO OBTAIN SOFTWARE INFORMATION page).

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

____
____
____
____
____
____

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

____
____
____
____
____
____

Is there sufficient documentation on associated system programs required for use of the software described in this manual? If not, what material is missing and where should it be placed?

____
____
____
____
____
____

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Non-programmer interested in computer concepts and capabilities

Name ______________________ Date ______________

Organization ______________________________________

Street ____

City ______________ State __________ Zip Code or Country __________

If you do not require a written reply, please check here. ☐

-- **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

Software Communications
P. O. Box F
Maynard, Massachusetts 01754

digital

DIGITAL EQUIPMENT CORPORATION
MAYNARD, MASSACHUSETTS 01754