C 13

```
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRRRRRRRRRRR     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  SSSSSSSSSSSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
```

```
_$2

Sym
---
NT$
NT$
NT$
NT$
NT$
NT$


NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$

NT$

NT$
NT$
NT$
NT$
NT$
NT$
NT$



NT$
NT$
NT$
NT$
NT$
NT$
PI(
**FILE**ID**RMOACCESS

RRRRRRRR    MM      MM    000000      AAAAAA      CCCCCCCC    CCCCCCCC  EEEEEEEEEE    SSSSSSSS    SSSSSSSS
RRRRRRRR    MM      MM    000000      AAAAAA      CCCCCCCC    CCCCCCCC  EEEEEEEEEE    SSSSSSSS    SSSSSSSS
RR      RR  MMMM  MMMM  00      00  AA      AA  CC          CC          EE          SS          SS
RR      RR  MMMM  MMMM  00      00  AA      AA  CC          CC          EE          SS          SS
RR      RR  MM  MM  MM  00    0000  AA      AA  CC          CC          EE          SS          SS
RR      RR  MM  MM  MM  00    0000  AA      AA  CC          CC          EE          SS          SS
RRRRRRRR    MM      MM  00  00  00  AA      AA  CC          CC          EEEEEEEE      SSSSSS        SSSSSS
RRRRRRRR    MM      MM  00  00  00  AA      AA  CC          CC          EEEEEEEE      SSSSSS        SSSSSS
RR  RR      MM      MM  0000    00  AAAAAAAAAA  CC          CC          EE                  SS          SS
RR  RR      MM      MM  0000    00  AAAAAAAAAA  CC          CC          EE                  SS          SS
RR    RR    MM      MM  00      00  AA      AA  CC          CC          EE                  SS          SS      ....
RR    RR    MM      MM  00      00  AA      AA  CC          CC          EE                  SS          SS      ....
RR      RR  MM      MM    000000    AA      AA    CCCCCCCC    CCCCCCCC  EEEEEEEEEE  SSSSSSSS    SSSSSSSS        ....
RR      RR  MM      MM    000000    AA      AA    CCCCCCCC    CCCCCCCC  EEEEEEEEEE  SSSSSSSS    SSSSSSSS        ....

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  ;++
0000  ; Facility: rms32
0000  ;
0000  ; Abstract:
0000  ;       this module performs the file access and
0000  ;       de-access fcp functions.
0000  ;
0000  ; Environment:
0000  ;               star processor running starlet exec.
0000  ;
0000  ; Author: L F Laverdure,         creation date: 10-MAR-1977
0000  ;
0000  ; Modified By:
0000  ;
0000  ;       V04-001 JWT0196         Jim Teague              14-Sep-1984
0000  ;               Restore V3 behavior of ignoring UPI for relative
0000  ;               and ISAM files.
0000  ;
0000  ;       V03-043 RAS0326         Ron Schaefer            23-Jul1984
0000  ;               Fix RAS0309 to force GET access to be allowed internally
0000  ;               if a valid EXE access is requested.  This makes
0000  ;               execute-only command procedures work.
0000  ;
0000  ;       V03-042 JWT0188         Jim Teague              21-Jul-1984
0000  ;               Don't allow $OPEN with sharing on magtapes.  RMS
0000  ;               was letting this slip through for 512-byte fixed
0000  ;               sequential files.
0000  ;
0000  ;       V03-041 RAS0309         Ron Schaefer            15-Jun-1984
0000  ;               Add support for execute-only images and command files.
0000  ;
0000  ;       V03-040 JWT0179         Jim Teague              23-Apr-1984
0000  ;               Always check for an ATR work area before allocating
0000  ;               one.
0000  ;
0000  ;       V03-039 JWT0175         Jim Teague              12-Apr-1984
0000  ;               Finish access mode ATR implementation.
0000  ;
0000  ;       V03-038 SHZ0005         Stephen H. Zalewski     06-Apr-1984
0000  ;               Back out the second part of shz0004.  Two reasons, first,
0000  ;               global buffers is a connect time option, no open time option.
0000  ;               Second, we would be record locking read only isam files, and
0000  ;               we never did before.
0000  ;
0000  ;       V03-037 JWT0173         Jim Teague               1-Apr-1984
0000  ;               Disable access mode ATRs for now.
0000  ;
0000  ;       V03-036 JWT0172         Jim Teague              28-Mar-1984
0000  ;               Keep exec mode byte at end of ATR work area.
0000  ;
0000  ;       V03-035 SHZ0004         Stephen H. Zalewski,    21-Mar-1984
0000  ;               Do not take out a file lock if UPI was specified in the
0000  ;               SHR field.
0000  ;
0000  ;               If file is READ ONLY, and global buffers specified, turn
0000  ;               on sharing so that global buffers can be used.  Old behavior
0000  ;               was to not use sharing since no locking was necessary, however
```

```
0000  ;                this prevented global buffering from being turned on.
0000  ;
0000  ;        V03-034 RAS0276          Ron Schaefer           20-Mar-1984
0000  ;                Prevent truncate-on-close (TEF FOP option) from being
0000  ;                honored for relative or indexed files.
0000  ;
0000  ;        V03-033 JWT0167          Jim Teague             15-Mar-1984
0000  ;                Allow write access with buffer offset as long as
0000  ;                BIO is set.  Also implement access-mode ATRs.
0000  ;
0000  ;        V03-032 DGB0012          Donald G. Blair        01-Mar-1984
0000  ;                Make changes related to ACP calls as part of the
0000  ;                restructuring necessary to support access mode
0000  ;                protected files.
0000  ;
0000  ;        V03-031 JWT0158          Jim Teague             27-Feb-1984
0000  ;                Adjustment to ANSI buffer offset stuff.  I had placed
0000  ;                the code to request the ATR$C_BUFFER_OFFSET attribute
0000  ;                in a common path for both $OPEN and $CREATE.  It
0000  ;                should only have been in the $OPEN access path.
0000  ;
0000  ;        V03-030 SHZ0003          Stephen H. Zalewski,   27-Feb-1984
0000  ;                Do not bump the available local buffer count in routine
0000  ;                RM$SETEBK as the local buffer it was trying to give back
0000  ;                (used for FWA) no longer exists.
0000  ;
0000  ;        V03-029 SHZ0002          Stephen H. Zalewski,   21-Feb-1984
0000  ;                If user opens file no-sharing, multi-streaming read only,
0000  ;                force locking to occur, otherwise no interlocking occurs,
0000  ;                and stream 2 could try to read from a bucket stream 1 is still
0000  ;                reading into cache.
0000  ;
0000  ;        V03-028 JWT0150          Jim Teague             01-Feb-1984
0000  ;                Implement ANSI buffer offset.
0000  ;
0000  ;        V03-027 JWT0148          Jim Teague             15-Dec-1983
0000  ;                Enforce ONLY_RU for $OPENs.
0000  ;
0000  ;        V03-026 RAS0218          Ron Schaefer           5-Dec-1983
0000  ;                Make node names work as search list elements.
0000  ;
0000  ;        V03-025 DAS0003          David Solomon          13-Sep-1983
0000  ;                Set RJB$V_OPEN before call to RM$MAPJNL.
0000  ;
0000  ;        V03-024 KBT0582          Keith B. Thompson      12-Aug-1983
0000  ;                Clean up some fwa constants
0000  ;
0000  ;        V03-023 DAS0002          David Solomon          20-Jul-1983
0000  ;                IFB$V_RUP moved from IFB$B_JNLFLG to IFB$B_JNLFLG2. Migrate
0000  ;                FAB$B_RCF recovery bits in RM$ACCESS (to catch both opens and
0000  ;                creates).
0000  ;
0000  ;        V03-022 KPL0012          Peter Lieberwirth      1-Jul-1983
0000  ;                Fix bug introduced in V03-020 that caused the PCB address
0000  ;                to be returned as the status code.
0000  ;
0000  ;        V03-021 DAS0001          David Solomon          22-Jun-1983
```

```
0000  ;               If opening a file for RU recovery, use FIB$V_NOLOCK
0000  ;               (open regardless).
0000  ;
0000  ;       V03-020 KPL0013          Peter Lieberwirth         21-Jun-1983
0000  ;               Don't migrate FAB recovery bits unless we're in recovery.
0000  ;
0000  ;       V03-019 KPL0012          Peter Lieberwirth         17-Jun-1983
0000  ;               Delay writing AT mapjnl entry until OPEN/CREATE is
0000  ;               complete.
0000  ;
0000  ;       V03-018 TSK0001          Tamar Krichevsky          12-Jun-1983
0000  ;               Fix broken branches to journaling routines.
0000  ;
0000  ;       V03-017 RAS0148          Ron Schaefer              26-Apr-1983
0000  ;               Initial support for extended XABPRO.
0000  ;
0000  ;       V03-016 LJA0059          Laurie J. Anderson        16-Feb-1983
0000  ;               Check for Multi-streaming even if NIL is set in the FAB share
0000  ;               field.
0000  ;
0000  ;       V03-015 KBT0491          Keith B. Thompson         9-Feb-1983
0000  ;               Checking for "proper" sharing is now done in rm$init_sfsb
0000  ;
0000  ;       V03-014 TMK0001          Todd M. Katz              01-Feb-1983
0000  ;               Add support for Recovery Unit Journalling and RU ROLLBACK
0000  ;               Recovery of ISAM files. Under the following set of conditions
0000  ;               set the journalling state bit IFB$V_RU_RLK within IFB$B_JNLFLG:
0000  ;
0000  ;               1. The file is an ISAM file.
0000  ;               2. The file is Recovery Unit Journallable.
0000  ;               3. The file has been opened for exclusive access (no sharing).
0000  ;
0000  ;               Setting of this bit will enable pseudo record locking.
0000  ;
0000  ;       V03-013 LJA0054          Laurie J. Anderson        12-Jan-1983
0000  ;               Fill in SHR field in IFB from Users FAB in rm$creacc_set1
0000  ;
0000  ;       V03-012 KPL0011          Peter Lieberwirth         17-Jan-1983
0000  ;               Migrate FAB bits that indicate file is being opened for
0000  ;               recovery into the IFB.
0000  ;
0000  ;       V03-011 SHZ0001          Stephen H. Zalewski       16-Dec-1982
0000  ;               Keep disk-structured hbk and ebk in different places in
0000  ;               ifb than we keep the swapped hbk and ebk.
0000  ;
0000  ;       V03-010 ACG0306          Andrew C. Goldstein,      13-Dec-1982  14:55
0000  ;               Remove obsolete file header symbols
0000  ;
0000  ;       V03-009 KBT0412          Keith B. Thompson         30-Nov-1982
0000  ;               Change ifb$w_devbufsiz to ifb$l_devbufsiz
0000  ;
0000  ;       V03-008 JWH0103          Jeffrey W. Horn           20-Sep-1982
0000  ;               Move the journaling set-up to RM$SETEBK.
0000  ;
0000  ;       V03-007 KBT0335          Keith B. Thompson         10-Sep-1982
0000  ;               Remove all S0 sharing code
0000  ;
```

```
0000  ;       V03-006 JWH0003         Jeffrey W. Horn         31-Aug-1982
0000  ;               Put in support for recovery unit journals.
0000  ;
0000  ;       V03-005 KBT0198         Keith B. Thompson       23-Aug-1982
0000  ;               Reorganize psects
0000  ;
0000  ;       V03-004 KBT0120         Keith B. Thompson       6-Aug-1982
0000  ;               Remove ref. to set_sifb_adr and fix all of the version 3
0000  ;               rev. numbers
0000  ;
0000  ;       V03-003 JWH0002         Jeffrey W. Horn         06-Jul-1982
0000  ;               Add in call to RM$RTVJNL to get journal control bits and
0000  ;               journal names.
0000  ;
0000  ;       V03-002 KPL0010         Peter Liebrwirth        25-Jun-1982
0000  ;               Complete V02-048 by checking for execute-only access
0000  ;               whether or not UFO is set.  Previously, if UFO was not
0000  ;               set, the check for execute-only access was skipped.
0000  ;
0000  ;       V03-001 JWH0001         Jeffrey W. Horn         23-Mar-1982
0000  ;               Add in call to RM$ASSJNL to set up journaling on this
0000  ;               file.
0000  ;
0000  ;       V02-050 KEK0018         K. E. Kinnear           3-Feb-1982
0000  ;               Replace FWA$C_RNSBUFSIZ with the real total size
0000  ;               of the concatenated NAME,TYPE, and VER buffer sizes.
0000  ;
0000  ;       V02-049 CDS0030         C Saether               20-Dec-1981
0000  ;               Allow deferred write for shared files.
0000  ;
0000  ;       V02-048 KPL0009         Peter Lieberwirth       17-Dec-1981
0000  ;               Provide support for execute only command files by having ACP
0000  ;               check for execute protection in SUPER mode as well as EXEC
0000  ;               and KERNEL.
0000  ;
0000  ;       V02-047 CDS0029         C Saether               16-Sep-1981
0000  ;               Allow BIO, BRO with MSE for rel, isam.  (same as pre 040).
0000  ;
0000  ;       V02-046 CDS0028         C Saether               14-Sep-1981
0000  ;               Clear NORECLK before UPI check.
0000  ;
0000  ;       V02-045 CDS0027         C Saether               6-Sep-1981
0000  ;               Init BLB queue header when noreclk is cleared.
0000  ;
0000  ;       V02-044 CDS0026         C Saether               4-Sep-1981
0000  ;               NORECLK now set by fseti - clear if locking.
0000  ;
0000  ;       V02-043 CDS0025         C Saether               31-Aug-1981
0000  ;               Always set noreclk.
0000  ;
0000  ;       V02-042 CDS0024         C Saether               23-Aug-1981
0000  ;               Init queue header and allocate a BLB if sharing.
0000  ;               Fix bug so that SFSB is allocated for 512 fix len.
0000  ;
0000  ;       V02-041 KPL0008         Peter Lieberwirth       15-Jul-1981
0000  ;               Allocate an SFSB in all cases, including sequential.
0000  ;
```

```
0000   ;       V02-040 KPL0007         Peter Lieberwirth       28-Apr-1981
0000   ;               Allocate an SFSB via RM$INIT_SFSB if necessary.
0000   ;
0000   ;       V02-039 CDS0023         C Saether       24-Feb-81       8:30
0000   ;               Check fixed length against RSIZ record attribute (ifb$w_lrl)
0000   ;               instead of max rec size (ifb$w_mrs).
0000   ;
0000   ;       V02-038 CDS0022         C Saether       23-Dec-80       15:10
0000   ;               Reverse order of attributes on stack so that rewriting
0000   ;               record attributes occurs before protection changes.
0000   ;
0000   ;       V02-037 REFORMAT        C Saether       30-Jul-80       20:20
0000   ;
0000
```

```
                   0000    271         .SBTTL  DECLARATIONS
                   0000    272
                   0000    273 ;
                   0000    274 ; Include Files:
                   0000    275 ;
                   0000    276
                   0000    277 ;
                   0000    278 ; Macros:
                   0000    279 ;
                   0000    280
                   0000    281         $ARMDEF
                   0000    282         $ATRDEF
                   0000    283         $BDBDEF
                   0000    284         $DEVDEF
                   0000    285         $FABDEF
                   0000    286         $FCHDEF
                   0000    287         $FIBDEF
                   0000    288         $FWADEF
                   0000    289         $IFBDEF
                   0000    290         $IMPDEF
                   0000    291         $IODEF
                   0000    292         $PCBDEF
                   0000    293         $PSLDEF
                   0000    294         $RJBDEF
                   0000    295         $RMSDEF
                   0000    296         $RUCBDEF
                   0000    297         $XABPRODEF
                   0000    298         $XABRDTDEF
                   0000    299
                   0000    300 ;
                   0000    301 ; Equated Symbols:
                   0000    302 ;
                   0000    303
          00000020 0000    304         FOP=FAB$L_FOP*8
                   0000    305
                   0000    306 ;
                   0000    307 ; Own Storage:
                   0000    308 ;
                   0000    309
```

RMO
V04

```
0000            .SBTTL  RM$ACCESS - PERFORM FCP ACCESS FUNCTION
0000
0000    ;++
0000    ;
0000    ; RM$ACCESS - perform file access function
0000    ;
0000    ; This routine sets up the access control word of the fib
0000    ; from the various user specifications, builds the
0000    ; attribute list to read in the record attributes and
0000    ; statistics block, builds the qio parameter list on
0000    ; the stack using the filename descriptor, issues
0000    ; the qio to the acp to perform the access,
0000    ; and finally initializes the ebk and hbk fields of
0000    ; the ifab.
0000    ;
0000    ; Calling sequence:
0000    ;
0000    ;       BSBW    RM$ACCESS
0000    ;
0000    ; Input Parameters:
0000    ;
0000    ;       r11     impure area address
0000    ;       r10     fwa address
0000    ;       r9      ifab addresss
0000    ;       r8      fab address
0000    ;
0000    ; Implicit Inputs:
0000    ;
0000    ;       fwa$t_fibbuf (fid & did set as required, remainder zero)
0000    ;       ifb$v_wrtacc
0000    ;       ifb$b_fac
0000    ;       fab$l_fop
0000    ;       ifb$l_chnl
0000    ;       fwa$l_atrladr
0000    ;       fwa$q_name
0000    ;
0000    ; Output Parameters:
0000    ;
0000    ;       r0      status code
0000    ;       r1-r7,ap destroyed
0000    ;
0000    ;
0000    ; Implicit Outputs:
0000    ;
0000    ;       ifb$v_accessed set
0000    ;       the record attributes area of the ifab is initialized
0000    ;       the record string is set (fwa$q_rns) over-writing
0000    ;               the filename string
0000    ;       ifb$l_ios
0000    ;       fab$v_ctg set if file contiguous, else cleared
0000    ;       fab$l_stv set to system error code on failure
0000    ;
0000    ; Completion Codes:
0000    ;
0000    ;       standard rms including suc, fnf, rer, wer, flk, prv,
0000    ;       and acc.
0000    ;
```

```
             0000    368 ; Side Effects:
             0000    369 ;
             0000    370 ;          may have switched to running at ast level.
             0000    371 ;          all user structures except fab must be reprobed.
             0000    372 ;--
             0000    373
```

```
                               0000  RMS$ACCESS::
                               0000           $TSTPT   ACCESS
          3C 69    03    E0    0006           BBS      #DEV$V_DIR,IFB$L_PRIM_DEV(R9),RMACC ; branch if files-oriented
          04 6A    19    E0    000A           BBS      #FWA$V_NODE,(R10),NTACC ; branch if network function
                               000E           RMSSUC                           ; show success
                         05    0011           RSB                              ; return to caller
                               0012
                               0012  ;++
                               0012  ;
                               0012  ;  perform network access function
                               0012  ;
                               0012  ;--
                               0012
                               0012  NTACC:
          0D 69    3E    E0    0012           BBS      #IFB$V_DAP,(R9),10$     ; branch if network file access
  04 A8   40020000 8F    D3    0016           BITL     #<<FAB$M_KFO>!-         ; disallow kfo and ufo options
                               001E                    <FAB$M_UFO>!-           ;  if task-to-task (to prevent
                               001E                    0>,FAB$L_FOP(R8)        ;  '$run node::"task=abc"'
                   03    13    001E           BEQL     10$                     ; branch if neither bits set
                FFDD'    31    0020           BRW      NT$SUP_FOP              ; return to caller with rms$_sup
                FFDA'    30    0023  10$:     BSBW     NT$ACCESS               ; establish logical link
                03 50    E8    0026           BLBS     R0,60$
                 01FF    31    0029           BRW      ERRACCESS               ; branch on failure
          04 68    26    E1    002C  60$:     BBC      #FAB$V_SQO+FOP,(R8),20$ ; branch if sqo not specified
                               0030           SSB      #IFB$V_SQO,(R9)         ; and save bit in ifab
          06 69    3F    E0    0034  20$:     BBS      #IFB$V_NSP,(R9),30$     ; branch if task-to-task oper.
                FFC5'    30    0038           BSBW     NT$OPEN                 ; open file via remote fal
                07 50    E9    003B           BLBC     R0,RET                  ; branch on failure
                               003E  30$:     SSB      #IFB$V_NORECLK,(R9)     ; say no record locking needed
                               0042           RMSSUC                           ; show success
                               0045
                         05    0045  RET:     RSB                              ; return to caller
                               0046
                               0046  RMACC:
                               0046
                               0046  ;
                               0046  ; Migrate FAB recovery bits to the IFB, (don't do so if this process is
                               0046  ; not entitled to do recovery).
                               0046  ;
      51  00000000'9F    D0    0046           MOVL     @#CTL$GL_PCB,R1         ; get PCB address
                   1A    E1    004D           BBC      #PCB$V_RECOVER,-        ; skip if not a recovery process
             26 24 A1          004F                    PCB$L_STS(R1),30$       ;
                4B A8    95    0052           TSTB     FAB$B_RCF(R8)           ; any bits set?
                   21    13    0055           BEQL     30$                     ; if eql no
                   00    E1    0057           BBC      #FAB$V_RU,-             ; branch if not RU recovery
             06 4B A8          0059                    FAB$B_RCF(R8),10$       ;
                               005C           SSB      #IFB$V_RU_RECVR,-       ; translate RU to IFB RU_RECVR
                               005C                    IFB$B_RECVRFLGS(R9)     ;
                   01    E1    0062  10$:     BBC      #FAB$V_AI,-             ; branch if not roll forward
             06 4B A8          0064                    FAB$B_RCF(R8),20$       ;
                               0067           SSB      #IFB$V_AI_RECVR,-       ; translate AI to IFB AI_RECVR
                               0067                    IFB$B_RECVRFLGS(R9)     ;
                   02    E1    006D  20$:     BBC      #FAB$V_BI,-             ; branch if not roll back
             06 4B A8          006F                    FAB$B_RCF(R8),30$       ;
                               0072           SSB      #IFB$V_BI_RECVR,-       ; translate BI to IFB BI_RECVR
                               0072                    IFB$B_RECVRFLGS(R9)     ;
                               0078
```

```
                                   0078       ;
                                   0078       ; Set up for the access.
                                   0078       ;
                                   0078
                       01B8   30   0078       30$:    BSBW    RM$CREACC_SET1                  ; perform first part of setups
                      C7 50   E9   007B               BLBC    R0,RET                          ; quit on error
                                   007E
                                   007E       ;
                                   007E       ; put a user-mode ATR on the list first
                                   007E       ;
                      85 01   B0   007E               MOVW    #1,(R5)+                        ; length of access mode byte
                      85 2D   B0   0081               MOVW    #ATR$C_ACCESS_MODE,(R5)+ ; access mode attribute
                   85 0A A9   9E   0084               MOVAB   IFB$B_MODE(R9),(R5)+            ; access mode for ACP to read
                                   0088
                      FF75'   30   0088               BSBW    RM$OPEN_XAB                     ; go process rms open xabs
                      B7 50   E9   008B               BLBC    R0,RET                          ; continue on success
                                   008E
                                   008E       ;
                                   008E       ; now an exec-mode ATR
                                   008E       ;
                                   008E
                      85 01   B0   008E               MOVW    #1,(R5)+                              ; 1 byte length
                      85 2D   B0   0091               MOVW    #ATR$C_ACCESS_MODE,(R5)+              ; access mode ATR
  85    58 AA    000001FC 8F  C1   0094               ADDL3   #508,FWA$L_ATR_WORK(R10),(R5)+        ; 1 byte signifying EXEC mode
                                   009D
                   06 69 1C   E1   009D               BBC     #DEV$V_RND,IFB$L_PRIM_DEV(R9),8$ ; branch if not disk
                                   00A1
                  00000000'EF 16   00A1               JSB     RM$RTVJNL                       ; get journal bits, names
                                   00A7
                       0275   30   00A7       8$:     BSBW    RM$CREACC_SET2                  ; finish setups
                                   00AA
                                   00AA       ;
                                   00AA       ; set the qio function code and go access the file
                                   00AA       ;
                                   00AA
                   50 72 8F   9A   00AA               MOVZBL  #IO$_ACCESS!IO$M_ACCESS,R0 ; function code
                                   00AE
                      FF4F'   30   00AE               BSBW    RM$FCPFNC                       ; do the access
                                   00B1
                      03 50   E8   00B1               BLBS    R0,RM$SETHBK                    ; continue on RM$FCPFNC success
                       0174   31   00B4               BRW     ERRACCESS                       ; branch on failure
```

```
                              00B7          .SBTTL  RM$SETHBK
                              00B7
                              00B7  ;++
                              00B7  ;
                              00B7  ; RM$SETHBK - entry for "create if" that becomes an open
                              00B7  ;
                              00B7  ; check the file for contiguous and if so set the ctg bit in fop,
                              00B7  ; then pick up highest allocated vbn from the statistics block
                              00B7  ; and copy to ifab, overwriting the hi vbn field of
                              00B7  ; the record attributes. note that the hi-and lo-order words of this vbn
                              00B7  ; are reversed on disk and hence are read in reverse order.
                              00B7  ; rearrange to give an understandable longword hi vbn. do same for
                              00B7  ; eof vbn.
                              00B7  ;
                              00B7  ;  entry point for "create if" turned into an open.
                              00B7  ;
                              00B7  ;  set fop output bits according to file attributes.
                              00B7  ;
                              00B7  ;--
                              00B7
                              00B7  RM$SETHBK::
           04     04     EF   00B7          EXTZV   #IFB$V_ORG,#IFB$S_ORG,-
       51    50 A9            00BA                  IFB$B_RFMORG(R9),R1          ; get org
       23 A9      51     90   00BD          MOVB    R1,IFB$B_ORGCASE(R9)         ; into separate ifab byte
    00B00200      8F     CA   00C1          BICL2   #<FAB$M_CTG!FAB$M_CBT!FAB$M_RCK!FAB$M_WCK>,-
           04 A8              00C7                  FAB$L_FOP(R8)                ; clear fop output bits
                  07     E1   00C9          BBC     #FCH$V_CONTIG,-
        04 44 AA              00CB                  FWA$W_UCHAR(R10),10$         ; branch if file not ctg.
                              00CE          SSB     #FAB$V_CTG+FOP,(R8)          ; set the ctg bit
                  05     E1   00D2  10$:    BBC     #FCH$V_CONTIGB,-
        04 44 AA              00D4                  FWA$W_UCHAR(R10),20$         ; branch if not ctg best try
                              00D7          SSB     #FAB$V_CBT+FOP,(R8)          ; set ctg best try in fop
                  03     E1   00DB  20$:    BBC     #FCH$V_READCHECK,-
        04 44 AA              00DD                  FWA$W_UCHAR(R10),30$         ; branch if no read checking
                              00E0          SSB     #FAB$V_RCK+FOP,(R8)          ; set fop rck bit
                  04     E1   00E4  30$:    BBC     #FCH$V_WRITCHECK,-
        04 44 AA              00E6                  FWA$W_UCHAR(R10),40$         ; branch if no write checking
                              00E9          SSB     #FAB$V_WCK+FOP,(R8)          ; set fop wck bit
                              00ED  40$:
    54 A9    01AC CA     D0   00ED          MOVL    FWA$L_HBK(R10),IFB$L_HBK_DISK(R9)     ; move unswapped hbk to ifb
 70 A9   54 A9    10     9C   00F3          ROTL    #16,IFB$L_HBK_DISK(R9),IFB$L_HBK(R9) ; swap words of hbk
                              00F9
        09 66     10     E1   00F9          BBC     #FIB$V_EXECUTE,(R6),50$ ; branch if not execute
                  01     E1   00FD          BBC     #FIB$V_ALT_GRANTED,-
        04 38 A6              00FF                  FIB$L_STATUS(R6),50$         ; branch if no read access
       16 A8      02     88   0102          BISB2   #FAB$M_GET,FAB$B_FAC(R8); flag read access also permitted
                              0106  50$:
              FEF7'      30   0106          BSBW    RM$OPEN_XAB1                 ; finish up xab processing
                              0109
```

```
                            0109         .SBTTL  RM$SETEBK
                            0109
                            0109  ;++
                            0109  ;
                            0109  ; RM$SETEBK - check for shared access
                            0109  ;
                            0109  ;  entry point to swap the words of eof block and set ifab bookeeping bit saying fil
                            0109  ;
                            0109  ;  set up journaling on the file
                            0109  ;
                            0109  ;  if this is not a sequential file, the shared ifab processing
                            0109  ;  is performed, if needed.
                            0109  ;
                            0109  ;  inputs:
                            0109  ;        r11      impure area address
                            0109  ;        r10      fwa address
                            0109  ;        r9       ifab address
                            0109  ;        r8       fab address
                            0109  ;
                            0109  ;  outputs:
                            0109  ;        r0               - status
                            0109  ;        r1-r7, ap        - destroyed
                            0109  ;        ifb$v_accessed   - set
                            0109  ;        ifb$l_ebk        - filled with swapped ebk words form disk
                            0109  ;--
                            0109
                            0109  RM$SETEBK::
                            0109         SSB     #IFB$V_ACCESSED,(R9)            ; declare file accessed
                            010D  ;
                            010D  ; Deallocate the ATR work area -- we're through with it now
                            010D  ;
                 3F    BB   010D         PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; Save regs
        54    58 AA    D0   010F         MOVL    FWA$L_ATR_WORK(R10),R4  ; Pass address of scratch page
               FEEA'   30   0113         BSBW    RM$RET1PAG              ; Return scratch page
              58 AA    D4   0116         CLRL    FWA$L_ATR_WORK(R10)     ; Indicate no work area now
                 3F    BA   0119         POPR    #^M<R0,R1,R2,R3,R4,R5>  ; Restore regs
                            011B
  74 A9    58 A9    10  9C  011B         ROTL    #16,IFB$L_EBK_DISK(R9),IFB$L_EBK(R9) ; swap words of ebk
                            0121
                            0121  ;
                            0121  ; Make sure user doesn't intend to write access an ANSI
                            0121  ;        buffer offset (b. o.) tape unless BIO is set
                            0121  ;
            00A8 C9    B5   0121         TSTW    IFB$W_BUFFER_OFFSET(R9)         ; is there a non-0 b. o.?
                 13    13   0125         BEQLU   5$                              ; if 0, skip next two tests
         09 69   05    E1   0127         BBC     #DEV$V_SQD,IFB$L_PRIM_DEV(R9),4$ ; if not a tape, error
         0B 69   30    E1   012B         BBC     #IFB$V_WRTACC,(R9),5$           ; if no write access, we're cool
      06 16 A8   05    E0   012F         BBS     #FAB$V_BIO,FAB$B_FAC(R8),5$     ; write access is ok with BIO
                            0134  4$:    RMSERR  IFF                             ;  otherwise no write access
                       05   0139         RSB                                     ;      so cease and desist
                            013A
                            013A  ;
                            013A  ; set up journaling on the file
                            013A  ;
                            013A
       00A0 C9   20    8A   013A  5$:    BICB2   #IFB$M_NEVER_RU,IFB$B_JNLFLG(R9) ; Ignoring NEVER_RU, is
            00A0 C9    95   013F         TSTB    IFB$B_JNLFLG(R9)                 ; any journaling bit set?
```

```
             64   13  0143          BEQL    SHRCHK                          ; branch if not
                      0145  ;
                      0145  ; Enforce RU bit settings, specifically ONLY_RU
                      0145  ;
     00A0 C9     03   93  0145          BITB    #IFB$M_RU!IFB$M_ONLY_RU,IFB$B_JNLFLG(R9) ; RU bits set?
             1B   13  014A          BEQL    20$                             ; If not, go on with jnl stuff
51    00000000'9F     D0  014C          MOVL    @#CTL$GL_RUF,R1                 ; RUF loaded?
             05   13  0153          BEQL    10$                             ; No RUF, verify ONLY_RU clear
   0D 11 A1      01   E0  0155          BBS     #RUCB$V_ACTIVE,RUCB$B_CTRL(R1),20$ ; In RU? Then go set up
                      015A  10$:
             01   93  015A          BITB    #IFB$M_ONLY_RU,-                ; If ONLY_RU clear (RU
          00A0 C9         015C                  IFB$B_JNLFLG(R9)                ;  must be set), and not
             06   13  015F          BEQL    20$                             ;  in RU then that's ok
                      0161          RMSERR  NRU             ; However, if ONLY_RU set and not in RU: error
                  05  0166          RSB
                      0167
                      0167  20$:
     00000000'EF      16  0167          JSB     RM$ASSJNL                       ; set up journaling
          55 50       E9  016D          BLBC    R0,RETURN                       ; get out on error
     00A0 C9          95  0170          TSTB    IFB$B_JNLFLG(R9)                ; ASSJNL can clear this
             33   13  0174          BEQL    SHRCHK                          ; branch if now clear
                      0176
                      0176  ;
                      0176  ; Turn off AT for this MAPJNL call so the AT info can be filled in
                      0176  ; during the operation and flushed later.
                      0176  ;
51     00A4 C9        D0  0176          MOVL    IFB$L_RJB(R9),R1                ; get RJB address
   7E     0A A1       B0  017B          MOVW    RJB$W_FLAGS(R1),-(SP)           ; save current flags
                          017F          CSB     #RJB$V_AT,RJB$W_FLAGS(R1)       ; turn off AT for now
   0A A1         10   A8  0184          BISW2   #RJB$M_OPEN,RJB$W_FLAGS(R1)     ; set flag that this is an open
  00000000'EF         16  0188          JSB     RM$MAPJNL                       ; write out mapping entries
51     00A4 C9        D0  018E          MOVL    IFB$L_RJB(R9),R1                ; get RJB address again
    0A A1        8E   B0  0193          MOVW    (SP)+,RJB$W_FLAGS(R1)           ; restore original flags
          2B 50       E9  0197          BLBC    R0,RETURN                       ; get out on error
 09 00A2 C9      02   E1  019A          BBC     #IFB$V_RUP,IFB$B_JNLFLG2(R9),SHRCHK ; branch if not in RU
  00000000'EF         16  01A0          JSB     RM$MAPJNL_RU                    ; write out RU mapping entry
          1C 50       E9  01A6          BLBC    R0,RETURN                       ; get out on error
                          01A9
    1D 69        33   E1  01A9  SHRCHK: BBC     #IFB$V_NORECLK,(R9),CHKSHR      ; not set, then check sharing
                          01AD
                          01AD  ;
                          01AD  ; If this is a Recovery Unit Journalable ISAM file which is being
                          01AD  ; opened for exclusive access then set the state bit IFB$V_RU_RLK to
                          01AD  ; enable pseudo record locking.
                          01AD  ;
                          01AD
    23 A9        02   91  01AD  EXIT:   CMPB    #IFB$C_IDX,IFB$B_ORGCASE(R9)    ; return if this is not an
             12   12  01B1          BNEQ    RETURN                          ; access of an index file
                          01B3
  00A0 C9        01   E1  01B3          BBC     #IFB$V_RU,IFB$B_JNLFLG(R9),-    ; return if this ISAM file is
             0C           01B8                  RETURN                          ; not Recovery Unit journallable
                          01B9
    17 A8        1F   93  01B9          BITB    #FAB$M_SHRGET!FAB$M_SHRPUT-     ; return if any form of sharing
                          01BD                  !FAB$M_SHRDEL!FAB$M_SHRUPD-     ; is enabled (inter-process or
                          01BD                  !FAB$M_MSE,FAB$B_SHR(R8)        ; inter-stream) - record locking
             06   12  01BD          BNEQ    RETURN                          ; will already be enabled
                          01BF
                      01BF          SSB     #IFB$V_RU_RLK,IFB$B_JNLFLG2(R9) ; permit pseudo record locking
                      01C5
                  05  01C5  RETURN: RSB
                      01C6
                      01C6
                      01C6  SETNORECLK:
      E3 69    33  E3  01C6          BBCS    #IFB$V_NORECLK,(R9),EXIT ; set NORECLK & exit (always clear)
                      01CA
                      01CA  CHKSHR:
                      01CA
                      01CA  ;
                      01CA  ; check whether sharing is required
                      01CA  ;
                      01CA
   05 17 A8    05  E1  01CA          BBC     #FAB$V_NIL,FAB$B_SHR(R8),10$     ; If nil spec'd, check MSE
   F2 17 A8    04  E1  01CF          BBC     #FAB$V_MSE,FAB$B_SHR(R8),SETNORECLK      ; No locking required
                      01D4
                      01D4          ASSUME  FAB$C_SEQ         EQ        0
                      01D4
         23 A9    95  01D4  10$:    TSTB    IFB$B_ORGCASE(R9)          ; is this sequential org?
            1B    13  01D7          BEQL    CHKSEQSHR                  ; special checks for 512 fix len recs.
                      01D9
                      01D9  SHARE:
         FE24'    30  01D9          BSBW    RM$INIT_SFSB               ; get parent lock for record and
                      01DC                                             ; bucket locks.
         0E 50    E9  01DC          BLBC    R0,10$                     ; exit on error.
            5A    DD  01DF          PUSHL   R10                        ; Save FWA address.
      5A    59    D0  01E1          MOVL    R9,R10                     ; ALBLB wants ifab in r10.
         FE19'    30  01E4          BSBW    RM$ALBLB                   ; allocate a BLB to go with BDB (FWA).
         5A 8ED0      01E7          POPL    R10                        ; Restore FWA address.
         C0 50    E8  01EA          BLBS    R0,EXIT                    ; finish up
                  05  01ED  10$:    RSB
                      01EE
                      01EE  UPIERR: RMSERR  UPI
                  05  01F3          RSB
                      01F4
                      01F4  CHKSEQSHR:
                      01F4
                      01F4  ;
                      01F4  ; want sharing on sequential file - make a few more checks
                      01F4  ;
   CD 17 A8    06  E0  01F4          BBS     #FAB$V_UPI,FAB$B_SHR(R8),SETNORECLK      ; Branch if UPI.
      28 69    1C  E1  01F9          BBC     #DEV$V_RND,IFB$L_PRIM_DEV(R9),SHRERR     ; Magtape?!? No way!
   16 A8    60 8F  93  01FD          BITB    #FAB$M_BIO!FAB$M_BRO,FAB$B_FAC(R8)       ; any form of block i/o?
            EA    12  0202          BNEQ    UPIERR                             ; UPI must be set for block i/o.
                      0204
                      0204          ASSUME  FAB$C_SEQ         EQ        0
                      0204
      01    50 A9  91  0204          CMPB    IFB$B_RFMORG(R9),#FAB$C_FIX ; only for fixed length recs
            1B    12  0208          BNEQ    SHRERR                     ; neg sorry
 0200 8F    52 A9  B1  020A          CMPW    IFB$W_LRL(R9),#512         ; 512 byte records only
            13    12  0210          BNEQ    SHRERR                     ; sorry, can't share
      5E A9    01  90  0212          MOVB    #1,IFB$B_BKS(R9)           ; bucket size is one
                      0216
                      0216          ASSUME  <IFB$C_SEQ + 1> EQ      IFB$C_REL
                      0216
         23 A9    96  0216          INCB    IFB$B_ORGCASE(R9)          ; presto - now you're relative
```

```
RMO
Sym

SHR
TPT
TPT
UPI
XAB
XAB
XAB
XAB
XBC
XBC

PSE
---
.
RM$
$AB

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

The
116
The
116
34

Mac
---
_$2
_$2
_$2
TOT

221

The
```

```
       00B0 C9   01   D0  0219   695            MOVL    #1,IFB$L_DVBN(R9)          ; no prologue for seq file
                          021E   696            SSB     #IFB$V_SEQFIL,(R9)         ; note this is really seq file
                 FFB4  31  0222   697            BRW     SHARE                      ; finish shared open
                          0225   698 SHRERR:
                          0225   699            RMSERR  SHR                        ; can't do that
                      05  022A   700            RSB                                ; get back
                          022B   701
                          022B   702 ;++
                          022B   703 ;
                          022B   704 ;   handle access failure
                          022B   705 ;
                          022B   706 ;--
                          022B   707
                          022B   708 ERRACCESS:
                          022B   709            RMSERR  ACC,R1                     ; default error code
                FDCD'  31  0230   710            BRW     RM$MAPERR                  ; go map error code to rms
                          0233   711                                               ; and return to caller
```

```
                        0233          .SBTTL  RM$CREACC_SET1
                        0233
                        0233  ;++
                        0233  ;
                        0233  ; RM$CREACC_SET1 - access, protection, datacheck options fib setup
                        0233  ;
                        0233  ;   this subroutine initializes the access control word of the fib from
                        0233  ;   the various fop options, sets the retrieval window size, and initializes
                        0233  ;   r5 to address at which to build a files attributes list
                        0233  ;
                        0233  ;   inputs:
                        0233  ;         r10     fwa address
                        0233  ;         r9      ifab address
                        0233  ;         r8      fab address
                        0233  ;
                        0233  ;   outputs:
                        0233  ;         r6      fib address
                        0233  ;         r5      address for next entry to be added to attribute's list
                        0233  ;         r0      success/fail status
                        0233  ;
                        0233  ;--
                        0233
                        0233  RM$CREACC_SET1::
 56    14 BA    9E      0233          MOVAB   @FWA$Q_FIB+4(R10),R6    ; get fib address
                        0237
                        0237  ;
                        0237  ; initialize the access control word.  it is zero; set desired bits.
                        0237  ;
                        0237
                        0237          ASSUME  FIB$L_ACCTL EQ 0
 04 69    30    E1      0237          BBC     #IFB$V_WRTACC,(R9),5$   ; branch if read access only
                        023B          SSB     #FIB$V_WRITE,(R6)       ; set write access bit
                        023F
                        023F  ;
                        023F  ;   set sharing as desired and determine whether record locking required.
                        023F  ;
                        023F  ;   record locking will be required if there is any form of sharing (inter
                        023F  ;   or intra process) and there can be any writers of the file.
                        023F  ;
                        023F
 50    17 A8    90      023F  5$:     MOVB    FAB$B_SHR(R8),R0                ; get shr field
 4E A9    50    90      0243          MOVB    R0,IFB$B_SHR(R9)                ; Save share field in IFB
 04 50    04    E1      0247          BBC     #FAB$V_MSE,R0,10$               ; branch if no multi-streams
                        024B          SSB     #IFB$V_MSE,(R9)                 ; set mse bit
 10 50    05    E0      024F  10$:    BBS     #FAB$V_NIL,R0,20$               ; branch if no sharing
    50    0D    93      0253          BITB    #FAB$M_PUT!FAB$M_UPD!FAB$M_DEL,R0 ; any form of write sharing?
          13    12      0256          BNEQ    30$                             ; branch if yes
    66    01    88      0258          BISB2   #FIB$M_NOWRITE,(R6)             ; disallow other writers
                        025B                                                  ; at most "get" sharing
 08 69    30    E1      025B          BBC     #IFB$V_WRTACC,(R9),25$          ; branch if not write accessed
 08 50    01    E0      025F          BBS     #FAB$V_GET,R0,30$               ; branch if allowing other readers
                        0263                                                  ; default write accessor to nil
                        0263  20$:    SSB     #FIB$V_NOREAD,(R6)              ; disallow other readers
 1A 50    04    E1      0267  25$:    BBC     #FAB$V_MSE,R0,35$               ; branch if no multi streams
                        026B
                        026B  ;
                        026B  ;   record locking required - unless upi set.  require sharers to specify
```

```
                            026B   770 ;     rms locking.
                            026B   771
                            026B   772 30$:       CSB     #IFB$V_NORECLK, (R9)       ; clear no locking flag.
  0098 C9    0098 C9    DE  026F   773            MOVAL   IFB$L_BLBFLNK(R9), IFB$L_BLBFLNK(R9) ; Init BLB queue header.
  009C C9    0098 C9    DE  0276   774            MOVAL   IFB$L_BLBFLNK(R9), IFB$L_BLBBLNK(R9) ; Init BLB queue header.
        04 50    06     E0  027D   775            BBS     #FAB$V_UPI,R0,35$          ;
                            0281   776            SSB     #FIB$V_RMSLOCK,(R6)        ; set fib bit for locking
                            0285   777
                            0285   778 ;
                            0285   779 ;   set deferred write ifab flag as required
                            0285   780 ;
                            0285   781
        04 68    25     E1  0285   782 35$:       BBC     #FAB$V_DFW+FOP,(R8),40$ ; branch if deferred write not
                            0289   783                                               ; specified
                            0289   784            SSB     #IFB$V_DFW,(R9)            ; set deferred write flag
                            028D   785
                            028D   786 ;
                            028D   787 ;   set read checking, write checking, and seq. operations only flags
                            028D   788 ;
                            028D   789
        07 68    29     E1  028D   790 40$:       BBC     #FAB$V_WCK+FOP,(R8),50$ ; branch if no write-checking
           66    20     88  0291   791            BISB2   #1@FIB$V_WRITECK,(R6)      ; enable write-checking
        44 AA    10     88  0294   792            BISB2   #1@FCH$V_WRITCHECK,FWA$W_UCHAR(R10) ; & give file wck attribute
        08 68    37     E1  0298   793 50$:       BBC     #FAB$V_RCK+FOP,(R8),60$ ; branch if no read-checking
                            029C   794            SSB     #FIB$V_READCK,(R6)         ; enable read-checking
        44 AA    08     88  02A0   795            BISB2   #1@FCH$V_READCHECK,FWA$W_UCHAR(R10) ; & give file rck attribute
        08 68    26     E1  02A4   796 60$:       BBC     #FAB$V_SQO+FOP,(R8),70$ ; branch if sqo not specified
                            02A8   797            SSB     #FIB$V_SEQONLY,(R6)        ; set sequential only bit
                            02AC   798            SSB     #IFB$V_SQO,(R9)            ; and save bit in ifab
                            02B0   799
                            02B0   800 ;
                            02B0   801 ; if magtape, check and set positioning flags (rwo, pos, nef)
                            02B0   802 ;
                            02B0   803
        0E 69    05     E1  02B0   804 70$:       BBC     #DEV$V_SQD,IFB$L_PRIM_DEV(R9),80$ ; branch if not magtape
                            02B4   805            SSB     #FIB$V_PRSRV_ATR,(R6)      ; read rat bits as stored
                            02B8   806
                            02B8   807 ;
                            02B8   808 ; the rms fop bits for magtape positioning are in the same
                            02B8   809 ; relative position to each other as the corresponding fib bits
                            02B8   810 ; and additionally have the same polarity - use an extract
                            02B8   811 ; and insert field to set them appropriately
                            02B8   812 ; (note:  the wck bit is imbedded - so it gets set or cleared again)
                            02B8   813 ;
                            02B8   814
                            02B8   815            ASSUME  <FAB$V_RWO+1> EQ FAB$V_POS
                            02B8   816            ASSUME  <FAB$V_POS+1> EQ FAB$V_WCK
                            02B8   817            ASSUME  <FAB$V_WCK+1> EQ FAB$V_NEF
                            02B8   818            ASSUME  <FIB$V_REWIND+1> EQ FIB$V_CURPOS
                            02B8   819            ASSUME  <FIB$V_CURPOS+1> EQ FIB$V_WRITECK
                            02B8   820            ASSUME  <FIB$V_WRITECK+1> EQ FIB$V_UPDATE
  50    68    04    27  EF  02B8   821            EXTZV   #FAB$V_RWO+FOP,#4,(R8),R0; get the fop bits
  66    04    03    50  F0  02BD   822            INSV    R0,#FIB$V_REWIND,#4,(R6)
                            02C2   823
                            02C2   824 ;
                            02C2   825 ;  if this is ufo set fib$v_notrunc unless trn bit set in fac
                            02C2   826 ;
```

```
                        02C2
   09 68      31   E1   02C2  80$:    BBC     #FAB$V_UFO+FOP,(R8),90$ ; branch if not ufo
04 16 A8      04   E0   02C6          BBS     #FAB$V_TRN,FAB$B_FAC(R8),90$ ; branch if trn set
                        02CB          SSB     #FIB$V_NOTRUNC,(R6)      ; don't allow truncates
                        02CF
                        02CF  ;
                        02CF  ;  check for execute protection
                        02CF  ;
                        02CF
12 16 A8      07   E1   02CF  90$:    BBC     #FAB$V_EXE,FAB$B_FAC(R8),100$ ; branch if not execute access
      0A A9   91        02D4          CMPB    IFB$B_MODE(R9),-          ;
         02             02D7                  #PSL$C_SUPER              ; super (or exec or kernel) mode?
         0C   1A        02D8          BGTRU   100$                      ; branch if not (ignore)
   22 A9      02   88   02DA          BISB2   #FAB$M_GET,IFB$B_FAC(R9); flag read access also permitted
                        02DE          SSB     #FIB$V_EXECUTE,(R6)       ; have acp check on execute access
              01   D0   02E2          MOVL    #ARM$M_READ,-             ; also ask if read access permitted
      3C A6             02E4                  FIB$L_ALT_ACCESS(R6)
                        02E6
                        02E6  ;
                        02E6  ; Set override exclusive access if opening a file for RU recovery.
                        02E6  ;
                        02E6
              00   E1   02E6  100$:   BBC     #IFB$V_RU_RECVR,-         ; skip if not RU recovery.
      00A1 C9           02E8                  IFB$B_RECVRFLGS(R9),-     ;
           0E           02EB                  SETRTV                    ;
 00100000 8F       C8   02EC          BISL2   #FIB$M_NOLOCK,-           ; set nolock (access regardless) flag.
           66           02F2                  FIB$L_ACCTL(R6)
 00000401 8F       CA   02F3          BICL2   #FIB$M_NOREAD!FIB$M_NOWRITE,-
           66           02F9                  FIB$L_ACCTL(R6)           ; noread/nowrite must be clear.
                        02FA
                        02FA  ;
                        02FA  ; set the retrieval window size
                        02FA  ;
                        02FA
03 A6    1C A8     90   02FA  SETRTV: MOVB    FAB$B_RTV(R8),FIB$B_WSIZE(R6)
                        02FF
                        02FF  ;
                        02FF  ; the fib is now set up.
                        02FF  ; set the attribute control list address into r5
                        02FF  ;
                        02FF
   55    58 AA     D0   02FF          MOVL    FWA$L_ATR_WORK(R10),R5    ; Do we need one?
              11   12   0303          BNEQ    10$                       ; If not, don't ask for one
                        0305
              0E   BB   0305          PUSHR   #^M<R1,R2,R3>             ; Save regs
           FCF6'   30   0307          BSBW    RM$GET1PAG                ; Grab a scratch page
         11 50     E9   030A          BLBC    R0,20$                    ; Die if none available
   58 AA      53   D0   030D          MOVL    R3,FWA$L_ATR_WORK(R10)    ; Save scratch page address
      55      53   D0   0311          MOVL    R3,R5                     ;  and put it in R5
              0E   BA   0314          POPR    #^M<R1,R2,R3>             ; Restore regs
 01FC C5      01   D0   0316  10$:    MOVL    #PSL$C_EXEC,508(R5)       ; Keep exec mode byte in last lword
      50      01   D0   031B          MOVL    #1,R0                     ; set success
              05        031E  20$:    RSB
```

```
                         031F          .SBTTL  RM$CREACC_SET2
                         031F
                         031F   ;++
                         031F   ;
                         031F   ; RM$CREACC_SET2 - set up stat block, fall thru to creac_3
                         031F   ;
                         031F   ;  subroutine to finish fcp access & create setups started by rm$creacc_set1
                         031F   ;
                         031F   ;  if this is for an access it puts an entry on the attributes list
                         031F   ;  to cause the statistics block to be read
                         031F   ;
                         031F   ;  it then adds attribute list entries for rms record attributes,
                         031F   ;  user characteristics, and, if device is magtape, block size.
                         031F   ;  it then ends the attributes list and builds p6 thru p2 of the fcp's
                         031F   ;  qio parameter block and returns.
                         031F   ;
                         031F   ;  inputs:
                         031F   ;       r10     fwa address
                         031F   ;       r5      attributes list next entry address
                         031F   ;
                         031F   ;  outputs:
                         031F   ;       p6 thru p2 on stack
                         031F   ;       r0, r5 destroyed
                         031F   ;--
                         031F   ;
                         031F   ;  entry point to finish fcp access setups
                         031F   ;
                         031F   ;--
                         031F
                         031F   RM$CREACC_SET2::
      85   0A   B0       031F          MOVW    #FWA$S_STATBLK,(R5)+     ; specify # of bytes wanted
      85   09   B0       0322          MOVW    #ATR$C_STATBLK,(R5)+     ; read statistics block
 85 01A8   CA   9E       0325          MOVAB   FWA$T_STATBLK(R10),(R5)+ ; address for read
                         032A
                         032A   ;
                         032A   ; If magtape, then inquire about buffer offset -- otherwise proceed to
                         032A   ;    CREACC_3.  Note that this inquiry is not made for $CREATE.
                         032A   ;
                         032A
      69   05   E1       032A          BBC     #DEV$V_SQD,IFB$L_PRIM_DEV(R9),-
           0B            032D                  RM$CREACC_SET3                   ; magtape?
      85   02   B0       032E          MOVW    #ATR$S_BUFFER_OFFSET,(R5)+       ; size of b.o. field (2)
      85   30   B0       0331          MOVW    #ATR$C_BUFFER_OFFSET,(R5)+       ; buffer offset item code
 85 00A8   C9   3E       0334          MOVAW   IFB$W_BUFFER_OFFSET(R9),(R5)+    ; directly to/from ifab
                         0339
                         0339   ;++
                         0339   ;
                         0339   ; RM$CREACC_SET3 - set up for record attributes and user characteristics
                         0339   ;
                         0339   ;  entry point to finish create function setup without getting a statistics block
                         0339   ;
                         0339   ;       put in entries to cause record attributes and user characteristics
                         0339   ;       to be read/written
                         0339   ;
                         0339   ;--
                         0339
                         0339   RM$CREACC_SET3::
```

```
                 50 8ED0  0339               POPL    R0                              ; save return pc
            85   16   B0  033C               MOVW    #<IFB$C_FHAEND-IFB$B_RFMORG>,(R5)+ ; # bytes rec attr to xfer
            85   04   B0  033F               MOVW    #ATR$C_RECATTR,(R5)+            ; get rms record attributes
       85   50   A9   DE  0342               MOVAL   IFB$B_RFMORG(R9),(R5)+          ; xfer attr's directly to/from ifab
            85   04   B0  0346               MOVW    #ATR$S_UCHAR,(R5)+              ; size of user characteristics
            85   03   B0  0349               MOVW    #ATR$C_UCHAR,(R5)+              ; specify read/write of   "
       85   44   AA   3E  034C               MOVAW   FWA$W_UCHAR(R10),(R5)+          ; addr to read/write      "
       0A   69   05   E1  0350               BBC     #DEV$V_SQD,IFB$L_PRIM_DEV(R9),5$; branch if not magtape
            85   02   B0  0354               MOVW    #ATR$S_BLOCKSIZE,(R5)+          ; specify blocksize size (2)
            85   0B   B0  0357               MOVW    #ATR$C_BLOCKSIZE,(R5)+          ; specify read/write of blksiz
       85   48   A9   DE  035A               MOVAL   IFB$L_DEVBUFSIZ(R9),(R5)+       ; xfer directly to/from ifab
                 65   D4  035E      5$:      CLRL    (R5)                            ; flag end of attribute list
                          0360
                          0360      ;
                          0360      ; start building qio argument list on stack
                          0360      ;
                          0360
                 00   DD  0360               PUSHL   #0                              ; p6
            58   AA   DD  0362               PUSHL   FWA$L_ATR_WORK(R10)             ; p5 = attribute list address
          0188   CA   7F  0365      P4_P2:   PUSHAQ  FWA$Q_RNS(R10)                  ; p4 = resultant name string descriptor
          0170   CA   DF  0369               PUSHAL  FWA$Q_NAME(R10)                 ; p3 = address of long word
                          036D                                                       ; to receive resultant string length
      0000012E   8F   D0  036D               MOVL    #FWA$S_NAMEBUF+FWA$S_TYPEBUF+FWA$S_VERBUF,-
          0188   CA       0373                       FWA$Q_RNS(R10)                  ; length of rns buffer
018C CA   04B6   CA   9E  0376               MOVAB   FWA$T_NAMEBUF(R10),FWA$Q_RNS+4(R10) ; overlay input filename
                          037D                                                       ; with resultant string
          0170   CA   7F  037D               PUSHAQ  FWA$Q_NAME(R10)                 ; p2 = filename string
                 60   17  0381               JMP     (R0)                            ; return to caller
                          0383
                          0383      ;++
                          0383      ; RM$FCP_P4_P2 - push p4 thru p2 onto stack
                          0383      ;
                          0383      ;  entry point to push p4 through p2 onto stack for fcp argument list
                          0383      ;  build for the $erase function (delete file)
                          0383      ;--
                          0383
                          0383      RM$FCP_P4_P2::
                 01   BA  0383               POPR    #^M<R0>                         ; save return pc
                 DE   11  0385               BRB     P4_P2                           ; go do it
```

```
0387          .SBTTL  RM$DEACCESS - PERFORM FCP DEACCESS FUNCTION
0387
0387    ;++
0387    ;
0387    ; RM$DEACCESS - perform file deaccess function
0387    ;
0387    ; This routine builds an attribute list to cause the record
0387    ; attributes in the ifab to be rewritten to the file
0387    ; header, if the file was write accessed, and
0387    ; calls rm$fcpfnc to perform the deaccess.
0387    ;
0387    ; Calling sequence:
0387    ;
0387    ;       BSBW    RM$DEACCESS
0387    ;
0387    ; Input Parameters:
0387    ;
0387    ;       r11     impure area address
0387    ;       r9      ifab address
0387    ;       r8      fab address
0387    ;
0387    ; Implicit Inputs:
0387    ;
0387    ;       ifb$l_chnl
0387    ;
0387    ; outputs:
0387    ;
0387    ;       r0      status code
0387    ;       r1-r6,ap destroyed
0387    ;
0387    ; Implicit Outputs:
0387    ;
0387    ;       ifb$l_ios
0387    ;
0387    ; Completion Codes:
0387    ;
0387    ;       standard rms, in particular, suc, dac, fno.
0387    ;
0387    ; Side Effects:
0387    ;
0387    ;       on return rms may be running at ast level
0387    ;       requiring a reprobe of any user structures except
0387    ;       the fab.
0387    ;--
0387
```

```
                           0387
                           0387     ;++
                           0387     ;
                           0387     ;  xab processing arguments for close
                           0387     ;
                           0387     ;--
                           0387
                           0387     CLS_XAB_ARGS:
              00'14 1E     0387             .BYTE   XAB$C_RDT,XAB$C_RDTLEN,XBC$C_CLSRDT ; handle rdt xab
              00'10 13     038A             .BYTE   XAB$C_PRO,XAB$C_PROLEN_V3,XBC$C_CLSPRO ; handle pro xab
                    00     038D             .BYTE   0
                           038E
                           038E     ;++
                           038E     ;
                           038E     ;  perform network deaccess function
                           038E     ;
                           038E     ;--
                           038E
                           038E             ASSUME  IFB$V_DAP GE 56
                           038E             ASSUME  IFB$V_DAP LE 63
                           038E             ASSUME  IFB$V_NSP GE 56
                           038E             ASSUME  IFB$V_NSP LE 63
           00000007        038E     BKP3    = <56/8>                                ; byte offset to flags byte
           000000C0        038E     NETMASK = <1@<IFB$V_DAP-56>> ! <1@<IFB$V_NSP-56>> ; network access-type flags
                           038E
                           038E     NTDAC:
   0A 69    3D      E5     038E             BBCC    #IFB$V_DAP_OPEN,(R9),10$ ; branch if close not necessary
   06 6B    04      E0     0392             BBS     #IMP$V_IORUNDOWN,(R11),10$ ; branch if i/o rundown in progress
            FC67'   30     0396             BSBW    NT$CLOSE                 ; yes, close it there
            11 50   E9     0399             BLBC    R0,20$                   ; branch on failure
            FC61'   30     039C     10$:    BSBW    NT$DEACCESS              ; destroy logical link with partner
 07 A9      C0 8F   8A     039F             BICB2   #NETMASK,BKP3(R9)        ; clear network access-type flags
            07 50   E9     03A4             BLBC    R0,30$                   ; branch on failure
            FC56'   30     03A7             BSBW    NT$NWA_FREE              ; discard nwa
                           03AA             RMSSUC                           ; show success
                    05     03AD     20$:    RSB                              ; exit to caller
            00A8    31     03AE     30$:    BRW     ERRDAC                   ; branch aid
                           03B1
                           03B1     ;++
                           03B1     ;
                           03B1     ;  entry point for rm$deaccess
                           03B1     ;
                           03B1     ;--
                           03B1
                           03B1     RM$DEACCESS::
                           03B1             $TSTPT  DEACCES
   D3 69    0D      E0     03B7             BBS     #DEV$V_NET,IFB$L_PRIM_DEV(R9),NTDAC ; br if network device
                           03BB             RMSSUC  SUC,R6                   ; indicate success
                           03BE
                    00  DD 03BE             PUSHL   #0                       ; signal end of attribute list
   5C       C4 AF   9E     03C0             MOVAB   CLS_XAB_ARGS,AP          ; arg list addr for rm$xab_scan
            FC39'   30     03C4             BSBW    RM$XAB_SCAN              ; process xab chain
      56    50      D0     03C7             MOVL    R0,R6                    ; save status
                           03CA     ;
                           03CA     ; build attribute list on stack to rewrite record attributes
                           03CA     ;
                           03CA
            1A 69   30   E1 03CA       BBC     #IFB$V_WRTACC,(R9),10$    ; branch if not write accessed
            06 69   38   E1 03CE       BBC     #IFB$V_SEQFIL,(R9),5$     ; skip next few lines if really rel
                            03D2       ASSUME  <IFB$C_SEQ + 1> EQ IFB$C_REL
               23 A9   97 03D2         DECB    IFB$B_ORGCASE(R9)         ; turn back into sequential file
               5E A9   94 03D5         CLRB    IFB$B_BKS(R9)             ; make sure this clear also
                          03D8  5$:
               50 A9   DF 03D8         PUSHAL  IFB$B_RFMORG(R9)          ; write attributes from ifab
         00040016 8F   DD 03DB         PUSHL   #<ATR$C_RECATTR@16>+<IFB$C_FHAEND-IFB$B_RFMORG>
                          03E1                                           ; length & record attriubtes code
                          03E1
                          03E1  ;
                          03E1  ;  put org back into rfmorg byte
                          03E1  ;
                          03E1
 50 A9    04   04   23 A9   F0 03E1    INSV    IFB$B_ORGCASE(R9),#IFB$V_ORG,#IFB$S_ORG,IFB$B_RFMORG(R9)
                          03E8
                          03E8  ;
                          03E8  ;  allocate a fib to handle various options
                          03E8  ;
                          03E8
            52   40 8F   9A 03E8  10$:  MOVZBL  #FIB$C_LENGTH,R2         ; set size of fib
                 FC11'   30 03EC        BSBW    RM$GETSPC1               ; allocate fib
                            03EF                                         ; build fib descriptor on stack
                    51   DD 03EF        PUSHL   R1                       ; addr of fib
            7E   40 8F   9A 03F1        MOVZBL  #FIB$C_LENGTH,-(SP)      ; and length of fib
                            03F5
                            03F5  ;
                            03F5  ;  handle "tef" option (truncate at end of file) if this is a write-accessed
                            03F5  ;  disk file.
                            03F5  ;
                            03F5
               23 A9   95 03F5         TSTB    IFB$B_ORGCASE(R9)                   ; check for seg file
                  26   12 03F8         BNEQ    20$                                 ; don't do it if not seq
               78 A9   D5 03FA         TSTL    IFB$L_SFSB_PTR(R9)                  ; check for shared file
                  21   12 03FD         BNEQ    20$                                 ; bypass if shared file
            0C 69   36   E0 03FF       BBS     #IFB$V_TEF,(R9),15$                 ; branch if auto extend set flag
            19 68   3C   E1 0403       BBC     #FAB$V_TEF+FOP,(R8),20$             ; branch if option not speced
            15 69   1C   E1 0407       BBC     #DEV$V_RND,IFB$L_PRIM_DEV(R9),20$ ; or if not disk
            11 69   30   E1 040B       BBC     #IFB$V_WRTACC,(R9),20$              ; or if not write accessed
                            040F
                            040F       ASSUME  FIB$V_TRUNC      GE       8
                            040F
            17 A1   01   88 040F  15$: BISB2   #<FIB$M_TRUNC @-8>,FIB$W_EXCTL+1(R1) ; ask for truncate
         1C A1   74 A9   D0 0413       MOVL    IFB$L_EBK(R9),FIB$L_EXVBN(R1)     ; truncate at eof block
               5C A9   B5 0418         TSTW    IFB$W_FFB(R9)                       ; any bytes used this block?
                  03   13 041B         BEQL    20$                                 ; branch if none
               1C A1   D6 041D         INCL    FIB$L_EXVBN(R1)                     ; yes - don't truncate block
                          0420
                          0420  ;
                          0420  ;  check for magtape rewind
                          0420  ;
                          0420
            07 69   05   E1 0420  20$: BBC     #DEV$V_SQD,IFB$L_PRIM_DEV(R9),40$ ; branch if not magtape
            03 69   27   E1 0424       BBC     #IFB$V_RWC,(R9),40$       ; branch if not speced
               61   08   88 0428       BISB2   #FIB$M_REWIND,FIB$L_ACCTL(R1) ; cause rewind to happen
                          042B
                          042B  ;
```

```
                              042B      ;      swap the words of ifb$l_hbk and ifb$l_ebk to match files-11
                              042B      ;      on-disk structure
                              042B      ;
                              042B
  54 A9    70 A9    10    9C  042B      40$:      ROTL     #16,IFB$L_HBK(R9),IFB$L_HBK_DISK(R9)
  58 A9    74 A9    10    9C  0431                ROTL     #16,IFB$L_EBK(R9),IFB$L_EBK_DISK(R9)
                              0437
                              0437      ;
                              0437      ;      do the deaccess qio
                              0437      ;
                              0437
            50      34    9A  0437                MOVZBL   #IO$_DEACCESS,R0             ; deaccess function code
                    00    DD  043A                PUSHL    #0                           ; p6 = 0 for qio
                 0C AE    DF  043C                PUSHAL   12(SP)                       ; p5 = address of attribute list
                 FBBE'    30  043F                BSBW     RM$FCPFNC_P4                 ; do the deaccess acp function
                    14    BA  0442                POPR     #^M<R2,R4>                   ; get fib len & addr
                    8E    D5  0444      50$:      TSTL     (SP)+                        ; remove attribute list from stack
                    FC    12  0446                BNEQ     50$                          ;             ''
                    50    DD  0448                PUSHL    R0                           ; save status code
                 FBB3'    30  044A                BSBW     RM$RETSPC1                   ; deallocate the fib
                    01    BA  044D                POPR     #^M<R0>                      ; restore the status code
                 07 50    E9  044F                BLBC     R0,ERRDAC                    ; branch if error
                 03 56    E8  0452                BLBS     R6,60$                       ; branch if no xab error
            50      56    D0  0455                MOVL     R6,R0                        ; report xab error
                          05  0458      60$:      RSB
                              0459
                              0459      ERRDAC:
                              0459                RMSERR   DAC,R1                       ; default error code
                 FB9F'    31  045F                BRW      RM$MAPERR                    ; go handle error
                              0461
                              0461                .END
```

```
$$.PSECT_EP                  = 00000000                 FAB$V_DFW                  = 00000005
$$RMSTEST                    = 0000001A                 FAB$V_EXE                  = 00000007
$$RMS_PBUGCHK                = 00000010                 FAB$V_GET                  = 00000001
$$RMS_TBUGCHK                = 00000008                 FAB$V_MSE                  = 00000004
$$RMS_UMODE                  = 00000004                 FAB$V_NEF                  = 0000000A
ARM$M_READ                   = 00000001                 FAB$V_NIL                  = 00000005
ATR$C_ACCESS_MODE            = 0000002D                 FAB$V_POS                  = 00000008
ATR$C_BLOCKSIZE              = 0000000B                 FAB$V_RCK                  = 00000017
ATR$C_BUFFER_OFFSET          = 00000030                 FAB$V_RU                   = 00000000
ATR$C_RECATTR                = 00000004                 FAB$V_RWO                  = 00000007
ATR$C_STATBLK                = 00000009                 FAB$V_SQO                  = 00000006
ATR$C_UCHAR                  = 00000003                 FAB$V_TEF                  = 0000001C
ATR$S_BLOCKSIZE              = 00000002                 FAB$V_TRN                  = 00000004
ATR$S_BUFFER_OFFSET          = 00000002                 FAB$V_UFO                  = 00000011
ATR$S_UCHAR                  = 00000004                 FAB$V_UPI                  = 00000006
BKP3                         = 00000007                 FAB$V_WCK                  = 00000009
CHKSEQSHR                      000001F4 R      01       FCH$V_CONTIG               = 00000007
CHKSHR                         000001CA R      01       FCH$V_CONTIGB              = 00000005
CLS_XAB_ARGS                   00000387 R      01       FCH$V_READCHECK            = 00000003
CTL$GL_PCB                     ********   X    01       FCH$V_WRITCHECK            = 00000004
CTL$GL_RUF                     ********   X    01       FIB$B_WSIZE                = 00000003
DEV$V_DIR                    = 00000003                 FIB$C_LENGTH               = 00000040
DEV$V_NET                    = 0000000D                 FIB$L_ACCTL                = 00000000
DEV$V_RND                    = 0000001C                 FIB$L_ALT_ACCESS           = 0000003C
DEV$V_SQD                    = 00000005                 FIB$L_EXVBN                = 0000001C
ERRACCESS                      0000022B R      01       FIB$L_STATUS               = 00000038
ERRDAC                         00000459 R      01       FIB$M_NOLOCK               = 00100000
EXIT                           000001AD R      01       FIB$M_NOREAD               = 00000400
FAB$B_FAC                    = 00000016                 FIB$M_NOWRITE              = 00000001
FAB$B_RCF                    = 0000004B                 FIB$M_REWIND               = 00000008
FAB$B_RTV                    = 0000001C                 FIB$M_TRUNC                = 00000100
FAB$B_SHR                    = 00000017                 FIB$V_ALT_GRANTED          = 00000001
FAB$C_FIX                    = 00000001                 FIB$V_CURPOS               = 00000004
FAB$C_SEQ                    = 00000000                 FIB$V_EXECUTE              = 00000010
FAB$L_FOP                    = 00000004                 FIB$V_NOREAD               = 0000000A
FAB$M_BIO                    = 00000020                 FIB$V_NOTRUNC              = 0000000B
FAB$M_BRO                    = 00000040                 FIB$V_PRSRV_ATR            = 00000011
FAB$M_CBT                    = 00200000                 FIB$V_READCK               = 00000009
FAB$M_CTG                    = 00100000                 FIB$V_REWIND               = 00000003
FAB$M_DEL                    = 00000004                 FIB$V_RMSLOCK              = 00000012
FAB$M_GET                    = 00000002                 FIB$V_SEQONLY              = 00000006
FAB$M_KFO                    = 40000000                 FIB$V_TRUNC                = 00000008
FAB$M_MSE                    = 00000010                 FIB$V_UPDATE               = 00000006
FAB$M_PUT                    = 00000001                 FIB$V_WRITE                = 00000008
FAB$M_RCK                    = 00800000                 FIB$V_WRITECK              = 00000005
FAB$M_SHRDEL                 = 00000004                 FIB$W_EXCTL                = 00000016
FAB$M_SHRGET                 = 00000002                 FOP                        = 00000020
FAB$M_SHRPUT                 = 00000001                 FWA$L_ATR_WORK             = 00000058
FAB$M_SHRUPD                 = 00000008                 FWA$L_HBK                  = 000001AC
FAB$M_UFO                    = 00020000                 FWA$Q_FIB                  = 00000010
FAB$M_UPD                    = 00000008                 FWA$Q_NAME                 = 00000170
FAB$M_WCK                    = 00000200                 FWA$Q_RNS                  = 00000188
FAB$V_AI                     = 00000001                 FWA$S_NAMEBUF              = 00000100
FAB$V_BI                     = 00000002                 FWA$S_STATBLK              = 0000000A
FAB$V_BIO                    = 00000005                 FWA$S_TYPEBUF              = 00000028
FAB$V_CBT                    = 00000015                 FWA$S_VERBUF               = 00000006
FAB$V_CTG                    = 00000014                 FWA$T_NAMEBUF              = 000004B6
```

```
FWA$T_STATBLK                    = 000001A8          NETMASK                        = 000000C0
FWA$V_NODE                       = 00000019          NT$ACCESS                        ********    X    01
FWA$W_UCHAR                      = 00000044          NT$CLOSE                         ********    X    01
IFB$B_BKS                        = 0000005E          NT$DEACCESS                      ********    X    01
IFB$B_FAC                        = 00000022          NT$NWA_FREE                      ********    X    01
IFB$B_JNLFLG                     = 000000A0          NT$OPEN                          ********    X    01
IFB$B_JNLFLG2                    = 000000A2          NT$SUP_FOP                       ********    X    01
IFB$B_MODE                       = 0000000A          NTACC                            00000012 R       01
IFB$B_ORGCASE                    = 00000023          NTDAC                            0000038E R       01
IFB$B_RECVRFLGS                  = 000000A1          P4_P2                            00000365 R       01
IFB$B_RFMORG                     = 00000050          PCB$L_STS                      = 00000024
IFB$B_SHR                        = 0000004E          PCB$V_RECOVER                  = 0000001A
IFB$C_FHAEND                     = 00000066          PIO$A_TRACE                      ********    X    01
IFB$C_IDX                        = 00000002          PSL$C_EXEC                     = 00000001
IFB$C_REL                        = 00000001          PSL$C_SUPER                    = 00000002
IFB$C_SEQ                        = 00000000          RET                              00000045 R       01
IFB$L_BLBBLNK                    = 0000009C          RETURN                           000001C5 R       01
IFB$L_BLBFLNK                    = 00000098          RJB$M_OPEN                     = 00000010
IFB$L_DEVBUFSIZ                  = 00000048          RJB$V_AT                       = 00000003
IFB$L_DVBN                       = 000000B0          RJB$W_FLAGS                    = 0000000A
IFB$L_EBK                        = 00000074          RM$ACCESS                        00000000 RG      01
IFB$L_EBK_DISK                   = 00000058          RM$ALBLB                         ********    X    01
IFB$L_HBK                        = 00000070          RM$ASSJNL                        ********    X    01
IFB$L_HBK_DISK                   = 00000054          RM$CREACC_SET1                   00000233 RG      01
IFB$L_PRIM_DEV                   = 00000000          RM$CREACC_SET2                   0000031F RG      01
IFB$L_RJB                        = 000000A4          RM$CREACC_SET3                   00000339 RG      01
IFB$L_SFSB_PTR                   = 00000078          RM$DEACCESS                      000003B1 RG      01
IFB$M_NEVER_RU                   = 00000020          RM$FCPFNC                        ********    X    01
IFB$M_ONLY_RU                    = 00000001          RM$FCPFNC_P4                     ********    X    01
IFB$M_RU                         = 00000002          RM$FCP_P4_P2                     00000383 RG      01
IFB$S_ORG                        = 00000004          RM$GETTPAG                       ********    X    01
IFB$V_ACCESSED                   = 00000025          RM$GETSPC1                       ********    X    01
IFB$V_AI_RECVR                   = 00000001          RM$INIT_SFSB                     ********    X    01
IFB$V_BI_RECVR                   = 00000002          RM$MAPERR                        ********    X    01
IFB$V_DAP                        = 0000003E          RM$MAPJNL                        ********    X    01
IFB$V_DAP_OPEN                   = 0000003D          RM$MAPJNL_RU                     ********    X    01
IFB$V_DFW                        = 0000002C          RM$OPEN_XAB                      ********    X    01
IFB$V_MSE                        = 00000031          RM$OPEN_XAB1                     ********    X    01
IFB$V_NORECLK                    = 00000033          RM$RET1PAG                       ********    X    01
IFB$V_NSP                        = 0000003F          RM$RETSPC1                       ********    X    01
IFB$V_ORG                        = 00000004          RM$RTVJNL                        ********    X    01
IFB$V_RU                         = 00000001          RM$SETEBK                        00000109 RG      01
IFB$V_RUP                        = 00000002          RM$SETHBK                        000000B7 RG      01
IFB$V_RU_RECVR                   = 00000000          RM$XAB_SCAN                      ********    X    01
IFB$V_RU_RLK                     = 00000003          RMACC                            00000046 R       01
IFB$V_RWC                        = 00000027          RMS$_ACC                       = 0001C002
IFB$V_SEQFIL                     = 00000038          RMS$_DAC                       = 0001C012
IFB$V_SQO                        = 0000002D          RMS$_IFF                       = 00018804
IFB$V_TEF                        = 00000036          RMS$_NRU                       = 000187FC
IFB$V_WRTACC                     = 00000030          RMS$_SHR                       = 000186B4
IFB$W_BUFFER_OFFSET              = 000000A8          RMS$_UPI                       = 000187AC
IFB$W_FFB                        = 0000005C          RUCB$B_CTRL                    = 00000011
IFB$W_LRL                        = 00000052          RUCB$V_ACTIVE                  = 00000001
IMP$V_IORUNDOWN                  = 00000004          SETNORECLK                       000001C6 R       01
IO$M_ACCESS                      = 00000040          SETRTV                           000002FA R       01
IO$_ACCESS                       = 00000032          SHARE                            000001D9 R       01
IO$_DEACCESS                     = 00000034          SHRCHK                           000001A9 R       01
```

```
                                                         C 11
RMOACCESS                            ACCESS/DEACCESS ROUTINES          16-SEP-1984 00:09:38  VAX/VMS Macro V04-00         Page  28
Symbol table                                                           14-SEP-1984 22:32:30  [RMS.SRC]RMOACCESS.MAR;2           (12)

SHRERR                               00000225 R      01
TPT$L_ACCESS                         ********    X   01
TPT$L_DEACCES                        ********    X   01
UPIERR                               000001EE R      01
XAB$C_PRO                          = 00000013
XAB$C_PROLEN_V3                    = 00000010
XAB$C_RDT                          = 0000001E
XAB$C_RDTLEN                       = 00000014
XBC$C_CLSPRO                         ********    X   01
XBC$C_CLSRDT                         ********    X   01

                                                +----------------+
                                                ! Psect synopsis !
                                                +----------------+

PSECT name                           Allocation          PSECT No.  Attributes
----------                           ----------          ---------  ----------
.  ABS  .                            00000000  (     0.)  00 (  0.)  NOPIC    USR    CON    ABS    LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
RM$RMS0                              00000461  (  1121.)  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    00:00:01.24
Command processing                   127    00:00:00.69    00:00:05.33
Pass 1                               495    00:00:20.28    00:00:53.79
Symbol table sort                      2    00:00:03.04    00:00:05.93
Pass 2                               206    00:00:04.55    00:00:09.88
Symbol table output                   28    00:00:00.20    00:00:00.30
Psect synopsis output                  1    00:00:00.02    00:00:00.02
Cross-reference output                 0    00:00:00.00    00:00:00.00
Assembler run totals                 890    00:00:28.84    00:01:16.49

The working set limit was 1800 pages.
116398 bytes (228 pages) of virtual memory were used to buffer the intermediate code.
There were 110 pages of symbol table space allocated to hold 2087 non-local and 70 local symbols.
1168 source lines were read in Pass 1, producing 17 object records in Pass 2.
34 pages of virtual memory were used to define 33 macros.

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

Macro library name                          Macros defined
------------------                          --------------
_$255$DUA28:[RMS.OBJ]RMS.MLB;1                          15
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                           4
_$255$DUA28:[SYSLIB]STARLET.MLB;2                       10
TOTALS (all libraries)                                  29

2213 GETS were required to define 29 macros.

There were no errors, warnings or information messages.
```

```
RMOACCESS                        ACCESS/DEACCESS ROUTINES                  16-SEP-1984 00:09:38  VAX/VMS Macro V04-00         Page  29
VAX-11 Macro Run Statistics                                                14-SEP-1984 22:32:30  [RMS.SRC]RMOACCESS.MAR;2           (12)

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

RMO
V04

0317 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

NT0SCNXAB
LIS

RM0CACHE
LIS

NT0RENAME
LIS

NT0OPEN
LIS

RM0BUFMGR
LIS

NT0SEARCH
LIS

NT0PUT
LIS

RM0ACCESS
LIS