

RRRRRRRRRRRR		MMM		MMM	SSSSSSSSSSSS	
RRRRRRRRRRRR		MMM		MMM	SSSSSSSSSSSS	
RRRRRRRRRRRR		MMM		MMM	SSSSSSSSSSSS	
RRR	RRR	MMMMMM	MMMMMM	SSS		
RRR	RRR	MMMMMM	MMMMMM	SSS		
RRR	RRR	MMMMMM	MMMMMM	SSS		
RRR	RRR	MMM	MMM	MMM	SSS	
RRR	RRR	MMM	MMM	MMM	SSS	
RRR	RRR	MMM	MMM	MMM	SSS	
RRRRRRRRRRRR		MMM		MMM	SSSSSSSSSS	
RRRRRRRRRRRR		MMM		MMM	SSSSSSSSSS	
RRRRRRRRRRRR		MMM		MMM	SSSSSSSSSS	
RRR	RRR	MMM		MMM		SSS
RRR	RRR	MMM		MMM		SSS
RRR	RRR	MMM		MMM		SSS
RRR	RRR	MMM		MMM		SSS
RRR	RRR	MMM		MMM		SSS
RRR	RRR	MMM		MMM		SSS
RRR	RRR	MMM		MMM		SSS
RRR	RRR	MMM		MMM	SSSSSSSSSSSS	
RRR	RRR	MMM		MMM	SSSSSSSSSSSS	
RRR	RRR	MMM		MMM	SSSSSSSSSSSS	

SY

NT

NT

NT

NT

NT

NT

NT

NT

NT

NT

NT

NT

NT

NT

NT

NT

NT

NT

NT

NT

NT

NT

NT

NT

PI

```
RRRRRRRR      MM      MM      SSSSSSSS      000000      000000      PPPPPPPP      EEEEEEEEEEE      NN      NN
RRRRRRRR      MM      MM      SSSSSSSS      000000      000000      PPPPPPPP      EEEEEEEEEEE      NN      NN
RR      RR      MMMM      MMMM      SS      00      00      00      00      PP      PP      EE      NN      NN
RR      RR      MMMM      MMMM      SS      00      00      00      00      PP      PP      EE      NN      NN
RR      RR      MM      MM      SS      00      0000      00      00      PP      PP      EE      NNNN      NN
RR      RR      MM      MM      SS      00      0000      00      00      PP      PP      EE      NNNN      NN
RRRRRRRR      MM      MM      SSSSSS      00      00      00      00      PPPPPPPP      EEEEEEEEEEE      NN      NN
RRRRRRRR      MM      MM      SSSSSS      00      00      00      00      PPPPPPPP      EEEEEEEEEEE      NN      NN
RR      RR      MM      MM      SS      0000      00      00      00      PP      EE      NN      NNNN
RR      RR      MM      MM      SS      0000      00      00      00      PP      EE      NN      NNNN
RR      RR      MM      MM      SS      00      00      00      00      PP      EE      NN      NNNN
RR      RR      MM      MM      SS      00      00      00      00      PP      EE      NN      NNNN
RR      RR      MM      MM      SSSSSSSS      000000      000000      PP      EE      NN      NN
RR      RR      MM      MM      SSSSSSSS      000000      000000      PP      EE      NN      NN
                                         ....
                                         ....
                                         ....
                                         ....
```

```
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
LLLLLLLLLLLL      IIIIII      SSSSSSSS
LLLLLLLLLLLL      IIIIII      SSSSSSSS
```


(3) 221
(4) 270
(14) 739
(15) 877
(16) 978

DECLARATIONS
RMS\$OPEN - \$OPEN ROUTINE
RMS\$CHK_SLIST - Process the searchlist loop
RMS\$KNOWNFILE - Kernel Mode Known FILE Support
WRITE_AT_JNL - Local Subroutine

```
0000 1          $BEGIN RMSOOPEN,000,RMSRMS,<DISPATCH FOR OPEN OPERATION>
0000 2
0000 3
0000 4 :*****
0000 5 :*
0000 6 :*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 7 :*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 8 :*  ALL RIGHTS RESERVED.
0000 9 :*
0000 10 :*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 11 :*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 12 :*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 13 :*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 14 :*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 15 :*  TRANSFERRED.
0000 16 :*
0000 17 :*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 18 :*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 19 :*  CORPORATION.
0000 20 :*
0000 21 :*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 22 :*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 23 :*
0000 24 :*
0000 25 :*****
0000 26 :
```



```
0000 28 :++
0000 29 : Facility: RMS32
0000 30 :
0000 31 : Abstract:
0000 32 : This module is the highest level control routine
0000 33 : to perform the $open function.
0000 34 :
0000 35 : Environment:
0000 36 : Star processor running Starlet exec.
0000 37 :
0000 38 : Author: L. F. Laverdure Creation Date: 3-JAN-1977
0000 39 :
0000 40 : Modified By:
0000 41 :
0000 42 : V03-033 RAS0320 Ron Schaefer 6-Jul-1984
0000 43 : Narrow the list of continuable search list errors.
0000 44 :
0000 45 : V03-032 RAS0309 Ron Schaefer 14-Jun-1984
0000 46 : Check the protection of installed open images exactly
0000 47 : as the ACP would. Also return execute-only access
0000 48 : permission information as well.
0000 49 :
0000 50 : V03-031 RAS0301 Ron Schaefer 30-Apr-1984
0000 51 : Fix bug in RAS0286 that caused an infinite loop
0000 52 : if doing a knownfile lookup over the net or for an
0000 53 : explicit version number.
0000 54 :
0000 55 : V03-030 RAS0286 Ron Schaefer 3-Apr-1984
0000 56 : Change the knownfile lookup logic with respect to
0000 57 : searchlists so that if a knownfile is requested,
0000 58 : examine each searchlist element as a known file;
0000 59 : then restart at the beginning for ordinary file opens.
0000 60 : Revise the searchlist continuation rules/errors.
0000 61 :
0000 62 : V03-029 RAS0285 Ron Schaefer 2-Apr-1984
0000 63 : Complete RAS0270 by copying/preserving the DID in the FWA's
0000 64 : FIB buffer. For normal $OPEN/$CREATE, the DID is already
0000 65 : in the FIB buffer; however, for open by NAM block and FID, the
0000 66 : DID in the FIB must be 0 at the access. If that happens,
0000 67 : we copy the DID from the NAM block over into the FIB after
0000 68 : we filled in the remaining NAM block fields.
0000 69 : Note that this will NOT work for people who fondle their NAM
0000 70 : blocks between the $SEARCH and the $OPEN, but they didn't
0000 71 : work before when we depended on the NAM block.
0000 72 : This also doesn't work for known files since the DID is not
0000 73 : saved. However, deleting, submitting or spooling known files
0000 74 : doesn't seem to be too important.
0000 75 :
0000 76 : V03-028 DAS0002 David Solomon 25-Mar-1984
0000 77 : Fix incorrect reference to XABOPN_ARGS (use RMS symbol).
0000 78 :
0000 79 : V03-027 DAS0001 David Solomon 25-Mar-1984
0000 80 : Fix broken CASE.
0000 81 :
0000 82 : V03-026 RAS0268 Ron Schaefer 12-Mar-1984
0000 83 : Move searchlist loop decision from RMS$PRFLNM to here
0000 84 : so as to get consistent processing.
```

0000	85	:			
0000	86	:	V03-025	RAS0243	Ron Schaefer 23-Jan-1984
0000	87	:		Add \$PRDEF macro and add RMS prefix to global symbol.	
0000	88	:			
0000	89	:	V03-024	RAS0218	Ron Schaefer 5-Dec-1983
0000	90	:		Make node names work as search list elements.	
0000	91	:		Preserve NAM block context properly across search list elements.	
0000	92	:			
0000	93	:	V03-023	RAS0209	Ron Schaefer 4-Nov-1983
0000	94	:		Clean-up returned device characteristics by calling	
0000	95	:		a central routine RMSRET_DEV_CHAR.	
0000	96	:			
0000	97	:	V03-022	RAS0198	Ron Schaefer 6-Oct-1983
0000	98	:		Merge \$OPEN and \$CREATE searchlist looping logic into a	
0000	99	:		single routine RMSCHECK_SRCHLIST. Only loop for	
0000	100	:		certain types of error; not for all.	
0000	101	:			
0000	102	:	V03-021	RAS0196	Ron Schaefer 3-Oct-1983
0000	103	:		Fix RAS0195 to preserve the status value of the previous	
0000	104	:		searchlist element when deassigning the channel.	
0000	105	:		Also correct wrong data type usage in RAS0189.	
0000	106	:			
0000	107	:	V03-020	RAS0195	Ron Schaefer 27-Sep-1983
0000	108	:		Return unsuccessful channels assigned when looping	
0000	109	:		thru the searchlist.	
0000	110	:			
0000	111	:	V03-019	RAS0189	Ron Schaefer 9-Sep-1983
0000	112	:		Complete known file support by implementing a	
0000	113	:		kernel mode routine to properly update system	
0000	114	:		refcnts et. al.	
0000	115	:			
0000	116	:	V03-018	KPL0005	Peter Lieberwirth 15-Aug-1983
0000	117	:		Don't clear pointer to JNLBDB before RELEASEALL because	
0000	118	:		RELEASEALL no longer releases journal BDBs.	
0000	119	:			
0000	120	:	V03-017	LJA0087	Laurie J. Anderson 5-Aug-1983
0000	121	:		Clear the create bit at the end of the create. Do	
0000	122	:		not want that bit set for the duration of the file being	
0000	123	:		open.	
0000	124	:			
0000	125	:	V03-016	RPG0016	Bob Grosso 29-Jul-1983
0000	126	:		Skip known file open if there was an explicit version.	
0000	127	:		Use FID to open file if installed but not open.	
0000	128	:			
0000	129	:	V03-015	KPL0004	Peter Lieberwirth 25-Jul-1983
0000	130	:		Fix bug in AT journaling introduced in v03-011.	
0000	131	:			
0000	132	:	V03-014	LJK0224	Lawrence J. Kenah 6-Jul-1983
0000	133	:		Store WCB address in CCB for files installed /OPEN	
0000	134	:			
0000	135	:	V03-013	KBT0552	Keith B. Thompson 5-Jul-1983
0000	136	:		More kf work	
0000	137	:			
0000	138	:	V03-012	KBT0547	Keith B. Thompson 22-Jun-1983
0000	139	:		Use new kf routine	
0000	140	:			
0000	141	:	V03-011	KPL0003	Peter Lieberwirth 20-Jun-1983

0000 142 : Add local routine WRITE_AT_JNL to flush file-related
0000 143 : AT journal record to the journal. This is not done at
0000 144 : ASSJNL time because the AT journal record accumulates
0000 145 : info during the operation, and is only complete at the
0000 146 : end.
0000 147 :
0000 148 : V03-010 KBT0531 Keith B. Thompson 17-Jun-1983
0000 149 : Implement search list
0000 150 :
0000 151 : V03-009 KPL0002 Peter Lieberwirth 30-May-1983
0000 152 : Clear pointer to journal BDB when it is deallocated.
0000 153 :
0000 154 : V03-008 RAS0155 Ron Schaefer 3-May-1983
0000 155 : Make indirect PPF I/O look like a terminal regardless
0000 156 : of the SQO bit.
0000 157 :
0000 158 : V03-007 ROW0170 Ralph O. Weber 3-MAR-1983
0000 159 : Correct broken branch destination to RM\$CLOSE3 in error
0000 160 : processing.
0000 161 :
0000 162 : V03-006 LJA0061 Laurie J. Anderson 23-Feb-1983
0000 163 : Move RM\$FILLNAM to RMONAMSTR which is where it belongs.
0000 164 : It fills in the NAM block with the resultant name string plus...
0000 165 :
0000 166 : V03-005 KBT0421 Keith B. Thompson 30-Nov-1982
0000 167 : Change ifb\$w_devbufsiz to ifb\$l_devbufsiz
0000 168 :
0000 169 : V03-004 LJA0020 Laurie Anderson 02-Sep-1982
0000 170 : Define an alternate entry point for top level open.
0000 171 : If restart from create, just try to open the file.
0000 172 :
0000 173 : V03-003 KBT0187 Keith B. Thompson 23-Aug-1982
0000 174 : Reorganize psepts and rename entry point to single '\$'
0000 175 :
0000 176 : V03-002 RAS0084 Ron Schaefer 2-Apr-1982
0000 177 : Return RAT=CR for stream format files even if
0000 178 : the file attribute is none.
0000 179 :
0000 180 : V03-001 KPL0001 Peter Lieberwirth 22-Mar-1982
0000 181 : Fix bugcheck on shared file ISAM UFO open by making
0000 182 : sure IFAB address is in R10.
0000 183 :
0000 184 : V02-052 CDS0002 C Saether 5-Feb-1982
0000 185 : Return GBC field to FAB from ifab.
0000 186 :
0000 187 : V02-051 TMK0001 Todd M. Katz 17-Jan-1982
0000 188 : If the device is mounted foreign, arrange it so that
0000 189 : when the resultant name string is written out it
0000 190 : contains a null length directory (ie []).
0000 191 :
0000 192 : V02-050 RAS0051 Ron Schaefer 22-Dec-1981
0000 193 : Properly probe the NAM block whenever its address is
0000 194 : fetched from the user's FAB.
0000 195 :
0000 196 : V02-049 JWH0001 Jeffrey W. Horn 16-Dec-1981
0000 197 : Clear FWASV_DIR bit if either not directory device, or
0000 198 : SDI device.


```
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 :  
  
V02-048 TMH0048 Tim Halvorsen 05-Sep-1981  
Translate FWASQ_DEVICE string once before storing it  
into DVI, so that DVI always gets the actual device name.  
  
V02-047 PSK0001 P. S. Knibbe 31-Aug-1981  
Change default length for key xab's to be  
keylen_v2. Anything longer will be legal.  
  
V02-046 KEK0008 K. E. Kinnear 12-Aug-1981  
On foreign devices, return both DEV and SDC characteristics  
bits as we got them from the exec, not SDC with DIR clear.  
  
V02-045 MCN0005 Maria del C. Nasr 10-Feb-1981  
Make sure magtape is mounted before opening file.  
  
V02-044 SPR35461 Maria del C. Nasr 02-Feb-1981  
Change return from CIF processing to check for any errors
```



```
0000 221      .SBTTL  DECLARATIONS
0000 222
0000 223      :
0000 224      : Include Files:
0000 225      :
0000 226      :
0000 227      :
0000 228      : Macros:
0000 229      :
0000 230      $ARMDEF
0000 231      $CCBDEF
0000 232      $CHPCTLDEF
0000 233      $DEVDEF
0000 234      $FABDEF
0000 235      $FCBDEF
0000 236      $FIBDEF
0000 237      $FWADEF
0000 238      $IFBDEF
0000 239      $IPLDEF
0000 240      $KFEDEF
0000 241      $NAMDEF
0000 242      $PCBDEF
0000 243      $PRDEF
0000 244      $PSLDEF
0000 245      $RJBDEF
0000 246      $RJRDEF
0000 247      $RMSDEF
0000 248      $UCBDEF
0000 249      $WCBDEF
0000 250      $XABALLDEF
0000 251      $XABKEYDEF
0000 252      $XABSUMDEF
0000 253
0000 254      :
0000 255      : Equated Symbols:
0000 256      :
0000 257
00000020 0000 258      FOP=FAB$L_FOP*8          ; bit offset to fop
0000 259
0000 260      :
0000 261      : Own Storage:
0000 262      :
0000 263
0000 264      RMSXABOPN ARG$::
00'0C 16 0000 265      .BYTE  XAB$C_SUM,XAB$C_SUMLEN,XBC$C_OPNSUM3
00'40 15 0003 266      .BYTE  XAB$C_KEY,XAB$C_KEYLEN_V2,XBC$C_OPNKEY3
00'20 14 0006 267      .BYTE  XAB$C_ALL,XAB$C_ALLLEN,XBC$C_OPNALL3
00      00 0009 268      .BYTE  0
```



```
000A 270          .SBTTL  RMS$OPEN - $OPEN ROUTINE
000A 271
000A 272 :++
000A 273
000A 274 : RMS$OPEN -- Open routine.
000A 275
000A 276 : This routine performs the highest level $open processing.
000A 277 : its functions include:
000A 278
000A 279 :     1. Common setup.
000A 280 :     2. Dispatch to organization-dependent code.
000A 281 :     3. Dispatch to the display routine.
000A 282
000A 283
000A 284 : Calling Sequence:
000A 285
000A 286 :     Entered from exec as a result of user's calling sys$open
000A 287 :     (e.g., by using the $open macro).
000A 288
000A 289 : Input Parameters:
000A 290
000A 291 :     AP      user's argument list addr
000A 292
000A 293 : Implicit Inputs:
000A 294
000A 295 :     The contents of the fab and possible related user interface
000A 296 :     blocks.
000A 297
000A 298 : Output Parameters:
000A 299
000A 300 :     R0      status code
000A 301 :     R1      destroyed
000A 302
000A 303 : Implicit Outputs:
000A 304
000A 305 :     The various fields of the fab are filled in to reflect
000A 306 :     the status of the open file. (see rms functional spec for
000A 307 :     a complete list.)
000A 308 :     An ifab is initialized to reflect the open file.
000A 309
000A 310 :     A completion ast is queued if so specified by the user.
000A 311
000A 312 : Completion Codes:
000A 313
000A 314 :     Standard rms (see functional spec for list).
000A 315
000A 316 : Side Effects:
000A 317
000A 318 :     none
000A 319
000A 320 :--
000A 321
000A 322 : $ENTRY  RMS$OPEN
000A 323 : $TSTPT  OPEN
000A 324 : $BSBW   RMS$SETI
000A 325
000A 326
FFED' 30 0010 324          ; do common setup
0013 325          ; note: does not return on error
0013 326
```



```
0013 327 :  
0013 328 : Alternate entry point for open. Called from create when the create call  
0013 329 : was a restart operation.  
0013 330 :  
0013 331 :  
0013 332 RMS$OPEN_ALT::  
0013 333 :  
0013 334 :  
0013 335 : An ifab has been set up  
0013 336 :  
0013 337 :  
05 22 A9 06 E1 0013 338 CLRL R10 ; no FWA to start with  
FFEB' 30 0015 339 BSBW RMS$PRFLNM ; process file name  
0018 340 :  
0018 341 BBC #FAB$V_BRO,IFB$B_FAC(R9),10$ ; branch if bro not set  
001D 342 CSB #FAB$V_BIO,IFB$B_FAC(R9) ; clear bio (implied  
0022 343 : by bro without restrictions)  
47 68 7D 50 E9 0022 344 10$: BLBC R0,50$ ; exit on error from RMS$PRFLNM  
57 68 3E E1 0025 345 20$: BBC #FAB$V_KFO+FOP,(R8),22$ ; branch if kfo not set  
28 A8 D0 0029 346 MOVL FAB$L_NAM(R8),R7 ; get name block  
FFD0' 30 002D 347 BSBW RMS$CHRNAM ; can we use it?  
3D 50 E9 0030 348 BLBC R0,22$ ; nope  
3B 6A 19 E0 0033 349 BBS #FWA$V_NODE,(R10),25$ ; branch on network operation  
37 6A 10 E0 0037 350 BBS #FWA$V_EXP_VER,(R10),25$ ; explicit version, skip KF lookup  
003B 351 :  
003B 352 :  
003B 353 : INSS$KF_SCAN returns R0:  
003B 354 :  
003B 355 SS$_NORMAL - known file found but not open  
003B 356 :  
003B 357 RMSS$_KFF - known file found and it was open  
003B 358 :  
003B 359 RMSS$_FNF - known file not found  
003B 360 :  
003B 361 :  
18 A8 DF 003B 362 PUSHAL FAB$L_CTX(R8) ; return KFE in fab  
57 DD 003E 363 PUSHL R7 ; filled in name block  
00000000'EF 02 FB 0040 364 CALLS #2,INSS$KF_SCAN ; go try known file table  
2D 50 E9 0047 365 BLBC R0,30$ ; not installed  
004A 366 :  
50 00018031 8F D1 004A 367 CMPL #RMSS$_KFF,R0 ; was the file installed open  
3A 12 0051 368 BNEQ 40$ ; installed, but not open  
0053 369 :  
50 DD 0053 370 PUSHL R0 ; preserve status  
0055 371 $CMKRNL S - ; kernel mode routine to  
0055 372 -ROUTIN=RM$KNOWNFILE ; modify system refcnts et.al.  
51 8E D0 0064 373 MOVL (SP)+,R1 ; recover status  
OD 50 E9 0067 374 BLBC R0,30$ ; can't access this file  
50 51 D0 006A 375 MOVL R1,R0 ; set appropriate success code  
01D3 31 006D 376 BRW RM$CREATEEXIT ; exit from open immediately  
0070 377 : if found/open or error  
0070 378 :  
3E 11 0070 379 22$: BRB 100$ ; helper branch  
0072 380 :  
0072 381 :  
0072 382 : Couldn't find this file spec as a known file,  
0072 383 : try to get another if searchlist is present
```



```

0072 384 ;
0072 385 25$: RMSERR FNF ; setup appropriate error in R0
35 6A 38 E1 0077 386 30$: BBC #FWA$V_SLPRESENT,(R10),100$ ; if no searchlist do normal open
01ED 30 007B 387 BSBW RM$CHK_SLIST ; try again
A4 50 E8 007E 388 BLBS R0,20$ ; did it work?
1B 51 E9 0081 389 BLBC R1,60$ ; should we try again?
FF79' 30 0084 390 BSBW RM$DEALLOCATE FWA ; release exhausted searchlist
0087 391 CSB #FAB$V_KFO+FOP,(R8) ; Make sure not KFO again
86 11 008B 392 BRB RM$OPEN_ALT ; try for non-KFO open
008D 393 ;
008D 394 ;
008D 395 ; Try to open the knownfile normally; if this fails, then go thru the known
008D 396 ; file searchlist lookup logic
008D 397 ;
008D 398 ;
008D 399 40$: SSB #FAB$V_NAM+FOP,(R8) ; Force NAM block open with FID
FF6C' 30 0091 400 BSBW RM$SETDID ; process the directory id
EO 50 E9 0094 401 BLBC R0,30$ ; check search list on error
FF66' 30 0097 402 BSBW RM$ACCESS ; access the file
DA 50 E9 009A 403 BLBC R0,30$ ; check search list on error
1D 11 009D 404 BRB RM$OPEN_CIF ; continue with open
009F 405 ;
0093 31 009F 406 60$: BRW ERROR ; no, return error
00A2 407 ;
00A2 408 ;
00A2 409 ; There was a problem with the file spec, try to get another if search list
00A2 410 ; are present
00A2 411 ;
00A2 412 ;
00A2 413 50$: TSTL R10 ; have a FWA?
F5 6A 5A D5 00A2 414 BEQL 60$ ; if not don't check it
F9 13 00A4 415 BBC #FWA$V_SLPRESENT,(R10),60$ ; if no search list exit
01BE 30 00AA 416 BSBW RM$CHK_SLIST ; try again
EF 50 E9 00AD 417 BLBC R0,60$ ; did it work?
00B0 418 ;
FF4D' 30 00B0 419 100$: BSBW RM$SETDID ; process the directory id
EC 50 E9 00B3 420 BLBC R0,50$ ; check search list on error
FF47' 30 00B6 421 BSBW RM$ACCESS ; access the file
E6 50 E9 00B9 422 BLBC R0,50$ ; check search list on error
00BC 423 ;
00BC 424 ;
00BC 425 ; Return point for create turned into open via 'cif' bit.
00BC 426 ;
00BC 427 ;
00BC 428 RM$OPEN_CIF::
76 50 E9 00BC 429 BLBC R0,ERROR ; exit on error
FF3E' 30 00BF 430 BSBW RM$FILLNAM ; fill in nam block
70 50 E9 00C2 431 BLBC R0,ERROR ; exit on error
00C5 432 ;
00C5 433 ;
00C5 434 ; Copy the DID from the NAM block into the FIB if this is an OPEN by FID
00C5 435 ;
00C5 436 ;
19 68 38 E1 00C5 437 BBC #FAB$V_NAM+FOP,(R8),5$ ; skip if not open with FID
57 D5 00C9 438 TSTL R7 ; have a NAM block (from RM$FILLNAM)
15 13 00CB 439 BEQL 5$ ; ce la vie'
51 14 AA D0 00CD 440 MOVL FWA$Q_FIB+4(R10),R1 ; get addr of FIB
```



```

      OF 13 00D1 441      BEQL 5$      ; no FIB, no DID
0A A1 0A B5 00D3 442      TSTW FIB$W_DID(R1) ; have a DID?
      0A 12 00D6 443      BNEQ 5$      ; continue if so
OE A1 2A A7 D0 00D8 444      MOVL NAM$W_DID(R7),FIB$W_DID(R1) ; copy DID
      2E A7 B0 00DD 445      MOVW NAM$W_DID+4(R7),FIB$W_DID+4(R1) ; copy DID last word
      00E2 446
      00E2 447
      00E2 448 ; Make sure eof info is in 'eof blk + 1, 0 offset' form.
      00E2 449
      00E2 450
      5C A9 B1 00E2 451 5$: CMPW IFB$W_FFB(R9),- ; is last block full?
      48 A9 00E5 452      IFB$L_DEVBUSIZ(R9)
      06 1F 00E7 453      BLSSU 10$      ; branch if not
      74 A9 D6 00E9 454      INCL IFB$L_EBK(R9) ; bump eof block
      5C A9 B4 00EC 455      CLRW IFB$W_FFB(R9) ; and zero offset
73 69 3E E0 00EF 456 10$: BBS #IFB$V_DAP,(R9),DAPRTN ; branch if network operation
      00F3 457
      00F3 458 ;
      00F3 459 ; Dispatch to organization-dependent open code.
      00F3 460
      00F3 461
      69 05 E1 00F3 462      BBC #DEV$V_SQD,IFB$L_PRIM_DEV(R9),- ; branch if not magtape
      08 00F6 463      20$
      69 13 E1 00F7 464      BBC #DEV$V_MNT,IFB$L_PRIM_DEV(R9),- ; error, if magtape not mounted
      2E 00FA 465      ERDNR
      69 15 E0 00FB 466      BBS #DEV$V_DMT,IFB$L_PRIM_DEV(R9),- ; error, if magtape marked for dismo
      2A 00FE 467      ERDNR
      00FF 468 20$: CASE TYPE=B,- ; pick up correct routine
      00FF 469      SRC=IFB$B_ORGCASE(R9),-
      00FF 470      DISPLIST=<RMS$OPEN1,RM_OPEN2_BR,RMS$OPEN3>
```

			010A	472				
			010A	473	:	++		
			010A	474	:	.		
			010A	475	:	Error returns		
			010A	476	:	.		
			010A	477	:	--		
			010A	478	:			
			010A	479	:			
			010A	480	:	Unknown file organization.	- verify bio (or bro) accessed.	
			010A	481	:			
			010A	482	:			
			010A	483		RMSERR	ORG	; org not supported
60	8F	93	010F	484		BITB	#FAB\$M_BIO!FAB\$M_BRO,-	; either bio or bro set?
22	A9		0112	485			IFBSB_FAC(R9)	
	1F	13	0114	486		BEQL	ERROR	; branch if not (error)
			0116	487		RMSSUC		
0038		31	0119	488		BRW	RM\$COPRTN	; all finished open
			011C	489				
			011C	490		RM_OPEN2 BR:		
00000000'	Ef	17	011C	491		JMP	RM\$OPEN2	; branch aid
			0122	492				
			0122	493		ERRRFM: RMSERR	RFM	; bad rfm field
	0c	11	0127	494		BRB	ERROR	
			0129	495				
			0129	496		ERRDNR: RMSERR	DNR	; device not mounted
	05	11	012E	497		BRB	ERROR	
			0130	498				
			0130	499		ERRIRC: RMSERR	IRC	; illegal fixed record size
			0135	500				
			0135	501		ERROR: CSB	#IFBSV_ACCESSED,(R9)	; don't write file attributes
23 A9	02	91	0139	502		CMPB	#IFBS_C_IDX,IFBSB_ORGCASE(R9)	; indexed file?
	12	12	013D	503		BNEQ	\$S	; branch if not...clean up
Oe 69	3E	E0	013F	504		BBS	#IFBSV_DAP,(R9),\$S	; branch if network operation
	50	DD	0143	505		PUSHL	R0	; push error code on stack
5A	59	D0	0145	506		MOVL	R9,R10	; RMSCLOSE3 expects IFAB in R10
00000000'	Gf	16	0148	507		JSB	G^RMSCLOSE3	; close indexed file
	50	8ED0	014E	508		POPL	R0	; pop error code from stack
FEAc'		31	0151	509		\$\$: BRW	RM\$CLSCU	; clean up and return


```

0154 511
0154 512
0154 513 : Return here from org-dependent routines.
0154 514
0154 515
0154 516 RM$COPRTN::
OB 69 DE 50 E9 0154 517 BLBC R0,ERROR
FE 3E E0 0157 518 BBS #IFBSV_DAP,(R9),DAPRTN ; branch if network operation
0158 519
0158 520
0158 521 : Now handle summary, allocation, and key xab's.
0158 522
0158 523
0158 524 MOVAB RM$XABOPN,ARGS,AP ; move addr of xab table to ap
SC FEA1 CF 9E 0158 525 BSBW RM$XAB_SCAN ; scan the xab chain
FE9D' 30 0160 526 BLBC R0,ERROR ; get out on error
CF 50 E9 0163 527
0166 528
0166 529 : Override run-time deq with user value, if any.
0166 530
0166 531
0166 532 DAPRTN: MOVW FAB$W_DEQ(R8),IFB$W_RTDEQ(R9)
4C A9 14 A8 B0 0166 533 BNEQ 5$ ; branch if speced
OA 12 0168 534 MOVW IFB$W_DEQ(R9),IFB$W_RTDEQ(R9) ; otherwise pick up deq
4C A9 62 A9 B0 016D 535
0172 536
0172 537 : From file header.
0172 538
0172 539
0172 540 MOVW IFB$W_DEQ(R9),FAB$W_DEQ(R8) ; and put in fab
14 A8 62 A9 B0 0177 541
0177 542
0177 543 : Return bdb and i/o buffer to free space and page lists.
0177 544
0177 545
0177 546 5$: BBC #IFBSV_AT,IFB$B_JNLFLG(R9),7$ ; skip if not AT journaling
03 00A0 C9 04 E1 0177 547 BSBW WRITE_AT_JNL ; write AT record - eat any error
0231 30 017D 548 7$: BSBW RMSRECEASALL ; return bdb and buffer
FE7D' 30 0180 549
0183 550
0183 551 : Validate rfm.
0183 552
0183 553
0183 554 ASSUME IFB$V_RFM EQ 0
0183 555 ASSUME IFB$S_RFM EQ 4
0183 556
50 A9 F0 8F 8A 0183 557 BICB2 #^XF0,IFB$B_RFMORG(R9) ; leave only rfm in rfmorg
0188 558
0188 559
0188 560 : Check for rfm in supported range.
0188 561
0188 562
0188 563 BBS #IFBSV_BIO,IFB$B_FAC(R9),10$ ; don't check if bio
06 22 A9 05 E0 0188 564 CMPB IFB$B_RFMORG(R9),#FAB$C_MAXRFM
06 06 50 A9 91 018D 565 BGTRU ERRRFM
8F 1A 0191 566
0193 567
0193 567

```

```
0193 568 : If fixed length record format, then set mrs from lrl in case this
0193 569 : is an fcs-11 file.
0193 570 :
0193 571 :
01 50 A9 91 0193 572 10$: CMPB IFB$B_RFMORG(R9),#FAB$C_FIX ; fixed len rec?
60 A9 52 A9 12 0197 573 BNEQ 20$ ; branch if not
07 80 0199 574 MOVW IFB$W_LRL(R9),IFB$W_MRS(R9) ; set record length
90 15 019E 575 BLEQ ERRIRC ; branch if invalid
01A0 576 :
01A0 577 : force stream format files to appear to have RAT non-null,
01A0 578 : even if they don't.
01A0 579 :
01A0 580 :
01A0 581 ASSUME FAB$C_STM LT FAB$C_STMLF
01A0 582 ASSUME FAB$C_STM LT FAB$C_STMCR
01A0 583 :
04 50 A9 91 01A0 584 20$: CMPB IFB$B_RFMORG(R9),#FAB$C_STM ; stream format?
0A 1F 01A4 585 BLSSU RM$COPRTN1 ; nope
07 93 01A6 586 BITB #<FAB$M_CR!FAB$M_FTN!FAB$M_PRN>,-
51 A9 04 12 01A8 587 IFB$B_RAT(R9) ; carriage control already set?
02 88 01AA 588 BNEQ RM$COPRTN1 ; ok
51 A9 02 88 01AC 589 BISB2 #FAB$M_CR,IFB$B_RAT(R9) ; force RAT=CR
```



```
01B0 591 :  
01B0 592 : Return point for indirect open of process permanent file.  
01B0 593 :  
01B0 594 :  
01B0 595 : Set the rfm, rat, org, and mrs fields into the fab.  
01B0 596 :  
01B0 597 :  
1F A8 50 A9 90 01B0 598 RM$COPRTN1::  
1E A8 51 A9 90 01B0 599      MOVB      IFB$B_RFMORG(R9),FAB$B_RFM(R8) ; set rfm  
01B5 600      MOVB      IFB$B_RAT(R9),FAB$B_RAT(R8) ; set rat  
01BA 601 :  
01BA 602 :  
01BA 603 : Return point for indirect open of process permanent file and rfm and  
01BA 604 : rat already set.  
01BA 605 :  
01BA 606 :  
01BA 607 RM$COPRTN2::  
01BA 608      INSV      IFB$B_ORGCASE(R9),- ; set org  
01BD 609      #FAB$V_ORG,#FAB$S_ORG,-  
01BF 610      FAB$B_ORG(R8)  
03 69 38 E1 01C1 611      BBC      #IFB$V_SEQFIL,(R9),10$ ; branch if not seq file shr'd  
01C5 612 :  
01C5 613      ASSUME   FAB$C_SEQ      EQ      0  
01C5 614 :  
01C5 615      CLRB      FAB$B_ORG(R8) ; this is really a sequential file.  
01C8 616 : ; Orgcase says relative when shared  
36 A8 60 A9 B0 01C8 617 10$: MOVW      IFB$W_MRS(R9),FAB$W_MRS(R8) ; set mrs  
48 A8 64 A9 B0 01CD 618      MOVW      IFB$W_GBC(R9),FAB$W_GBC(R8) ; set gbc  
01D2 619 :  
01D2 620 :  
01D2 621 : If vfc record format, check for 0 fixed header size and if  
01D2 622 : found make it 2 bytes.  
01D2 623 :  
01D2 624 :  
03 50 A9 91 01D2 625      CMPB      IFB$B_RFMORG(R9),#FAB$C_VFC  
01D6 626      BNEQ      20$ ; omit check if not vfc  
01D8 627      TSTB      IFB$B_FSZ(R9) ; check for default  
01DB 628      BNEQ      30$ ; branch if value specified  
5F A9 02 90 01DD 629      MOVB      #2,IFB$B_FSZ(R9) ; set default value  
01E1 630      BRB      30$ ; continue  
5F A9 03 11 01E1 630 :  
01E3 631 20$: CLRB      IFB$B_FSZ(R9) ; guarantee 0 fsz for non-vfc rfm  
01E6 632 : ; (note: fixes rms-11 bug of fsz=2)  
01E6 633 30$: RMSSUC ; inidcate successful open
```

```
01E9 635
01E9 636 :++
01E9 637 :
01E9 638 : Common exit for $create and $open.
01E9 639 :
01E9 640 :--
01E9 641
01E9 642 CREOPEN_EXIT:
03 50 E8 01E9 643 : BLBS R0,2$ ; branch if no error
FF46 31 01EC 644 1$: BRW ERROR ; otherwise go to exit on error
01EF 645 :
01EF 646 :
01EF 647 : Save the various close option bits in ifab
01EF 648 :
01EF 649 :
01EF 650 2$: CSB #IFBSV_CREATE,(R9) ; clear the 'doing create' bit
1C 69 22 E0 01F3 651 BBS #IFBSV_PPF_IMAGE,(R9),5$ ; don't save options if indirect
01F7 652 :
01F7 653 ASSUME FAB$V_RWC+1 EQ FAB$V_DMO
01F7 654 ASSUME FAB$V_DMO+1 EQ FAB$V_SPL
01F7 655 ASSUME FAB$V_SPL+1 EQ FAB$V_SCF
01F7 656 ASSUME FAB$V_SCF+1 EQ FAB$V_DLT
01F7 657 :
51 68 05 2B EF 01F7 658 EXTZV #FAB$V_RWC+FOP,#5,(R8),R1 ; get option bits
01FC 659 :
01FC 660 ASSUME IFBSV_RWC+1 EQ IFBSV_DMO
01FC 661 :
01FC 662 ASSUME IFBSV_SPL+1 EQ IFBSV_SCF
01FC 663 ASSUME IFBSV_SCF+1 EQ IFBSV_DLT
01FC 664 :
69 05 27 51 F0 01FC 665 INSV R1,#IFBSV_RWC,#5,(R9) ; and save them
0201 666 :
0201 667 :
0201 668 : If this is foreign magtape, rewind the tape if rwo is set.
0201 669 :
0201 670 :
0E 69 18 E1 0201 671 BBC #DEV$V_FOR,IFBSL_PRIM_DEV(R9),5$; branch if not foreign
0A 69 05 E1 0205 672 BBC #DEV$V_SQD,IFBSL_PRIM_DEV(R9),5$; or if not magtape
06 68 27 E1 0209 673 BBC #FAB$V_RWO+FOP,(R8),5$ ; or if rwo not speced
FDF0' 30 020D 674 BSBW RMSREWIND_MT ; rewind the tape
D9 50 E9 0210 675 BLBC R0,1$ ; branch on error
0213 676 :
0213 677 :
0213 678 : Set 'blk' bit in ifab for magtape.
0213 679 :
0213 680 :
04 69 05 E1 0213 681 5$: BBC #DEV$V_SQD,IFBSL_PRIM_DEV(R9),8$; branch if not magtape
51 A9 08 88 0217 682 BISB2 #FAB$M_BLK,IFBSB_RAT(R9) ; set no spanning bit
021B 683 :
021B 684 :
021B 685 : Set the fsz, bks, stv, alq, dev, and sdc fields into fab.
021B 686 :
021B 687 :
3F A8 5F A9 90 021B 688 8$: MOVB IFBSB_FSZ(R9),FAB$B_FSZ(R8) ; set fsz
3E A8 5E A9 90 0220 689 MOVB IFBSB_BKS(R9),FAB$B_BKS(R8) ; set bks
03 69 38 E1 0225 690 BBC #IFBSV_SEQFIL,(R9),9$ ; branch not seq file shared
3E A8 94 0229 691 CLRB FAB$B_BKS(R8) ; always zero for seq file
```



```
0C AB 20 A9 B0 022C 692 9$: MOVW IFB$W_CHNL(R9),FAB$L_STV(R8) ; set stv to chan #
10 AB 70 A9 D0 0231 693 : MOVL IFB$L_HBK(R9),FAB$L_ALQ(R8) ; set alq
      0236 694 :
      0236 695 :
      0236 696 : Move device characteristics bits into the fab.
      0236 697 :
      0236 698 :
      FDC7' 30 0236 699 BSBW RMS$RET_DEV_CHAR ; set DEV and SDC fields
      0239 700 :
      0239 701 :
      0239 702 : Check for user file open option.
      0239 703 :
      0239 704 :
      03 68 31 E0 0239 705 20$: BBS #FAB$V_UFO+FOP,(R8),40$ ; branch if ufo option
      FDC0' 31 023D 706 BRW RM$EXRMS ; return to user
      0240 707 :
      0240 708 :
      0240 709 : Leave file open for user but remove ifab
      0240 710 : (no further rms operations available on this file).
      0240 711 :
      0240 712 :
      FDBD' 31 0240 713 40$: BRW RM$RETIFB
```

```
0243 715
0243 716 :++
0243 717 :
0243 718 : Common create clean up and exit
0243 719 :
0243 720 : Return all bdb's and buffers to free space list, causing unlock if locked.
0243 721 :
0243 722 :--
0243 723 :
50 DD 0243 724 RM$CREATEEXIT::
0243 725 PUSH  R0 ; save status code
0245 726 :
0245 727 :
0245 728 : Entry point with status already pushed on the stack.
0245 729 :
0245 730 :
0245 731 RM$CREATEEXIT1::
03 00A0 C9 04 E1 0245 732 BBC #IFB$V AT,IFB$B_JNLFLG(R9),10$ ; skip if not AT journaling
0163 30 024B 733 BSBW WRITE AT JNL ; write AT record - eat any error
FDAF' 30 024E 734 10$: BSBW RMS$RECEASALL ; release all bdb's
50 8ED0 0251 735 POPL R0 ; restore status
FF92 31 0254 736 BRW CREOPEN_EXIT ; join open finish up code
0257 737
```



```
0257 739 .SUBTITLE RMSCHK_SLIST - Process the searchlist loop
0257 740 :++
0257 741 :
0257 742 : These routines are called when a file access failed and a searchlist
0257 743 : is present. It evaluates whether the error allows the list to
0257 744 : continue and updates the list to the next searchlist element.
0257 745 :
0257 746 : RMSCHK_SLIST - Normal file access failures can continue
0257 747 : RMSCHK_SLIST1 - File access failures that create-if can continue
0257 748 :
0257 749 : Inputs:
0257 750 : R0 - failure status of previous access operation.
0257 751 :
0257 752 : Outputs:
0257 753 : R0 - success/fail
0257 754 : (if searchlist is exhausted, status is previous access status.)
0257 755 : R1 - undefined if R0 = success
0257 756 : if R0 = fail, success if processing may continue, failure otherwise
0257 757 :
0257 758 : Implicit inputs:
0257 759 : R11 - impure ptr
0257 760 : R10 - FWA ptr
0257 761 : R9 - IFB ptr
0257 762 : R8 - FAB ptr
0257 763 :
0257 764 : Implicit outputs:
0257 765 : FWA and IFB fields modified.
0257 766 :
0257 767 : Saved stack:
0257 768 : ERR(SP) => R0 error code
0257 769 : STV(SP) => FABSL_STV(R8)
0257 770 : FNB(SP) => NAMSL_FNB
0257 771 : RSL(SP) => NAMSB_RSL
0257 772 : ESL(SP) => NAMSB_ESL
0257 773 :
0257 774 :--
0257 775 :
0257 776 :
0257 777 : Stack offsets for saved context
0257 778 :
0257 779 :
00000000 0257 780 ESL = 0 : NAMSB_ESL
00000001 0257 781 RSL = 1 : NAMSB_RSL
00000004 0257 782 FNB = 4 : NAMSL_FNB
00000008 0257 783 STV = 8 : FABSL_STV
0000000C 0257 784 ERR = 12 : R0
00000010 0257 785 STACK_SIZE = 16 : Size of stack to allocate
0257 786 :
0257 787 :
0257 788 : Errors that searchlist processing should continue from
0257 789 :
0257 790 :
0257 791 RM$SLIST ERRS::
0257 792 RMSERR_WORD DEV ; invalid device name
0259 793 RMSERR_WORD DNF ; directory not found
025B 794 RMSERR_WORD DNR ; device not ready
025D 795 RMSERR_WORD FNF ; file not found
```



```
0000000A 025F 796 RMSERR WORD NMF ; no more files found
0261 797 RM$SLIST_ERR_CNT1 == .-RM$SLIST_ERRS
0261 798 RMSERR_WORD ACC ; ACP file access error
0263 799 RMSERR_WORD FND ; ACP file lookup error
0265 800 RMSERR_WORD PRV ; privilege violation
00000010 0267 801 RM$SLIST_ERR_CNT == .-RM$SLIST_ERRS
0267 802
0267 803 RM$CHK_SLIST1::
05 DD 0267 804 PUSHL S^#<RM$SLIST_ERR_CNT1/2> ; get number of errs to check
02 11 0269 805 BRB CHK_SLIST ; and check
0268 806
08 DD 0268 807 RM$CHK_SLIST::
0268 808 PUSHL S^#<RM$SLIST_ERR_CNT/2> ; get number of errs to check
026D 809
026D 810 CHK_SLIST:
51 6E D0 026D 811 MOVL (SP),R1 ; get count
EO AF41 50 B1 0270 812 10$: CMPW R0,B^RM$SLIST_ERRS-2[R1] ; continue from this err?
OF 12 0275 813 BNEQ 40$ ; nope
14 10 0277 814 BSBB S_LOOP ; try next element
03 50 E8 0279 815 BLBS R0,30$ ; got a good one
EE 51 E9 027C 816 BLBC R1,CHK_SLIST ; get any kind of element?
51 01 D0 027F 817 30$: MOVL #1,R1 ; processing can continue
5E 04 C0 0282 818 ADDL2 #4,SP ; discard count
05 0285 819 RSB ; return
0286 820
E7 51 F5 0286 821 40$: SOBGTR R1,10$ ; try another
5E 04 C0 0289 822 ADDL2 #4,SP ; discard count
05 028C 823 RSB ; return don't continue and
028D 824 ; previous input status
028D 825
028D 826 S_LOOP: SSB #FWASV_SL_PASS,(R10) ; flag search list pass
50 DD 0291 827 PUSHL R0 ; save error status
OC A8 DD 0293 828 PUSHL FAB$_STV(R8) ; save stv secondary code
7E 7C 0296 829 CLRQ -(SP) ; room for NAM block data
57 28 A8 D0 0298 830 MOVL FAB$_NAM(R8),R7 ; get nam address
26 13 029C 831 BEQL 22$ ; branch if none
FD5F' 30 029E 832 BSBW RM$CHKNAM ; check nam validity
20 50 E9 02A1 833 BLBC R0,22$ ; branch if illegal (no error)
04 AE 34 A7 D0 02A4 834 MOVL NAM$_FNB(R7),FNB(SP) ; save file name status
34 A7 D4 02A9 835 CLRL NAM$_FNB(R7) ; clear file name status
01 AE 03 A7 90 02AC 836 MOVW NAM$_RSL(R7),RSL(SP) ; save result string
03 A7 94 02B1 837 CLRB NAM$_RSL(R7) ; clear size
6E 0B A7 90 02B4 838 MOVW NAM$_ESL(R7),ESL(SP) ; and expanded string
0B A7 94 02B8 839 CLRB NAM$_ESL(R7) ; clear size
14 A7 94 02BB 840 CLRB NAM$_DVI(R7) ; clear device ID
02BE 841
02BE 842 ASSUME NAM$_DID EQ NAM$_FID+6
02BE 843
24 A7 7C 02BE 844 CLRQ NAM$_FID(R7) ; and file ID's
2C A7 D4 02C1 845 CLRL NAM$_DID+2(R7) ;
02C4 846
03 69 25 E5 02C4 847 22$: BBCC #IFBSV_ACCESSED,(R9),25$ ; deaccess any open file or
FD35' 30 02C8 848 BSBW RM$DEACCESS ; network links
02CB 849 25$: $DASSGN S CHAN=IFBSV_CHNL(R9) ; deassign old channel
20 A9 B4 02D6 850 CLRW IFBSV_CHNL(R9) ; clear it
FD24' 30 02D9 851 BSBW RM$PRFLNMALT ; try again
04 50 E8 02DC 852 BLBS R0,30$ ; try next element
```



```
50 D5 02DF 853 TSTL R0 ; end of list?
06 13 02E1 854 BEQL 40$ ; no, so return the error
51 D4 02E3 855 30$: CLRL R1 ; found an element
SE 10 C0 02E5 856 ADDL2 #STACK_SIZE,SP ; discard saved context
05 02E8 857 RSB
02E9 858
02E9 859 ;
02E9 860 ; XPFN exited with RMS$_NOMLIST, no more search list to parse, so restore
02E9 861 ; the original error code and name block string lengths
02E9 862 ;
02E9 863 ;
57 28 A8 D0 02E9 864 40$: MOVL FAB$_NAM(R8),R7 ; get nam address
14 13 02ED 865 BEQL 42$ ; branch if none
FDOE' 30 02EF 866 BSBW RMSCHKNAM ; check nam validity
OE 50 E9 02F2 867 BLBC R0,42$ ; branch if illegal (no error)
0B A7 6E 90 02F5 868 MOVB ESL(SP),NAM$_ESL(R7) ; restore expanded string
03 A7 01 AE 90 02F9 869 MOVB RSL(SP),NAM$_RSL(R7) ; and result string
34 A7 04 AE D0 02FE 870 MOVL FNB(SP),NAM$_FNB(R7) ; restore file name flags
5E 08 C0 0303 871 42$: ADDL2 #8,SP ; discard NAM space
OC A8 8E D0 0306 872 MOVL (SP)+,FAB$_STV(R8) ; set stv secondary code
50 8E D0 030A 873 MOVL (SP)+,R0 ; restore error status
51 01 D0 030D 874 MOVL #1,R1 ; end-of-list encountered
05 0310 875 RSB ; exit
```



```
0311 877 .SUBTITLE RMSKNOWNFILE - Kernel Mode Known FILE Support
0311 878 :++
0311 879 :
0311 880 This routine is called, in kernel mode, when an open known file
0311 881 is found. The following operations are performed:
0311 882 :
0311 883 1. Check the volume and file protection to see if the user
0311 884 as access to the file.
0311 885 2. If the user has read access in addition to execute,
0311 886 report that fact as well.
0311 887 3. Increment the refcnt on the shared file
0311 888 window and to set the channel appropriately.
0311 889 :
0311 890 These operations must be interlocked against process deletion
0311 891 by executing at IPL 2.
0311 892 :
0311 893 Inputs:
0311 894 none
0311 895 :
0311 896 Outputs:
0311 897 R0 - $$$_NORMAL if access allowed
0311 898 $$$_NOPRIV if access denied
0311 899 :
0311 900 Implicit inputs:
0311 901 R11 - impure ptr
0311 902 R10 - FWA ptr
0311 903 R9 - IFB ptr
0311 904 R8 - FAB ptr
0311 905 :
0311 906 Implicit outputs:
0311 907 channel and window control blocks modified.
0311 908 :
0311 909 :--
0311 910 :
0311 911 ASSUME CHPCTL$_ACCESS EQ 0
0311 912 ASSUME CHPCTL$_FLAGS EQ 4
0311 913 ASSUME CHPCTL$_MODE EQ 8
0311 914 :
0311 915 RMSKNOWNFILE::
0311 916 .WORD ^M<R2,R3,R4,R5> ; save registers
0311 917 PUSHL #0 ; null access mode for volume
0311 918 PUSHL #<CHPCTL$_READ!CHPCTL$_USEREADALL> ; set CHPCTL flags
0311 919 PUSHL #ARMSM_READ ; set CHPCTL access
0311 920 MOVL SP,R2 ; point to CHPCTL
0311 921 BBC #IFB$V_WRTACC,(R9),10$ ; write access?
0311 922 BISL2 #ARMSM_WRITE,CHPCTL$_ACCESS(R2); check for it too
0311 923 XORL2 #<CHPCTL$_WRITE!CHPCTL$_USEREADALL>,-
0311 924 CHPCTL$_FLAGS(R2) ; set WRITE and clear USEREADALL
0311 925 10$: SETIPL #IPL$_ASTDEL ; prevent process deletion
0311 926 MOVL FAB$_CTX(R8),R0 ; get KFE
0311 927 MOVL KFESL_WCB(R0),R5 ; get WCB
0311 928 MOVL @#CTL$GL_PCB,R4 ; get PCB addr
0311 929 CLRL R3 ; no CHPRET
0311 930 :
0311 931 :
0311 932 : Check the volume protection in the UCB.
0311 933 :
```

003C 00 DD 0313 917
05 DD 0315 918
01 DD 0317 919
52 5E D0 0319 920
07 69 30 E1 031C 921
62 02 C8 0320 922
06 CC 0323 923
04 A2 0325 924
50 18 A8 D0 032A 926
55 18 A0 D0 032E 927
54 00000000 9F D0 0332 928
53 D4 0339 929


```
51 10 A5 DO 033B 934
51 1C A1 DO 033B 935      MOVL WCB$$_ORGUCB(R5),R1      ; get UCB
50 008C C4 DO 033F 936      MOVL UCB$$_ORB(R1),R1      ; get ORB addr
00000000 GF 16 0343 937      MOVL PCB$$_ARB(R4),R0      ; get ARB addr
5C 50 E9 0348 938      JSB G^EXE$CHKPRO_INT      ; check for volume access
                                034E 939      BLBC R0,100$      ; give up if no access
                                0351 940
                                0351 941      ;
                                0351 942      ; Now see if the user has requested access (READ implies EXECUTE)
                                0351 943      ;
                                0351 944      ;
0A A9 90 0351 945      MOVB IFB$$_MODE(R9),-      ; set CHPCTL access mode
08 A2 0354 946      CHPCTL$$_MODE(R2)      ; get FCB
51 18 A5 DO 0356 947      MOVL WCB$$_FCB(R5),R1      ; save high block for ifb
38 A1 DD 035A 948      PUSHL FCB$$_FILESIZE(R1)      ; get ORB addr
51 58 A1 DE 035D 949      MOVAL FCB$$_ORB(R1),R1      ; get ARB addr
50 008C C4 DO 0361 950      MOVL PCB$$_ARB(R4),R0      ; check for read access
00000000 GF 16 0366 951      JSB G^EXE$CHKPRO_INT      ; nope
06 50 E9 036C 952      BLBC R0,20$      ; tell user if so
16 A8 02 88 036F 953      BISB2 #FAB$$_GET,FAB$$_FAC(R8)      ; and continue
20 11 0373 954      BRB 30$
                                0375 955
                                0375 956      ;
                                0375 957      ; See if the user has execute-only access
                                0375 958      ;
                                0375 959      ;
33 22 A9 07 E1 0375 960 20$: BBC #FAB$$_EXE,IFB$$_FAC(R9),100$      ; execute access?
2F 69 30 E0 037A 961      BBS #IFB$$_WRTACC,(R9),100$      ; no special check if write
02 0A A9 91 037E 962      CMPB IFB$$_MODE(R9),#PSL$$_SUPER      ; can he ask for exe access?
29 1A 0382 963      BGTRU 100$      ; nope
04 DO 0384 964      MOVL #ARM$$_EXECUTE,-      ; try execute-only access
62 0386 965      CHPCTL$$_ACCESS(R2)      ; get ARB addr
50 008C C4 DO 0387 966      MOVL PCB$$_ARB(R4),R0      ; check for execute access
00000000 GF 16 038C 967      JSB G^EXE$CHKPRO_INT      ; nope, then return failure
18 50 E9 0392 968      BLBC R0,100$      ; get channel number
50 20 A9 3C 0395 969 30$: MOVZWL IFB$$_CHNL(R9),R0      ; R1 contains CCB address
00000000 GF 16 0399 970      JSB G^IOC$VERIFYCHAN      ; store WCB address in CCB
04 A1 55 DO 039F 971      MOVL R5,CCB$$_WIND(R1)      ; and count this user
OE A5 B6 03A3 972      INCW WCB$$_REFCNT(R5)      ; stuff high block into ifb
70 A9 6E DO 03A6 973      MOVL (SP),IFB$$_HBK(R9)      ; success!
50 01 DO 03AA 974      MOVL #1,R0      ; restore IPL
                                03AD 975 100$: SETIPL #0
                                03B0 976      RET
```

```
03B1 978 .SUBTITLE WRITE_AT_JNL - Local Subroutine
03B1 979 :++
03B1 980 : WRITE_AT_JNL
03B1 981 :
03B1 982 : This routine calls MAPJNL to write the AT journal buffer for file
03B1 983 : related information, if necessary.
03B1 984 :
03B1 985 : Inputs:
03B1 986 :
03B1 987 : r9 ifab
03B1 988 :
03B1 989 : Outputs:
03B1 990 :
03B1 991 : r0 success or failure
03B1 992 :
03B1 993 :--
03B1 994 :
03B1 995 WRITE_AT_JNL:
03B1 996
54 00A4 C9 D0 03B1 997 MOVL IFB$L_RJB(R9),R4 ; get the RJB address
17 0A A4 03 13 03B6 998 BEQL 50$ ; return to caller if none
55 0A A4 3C 03B8 999 BBC #RJB$V_AT,RJB$W_FLAGS(R4),50$ ; return if not AT
03C1 1000 MOVZWL RJB$W_FLAGS(R4),R5 ; save flags so we can turn everything
03C1 1001 ; off but AT for this MAPJNL call
03C1 1002 ; (AI, BI, RU written earlier)
0A A4 08 B0 03C1 1003 MOVW #RJB$M_AT,RJB$W_FLAGS(R4) ; could have used a BICW with
03C5 1004 ; complement operator, but this is easier
00000000'EF 16 03C5 1005 JSB RMS$MAPJNL ; write the AT buffer
54 00A4 C9 D0 03CB 1006 MOVL IFB$L_RJB(R9),R4 ; get the RJB address MAPJNL destroyed
0A A4 55 B0 03D0 1007 MOVW R5,RJB$W_FLAGS(R4) ; restore original flags
05 03D4 1008 50$: RSB ; return to caller
03D5 1009
03D5 1010 .END
```


RMSOOPEN
Symbol table

DISPATCH FOR OPEN OPERATION

H 8

16-SEP-1984 01:24:41 VAX/VMS Macro V04-00
5-SEP-1984 16:25:12 [RMS.SRC]RMSOOPEN.MAR;1

Page 24
(16)

```

$$PSECT_EP      = 00000000
$$RMSTEST       = 0000001A
$$RMS_PBUGCHK   = 00000010
$$RMS_TBUGCHK   = 00000008
$$RMS_UMODE     = 00000004
ARMSM_EXECUTE   = 00000004
ARMSM_READ      = 00000001
ARMSM_WRITE     = 00000002
CCBSL_WIND      = 00000004
CHK_SCIST       = 0000026D R      01
CHPCTLSB_MODE   = 00000008
CHPCTLSL_ACCESS = 00000000
CHPCTLSL_FLAGS  = 00000004
CHPCTLSM_READ   = 00000001
CHPCTLSM_USEREADALL = 00000004
CHPCTLSM_WRITE  = 00000002
CREOPEN_EXIT    = 000001E9 R      01
CTLSGL_PCB      = ***** X      01
DAPRTN         = 00000166 R      01
DEVSV_DMT       = 00000015
DEVSV_FOR       = 00000018
DEVSV_MNT       = 00000013
DEVSV_SQD       = 00000005
ERR             = 0000000C
ERRDNR         = 00000129 R      01
ERRIRC         = 00000130 R      01
ERROR          = 00000135 R      01
ERRRFM         = 00000122 R      01
ESL            = 00000000
EXESCHKPRO_INT = ***** X      01
FABS_BKS       = 0000003E
FABS_FAC       = 00000016
FABS_FSZ       = 0000003F
FABS_ORG       = 0000001D
FABS_RAT       = 0000001E
FABS_RFM       = 0000001F
FABS_FIX       = 00000001
FABS_MAXRFM    = 00000006
FABS_SEQ       = 00000000
FABS_STM       = 00000004
FABS_STMCR     = 00000006
FABS_STMLF     = 00000005
FABS_VFC       = 00000003
FABS_ALQ       = 00000010
FABS_CIX       = 00000018
FABS_FOP       = 00000004
FABS_NAM       = 00000028
FABS_STV       = 0000000C
FABS_BIO       = 00000020
FABS_BLK       = 00000008
FABS_BRO       = 00000040
FABS_CR        = 00000002
FABS_FTN       = 00000001
FABS_GET       = 00000002
FABS_PRN       = 00000004
FABS_ORG       = 00000004
FABS_BIO       = 00000005

```

```

FABS_BRO       = 00000006
FABS_DLT       = 0000000F
FABS_DMO       = 0000000C
FABS_EXE       = 00000007
FABS_KFO       = 0000001E
FABS_NAM       = 00000018
FABS_ORG       = 00000004
FABS_RWC       = 0000000B
FABS_RWO       = 00000007
FABS_SCF       = 0000000E
FABS_SPL       = 0000000D
FABS_UFO       = 00000011
FABS_DEQ       = 00000014
FABS_GBC       = 00000048
FABS_MRS       = 00000036
FCBSL_FILESIZE = 00000038
FCBSR_ORB      = 00000058
FIBSW_DID      = 0000000A
FNB            = 00000004
FOP            = 00000020
FWASQ_FIB      = 00000010
FWASV_EXP_VER  = 00000010
FWASV_NODE     = 00000019
FWASV_SLPRESENT = 00000038
FWASV_SL_PASS  = 00000002
IFBS_BKS       = 0000005E
IFBS_FAC       = 00000022
IFBS_FSZ       = 0000005F
IFBS_JNLFLG    = 000000A0
IFBS_MODE      = 0000000A
IFBS_ORGCASE   = 00000023
IFBS_RAT       = 00000051
IFBS_RFMORG    = 00000050
IFBS_IDX       = 00000002
IFBSL_DEVBUFSIZ = 00000048
IFBSL_EBK      = 00000074
IFBSL_HBK      = 00000070
IFBSL_PRIM_DEV = 00000000
IFBSL_RJB      = 000000A4
IFBS_RFM       = 00000004
IFBSV_ACCESSED = 00000025
IFBSV_AT       = 00000004
IFBSV_BIO      = 00000005
IFBSV_CREATE   = 00000032
IFBSV_DAP      = 0000003E
IFBSV_DLT      = 0000002B
IFBSV_DMO      = 00000028
IFBSV_PPF_IMAGE = 00000022
IFBSV_RFM      = 00000000
IFBSV_RWC      = 00000027
IFBSV_SCF      = 0000002A
IFBSV_SEQFIL   = 00000038
IFBSV_SPL      = 00000029
IFBSV_WRTACC   = 00000030
IFBSW_CHNL     = 00000020
IFBSW_DEQ      = 00000062
IFBSW_FFB      = 0000005C

```

RMS
V04-

RMSOOPEN
Symbol table

DISPATCH FOR OPEN OPERATION

I 8

16-SEP-1984 01:24:41 VAX/VMS Macro V04-00
5-SEP-1984 16:25:12 [RMS.SRC]RMSOOPEN.MAR;1

Page 25
(16)

IFBSW_GBC	= 00000064		
IFBSW_LRL	= 00000052		
IFBSW_MRS	= 00000060		
IFBSW_RTDEQ	= 0000004C		
INSSKF_SCAN	*****	X	01
IOCSVERIFYCHAN	*****	X	01
IPLS_ASTDEL	= 00000002		
KFESC_WCB	= 00000018		
NAMSB_ESL	= 0000000B		
NAMSB_RSL	= 00000003		
NAMSL_FNB	= 00000034		
NAMST_DVI	= 00000014		
NAMSW_DID	= 0000002A		
NAMSW_FID	= 00000024		
PCBSL_ARB	= 0000008C		
PIOSA_TRACE	*****	X	01
PRS_IPL	= 00000012		
PSLSC_SUPER	= 00000002		
RJBSP_AT	= 00000008		
RJBSPV_AT	= 00000003		
RJBSPV_FLAGS	= 0000000A		
RMSACCESS	*****	X	01
RMSCHKNAM	*****	X	01
RMSCHK_SLIST	0000026B	RG	01
RMSCHK_SLIST1	00000267	RG	01
RMSCLOSE3	*****	X	01
RMSCLSCU	*****	X	01
RMSCOPRTN	00000154	RG	01
RMSCOPRTN1	000001B0	RG	01
RMSCOPRTN2	000001BA	RG	01
RMSCREATEEXIT	00000243	RG	01
RMSCREATEEXIT1	00000245	RG	01
RMSDEACCESS	*****	X	01
RMSDEALLOCATE_FWA	*****	X	01
RMSXRMS	*****	X	01
RMSFILLNAM	*****	X	01
RMSFSETI	*****	X	01
RMSKNOWNFILE	00000311	RG	01
RMSMAPJNL	*****	X	01
RMSOPEN1	*****	X	01
RMSOPEN2	*****	X	01
RMSOPEN3	*****	X	01
RMSOPEN_ALT	00000013	RG	01
RMSOPEN_CIF	000000BC	RG	01
RMSPRFLRM	*****	X	01
RMSPRFLNMALT	*****	X	01
RMSRELEASALL	*****	X	01
RMSRETIFB	*****	X	01
RMSRET_DEV_CHAR	*****	X	01
RMSREWIND_AT	*****	X	01
RMSSETDID	*****	X	01
RMSLIST_ERRS	00000257	RG	01
RMSLIST_ERR_CNT	= 00000010	G	
RMSLIST_ERR_CNT1	= 0000000A	G	
RMSXABOPN_ARGS	00000000	RG	01
RMSXAB_SCAN	*****	X	01
RMS\$OPEN	= 00000008	RG	01

RMS\$_ACC	= 0001C002		
RMS\$_DEV	= 000184C4		
RMS\$_DNF	= 0001C04A		
RMS\$_DNR	= 00018272		
RMS\$_FND	= 0001C02A		
RMS\$_FNF	= 00018292		
RMS\$_IRC	= 0001857C		
RMS\$_KFF	= 00018031		
RMS\$_NMF	= 000182CA		
RMS\$_ORG	= 0001860C		
RMS\$_PRV	= 0001829A		
RMS\$_RFM	= 00018664		
RM_OPEN2_BR	0000011C	R	01
RSC	= 00000001		
STACK_SIZE	= 00000010		
STV	= 00000008		
SYSSCMKRN	*****	GX	01
SYSSDASSGN	*****	GX	01
S_LOOP	0000028D	R	01
TPTSL_OPEN	*****	X	01
UCBSL_ORB	= 0000001C		
WCB\$\$_FCB	= 00000018		
WCB\$\$_ORGUCB	= 00000010		
WCB\$\$_REFCNT	= 0000000E		
WRITE_AT_JNL	000003B1	R	01
XAB\$\$_ALL	= 00000014		
XAB\$\$_ALLLEN	= 00000020		
XAB\$\$_KEY	= 00000015		
XAB\$\$_KEYLEN_V2	= 00000040		
XAB\$\$_SUM	= 00000016		
XAB\$\$_SUMLEN	= 0000000C		
XBC\$\$_OPNALL3	*****	X	01
XBC\$\$_OPNKEY3	*****	X	01
XBC\$\$_OPNSUM3	*****	X	01

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

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
RM\$RMS	000003D5 (981.)	01 (1.)	PIC USR CON REL GBL NOSHR EXE RD NOWRT NOVEC BYTE
\$ABSS	00000000 (0.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE

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

Phase	Page faults	CPU Time	Elapsed Time
Initialization	30	00:00:00.09	00:00:00.77
Command processing	129	00:00:00.82	00:00:08.71
Pass 1	576	00:00:24.33	00:00:59.44
Symbol table sort	0	00:00:03.74	00:00:04.74
Pass 2	185	00:00:04.76	00:00:09.34
Symbol table output	24	00:00:00.25	00:00:00.31
Psect synopsis output	2	00:00:00.02	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	948	00:00:34.01	00:01:23.34

The working set limit was 1800 pages.
136208 bytes (267 pages) of virtual memory were used to buffer the intermediate code.
There were 140 pages of symbol table space allocated to hold 2557 non-local and 49 local symbols.
1010 source lines were read in Pass 1, producing 17 object records in Pass 2.
46 pages of virtual memory were used to define 45 macros.

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

Macro library name	Macros defined
_\$255\$DUA28:[RMS.OBJ]RMS.MLB;1	19
-\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	10
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2	12
TOTALS (all libraries)	41

2742 GETs were required to define 41 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:RMSOOPEN/OBJ=OBJ\$:RMSOOPEN MSRC\$:RMSOOPEN/UPDATE=(ENH\$:RMSOOPEN)+EXECML\$/LIB+LIB\$:RMS/LIB

0330 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

