```
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
```

```
RRRRRRRR    MM      MM    SSSSSSSS    000000    MM      MM    IIIIII      SSSSSSSS    CCCCCCCC
RRRRRRRR    MM      MM    SSSSSSSS    000000    MM      MM    IIIIII      SSSSSSSS    CCCCCCCC
RR      RR  MMMM  MMMM  SS          00      00  MMMM  MMMM      II      SS          CC
RR      RR  MMMM  MMMM  SS          00      00  MMMM  MMMM      II      SS          CC
RR      RR  MM  MM  MM  SS          00    0000  MM  MM  MM      II      SS          CC
RR      RR  MM  MM  MM  SS          00    0000  MM  MM  MM      II      SS          CC
RRRRRRRR    MM      MM    SSSSSS    00  00  00  MM      MM      II        SSSSSS    CC
RRRRRRRR    MM      MM    SSSSSS    00  00  00  MM      MM      II        SSSSSS    CC
RR  RR      MM      MM          SS  0000    00  MM      MM      II              SS  CC
RR  RR      MM      MM          SS  0000    00  MM      MM      II              SS  CC
RR    RR    MM      MM          SS  00      00  MM      MM      II              SS  CC                ....
RR    RR    MM      MM          SS  00      00  MM      MM      II              SS  CC                ....
RR      RR  MM      MM  SSSSSSSS      000000    MM      MM    IIIIII    SSSSSSSS      CCCCCCCC      ....
RR      RR  MM      MM  SSSSSSSS      000000    MM      MM    IIIIII    SSSSSSSS      CCCCCCCC      ....

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
```

Table of contents

```
0000     1          $BEGIN  RMSOMISC,000,RM$RMS,<$FREE,$RELEASE,$FLUSH>
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  ;
0000    32  ;       This module provides the high level control for the
0000    33  ;       $FREE, $RELEASE, and $FLUSH RMS services.
0000    34  ;
0000    35  ; Environment:
0000    36  ;               Star processor running Starlet exec.
0000    37  ;
0000    38  ; Author: L. F. Laverdure        Creation Date: 25-OCT-1977
0000    39  ;
0000    40  ; Modified By:
0000    41  ;
0000    42  ;       V03-013 RAS0310         Ron Schaefer            14-Jun-1984
0000    43  ;               Fix a broken branch.
0000    44  ;
0000    45  ;       V03-012 DGB0013         Donald G. Blair         01-Mar-1984
0000    46  ;               Make changes related to ACP calls as part of the
0000    47  ;               restructuring necessary to implement access mode
0000    48  ;               protected files.
0000    49  ;
0000    50  ;       V03-011 JWT0159         Jim Teague              28-Feb-1983
0000    51  ;               Add an alternate entry point to RM$FLUSH.  If the
0000    52  ;               caller desires to invalidate the cache, then the
0000    53  ;               entry point should be RM$FLUSH_ALT.  Otherwise,
0000    54  ;               buffers are not invalidated.
0000    55  ;
0000    56  ;       V03-010 RAS0206         Ron Schaefer            26-Oct-1983
0000    57  ;               Change $FLUSH to invalidate all buffers.  This is a
```

```
0000     58 ;
0000     59 ;               temporary measure to make RU cancel/rollback work properly
0000     60 ;               until RU cancel can be modified to have a private version
0000     61 ;               of $FLUSH.
0000     62 ;
0000     63 ;       V03-009 SHZ0007         Stephen H. Zalewski,    25-Mar-1983
0000     64 ;               Do not attempt to write the header attributes of a file to
0000     65 ;               disk if the file is not write accessed.
0000     66 ;
0000     67 ;       V03-008 SHZ0006         Stephen H. Zalewski,    21-Jan-1983
0000     68 ;               Fixed undefined symbol FHCLEN.
0000     69 ;
0000     70 ;       V03-007 SHZ0005         Stephen H. Zalewski,    17-Jan-1983
0000     71 ;               Check ifb$v_rw_attr bit in ifb to see if $FLUSH should
0000     72 ;               write the file header out to disk.  This is only helpful
0000     73 ;               for sequential files where the FFB field in the file header
0000     74 ;               changed, but the hbk and ebk fields did not.
0000     75 ;
0000     76 ;       V03-006 SHZ0004         Stephen H. Zalewski,    17-Jan-1983  15:58
0000     77 ;               Modified $FLUSH so that a file header is only written if
0000     78 ;               the hbk or ebk of the file has changed.  This prevents
0000     79 ;               unnecessary writes of the file header to disk.
0000     80 ;
0000     81 ;       V03-005 SHZ0003         Stephan H. Zalewski,    11-Dec-1983
0000     82 ;               Fix bug that caused ORGCASE to be picked up from IRB
0000     83 ;               instead of IFB when doing a $FLUSH.
0000     84 ;
0000     85 ;       V03-004 SHZ0002         Stephen H. Zalewski,    5-Dec-1982
0000     86 ;               Change entry point RM$WRITE_ATTR:: to WRITE_ATTR:.  Allow
0000     87 ;               flush to write out the header for any file (used to only
0000     88 ;               flush nonshared sequential files).
0000     89 ;
0000     90 ;       V03-003 SHZ0001         Stephen H. Zalewski,    16-Dec-1982  5:07
0000     91 ;               Remove assume statements for locations of hbk and ebk in ifb.
0000     92 ;
0000     93 ;       V03-002 KBT0185         Keith B. Thompson       23-Aug-1982
0000     94 ;               Reorganize psects
0000     95 ;
0000     96 ;       V03-001 KBT0103         Keith B. Thompson       13-Jul-1982
0000     97 ;               Clean up psects
0000     98 ;--
0000     99 ;
0000    100
```

```
                     0000   102          .SBTTL  DECLARATIONS
                     0000   103
                     0000   104 ;
                     0000   105 ; Include Files:
                     0000   106 ;
                     0000   107
                     0000   108 ;
                     0000   109 ; Macros:
                     0000   110 ;
                     0000   111
                     0000   112          $FABDEF
                     0000   113          $RABDEF
                     0000   114          $FIBDEF
                     0000   115          $DEVDEF
                     0000   116          $BDBDEF
                     0000   117          $IFBDEF
                     0000   118          $IRBDEF
                     0000   119          $IODEF
                     0000   120          $ATRDEF
                     0000   121          $RLSDEF
                     0000   122          $RMSDEF
                     0000   123
                     0000   124 ;
                     0000   125 ; Equated Symbols:
                     0000   126 ;
                     0000   127
00000020             0000   128          BKP     = IRB$L_BKPBITS*8                ; bit offset to flags
00000060             0000   129          LSTMSK  = <1@<IRB$V_FIND_LAST-BKP>>-
                     0000   130                    !<1@<IRB$V_PUTS_LAST-BKP>>     ; mask for resetting irab status
00000016             0000   131          FHCLEN  = IFB$C_FHAEND - IFB$B_RFMORG    ; file header characteristics length
                     0000   132
                     0000   133 ;
                     0000   134 ; Own Storage:
                     0000   135 ;
                     0000   136
```

```
0000    138          .SBTTL  MISCELLANEOUS RECORD SERVICE ENTRY POINTS
0000    139
0000    140 ;++
0000    141 ; Functional Description:
0000    142 ;
0000    143 ;       This module provides the high level control routines
0000    144 ;       for the miscellaneous rab functions $free, $release, and
0000    145 ;       $flush. see individual descriptions.
0000    146 ;
0000    147 ; Calling Sequence:
0000    148 ;
0000    149 ;       Entered as a result of user's calling SYS$FREE,
0000    150 ;       SYS$RELEASE, SYS$FLUSH.
0000    151 ;
0000    152 ; Input Parameters:
0000    153 ;
0000    154 ;       AP      user's argument list
0000    155 ;
0000    156 ; Implicit Inputs:
0000    157 ;
0000    158 ;       The contents of the rab.
0000    159 ;
0000    160 ; Output Parameters:
0000    161 ;
0000    162 ;       R0      status code
0000    163 ;       R1      destroyed
0000    164 ;
0000    165 ; Implicit Outputs:
0000    166 ;
0000    167 ;       The sts and stv fields of the rab are output by all of
0000    168 ;       these services. See individual service descriptions
0000    169 ;       for additional rab outputs.
0000    170 ;
0000    171 ; Completion Codes:
0000    172 ;
0000    173 ;       Standard rms
0000    174 ;
0000    175 ; Side Effects:
0000    176 ;
0000    177 ;       none
0000    178 ;
0000    179 ;--
0000    180
```

```
                     0000   182
                     0000   183 ;++
                     0000   184 ;
                     0000   185 ; Entry point for $free.
                     0000   186 ;
                     0000   187 ; Call RM$UNLOCKALL to release locks on all records.
                     0000   188 ;
                     0000   189 ;--
                     0000   190
                     0000   191         $ENTRY  RMS$FREE
                     0000   192         $TSTPT  FREE
                     0006   193         $RABSET                             ; set up stream
33 6A   3E   E0      000A   194         BBS     #IFB$V_DAP,(R10),NTFREE     ; branch if network operation
     FFEF'   30      000E   195         BSBW    RM$UNLOCKALL                ; free all locked records
       2B    11      0011   196         BRB     EXIT                        ; all set
                     0013   197
                     0013   198 ;++
                     0013   199 ;
                     0013   200 ; Entry point for $release.
                     0013   201 ;
                     0013   202 ; Call RM$UNLOCK to release lock on record specified by
                     0013   203 ; the rfa field of the rab.
                     0013   204 ;
                     0013   205 ;--
                     0013   206
                     0013   207         $ENTRY  RMS$RELEASE
                     0013   208         $TSTPT  RELEASE0
                     0019   209         $RABSET                             ; set up stream
25 6A   3E   E0      001D   210         BBS     #IFB$V_DAP,(R10),NTREL      ; branch if network operation
51   10 A8   D0      0021   211         MOVL    RAB$W_RFA(R8),R1            ; get rfa
52   14 A8   3C      0025   212         MOVZWL  RAB$W_RFA+4(R8),R2          ; and zero extend last word
     FFD4'   30      0029   213         BSBW    RM$UNLOCK                   ; release lock on record
       10    11      002C   214         BRB     EXIT
                     002E   215
                     002E   216 ;++
                     002E   217 ;
                     002E   218 ; Entry point for $FLUSH.
                     002E   219 ;
                     002E   220 ; Call internal flush routine.
                     002E   221 ;
                     002E   222 ;--
                     002E   223
                     002E   224         $ENTRY  RMS$FLUSH
                     002E   225         $TSTPT  FLUSH
                     0034   226         $RABSET                             ; set up stream
0F 6A   3E   E0      0038   227         BBS     #IFB$V_DAP,(R10),NTFLUSH    ; branch if network operation
       17    10      003C   228         BSBB    RM$FLUSH                    ; do flush
     FFBF'   31      003E   229 EXIT:   BRW     RM$EXRMS
                     0041   230
                     0041   231 ;++
                     0041   232 ;
                     0041   233 ; Perform network functions.
                     0041   234 ;
                     0041   235 ;--
                     0041   236
                     0041   237 NTFREE:
     FFBC'   30      0041   238         BSBW    NT$FREE                     ; perform function via
```

```
                0044   239          BRB     EXIT                    ;  remote fal
                0046   240
                0046   241 NTREL:
FFB7'   30      0046   242          BSBW    NT$RELEASE              ; perform function via
  F3    11      0049   243          BRB     EXIT                    ;  remote fal
                004B   244
                004B   245 NTFLUSH:
FFB2'   30      004B   246          BSBW    NT$FLUSH                ; perform function via
  EE    11      004E   247          BRB     EXIT                    ;  remote fal
                0050   248
```

```
0050   250
0050   251          .SBTTL  RM$FLUSH - COMPLETE ACTIVITY ON STREAM
0050   252
0050   253 ;++
0050   254 ; RM$FLUSH -- Complete Activity on Stream.
0050   255 ;
0050   256 ;       This routine performs the following steps to guarantee
0050   257 ;       that all modified i/o buffers are written back to the
0050   258 ;       file and the files attributes are written back to the
0050   259 ;       file header.
0050   260 ;
0050   261 ;       1.  If sequential file org, call RM$WTLST1.
0050   262 ;
0050   263 ;       2.  Get the bdb list head.
0050   264 ;
0050   265 ;       3.  Get next bdb, if no more call RM$WRITE_ATTR, clear the
0050   266 ;           FIND_LAST, PUTS_LAST, and BIO_LAST IRAB bookkeeping
0050   267 ;           bits and exit.
0050   268 ;
0050   269 ;       4.  If bdb not dirty, go to 3.
0050   270 ;
0050   271 ;       5.  Call cache to get exclusive access (lock) to the buffer.
0050   272 ;
0050   273 ;       6.  Release the buffer with write through.
0050   274 ;
0050   275 ;       7.  Go to step 2.
0050   276 ;
0050   277 ;    RM$FLUSH_ALT does basically the same except it invalidates all
0050   278 ;    buffers as it steps through the BDBs.
0050   279 ;
0050   280 ; Calling Sequence:
0050   281 ;
0050   282 ;       BSBW    RM$FLUSH or RM$FLUSH_ALT
0050   283 ;
0050   284 ; Input Parameters:
0050   285 ;
0050   286 ;       R11     impure area address
0050   287 ;       R10     ifab address
0050   288 ;       R9      irab address
0050   289 ;       R8      rab address
0050   290 ;
0050   291 ; Implicit Inputs:
0050   292 ;
0050   293 ;       The contents of the ifab, irab, and rab.
0050   294 ;
0050   295 ; Output Parameters:
0050   296 ;
0050   297 ;       R0              status code
0050   298 ;       R1-R7,AP        destroyed
0050   299 ;
0050   300 ; Implicit Outputs:
0050   301 ;
0050   302 ;       none.
0050   303 ;
0050   304 ; Side Effects:
0050   305 ;
0050   306 ;       For the sequential file organization, guarantees that any
```

```
0050   307 ;       write behinds have terminated.
0050   308 ;
0050   309 ;       For the other organizations, $flush has a possibility of
0050   310 ;       never terminating if other streams continually leave
0050   311 ;       around dirty buffers. If this is a problem for a given
0050   312 ;       application, some other mechanism is required in addition
0050   313 ;       to flush to guarantee a quiet point. Note that these
0050   314 ;       problems only occur with the deferred write option.
0050   315 ;--
0050   316
```

```
                     0050   318 RM$FLUSH_ALT::
        56   01   D0  0050   319         MOVL    #1,R6           ; Set flag to invalidate buffers
             02   11  0053   320         BRB     FLUSH           ;   and proceed
                      0055   321
                      0055   322 RM$FLUSH::
        56   D4       0055   323         CLRL    R6              ; don't invalidate buffers
04 A9   60   8F   8A  0057   324 FLUSH:  BICB    #LSTMSK,IRB$L_BKPBITS(R9); reset irab flags
                      005C   325
                      005C   326         ASSUME  FAB$C_SEQ EQ 0
                      005C   327
        23   AA   95  005C   328         TSTB    IFB$B_ORGCASE(R10)      ; seq. file org?
             15   12  005F   329         BNEQ    SCAN_BDB_LIST           ; branch if not
10 22 AA     05   E0  0061   330         BBS     #IFB$V_BIO,IFB$B_FAC(R10),10$; branch if block i/o
   0C 69     27   E4  0066   331         BBSC    #IRB$V_BIO_LAST,(R9),10$; or last op was block i/o
                      006A   332                                 ;  (i.e., mixed block and record op)
 00000000'EF     16   006A   333         JSB     RM$WTLST1               ; write out partial buffer
        62   A9   B4  0070   334         CLRW    IRB$W_CSIZ(R9)          ; say no current record
        47   50   E9  0073   335         BLBC    R0,FLXIT                ; get out on error
                      0076   336 10$:
                      0076   337
                      0076   338 ;
                      0076   339 ; Scan bdb list for any dirty buffers and, if found, write them
                      0076   340 ; out. If any have a write in progress, wait for it to finish.
                      0076   341 ; Cache is called to access the buffer so that all necessary
                      0076   342 ; interlocking is done correctly there.
                      0076   343 ;
                      0076   344
                      0076   345 SCAN_BDB_LIST:
     54   40 AA   DE  0076   346         MOVAL   IFB$L_BDB_FLNK(R10),R4  ; get bdb list head addr
                      007A   347
                      007A   348         ASSUME  BDB$L_FLINK EQ 0
                      007A   349
        54   64   D0  007A   350 NXTBDB: MOVL    (R4),R4                 ; get next bdb addr
   40 AA     64   D1  007D   351         CMPL    (R4),IFB$L_BDB_FLNK(R10); is this the list head?
             3B   13  0081   352         BEQL    WRITE_ATTR              ; branch if yes
26 0A A4     01   E1  0083   353 CHKDRT: BBC     #BDB$V_DRT,BDB$B_FLGS(R4),CLEAR_VAL; branch if buffer not dirty
                      0088   354
                      0088   355 ;
                      0088   356 ; Buffer is dirty.  Deferred write may be in use or another stream may
                      0088   357 ; currently have the buffer accessed.  At any rate, get exclusive access
                      0088   358 ; to the buffer, then release with write-through.
                      0088   359 ;
                      0088   360
                      0088   361         $CSHFLAGS       LOCK            ; say we want lock
        51   1C A4   D0  008B   362         MOVL    BDB$L_VBN(R4), R1       ; get vbn for this bdb
        52   14 A4   3C  008F   363         MOVZWL  BDB$W_NUMB(R4), R2      ; get size of this buffer.
             FF6A'   30  0093   364         BSBW    RM$CACHE                ; access the bucket.
             14 50   E9  0096   365         BLBC    R0, ERRX                ; and go write it.
             05 56   E9  0099   366         BLBC    R6,15$                  ; If no invalidate, don't do it
        53   08   D0  009C   367         MOVL    #RLS$M_DEQ,R3           ; flag to write and invalidate
             03   11  009F   368         BRB     20$
        53   02   D0  00A1   369 15$:     MOVL    #RLS$M_WRT_THRU,R3      ; flag to cause write only
 00000000'EF     16   00A4   370 20$:     JSB     RM$RELEASE              ; and write bucket out
        C9   50   E8  00AA   371         BLBS    R0,SCAN_BDB_LIST        ; and branch if o.k.
                  05  00AD   372 ERRX:   RSB                             ; get out on error
                      00AE   373 CLEAR_VAL:
        C9   56   E9  00AE   374         BLBC    R6,NXTBDB               ; If no invalidate, get next BDB
```

```
                0A A4   01   8A  00B1   375          BICB2   #BDB$M_VAL,BDB$B_FLGS(R4) ;  else clear valid bit, then
                        C3   11  00B5   376          BRB     NXTBDB                     ;    get next BDB
```

```
                     00B7   378
                     00B7   379 ;
                     00B7   380 ;  Clear nrp offset on magtape flush and avoid rewrite of file attributes.
                     00B7   381 ;
                     00B7   382
 44 A9    B4         00B7   383 CLRNRP: CLRW    IRB$W_NRP_OFF(R9)       ; get start of next block
                     00BA   384
                     00BA   385 SUCXIT: RMSSUC                           ; show success
                  05 00BD   386 FLXIT:  RSB
```

RMS
Sym

```
$$.
$$R
$$R
$$R
$$R
FAB
FAB
FAB
FAB
FAB
FAB
FAB
FAB
FAB
FOP
IFB
IFB
IFB
IFB
IFB
IFB
MOD
PPF
PSL
RMS
RMS
RME
RMS
RMS
RMS
RMS
SET


PSE
---

RMS
$AB


Pha
---
Ini
Com
Pas
Sym
Pas
Sym
Pse
Cro
```

```
                                00BE   388          .SBTTL  WRITE_ATTR - REWRITE FILE ATTRIBUTES
                                00BE   389
                                00BE   390 ;++
                                00BE   391 ; WRITE_ATTR -- Rewrite File Attributes.
                                00BE   392 ;
                                00BE   393 ; This subroutine examines the current hbk and ebk marks for the file,
                                00BE   394 ; and if either of them has changed,  rewrites the file attributes.
                                00BE   395 ;
                                00BE   396 ; Acp write through is forced by setting FIB$V_WRITETHRU.
                                00BE   397 ;
                                00BE   398 ; Inputs:
                                00BE   399 ;
                                00BE   400 ;       R11             impure area address
                                00BE   401 ;       R10             ifab address
                                00BE   402 ;       R9              ifab or irab address
                                00BE   403 ;
                                00BE   404 ; Outputs:
                                00BE   405 ;
                                00BE   406 ;       R0              status code
                                00BE   407 ;       R1-R6,AP        destroyed
                                00BE   408 ;--
                                00BE   409
                                00BE   410 WRITE_ATTR:
          F5 6A    05    E0    00BE   411          BBS     #DEV$V_SQD,IFB$L_PRIM_DEV(R10),CLRNRP; branch if magtape
          F1 6A    03    E1    00C2   412          BBC     #DEV$V_DIR,IFB$L_PRIM_DEV(R10),CLRNRP; or if non-directory device
          F0 6A    30    E1    00C6   413          BBC     #IFB$V_WRTACC,(R10),SUCXIT      ; Exit with success if no write acce
          16 6A    34    E4    00CA   414          BBSC    #IFB$V_RW_ATTR,(R10),5$         ; Write header if this bit set.
    50    58 AA    10    9C    00CE   415          ROTL    #16,IFB$L_EBK_DISK(R10),R0      ; Get old ebk mark
          74 AA    50    D1    00D3   416          CMPL    R0,IFB$L_EBK(R10)               ; Has it changed?
                   0B    12    00D7   417          BNEQ    5$                              ; Yes, write the header
    50    54 AA    10    9C    00D9   418          ROTL    #16,IFB$L_HBK_DISK(R10),R0      ; Get old hbk mark
          70 AA    50    D1    00DE   419          CMPL    R0,IFB$L_HBK(R10)               ; Has this changed?
                   D6    13    00E2   420          BEQL    SUCXIT                          ; No, then exit with success
                               00E4   421
                               00E4   422 ;
                               00E4   423 ; Build attribute list to rewrite record attributes.
                               00E4   424 ; Allocate space for the record attributes and move them there from
                               00E4   425 ; the ifab, converting them to their on disk format.
                               00E4   426 ;
                               00E4   427
          52    56 8F    9A    00E4   428 5$:      MOVZBL  #FHCLEN+FIB$C_LENGTH,R2 ; size of rec. attr. + fib
                FF15'    30    00E8   429          BSBW    RM$GETSPC1              ; get the space
                64 50    E9    00EB   430          BLBC    R0,DONE                 ; get out on error
                   7E    D4    00EE   431          CLRL    -(SP)                   ; end of attribute list
                   51    DD    00F0   432          PUSHL   R1                      ; push addr of allocated space
          00040016 8F    DD    00F2   433          PUSHL   #<ATR$C_RECATTR@16>+FHCLEN; and set attr. code and length
                               00F8   434
                               00F8   435          ASSUME  IFB$B_RAT EQ IFB$B_RFMORG+1
                               00F8   436          ASSUME  IFB$W_LRL EQ IFB$B_RFMORG+2
                               00F8   437          ASSUME  <IFB$C_SEQ + 1> EQ IFB$C_REL
                               00F8   438
          61    50 AA    D0    00F8   439          MOVL    IFB$B_RFMORG(R10),(R1)  ; copy rfm, rat, lrl
          50    23 AA    90    00FC   440          MOVB    IFB$B_ORGCASE(R10), R0  ; move orgcase into R0
          02 6A    38    E1    0100   441          BBC     #IFB$V_SEQFIL,(R10),10$ ; if this is really a sequential
                   50    97    0104   442          DECB    R0                      ; file then turn it back into one.
61  04    04     50    F0    0106   443 10$:      INSV    R0,#IFB$V_ORG,#IFB$S_ORG,(R1); insert org
                   81    D5    010B   444          TSTL    (R1)+                   ; move to next field
```

```
                                 010D   445
                                 010D   446          ASSUME  IFB$L_HBK_DISK EQ IFB$B_RFMORG+4
                                 010D   447          ASSUME  IFB$L_EBK_DISK EQ IFB$B_RFMORG+8
                                 010D   448
        54 AA   70 AA    10   9C  010D   449          ROTL    #16,IFB$L_HBK(R10),IFB$L_HBK_DISK(R10)  ; switch words of hbk
                81   54 AA   D0  0113   450          MOVL    IFB$L_HBK_DISK(R10),(R1)+               ; copy disk structured hbk
        58 AA   74 AA    10   9C  0117   451          ROTL    #16,IFB$L_EBK(R10),IFB$L_EBK_DISK(R10)  ; switch words of ebk
                81   58 AA   D0  011D   452          MOVL    IFB$L_EBK_DISK(R10),(R1)+               ; copy disk-structured ebk
                                 0121   453
                                 0121   454          ASSUME  <IFB$W_GBC-IFB$W_FFB> EQ 8
                                 0121   455          ASSUME  IFB$W_FFB EQ IFB$L_EBK_DISK+4
                                 0121   456          ASSUME  IFB$C_FHAEND EQ <IFB$W_GBC+2>
                                 0121   457
                81   5C AA   7D  0121   458          MOVQ    IFB$W_FFB(R10),(R1)+    ; copy record attributes to GBC
                81   64 AA   B0  0125   459          MOVW    IFB$W_GBC(R10), (R1)+   ; copy GBC
                                 0129   460
                                 0129   461          ASSUME  FIB$L_ACCTL EQ 0
                                 0129   462
                                 0129   463          SSB     #FIB$V_WRITETHRU,(R1)   ; force write thru
                     51     DD  012C   464          PUSHL   R1                      ; push addr of fib
                7E     40 8F   9A  012F   465          MOVZBL  #FIB$C_LENGTH,-(SP)     ; and length of fib
                   50   36     9A  0133   466          MOVZBL  #IO$_MODIFY,R0          ; specify modify i/o function
                        00     DD  0136   467          PUSHL   #0                      ; p6 = 0 for qio
                     0C AE     DF  0138   468          PUSHAL  12(SP)                  ; p5 = address of attribute list
                      FEC2'    30  013B   469          BSBW    RMSFCPFNC_P4            ; call acp to write attributes
                   5E   0C     C0  013E   470          ADDL2   #12,SP                  ; clean fib descriptor off stack
                                 0141   471
                                 0141   472 ;
                                 0141   473 ; And rest of stack clean.
                                 0141   474 ;
                                 0141   475
                  54   8E     7D  0141   476          MOVQ    (SP)+,R4                ; addr of space to deallocate
            52    56 8F     9A  0144   477          MOVZBL  #FHCLEN+FIB$C_LENGTH,R2 ; length of space to deallocate
                     50     DD  0148   478          PUSHL   R0                      ; save status from write attr
                   FEB3'    30  014A   479          BSBW    RMSRETSPC1              ; deallocate the space
                     01     BA  014D   480          POPR    #^M<R0>                 ; restore status code
                  01 50     E9  014F   481          BLBC    R0,RWERR                ; branch on error
                            05  0152   482 DONE:    RSB
                                 0153   483
                                 0153   484 ;
                                 0153   485 ; Map error code from qio to rms code.
                                 0153   486 ;
                                 0153   487
                                 0153   488 RWERR:
                                 0153   489          RMSERR  WER,R1                  ; default error code
                   FEA5'     31  0158   490          BRW     RMSMAPERR               ; map the error code
                                 015B   491
                                 015B   492          .END
```

```
$$.PSECT_EP                     = 00000000
$$.TMP                          = 00000001
$$RMSTEST                       = 0000001A
$$RMS_PBUGCHK                   = 00000010
$$RMS_TBUGCHK                   = 00000008
$$RMS_UMODE                     = 00000004
ATR$C_RECATTR                   = 00000004
BDB$B_FLGS                      = 0000000A
BDB$L_FLINK                     = 00000000
BDB$L_VBN                       = 0000001C
BDB$M_VAL                       = 00000001
BDB$V_DRT                       = 00000001
BDB$W_NUMB                      = 00000014
BKP                             = 00000020
CHKDRT                            00000083 R      01
CLEAR_VAL                         000000AE R      01
CLRNRP                            000000B7 R      01
CSH$M_LOCK                      = 00000001
CSH$M_NOBUFFER                  = 00000008
DEV$V_DIR                       = 00000003
DEV$V_SQD                       = 00000005
DONE                              00000152 R      01
ERRX                              000000AD R      01
EXIT                              0000003E R      01
FAB$C_SEQ                       = 00000000
FHCLEN                          = 00000016
FIB$C_LENGTH                    = 00000040
FIB$L_ACCTL                     = 00000000
FIB$V_WRITETHRU                 = 00000013
FLUSH                             00000057 R      01
FLXIT                             000000BD R      01
IFB$B_FAC                       = 00000022
IFB$B_ORGCASE                   = 00000023
IFB$B_RAT                       = 00000051
IFB$B_RFMORG                    = 00000050
IFB$C_FHAEND                    = 00000066
IFB$C_REL                       = 00000001
IFB$C_SEQ                       = 00000000
IFB$L_BDB_FLNK                  = 00000040
IFB$L_EBK                       = 00000074
IFB$L_EBK_DISK                  = 00000058
IFB$L_HBK                       = 00000070
IFB$L_HBK_DISK                  = 00000054
IFB$L_PRIM_DEV                  = 00000000
IFB$S_ORG                       = 00000004
IFB$V_BIO                       = 00000005
IFB$V_DAP                       = 0000003E
IFB$V_ORG                       = 00000004
IFB$V_RW_ATTR                   = 00000034
IFB$V_SEQFIL                    = 00000038
IFB$V_WRTACC                    = 00000030
IFB$W_FFB                       = 0000005C
IFB$W_GBC                       = 00000064
IFB$W_LRL                       = 00000052
IO$_MODIFY                      = 00000036
IRB$L_BKPBITS                   = 00000004
IRB$V_BIO_LAST                  = 00000027
IRB$V_FIND_LAST                 = 00000025
IRB$V_PUTS_LAST                 = 00000026
IRB$W_CSIZ                      = 00000062
IRB$W_NRP_OFF                   = 00000044
LSTMSR                          = 00000060
NT$FLUSH                          ********    X   01
NT$FREE                           ********    X   01
NT$RELEASE                        ********    X   01
NTFLUSH                           0000004B R      01
NTFREE                            00000041 R      01
NTREL                             00000046 R      01
NXTBDB                            0000007A R      01
PIO$A_TRACE                       ********    X   01
RAB$W_RFA                       = 00000010
RLS$M_DEQ                       = 00000008
RLS$M_WRT_THRU                  = 00000002
RM$CACHE                          ********    X   01
RM$EXRMS                          ********    X   01
RM$FCPFNC_P4                      ********    X   01
RM$FLUSH                          00000055 RG     01
RM$FLUSH_ALT                      00000050 RG     01
RM$GETSPC1                        ********    X   01
RM$MAPERR                         ********    X   01
RM$RELEASE                        ********    X   01
RM$RETSPC1                        ********    X   01
RM$RSET                           ********    X   01
RM$UNLOCK                         ********    X   01
RM$UNLOCKALL                      ********    X   01
RM$WTLST1                         ********    X   01
RMS$FLUSH                       = 0000002C RG     01
RMS$FREE                        = FFFFFFFE RG     01
RMS$RELEASE                     = 00000011 RG     01
RMS$_WER                        = 0001C114
RWERR                             00000153 R      01
SCAN_BDB_LIST                     00000076 R      01
SUCXIT                            000000BA R      01
TPT$L_FLUSH                       ********    X   01
TPT$L_FREE                        ********    X   01
TPT$L_RELEASE0                    ********    X   01
WRITE_ATTR                        000000BE 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
RMS$RMS	0000015B (347.)	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	29	00:00:00.06	00:00:00.63
Command processing	108	00:00:00.81	00:00:05.56
Pass 1	418	00:00:15.30	00:00:37.48
Symbol table sort	0	00:00:02.41	00:00:03.60
Pass 2	100	00:00:02.84	00:00:05.24
Symbol table output	13	00:00:00.15	00:00:00.66
Psect synopsis output	2	00:00:00.02	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	672	00:00:21.60	00:00:53.36

The working set limit was 1500 pages.
86410 bytes (169 pages) of virtual memory were used to buffer the intermediate code.
There were 90 pages of symbol table space allocated to hold 1714 non-local and 6 local symbols.
492 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
_255DUA28:[RMS.OBJ]RMS.MLB;1	17
_255DUA28:[SYS.OBJ]LIB.MLB;1	0
_255DUA28:[SYSLIB]STARLET.MLB;2	8
TOTALS (all libraries)	25

1852 GETS were required to define 25 macros.

There were no errors, warnings or information messages.

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