```
RRRRRRRRRRR      MMM          MMM      SSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM      SSSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM      SSSSSSSSSSSS
RRR          RRR MMMMMMM    MMMMMMM   SSS
RRR          RRR MMMMMMM    MMMMMMM   SSS
RRR          RRR MMMMMMM    MMMMMMM   SSS
RRR          RRR MMM   MMM   MMM   MMM SSS
RRR          RRR MMM   MMM   MMM   MMM SSS
RRR          RRR MMM   MMM   MMM   MMM SSS
RRRRRRRRRRR      MMM          MMM      SSSSSSSSS
RRRRRRRRRRRR     MMM          MMM      SSSSSSSSS
RRRRRRRRRRRR     MMM          MMM      SSSSSSSSS
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  SSSSSSSSSSS
RRR       RRR    MMM          MMM               SSS
RRR          RRR MMM          MMM  SSSSSSSSSSS
RRR          RRR MMM          MMM  SSSSSSSSSSS
RRR          RRR MMM          MMM  SSSSSSSSSSS
```

FILEID**RM1PUTBLD


```
RRRRRRRR    MM       MM      11      PPPPPPPP    UU      UU  TTTTTTTTTT  BBBBBBBB    LL          DDDDDDDD
RRRRRRRR    MM       MM      11      PPPPPPPP    UU      UU  TTTTTTTTTT  BBBBBBBB    LL          DDDDDDDD
RR      RR  MMMM   MMMM    1111      PP      PP  UU      UU      TT      BB      BB  LL          DD      DD
RR      RR  MMMM   MMMM    1111      PP      PP  UU      UU      TT      BB      BB  LL          DD      DD
RR      RR  MM  MM  MM      11      PP      PP  UU      UU      TT      BB      BB  LL          DD      DD
RR      RR  MM  MM  MM      11      PP      PP  UU      UU      TT      BB      BB  LL          DD      DD
RRRRRRRR    MM       MM      11      PPPPPPPP    UU      UU      TT      BBBBBBBB    LL          DD      DD
RRRRRRRR    MM       MM      11      PPPPPPPP    UU      UU      TT      BBBBBBBB    LL          DD      DD
RR  RR      MM       MM      11      PP          UU      UU      TT      BB      BB  LL          DD      DD
RR  RR      MM       MM      11      PP          UU      UU      TT      BB      BB  LL          DD      DD
RR    RR    MM       MM      11      PP          UU      UU      TT      BB      BB  LL          DD      DD      ....
RR    RR    MM       MM      11      PP          UU      UU      TT      BB      BB  LL          DD      DD      ....
RR      RR  MM       MM    111111    PP          UUUUUUUUUU      TT      BBBBBBBB    LLLLLLLLLL  DDDDDDDD        ....
RR      RR  MM       MM    111111    PP          UUUUUUUUUU      TT      BBBBBBBB    LLLLLLLLLL  DDDDDDDD        ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

```
0000     1          $BEGIN  RM1PUTBLD,000,RM$RMS1,<SEQ. ORG. PUT TO BLOCK DEVICE>
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 ; Facility: RMS32
0000    29 ;
0000    30 ; Abstract:
0000    31 ;                   This routine is called by RM1PUT to handle the
0000    32 ;                   blocking of records required when doing $PUTS to
0000    33 ;                   a block device.
0000    34 ;
0000    35 ; Environment:
0000    36 ;                   Star processor running Starlet exec.
0000    37 ;
0000    38 ; Author: L. F. Laverdure        Creation Date: 25-MAR-1977
0000    39 ;
0000    40 ; Modified By:
0000    41 ;
0000    42 ;           V03-009 JWT0190         Jim Teague              31-Jul-1984
0000    43 ;                   Undo JEJ0049 temporarily until some unexpected side
0000    44 ;                   effects can be better understood.
0000    45 ;
0000    46 ;           V03-008 JEJ0049         J E Johnson             23-Jul-1984
0000    47 ;                   Alter the BLDREC logic to force the current buffer to be
0000    48 ;                   flushed when it is exactly filled, rather than waiting
0000    49 ;                   for the next operation to force it out.
0000    50 ;
0000    51 ;           V03-007 DAS0002         David Solomon           28-Jun-1984
0000    52 ;                   Don't get the RSZ from the user's RAB after we've stalled. It
0000    53 ;                   may change.
0000    54 ;
0000    55 ;           V03-006 DAS0001         David Solomon           08-Feb-1984
0000    56 ;                   Re-lay out code to improve performance for sequential VAR case.
0000    57 ;                   Some formatting cleanup.
```

```
0000     58 ;
0000     59 ;       V03-005 TSK0052         Tamar Krichevsky        22-Jun-1983
0000     60 ;               Fix broken branch to RM$SEQJNL.
0000     61 ;
0000     62 ;       V03-004 TSK0051         Tamar Krichevsky        30-May-1983
0000     63 ;               Add support for journaling of $PUT operations.
0000     64 ;
0000     65 ;       V03-003 SHZ0001         Stephen H. Zalewski     17-Dec-1982
0000     66 ;               Fix assume statement to reflect change in ifb.  Change
0000     67 ;               way eof is updated from MOVC to MOVL.
0000     68 ;
0000     69 ;       V03-002 KBT0144         Keith B. Thompson       20-Aug-1982
0000     70 ;               Reorganize psects
0000     71 ;
0000     72 ;       V03-001 RAS0090         Ron Schaefer            8-Jun-1982
0000     73 ;               Fix potential bugcheck caused by having the NRP_OFF
0000     74 ;               be odd-aligned for VAR and VFC files.
0000     75 ;
0000     76 ;       V02-024 CDS0001         C Saether               3-Nov-1981
0000     77 ;               Fix broken branch.
0000     78 ;
0000     79 ;       V02-023 RAS0028         Ron Schaefer            20-Aug-1981
0000     80 ;               Change FAB$C_STM11 to FAB$C_STM.
0000     81 ;
0000     82 ;       V02-022 RAS0027         Ron Schaefer            20-Aug-1981
0000     83 ;               Do not pad UDF files to magtape.
0000     84 ;
0000     85 ;       V02-021 RAS0025         Ron Schaefer            18-Aug-1981
0000     86 ;               Add support for $PUT to UDF files.
0000     87 ;
0000     88 ;       V02-020 RAS0019         Ron Schaefer            6-Aug-1981
0000     89 ;               Fix block padding algorithm for magtape and stream.
0000     90 ;
0000     91 ;       V02-019 RAS0016         Ron Schaefer            31-Jul-1981
0000     92 ;               Add stream format support.
0000     93 ;
0000     94 ;       V02-018 REFORMAT        K. E. Kinnear           31-Jul-1980     9:04
0000     95 ;
0000     96 ;--
```

```
                     0000     98          .SBTTL  DECLARATIONS
                     0000     99
                     0000    100 ;
                     0000    101 ; Include Files:
                     0000    102 ;
                     0000    103
                     0000    104 ;
                     0000    105 ; Macros:
                     0000    106 ;
                     0000    107
                     0000    108          $IFBDEF
                     0000    109          $DEVDEF
                     0000    110          $IRBDEF
                     0000    111          $RABDEF
                     0000    112          $FABDEF
                     0000    113          $IMPDEF
                     0000    114          $BDBDEF
                     0000    115          $RMSDEF
                     0000    116          $RJRDEF
                     0000    117
                     0000    118 ;
                     0000    119 ; Equated Symbols:
                     0000    120 ;
00000020             0000    121          ROP=RAB$L_ROP*8
0000000A             0000    122          LF=10
0000000D             0000    123          CR=13
                     0000    124 ;
                     0000    125 ; Own Storage:
                     0000    126 ;
                     0000    127
                     0000    128 ;
                     0000    129 ; Stream format default terminators (DFT)
                     0000    130 ;
                     0000    131 STM_FMT_DFT:
00 0A 0D 02          0000    132          .BYTE   2, CR, LF, 0
00 00 0A 01          0004    133          .BYTE   1, LF,  0, 0
00 00 0D 01          0008    134          .BYTE   1, CR,  0, 0
                     000C    135
```

```
                     137          .SBTTL  RM$PUT_BLK_DEV - ROUTINE TO PERFORM SEQ. ORG PUT TO FILES DEVICE
                     138
                     139 ;++
                     140 ; RM$PUT_BLK_DEV -- Routine to Performa Seq. Organization Put to Files Device.
                     141 ;
                     142 ;       This module performs the $put record processing for
                     143 ;       the sequential file organization to a files-oriented
                     144 ;       (i.e., blocked) device, performing the required record
                     145 ;       blocking.
                     146 ;
                     147 ; Calling Sequence:
                     148 ;
                     149 ;       BSBW    RM$PUT_BLK_DEV
                     150 ;
                     151 ; Input Parameters:
                     152 ;
                     153 ;       R11     impure area addr
                     154 ;       R10     IFAB addr
                     155 ;       R9      IRAB addr
                     156 ;       R8      rab addr
                     157 ;       R6      record data length in bytes
                     158 ;       R5      record address
                     159 ;
                     160 ; Implicit Inputs:
                     161 ;
                     162 ;       Sequential org temp.
                     163 ;               IRB$W_ROVHDSZ:  record overhead size in bytes
                     164 ;               IRB$W_RTOTLSZ:  total record length including
                     165 ;                               overhead bytes
                     166 ;       First block of user buffer probed
                     167 ;       Record size checked for validity
                     168 ;
                     169 ; Output Parameters:
                     170 ;
                     171 ;       R0      status code
                     172 ;       R1-R7   destroyed
                     173 ;
                     174 ; Implicit Outputs:
                     175 ;
                     176 ;       RAB$W_RFA
                     177 ;       Updates IRAB and bdb(s) as per the put.
                     178 ;
                     179 ; Completion Codes:
                     180 ;
                     181 ;       standard rms
                     182 ;
                     183 ; Side Effects:
                     184 ;
                     185 ;       none
                     186 ;
                     187 ;--
                     188
                     189 SEQJNL_BR:
0231    31   000C    190          BRW     SEQJNL
                     191 CHKBLK1_BR:
00E4    31   000F    192          BRW     CHKBLK1
                     193 ANSIZE_BR:
```

```
                00F6   31  0012   194          BRW     ANSIZE
                       0015   195
                       0015   196 RM$PUT_BLK_DEV::
                       0015   197          $TSTPT  PUTBLD
                       001B   198
                       001B   199 ;
                       001B   200 ; For non-magtape devices, fix potential odd-aligned NRP_OFF values,
                       001B   201 ; for VAR and VFC files.
                       001B   202 ;
                       001B   203
                   05  E0  001B   204          BBS     #DEV$V_SQD,-            ; don't worry if tape
                13 6A           001D   205                  IFB$L_PRIM_DEV(R10),6$
                50 AA  91  001F   206          CMPB    IFB$B_RFMORG(R10),-     ; VAR format?
                   02           0022   207                  #FAB$C_VAR
                   06  13  0023   208          BEQL    5$                      ; yes
                50 AA  91  0025   209          CMPB    IFB$B_RFMORG(R10),-     ; or VFC?
                   03           0028   210                  #FAB$C_VFC
                   07  12  0029   211          BNEQ    6$                      ; not VFC or VAR
                44 A9  B6  002B   212 5$:      INCW    IRB$W_NRP_OFF(R9)       ; round offset
            44 A9  01  AA  002E   213          BICW2   #1,IRB$W_NRP_OFF(R9)
                       0032   214
                       0032   215 ;
                       0032   216 ; The following code causes the current buffer to be set up correctly
                       0032   217 ; dependent upon whether the corresponding block must be read or not.
                       0032   218 ;
                       0032   219
                   53  D4  0032   220 6$:      CLRL    R3                      ; indicate read required
                44 A9  B5  0034   221          TSTW    IRB$W_NRP_OFF(R9)       ; check for non-zero offset
                   02  12  0037   222          BNEQ    20$                     ; and branch if so
                   53  D6  0039   223          INCL    R3                      ; zero offset - flag no read required
                FFC2'  30  003B   224 20$:     BSBW    RM$GETBLKNRP            ; get the block
                3A 50  E9  003E   225          BLBC    R0,ZERO_RFA_BR2         ; continue if successful
                       0041   226
                       0041   227 ;
                       0041   228 ; Current register contents:
                       0041   229 ;
                       0041   230 ;       R8-R11  same as upon entry
                       0041   231 ;       R7      end of block addr + 1
                       0041   232 ;       R6      data record size
                       0041   233 ;       R5      data record addr
                       0041   234 ;       R4      bdb addr
                       0041   235 ;       R1      addr of block buffer
                       0041   236 ;
                       0041   237
                       0041   238 ;
                       0041   239 ; Make R1 into addr of location to build the record.
                       0041   240 ;
                       0041   241
              51   44 A9  C0  0041   242 PUT00:  ADDL2   IRB$L_NRP_OFF(R9),R1    ; make into address
                       0045   243
                       0045   244 ;
                       0045   245 ; Set rp from nrp.
                       0045   246 ;
                       0045   247
            48 A9   40 A9  7D  0045   248          MOVQ    IRB$L_NRP_VBN(R9),IRB$L_RP_VBN(R9)
                       004A   249
                       004A   250 ;
```

```
                     004A   251 ; Compute # bytes left in block and check for fit
                     004A   252 ; if records not allowed to cross block boundaries.
                     004A   253 ;
                     004A   254
          50 57 51 C3 004A   255         SUBL3   R1,R7,R0                ; compute # bytes remaining
             03 E0   004E   256         BBS     #FAB$V_BLK,-            ; branch if boundary restrictions
           51 AA     0050   257                 IFB$B_RAT(R10),-
             BC      0052   258                 CHKBLK1_BR
                     0053   259
                     0053   260 ;
                     0053   261 ; If journaling is enabled for this file, create and write the journal entry
                     0053   262 ; for the current record.
                     0053   263 ;
                     0053   264
        00A0 CA 95   0053   265 CHKJNL: TSTB    IFB$B_JNLFLG(R10)       ; Any journaling enabled?
             B3 12   0057   266         BNEQ    SEQJNL_BR               ; yes, journal record
                     0059   267
                     0059   268 ;
                     0059   269 ; Build record starting at the address noted in R3.
                     0059   270 ;
                     0059   271
          64 A9 B5   0059   272 BUILD:  TSTW    IRB$W_ROVHDSZ(R9)       ; any overhead size?
             1A 13   005C   273         BEQL    MOVREC                  ; branch if none
                     005E   274
                     005E   275 ;
                     005E   276 ; Rec format is not fixed.
                     005E   277 ; Must write out size field if ANSI, or var or vfc.
                     005E   278 ;
                     005E   279
          6A 26 E0   005E   280         BBS     #IFB$V_ANSI_D,(R10),-   ; branch if ansi
             B0      0061   281                 ANSIZE_BR
                     0062   282
                     0062   283 ;
                     0062   284 ; Store binary size field.
                     0062   285 ;
                     0062   286
                     0062   287 SETSIZ: ASSUME  FAB$C_VFC GT FAB$C_VAR
                     0062   288         ASSUME  FAB$C_STM GT FAB$C_VFC
          50 AA 91   0062   289         CMPB    IFB$B_RFMORG(R10),-     ; stream format?
             04      0065   290                 #FAB$C_STM
             10 1E   0066   291         BGEQU   MOVREC                  ; no prefix for stream
          02 50 D1   0068   292         CMPL    R0,#2                   ; room for count?
             79 19   006B   293         BLSS    N_BLK_BR                ; nope
        66 A9 02 A3 006D   294         SUBW3   #2,IRB$W_RTOTLSZ(R9),-  ; store size of data + rhb
             81      0071   295                 (R1)+
                     0072   296
                     0072   297 ;
                     0072   298 ; If vfc format store record header data.
                     0072   299 ;
                     0072   300
          50 AA 91   0072   301 CHKVFC: CMPB    IFB$B_RFMORG(R10),-     ; is it vfc?
             03      0075   302                 #FAB$C_VFC
             70 13   0076   303         BEQL    VFCRH_BR
                     0078   304
                     0078   305 ;
                     0078   306 ; Now move the data record.
                     0078   307 ;
```

```
                     0078   308
          0187   30  0078   309 MOVREC: BSBW    BLDREC                      ; go move it
                     007B   310 ZERO_RFA_BR2:                               ; branch aid
        65 50    E9  007B   311         BLBC    R0,ZERO_RFA_BR              ; get out on error
                     007E   312
                     007E   313 ;
                     007E   314 ; Now append DFT to stream format if necessary
                     007E   315 ;
                     007E   316
                     007E   317         ASSUME  FAB$C_UDF EQ 0
                     007E   318         ASSUME  FAB$C_STM GT FAB$C_VFC
                     007E   319         ASSUME  <FAB$C_STM+1> EQ FAB$C_STMLF
                     007E   320         ASSUME  <FAB$C_STMLF+1> EQ FAB$C_STMCR
                     007E   321
     55    50 AA    9A  007E   322         MOVZBL  IFB$B_RFMORG(R10),R5        ; get format type
           24       13  0082   323         BEQL    20$                         ; skip to not round for UDF
        55  04       C2  0084   324         SUBL2   #FAB$C_STM,R5               ; normalize type
           16       1F  0087   325         BLSSU   10$                         ; not stream format
        64 A9       B5  0089   326         TSTW    IRB$W_ROVHDSZ(R9)           ; anything to add?
           1A       13  008C   327         BEQL    20$                         ; nope
     55   FF6D CF45   DE  008E   328         MOVAL   W^STM_FMT_DFT[R5],R5        ; point to DFT table
        56    85      9A  0094   329         MOVZBL  (R5)+,R6                    ; get length
          0168   30  0097   330         BSBW    BLDREC                      ; append the DFT
        46 50    E9  009A   331         BLBC    R0,ZERO_RFA_BR              ; quit on failure
           09       11  009D   332         BRB     20$                         ; rejoin main flow
                     009F   333
                     009F   334 ;
                     009F   335 ; PUT operation now complete.
                     009F   336 ; Calculate new value for nrp.
                     009F   337 ;
                     009F   338 ; Round up to word boundary unless stream or magtape.
                     009F   339 ;
                     009F   340
           05    E0  009F   341 10$:    BBS     #DEV$V_SQD,-
        05 6A           00A1   342                 IFB$L_PRIM_DEV(R10),20$ ; branch if magtape
           51    D6  00A3   343         INCL    R1                          ; round up
        51 01    CA  00A5   344         BICL2   #1,R1                       ; round up
     54    20 A9    D0  00A8   345 20$:    MOVL    IRB$L_CURBDB(R9),R4         ; get current bdb
        4C A4    C3  00AC   346         SUBL3   BDB$L_CURBUFADR(R4),-       ; save offset in nrp
     44 A9       51       00AF   347                 R1,IRB$L_NRP_OFF(R9)
  50    57       51  C3  00B2   348         SUBL3   R1,R7,R0                    ; set r0 to # of bytes remaining in block
           32       13  00B6   349         BEQL    BLK_FULL_BR                 ; and branch if full
                     00B8   350
                     00B8   351 ;
                     00B8   352 ; Make sure there is sufficient room in block for
                     00B8   353 ; ansi_d byte count or entire fixed length record if
                     00B8   354 ; records not allowed to cross block boundaries.
                     00B8   355 ;
                     00B8   356
           03    E0  00B8   357         BBS     #FAB$V_BLK,-                ; branch if boundary restriction
        51 AA           00BA   358                 IFB$B_RAT(R10),-
           30           00BC   359                 CHKBLK2_BR
                     00BD   360
                     00BD   361 ;
                     00BD   362 ; Update eof data in IFAB.
                     00BD   363 ;
                     00BD   364
```

```
                40 A9   D0  00BD   365 PUT01:  MOVL    IRB$L_NRP_VBN(R9),-     ; update ebk
                74 AA       00C0   366                 IFB$L_EBK(R10)
                44 A9   B0  00C2   367         MOVW    IRB$W_NRP_OFF(R9),-     ; update first free byte in block
                5C AA       00C5   368                 IFB$W_FFB(R10)
                            00C7   369         SSB     #IFB$V_RW_ATTR,(R10)    ; flag rewrite of attr. needed
                            00CB   370
                            00CB   371 ;
                            00CB   372 ; Update largest record in file.
                            00CB   373 ;
                            00CB   374
                64 A9   A3  00CB   375         SUBW3   IRB$W_ROVHDSZ(R9),-     ; get actual record size
                66 A9       00CE   376                 IRB$W_RTOTLSZ(R9),-     ;
                   50       00D0   377                 R0                      ;
            52 AA  50   B1  00D1   378         CMPW    R0,IFB$W_LRL(R10)       ; new largest?
                   04   1B  00D5   379         BLEQU   10$                     ; branch if not
            52 AA  50   B0  00D7   380         MOVW    R0,IFB$W_LRL(R10)       ; yes - save size
                            00DB   381
                            00DB   382 ;
                            00DB   383 ; Return rfa to user and exit.
                            00DB   384 ;
                            00DB   385
                            00DB   386 10$:    ASSUME  IRB$W_RP_OFF EQ IRB$L_RP_VBN+4
                48 A9   7D  00DB   387         MOVQ    IRB$L_RP_VBN(R9),-      ; return RFA to user
                10 A8       00DE   388                 RAB$W_RFA(R8)
                 FF1D'  31  00E0   389         BRW     RM$EXSUC                ; exit with success
```

```
                     00E3    391 ;
                     00E3    392 ; Branch aids.
                     00E3    393 ;
                     00E3    394
                     00E3    395 ZERO_RFA_BR:
0106    31           00E3    396         BRW     ZERO_RFA
                     00E6    397 N_BLK_BR:
  14    11           00E6    398         BRB     N_BLK
                     00E8    399 VFCRH_BR:
  3D    11           00E8    400         BRB     VFCRH
                     00EA    401 BLK_FULL_BR:
00DC    31           00EA    402         BRW     BLK_FULL
                     00ED    403 CHKBLK2_BR:
00C2    31           00ED    404         BRW     CHKBLK2
                     00F0    405 CHKJNL BR:
FF60    31           00F0    406         BRW     CHKJNL
                     00F3    407 PUT00_BR:
FF4B    31           00F3    408         BRW     PUT00
```

```
                     00F6   410 ;
                     00F6   411 ; Check for fit if records not allowed to cross block boundaries.
                     00F6   412 ;
                     00F6   413
                     00F6   414 CHKBLK1:
50   66 A9    B1     00F6   415          CMPW    IRB$W_RTOTLSZ(R9),R0    ; does record fit?
       F4     1B     00FA   416          BLEQU   CHKJNL_BR               ; branch if yes
     0174     30     00FC   417 N_BLK:   BSBW    RM$PADBLK1              ; pad out block if necc.
  53   01     D0     00FF   418          MOVL    #1,R3                   ; flag no read required
     FEFB'    30     0102   419          BSBW    RM$NXTBLK1              ; call & return from next block routine
    EB 50     E8     0105   420          BLBS    R0,PUT00_BR             ; and continue if ok.
     00E1     31     0108   421          BRW     ZERO_RFA                ; get out on error
```

```
                                010B   423 ;
                                010B   424 ; Size field is ansi_d format, i.e., 4-decimal ascii bytes.
                                010B   425 ;
                                010B   426
                     50    03   D0  010B   427 ANSIZE: MOVL    #3,R0                       ; counter for 4 digits
                52     66 A9   3C  010E   428         MOVZWL  IRB$W_RTOTLSZ(R9),R2        ; get total record size
                          53   D4  0112   429         CLRL    R3                          ; zero extend
             5C    52    52    0A   7B  0114   430 10$:    EDIV    #10,R2,R2,AP                ; divide out next digit
                6140    5C    30   81  0119   431         ADDB3   #^A/0/,AP,(R1)[R0]          ; convert to asci  and store
                         F3 50   F4   011E   432         SOBGEQ  R0,10$                      ; continue if more
                      51    04   C0  0121   433         ADDL2   #4,R1                       ; advance past count
                         FF4B   31   0124   434         BRW     CHKVFC                      ; and continue
```

```
                     0127   436 ;
                     0127   437 ; Process vfc record header.
                     0127   438 ;
                     0127   439
      7E    55   7D  0127   440 VFCRH:  MOVQ    R5,-(SP)                ; save record addr & size
   56   5F AA   9A  012A   441         MOVZBL  IFB$B_FSZ(R10),R6       ; get header length
62 51 AA    02   E1  012E   442         BBC     #FAB$V_PRN,IFB$B_RAT(R10),RHBADR; branch if not print file
                     0133   443
                     0133   444         ASSUME  IMP$W_RMSSTATUS EQ 0
                     0133   445         ASSUME  IMP$V_IIOS EQ 0
                     0133   446
         5F 6B   E8  0133   447         BLBS    (R11),RHBADR            ; branch if image io seg.
                     0136   448
                     0136   449 ;
                     0136   450 ; This is a process-permanent 'print' file.
                     0136   451 ; Use the connect-set rat from isi and convert the specified
                     0136   452 ; carriage control to print file format and store in rhb.
                     0136   453 ;
                     0136   454
            00000016 0136   455         OFF=<RAB$W_ISI*8>+RAB$V_PPF_RAT ; define offset to isi rat
                     0136   456
   5B 68    18   E0  0136   457         BBS     #FAB$V_PRN+OFF,(R8),RHBADR; branch if 'prn'
   24 68    17   E0  013A   458         BBS     #FAB$V_CR+OFF,(R8),1$   ; branch if 'cr'
   29 68    16   E1  013E   459         BBC     #FAB$V_FTN+OFF,(R8),ZERO_RHB; branch if not 'ftn'
                     0142   460
                     0142   461 ;
                     0142   462 ; Fortran carriage control:
                     0142   463 ; interpret fortran carriage control byte in record and convert to prn format.
                     0142   464 ;
                     0142   465
      04 AE    D5  0142   466         TSTL    4(SP)                   ; zero length record?
         1B    13  0145   467         BEQL    1$                      ; branch if yes (lf-rec-cr)
      04 AE    D7  0147   468         DECL    4(SP)                   ; decr size of record
      FE A1    B7  014A   469         DECW    -2(R1)                  ; decr size of record in buffer
07 6A    26   E1  014D   470         BBC     #IFB$V_ANSI_D,(R10),10$ ; branch if not ansi magtape
      66 A9    B7  0151   471         DECW    IRB$W_RTOTLSZ(R9)       ; decrement total record size
         71    D5  0154   472         TSTL    -(R1)                   ; back up to length field
         B3    10  0156   473         BSBB    ANSIZE                  ; store adjusted record length
   50    65    90  0158   474 10$:    MOVB    (R5),R0                 ; get fortran byte
         6E    D6  015B   475         INCL    (SP)                    ; and incr rec addr
      FEA0'    30  015D   476         BSBW    RM$MAPFTN               ; map fortran to pre/post format
         05    11  0160   477         BRB     2$
                     0162   478
                     0162   479 ;
                     0162   480 ; LF - record - CR carriage control required.
                     0162   481 ;
                     0162   482
52   8D01 8F    B0  0162   483 1$:     MOVW    #1+<<128+13>@8>,R2      ; lf-rec-cr in pre/post
  0C A8    52   B0  0167   484 2$:     MOVW    R2,RAB$L_STV(R8)        ; copy carriage ctl to stv area
                     016B   485
                     016B   486 ;
                     016B   487 ; No record header specified so zero it.
                     016B   488 ;
                     016B   489
                     016B   490 ZERO_RHB:
         56    DD  016B   491         PUSHL   R6                      ; save rhb size
                     016D   492 ZERO_RHB1:
```

```
                55   0C A8   DE  016D   493          MOVAL   RAB$L_STV(R8),R5          ; get address of 4 zero bytes
                   04   56   D1  0171   494          CMPL    R6,#4                    ; rhb > 4 bytes?
                        03   1B  0174   495          BLEQU   SETVFC                   ; branch if not
                   56   04   D0  0176   496          MOVL    #4,R6                    ; just move 4 bytes this time
                   6E   56   C2  0179   497 SETVFC: SUBL2   R6,(SP)                  ; adjust remaining count
                      0083   30  017C   498          BSBW    BLDREC                   ; move the zeroes
                     08 50   E9  017F   499          BLBC    R0,30$                   ; branch on error
                   0C A8     D4  0182   500          CLRL    RAB$L_STV(R8)            ; re-zero stv
                   56   6E   D0  0185   501          MOVL    (SP),R6                  ; get remaining count
                        E3   12  0188   502          BNEQ    ZERO_PHB1                ; and continue if not done
                        8E   D5  018A   503 30$:     TSTL    (SP)+                    ; pop temporary count
                        15   11  018C   504          BRB     CHKERR
                             018E   505
                             018E   506 ;
                             018E   507 ; Handle bad record header buffer.
                             018E   508 ;
                             018E   509
                             018E   510 ERRRHB: RMSERR  RHB                       ; change error code
                        57   11  0193   511          BRB     ZERO_RFA                 ; & get out
                             0195   512
                             0195   513 ;
                             0195   514 ; Get address of record header buffer and process it.
                             0195   515 ;
                             0195   516
                55   2C A8   D0  0195   517 RHBADR: MOVL    RAB$L_RHB(R8),R5         ; get address
                        D0   13  0199   518          BEQL    ZERO_RHB                 ; branch if none
                             019B   519          IFNORD  R6,(R5),ERRRHB           ; probe it
                             01A1   520
                             01A1   521 ;
                             01A1   522 ; Move the fixed header and reprobe user buffer.
                             01A1   523 ;
                             01A1   524
                           5F   10  01A1   525          BSBB    BLDREC                   ; just like normal record
                55       8E   7D  01A3   526 CHKERR: MOVQ    (SP)+,R5                 ; restore user buffer regs
                  43 50         E9  01A6   527          BLBC    R0,ZERO_RFA              ; get out on error
                   FE54'        30  01A9   528          BSBW    RM$PROBEREAD             ; reprobe it
                  3D 50         E9  01AC   529          BLBC    R0,ZERO_RFA
                   FEC6         31  01AF   530          BRW     MOVREC                   ; move the data record
```

```
                     01B2    532 ;
                     01B2    533 ; Check for fit if records not allowed to cross block boundaries.
                     01B2    534 ;
                     01B2    535
                     01B2    536 CHKBLK2:
      50 AA    91    01B2    537         CMPB    IFB$B_RFMORG(R10),-     ; fixed format?
         01          01B5    538                 #FAB$C_FIX
         0B    13    01B6    539         BEQL    FIXRFM                  ; branch if yes
   6A    26    E1    01B8    540         BBC     #IFB$V_ANSI_D,(R10),-   ; branch if not ansi
         2D          01BB    541                 PUT01_BR                ;  as there is at least
                     01BC    542                                         ;  the required word left
   04    50    D1    01BC    543         CMPL    R0,#4                   ; ansi d requires 4 bytes min.
         28    18    01BF    544         BGEQ    PUT01_BR                ; a.o.k.
         06    11    01C1    545         BRB     BLK_FULL
                     01C3    546
                     01C3    547 ;
                     01C3    548 ; Make sure there is room for fixed length record.
                     01C3    549 ;
                     01C3    550
52 AA    50    B1    01C3    551 FIXRFM: CMPW    R0,IFB$W_LRL(R10)
         20    1E    01C7    552         BGEQU   PUT01_BR                ; branch if sufficient room
                     01C9    553
                     01C9    554 ;
                     01C9    555 ; This block is full or at least the next record can't possibly
                     01C9    556 ; fit in it, so change to next block.
                     01C9    557 ;
                     01C9    558
                     01C9    559 BLK_FULL:
         1C    E0    01C9    560         BBS     #DEV$V_RND,-            ; branch if disk
         6A          01CB    561                 IFB$L_PRIM_DEV(R10),-
         0F          01CC    562                 B_FULL
      00A3    30    01CD    563         BSBW    RM$PADBLK1              ; pad out buffer
53    01    D0    01D0    564         MOVL    #1,R3                   ; flag no read required
    FE2A'   30    01D3    565         BSBW    RM$NXTBLK1              ; call & return from next block routine
   10 50    E8    01D6    566         BLBS    R0,PUT01_BR             ; continue if ok
      0010  31    01D9    567         BRW     ZERO_RFA                ; process error
   40 A9    D6    01DC    568 B_FULL: INCL    IRB$L_NRP_VBN(R9)       ; bump vbn
   44 A9    B4    01DF    569         CLRW    IRB$W_NRP_OFF(R9)       ; and zero offset
   48 A4    96    01E2    570         INCB    BDB$B_REL_VBN(R4)       ; increment relative vbn
      01    8A    01E5    571         BICB2   #BDB$M_VAL,-            ; make invalid
   0A A4          01E7    572                 BDB$B_FLGS(R4)
                  01E9    573 PUT01_BR:
      FED1  31    01E9    574         BRW     PUT01                   ; rejoin code
```

```
                     01EC    576 ;
                     01EC    577 ; Error during put - zero rfa and exit.
                     01EC    578 ;
                     01EC    579
                     01EC    580 ZERO_RFA:
          10 A8  D4  01EC    581          CLRL    RAB$W_RFA(R8)
          14 A8  B4  01EF    582          CLRW    RAB$W_RFA+4(R8)
                     01F2    583
                     01F2    584 ;
                     01F2    585 ; Restore IRB$L_NRP_VBN and IRB$L_NRP_OFF  to the ebk and ffb values
                     01F2    586 ; since $PUT failed and the next $PUT may succeed.
                     01F2    587 ;
                     01F2    588
40 A9     74 AA  D0  01F2    589          MOVL    IFB$L_EBK(R10),IRB$L_NRP_VBN(R9)
44 A9     5C AA  B0  01F7    590          MOVW    IFB$W_FFB(R10),IRB$W_NRP_OFF(R9)
          62 A9  B4  01FC    591          CLRW    IRB$W_CSIZ(R9)          ; make sure there is no current
                     01FF    592                                          ; record, in case anyone wants to
                     01FF    593                                          ; try an update
           FDFE'  31  01FF    594          BRW     RM$EXRMS
```

```
0202   596 ;++
0202   597 ; BLDREC -- Build Record Routine.
0202   598 ;
0202   599 ; This subroutine moves a record from the user record buffer
0202   600 ; to the rms i/o buffer, crossing block boundaries as needed.
0202   601 ;
0202   602 ; Calling Sequence:
0202   603 ;
0202   604 ;       BSBW    BLDREC
0202   605 ;
0202   606 ; Input Parameters:
0202   607 ;
0202   608 ;       R11     impure area address
0202   609 ;       R10     ifab address
0202   610 ;       R9      irab address
0202   611 ;       R8      rab address
0202   612 ;       R7      end of block address + 1
0202   613 ;       R6      # of bytes in record
0202   614 ;       R5      address of record (source)
0202   615 ;       R1      address in rms i/o buffer (destination)
0202   616 ;
0202   617 ; Implicit Inputs:
0202   618 ;
0202   619 ;       The contents of the various structures,
0202   620 ;       in particular, IRB$L_CURBDB.
0202   621 ;
0202   622 ; Output Parameters:
0202   623 ;
0202   624 ;       R1      address of byte following the moved record
0202   625 ;               in rms i/o buffer
0202   626 ;       R0      status code
0202   627 ;       R2-R6   destroyed
0202   628 ;
0202   629 ; Implicit Outputs:
0202   630 ;
0202   631 ;       BDB$B_FLGS - marked dirty
0202   632 ;       IRB$L_CURBDB - updated if block boundary crossed
0202   633 ;
0202   634 ;       IRB$L_NRP_VBN - updated if block boundary crossed
0202   635 ;       IRB$W_NRP_OFF - updated if block boundary crossed
0202   636 ;
0202   637 ; Completion Codes:
0202   638 ;
0202   639 ;       standard rms.
0202   640 ;
0202   641 ; Side Effects:
0202   642 ;
0202   643 ;       If i/o stall occurs will have changed to
0202   644 ;       running at ast level; reprobing any non-rab
0202   645 ;       user address will be required.
0202   646 ;--
0202   647
0202   648
0202   649 BLDREC:
50  57  51  C3  0202   650         SUBL3   R1,R7,R0                ; get # bytes left in buffer
    56  50  D1  0206   651         CMPL    R0,R6                   ; < record size?
        03  1B  0209   652         BLEQU   20$                     ; branch if so
```

```
                50    56    D0  020B    653              MOVL    R6,R0                           ; no - just use buffer size
                56    50    C2  020E    654 20$:         SUBL2   R0,R6                           ; adjust remaining count
            61  65    50    28  0211    655              MOVC3   R0,(R5),(R1)                    ; move (partial) record to buffer
              54    20 A9    D0  0215    656              MOVL    IRB$L_CURBDB(R9),R4             ; get current bdb
              0A A4    03    88  0219    657              BISB2   #BDB$M_VAL.BDB$M_DRT,BDB$B_FLGS(R4); say valid & dirty
                       56    D5  021D    658              TSTL    R6                              ; done?
                       15    13  021F    659              BEQL    40$                             ; branch if yes
                       51    DD  0221    660              PUSHL   R1                              ; save source addr
                53     01    D0  0223    661              MOVL    #1,R3                           ; flag no read required
                   FDD7'     30  0226    662              BSBW    RM$NXTBLK1                      ; call & return from next block routine
                      55 8ED0  0229    663              POPL    R5                              ; restore source addr
                  0D 50    E9  022C    664              BLBC    R0,50$                          ; get out on error
                   FDCE'     30  022F    665              BSBW    RM$PROBEREAD                    ; reprobe user buffer
                  CD 50    E8  0232    666              BLBS    R0,BLDREC                       ; and go again if no error
                           05  0235    667              RSB
                               0236    668
                               0236    669 ;
                               0236    670 ; Move to buffer is complete.
                               0236    671 ;
                               0236    672
                               0236    673 ;40$:        CMPL    R7,R3                           ; Did we exactly fill the buffer?
                               0236    674 ;            BEQL    60$                             ; If they're equal then yes we did.
                51    53    D0  0236    675 40$:         MOVL    R3,R1                           ; next byte pointer to correct reg.
                               0239    676              RMSSUC                                  ; indicate success
                           05  023C    677 50$:         RSB
                               023D    678
                               023D    679 ;60$:        MOVL    #1,R3                           ; flag no read required
                               023D    680 ;            BSBW    RM$NXTBLK1                      ; call & return from next block routine
                               023D    681 ;            BLBC    R0,50$                          ; get out on error
                               023D    682 ;            BSBW    RM$PROBEREAD                    ; reprobe user buffer
                               023D    683 ;            RSB
```

```
                     023D   685 ;++
                     023D   686 ; Journal record.
                     023D   687 ;
                     023D   688 ; Be sure the record will be written to the current record postion.  The count
                     023D   689 ; field cannot span block boundaries.  So, if the record has a byte_count prefix
                     023D   690 ; (VAR or VFC format), at least two bytes must remain in the buffer.  If there
                     023D   691 ; is not enough room, don't journal the record yet.  The RFA which we have in
                     023D   692 ; hand is not correct.
                     023D   693 ;
                     023D   694 ; ************ NOTE:  When journaling is permitted for magtape devices, the
                     023D   695 ; ************       next section of code will have to be modified.
                     023D   696 ;--
                     023D   697
                     023D   698 ZERO_RFA_BR1:
        FFAC     31  023D   699         BRW     ZERO_RFA
                     0240   700
                     0240   701 SEQJNL: ASSUME  FAB$C_VFC GT FAB$C_VAR
                     0240   702         ASSUME  FAB$C_STM GT FAB$C_VFC
                     0240   703
      64 A9      B5  0240   704         TSTW    IRB$W_ROVHDSZ(R9)       ; Any overhead for the record
         0E      13  0243   705         BEQL    5$                      ; No, then no byte-count prefix
      50 AA      91  0245   706         CMPB    IFB$B_RFMORG(R10),-     ; Stream format?
         04          0248   707                 #FAB$C_STM
         08      1E  0249   708         BGEQU   5$                      ; Yes, stream has no prefix
   02    50      D1  024B   709         CMPL    R0,#2                   ; Room for count?
         03      18  024E   710         BGEQ    5$                      ; Yes, continue
        FE06     31  0250   711         BRW     BUILD                   ; No, get next possible record position
                     0253   712
                     0253   713 ;
                     0253   714 ; The record will be written at the current record position.
                     0253   715 ;
                     0253   716
         03      BB  0253   717 5$:     PUSHR   #^M<R0,R1>              ; Yes, save # bytes in blk, nxt byte ptr
      68 21      E1  0255   718         BBC     #<RAB$V_TPT + ROP>, -   ; Was this the TPT bit set?
         04          0258   719                   (R8), 10$
         1F      DD  0259   720         PUSHL   #RJR$_TPT               ; Yes, operation is $PUT w/ truncate
         02      11  025B   721         BRB     15$                     ; Set up journal entry & write it out
         13      DD  025D   722 10$:    PUSHL   #RJR$_PUT               ; Operation to be journaled is a $PUT
00000000'EF      16  025F   723 15$:    JSB     RM$SEQJNL               ; Journal record
      52 50      D0  0265   724         MOVL    R0, R2                  ; Save the status
      5E 04      C0  0268   725         ADDL2   #4, SP                  ; Remove argument from stack
         03      BA  026B   726         POPR    #^M<R0,R1>              ; Restore # bytes in blk & next byte ptr
        CD 52    E9  026D   727         BLBC    R2, ZERO_RFA_BR1        ; Clean up and exit on error
          FDE6   31  0270   728         BRW     BUILD                   ; rejoin main path
```

```
                     0273   730 ;++
                     0273   731 ; RM$PADBLK1 -- Pad Out Buffer Routine.
                     0273   732 ;
                     0273   733 ; Inputs:
                     0273   734 ;       R1      address in buffer to start padding
                     0273   735 ;       R0      # bytes left in block
                     0273   736 ;       R4      bdb address
                     0273   737 ;       R10     ifab address
                     0273   738 ;
                     0273   739 ; Outputs:
                     0273   740 ;
                     0273   741 ;       Block padded, if needed, and marked dirty
                     0273   742 ;       R0-R3   destroyed
                     0273   743 ;--
                     0273   744
                     0273   745 RM$PADBLK1::
           52   D4  0273   746         CLRL    R2                      ; NUL padding for stream
     53 50 AA   90  0275   747         MOVB    IFB$B_RFMORG(R10),R3    ; get record format (set CCs)
        6A   26   E0  0279   748         BBS     #IFB$V_ANSI_D,(R10),-
              1A       027C   749                 PADANS                  ; branch if ansi d
                     027D   750         CASE    TYPE=B,SRC=R3,-         ; dispatch on rfm
                     027D   751                 LIMIT=#FAB$C_UDF -
                     027D   752                 DISPLIST=<-
                     027D   753                 PADBLK,-                ; UDF (pads like STMx)
                     027D   754                 PADXIT,-                ; FIX
                     027D   755                 PADVAR,-                ; VAR
                     027D   756                 PADVAR,-                ; VFC
                     027D   757                 PADBLK,-                ; STM
                     027D   758                 PADBLK,-                ; STMLF
                     027D   759                 PADBLK>                 ; STMCR
                     028F   760
                     028F   761 ;
                     028F   762 ; Var or vfc record format and not ansi.
                     028F   763 ; Just put a -1 in word pointed at by r1.
                     028F   764 ;
                     028F   765
           61   01   AE  028F   766 PADVAR: MNEGW   #1,(R1)                ; end block flag
                     0292   767
                     0292   768 ;
                     0292   769 ; Flag block dirty and return
                     0292   770 ;
                     0292   771
     0A A4   03   88  0292   772 PADIRT: BISB2   #BDB$M_VAL!BDB$M_DRT,BDB$B_FLGS(R4)
              05      0296   773 PADXIT: RSB
                     0297   774
                     0297   775 ;
                     0297   776 ; pad out the block
                     0297   777 ;
              FD   13  0297   778 PADANS: BEQL    PADXIT                  ; UDF is not padded on tape
           52   5E 8F   90  0299   779         MOVB    #^A\^\,R2               ; set ansi padding
              30   BB  029D   780 PADBLK: PUSHR   #^M<R4,R5>              ; save regs
61   50   52   6E   00   2C  029F   781         MOVC5   #0,(SP),R2,R0,(R1)      ; pad it
              30   BA  02A5   782         POPR    #^M<R4,R5>              ; restore regs
              E9   11  02A7   783         BRB     PADIRT
                     02A9   784
                     02A9   785         .END
```

```
$$.PSECT_EP                     = 00000000
$$RMSTEST                       = 0000001A
$$RMS_PBUGCHK                   = 00000010
$$RMS_TBUGCHK                   = 00000008
$$RMS_UMODE                     = 00000004
ANSIZE                            0000010B R      01
ANSIZE_BR                         00000012 R      01
BDB$B_FLGS                      = 0000000A
BDB$B_REL_VBN                   = 00000048
BDB$L_CURBUFADR                 = 0000004C
BDB$M_DRT                       = 00000002
BDB$M_VAL                       = 00000001
BLDREC                            00000202 R      01
BLK_FULL                          000001C9 R      01
BLK_FULL_BR                       000000EA R      01
BUICD                             00000059 R      01
B_FULL                            000001DC R      01
CHKBLK1                           000000F6 R      01
CHKBLK1_BR                        0000000F R      01
CHKBLK2                           000001B2 R      01
CHKBLK2_BR                        000000ED R      01
CHKERR                            000001A3 R      01
CHKJNL                            00000053 R      01
CHKJNL_BR                         000000F0 R      01
CHKVFC                            00000072 R      01
CR                              = 0000000D
DEV$V_RND                       = 0000001C
DEV$V_SQD                       = 00000005
ERRRHB                            0000018E R      01
FAB$C_FIX                       = 00000001
FAB$C_STM                       = 00000004
FAB$C_STMCR                     = 00000006
FAB$C_STMLF                     = 00000005
FAB$C_UDF                       = 00000000
FAB$C_VAR                       = 00000002
FAB$C_VFC                       = 00000003
FAB$V_BLK                       = 00000003
FAB$V_CR                        = 00000001
FAB$V_FTN                       = 00000000
FAB$V_PRN                       = 00000002
FIXRFA                            000001C3 R      01
IFB$B_FSZ                       = 0000005F
IFB$B_JNLFLG                    = 000000A0
IFB$B_RAT                       = 00000051
IFB$B_RFMORG                    = 00000050
IFB$L_EBK                       = 00000074
IFB$L_PRIM_DEV                  = 00000000
IFB$V_ANSI_D                    = 00000026
IFB$V_RW_ATTR                   = 00000034
IFB$W_FFB                       = 0000005C
IFB$W_LRL                       = 00000052
IMP$V_IIOS                      = 00000000
IMP$W_RMSSTATUS                 = 00000000
IRB$L_CURBDB                    = 00000020
IRB$L_NRP_OFF                   = 00000044
IRB$L_NRP_VBN                   = 00000040
IRB$L_RP_VBN                    = 00000048
IRB$W_CSIZ                      = 00000062
IRB$W_NRP_OFF                   = 00000044
IRB$W_ROVRDSZ                   = 00000064
IRB$W_RP_OFF                    = 0000004C
IRB$W_RTOTLSZ                   = 00000066
LF                              = 0000000A
MOVREC                            00000078 R      01
N_BLK                             000000FC R      01
N_BLK_BR                          000000E6 R      01
OFF                             = 00000016
PADANS                            00000297 R      01
PADBLK                            0000029D R      01
PADIRT                            00000292 R      01
PADVAR                            0000028F R      01
PADXIT                            00000296 R      01
PIO$A_TRACE                       ********   X    01
PUT00                             00000041 R      01
PUT00_BR                          000000F3 R      01
PUT01                             000000BD R      01
PUT01_BR                          000001E9 R      01
RAB$L_RHB                       = 0000002C
RAB$L_ROP                       = 00000004
RAB$L_STV                       = 0000000C
RAB$V_PPF_RAT                   = 00000006
RAB$V_TPT                       = 00000001
RAB$W_ISI                       = 00000002
RAB$W_RFA                       = 00000010
RHBADR                            00000195 R      01
RJR$_PUT                        = 00000013
RJR$_TPT                        = 0000001F
RMS$_RMS                          ********   X    01
RMS$_SUC                          ********   X    01
RMS$GETBLKNRP                     ********   X    01
RMS$MAPFTN                        ********   X    01
RMS$NXTBLK1                       ********   X    01
RMS$PADBLK1                       00000273 RG     01
RMS$PROBEREAD                     ********   X    01
RMS$PUT_BLK_DEV                   00000015 RG     01
RMS$SEQJNL                        ********   X    01
RMS$_RHB                        = 0001866C
ROP                             = 00000020
SEQJNL                            00000240 R      01
SEQJNL_BR                         0000000C R      01
SETSIZ                            00000062 R      01
SETVFC                            00000179 R      01
STM_FMT_DFT                       00000000 R      01
TPT$L_POTBLD                      ********   X    01
VFCRH                             00000127 R      01
VFCRH_BR                          000000E8 R      01
ZERO_RFA                          000001EC R      01
ZERO_RFA_BR                       000000E3 R      01
ZERO_RFA_BR1                      0000023D R      01
ZERO_RFA_BR2                      0000007B R      01
ZERO_RHB                          0000016B R      01
ZERO_RHB1                         0000016D 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
RM$RMS1	000002A9 (681.)	01 (1.)	PIC USR CON REL GBL NOSHR EXE RD NOWRT NOVEC BYTE
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	35	00:00:00.08	00:00:01.43
Command processing	138	00:00:00.76	00:00:04.53
Pass 1	354	00:00:12.39	00:00:33.37
Symbol table sort	0	00:00:01.68	00:00:02.79
Pass 2	136	00:00:02.81	00:00:07.51
Symbol table output	14	00:00:00.10	00:00:00.10
Psect synopsis output	2	00:00:00.03	00:00:00.06
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	681	00:00:17.85	00:00:49.79

The working set limit was 1500 pages.
69827 bytes (137 pages) of virtual memory were used to buffer the intermediate code.
There were 70 pages of symbol table space allocated to hold 1277 non-local and 20 local symbols.
785 source lines were read in Pass 1, producing 15 object records in Pass 2.
26 pages of virtual memory were used to define 25 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[RMS.OBJ]RMS.MLB;1	13
_255DUA28:[SYS.OBJ]LIB.MLB;1	3
_255DUA28:[SYSLIB]STARLET.MLB;2	5
TOTALS (all libraries)	21

1374 GETS were required to define 21 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:RM1PUTBLD/OBJ=OBJ$:RM1PUTBLD MSRC$:RM1PUTBLD/UPDATE=(ENH$:RM1PUTBLD)+EXECML$/LIB+LIB$:RMS/LIB
```

RM1PUTREC
LIS

RM1PUTSET
LIS

RM1UPDATE
LIS

RM1NXTBLK
LIS

RM1PUTBLD
LIS

RM1RELBLK
LIS

RM1SEQXFR
LIS

RM1PUT
LIS

RM2CONN
LIS

RM1OPEN
LIS

RM1WTLST
LIS

RM1STMFMT
LIS