

```

RRRRRRRRRRRRRRR      MMM      MMM      SSSSSSSSSSSSSS
RRRRRRRRRRRRRRR      MMM      MMM      SSSSSSSSSSSSSS
RRRRRRRRRRRRRRR      MMM      MMM      SSSSSSSSSSSSSS
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
RRRRRRRRRRRRRRR      MMM      MMM      SSSSSSSSSS
RRRRRRRRRRRRRRR      MMM      MMM      SSSSSSSSSS
RRRRRRRRRRRRRRR      MMM      MMM      SSSSSSSSSS
RRR      RRR      MMM      MMM      MMM      SSS
RRR      RRR      MMM      MMM      MMM      SSS
RRR      RRR      MMM      MMM      MMM      SSS
RRR      RRR      MMM      MMM      MMM      SSS
RRR      RRR      MMM      MMM      MMM      SSS
RRR      RRR      MMM      MMM      SSSSSSSSSSSSSS
RRR      RRR      MMM      MMM      SSSSSSSSSSSSSS
RRR      RRR      MMM      MMM      SSSSSSSSSSSSSS

```

53

Syn

NTS

NTS
NTS

NTS

NTS
NTS

N13

NTS

NTS
NTS

NTS

NTS
NTSNTS
NTS

NTS

NTS
NTS

NTS

NTS
NTS

NTS

NTS
NTS

NTS

NTS
NTSNTS
NTS

NTS

NTS

NTA

NTS
NTSNTS
NTS

NTS

NTS
NTS

NTS

10

10

NTC

NT
NT

NTS

NT
NT

NT

PI

10

100

100

10

1

11. *Journal of the American Medical Association*, 273, 1995, 1037-1041.

1

```
RRRRRRRR      MM      MM      SSSSSSSS      000000      CCCCCCCC      000000      NN      NN      NN      NN
RRRRRRRR      MM      MM      SSSSSSSS      000000      CCCCCCCC      000000      NN      NN      NN      NN
RR      RR      MMMM      MMMM      SS      00      00      CC      00      00      NN      NN      NN      NN
RR      RR      MMMM      MMMM      SS      00      00      CC      00      00      NN      NN      NN      NN
RR      RR      MM      MM      SS      00      0000      CC      00      00      NNNN      NN      NNNN      NN
RR      RR      MM      MM      SS      00      0000      CC      00      00      NNNN      NN      NNNN      NN
RRRRRRRR      MM      MM      SSSSSS      00      00      00      CC      00      00      NN      NN      NN      NN
RRRRRRRR      MM      MM      SSSSSS      00      00      00      CC      00      00      NN      NN      NN      NN
RR      RR      MM      MM      SS      0000      00      CC      00      00      NN      NN      NN      NN
RR      RR      MM      MM      SS      0000      00      CC      00      00      NN      NN      NN      NN
RR      RR      MM      MM      SS      00      00      CC      00      00      NN      NN      NN      NN
RR      RR      MM      MM      SSSSSSSS      000000      CCCCCCCC      000000      NN      NN      NN      NN
RR      RR      MM      MM      SSSSSSSS      000000      CCCCCCCC      000000      NN      NN      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
LLLLLLLLLL      IIIIII      SSSSSSSS
LLLLLLLLLL      IIIIII      SSSSSSSS
```

RMS
Pse

PSE

RMS
\$AE

Pha

Int
Com
Pas
Syn
Pas
Syn
Pse
Cro
Ass

The
603
The
440
30

Mac

-S
-S
-S
TOT
129
The
MA

(2) 89
(3) 129

DECLARATIONS
RMSSCONNECT - \$CONNECT ROUTINE

```
0000 1          $BEGIN RMSOCONN,000,RMSRMS,<DISPATCH ROUTINE FOR CONNECT>
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 27
0000 28 ++
0000 29 Facility: rms32
0000 30
0000 31 Abstract:
0000 32         this routine is the highest level control
0000 33         routine to perform the $connect function.
0000 34
0000 35 Environment:
0000 36         star processor running starlet exec.
0000 37
0000 38 Author: L F LAVERDURE,          Creation Date: 5-JAN-1977
0000 39
0000 40 Modified By:
0000 41
0000 42 V03-009 KPL0001          Peter Lieberwirth          20-Jun-1983
0000 43         Change some references to JNLFLG to JNLFLG2.
0000 44
0000 45 V03-008 TSK0001          Tamar Krichevsky           12-Jun-1983
0000 46         Fix broken branch to journaling psect.
0000 47
0000 48 V03-007 RAS0151          Ron Schaefer              29-Apr-1983
0000 49         Fix broken branches to RM$CONNECT_BIO and RM$CONNECTx.
0000 50
0000 51 V03-006 KBT0363          Keith B. Thompson          11-Oct-1982
0000 52         Allocate the asb as a seperate structure
0000 53
0000 54 V03-005 LJA0026          Laurie J. Anderson         11-Oct-1982
0000 55         No longer need R0 passed to RM$GTSLT, makes the call cleaner
0000 56
0000 57 V03-004 JWH0002          Jeffrey W. Horn            10-Sep-1982
```



```
0000 58 : Fill IRB$ IDENT with a unique identifier for each
0000 59 : IRB connected throughout the life of a process.
0000 60 :
0000 61 : V03-003 LJA0019 Laurie J. Anderson 03-Sep-1982
0000 62 : Check to see if this is a restart connect operation by looking
0000 63 : for a context XAB (XABCXR) and whether the restart option is set
0000 64 :
0000 65 : V03-002 KBT0301 Keith B. Thompson 28-Aug-1982
0000 66 : Reorganize psects and rename entry point to single '$'
0000 67 :
0000 68 : V03-001 JWH0001 Jeffrey W. Horn 18-May-1982
0000 69 : Add call to RMSCONJNL if journaling.
0000 70 :
0000 71 : V02-017 KDM0037 Kathleen D. Morse 12-Feb-1980
0000 72 : Change non-kernel references to SCH$GL_CURPCB to
0000 73 : CTL$GL_PCB instead.
0000 74 :
0000 75 : V016 REFORMAT D M WALP 25-JUL-1980
0000 76 :
0000 77 : V015 RAN0003 R A NEWELL 20-DEC-1978 1:30
0000 78 : fixed pid bug
0000 79 :
0000 80 : V014 RAN0002 R A NEWELL 1-SEP-1978 12:00
0000 81 : rms32 isam modification. redefinition of entry point to
0000 82 : resolve out of range branches.
0000 83 :
0000 84 :
0000 85 :
0000 86 : --
0000 87 :
```

```
0000 89      .SBTTL  DECLARATIONS
0000 90
0000 91  :
0000 92  : Include Files:
0000 93  :
0000 94  :
0000 95  :
0000 96  : Macros:
0000 97  :
0000 98
0000 99      $FABDEF
0000 100     $RABDEF
0000 101     $IMPDEF
0000 102     $PIODEF
0000 103     $IFBDEF
0000 104     $IRBDEF
0000 105     $PCBDEF
0000 106     $ASBDEF
0000 107     $RMSDEF
0000 108
0000 109  :
0000 110  : Equated Symbols:
0000 111  :
0000 112
00000020 0000 113      ROP=RAB$L_ROP*8          ; bit offset to rop field
0000 114
0000 115  :
0000 116  : Own Storage:
0000 117  :
0000 118
0000 119  IRAB_SIZE TBL:
1B 0000 120      .BYTE  <IRB$C_BLN_SEQ>/4
19 0001 121      .BYTE  <IRB$C_BLN_REL>/4
31 0002 122      .BYTE  <IRB$C_BLN_IDX>/4
0003 123  ASB_SIZE TBL:
002F 0003 124      .WORD  <ASB$C_BLN_SEQ>/4
0030 0005 125      .WORD  <ASB$C_BLN_REL>/4
0080 0007 126      .WORD  <ASB$C_BLN_IDX>/4
0009 127
```



```
0009 129      .SBTTL  RMS$CONNECT - $CONNECT ROUTINE
0009 130
0009 131      :++
0009 132      :  RMS$CONNECT
0009 133      :
0009 134      :  RMS$CONNECT-
0009 135      :
0009 136      :  this routine performs the highest level $connect
0009 137      :  processing.  its functions include:
0009 138      :
0009 139      :  1. allocate and init an irab along with its asb
0009 140      :  2. perform common setup
0009 141      :  3. dipatch to organization-dependent routine
0009 142      :
0009 143      :
0009 144      :  Calling sequence:
0009 145      :
0009 146      :  entered from exec as a result of user's calling sys$connect
0009 147      :  (e.g., by using the $connect macro).
0009 148      :
0009 149      :  Input Parameters:
0009 150      :
0009 151      :  ap      user's argument list address
0009 152      :
0009 153      :  Implicit Inputs:
0009 154      :
0009 155      :  the contents of the rab and the ifi field of the fab.
0009 156      :
0009 157      :  Output Parameters:
0009 158      :
0009 159      :  r0      status code
0009 160      :  r1      destroyed
0009 161      :
0009 162      :  Implicit Outputs:
0009 163      :
0009 164      :  various fields of the rab are filled in to reflect
0009 165      :  the status of the connected stream.  (see rms functional
0009 166      :  spec for a complete list.)
0009 167      :
0009 168      :  an irab is initialized to reflect the connected stream.
0009 169      :
0009 170      :  a completion ast is queued if so specified by the user.
0009 171      :
0009 172      :  Completion Codes:
0009 173      :
0009 174      :  standard rms (see functional spec for list).
0009 175      :
0009 176      :  Side Effects:
0009 177      :
0009 178      :  none
0009 179      :
0009 180      :  --
0009 181      :
0009 182
```

```
0009 184 $ENTRY RMS$CONNECT
0009 185 $TSTPT CONNECT
000F 186
000F 187
000F 188
000F 189
000F 190
000F 191
000F 192
FFEE' 30 000F 193 BSBW RMS$RABCHK ; return from setup only if aok
0012 194
0012 195
0012 196
0012 197
0012 198
5A 3C A8 D0 0012 198 MOVL RAB$FAB(R8),R10 ; get fab addr
0016 199 IFNORD #FAB$C_BLN,(R10),ERRFAB ; must be able to access
001E 200 ASSUME FAB$B_BID EQ 0
03 6A 91 001E 201 CMPB (R10),#FAB$C_BID ; is it a fab?
79 12 0021 202 BNEQ ERRFAB ; branch if not
50 8F 01 AA 91 0023 203 CMPB FAB$B_BLN(R10),#FAB$C_BLN ; long enough?
0028 204
0028 205 BLSS ERRBLN
59 02 AA 3C 002A 206 MOVZWL FAB$W_IFI(R10),R9 ; get ifi
50 06 D0 002E 207 MOVL #IMP$C_IFABTBL/4,R0 ; ifab table offset/4
FFCC' 30 0031 208 BSBW RMS$GTIADR ; get ifab addr (r9)
62 13 0034 209 BEQL ERRIFI ; error if none
00000008 0036 210 .IF NE $$RMS$TEST&$$RMS_TBUGCHK
0036 211 ASSUME IFB$B_BID EQ IFB$B_BLN-1
08 A9 B1 0036 212 CMPW IFB$B_BID(R9),-
2E0B 8F 0039 213 #IFB$C_BID+<<IFB$C_BLN/4>*256>
70 12 003C 214 BNEQ ERRBUG ; branch if not a valid ifab
003E 215 .ENDC
52 69 20 E0 003E 216 BBS #IFB$V_BUSY,(R9),ERRACT ; branch if ifab already in use
1E E0 0042 217 BBS #FAB$V_PPF_IND+<FAB$W_IFI*8>,-
6F 6A 0044 218 (R10),PPF_IND ; branch if indirect ppf
5A 59 D0 0046 219 MOVL R9,R10 ; save ifab addr in r10
0049 220
0049 221
0049 222
0049 223
0049 224
55 23 AA 9A 0049 225 MOVZBL IFB$B_ORGCASE(R10),R5 ; get file org index
004D 226 ASSUME IFB$C_SEQ EQ 0
0D 13 004D 227 BEQL 10$ ; branch if sequential
004F 228 ; (no multi streaming)
02 55 91 004F 229 CMPB R5,#IFB$C_MAXORG ; is this a known file org?
04 15 0052 230 BLEQ 5$ ; branch if yes
55 D4 0054 231 CLRL R5 ; no - allocate seq. len irab
04 11 0056 232 BRB 10$ ; check no multi-streams
05 6A 31 E0 0058 233 5$: BBS #IFB$V_MSE,(R10),20$ ; omit check if multi-streams enabled
1C AA D5 005C 234 10$: TSTL IFB$L_IRAB_LNK(R10) ; already got an irab?
43 12 005F 235 BNEQ ERRCCR ; branch if yes (error)
0061 236 20$:
0061 237
0061 238
0061 239 ; Allocate irab and asb. Both structures are variable in
0061 240 ; size depending on organization.
```



```
0061 241 ;
0061 242 ;
52 9B AF45 9A 0061 243 MOVZBL IRAB_SIZE_TBL[R5],R2 ; get irab size in longwords
51 5A D0 0066 244 MOVL R10,R1 ; page having free space header
FF94' 30 0069 245 BSBW RMS$GETBLK ; allocate irab
64 50 E9 006C 246 BLBC R0,EX_NOSTR ; branch if none
59 51 D0 006F 247 MOVL R1,R9 ; stuff irab
52 8D AF45 3C 0072 248 MOVZWL ASB_SIZE_TBL[R5],R2 ; get asb size in longwords
51 5A D0 0077 249 MOVL R10,R1 ; restore page having header
FF83' 30 007A 250 BSBW RMS$GETBLK ; allocate asb
56 50 E9 007D 251 BLBC R0,ERRASB ; branch if none
08 A1 0D 90 0080 252 MOVB #ASB$C_BID,ASB$B_BID(R1) ; make it a real one
0084 253
0084 254 ASSUME ASB$W_STKLEN EQ 0
0084 255
61 FF79 0C A3 0084 256 SUBW3 #<ASB$C_BLN_FIX/4>,- ; calculate the size of the
61 04 A4 0086 257 ASB_SIZE_TBL[R5],(R1) ; stack save space in longwords
14 A9 51 D0 008B 258 MULW2 #4,(R1) ; convert it to bytes
5F 11 008E 259 MOVL R1,IRB$L_ASBADDR(R9) ; save address of asb
0092 260 BRB INIT_IRAB ; finish irab initialization
0094 261
```

```
0094 263
0094 264 ;
0094 265 ; error handling
0094 266 ;
0094 267
12 10 0094 268 ERRACT:
0094 269 BSBB ERROR
0096 270 RMSERR_WORD ACT ; fab function already active
0098 271
0E 10 0098 272 ERRIFI:
0098 273 BSBB ERROR
009A 274 RMSERR_WORD IFI ; invalid ifi value in fab
009C 275
0A 10 009C 276 ERRFAB:
009C 277 BSBB ERROR
009E 278 RMSERR_WORD FAB ; invalid fab
00A0 279
06 10 00A0 280 ERRBLN:
00A0 281 BSBB ERROR
00A2 282 RMSERR_WORD BLN ; invalid block length
00A4 283
02 10 00A4 284 ERRCCR:
00A4 285 BSBB ERROR
00A6 286 RMSERR_WORD CCR ; can't connect rab
00A8 287
50 9E 3C 00A8 288 ERROR:
FF52' 31 00A8 289 MOVZWL @(SP)+,R0 ; in-line error code to r0
00AB 290 BRW RMSEX_NOSTR ; report error
00AE 291
00AE 292 ERRBUG: RMSTBUG FTL$_BADIFAB ; invalid ifab table pointer
```



```
00B5 294
00B5 295 :
00B5 296 : this is an indirect connect for a process permanent file
00B5 297 :
00B5 298 : perform various checks to see if allowed
00B5 299 :
00B5 300
00B5 301 PPF_IND:
00B5 302 MOVW FAB$W_IFI(R10),-
00B8 303 RAB$W_ISI(R8) ; save rat value in isi
59 5A 59 D0 00BA 304 R9,R10 ; get ifab addr to right reg
1C A9 D0 00BD 305 IFB$L_IRAB_LNK(R9),R9 ; get irab addr
E1 13 00C1 306 BEQL ERRCCR ; branch if none
CD 69 20 E0 00C3 307 BBS #IRB$V_BUSY,(R9),ERRACT ; branch if busy
00C7 308
00C7 309 :
00C7 310 : do miscellaneous context cleanups
00C7 311 :
00C7 312
62 A9 B4 00C7 313 CLRW IRB$W_CSIZ(R9) ; say no current record
00CA 314
00CA 315 :
00CA 316 : return the isi value to the rab and do a structureless exit
00CA 317 :
00CA 318
52 A9 F0 00CA 319 20$: INSV IRB$B_PPF_ISI(R9),-
68 10 00CD 320 #RAB$W_ISI*8,-
06 06 00CE 321 #RAB$V_PPF_RAT,(R8) ; set table index into isi
00D0 322 RMSSUC
FF2A' 31 00D3 323 EX_NOSTR:
00D3 324 BRW RM$EX_NOSTR ; do structureless exit
00D6 325
00D6 326 :
00D6 327 : irab was allocated but no irab table slot or error allocating asb -
00D6 328 : deallocate the irab and return an error
00D6 329 :
00D6 330
00D6 331 ERRNOSLT:
00D6 332 ERRASB:
53 50 DD 00D6 333 PUSHL R0 ; Save the error code.
54 53 5A D0 00D8 334 MOVL R10,R3 ; page addr of free space hdr
14 A9 D0 00DB 335 MOVL IRB$L_ASBADDR(R9),R4 ; get addr of asb
06 13 00DF 336 BEQL 10$ ; skip if none
FF1C' 30 00E1 337 BSBW RM$RETBK ; give it back
53 5A D0 00E4 338 MOVL R10,R3 ; restore page addr of free space hdr
54 59 D0 00E7 339 10$: MOVL R9,R4 ; get addr of irab
FF13' 30 00EA 340 BSBW RM$RETBK ; give it back
50 8ED0 00ED 341 POPL R0 ; Restore error code
FF0D' 31 00F0 342 BRW RM$EX_NOSTR ; Exit to user without internal structure
```

```
00F3 344
00F3 345 :
00F3 346 : set irab bid, bln
00F3 347 :
00F3 348 :
00F3 349 INIT_IRAB:
08 A9 0A 90 00F3 350         MOVB    #IRB$C_BID,IRB$B_BID(R9)
00F7 351
00F7 352 :
00F7 353 : Check to see if this is a restart connect operation by looking for
00F7 354 : a context XAB and whether the restart option bit is set.
00F7 355 :
00F7 356         PUSHL   AP                ; Save argument pointer.
FF04' 30 00F9 357         BSBW    RMS$CONNXAB          ; Look for Context XAB - calls XAB_SCAN
00FC 358         POPL    AP                ; Restore AP
D4 50 8ED0 00FF 359         BLBC    R0,ERRNOSLT          ; error: return, after returning IRB pg
04 69 3B E1 0102 360         BBC     #IRB$V_RESTART,(R9),5$ ; If not restart operation, continue
56 02 AB 3C 0106 361         MOVZWL  RAB$W_ISI(R8),R6    ; Set up ISI value for get slot.
010A 362
010A 363 :
010A 364 : allocate and initialize a slot in the irab table
010A 365 :
010A 366
55 1C AB D0 010A 367 5$:        MOVL    IMP$L_IRABTBL(R11),R5 ; set table addr
FEEF' 30 010E 368        BSBW    RMS$GTSLT          ; and get a slot
C2 50 E9 0111 369        BLBC    R0,ERRNOSLT          ; branch if none
02 AB 56 B0 0114 370        MOVW    R6,RAB$W_ISI(R8)    ; store isi value
52 A9 56 B0 0118 371        MOVW    R6,IRB$W_OWN_ISI(R9) ; save isi value
50 00000000'9F D0 011C 372        MOVL    @#CTL$GL_PCB,R0 ; get addr of pcb for process
60 A0 B0 0123 373        MOVW    PCB$S_PID(R0),-
50 A9 0126 374        IRB$W_OWN_ID(R9) ; get process id index
0128 375
0128 376 :
0128 377 : give the irab a unique identifier
0128 378 :
0128 379
34 A9 00000000'9F D6 0128 380        INCL    @#PIO$GL_NXTIRBSEQ ; up next IRB sequence number
00000000'9F D0 012E 381        MOVL    @#PIO$GL_NXTIRBSEQ,IRB$S_IDENT(R9) ; move into IRB
0136 382
0136 383 :
0136 384 : link the irab into the ifab's chain (at start)
0136 385 :
0136 386
1C A9 69 5A D0 0136 387        MOVL    R10,IRB$S_IFAB_LNK(R9) ; set ifab ptr
1C AA 1C AA D0 0139 388        MOVL    IFB$S_IRAB_LNK(R10),IRB$S_IRAB_LNK(R9)
1C AA 59 D0 013E 389        MOVL    R9,IFB$S_IRAB_LNK(R10)
0142 390
0142 391 :
0142 392 : if this file is being journaled, call RMS$CONJNL
0142 393 :
0142 394
OC 00A2 CA 01 E1 0142 395        BBC     #IFB$V_JNL,IFB$B_JNLFLG2(R10),10$ ; branch if not journaling
00000000'EF 16 0148 396        JSB     RMS$CONJNL          ; get journaling BDB and buffer
03 50 E8 014E 397        BLBS    R0,10$                ; continue if ok
0042 31 0151 398        BRW     ERRORG                ; get out on error
0154 399
0154 400 : set stream busy and call rm$rset_alt to perform various other setups
```



```
0154 401 ;
0154 402 ;
FEA5' 30 0154 403 10$: SSB #IRB$V_BUSY,(R9) ; set stream busy
00 0158 404 BSBW RMS$RSET_ALT ; no error possible
015B 405 .BYTE 0 ; no in-line checks
015C 406 ;
015C 407 ;
015C 408 ; dispatch to organization-dependent routine
015C 409 ;
015C 410 ;
015C 411 CASE TYPE=B,- ; pickup correct routine
015C 412 SRC=IFB$B_ORGCASE(R10),-
015C 413 DISPLIST=Z100$,200$,300$> ; seq, rel, idx orgs
0167 414 ;
0167 415 ;++
0167 416 ;
0167 417 ; connect for unknown org. verify that only block i/o will be done,
0167 418 ; giving error otherwise.
0167 419 ;
0167 420 ;--
0167 421 ;
0167 422 RMSERR ORG ; anticipate error
016C 423 BBC #RAB$V_BIO+ROP,(R8),20$ ; branch if bio rop option clear
04 22 AA 06 E5 0170 424 BBCC #FAB$V_BRO,IFB$B_FAC(R10),20$ ; branch if fac not bro
22 AA 20 88 0175 425 ;
0175 426 BISB2 #FAB$M_BIO,IFB$B_FAC(R10) ; switch bro to bio
0179 427 ;
22 AA 05 E1 0179 428 20$: BBC #FAB$V_BIO,IFB$B_FAC(R10),-
18 017D 429 ERRORG ; error if not block i/o
00000000'EF 17 017E 430 JMP RMS$CONNECT_BIO ; connect for block i/o
0184 431 ;
00000000'EF 17 0184 432 100$: JMP RMS$CONNECT1
00000000'EF 17 018A 433 200$: JMP RMS$CONNECT2
00000000'EF 17 0190 434 300$: JMP RMS$CONNECT3
0196 435 ;
0196 436 ;
FE67' 30 0196 437 ERRORG: BSBW RMS$CCLN1 ; deallocate irab
FE64' 31 0199 438 BRW RMSEX_NOSTR ; and get out
019C 439 ;
019C 440 .END
```

Variable	Value	Mode	Priority
\$\$PSECT EP	= 00000000		
\$\$RMSTEST	= 0000001A		
\$\$RMS_PBUGCHK	= 00000010		
\$\$RMS_TBUGCHK	= 00000008		
\$\$RMS_UMODE	= 00000004		
ASB\$B-BID	= 00000008		
ASB\$C-BID	= 0000000D		
ASB\$C-BLN_FIX	= 00000030		
ASB\$C-BLN_IDX	= 00000200		
ASB\$C-BLN_REL	= 000000C0		
ASB\$C-BLN_SEQ	= 000000BC		
ASB\$W-STKLEN	= 00000000		
ASB_SIZE_TBL	= 00000003	R	01
CTL\$GL_PCB	*****	X	01
ERRACT	00000094	R	01
ERRASB	000000D6	R	01
ERRBLN	000000A0	R	01
ERRBUG	000000AE	R	01
ERRCCR	000000A4	R	01
ERRFAB	0000009C	R	01
ERRIFI	00000098	R	01
ERRNOSLT	000000D6	R	01
ERROR	000000A8	R	01
ERRORG	00000196	R	01
EX_NOSTR	000000D3	R	01
FAB\$B-BID	= 00000000		
FAB\$B-BLN	= 00000001		
FAB\$C-BID	= 00000003		
FAB\$C-BLN	= 00000050		
FAB\$M-BIO	= 00000020		
FAB\$V-BIO	= 00000005		
FAB\$V-BRO	= 00000006		
FAB\$V-PPF_IND	= 0000000E		
FAB\$W-IFI	= 00000002		
FTL\$BADIFAB	= FFFFFFFD		
IFB\$B-BID	= 00000008		
IFB\$B-BLN	= 00000009		
IFB\$B-FAC	= 00000022		
IFB\$B-JNLFLG2	= 000000A2		
IFB\$B-ORGCASE	= 00000023		
IFB\$C-BID	= 0000000B		
IFB\$C-BLN	= 000000B8		
IFB\$C-MAXORG	= 00000002		
IFB\$C-SEQ	= 00000000		
IFB\$L-IRAB_LNK	= 0000001C		
IFB\$V-BUSY	= 00000020		
IFB\$V-JNL	= 00000001		
IFB\$V-MSE	= 00000031		
IMP\$L-IFABTBL	= 00000018		
IMP\$L-IRABTBL	= 0000001C		
INIT_IRAB	000000F3	R	01
IRAB_SIZE_TBL	00000000	R	01
IRB\$B-BID	= 00000008		
IRB\$B-PPF_ISI	= 00000052		
IRB\$C-BID	= 0000000A		
IRB\$C-BLN_IDX	= 000000C4		
IRB\$C-BLN_REL	= 00000064		

IRB\$C_BLN_SEQ
IRB\$L_ASBADDR
IRB\$L_IDENT
IRB\$L_IFAB_LNK
IRB\$L_IRAB_LNK
IRB\$V_BUSY
IRB\$V_RESTART
IRB\$W_CSIZ
IRB\$W_OWN_ID
IRB\$W_OWN_ISI
PCB\$L_PID
PIO\$A_TRACE
PIO\$G_NXTIRBSEQ
PPF_IND
RAB\$L_FAB
RAB\$L_ROP
RAB\$V_BIO
RAB\$V_PPF_RAT
RAB\$W_ISI
RMSBUG
RMS\$CCLN1
RMS\$CONJNL
RMS\$CONNECT1
RMS\$CONNECT2
RMS\$CONNECT3
RMS\$CONNECT_BIO
RMS\$CONNAB
RMSEX_NOSTR
RMS\$GETBLK
RMS\$GTIADR
RMS\$GTSLT
RMS\$RABCHK
RMS\$RETBK
RMS\$RSET_ALT
RMS\$CONNECT
RMS\$_ACT
RMS\$_BLN
RMS\$_CCR
RMS\$_FAB
RMS\$_IFI
RMS\$_ORG
ROP
TPT\$L_CONNECT

```
= 0000006C  
= 00000014  
= 00000034  
= 00000000  
= 0000001C  
= 00000020  
= 0000003B  
= 00000062  
= 00000050  
= 00000052  
= 00000060  
***** X 01  
***** X 01  
000000B5 R 01  
= 0000003C  
= 00000004  
= 0000000B  
= 00000006  
= 00000002  
***** X 01  
***** X 01  
***** X 01  
***** X 01  
***** X 01  
***** X 01  
***** X 01  
***** X 01  
***** X 01  
***** X 01  
***** X 01  
***** X 01  
***** X 01  
***** X 01  
***** X 01  
***** X 01  
***** X 01  
***** X 01  
= 00000007 RG 01  
= 0001825A  
= 0001842C  
= 00018494  
= 0001850C  
= 00018564  
= 0001860C  
= 00000020  
***** 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	0000019C (412.)	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	35	00:00:00.07	00:00:00.37
Command processing	132	00:00:00.75	00:00:04.70
Pass 1	338	00:00:11.05	00:00:26.78
Symbol table sort	0	00:00:01.50	00:00:02.76
Pass 2	88	00:00:02.12	00:00:05.66
Symbol table output	13	00:00:00.12	00:00:00.79
Psect synopsis output	2	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	610	00:00:15.64	00:00:41.09

The working set limit was 1500 pages.
60347 bytes (118 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 1152 non-local and 14 local symbols.
440 source lines were read in Pass 1, producing 14 object records in Pass 2.
30 pages of virtual memory were used to define 29 macros.

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

Macro library name	Macros defined
-\$255\$DUA28:[RMS.OBJ]RMS.MLB;1	17
-\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	4
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2	4
TOTALS (all libraries)	25

1294 GETS were required to define 25 macros.

There were no errors, warnings or information messages.

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

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

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

