



```

RRRRRRRR      MM      MM      SSSSSSSS      000000      CCCCCCCC      LL      000000      SSSSSSSS      EEEEEEEEEEE
RRRRRRRR      MM      MM      SSSSSSSS      000000      CCCCCCCC      LL      000000      SSSSSSSS      EEEEEEEEEEE
RR      RR      MMMM      MMMM      SS      00      00      00      CC      LL      00      00      SS      EE
RR      RR      MMMM      MMMM      SS      00      00      00      CC      LL      00      00      SS      EE
RR      RR      MM      MM      SS      00      0000      CC      LL      00      00      SS      EE
RR      RR      MM      MM      SS      00      0000      CC      LL      00      00      SS      EE
RRRRRRRR      MM      MM      SSSSSS      00      00      00      CC      LL      00      00      SSSSSS      EEEEEEEEE
RRRRRRRR      MM      MM      SSSSSS      00      00      00      CC      LL      00      00      SSSSSS      EEEEEEEEE
RR      RR      MM      MM      SS      0000      00      CC      LL      00      00      SS      EE
RR      RR      MM      MM      SS      0000      00      CC      LL      00      00      SS      EE
RR      RR      MM      MM      SS      00      00      CC      LL      00      00      SS      EE
RR      RR      MM      MM      SS      00      00      CC      LL      00      00      SS      EE
RR      RR      MM      MM      SSSSSSSS      000000      CCCCCCCC      LLLLLLLLLL      000000      SSSSSSSS      EEEEEEEEEEE
RR      RR      MM      MM      SSSSSSSS      000000      CCCCCCCC      LLLLLLLLLL      000000      SSSSSSSS      EEEEEEEEEEE

```

```

LL      I I I I I      SSSSSSSS
LL      I I I I I      SSSSSSSS
LL      I I      SS
LL      I I      SS
LL      I I      SS
LL      I I      SS
LL      I I      SSSSSS
LL      I I      SSSSSS
LL      I I      SS
LL      I I      SS
LL      I I      SS
LL      I I      SS
LLLLLLLLLLLL      I I I I I      SSSSSSSS
LLLLLLLLLLLL      I I I I I      SSSSSSSS

```

(3)	172
(4)	215
(7)	449
(8)	481
(9)	503
(11)	679
(12)	824

DECLARATIONS

RMS\$CLOSE, \$CLOSE Routine
RMS\$CLSCU, Cleanup IFAB and Exit RMS
RMS\$RETIFB, Return IFAB but Leave File Open
RMS\$CLEANUP, Cleanup IFAB and Associated Storage
RMS\$SPL SCF - \$CLOSE routine for spool/submit options
RMS\$RELEASALL, Release all BDB's

```
0000 1          $BEGIN RMSOCLOSE,000,RMSRMS,<DISPATCH FOR CLOSE 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 :
0000 30 : Facility: rms32
0000 31 :
0000 32 : Abstract:
0000 33 :         this module is the highest level control routine
0000 34 :         to perform the $close function.
0000 35 :
0000 36 : Environment:
0000 37 :         star processor running starlet exec.
0000 38 :
0000 39 : Author: L F Laverdure,         creation date: 1-MAR-1977
0000 40 :
0000 41 : Modified By:
0000 42 :
0000 43 :         V03-033 SHZ0010         Stephen H. Zalewski,         04-May-1984
0000 44 :         Do not recreate address space in rms$unmap_gbl because
0000 45 :         the space is now taken directly from P0 space.
0000 46 :
0000 47 :         V03-032 JEJ0011         J E Johnson         20-Mar-1984
0000 48 :         Include global buffer quota accounting.
0000 49 :
0000 50 :         V03-031 JEJ0020         J E Johnson         28-Mar-1984
0000 51 :         Correct multiple problems in RAS270.
0000 52 :
0000 53 :         V03-030 RAS0270         Ron Schaefer         14-Mar-1984
0000 54 :         Remove the NAM block dependency for the SPL/SCF/DLT on
0000 55 :         $CLOSE options. Eliminate RM$CLOSE1.
0000 56 :
0000 57 :         V03-029 DGB0011         Donald G. Blair         01-Mar-1984
0000 58 :         Change the way the ACP is called as part of the restructuring
0000 59 :         necessary for access mode protected files.
0000 60 :
0000 61 :         V03-028 JWT0160         Jim Teague         29-Feb-1984
0000 62 :         Remove calls to RM$DEALLEFN.
0000 63 :
0000 64 :         V03-027 SHZ0009         Stephen H. Zalewski         12-Sep-1983
0000 65 :         Change the sense of a branch so that the NWA DOES get
0000 66 :         deallocated during the close.
0000 67 :
0000 68 :         V03-026 SHZ0008         Stephen H. Zalewski         10-Aug-1983
0000 69 :         Set a bit in the GBSB after we have decremented the accessor
0000 70 :         count in the global buffer section (and possibly flushed the
0000 71 :         cache as well) to prevent last chance
0000 72 :         from decrementing the count again in the case where the
0000 73 :         process gets stopped before we have completely cleaned up.
0000 74 :
0000 75 :         V03-025 SHZ0007         Stephen H. Zalewski         02-Aug-1983
0000 76 :         If last accessor to a global buffer section, then zero
0000 77 :         the global section size, and global buffer count fields
0000 78 :         in the lock value block for the section. This is to prevent
0000 79 :         the case where the next accessor takes a lock before we
0000 80 :         are done cleaning up, thus causing this subsequent accessor
0000 81 :         to get incorrect data in the value block.
0000 82 :
0000 83 :         V03-024 KBT0567         Keith B. Thompson         28-Jul-1983
0000 84 :         Check for the NWA pointer not the flag
```

```
0000 85 :
0000 86 :
0000 87 :
0000 88 :
0000 89 :
0000 90 :
0000 91 :
0000 92 :
0000 93 :
0000 94 :
0000 95 :
0000 96 :
0000 97 :
0000 98 :
0000 99 :
0000 100 :
0000 101 :
0000 102 :
0000 103 :
0000 104 :
0000 105 :
0000 106 :
0000 107 :
0000 108 :
0000 109 :
0000 110 :
0000 111 :
0000 112 :
0000 113 :
0000 114 :
0000 115 :
0000 116 :
0000 117 :
0000 118 :
0000 119 :
0000 120 :
0000 121 :
0000 122 :
0000 123 :
0000 124 :
0000 125 :
0000 126 :
0000 127 :
0000 128 :
0000 129 :
0000 130 :
0000 131 :
0000 132 :
0000 133 :
0000 134 :
0000 135 :
0000 136 :
0000 137 :
0000 138 :
0000 139 :
0000 140 :
0000 141 :
```

V03-023 DAS001 David Solomon 20-Jul-1983  
IFB\$V\_JNL moved from IFB\$B\_JNLFLG to IFB\$B\_JNLFLG2.

V03-022 KPL003 Peter Lieberwirth 20-Jun-1983  
Correct commentary regarding AT journaling of \$CLOSE.

V03-021 SHZ006 Stephen H. Zalewski 20-Jun-1983  
Add code for cluster global buffers.

V03-020 KPL002 Peter Lieberwirth 31-May-1983  
Write CLOSE mapping journal entry before buffers are  
deallocated.

V03-019 KBT0520 Keith B. Thompson 23-May-1983  
The routine RM\$CHKNAMBLK no longer exist and remove  
the RM\$CLS\_CHKNAM hack.

V03-018 KBT0509 Keith B. Thompson 5-May-1983  
Use RM\$DEALLOCATE\_FWA

V03-017 KPL001 Peter Lieberwirth 29-Apr-1983  
Always look for IFAB journaling structures to deallocate.

V03-016 JWH0200 Jeffrey W. Horn 22-Mar-1983  
Back out JWH0102, \$CLOSE now allowed within an active  
RU.

V03-015 KBT0457 Keith B. Thompson 10-Jan-1983  
Deallocate fwa along with ifab.

V03-014 SHZ005 Stephen H. Zalewski, 19-Oct-1982 16:32  
Remove code that checks for corrupt gbd's in a global  
buffer section.

V03-013 JWH0104 Jeffrey W. Horn 22-Sep-1982  
Relocate calls to RMSMAPJNL, RM\$DEAJNL to after deaccess.

V03-012 JWH0102 Jeffrey W. Horn 22-Sep-1982  
If the file is in a recovery-unit then don't allow the user  
to close the file.

V03-011 SHZ0004 Stephen H. Zalewski, 10-Sep-1982 17:04  
Remove all references to SFD, SIFB and FRB as they no longer  
exist.

V03-010 KBT0339 Keith B. Thompson 16-Sep-1982  
Always try to deallocate the gbsb (i.e. take it out of unmap)

V03-009 SHZ0003 Stephen H. Zalewski, 7-Sep-1982 22:24  
Add TEMPORARY code that checks for corruption of GBDs  
in global buffer section.

Move dequeuing of global buffer section lock so that when  
a global section is unmapped the lock is immediately dequeued.  
This prevents a second process from initializing a new section  
incorrectly.

```
0000 142 :  
0000 143 :  
0000 144 :  
0000 145 :  
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 :  
0000 159 :  
0000 160 :  
0000 161 :  
0000 162 :  
0000 163 :  
0000 164 :  
0000 165 :  
0000 166 :  
0000 167 :  
0000 168 :  
0000 169 :  
0000 170 :--
```

V03-008 SHZ0002 Stephen H. Zalewski, 1-Sep-1982 15:37  
If stream has global buffers, then dequeue the lock it  
had on the global buffer section, and remove its GBSB.

V03-007 KBT0179 Keith B. Thompson 23-Aug-1982  
Reorganize psects and rename entry point to single 's'

V03-006 KBT0110 Keith B. Thompson 16-Jul-1982  
Deallocate the sfsb at a more appropriate time

V03-005 TMK0001 Todd M. katz 02-Jul-1982  
Implement the RMS cluster solution for next record positioning.  
As the next record context is now kept locally in the IRAB  
instead of in individual NRP cells, there is no NRP list  
to be returned.

V03-004 KDM0002 Kathleen D. Morse 28-Jun-1982  
Added \$PRDEF.

V03-003 JWH0002 Jeffrey W. Horn 19-May-1982  
Add support for journaling; Write out closing mapping entry  
and then call RMSDEAJNL.

V03-002 SHZ0001 Stephen H. Zalewski, 8-Jun-1982 17:02  
Remove instruction that cleared kernel mode flag after the  
SIFAB was released (moved to RMOSHARE).

```

0000 172      .SBTTL  DECLARATIONS
0000 173
0000 174      :
0000 175      : Macros:
0000 176      :
0000 177
0000 178      $BDBDEF
0000 179      $BLBDEF      ; bucket lock block defs
0000 180      $DEVDEF
0000 181      $FABDEF
0000 182      $FIBDEF
0000 183      $FWADEF
0000 184      $GBDDEF
0000 185      $GBHDEF
0000 186      $GBPDEF
0000 187      $GBSDEF
0000 188      $IFBDEF
0000 189      $IMPDEF
0000 190      $IODEF      ; io definitions
0000 191      $IRBDEF
0000 192      $LCKDEF
0000 193      $NAMDEF
0000 194      $PSLDEF      ; psl definitions
0000 195      $RABDEF
0000 196      $RLBDEF      ; record lock block defintions
0000 197      $RLSDEF
0000 198      $RMSDEF
0000 199      $SJCDEF
0000 200
0000 201      :
0000 202      : Equated Symbols:
0000 203      :
0000 204
00000020 0000 205      FOP=FAB$L_FOP*8
00000020 0000 206      BKP=IRB$L_BKPBITS*8
0000 207
0000 208      :
0000 209      : Own Storage:
0000 210      :
0000 211
000001FF 0000 212      MASK      = ^X1FF      ; mask for getting to page boundary
0000 213

```



```

0000 215      .SBTTL  RMS$CLOSE, $CLOSE Routine
0000 216      :++
0000 217      :
0000 218      : RMS$$CLOSE - highest level file close routine
0000 219      :
0000 220      : this routine performs the highest level $close processing.
0000 221      : its functions include:
0000 222      :
0000 223      : 1. common setup
0000 224      : 2. check for all streams idle, exiting if not
0000 225      : 3. force a disconnect for all streams
0000 226      : 4. dispatch to organization-dependent code
0000 227      : 5. if the dlt fop bit is set and neither spl nor scf is set,
0000 228      :    delete the file
0000 229      : 6. return all bdb's and buffers
0000 230      : 7. deaccess the file if accessed
0000 231      : 8. return the asb and nwa (if any), the ifab, and all pages used for
0000 232      :    rms internal structures for this file
0000 233      : 9. zero the ifab table pointer and fab$w_ifi
0000 234      : 10. exit to the user, generating an ast if requested
0000 235      :
0000 236      :
0000 237      : Calling sequence:
0000 238      :
0000 239      : entered from exec as a result of user's calling sys$close
0000 240      : (e.g., by using the $close macro).
0000 241      :
0000 242      : Input Parameters:
0000 243      :
0000 244      : ap      user's argument list addr
0000 245      :
0000 246      : Implicit Inputs:
0000 247      :
0000 248      : the contents of the fab and possible related user interface
0000 249      : blocks.
0000 250      :
0000 251      : Output Parameters:
0000 252      :
0000 253      : r0      status code
0000 254      : r1      destroyed
0000 255      :
0000 256      : Implicit Outputs:
0000 257      :
0000 258      : the ifab and all related internal rms structures are vaporized.
0000 259      : fab$l_sts and fab$l_stv are output and fab$w_ifi is zeroed if the
0000 260      : close was successful.
0000 261      :
0000 262      : a completion ast is queued if so specified by the user.
0000 263      :
0000 264      : Completion Codes:
0000 265      :
0000 266      : standard rms
0000 267      :
0000 268      : Side Effects:
0000 269      :
0000 270      : none
0000 271      :

```

```

0000 272 :--
0000 273 :
0000 274 $ENTRY RMS$CLOSE
0000 275 $STPT CLOSE
FFF7' 30 0006 276 BSBW RMS$FSET ; do common setup
0009 277 ; note: does not return on error
0009 278
0009 279 :++
0009 280 : force a disconnect on all streams
0009 281 :
0009 282 :
01 DD 0009 283 PUSHL #1 ; status code to stack
3B 11 000B 284 BRB NXTIRAB ; go check if any irabs linked
000D 285
000D 286 :
000D 287 : do an effective bsb to the org-specific disconnect
000D 288 :
000D 289 : note: this makes the fab look like a rab, but is of no consequence
000D 290 : since there are no rab inputs to the internal disconnect
000D 291 : and the only outputs are stv and isi (zeroed)
000D 292 :
000D 293 :
000D 294 ASSUME RAB$L_STV EQ FAB$L_STV
000D 295 ASSUME RAB$W_ISI EQ FAB$W_IFI
000D 296 ASSUME RAB$C_BLN LE FAB$C_BLN ; (necessary for the re-probe on stall)
000D 297
000D 298 :
000D 299 : must clear the async operation bit to avoid arglist copy to asb, set busy,
000D 300 : and initialize ppf_image bit correctly
000D 301 :--
000D 302 :
000D 303 :
000D 304 NXTDISC:
04 OC 8A 000D 305 BICB #<1a<IRB$V_ASYNC-BKP>>!<1a<IRB$V_PPF_IMAGE-BKP>>,-
04 A9 000F 306 IRB$L_BKPBITS(R9)
0A A9 57 90 0011 307 SSB #IRB$V_BUSY,(R9) ; say rab busy
04 6A 22 E1 0015 308 MOVB R7,IRB$B_MODE(R9) ; set mode into irab
0280 8F BB 0019 309 BBC #IFB$V_PPF_IMAGE,(R10),10$ ; branch unless indirect ppf
36 AF 9F 001D 310 SSB #IRB$V_PPF_IMAGE,(R9) ; say irab accessed indirectly
0021 311 10$: PUSHR #^M<R7,R9> ; save mode and irab addr
0025 312 PUSHAB B^NXTIRAB ; return pc to stack
0028 313 CASE TYPE=B, SRC=IFB$B_ORGCASE(R10),-
0028 314 DISPLIST=<RMS$DISCONNECT1,RMS$DISCOMMONSUC,RMS$DISCONNECT3>
FFCA' 31 0033 315 BRW RMS$DISCOMMONSUC ; handle unknown org (blk i/o)
0280 8F BA 0036 316 NXTIRAB: POPR #^M<R7,R9> ; restore mode and (deallocated irab addr
003A 317 ; (link still valid)
04 6A 22 E1 003A 318 BBC #IFB$V_PPF_IMAGE,(R10),10$ ; branch unless indirect ppf
003E 319 CSB #IRB$V_BUSY,(R9) ; say irab not busy anymore
03 50 E8 0042 320 10$: BLBS R0,NXTIRAB ; branch if no error
6E 50 D0 0045 321 MOVL R0,(SP) ; replace status code
0048 322
0048 323 ASSUME IFB$L_IRAB_LNK EQ IRB$L_IRAB_LNK
0048 324
0048 325 NXTIRAB:
59 1C A9 D0 0048 326 MOVL IRB$L_IRAB_LNK(R9),R9 ; get next irab
004C 327
004C 328 ;*****

```

```

004C 329 :
004C 330 : note: the next irab link must still be good even though previous irab
004C 331 : is deallocated, since nothing else could have re-used the space.
004C 332 :
004C 333 :****
004C 334 :
59 BF 12 004C 335 BNEQ NXTDISC ; loop if more irabs
5A 5A D0 004E 336 MOVL R10,R9 ; restore ifab address
0051 337 :
0051 338 :++
0051 339 :
0051 340 : get or of fab options that are input to either open/create or close
0051 341 :
0051 342 :--
12 69 22 E0 0051 343 :
0051 344 BBS #IFBSV_PPF_IMAGE,(R9),10$ ; branch if indirect ppf
0055 345 :
0055 346 ASSUME FAB$V_RWC+1 EQ FAB$V_DMO
0055 347 ASSUME FAB$V_DMO+1 EQ FAB$V_SPL
0055 348 ASSUME FAB$V_SPL+1 EQ FAB$V_SCF
0055 349 ASSUME FAB$V_SCF+1 EQ FAB$V_DLT
51 68 05 2B EF 0055 350 :
0055 351 EXTZV #FAB$V_RWC+FOP,#5,(R8),R1 ; get option bits from fab
005A 352 :
005A 353 ASSUME IFBSV_RWC+1 EQ IFBSV_DMO
005A 354 ASSUME IFBSV_DMO+1 EQ IFBSV_SPL
005A 355 ASSUME IFBSV_SPL+1 EQ IFBSV_SCF
005A 356 ASSUME IFBSV_SCF+1 EQ IFBSV_DLT
50 69 05 27 EF 005A 357 :
005A 358 EXTZV #IFBSV_RWC,#5,(R9),R0 ; get saved ifab copies from $open
50 69 50 51 88 005F 359 BISB2 R1,R0 ; or them together
69 05 27 50 F0 0062 360 INSV R0,#IFBSV_RWC,#5,(R9) ; and restore in ifab flags
0067 361 :
0067 362 :++
0067 363 :
0067 364 : dispatch to organization-dependent close code
0067 365 :
0067 366 : register state for dispatch:
0067 367 :
0067 368 : r11 impure area address
0067 369 : r10 ifab address
0067 70 : r9 ifab address
0067 371 : r8 fab address
0067 372 : (sp) return address
0067 373 : 4(sp) status code
0067 374 :
0067 375 :--
0067 376 :
0067 377 : PUSHAB B^CLSDLT ; return pc to stack
0067 378 : CASE TYPE=B,SRC=IFBSB_ORGCASE(R9),-
0067 379 : DISPLIST=<RMS$NULL,RMS$NULL,RMS$CLOSE3> ; pick up correct routine
0067 380 : TSTL (SP)+ ; remove return pc for other orgs
0067 381 :
0067 382 :++
0067 383 :
0067 384 : NOTE: Since there is only a special close routine for isam make life
0067 385 : a little simpler. If the above code is ever used the call to

```

```
0067 386 : rm$close3 will have to be done through rm3face since the word
0067 387 : branch of the case will not reach the real rm$close3
0067 388 :
0067 389 :--
0067 390 :
23 A9 02 91 0067 391 10$: CMPB #IFB$C_IDX,IFB$B_ORGCASE(R9) ; Is this isam
00000000'EF 06 12 0068 392 BNEQ CLSDLT ; Branch if not
16 006D 393 JSB RMS$CLOSE3 ; Do the close
0073 394
0073 395
```

```

0073 397
0073 398 :++
0073 399 :
0073 400 : return here from organization-dependent close routines
0073 401 :
0073 402 :--
0073 403 :
0073 404 : check for dlt fop bit set.
0073 405 : if set and the spl and scf bits are clear, delete the file.
0073 406 :
0073 407 :--
0073 408 :
47 69 2B E1 0073 409 CLSDLT: BBC #IFBSV_DLT,(R9),CLSCU1 ; branch if dlt not speced
0077 410 :
0077 411 :
0077 412 : if this is a network operation, do not process dlt option; it will be
0077 413 : handled by network code during deaccess.
0077 414 :
0077 415 :
69 0D E0 0077 416 BBS #DEVSV_NET,IFBSL_PRIM_DEV(R9),- ; branch if network operation
43 007A 417 CLSCU1
007B 418
007B 419 ASSUME IFBSV_SCF EQ IFBSV_SPL+1
007B 420
00 69 02 29 ED 007B 421 CMPZV #IFBSV_SPL,#2,(R9),#0 ; spl and scf both 0?
3C 12 0080 422 BNEQ CLSCU1 ; branch if not
69 1C E1 0082 423 BBC #DEVSV_RND,IFBSL_PRIM_DEV(R9),- ; ignore if not disk
38 0085 424 CLSCU1
69 18 E0 0086 425 BBS #DEVSV_FOR,IFBSL_PRIM_DEV(R9),- ; ignore if mntd foreign
34 0089 426 CLSCU1
30 69 22 E0 008A 427 BBS #IFBSV_PPF_IMAGE,(R9),CLSCU1 ; ignore if indirect ppf
008E 428
008E 429 :
008E 430 : build fib for delete
008E 431 :
008E 432 :
5A 38 A9 D0 008E 433 BLDFIB: MOVL IFBSL_FWA_PTR(R9),R10 ; get FWA addr
51 14 AA D0 0092 434 MOVL FWASQ_FIB#4(R10),R1 ; get FIB addr
05 69 35 E0 0096 435 BBS #IFBSV_TMP,(R9),DELETE ; branch if no directory entry
009A 436 SSB #IFBSV_FINDFID,IFBSW_NMC1L(R1) ; indicate find via fid
009F 437 : (0 fid => current file)
50 0135 8F 3C 009F 438 DELETE: MOVZWL #<IOS_DELETE!IOSM_DELETE>,R0 ; set i/o func code
7E 7C 00A4 439 CLRQ -(SP) ; p6 = p5 = 0 for qio
7E 7C 00A6 440 CLRQ -(SP) ; p4 = p3 = 0 for qio
7E D4 00A8 441 CLRL -(SP) ; p2 = 0 for qio
FF53' 30 00AA 442 BSBW RMSFCPFNC ; call acp to do the delete
OE 50 E8 00AD 443 BLBS R0,CLSCU1 ; branch if okay
00B0 444 RMSERR MKD,(SP) ; replace error code
OC A8 OC A9 DU 00B5 445 MOVL IFBSL_IOS(R9),FABSL_STV(R8) ; return system error code
02 11 00BA 446 BRB CLSCUT ; return to common close
00BC 447

```

```
00BC 449 .SBTTL RMSCLSCU, Cleanup IFAB and Exit RMS
00BC 450
00BC 451 :++
00BC 452 : RMSCLSCU - clean part of close operation for error paths
00BC 453 :
00BC 454 : entry point to clean up on aborted $open or $create
00BC 455 : or successful $erase,$sparse or other control routine
00BC 456 :
00BC 457 :--
00BC 458
00BC 459 RMSCLSCU::
50 DD 00BC 460 PUSHL R0 ; save error code
00BE 461
00BE 462 :
00BE 463 : close of indirectly opened process permanent file.
00BE 464 : need merely zero the ifi field.
00BE 465 :
00BE 466
09 69 22 E1 00BE 467 CLSCU1: BBC #IFBSV PPF IMAGE,(R9),20$ ; branch if not indirect ppf
02 AB B4 00C2 468 CLRW FABSW_IFI(R8)
50 8ED0 00C5 469 POPL R0 ; restore status code
FF35' 31 00CB 470 BRW RMSEX RMS ; all set
00CB 471
00CB 472 :
00CB 473 : cleanup ifab and all associated structures
00CB 474 :
00CB 475
03 11 10 00CB 476 20$: BSBB RMSCLEANUP ; cleanup ifab and stuff
50 50 E9 00CD 477 BLBC R0,CLS_EX ; branch if cleanup error
50 8ED0 00D0 478 POPL R0
FF2A' 31 00D3 479 CLS_EX: BRW RMSEX_NOSTR ; and do structureless exit
```

```

00D6 481      .SBTTL RMSRETIFB, Return IFAB but Leave File Open
00D6 482
00D6 483      :++
00D6 484      :
00D6 485      RMSRETIFB - evaporate internal structures but leave accessed
00D6 486      :
00D6 487      Entry point to leave file open for user but remove all rms
00D6 488      knowledge of the file
00D6 489      :
00D6 490      outputs:
00D6 491      :
00D6 492      r0 = status code
00D6 493      :
00D6 494      :--
00D6 495
00D6 496 RMSRETIFB::
FA AF 9F 00D6 497 PUSHAB B^CLS_EX      ; return address from cleanup
      50 DD 00D9 498 PUSHL  R0          ; save status code
0075 31 00DB 499 BRW      CLNJNL      ; only return ifab, etc.
      00DE 500      ; by jumping into cleanup
      00DE 501

```

RM  
SY  
SS  
SS  
SS  
SS  
SS  
SS  
BDI  
BDI  
BDI  
BDI  
BDI  
BK  
BLI  
BL  
CH  
CL  
CL  
CL  
CL  
CL  
CL  
DE  
DE  
DE  
DE  
DE  
DE  
DE  
DE  
DO  
DR  
ER  
FA  
FA  
FA  
FA  
FA  
FA  
FA  
FI  
FI  
FI  
FO  
FT  
FT  
FW  
FW  
GB  
GB  
GB





```

0102 560 :
0102 561 : if this is a network operation, do not process scf and spl options by
0102 562 : rm$close1; these will be handled by network code during deaccess.
0102 563 :
0102 564 :
15 69 0D E0 0102 565 5$: BBS #DEV$V_NET,IFB$L_PRIM_DEV(R9),10$ ; branch if network operation
0106 566
0106 567 ASSUME FAB$C_SEQ EQ 0
0106 568
0106 569 TSTB IFB$B_ORGCASE(R9) ; sequential file org?
0109 570 BNEQ 10$ ; branch if not
00 69 02 29 ED 0108 571 CMPZV #IFB$V_SPL,#2,(R9),#0 ; spl and scf both 0?
0110 572 BEQL 10$ ; branch if so
0086 30 0112 573 BSBW RM$SPL_SCF ; check for spool or submit
03 50 E8 0115 574 BLBS R0,10$ ; branch on success
6E 50 D0 0118 575 MOVL R0,(SP) ; save error code
0118 576
1F 69 14 E0 0118 577 10$: BBS #DEV$V_MBX,IFB$L_PRIM_DEV(R9),CLSMLBX ; branch if mailbox
25 69 05 E1 011F 578 BBC #DEV$V_SQD,IFB$L_PRIM_DEV(R9),DEASSIGN ; branch if not magtape
0123 579
0123 580 :
0123 581 : foreign magtape - write end of tape if ifb$v_eof is set
0123 582 : rewind if rwc (rewind on close) is set
0123 583 :
0123 584 :
0123 585 CLSMAGTAP:
21 69 18 E1 0123 586 BBC #DEV$V_FOR,IFB$L_PRIM_DEV(R9),DEASSIGN ; branch if not foreign
FED6' 30 0127 587 BSBW RM$WTTAPMARK ; write tape marks
18 50 E9 012A 588 BLBC R0,DEASSIGN_ALT ; go away if error
012D 589
17 69 27 E1 012D 590 BBC #IFB$V_RWC,(R9),DEASSIGN ; branch if no rewind
5A 5A DD 0131 591 PUSHL R10 ; rewind wants ifab in r10
5A 59 DD 0133 592 MOVL R9,R10 ; r10 <- ifab
FEC7' 30 0136 593 BSBW RM$REWIND_MT ; do rewind
5A 8ED0 0139 594 POPL R10 ; restore r10
07 11 013C 595 BRB DEASSIGN_ALT ; join mainstream
013E 596
013E 597 :
013E 598 : mail box - write end of file if write access was allowed
013E 599 :
013E 600 :
013E 601 CLSMLBX:
06 69 30 E1 013E 602 BBC #IFB$V_WRTACC,(R9),DEASSIGN ; branch if no write
FEBB' 30 0142 603 BSBW RM$WRITEOF ; write end of file
0145 604
0145 605 DEASSIGN ALT:
6E 50 D0 0145 606 MOVL R0,(SP) ; save status
0148 607
0148 608 :
0148 609 : deassign i/o channel
0148 610 :
0148 611 :
0148 612 DEASSIGN:
0148 613 $DASSGN_S IFB$W_CHNL(R9)
0153 614

```

RMS  
Pse  
  
PSE  
---  
RM  
\$AE  
  
Pha  
---  
Ini  
Com  
Pas  
Syn  
Pas  
Syn  
Pse  
Crc  
Ass  
  
The  
14C  
The  
991  
48  
  
Mac  
---  
-S  
-S  
-S  
TO1  
  
289  
  
The  
MAC

```

0153 616 :
0153 617 : clean up journaling
0153 618 :
0153 619 :
0153 620 CLNJNL:
00000000'EF 16 0153 621 JSB RMSDEAJNL ; clean up journaling
06 50 E8 0159 622 BLBS RO,RETIFB ; branch on success
03 6E E9 015C 623 BLBC (SP),RETIFB ; branch if code already error
6E 50 D0 015F 624 MOVL RO,(SP) ; otherwise replace status code
0162 625 :
0162 626 :
0162 627 : Return the ifab, sfsb, asb, fwa(s) and nwa if any
0162 628 :
0162 629 :
0162 630 RETIFB:
5C 18 A9 D0 0162 631 MOVL IFB$$_ARGLST(R9),AP ; restore arglist address
57 0A A9 9A 0166 632 MOVZBL IFB$$_MODE(R9),R7 ; and caller's mode
FE93' 30 016A 633 BSBW RMSRLS_SFSB ; clean up sfsb if any
3C A9 D5 016D 634 TSTL IFB$$_NWA_PTR(R9) ; check for nwa
03 13 0170 635 BEQL 10$ ; branch if nwa not present
FE8B' 30 0172 636 BSBW NTSNWA_FREE ; deallocate nwa
54 14 A9 D0 0175 637 10$: MOVL IFB$$_ASBADDR(R9),R4 ; get asb addr
09 13 0179 638 BEQL 20$ ; branch if none
53 5B D0 017B 639 MOVL R11,R3 ; ifab asb came from imp. free space header
FE7F' 30 017E 640 BSBW RMSRETBLK ; deallocate asb
14 A9 D4 0181 641 CLRL IFB$$_ASBADDR(R9) ; clear pointer
FE79' 30 0184 642 20$: BSBW RMSDEALLOCATE_FWA ; deallocate fwa and related structures
53 59 7D 0187 643 MOVL R9,R3 ; ifab addr to right regs for retblk
FE73' 30 018A 644 BSBW RMSRETBLK ; deallocate ifab
018D 645 :
018D 646 :
018D 647 : Return the pages used for all internal structures
018D 648 :
018D 649 :
5A 08 C2 018D 650 SUBL2 #8,R10 ; get start of page addr
54 6A D0 0190 651 MOVL (R10),R4 ; get 1st hole (there must be
0193 652 ; at least 1 hole for ifab)
000001F8 8F 56 64 D0 0193 653 40$: MOVL (R4),R6 ; get next free space hole
08 A4 D1 0196 654 CMPL 8(R4),#504 ; all holes should now be equal
019E 655 ; to one page in length
019E 656 ; (less list head if page 1)
12 1F 019E 657 BLSSU ERRBUG ; branch if not
FE5D' 30 01A0 658 BSBW RMSRET1PAG ; return it
54 56 D0 01A3 659 MOVL R6,R4 ; get set to return next page
5A 56 D1 01A6 660 CMPL R6,R10 ; all done?
E8 12 01A9 661 BNEQ 40$ ; loop if not
01AB 662 :
01AB 663 :
01AB 664 : Zero ifi and ifab table pointer
01AB 665 :
01AB 666 :
FE52' 30 01AB 667 BSBW RMSZAPIFI ; zero ifi and ifab table entry
50 8E D0 01AE 668 POPL RO ; return status to caller
05 05 01B1 669 RSB ; return to caller
01B2 670 :
01B2 671 :
01B2 672 : Attempted to return an ifab-related page having some

```

01B2 673 : non-deallocated block(s) in it  
01B2 674 :  
01B2 675 :  
01B2 676 ERRBUG: RMSTBUG FTL\$\_DEALLERR  
01B9 677

```
0189 679 .SBTTL RMSSPL_SCF - $CLOSE routine for spool/submit options
0189 680
0189 681 :++
0189 682 : RMSSPL_SCF - specific close code for the SPL and SCF FOP options
0189 683 :
0189 684 : This routine performs the spl and scf options, and if set (either on
0189 685 : $open/$create or on $close), sends a message to the job controller to
0189 686 : queue the file to the sys$print or sys$batch queues respectively.
0189 687 : If both spl and scf are set, scf takes precedence. The dlt fop sub-option
0189 688 : is passed on to the job controller.
0189 689 :
0189 690 : The overall flow of the routine is as follows:
0189 691 :
0189 692 : 1. build the dvi, did and fid fields from the fwa onto the stack.
0189 693 : 2. allocate a buffer on the stack to build the item list for the job
0189 694 : controller.
0189 695 : 3. fill the queue name in the appropriate item; the job controller will
0189 696 : translate either SYSSPRINT or SYSSBATCH.
0189 697 : 4. point an item at the dvi, did, fid copy on th stack
0189 698 : 5. fill in the delete option if required.
0189 699 : 6. send the message to the job controller with a function code of
0189 700 : sjc$_queue.
0189 701 :
0189 702 : Calling sequence:
0189 703 :
0189 704 : E W RMSSPL_SCF
0189 705 :
0189 706 : Input Parameters:
0189 707 :
0189 708 : r10 ifab address
0189 709 : r9 ifab address
0189 710 : r8 fab address
0189 711 :
0189 712 : Implicit Inputs:
0189 713 :
0189 714 : the contents of the ifab (especially ifb$_v_spc, scf, and dlt)
0189 715 : the contents of the fwa (especially fwa$_q_shrfil and fwa$_t_fibbuf)
0189 716 :
0189 717 : Output Parameters:
0189 718 :
0189 719 : r1-r7 destroyed
0189 720 : r0 status code
0189 721 :
0189 722 : Implicit Outputs:
0189 723 :
0189 724 : fab$_l_stv is set to subsidiary error code on an error.
0189 725 :
0189 726 : Completion Codes:
0189 727 :
0189 728 : standard rms, in particular, spl.
0189 729 :
0189 730 : Side Effects:
0189 731 :
0189 732 : none
0189 733 :
0189 734 : Note: no need to check that PPF_IMAGE not set since can't get here if so.
0189 735 :
```

```

01B9 736 : job controller item list is currently 10 longwords
01B9 737 : long - for three items and terminator
01B9 738 :
01B9 739 :--
01B9 740 :
01B9 741 :
01B9 742 : Own Storage:
01B9 743 :
01B9 744 :
54 4E 49 52 50 24 53 59 53 01B9 745 SYSPRINT: .ASCII /SYSSPRINT/
00000009 01C2 746 SYSPRINT_LEN = .-SYSPRINT
01C2 747
48 43 54 41 42 24 53 59 53 01C2 748 SYSBATCH: .ASCII /SYSSBATCH/
00000009 01CB 749 SYSBATCH_LEN = .-SYSBATCH
01CB 750
0000001C 01CB 751 ID_SIZE = 16 + 6 + 6 ; DVI + DID + FID
01CB 752
01CB 753 RMSPL_SCF::
01CB 754
01CB 755 :
01CB 756 : Allocate DVI_DID_FID buffer on stack and fill it in.
01CB 757 :
01CB 758
01CB 759 SUBL2 #<ID_SIZE>,SP
01CE 760 MOVL IFB$C_FWA_PTR(R9),R7 ; get FWA ptr
50 57 38 A9 D0 01D2 761 MOVQ FWA$Q_SHRFILE(R7),R0 ; get DVI description
01D7 762 MOVB R0,(SP) ; make it ASCII
01DA 763 MOVCS R0,(R1),#0,#15,1(SP) ; copy dvi and fill to 16 bytes
01E1 764
01E1 765 ASSUME FIB$W_FID+6 EQ FIB$W_DID
01E1 766
63 01F8 C7 0C 28 01E1 767 MOVCS #<6+6>,FWA$T_FIBBUF+FIB$W_FID(R7),(R3)
01E7 768 ; copy did and fid
52 5E D0 01E7 769 MOVL SP,R2 ; remember addr of dvi_did_fid blk
01EA 770 ; R3 points to cleaned-off-SP
01EA 771
01EA 772 :
01EA 773 : Build the job controller item list
01EA 774 :
01EA 775
01EA 776 CLRL -(SP) ; end-of-list flag
08 69 7E D4 01EA 777 BBC #IFB$V_DLT,(R9),10$ ; branch if no delete requested
01EC 778 CLRQ -(SP) ; no retlen or addr
01F0 779 PUSHL #SJCS_DELETE_FILE@16 ; delete flag
00180000 8F DD 01F2 780 ; and zero buflen
01F8 781
01F8 782 :
01F8 783 : Now, point to the dvi_did_fid block to identify the file
01F8 784 :
01F8 785
01F8 786 10$: CLRL -(SP) ; no retlen
0027001C 7E D4 01FA 787 PUSHL R2 ; addr of block
52 DD 01FC 788 PUSHL #<<SJCS_FILE_IDENTIFICATION@16>+ID_SIZE> ; item type
8F DD 0202 789 ; and fill in size of identification
0202 790
0202 791 :
0202 792 : Fill in the initial item, which indicates to the job controller that a file is

```

```

0202 793 ; to be queued to either SYSSPRINT or SYSSBATCH.
0202 794 ;
0202 795 ;
0202 796 ASSUME SYSPRINT_LEN EQ SYSBATCH_LEN
0202 797
05 69 7E D4 0202 798 CLR - (SP) ; no retlen
AE AF E0 0204 799 BBS #IFB$V_SCF,(R9),20$ ; branch if submit command file
03 11 DF 0208 800 PUSHAL B^SYSPRINT ; point to queue name string
B2 AF DF 020B 801 BRB 30$ ; do next item
00860009 8F DD 020D 802 20$: PUSHAL B^SYSBATCH ; point to queue name
8F DD 0210 803 30$: PUSHL #<<SJC$_QUEUE@16>+SYSPRINT_LEN ; indicate function
0216 804 ; and fill in length of queue name
51 5E D0 0216 805 MOVL SP,R1 ; addr of itemlist
0219 807
0219 808 ;
0219 809 ; Call the job controller.
0219 810 ;
0219 811 ;
0219 812 $SNDJBC_S -
0219 813 _EFN = #IMP$C_ASYQIOEFN,- ; throw-away event flag
0219 814 FUNC = #SJC$_ENTER_FILE,- ; function
0219 815 ITMLST = (R1) ; item list
SE 53 D0 022D 816
09 50 E8 0230 817 MOVL R3,SP ; clean stack
OC A8 50 D0 0233 818 BLBS R0,40$ ; exit on error
D0 0237 819 MOVL R0,FAB$_STV(R8) ; save jobctl status
05 023C 820 RMSERR SPL ; and report error
023D 821 40$: RSB
023D 822

```

```

023D 824 .SBTTL RMS$RELEASALL, Release all BDB's
023D 825
023D 826 :++
023D 827 : RMS$RELEASALL - release bdb's and buffers
023D 828 :
023D 829 : Subroutine to release all bdb's and their associated buffers.
023D 830 : Assumes dirty buffers will not be found.
023D 831 : Also return all BLB's.
023D 832 :
023D 833 : inputs:
023D 834 : r11 impure area address
023D 835 : r9 ifab address
023D 836 : r8 fab address
023D 837 :
023D 838 : outputs:
023D 839 : r10 ifab address
023D 840 : r0-r6 destroyed
023D 841 :--
023D 842 :
023D 843 : return all buffers and bdb's
023D 844 :
023D 845 :--
023D 846
023D 847 RMS$RELEASALL::
56 5A 59 D0 023D 848 MOVL R9,R10 ; make sure r10 = ifab addr
56 40 AA DE 0240 849 MOVAL IFBSL_BDB_FLNK(R10),R6 ; get bdb list head
56 54 66 D0 0244 850 10$: MOVL (R6),R4 ; get 1st bdb in list
56 54 54 D1 0247 851 CMPL R4,R6 ; back at list head?
51 0A A4 01 E0 024A 852 BEQL 30$ ; branch if yes - all done
OC A4 B5 0251 853 BBS #BDB$V_DRT, BDB$B_FLGS(R4), DRTBUG ; Don't expect to find dirt.
OC A4 03 12 0254 854 TSTW BDB$W_USERS(R4) ; use count nonzero?
OC A4 B6 0256 855 BNEQ 20$ ; no, go release bdb
FDA4' 30 0259 856 20$: INCW BDB$W_USERS(R4) ; make it look accessed
025C 857 BSBW RMSRLNERR ; go release it and free buffer.
025C 858
025C 859 ASSUME BDB$B_BID EQ GBP$B_BID
025C 860 ASSUME <BDB$C_BID&1> EQ 0
025C 861 ASSUME <GBP$C_BID&1> EQ 1
025C 862
05 08 A4 E8 025C 863 BLBS BDB$B_BID(R4), 25$ ; br if gbp.
FD9D' 30 0260 864 BSBW RMSRETDBD ; return the bdb
DF 11 0263 865 BRB 10$ ; keep going until all gone.
FD98' 30 0265 866 25$: BSBW RMSRETGBP ; return gbp.
DA 11 0268 867 BRB 10$ ; keep going
026A 868
026A 869 ASSUME IFBSW_AVGBP EQ <IFBSW_AVLCL + 2>
026A 870
56 2C 0084 CA D4 026A 871 30$: CLRL IFBSW_AVLCL(R10) ; Note all buffers gone.
56 6A 33 E0 026E 872 BBS #IFBSV_NORECLK, (R10), RA_EX ; All done if no locking.
56 0098 CA DE 0272 873 MOVAL IFBSL_BLBFLNK(R10), R6 ; Get list head for BLB's.
56 54 66 D0 0277 874 40$: MOVL (R6), R4 ; Get next BLB.
56 54 54 D1 027A 875 CMPL R4, R6 ; Back at list head?
24 A4 12 13 027D 876 BEQL CHKGBL ; All done then.
08 13 027F 877 TSTL BLB$LOCK_ID(R4) ; This one still locked?
FD79' 30 0282 878 BEQL 45$ ; EQL no lock, so just return it.
54 009C CA D0 0284 879 BSBW RMSRLNER1 ; Release the lock first.
0287 880 MOVL IFBSL_BLBFLNK(R10), R4 ; Recover BLB address.

```

```
FD71' 30 028C 881 45$: BSBW RMSRETLB ; Return the BLB.  
E6 11 028F 882 BRB 40$ ; Go get next one.  
0088 CA D5 0291 883  
07 13 0291 884 CHKGBL: TSTL IFB$$_GBH_PTR(R10) ; Are global buffers present?  
FD66' 30 0295 885 BEQL RA_EX ; No, we are done.  
OD 10 0297 886 BSBW RMSRAISE_GBS_LOCK ; Get EX lock on global section.  
6E 10 029A 887 BSBB RMSRELEASE_GBL_BUFFERS ; Release and cleanup global buffers.  
FD5F' 30 029C 888 BSBB RMSUNMAP_GBL ; Disassociate from section.  
05 029E 889 RA_EX: BSBW RMSRLS_GBSB ; Deallocate the GBSB if any (also release l  
02A1 890 RSB  
02A2 891  
02A2 892 DRTBUG: RMSTBUG FTL$_RLSDRT ; A dirty buffer has been left  
02A9 893 ; behind by someone.  
02A9 894
```



```

02A9 896 :++
02A9 897 :
02A9 898 : RM$RELEASE_GBL_BUFFERS
02A9 899 :
02A9 900 : This routine decrements the access count for the global buffer section.
02A9 901 : If the access count goes to zero, then all cached buffers are released by
02A9 902 : dequeuing the system lock for each buffer, and the system file lock is
02A9 903 : also released.
02A9 904 :
02A9 905 : As a part of releasing the system lock on a buffer, we also give back the
02A9 906 : quota used when that lock was first converted. Notice that if this routine
02A9 907 : is called as part of a $CLOSE operation then there exists a non-closeable
02A9 908 : hole in which we can give back the quota and have the process deleted before
02A9 909 : dequeuing the lock. This will have the effect of increasing the global
02A9 910 : buffer quota by one. The reverse can occur during conversion to the system
02A9 911 : lock in RMORELEAS.
02A9 912 :
02A9 913 : Note: This routine assumes that an EX lock has already been taken on the
02A9 914 : global section.
02A9 915 :
02A9 916 : Inputs:
02A9 917 :
02A9 918 : R10 - Address of ifab.
02A9 919 :
02A9 920 : Outputs:
02A9 921 :
02A9 922 : none
02A9 923 :
02A9 924 :--
02A9 925 RM$RELEASE_GBL_BUFFERS::
54 7E 53 7D 02A9 926 MOVQ R3, -(SP) ; Save registers.
53 0088 CA D0 02AC 927 MOVL IFB$L_GBH_PTR(R10), R4 ; Get address of global section in R4.
53 7C AA D0 02B1 928 MOVL IFB$L_GBSB_PTR(R10), R3 ; Get gbsb address in R3.
07 0B A3 E0 02B5 929 BBS #GBSBSM_NOTACCESSED, - ; If set then access count is already decrem
1C A4 D7 02B7 930 GBSBSB_FLAGS(R3), 5$ ; go check access count (we are in last cha
01 88 02BD 932 BISB2 #GBSBSM_NOTACCESSED, - ; Set bit in GBSB saying accessor count
0B A3 02BF 933 GBSBSB_FLAGS(R3) ; has been decremented (for last chance)
1C A4 D5 02C1 934 5$: TSTL GBH$L_OSECNT(R4) ; Test accessor count.
42 12 02C4 935 BNEQ DONE ; Exit if not last accessor.
02C6 936
53 54 D0 02C6 937 MOVL R4, R3 ; Move address of section into R3.
53 63 C0 02C9 938 10$: ADDL2 (R3), R3 ; Get address to next GBD element.
54 53 D1 02CC 939 CMPL R3, R4 ; Are we back at queue header?
14 1F 13 02CF 940 BEQL RLS_FILE_LOCK ; Yes, go release system file lock.
14 A3 D5 02D1 941 TSTL GBD$L_LOCK_ID(R3) ; Is this buffer cached?
00000000'9F F3 13 02D4 942 BEQL 10$ ; No, go to next GBD.
14 A3 D4 02D6 943 ADAWI #1, @#RMS$GW_GBLBUFQUO ; Give the buffer back to the quota ctr
D9 11 02DD 944 $DEQ_S LKID = GBD$C_LOCK_ID(R3); DEQ the system lock on buffer.
02EB 945 CLRL GBD$L_LOCK_ID(R3) ; Mark this GBD as gone.
02EE 946 BRB 10$ ; Go to next GBD.
02F0 947
02F0 948 RLS_FILE_LOCK:
02F0 949 $DEQ_S LKID = GBH$L_LOCK_ID(R4); $DEQ system file lock.
02FE 950
54 7C AA D0 02FE 951 MOVL IFB$L_GBSB_PTR(R10), R4 ; Get address of GBSB.
38 A4 D4 0302 952 CLRL GBSBS$C_SIZE(R4) ; Zero all fields in lock value

```

34	A4	B4	0305	953	CLRW	GBSB\$W_GBC(R4)	; block.
			0308	954			
53	8E	7D	0308	955	DONE: MOVQ	(SP)+,R3	; Restore registers.
		05	030B	956	RSB		
			030C	957			

```

030C 959 :++
030C 960 :
030C 961 : RMSUNMAP_GBL
030C 962 :
030C 963 : This routine deletes the specified address range for the purpose of
030C 964 : un-mapping from a global section that has been used for i/o buffers.
030C 965 :
030C 966 : Note: This routine assume an EX lock is already held on the global section.
030C 967 :
030C 968 : Inputs:
030C 969 :
030C 970 : R0 - start address of range. (alt. entry pt.)
030C 971 : R1 - end address of range. (alt. entry pt.)
030C 972 : R10 - ifab address
030C 973 :
030C 974 : Outputs:
030C 975 :
030C 976 : Destroys R0 - R2.
030C 977 :
030C 978 :--
030C 979 RMSUNMAP_GBL::
51 50 0088 CA D0 030C 980      MOVL   IFBSL_GBH_PTR(R10),R0 ; Put address of global section in R0.
51 50 10 A0 C1 0311 981      ADDL3  GBHSL_GS_SIZE(R0),R0,R1 ; End addr of sec + 1
51 50 51 D7 0316 982      DECL   R1 ; End addr of section.
0318 983
0318 984 RMSUNMAP_GBL_ALT::
7E 50 7D 0318 985      MOVQ   R0, -(SP) ; Save range on stack.
52 5E D0 031B 986      MOVL   SP, R2 ; Remember that address.
50 8E 7D 031E 987      $DELTVA_S INADR=(R2) ; Delete the VA.
05 032B 988      MOVQ   -(SP)+, R0 ; Return address array.
032E 989      RSB ; And return.
032F 990
032F 991      .END

```

RMSOCLOSE  
Symbol table

DISPATCH FOR CLOSE OPERATION

M 6

16-SEP-1984 01:11:09 VAX/VMS Macro V04-00  
5-SEP-1984 16:24:38 [RMS.SRC]RMSOCLOSE.MAR;1

Page 25  
(14)

RM  
VO

```

$$PSECT_EP = 00000000
$$RMSTEST = 0000001A
$$RMS_PBUGCHK = 00000010
$$RMS_TBUGCHK = 00000008
$$RMS_UMODE = 00000004
$$T1 = 00000001
BDBSB_BID = 00000008
BDBSB_FLGS = 0000000A
BDBSC_BID = 0000000C
BDBSV_DRT = 00000001
BDBSW_USERS = 0000000C
BKP = 00000020
BLBSL_LOCK_ID = 00000024
BLDFIB = 0000008E R 01
CHKGBL = 00000291 R 01
CLNJNL = 00000153 R 01
CLSCUT = 000000BE R 01
CLSDLT = 00000073 R 01
CLSMAGTAP = 00000123 R 01
CLSMLBX = 0000013E R 01
CLS_EX = 000000D3 R 01
DEACCESS = 000000F5 R 01
DEASSIGN = 00000148 R 01
DEASSIGN_ALT = 00000145 R 01
DELETE = 0000009F R 01
DEVSV_FOR = 00000018
DEVSV_MBX = 00000014
DEVSV_NET = 0000000D
DEVSV_RND = 0000001C
DEVSV_SQD = 00000005
DONE = 00000308 R 01
DRTBUG = 000002A2 R 01
ERRBUG = 000001B2 R 01
FABSC_BLN = 00000050
FABSC_SEQ = 00000000
FABSL_FOP = 00000004
FABSL_STV = 0000000C
FABSV_DLT = 0000000F
FABSV_DMO = 0000000C
FABSV_RWC = 0000000B
FABSV_SCF = 0000000E
FABSV_SPL = 0000000D
FABSW_IFI = 00000002
FIBSV_FINDFID = 0000000B
FIBSW_DID = 0000000A
FIBSW_FID = 00000004
FIBSW_NMCTL = 00000014
FOP = 00000020
FTLS_DEALLERR = 000000FF
FTLS_RLSDRT = 000000D8
FWASQ_FIB = 00000010
FWASQ_SHRFIL = 00000190
FWAST_FIBBUF = 000001F4
GBDSL_LOCK_ID = 00000014
GBHSL_GS_SIZE = 00000010
GBHSL_LOCK_ID = 00000014
GBHSL_USECNT = 0000001C

```

```

GBPBSB_BID = 00000008
GBPBSB_BID = 00000015
GBSBSB_FLAGS = 0000000B
GBSBSL_GS_SIZE = 00000038
GBSBSM_NOTACCESSED = 00000001
GBSBSW_GBC = 00000034
ID_SIZE = 0000001C
IFBSB_JNLFLG2 = 000000A2
IFBSB_MODE = 0000000A
IFBSB_ORGCASE = 00000023
IFBSC_IDX = 00000002
IFBSL_ARGLST = 00000018
IFBSL_ASADDR = 00000014
IFBSL_BDB_FLNK = 00000040
IFBSL_BLBBLNK = 0000009C
IFBSL_BLBFLNK = 00000098
IFBSL_FWA_PTR = 00000038
IFBSL_GBH_PTR = 00000088
IFBSL_GBSB_PTR = 0000007C
IFBSL_IOS = 0000000C
IFBSL_IRAB_LNK = 0000001C
IFBSL_NWA_PTR = 0000003C
IFBSL_PRIM_DEV = 00000000
IFBSV_ACCESSED = 00000025
IFBSV_DLT = 0000002B
IFBSV_DMO = 00000028
IFBSV_JNL = 00000001
IFBSV_NORECLK = 00000033
IFBSV_PPF_IMAGE = 00000022
IFBSV_RWC = 00000027
IFBSV_SCF = 0000002A
IFBSV_SPL = 00000029
IFBSV_TMP = 00000035
IFBSV_WRTACC = 00000030
IFBSW_AVGBPB = 00000086
IFBSW_AVLCL = 00000084
IFBSW_CHNL = 00000020
IMPSC_ASYQIOEFN = 0000001F
IOSM_DELETE = 00000100
IOS_DELETE = 00000035
IRBSB_MODE = 0000000A
IRBSL_BKPBITS = 00000004
IRBSL_IRAB_LNK = 0000001C
IRBSV_ASYNC = 00000023
IRBSV_BUSY = 00000020
IRBSV_PPF_IMAGE = 00000022
MASK = 000001FF
NT$NWA_FREE = ***** X 01
NXTDISC = 0000000D R 01
NXTIRAB = 00000048 R 01
NXTRTN = 00000036 R 01
PIOSA_TRACE = ***** X 01
RABSC_BLN = 00000044
RABSL_STV = 0000000C
RABSW_ISI = 00000002
RA_EX = 0000029E R 01
RETIFB = 00000162 R 01

```

RMSOCLOSE  
Symbol table

DISPATCH FOR CLOSE OPERATION

N 6

16-SEP-1984 01:11:09  
5-SEP-1984 16:24:38

VAX/VMS Macro V04-00  
[RMS.SRC]RMSOCLOSE.MAR;1

Page 26  
(14)

RM  
VO

RLS_FILE_LOCK	000002F0	R	01
RMSBUG	*****	X	01
RMSCLEANUP	000000DE	RG	01
RMSCLOSE3	*****	X	01
RMSCLSCU	000000BC	RG	01
RMSDEACCESS	*****	X	01
RMSDEAJNL	*****	X	01
RMSDEALLOCATE_FWA	*****	X	01
RMSDISCOMMONSOC	*****	X	01
RMSDISCONNECT1	*****	X	01
RMSDISCONNECT3	*****	X	01
RMSEX RMS	*****	X	01
RMSEX_NOSTR	*****	X	01
RMSFCPFNC	*****	X	01
RMSFSET	*****	X	01
RMSMAPJNL	*****	X	01
RMSRAISE_GBS_LOCK	*****	X	01
RMSRELEASALL	0000023D	RG	01
RMSRELEASE_GBL_BUFFERS	000002A9	RG	01
RMSRET1PAG	*****	X	01
RMSRETBDB	*****	X	01
RMSRETLB	*****	X	01
RMSRETLK	*****	X	01
RMSRETGBP	*****	X	01
RMSRETIFB	000000D6	RG	01
RMSREWIND_MT	*****	X	01
RMSRLNERT	*****	X	01
RMSRLNERR	*****	X	01
RMSRLS_GBSB	*****	X	01
RMSRLS_SFSB	*****	X	01
RMS SPL_SCF	000001CB	RG	01
RMSUNMAP_GBL	0000030C	RG	01
RMSUNMAP_GBL_ALT	00000318	RG	01
RMSWRITEOF	*****	X	01
RMSWTTAPMARK	*****	X	01
RMSZAPIFI	*****	X	01
RMS\$CLOSE	= FFFFFFFE	RG	01
RMS\$GW_GBLBUFQUO	*****	X	01
RMS\$MRD	= 0001C032		
RMS\$SPL	= 0001C042		
SJCS_DELETE_FILE	= 00000018		
SJCS_ENTER_FILE	= 00000013		
SJCS_FILE_IDENTIFICATION	= 00000027		
SJCS_QUEUE	= 00000086		
SYSSDASSGN	*****	GX	01
SYSSDELTVA	*****	GX	01
SYSSDEQ	*****	GX	01
SYSSNDJBC	*****	GX	01
SYSBATCH	000001C2	R	01
SYSBATCH_LEN	= 00000009		
SYSPRINT	000001B9	R	01
SYSPRINT_LEN	= 00000009		
TPT\$L_CLOSE	*****	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
RMSRMS	0000032F ( 815.)	01 ( 1.)	PIC USR CON REL GBL NOSHR EXE RD NOWRT NOVEC BYTE
SABSS	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	31	00:00:00.09	00:00:00.95
Command processing	110	00:00:00.71	00:00:04.85
Pass 1	593	00:00:25.51	00:01:00.17
Symbol table sort	0	00:00:03.95	00:00:05.79
Pass 2	173	00:00:04.85	00:00:09.28
Symbol table output	21	00:00:00.22	00:00:00.74
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	932	00:00:35.36	00:01:21.91

The working set limit was 1950 pages.  
140168 bytes (274 pages) of virtual memory were used to buffer the intermediate code.  
There were 140 pages of symbol table space allocated to hold 2678 non-local and 29 local symbols.  
991 source lines were read in Pass 1, producing 17 object records in Pass 2.  
48 pages of virtual memory were used to define 47 macros.

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

Macro library name	Macros defined
_\$255\$DUA28:[RMS.OBJ]RMS.MLB;1	24
-\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	2
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2	17
TOTALS (all libraries)	43

2890 GETS were required to define 43 macros.

There were no errors, warnings or information messages.

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

RMS  
Syn  
\$\$  
\$\$F  
\$\$F  
\$\$F  
\$\$F  
ASE  
ASE  
ASE  
ASE  
ASE  
ASE  
ASE  
ASE  
ASE  
CTL  
ERF  
ERF  
ERF  
ERF  
ERF  
ERF  
ERF  
ERF  
EX  
FAE  
FAE  
FAE  
FAE  
FAE  
FAE  
FAE  
FTL  
IFE  
IFE  
IFE  
IFE  
IFE  
IFE  
IFE  
IFE  
IFE  
IMI  
IMI  
IN  
IR/  
IRI  
IRI  
IRI  
IRI



0329 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

