



```

RRRRRRRR      MM      MM      11      GGGGGGGG  EEEEEEEEE  TTTTTTTTTT
RRRRRRRR      MM      MM      11      GGGGGGGG  EEEEEEEEE  TTTTTTTTTT
RR      RR     MMMM   MMMM   1111     GG          EE          TT
RR      RR     MMMM   MMMM   1111     GG          EE          TT
RR      RR     MM     MM     MM     11     GG          EE          TT
RR      RR     MM     MM     MM     11     GG          EE          TT
RRRRRRRR      MM      MM      11      GG          EEEEEEEEE  TT
RRRRRRRR      MM      MM      11      GG          EEEEEEEEE  TT
RR  RR        MM      MM      11      GG  GGGGGG  EE          TT
RR  RR        MM      MM      11      GG  GGGGGG  EE          TT
RR      RR     MM      MM      11      GG          GG          EE          TT
RR      RR     MM      MM      11      GG          GG          EE          TT
RR      RR     MM      MM      11      GG          GG          EE          TT
RR      RR     MM      MM      111111   GGGGGG  EEEEEEEEE  TT
RR      RR     MM      MM      111111   GGGGGG  EEEEEEEEE  TT

```

```

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)	84
(4)	113
(14)	401

DECLARATIONS  
RMSGET1/RMS\$FIND1 - SEQ. \$GET & \$FIND  
RMS\$SEQKEY - ROUTINE TO CONVERT REC NO. TO RFA FOR SEQ. ORG

```
0000 1          $BEGIN RM1GET,000,RMSRMS1,<SEQUENTIAL SPECIFIC GET>,<NOWRT,QUAD>
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 provides sequential file organization-
0000 33 : specific processing for the $GET and $FIND functions.
0000 34 :
0000 35 : Environment:
0000 36 : STAR processor running STARLET EXEC.
0000 37 :
0000 38 : Author: L F Laverdure, CREATION DATE: 4-Feb-1977
0000 39 :
0000 40 : Modified By:
0000 41 :
0000 42 : V03-006 RAS0278 Ron Schaefer 24-Mar-1984
0000 43 : Minor tweaking for performance/size.
0000 44 :
0000 45 : V03-005 RAS0120 Ron Schaefer 25-Jan-1983
0000 46 : Add support to echo SYSS$INPUT to SYSS$OUTPUT.
0000 47 :
0000 48 : V03-004 KBT0414 Keith B. Thompson 30-Nov-1982
0000 49 : Change ifb$_devbufsiz to ifb$_devbufsiz
0000 50 :
0000 51 : V03-003 KBT0340 Keith B. Thompson 16-Sep-1982
0000 52 : Correct some opps from kbt0086
0000 53 :
0000 54 : V03-002 KBT0139 Keith B. Thompson 20-Aug-1982
0000 55 : Reorganize psects
0000 56 :
0000 57 : V03-001 KBT0086 Keith B. Thompson 13-Jul-1982
0000 58 : Clean up psects
0000 59 :
0000 60 : V02-022 RAS0063 Ron Schaefer 29-Jan-1982
0000 61 : Correct probes of user key and record buffers.
0000 62 :
0000 63 : V02-021 JWH0001 Jeffrey W. Horn 31-DEC-1981
0000 64 : Fix broken branches.
0000 65 :
0000 66 : V02-020 RAS0018 Ron Schaefer 7-Aug-1981
0000 67 : Fix broken branch caused by stream files.
0000 68 :
0000 69 : V02-019 REFORMAT Frederick E. Deen, Jr. 28-Jul-1980
0000 70 : This code was reformatted to adhere to RMS standards
0000 71 :
0000 72 : V018 CDS0001 C D Saether 11-Mar-1980
0000 73 : Fix divide by 0 caused by record size > blocksize and
0000 74 : no span attribute set (no..sense combination). Don't
0000 75 : look at RAC in RAB to determine whether saved NRP context
0000 76 : on find is to be restored.
0000 77 :
0000 78 : V017 JAK0001 J A Krycka 27-Aug-1978
0000 79 : Miscellaneous clean-up prior to DECNET V1.0 code freeze.
0000 80 : Add code to support network access by key.
0000 81 :
0000 82 :--
```

```
0000 84      .SBTTL  DECLARATIONS
0000 85
0000 86  :
0000 87  : INCLUDE FILES:
0000 88  :
0000 89  :
0000 90  :
0000 91  : MACROS:
0000 92  :
0000 93  :
0000 94      $IRBDEF
0000 95      $IFBDEF
0000 96      $RABDEF
0000 97      $FABDEF
0000 98      $DEVDEF
0000 99      $RMSDEF
0000 100
0000 101  :
0000 102  : EQUATED SYMBOLS:
0000 103  :
0000 104  :
00000020 0000 105      BKP=IRB$L_BKPBITS*8
00000020 0000 106      ROP=RAB$L_ROP*8           ; bit offset to ROP
0000 107
0000 108  :
0000 109  : OWN STORAGE:
0000 110  :
0000 111
```

```
0000 113      .SBTTL RMSGET1/RMS$FIND1 - SEQ. $GET & $FIND
0000 114
0000 115 :++
0000 116 : RMSGET1/RMS$FIND1 - Common sequential file organization for $GET and $FIND
0000 117 :
0000 118 : This module performs the following functions:
0000 119 :
0000 120 :     1. Common $GET/$FIND setup
0000 121 :     2. For UNIT RECORD DEVICES calls RMSGET_UNIT_REC
0000 122 :        otherwise, performs additional setup and calls RMSGET_BLK_DEV
0000 123 :     3. Set 'LAST-OPERATION-WAS-A-FIND' and NRP context
0000 124 :
0000 125 :
0000 126 : CALLING SEQUENCE:
0000 127 :
0000 128 :     Entered via CASE branch from RMSS$GET
0000 129 :     or RMSS$FIND at RMSGET1 or RMS$FIND1 respectively.
0000 130 :
0000 131 :     EXIT is to user via RM$EXRMS.
0000 132 :
0000 133 :
0000 134 : INPUT PARAMETERS:
0000 135 :
0000 136 :     R11      IMPURE AREA address
0000 137 :     R10      IFAB address
0000 138 :     R9       IRAB address
0000 139 :     R8       RAB address
0000 140 :
0000 141 :
0000 142 : IMPLICIT INPUTS:
0000 143 :
0000 144 :     The contents of the RAB and RELATED IRAB and IFAB.
0000 145 :     In particular, IRBSV_FIND must be set if doing $FIND, else clear.
0000 146 :
0000 147 : OUTPUT PARAMETERS:
0000 148 :
0000 149 :     R7 - R1      Destroyed
0000 150 :     R0           STATUS
0000 151 :
0000 152 :
0000 153 : IMPLICIT OUTPUTS:
0000 154 :
0000 155 :     Various fields of the RAB are filled in to reflect the status of
0000 156 :     the operation (see functional spec for details).
0000 157 :
0000 158 :     The IRAB is similarly updated.
0000 159 :
0000 160 :
0000 161 : COMPLETION CODES:
0000 162 :
0000 163 :     STANDARD RMS (see functional spec).
0000 164 :
0000 165 : SIDE EFFECTS:
0000 166 :
0000 167 :     None
0000 168 : --
0000 169
```

```

0000 171 RMSGET1::
0000 172 $STPT GET1
22 A8 B4 0J06 173 CLRW RAB$W_RSZ(R8) ; init the RSZ field
2A AA B5 0009 174 TSTW IFB$W_ECHO_ISI(R10) ; any ISI value?
10 13 000C 175 BEQL RMSFIND1 ; no echo if not
OC 6A 2E E1 000E 176 BBC #IFB$V_PPF_INPUT,(R10),RMSFIND1 ; not if not sysin
08 69 22 E1 0012 177 BBC #IRB$V_PPF_IMAGE,(R9),RMSFIND1 ; not if not indirect
0016 178 CSB #IRB$V_ASYNC,(R9) ; force synchronous
001A 179 SSB #IRB$V_PPF_ECHO,(R9) ; ask for echo
001E 180
001E 181 ;
001E 182 ; Make user input valid checks
001E 183 ;
001E 184
001E 185 RMSFIND1::
001E 186 $STPT FIND1
62 A9 B4 0024 187 CLRW IRB$W_CSIZ(R9) ; clear current record size
0027 188 ; indicates no current record
0027 189 ;
0027 190 ; Check for UNIT RECORD DEVICE and if so dispatch to appropriate code
0027 191 ;
0027 192
0027 193 ASSUME DEV$V_REC EQ 0
33 6A E8 0027 194 BLBS IFB$L_PRIM_DEV(R10),GET_UR ; branch if UNIT RECORD
002A 195
002A 196 ;
002A 197 ; Files oriented device - dispatch on record access type
002A 198 ;
002A 199
002A 200 CASE TYPE=B, SRC=RAB$B_RAC(R8),-
002A 201 DISPLIST=<GETSEQ,GETKEY,GETRFA>

```



```
0035 203 :  
0035 204 : Handle errors  
0035 205 :  
0035 206 :  
2D 11 0035 207 ERRRAC: RMSERR RAC  
003A 208 BRB G_XIT  
003C 209  
18 E0 003C 210 ERRRFA: BBS #DEV$V FOR,- ; allow LBN 0 for NFS operation  
46 6A 003E 211 IFB$SL_PRIM_DEV(R10),RFAOFF  
0040 212 RMSERR RFA  
22 11 0045 213 BRB G_XIT  
0047 214  
1B 11 0047 215 ERRIOP: RMSERR IOP  
004C 216 BRB G_XIT  
004E 217  
04 6A 2E E1 004E 218 EPRSQO: BBC #IFB$V_PPF_INPUT,(R10),10$ ; branch if not 'INPUT'  
2B 69 22 E1 0052 219 BBC #IRB$V_PPF_IMAGE,(R9),TSTRFA ; continue if not image  
0056 220 10$: RMSERR SQO ; not SEQ. operation  
0C 11 005B 221 BRB G_XIT
```

```
005D 223 :++
005D 224 :
005D 225 : GET for UNIT RECORD DEVICE
005D 226 :
005D 227 :--
005D 228 :
005D 229 :
05 6A 3E E0 005D 230 GET_UR: ASSUME RAB$C SEQ EQ 0
1E AB 95 0061 231 BBS #IFB$V DAP,(R10),10$ ; branch if network operation
CF 12 0064 232 TSTB RAB$B RAC(R8) ; allow sequential access only
FF97 30 0066 233 10$: BNEQ ERRRAC ; branch on error
6C 11 0069 234 G_XIT: BSBW RMSGET_UNIT_REC ; perform UNIT RECORD GET
006B 235 GETXIT ; and branch to GET EXIT
006B 236 :
006B 237 : Keyed access type
006B 238 :
006B 239 :
0098 30 006B 240 GETKEY: BSBW RMSSEQKEY ; convert key to RFA
66 50 E9 006E 241 BLBC RO,GETXIT ; get out on error
0071 242 :
0071 243 :
0071 244 : GET by RFA
0071 245 :
0071 246 :
04 69 29 E1 0071 247 GETRFA:
0071 248 BBC #IRB$V FIND,(R9),10$ ; branch if doing $GET
7E 40 A9 7D 0075 249 ASSUME IRB$W_NRP_OFF EQ IRB$N_NRP_VBN+4
0079 250 MOVQ IRB$N_NRP_VBN(R9),-(SP) ; save NRP context for random
0079 251 ; FIND keeping stack long aligned
0079 252 :
0079 253 :
0079 254 : Check for valid RFA value
0079 255 :
0079 256 :
0079 257 10$: BBC #DEV$V RND,- ; branch if no random access
CD 6A CA 6A E0 007B 258 IFB$N PRIM_DEV(R10),ERRIOP
10 AB D5 007D 259 BBS #IFB$V SQO,(R10),ERRSQO ; branch if SQO specified
B6 13 0081 260 TSTRFA: TSTL RAB$W RFA(R8) ; zero RFA?
14 AB B1 0084 261 BEQL ERRRFA ; branch if yes
48 AA 0086 262 RFAOFF: CMPW RAB$W_RFA+4(R8),- ; offset within a block?
AF 1E 0089 263 IFB$N_DEVBUFSIZ(R10)
008B 264 BGEQU ERRRFA ; branch if not
40 A9 10 AB 7D 008D 265 ASSUME IRB$W_NRP_OFF EQ IRB$N_NRP_VBN+4
46 A9 B4 008D 266 MOVQ RAB$W_RFA(R8),IRB$N_NRP_VBN(R9) ; copy RFA to NRP
0092 267 CLRW IRB$N_NRP_OFF+2(R9) ; guarantee offset valid long word
0095 268 :
0095 269 :
0095 270 : Do final setups for random access GET.
0095 271 : Clear LAST FIND and EOF status bits.
0095 272 :
0095 273 :
04 A9 22 8A 0095 274 BICB2 #<1@<IRB$V_FIND_LAST-BKP>>!-
0099 275 <1@<IRB$V_EOF-BRP>>.IRB$N_BKPBITS(R9)
0099 276 :
0099 277 :
0099 278 : Determine number of blocks to read in based upon the longest record
0099 279 : in the file.
```

				0099	280	:			
				0099	281				
				0099	282		ADDW3	IRBSW_NRP_OFF(R9),-	: compute total # bytes needed
				009C	283			IFBSW_LRL(R10),R2	
52	44	A9	A1	009F	284		CMPB	IFBSB_RFMORG(R10),#FABSC_FIX	: fixed length records?
01	50	AA	91	00A3	285		BEQL	10\$	: branch if yes
				00A5	286		ADDW2	#2,R2	: add in size field overhead
	52	02	A0	00A8	287	10\$:	DECW	R2	: round down
52	52	07	09	00AA	288		EXTZV	#9,#7,R2,R2	: get # blks - 1
				00AF	289		CMPB	R2,IRBSB_MBC(R9)	: < MBC?
				00B3	290		BGEQU	30\$	: branch if not
	53	02	D0	00B5	291		MOVL	#2,R3	: set flag for short read
				00B8	292		BSBW	RMSGETRANDOM	: go read and return the record
18	69	29	E1	00BB	293	20\$:	BBC	#IRBSV_FIND, (R9), GETXIT	: all done if this was a GET
				00BF	294		ASSUME	IRBSW_NRP_OFF EQ IRBSL_NRP_VBN+4	
40	A9	8E	7D	00BF	295		MOVQ	(SP)+, IRBSL_NRP_VBN(R9)	: clean saved NRP info off STACK
				00C3	296		BRB	GETXIT	: all done
				00C5	297	30\$:			
				00C5	298		BSBW	RMSGET_BLK_DEV	: GET the record
				00C8	299		BRB	20\$	: return to mainline

```

00CA 301
00CA 302 :++
00CA 303 :
00CA 304 : Sequential access $GET
00CA 305 :
00CA 306 :--
00CA 307
00CA 308 .ALIGN LONG
00CC 309 GETSEQ:
04 69 29 E0 00CC 310 BBS #IRBSV_FIND,(R9),GETSQ1 ; branch if doing $FIND
11 69 25 E0 00D0 311 BBS #IRBSV_FIND_LAST,(R9),RESET_NRP ; branch if $FIND was last
FF29' 30 00D4 312 GETSQ1: BSBW RMSGET_BLK_DEV ; go get the record
00D7 313
00D7 314 :++
00D7 315 :
00D7 316 : Exit GET function. Set LAST-OPERATION-WAS-A-FIND status as required.
00D7 317 :
00D7 318 :--
00D7 319
11 69 29 E4 00D7 320 GETXIT: CSB #IRBSV_FIND_LAST,(R9) ; clear LAST-OPERATION WAS A FIND
2F 50 E9 00DB 321 BBSC #IRBSV_FIND_LAST,(R9),FNDXIT ; branch if doing a FIND
FF1B' 31 00DF 322 CHKERR: BLBC R0,GETERR ; branch on error
00E2 323 EXIT: BRW RM$EXRMS ; R0 still has SUC/ERR code
00E5 324
00E5 325 :++
00E5 326 :
00E5 327 : This is a $GET operation after a $FIND. Reset NRP to RP.
00E5 328 :
00E5 329 :--
00E5 330
00E5 331 ASSUME IRBSW_RP_OFF EQ IRBSL_RP_VBN+4
00E5 332 ASSUME IRBSW_NRP_OFF EQ IRBSE_NRP_VBN+4
00E5 333 RESET_NRP:
48 A9 7D 00E5 334 MOVQ IRBSL_RP_VBN(R9),- ; reset NRP
40 A9 00E8 335 IRBSL_NRP_VBN(R9)
E6 69 21 E5 00EA 336 BBCC #IRBSV_EOF,(R9),GETSQ1 ; make sure EOF flag off
E4 11 00EE 337 BRB GETSQ1 ; rejoin GET code

```

```
00F0 339
00F0 340 :++
00F0 341 :
00F0 342 : Exiting from a $FIND. Check for PROCESS-PERMANENT FILE special processing
00F0 343 : and setting of the FIND_LAST flag.
00F0 344 :
00F0 345 :--
00F0 346
15 69 1E 50 E9 00F0 347 FNDXIT: BLBC RO,GETERR : branch if operation failed
06 69 2E E4 00F3 348 BBSC #IRBSV_PPF_EOF,(R9),PPF_EOF : branch if SYSS$INPUT EOF
EO 69 2F E4 00F7 349 BBSC #IRBSV_PPF_SKIP,(R9),PPF_SKIP : or must skip record
DE 25 E3 00FB 350 BBSC #IRBSV_FIND_LAST,(R9),CHKERR : set last opr. was a FIND
11 00FF 351 BRB CHKERR : rejoin GET code
```

```
0101 353
0101 354 :++
0101 355 :
0101 356 : IRBSV_PPF_SKIP was set (now clear).
0101 357 : This indicates that we have just skipped over a $DECK record on SYSSINPUT.
0101 358 :
0101 359 : Restore IRBSV_FIND from IRBSV_PPF_FNDSV and redo the $FIND or $GET.
0101 360 :
0101 361 :--
0101 362 :
0101 363 PPF_SKIP:
04 69 30 E5 0101 364 BBCC #IRBSV_PPF_FNDSV,(R9),10$ ; branch if not doing $FIND
FF12 31 0105 365 SSB #IRBSV_FIND,(R9) ; set FIND bit
0109 366 10$: BRW RMS$FIND1 ; redo $FIND or $GET
010C 367 :
010C 368 :++
010C 369 :
010C 370 : IRBSV_PPF_EOF was set (now clear).
010C 371 : This indicates that either a $EOD or user-defined EOD-STRING was seen
010C 372 : and has been skipped.
010C 373 :
010C 374 : Change status code to RMS$_EOF
010C 375 :
010C 376 :--
010C 377 :
010C 378 PPF_EOF:
010C 379 RMSERR EOF
0111 380 :
0111 381 :++
0111 382 :
0111 383 : An error has occurred.
0111 384 :
0111 385 : Unless error is 'RTB', zero current record size.
0111 386 :
0111 387 :--
0111 388 :
0111 389 GETERR:
81A8 8F 50 B1 0111 390 CMPW R0,RMS$_RTB&^XFFFF ; was error RTB?
0116 391 BEQL EXIT1 ; branch if yes
62 A9 B4 0118 392 CLRW IRBSW_CSIZ(R9) ; invalidate current rec indicator
CA 011B 393 BICL2 #<1@<IRBSV_PPF_EOF-BKP>>!- ; clean up status
011C 394 <1@<IRBSV_PPF_SKIP-BKP>>!-
011C 395 <1@<IRBSV_PPF_ECHO-BKP>>!-
011C 396 <1@<IRBSV_PPF_FNDSV-BKP>>,-
04 A9 0201C000 8F 011C 397 IRBSL BKPBITSTR9)
FEDA' 31 0123 398 EXIT1: BRW RMSEX RMS
0126 399
```

```

0126 401 .SBTTL RMSSEQKEY - ROUTINE TO CONVERT REC NO. TO RFA FOR SEQ. ORG
0126 402
0126 403 :++
0126 404 : RMSSEQKEY - Convert rec nbr. to RFA for sequential organization
0126 405 :
0126 406 : This routine converts a record number to an RFA for fixed length
0126 407 : records.
0126 408 :
0126 409 : CALLING SEQUENCE:
0126 410 :
0126 411 : BSBW RMSSEQKEY
0126 412 :
0126 413 : INPUT PARAMETERS:
0126 414 :
0126 415 : R10 IFAB address
0126 416 : R9 IRAB address
0126 417 : R8 RAB address
0126 418 :
0126 419 : RABSL_KBF Address of buffer having the relative record
0126 420 : number (RRN)
0126 421 : RAB$B_KSZ Size of key (must be 4 - defaulted if 0)
0126 422 :
0126 423 : IMPLICIT INPUTS:
0126 424 :
0126 425 : IFB$V_BLK Set if records cross block boundaries
0126 426 : IFB$W_MRS Fixed record length
0126 427 :
0126 428 : OUTPUT PARAMETERS:
0126 429 :
0126 430 : RAB$W_RFA Set to VBN and offset in block for record
0126 431 : R0 STATUS code
0126 432 : R1-R3 Destroyed
0126 433 :
0126 434 : COMPLETION CODES:
0126 435 :
0126 436 : Standard RMS, in particular, RAC, KBF, KSZ, KEY, IOP, or SUC.
0126 437 :
0126 438 : SIDE EFFECTS:
0126 439 :
0126 440 : None.
0126 441 :
0126 442 :--
0126 443
0126 444 RMSSEQKEY::
01 50 AA 91 0126 445 CMPB IFB$B_RFMORG(R10),#FAB$C_FIX ; must be fixed RFM
0126 446 BNEQ ERRRAC1 ; too bad if not
0126 447 TSTB RAB$B_KSZ(R8) ; zero size = 4
04 34 A8 95 012C 448 BEQL 10$ ; default buffer size
0126 449 CMPB RAB$B_KSZ(R8),#4 ; if not default must be 4
0126 450 BNEQ ERRKSZ
0126 451
0126 452 :
0126 453 : Pick up record number getting rid of bias (i.e., there is no record 0)
0126 454 :
0126 455 :
0126 456 10$: MOVAL @RAB$B_KBF(R8),R0 ; key buffer addr
50 30 B8 DE 0137 457 IFNORD #4,(R0),- ; check access

```

```

51 60 01 C3 013B 458 ERRKBF,IRBSB_MODE(R9)
51 60 51 19 0142 459 SUBL3 #1,(R0),R1 ; RRN - 1
0146 460 BLSS ERRKEY ; branch if bad
0148 461
0148 462 ::
0148 463 :: Get record length, rounding it up
0148 464 ::
0148 465 ::
50 60 AA 3C 0148 466 MOVZWL IFBSW_MRS(R10),R0 ; get fixed rec len
50 50 D6 014C 467 INCL R0 ; round up
7E 50 01 CA 014E 468 BICL2 #1,R0 ;
7E 0200 8F 3C 0151 469 MOVZWL #512,-(SP) ; useful constant to stack
10 51 AA E0 0156 470 BBS #FABS$V_BLK,- ; GET alternate calc. if
0158 471 IFBSB_RAT(R10),BLKSET ; records don't cross block
0158 472 ; boundaries
0158 473
0158 474 ::
0158 475 :: Records cross block boundaries.
0158 476 :: Compute byte address of record in file and convert to VBN and offset.
0158 477 ::
0158 478
52 6E 51 50 7A 0158 479 EMJL R0,R1,(SP),R2 ; compute byte addr =
14 A8 10 A8 52 8E 7B 0160 480 EDIV (SP)+,R2,- ; (RRN - 1)*rounded-rec-len+512
0167 481 RABS$W_RFA(R8),RABS$W_RFA+4(R8) ; compute VBN and offset
0167 482 ; (byte-addr/512)
05 016A 483 SUCRET: RMSSUC ; show success
484 RSB ; and return

```



```

0168 486
0168 487 :
0168 488 :   Alternate calculation for records not allowed to cross block boundaries
0168 489 :
0168 490
53 8E 52 D4 0168 491 BLKSET: CLRL R2 ; zero extend RRN-1
      50 C7 016D 492 DIVL3 R0,(SP)+,R3 ; get # RECS per block
      11 13 0171 493 ; =512/ROUNDED-REC-LEN
0171 494 BEQL ERRIOP_BR ; if zero, record size must be
0173 495 ; greater than 512 which
0173 496 ; doesn't make sense and we
0173 497 ; don't want to divide by
0173 498 ; zero, so get out
52 51 51 53 7B 0173 499 EDIV R3,R1,R1,R2 ; compute VBN-1 (in R1)
0178 500 ; and REC # within BLK (in R2)
0178 501 ; =(RRN-1)/# RECS-PER-BLK
10 A8 51 01 01 0178 502 ADDL3 #1,R1,RAB$W_RFA(R8) ; store VBN
14 A8 50 52 A5 017D 503 MULW3 R2,R0,RAB$W_RFA+4(R8) ; and offset (=REC # within BLK
      E3 11 0182 504 ; * ROUNDED-REC-LEN)
      0182 505 BRB SUCRET ; all set
      0184 506
      0184 507 :
      0184 508 :   Handle errors
      0184 509 :
      0184 510
      0184 511 ERRIOP_BR:
09 18 09 6A 51 0199 523 ERRKEY: BBC #DEV$V FOR,- ; definite error if not for.
      60 D5 019B 524 IFB$L_PRIM_DEV(R10),10$ ; attempt to read LBN 0?
      05 12 019F 525 TSTL (R0) ; branch if not (error)
10 A8 7C 01A1 526 BNEQ 10$ ; yes, zero LBN
      C1 11 01A4 527 CLRQ RAB$W_RFA(R8) ; continue
      01A6 528 BRB SUCRET
      01A6 529 10$:
      01AB 530 RMSERR KEY ; bad KEY value (not > 0)
      01AC 531 RSB
      01AC 532
      01AC 533 .END

```

```

$$PSECT EP = 00000000
$$RMSTEST = 0000001A
$$RMS_PBUGCHK = 00000010
$$RMS_TBUGCHK = 00000008
$$RMS_UMODE = 00000004
BKP = 00000020
BLKSET = 0000016B R 01
CHKERR = 000000DF R 01
DEV$V_FOR = 00000018
DEV$V_REC = 00000000
DEV$V_RND = 0000001C
ERRIOP = 00000047 R 01
ERRIOP_BR = 00000184 R 01
ERRKBF = 0000018D R 01
ERRKEY = 00000199 R 01
ERRKSZ = 00000193 R 01
ERRRAC = 00000035 R 01
ERRRAC1 = 00000187 R 01
ERRRFA = 0000003C R 01
ERRSQO = 0000004E R 01
EXIT = 000000E2 R 01
EXIT1 = 00000123 R 01
FAB$C_FIX = 00000001
FAB$V_BLK = 00000003
FNDXIT = 000000F0 R 01
GETERR = 00000111 R 01
GETKEY = 0000006B R 01
GETRFA = 00000071 R 01
GETSEQ = 000000CC R 01
GETSQ1 = 000000D4 R 01
GETXIT = 000000D7 R 01
GET UR = 0000005D R 01
G_XIT = 00000069 R 01
IFB$B_RAT = 00000051
IFB$B_RFMORG = 00000050
IFB$L_DEVBUFSIZ = 00000048
IFB$L_PRIM_DEV = 00000000
IFB$V_CAP = 0000003E
IFB$V_PPF_INPUT = 0000002E
IFB$V_SQO = 0000002D
IFB$W_ECHO_ISI = 0000002A
IFB$W_LRL = 00000052
IFB$W_MRS = 00000060
IRB$B_MBC = 00000055
IRB$B_MODE = 0000000A
IRB$L_BKPBITS = 00000004
IRB$L_NRP_OFF = 00000044
IRB$L_NRP_VBN = 00000040
IRB$L_RP_VBN = 00000048
IRB$V_ASYNC = 00000023
IRB$V_EOF = 00000021
IRB$V_FIND = 00000029
IRB$V_FIND_LAST = 00000025
IRB$V_PPF_ECHO = 00000039
IRB$V_PPF_EOF = 0000002E
IRB$V_PPF_FNDSV = 00000030
IRB$V_PPF_IMAGE = 00000022

```

```

IRB$V_PPF_SKIP = 0000002F
IRB$W_CSIZ = 00000062
IRB$W_NRP_OFF = 00000044
IRB$W_RP_OFF = 0000004C
PIO$A_TRACE ***** X 01
PPF_EOF = 0000010C R 01
PPF_SKIP = 00000101 R 01
RAB$B_KSZ = 00000034
RAB$B_RAC = 0000001E
RAB$C_SEQ = 00000000
RAB$L_KBF = 00000030
RAB$L_ROP = 00000004
RAB$W_RFA = 00000010
RAB$W_RSZ = 00000022
RESET_NRP = 000000E5 R 01
RFAOFF = 00000086 R 01
RM$EXRMS ***** X 01
RM$FIND1 = 0000001E RG 01
RM$GET1 = 00000000 RG 01
RM$GETRANDOM ***** X 01
RM$GET_BLK_DEV ***** X 01
RM$GET_UNIT_REC ***** X 01
RM$SEQKEY = 00000126 RG 01
RM$$_EOF = 0001827A
RM$$_IOP = 00018574
RM$$_KBF = 0001858C
RM$$_KEY = 00018594
RM$$_KSZ = 000185A4
RM$$_RAC = 00018644
RM$$_RFA = 0001865C
RM$$_RTB = 000181A8
RM$$_SQO = 000186C4
ROP = 00000020
SUCRET = 00000167 R 01
TPT$L_FIND1 ***** X 01
TPT$L_GET1 ***** X 01
TSTRFA = 00000081 R 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
RMSRMS1	000001AC ( 428.)	01 ( 1.)	PIC USR CON REL GBL NOSHR EXE RD NOWRT NOVEC QUAD
\$ABS\$	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	29	00:00:00.06	00:00:01.05
Command processing	107	00:00:00.67	00:00:04.39
Pass 1	303	00:00:09.63	00:00:27.97
Symbol table sort	0	00:00:01.24	00:00:01.67
Pass 2	106	00:00:02.17	00:00:05.34
Symbol table output	12	00:00:00.10	00:00:00.46
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	561	00:00:13.89	00:00:40.90

The working set limit was 1350 pages.  
53909 bytes (106 pages) of virtual memory were used to buffer the intermediate code.  
There were 50 pages of symbol table space allocated to hold 998 non-local and 15 local symbols.  
533 source lines were read in Pass 1, producing 14 object records in Pass 2.  
24 pages of virtual memory were used to define 23 macros.

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

Macro library name	Macros defined
-\$255\$DUA28:[RMS.OBJ]RMS.MLB;1	12
-\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	2
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2	5
TOTALS (all libraries)	19

1105 GETs were required to define 19 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:RM1GET/OBJ=OBJ\$:RM1GET MSRC\$:RM1GET/UPDATE=(ENH\$:RM1GET)+EXECMLS/LIB+LIB\$:RMS/LIB

RM1CONN  
LIS

RM1GET  
LIS

RM1INPSON  
LIS

RM1DISCON  
LIS

RM1GETINT  
LIS

RM1CREATE  
LIS

RM1JOURNL  
LIS