

BBBBBBBBBBBB		000000000		000000000		TTTTTTTTTTTT		SSSSSSSSSS
BBBBBBBBBBBB		000000000		000000000		TTTTTTTTTTTT		SSSSSSSSSS
BBBBBBBBBBBB		000000000		000000000		TTTTTTTTTTTT		SSSSSSSSSS
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBBBBBBBBBBB		000	000	000	000	TTT	SSS	
BBBBBBBBBBBB		000	000	000	000	TTT	SSS	
BBBBBBBBBBBB		000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBBBBBBBBBBB		000000000		000000000		TTT	SSSSSSSSSS	
BBBBBBBBBBBB		000000000		000000000		TTT	SSSSSSSSSS	
BBBBBBBBBBBB		000000000		000000000		TTT	SSSSSSSSSS	

```

RRRRRRRR      TTTTTTTTTT  FFFFFFFFFF  IIIIII  LL      RRRRRRRR  EEEEEEEEEEE  AAAAAA  DDDDDDDD
RRRRRRRR      TTTTTTTTTT  FFFFFFFFFF  IIIIII  LL      RRRRRRRR  EEEEEEEEEEE  AAAAAA  DDDDDDDD
RR          RR  TT          FF          II      LL      RR          RR  EE          AA          AA  DD          DD
RR          RR  TT          FF          II      LL      RR          RR  EE          AA          AA  DD          DD
RR          RR  TT          FF          II      LL      RR          RR  EE          AA          AA  DD          DD
RR          RR  TT          FF          II      LL      RR          RR  EE          AA          AA  DD          DD
RRRRRRRR      TT          FFFFFFFF      II      LL      RRRRRRRR  EEEEEEEEE  AA          AA  DD          DD
RRRRRRRR      TT          FFFFFFFF      II      LL      RRRRRRRR  EEEEEEEEE  AA          AA  DD          DD
RR  RR          TT          FF          II      LL      RR  RR          EE          AAAAAAAAAA  DD          DD
RR  RR          TT          FF          II      LL      RR  RR          EE          AAAAAAAAAA  DD          DD
RR          RR  TT          FF          II      LL      RR          RR  EE          AA          AA  DD          DD
RR          RR  TT          FF          II      LL      RR          RR  EE          AA          AA  DD          DD
RR          RR  TT          FF          IIIIII  LLLLLLLLLL  RR          RR  EEEEEEEEEEE  AA          AA  DDDDDDDD
RR          TT          FF          IIIIII  LLLLLLLLLL  RR          RR  EEEEEEEEEEE  AA          AA  DDDDDDDD

```

```

LL          IIIIII  SSSSSSSS
LL          IIIIII  SSSSSSSS
LL          II      SS
LL          II      SS
LL          II      SS
LL          II      SS
LL          II      SSSSSS
LL          II      SSSSSS
LL          II      SS
LL          II      SS
LL          II      SS
LL          II      SS
LLLLLLLLLLL  IIIIII  SSSSSSSS
LLLLLLLLLLL  IIIIII  SSSSSSSS

```


(2) 60
(3) 130

Declarations
RTF\$OPENFILE - Looks up a file on an RT-11 volume

```
0000 1 .TITLE RTFILREAD - Lookup facility for RT-11 Files Structure
0000 2 .IDENT 'V04-000'
0000 3
0000 4
0000 5 *****
0000 6
0000 7 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9 * ALL RIGHTS RESERVED.
0000 10
0000 11 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16 * TRANSFERRED.
0000 17
0000 18 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20 * CORPORATION.
0000 21
0000 22 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24
0000 25
0000 26 *****
0000 27
0000 28 ++
0000 29
0000 30 FACILITY:
0000 31
0000 32 Lookup facility for files located on a RT-11 Files Structure
0000 33
0000 34 ENVIRONMENT:
0000 35
0000 36 Runs at IPL 31, kernel mode, memory management is OFF, IS=1
0000 37 (running on interrupt stack), and code must be PIC.
0000 38
0000 39 ABSTRACT:
0000 40
0000 41 This module contains code to locate a named file on an RT-11
0000 42 volume.
0000 43
0000 44 AUTHOR:
0000 45
0000 46 Carol Peters 22 March 1979
0000 47
0000 48 MODIFIED BY:
0000 49
0000 50 V03-001 CWH0001 CW Hobbs 14-May-1983
0000 51 Corrected the interpretation of RT-11 file attributes:
0000 52 file type is a 4-bit field (not a byte) and TENTATIVE
0000 53 files should be treated as EMPTY files, not as fatal errors.
0000 54
0000 55 V02-002 KTA0039 Kerbey T. Altmann 28-Oct-1981
0000 56 Change PSECT to YFILEREAD to match other boot IO.
0000 57
```


RTFILREAD
V04-000

- Lookup facility for RT-11 Files Struct I 13 16-SEP-1984 00:00:43 VAX/VMS Macro V04-00 Page 2
4-SEP-1984 23:05:36 [BOOTS.SRC]RTFILREAD.MAR;1 (1)

0000 58 ;--

```
0000 60 .SBTTL Declarations
0000 61
0000 62 :
0000 63 : Macros
0000 64 :
0000 65
0000 66 $DSCDEF : String descriptor definitions
0000 67 $IODEF : I/O function code definitions
0000 68 $$$DEF : Status code definitions
0000 69
0000 70 :
0000 71 : Field definitions for RT-11 directory segments and directory entries.
0000 72 :
0000 73
0000 74 $DEFINI RTF
0000 75
0000 76 $DEF RTF_W_NUMSEGS : Number of directory segments
0000 77 .BLKW 1 : available on this volume.
00000002 0000 78 $DEF RTF_W_NEXTSEG : Number of the next directory
00000004 0002 79 .BLKW 1 : segment after this one.
00000006 0004 80 $DEF RTF_W_HIGHSEG : Number of highest directory
00000006 0004 81 .BLKW 1 : segment on this volume.
00000008 0006 82 $DEF RTF_W_EXTRBYTES : Number of extra bytes in each
00000008 0006 83 .BLKW 1 : directory entry.
0000000A 0008 84 $DEF RTF_W_STARTLBN : LBN of the file described by
0000000A 0008 85 .BLKW 1 : first entry in this segment.
0000000E 000A 86 $DEF RTF_L_ENTRYONE : First directory entry in
0000000E 000A 87 .BLKL 1 : directory segment.
00000000 000E 88
00000000 000E 89 .=0 : Reset base address.
0000 90
0000 91 $DEF RTF_W_STATUS : Directory entry status.
0000 92 $DEF RTF_B_EVENSTAT : Even byte of entry status.
00000001 0000 93 .BLKB 1
00000001 0001 94 $DEF RTF_B_ODDSTAT : Odd byte of entry status.
00000002 0001 95 .BLKB 1
00000002 0002 96 $DEF RTF_L_FILENAME : 6-character RAD50 format file
00000006 0002 97 .BLKL 1 : name.
00000006 0006 98 $DEF RTF_W_FILTYPE : 3 characters of file type in
00000008 0006 99 .BLKW 1 : RAD50 format.
00000008 0008 100 $DEF RTF_W_FILLENGTH : Length of file in blocks.
0000000A 0008 101 .BLKW 1
0000000A 000A 102 $DEF RTF_B_CHAN_NUM : Number of channel assigned to
0000000B 000A 103 .BLKB 1 : the entry's file.
0000000B 000B 104 $DEF RTF_B_JOBNUM : Number of job using the
0000000C 000B 105 .BLKB 1 : entry's file.
0000000C 000C 106 $DEF RTF_W_DATE : Date of the entry's file.
0000000E 000C 107 .BLKW 1
0000000E 000E 108 $DEF RTF_K_ENTLENGTH : Length of fixed entry.
0000 109
0000 110 $DEFEND RTF
0000 111 :
0000 112 : Equated symbols:
0000 113 :
0000 114
00000001 0000 115 RTF_C_TENTAFILE = 1 : Tentative (open) file entry.
00000002 0000 116 RTF_C_EMPTYFILE = 2 : Empty file entry.
```


RTFILREAD
V04-000

K 13
- Lookup facility for RT-11 Files Struct 16-SEP-1984 00:00:43 VAX/VMS Macro V04-00
Declarations 4-SEP-1984 23:05:36 [BOOTS.SRC]RTFILREAD.MAR;1

Page 4
(2)

```
00000004 0000 117      RTF_C_PERMAFILE = 4      ; Permanent (normal) file.
00000008 0000 118      RTF_C_ENDOFSEG = 8      ; End of segment entry.
          0000 119
          0000 120      ;
          0000 121      ; Own storage.
          0000 122      ;
          0000 123
00000000 124      .PSECT YFILEREAD, LONG
          0000 125
          0000 126      .SHOW EXPANSIONS
          0000 127
          0000 128      .DEFAULT DISPLACEMENT, WORD
```

0000 130 .SBTTL RTF\$OPENFILE - Looks up a file on an RT-11 volume

0000 131
0000 132 :++
0000 133 : Functional description:

0000 134 : This routine translates an ASCII file name and type to RAD50.
0000 135 : Then the routine interprets an RT-11 directory to see whether
0000 136 : the named file is in the directory. If the file is present, this
0000 137 : routine returns the starting LBN of the file and the file's
0000 138 : length in blocks.
0000 139 :
0000 140 :

0000 141 : The RT-11 directory structure consists of 1-31 2-block directory
0000 142 : segments. The first segment starts at block 6, and subsequent
0000 143 : segments follows contiguously; i.e., at blocks 8, 10, etc. Each
0000 144 : segment starts with a 5-word header:
0000 145 :

number of 2-block segments	:0
segment number of next segment	:2
highest segment currently open	:4
number of extra bytes/directory entry	:6
starting LBN of 1st file in this segment	:8

0000 146 :
0000 147 :
0000 148 :
0000 149 :
0000 150 :
0000 151 :
0000 152 :
0000 153 :
0000 154 :
0000 155 :
0000 156 :
0000 157 :
0000 158 : The segment number field identifies the next segment to search.
0000 159 : The LBN of the first block of a directory segment is

0000 160 : $6 + \ll \text{segment_number} - 1 \gg * 2 \gg$
0000 161 :

0000 162 : Following the directory segment header are the directory entries
0000 163 : for this segment. A directory entry is $7 + (n/2)$ words long:
0000 164 :
0000 165 :

status word: tells type of file	:0
1st 3 characters of filename in RAD50	:2
2nd 3 characters of filename in RAD50	:4
3 characters of filetype in RAD50	:6
number of blocks in file	:8
job number : channel number	:A
date	:C
'n' extra words	:E
.	
.	

0000 166 :
0000 167 :
0000 168 :
0000 169 :
0000 170 :
0000 171 :
0000 172 :
0000 173 :
0000 174 :
0000 175 :
0000 176 :
0000 177 :
0000 178 :
0000 179 :
0000 180 :
0000 181 :
0000 182 :
0000 183 :
0000 184 :
0000 185 :
0000 186 : Bits 8-11 of the status word have the following meanings:


```
0000 187 :
0000 188 :
0000 189 :
0000 190 :
0000 191 :
0000 192 :
0000 193 :
0000 194 :
0000 195 :
0000 196 :
0000 197 :
0000 198 :
0000 199 :
0000 200 :
0000 201 :
0000 202 :
0000 203 :
0000 204 :
0000 205 :
0000 206 :
0000 207 :
0000 208 :
0000 209 :
0000 210 :
0000 211 :
0000 212 :
0000 213 :
0000 214 :
0000 215 :
0000 216 :
0000 217 :
0000 218 :
0000 219 :
0000 220 :
0000 221 :
0000 222 :
0000 223 :
0000 224 :
0000 225 :
0000 226 :
0000 227 :
0000 228 :
0000 229 :
0000 230 :
0000 231 :
00000004 0000 232 :
00000008 0000 233 :
0000000C 0000 234 :
0000 235 :
OFFC 0000 236 :
0002 237 :
0002 238 :
0002 239 :
0002 240 :
0002 241 :
0002 242 :
0002 243 :
```

1 tentative file (open), treat as empty
2 empty file; length is the other valid field
4 normal file
10 end of segment marker

The algorithm to lookup a file is the following:

- a) Get first segment.
- b) Compare target filename with each entry in segment.
- c) If no match, get next segment, if any; goto b).
- d) If match, compute starting LBN and length of file.
- e) Return.

Inputs:

- FILE_DESC(AP) - string descriptor of target file specification
- DIRSEG_BLKs(AP) - address of 512-word buffer for the 2 blocks of a directory segment
- STAT_BLOCK(AP) - address of a 2 longword block in which the following is returned:
 - LBN of 1st block of file
 - size of files in blocks

Outputs:

- R0 - status code; can be the following:
 - error codes from FIL\$CVTFILNAM:
 - SS\$_BADFILENAME, file name too long or non 0 version #
 - error codes from FIL\$READ_LBN:
 - SS\$_BADIRECTORY, directory contains tentative (non-closed) directory entries, or entries with an invalid status field value
 - SS\$_NOSUCHFILE, file does not exist on volume

--

Offsets to input arguments:

- FILE_DESC = 4
- DIRSEG_BLKs = 8
- STAT_BLOCK = 12

.ENTRY RTFSOPENFILE,-
^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>

File name and type in RT-11 directories are in RAD50 format. Convert ASCII file specification from ASCII to RAD50. Use a 3-longword block on the stack to store the RAD50 characters.

	7E	7C	0002	244				
	7E	D4	0002	245	CLRQ	-(SP)	:	Allocate and zero a 6-word
	SE	DD	0004	246	CLRL	-(SP)	:	temp storage block.
			0006	247	PUSHL	SP	:	Use block address as an
			0008	248			:	argument.
0000'CF	04 AC	DD	0008	249	PUSHL	FILE_DESC(AP)	:	Push address of file desc.
	02	FB	000B	250	CALLS	#2,FIL\$CVTFILNAM	:	Convert filename to RAD50.
	01 50	E8	0010	251	BLBS	R0,PACK_FILNAM	:	On success, go pack filename.
		04	0013	252	RET		:	On failure, return with error
			0014	253				
			0014	254				
			0014	255	:	FIL\$CVTFILNAM writes the file name into words 0-2, the file type in		
			0014	256	:	word 3, and the binary file version number in word 4. RT-11 files		
			0014	257	:	are limited to 6-character file names, 3-character file types, and		
			0014	258	:	no version numbers. Thus the block looks like this:		
			0014	259	:			
			0014	260	:+-----+			
			0014	261	file name's 1st 3 characters	:	0	
			0014	262	+-----+			
			0014	263	file name's 2nd 3 characters	:	2	
			0014	264	+-----+			
			0014	265	must be zero	:	4	
			0014	266	+-----+			
			0014	267	3 character file type	:	6	
			0014	268	+-----+			
			0014	269	binary version number	:	8	
			0014	270	+-----+			
			0014	271	:			
			0014	272	:	Confirm that these requirements are met.		
			0014	273	:			
			0014	274	:			
			0014	275	PACK_FILNAM:	:	Pack filename/type in 3 bytes.	
	04 AE	B5	0014	276	TSTW	4(SP)	:	Name 6 characters only?
	05	12	0017	277	BNEQ	BAD_FILENAME	:	No. Name is too long. Branch.
	08 AE	B5	0019	278	TSTW	8(SP)	:	Version number 0?
	06	13	001C	279	BEQL	MVFE_FILE_TYPE	:	Yes. Go pack file type.
			001E	280				
			001E	281	BAD_FILENAME:	:	File name is bad.	
50	0818 8F	3C	001E	282	MOVZWL	#SS\$_BADFILENAME,R0	:	No. Load error status code.
		04	0023	283	RET		:	Return to caller.
			0024	284				
			0024	285	MVFE_FILE_TYPE:	:	Pack RAD50 file type.	
04 AE	06 AE	B0	0024	286	MOVB	6(SP),4(SP)	:	Move file type into 3rd word.
			0029	287				
			0029	288	:			
			0029	289	:	Read in the first 2-block directory segment into the buffer supplied		
			0029	290	:	as an input argument.		
			0029	291	:			
			0029	292	:			
55	06	D0	0029	293	MOVL	#6,R5	:	Directory segments begin at
			002C	294			:	LBN 6.
			002C	295				
			002C	296	READ_SEGMENT:	:	Read a directory segment.	
	08 AC	DD	002C	297	PUSHL	DIRSEG_BLK\$(AP)	:	Push input buffer address.
	55	DD	002F	298	PUSHL	R5	:	Send LBN to read.
	02	DD	0031	299	PUSHL	#2	:	Number of arguments pushed.
0000'CF	6E FA	FA	0033	300	CALLG	(SP),FIL\$READ_LBN	:	Read one block.


```
01 50 E8 0038 301 BLBS R0,GET_2ND_BLK ; On success, go get 2nd block.
04 003B 302 RET ; On failure, return with error.
003C 303
003C 304 GET_2ND_BLK: ; Read 2nd block of segment.
5E 04 C0 003C 305 ADDL #4,SP ; Take argument count off stack.
04 AE 00000200 8F D6 003F 306 INCL (SP) ; Increment LBN to read.
0000'CF 02 FB 0041 307 ADDL #512,4(SP) ; Add a page to buffer address.
004E 308 CALLS #2,FIL$READ_LBN ; Read second block. The RET
004E 309 ; from FIL$READ_LBN cleans up
004E 310 ; the stack.
01 50 E8 004E 311 BLBS R0,START_DIRSEG ; On success, process segment.
04 0051 312 RET ; On failure, return with error.
0052 313
0052 314
0052 315 ; If this is the first directory segment, save the number of the highest
0052 316 ; open segment. Then prepare some registers for use during the scan
0052 317 ; of this segment's directory entries:
0052 318
0052 319 R1 - address of directory segment
0052 320 R2 - address of current directory entry
0052 321 R3 - number of bytes per directory entry
0052 322 R4 - starting LBN of the file described by the next
0052 323 ; directory entry
0052 324 R5 - starting LBN of the current directory segment
0052 325 R6 - status of a directory entry;
0052 326 ; address of the statistics block from the argument list
0052 327 ; length of file corresponding to a directory entry
0052 328
0052 329
0052 330 START_DIRSEG: ; Start processing segment.
51 08 AC D0 0052 331 MOVL DIRSEG_BLK$AP,R1 ; Get address of segment buffer.
52 0A A1 9E 0056 332 MOVAB RTF_L_ENTRYONE(R1),R2 ; Address of first entry.
53 06 A1 3C 005A 333 MOVZWL RTF_W_EXTRBYTES(R1),R3 ; Get extra byte count/entry.
54 53 0E C0 005E 334 ADDL #RTF_R_ENTLENGTH,R3 ; Add to normal entry length.
0061 335 MOVZWL RTF_W_STARTLBN(R1),R4 ; Get 1st file's LBN.
0065 336
0065 337
0065 338 ; Now read one directory entry at a time. CASE on the status of the
0065 339 ; entry. Choices are
0065 340
0065 341 tentative file (just add length and go to next entry)
0065 342 empty entry (just add length and go to next entry)
0065 343 normal file (compare with our file name)
0065 344 end of segment (move to next segment, if any)
0065 345
0065 346
0065 347 LOOK_AT_ENTRY: ; Examine an entry.
56 01 A2 04 00 EF 0065 348 EXTZV #0,#4,RTF_B_ODDSTAT(R2),R6 ; Get the entry status.
08 56 D1 006B 349 CMPL R6,#RTF_C_ENDOFSEG ; If end of segment marker,
35 13 006E 350 BEQL END_OF_SEGMENT ; branch to process it.
01 56 D1 0070 351 CMPL R6,#RTF_C_TENTAFILE ; If entry describes an open
24 13 0073 352 BEQL NEXT_ENTRY ; tentative file, treat as empty.
02 56 D1 0075 353 CMPL R6,#RTF_C_EMPTYFILE ; If entry describes empty
1F 13 0078 354 BEQL NEXT_ENTRY ; space, add size, go to next.
04 56 D1 007A 355 CMPL R6,#RTF_C_PERMAFILE ; If not a normal file, report
38 12 007D 356 BNEQ BAD_DIRECTORY ; an error and return.
007F 357
```



```
007F 358 :  
007F 359 : This directory entry describes a normal permanent file. Compare the  
007F 360 : file name and type. If they match, return the file statistics.  
007F 361 :  
007F 362 :  
02 A2 D1 007F 363      CMPL   RTF_L_FILENAME(R2),-      ; If file name doesn't match,  
6E      364      (SP)                          ; move to next entry.  
14      0082 364      BNEQ   NEXT_ENTRY          ; Branch on no match.  
04 AE 06 A2 B1 0083 365      CMPW   RTF_W_FILTYPE(R2),4(SP); If file type doesn't match,  
OD      0085 366      BNEQ   NEXT_ENTRY          ; move to next entry.  
56 0C AC D0 008A 367      MOVL   STAT_BLOCK(AP),R6      ; Get address of statistics  
66 54 D0 008C 368      block.  
08 A2 3C 0090 369      MOVL   R4,(R6)              ; Return starting LBN of file.  
04 A6      0090 370      MOVZWL RTF_W_FILLENGTH(R2),-  ; Return length of file in  
0093 371      4(R6)                                ; blocks.  
0096 372 :  
0098 373 :  
0098 374 :  
0098 375 : R0 still contains a success code from the last call to READ_LBN.  
0098 376 :  
0098 377 :  
04 0098 378      RET                                ; Return with success.  
0099 379 :  
0099 380 :  
0099 381 : This entry does not match the file specification. Compute the new  
0099 382 : starting LBN of the next file, and move to the next directory entry.  
0099 383 :  
0099 384 :  
56 08 A2 3C 0099 385 NEXT_ENTRY:                    ; Try another entry.  
54 56 C0 0099 386      MOVZWL RTF_W_FILLENGTH(R2),R6      ; Get length of this file.  
52 53 C0 009D 387      ADDL   R6,R4-                    ; Add to starting LBN.  
C0 11 00A0 388      ADDL   R3,R2                        ; Compute address of next entry.  
00A3 389      BRB     LOOK_AT_ENTRY                    ; Check out next entry.  
00A5 390 :  
00A5 391 :  
00A5 392 : When an entry is an end-of-segment marker, move to the next segment,  
00A5 393 : if any. Compute LBN of the new directory segment. Equation is  
00A5 394 :  
00A5 395 :      6 + << segment_number - 1 > * 2 >  
00A5 396 :  
00A5 397 : Then loop back to read it into memory.  
00A5 398 :  
00A5 399 :  
55 02 A1 3C 00A5 400 END_OF_SEGMENT:                ; Look for another segment.  
12 13 00A5 401      MOVZWL RTF_W_NEXTSEG(R1),R5          ; Get link to next segment.  
55 55 D7 00A9 402      BEQL   NO_SUCH_FILE              ; If null, return with error.  
01 78 00AB 403      DECL   R5                          ; Decrement segment number.  
55 55 06 C0 00AD 404      ASHL   #1,R5,R5              ; Convert to segment modulo.  
FF75 31 00B1 405      ADDL   #6,R5                      ; Add base LBN of segments.  
00B4 406      BRW     READ_SEGMENT                    ; Read new segment into memory.  
00B7 407 :  
00B7 408 :  
00B7 409 : The directory was inconsistent. Return with an error code.  
00B7 410 :  
00B7 411 :  
50 0828 8F 3C 00B7 412 BAD_DIRECTORY:                ; Load error status code.  
00B7 413      MOVZWL #SS$_BADIRECTORY,R0              ; Directory is bad.  
04 00BC 414      RET
```


RTFILREAD
V04-000

D 14
- Lookup facility for RT-11 Files Struct 16-SEP-1984 00:00:43 VAX/VMS Macro V04-00
RTF\$OPENFILE - Looks up a file on an RT- 4-SEP-1984 23:05:36 [BOOTS.SRC]RTFILREAD.MAR;1

Page 10
(3)

```
00BD 415
00BD 416 :
00BD 417 : The file does not exist on this volume.
00BD 418 :
00BD 419 :
00BD 420 NO_SUCH_FILE: ; Load error status code.
50 0910 8F 3C 00BD 421 MOVZWL #SS$_NOSUCHFILE,R0 ; The file is not on the volume.
04 00C2 422 RET
00C3 423
00C3 424 .END
```


RTFILREAD
Symbol table

E 14

- Lookup facility for RT-11 Files Struct 16-SEP-1984 00:00:43 VAX/VMS Macro V04-00 Page 11
4-SEP-1984 23:05:36 [BOOTS.SRC]RTFILREAD.MAR;1 (3)

BAD_DIRECTORY	000000B7	R	02
BAD_FILENAME	0000001E	R	02
DIRSEG_BLKs	= 00000008		
END_OF_SEGMENT	000000A5	R	02
FIL\$CVTFILNAM	*****	X	02
FIL\$READ_LBN	*****	X	02
FILE_DESC	= 00000004		
GET_2ND_BLK	0000003C	R	02
LOOK_AT_ENTRY	00000065	R	02
MOVE_FILE_TYPE	00000024	R	02
NEXT_ENTRY	00000099	R	02
NO_SUCH_FILE	000000BD	R	02
PACK_FILNAM	0000C014	R	02
READ_SEGMENT	0000002C	R	02
RTF\$OPENFILE	00000000	RG	02
RTF_B_CHAN_NUM	0000000A		
RTF_B_EVENSTAT	00000000		
RTF_B_JOBNUM	0000000B		
RTF_B_ODDSTAT	00000001		
RTF_C_EMPTYFILE	= 00000002		
RTF_C_ENDOFSEG	= 00000008		
RTF_C_PERMAFILE	= 00000004		
RTF_C_TENTAFILE	= 00000001		
RTF_K_ENTLENGTH	0000000E		
RTF_L_ENTRYONE	0000000A		
RTF_L_FILENAME	00000002		
RTF_W_DATE	0000000C		
RTF_W_EXTRBYTES	00000006		
RTF_W_FILLENGTH	00000008		
RTF_W_FILTYPE	00000006		
RTF_W_HIGHSEG	00000004		
RTF_W_NEXTSEG	00000002		
RTF_W_NUMSEGS	00000000		
RTF_W_STARTLBN	00000008		
RTF_W_STATUS	00000000		
SS\$BADFILENAME	= 00000818		
SS\$BADIRECTORY	= 00000828		
SS\$NOSUCHFILE	= 00000910		
START_DIRSEG	00000052	R	02
STAT_BLOCK	= 0000000C		

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes														
ABS	00000000 (0.)	00 (0.)	NOPIC	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE				
\$ABSS	0000000E (14.)	01 (1.)	NOPIC	USR	CON	ABS	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE				
YFILEREAD	000000C3 (195.)	02 (2.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	LONG				

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
-----	-----	-----	-----
Initialization	29	00:00:00.09	00:00:01.21
Command processing	113	00:00:00.66	00:00:05.97
Pass 1	290	00:00:08.24	00:00:16.01
Symbol table sort	0	00:00:01.46	00:00:03.09
Pass 2	94	00:00:01.71	00:00:04.03
Symbol table output	6	00:00:00.07	00:00:00.07
Psect synopsis output	1	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	535	00:00:12.26	00:00:30.40

The working set limit was 1350 pages.
46954 bytes (92 pages) of virtual memory were used to buffer the intermediate code.
There were 50 pages of symbol table space allocated to hold 942 non-local and 0 local symbols.
424 source lines were read in Pass 1, producing 16 object records in Pass 2.
11 pages of virtual memory were used to define 10 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
-----	-----
_\$255\$DUA28:[BOOTS.OBJ]BOOTS.MLB;1	0
-\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	0
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	6

975 GETS were required to define 6 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:RTFILREAD/OBJ=OBJ\$:RTFILREAD MSRC\$:RTFILREAD/UPDATE=(ENH\$:RTFILREAD)+EXECML\$/LIB+LIB\$:BOOTS.MLB/LIB

0039 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY