```
FFFFFFFFFFFFFFF      AAAAAAAAA     LLL
FFFFFFFFFFFFFFF      AAAAAAAAA     LLL
FFFFFFFFFFFFFFF      AAAAAAAAA     LLL
FFF               AAA         AAA  LLL
FFF               AAA         AAA  LLL
FFF               AAA         AAA  LLL
FFF               AAA         AAA  LLL
FFF               AAA         AAA  LLL
FFF               AAA         AAA  LLL
FFFFFFFFFFFF      AAA         AAA  LLL
FFFFFFFFFFFF      AAA         AAA  LLL
FFFFFFFFFFFF      AAA         AAA  LLL
FFF               AAAAAAAAAAAAAAA  LLL
FFF               AAAAAAAAAAAAAAA  LLL
FFF               AAAAAAAAAAAAAAA  LLL
FFF               AAA         AAA  LLL
FFF               AAA         AAA  LLL
FFF               AAA         AAA  LLL
FFF               AAA         AAA  LLLLLLLLLLLLLLL
FFF               AAA         AAA  LLLLLLLLLLLLLLL
FFF               AAA         AAA  LLLLLLLLLLLLLLL
```

```
FFFFFFFFFF    AAAAAA    LL            AAAAAA     CCCCCCCC  TTTTTTTTTT  MM      MM    SSSSSSSS    GGGGGGGG
FFFFFFFFFF    AAAAAA    LL            AAAAAA     CCCCCCCC  TTTTTTTTTT  MM      MM    SSSSSSSS    GGGGGGGG
FF           AA    AA   LL           AA    AA   CC            TT      MMMM  MMMM   SS          GG
FF           AA    AA   LL           AA    AA   CC            TT      MMMM  MMMM   SS          GG
FF           AA    AA   LL           AA    AA   CC            TT      MM  MM  MM   SS          GG
FF           AA    AA   LL           AA    AA   CC            TT      MM  MM  MM   SS          GG
FFFFFFFF     AA    AA   LL           AA    AA   CC            TT      MM      MM     SSSSSS    GG
FFFFFFFF     AA    AA   LL           AA    AA   CC            TT      MM      MM     SSSSSS    GG
FF           AAAAAAAAAA LL           AAAAAAAAAA CC            TT      MM      MM           SS  GG  GGGGGG
FF           AAAAAAAAAA LL           AAAAAAAAAA CC            TT      MM      MM           SS  GG  GGGGGG
FF           AA    AA   LL           AA    AA   CC            TT      MM      MM           SS  GG      GG      ....
FF           AA    AA   LL           AA    AA   CC            TT      MM      MM           SS  GG      GG      ....
FF           AA    AA   LLLLLLLLLL   AA    AA     CCCCCCCC    TT      MM      MM    SSSSSSSS     GGGGGG       ....
FF           AA    AA   LLLLLLLLLL   AA    AA     CCCCCCCC    TT      MM      MM    SSSSSSSS     GGGGGG       ....

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          .TITLE  FALACTMSG - STATE TABLE ACTION ROUTINES
0000     2          .IDENT  'V04-000'
0000     3
0000     4 ;
0000     5 ;**********************************************************************
0000     6 ;*                                                                    *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                           *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.            *
0000     9 ;*  ALL RIGHTS RESERVED.                                              *
0000    10 ;*                                                                    *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
0000    12 ;*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE    *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER  *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
0000    15 ;*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY  *
0000    16 ;*  TRANSFERRED.                                                      *
0000    17 ;*                                                                    *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE  *
0000    19 ;*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT  *
0000    20 ;*  CORPORATION.                                                      *
0000    21 ;*                                                                    *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS  *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.           *
0000    24 ;*                                                                    *
0000    25 ;*                                                                    *
0000    26 ;**********************************************************************
0000    27 ;
0000    28
0000    29 ;++
0000    30 ; Facility: FAL (DECnet File Access Listener)
0000    31 ;
0000    32 ; Abstract:
0000    33 ;
0000    34 ;       This module contains action routines called by the state table manager.
0000    35 ;
0000    36 ; Environment: VAX/VMS, user mode
0000    37 ;
0000    38 ; Author: James A. Krycka,      Creation Date:  16-JUN-1977
0000    39 ;
0000    40 ; Modified By:
0000    41 ;
0000    42 ;       V03-007 JAK0136         J A Krycka      07-MAR-1984
0000    43 ;               Support FAL logging options that deal with fields in the DAP
0000    44 ;               Configuration message sent to partner.
0000    45 ;
0000    46 ;       V03-006 JAK0118         J A Krycka      12-JUL-1983
0000    47 ;               Fix bug in processing the DAP KEY field.
0000    48 ;
0000    49 ;       V03-005 KRM0104         K Malik         10-May-1983
0000    50 ;               Update symbols to match Dap V7.0 spec.
0000    51 ;
0000    52 ;       V03-004 JAK0104         J A Krycka      29-APR-1983
0000    53 ;               Make minor enhancements to FAL logging display.
0000    54 ;
0000    55 ;       V03-003 KRM0083         K Malik         23-Mar-1983
0000    56 ;               Add support for STMLF and STMCR formats.
0000    57 ;
```

```
0000     58 ;        V03-002 KRM0074          K Malik         23-Nov-1982
0000     59 ;                Added FAL$DECODE_NAM routine (to support $RENAME function).
0000     60 ;
0000     61 ;        V03-001 JAK0101          J A Krycka      09-OCT-1982
0000     62 ;                Fix bug in converting DAP OWNER value into binary format.
0000     63 ;
0000     64 ;--
```

```
0000     66          .SBTTL  DECLARATIONS
0000     67
0000     68 ;
0000     69 ; Include Files:
0000     70 ;
0000     71
0000     72          $DAPPLGDEF                      ; Define DAP prologue symbols
0000     73          $DAPHDRDEF                      ; Define DAP message header
0000     74          $DAPSSPDEF                      ; Define DAP system specific field
0000     75          $DAPCNFDEF                      ; Define DAP Configuration message
0000     76          $DAPATTDEF                      ; Define DAP Attributes message
0000     77          $DAPACCDEF                      ; Define DAP Access message
0000     78          $DAPCTLDEF                      ; Define DAP Control message
0000     79          $DAPCONDEF                      ; Define DAP Continue Transfer message
0000     80          $DAPCMPDEF                      ; Define DAP Access Complete message
0000     81          $DAPKEYDEF                      ; Define DAP Key Definition message
0000     82          $DAPALLDEF                      ; Define DAP Allocation message
0000     83          $DAPTIMDEF                      ; Define DAP Date and Time message
0000     84          $DAPPRODEF                      ; Define DAP Protection message
0000     85          $DAPNAMDEF                      ; Define DAP Name message
0000     86          $DEVDEF                         ; Define Device Characteristics symbols
0000     87          $FABDEF                         ; Define File Access Block symbols
0000     88          $FALWRKDEF                      ; Define FAL Work Area symbols
0000     89          $RABDEF                         ; Define Record Access Block sym**
0000     90          $XABDEF                         ; Define symbols common to all XABs
0000     91          $XABALLDEF                      ; Define Allocation XAB symbols
0000     92          $XABDATDEF                      ; Define Date and Time XAB symbols
0000     93          $XABKEYDEF                      ; Define Key Definition XAB symbols
0000     94          $XABPRODEF                      ; Define Protection XAB symbols
0000     95 ;        $XABRDTDEF                      ; Define Revision Date and Time symbols
0000     96
0000     97 ;
0000     98 ; Macros:
0000     99 ;
0000    100 ;       None
0000    101 ;
0000    102 ; Equated Symbols:
0000    103 ;
0000    104 ;
0000    105
0000    106          ASSUME  DAP$Q_DCODE_FLG EQ 0
0000    107          ASSUME  FAL$Q_FLG EQ 0
0000    108
0000    109 ;
0000    110 ; Own Storage:
0000    111 ;
```

```
                        .SBTTL  ACTION ROUTINES
00000000                .PSECT  FAL$CODE        NOSHR,EXE,RD,NOWRT,BYTE

                ;++
                ; Functional Description:
                ;
                ;       This module contains action routines invoked by the state table
                ;       manager (FAL$STATE).
                ;
                ;       The input parameters and completion codes listed below are applicable
                ;       for all of these action routines. Note that an action routine may use
                ;       R0-R7 and AP without restoring them on exit. R0 on exit, however, must
                ;       represent a status code to indicate success/failure of the routine or
                ;       a true/false condition, as appropriate. This status code is used by
                ;       the state table manager to advance to the next state.
                ;
                ; Calling Sequence:
                ;
                ;       BSBW    FAL$name
                ;
                ; Input Parameters:
                ;
                ;       R8      Address of FAL work area
                ;       R9      Address of DAP control block
                ;       R10     Address of FAB
                ;       R11     Address of RAB
                ;
                ; Implicit Inputs:
                ;
                ;       None
                ;
                ; Output Parameters:
                ;
                ;       R0      Status code
                ;       R1-R7   Destroyed
                ;       AP      Destroyed
                ;
                ; Implicit Outputs:
                ;
                ;       None
                ;
                ; Completion Codes:
                ;
                ;       R0      1 = success; 0 = failure
                ;
                ; Side Effects:
                ;
                ;       None
                ;
                ;--
```

```
                                0000   164          .SBTTL  FAL$DECODE_CNF
                                0000   165
                                0000   166 ;++
                                0000   167 ; Process the Configuration message which has been received and validated.
                                0000   168 ; Return a Configuration message to partner and determine the DAP buffer size
                                0000   169 ; to use which is the smaller of partner's buffer size and FAL's buffer size.
                                0000   170 ;--
                                0000   171
                                0000   172 FAL$DECODE_CNF::                          ; Entry point
                                0000   173          $SETBIT #FAL$V_CNF_MSG,(R8)     ; Denote Configuration message received
                                0004   174          $CLRBIT #FAL$V_ATT_MSG,(R8)     ;  and discard any previous Attributes
                                0008   175                                          ;  message
      57    18 A8    3C        0008   176          MOVZWL  FAL$W_QIOBUFSIZ(R8),R7  ; Get FAL's buffer size (i.e., largest
                                000C   177                                          ;  I/O buffer size supported by process)
         68    39    E1        000C   178          BBC     #FAL$V_USE_DBS,(R8),-   ; Branch to use calculated buffer size
                     05         000F   179                  SEND_CNF                ;
      57    00A0 C8    3C      0010   180          MOVZWL  FAL$W_USE_DBS(R8),R7    ; Override with user specified value
                                0015   181
                                0015   182 ;+
                                0015   183 ; Build and send Configuration message to partner.
                                0015   184 ;-
                                0015   185
                                0015   186 SEND_CNF:                              ;
                                0015   187          $SETBIT #FAL$V_LAST_MSG,(R8)    ; Declare this last message to block
         50    01    D0        0019   188          MOVL    #DAP$K_CNF_MSG,R0       ; Get message type value
            FFE1'    30        001C   189          BSBW    FAL$BUILD_HEAD          ; Construct message header
         83    57    B0        001F   190          MOVW    R7,(R3)+                ; Store BUFSIZ field
         83    07    90        0022   191          MOVB    #DAP$K_VAXVMS,(R3)+     ; Store OSTYPE field
         83    03    90        0025   192          MOVB    #DAP$K_RMS32,(R3)+      ; Store FILESYS field
      06 68    3A    E1        0028   193          BBC     #FAL$V_USE_SYS,(R8),2$  ; Branch to use standard values
   FE A3    00A2 C8    B0      002C   194          MOVW    FAL$W_USE_SYS(R8),-2(R3); Override with user specified values
         83    07    90        0032   195 2$:      MOVB    #DAP$K_VERNUM_V,(R3)+   ; Store VERNUM field
         83    00    90        0035   196          MOVB    #DAP$K_ECONUM_V,(R3)+   ; Store ECONUM field
         83    C0    90        0038   197          MOVB    #DAP$K_USRNUM_V,(R3)+   ; Store USRNUM field
         83    04    90        003B   198          MOVB    #DAP$K_DECVER_V,(R3)+   ; Store DECVER field
      06 68    3B    E1        003E   199          BBC     #FAL$V_USE_VER,(R8),4$  ; Branch to use standard values
   FC A3    00A4 C8    D0      0042   200          MOVL    FAL$L_USE_VER(R8),-4(R3); Override with user specified values
         83    00    90        0048   201 4$:      MOVB    #DAP$K_USRVER_V,(R3)+   ; Store USRVER field
                                004B   202
                                004B   203 ;
                                004B   204 ; Construct the system capabilities field.
                                004B   205 ; Also, check the debugging options to disable message blocking and DAP level
                                004B   206 ; CRC checking (after any user specified system capabilites bitmasks, if any,
                                004B   207 ; have been applied).
                                004B   208 ;
                                004B   209
   51    EFF67DF7 8F    D0      004B   210          MOVL    #DAP$K_SYSCAP1_V,R1     ; Get VAX supported capabilities
   52    00001962 8F    D0      0052   211          MOVL    #DAP$K_SYSCAP2_V,R2     ;  quadword bitmask
                                0059   212                                          ; ------ process debugging options -----
      05 68    3C    E1        0059   213          BBC     #FAL$V_USE_SC1,(R8),6$  ; Branch to use standard values
      51    00A8 C8    D0      005D   214          MOVL    FAL$L_USE_SC1(R8),R1    ; Override with user specified values
      05 68    3D    E1        0062   215 6$:      BBC     #FAL$V_USE_SC2,(R8),8$  ; Branch to use standard values
      52    00AC C8    D0      0066   216          MOVL    FAL$L_USE_SC2(R8),R2    ; Override with user specified values
      0F 68    31    E1        006B   217 8$:      BBC     #FAL$V_DIS_MBK,(R8),10$ ; Is DAP message blocking disabled?
   51    00140000 8F    CA      006F   218          BICL2   #<<1@DAP$V_MSGBLK>!-    ; Yes, clear message blocking bits in
                                0076   219                    <1@DAP$V_BIGBLK>!-    ;  system capabilities bitmask for
                                0076   220                    0>,R1                 ;  Configuration message to transmit
```

```
                28 A9   00140000 8F    CA   0076   221          BICL2   #<<1@DAP$V_MSGBLK>!-     ; Also, clear message blocking bits in
                                              007E   222                    <1@DAP$V_BIGBLK>!-     ;  system capabilities bitmask
                                              007E   223                  0>,DAP$Q_SYSCAP(R9)      ;  received from partner
                     09 68    30    E1       007E   224 10$:     BBC     #FAL$V_DIS_CRC,(R8),20$ ; Is file level CRC checksum disabled?
                                              0082   225          $CLRBIT #DAP$V_DAPCRC,R1         ; Yes, clear bits in both XMT and RCV
                                              0086   226          $CLRBIT #DAP$V_DAPCRC,-          ;  system capabilities fields
                                              0086   227                  DAP$Q_SYSCAP(R9)         ;
                                              008B   228                                           ; ------ finish debugging options -----
                          FF72'   30          008B   229 20$:     BSBW    FAL$CVT_BN8_EXT          ; Store SYSCAP as an extensible field
                          FF6F'   30          008E   230          BSBW    FAL$BUILD_TAIL           ; Finish building message
                          FF6C'   30          0091   231          BSBW    FAL$TRANSMIT             ;  end Configuration message
                                              0094   232
                                              0094   233 ;+
                                              0094   234 ; Determine the 'agreed upon' DAP buffer size to use and save this value.
                                              0094   235 ; It is the smaller of partner's buffer size and FAL's maximum buffer size.
                                              0094   236 ;-
                                              0094   237
                        40 A9    B0           0094   238          MOVW    DAP$W_BUFSIZ(R9),-       ; Assume we'll use partner's
                        1A A8                 0097   239                  FAL$W_DAPBUFSIZ(R8)      ;  buffer size
                           06    13           0099   240          BEQL    30$                      ; Branch if partner has unlimited space
                     57    40 A9    B1       009B   241          CMPW    DAP$W_BUFSIZ(R9),R7      ; Compare partner's buffer size with
                                              009F   242                                           ;  our buffer size
                           04    1B           009F   243          BLEQU   40$                      ; Branch if partner has less capacity
                     1A A8    57    B0       00A1   244 30$:     MOVW    R7,FAL$W_DAPBUFSIZ(R8)   ; We guessed wrong, so we'll use
                                              00A5   245                                           ;  our buffer size
                          04E4   31          00A5   246 40$:     BRW     EXIT_SUCCESS             ; Exit state with success
```

```
                        00A8   248          .SBTTL  FAL$DECODE_ATT
                        00A8   249
                        00A8   250 ;++
                        00A8   251 ; Process the Attributes message which has been received and validated.
                        00A8   252 ; Update the FAB and FHCXAB with information from this message.
                        00A8   253 ;--
                        00A8   254
                        00A8   255 FAL$DECODE_ATT::                     ; Entry point
                        00A8   256
                        00A8   257          $SETBIT #FAL$V_ATT_MSG,(R8)     ; Denote Attributes message received
                        00AC   258
                        00AC   259 ;
                        00AC   260 ; Save the DAP DATATYPE field for use later.
                        00AC   261 ;
                        00AC   262
01F4 C8   44 A9   90    00AC   263          MOVB    DAP$B_DATATYPE(R9),FAL$B_DATATYPE(R8)
                        00B2   264
                        00B2   265 ;
                        00B2   266 ; Process the DAP ORG, RFM and RAT fields.
                        00B2   267 ;
                        00B2   268
                        00B2   269          ASSUME  DAP$K_SEQ EQ FAB$C_SEQ
                        00B2   270          ASSUME  DAP$K_REL EQ FAB$C_REL
                        00B2   271          ASSUME  DAP$K_IDX EQ FAB$C_IDX
                        00B2   272
 1D AA   45 A9   90     00B2   273          MOVB    DAP$B_ORG(R9),FAB$B_ORG(R10)
                        00B7   274
                        00B7   275          ASSUME  DAP$K_UDF EQ FAB$C_UDF
                        00B7   276          ASSUME  DAP$K_FIX EQ FAB$C_FIX
                        00B7   277          ASSUME  DAP$K_VAR EQ FAB$C_VAR
                        00B7   278          ASSUME  DAP$K_VFC EQ FAB$C_VFC
                        00B7   279          ASSUME  DAP$K_STM EQ FAB$C_STM
                        00B7   280          ASSUME  DAP$K_STMLF EQ FAB$C_STMLF
                        00B7   281          ASSUME  DAP$K_STMCR EQ FAB$C_STMCR
                        00B7   282
1F AA   46 A9   90      00B7   283          MOVB    DAP$B_RFM(R9),FAB$B_RFM(R10)
                        00BC   284
                        00BC   285          ASSUME  DAP$V_FTN EQ FAB$V_FTN
                        00BC   286          ASSUME  DAP$V_CR  EQ FAB$V_CR
                        00BC   287          ASSUME  DAP$V_PRN EQ FAB$V_PRN
                        00BC   288          ASSUME  DAP$V_BLK EQ FAB$V_BLK
                        00BC   289
1E AA   47 A9   90      00BC   290          MOVB    DAP$B_RAT(R9),FAB$B_RAT(R10)
           10   8A      00C1   291          BICB2   #DAP$M_EMBEDDED,-       ; Ignore this bit
        1E AA           00C3   292                  FAB$B_RAT(R10)          ;
   0A 69    34   E0     00C5   293          BBS     #DAP$V_VAXVMS,(R9),10$  ; Branch if partner is VAX/VMS
   04   46 A9   91      00C9   294          CMPB    DAP$B_RFM(R9),#DAP$K_STM; Branch if not stream format
             04   12     00CD   295          BNEQ    10$                     ;
   1E AA    02   90      00CF   296          MOVB    #FAB$M_CR,FAB$B_RAT(R10); If it is, declare cc to be implied
                        00D3   297
                        00D3   298 ;
                        00D3   299 ; Process the DAP BLS, MRS, ALQ, BKS, FSZ, MRN, and DEQ fields.
                        00D3   300 ;
                        00D3   301
3C AA   48 A9   B0      00D3   302 10$:    MOVW    DAP$W_BLS(R9),FAB$W_BLS(R10)
36 AA   4A A9   B0      00D8   303          MOVW    DAP$W_MRS(R9),FAB$W_MRS(R10)
10 AA   4C A9   D0      00DD   304          MOVL    DAP$L_ALQ1(R9),FAB$L_ALQ(R10)
```

```
3E AA   50 A9   90  00E2   305          MOVB    DAP$B_BKS(R9),FAB$B_BKS(R10)
3F AA   51 A9   90  00E7   306          MOVB    DAP$B_FSZ(R9),FAB$B_FSZ(R10)
38 AA   58 A9   D0  00EC   307          MOVL    DAP$L_MRN(R9),FAB$L_MRN(R10)
14 AA   54 A9   B0  00F1   308          MOVW    DAP$W_DEQ1(R9),FAB$W_DEQ(R10)
                    00F6   309
                    00F6   310 ;
                    00F6   311 ; Process the DAP FOP field after saving it for use later.
                    00F6   312 ;
                    00F6   313
   51   64 A9   D0  00F6   314          MOVL    DAP$L_FOP1(R9),R1       ; Get DAP FOP bits and
01F8 C8   51   D0  00FA   315          MOVL    R1,FAL$L_FOP(R8)        ;  save field for use later
        0342   30  00FF   316          BSBW    MAP_FOP_FIELD           ; Update FOP in FAB
                    0102   317
                    0102   318 ;
                    0102   319 ; Process the DAP LRL field.
                    0102   320 ; This is the only FHCXAB field that is input to RMS, and then only for the
                    0102   321 ; $CREATE function where the record format is variable or VFC.
                    0102   322 ;
                    0102   323
      70 A9   B0    0102   324          MOVW    DAP$W_LRL(R9),-         ; Copy value to FHCXAB
   02F4'C8          0105   325                  FAL$L_FHCXAB+XAB$W_LRL(R8)
        0481   31   0108   326          BRW     EXIT_SUCCESS            ; Exit state with success
```

```
                        010B   328          .SBTTL  FAL$DECODE_ACC
                        010B   329
                        010B   330 ;++
                        010B   331 ; Process the Access message which has been received and validated.
                        010B   332 ; Update the FAB with information from this message.
                        010B   333 ;--
                        010B   334
                        010B   335 FAL$DECODE_ACC::                         ; Entry point
                        010B   336
                        010B   337 ;
                        010B   338 ; Save the DAP ACCFUNC, ACCOPT, and DISPLAY fields for use later.
                        010B   339 ;
                        010B   340
01F6 C8  40 A9   90     010B   341          MOVB    DAP$B_ACCFUNC(R9),FAL$B_ACCFUNC(R8)
01F5 C8  41 A9   90     0111   342          MOVB    DAP$B_ACCOPT(R9),FAL$B_ACCOPT(R8)
  70 A8  4C A9   B0     0117   343          MOVW    DAP$W_DISPLAY1(R9),FAL$W_DISPLAY(R8)
                        011C   344
                        011C   345 ;
                        011C   346 ; Process the DAP file specification field.
                        011C   347 ;
                        011C   348
         44 A9   90     011C   349          MOVB    DAP$Q_FILESPEC(R9),-    ; Store size of filespec string
         34 AA          011F   350                  FAB$B_FNS(R10)          ;  in FAB
         44 A9   28     0121   351          MOVC3   DAP$Q_FILESPEC(R9),-    ; Copy filespec string to buffer
         48 B9          0124   352                  @DAP$Q_FILESPEC+4(R9),- ;
         2C BA          0126   353                  @FAB$L_FNA(R10)         ;
      51 40 A9   9A     0128   354          MOVZBL  DAP$B_ACCFUNC(R9),R1    ; Get access function code
      52 44 A9   7E     012C   355          MOVAQ   DAP$Q_FILESPEC(R9),R2   ; Get address of filename descriptor
          FECD'  30     0130   356          BSBW    FAL$LOG_REQNAM          ; Log requested name in print file
                        0133   357
                        0133   358 ;
                        0133   359 ; Process the DAP FAC field.
                        0133   360 ;
                        0133   361
                        0133   362          ASSUME  DAP$V_PUT EQ FAB$V_PUT
                        0133   363          ASSUME  DAP$V_GET EQ FAB$V_GET
                        0133   364          ASSUME  DAP$V_DEL EQ FAB$V_DEL
                        0133   365          ASSUME  DAP$V_UPD EQ FAB$V_UPD
                        0133   366          ASSUME  DAP$V_TRN EQ FAB$V_TRN
                        0133   367          ASSUME  DAP$V_BIO EQ FAB$V_BIO
                        0133   368          ASSUME  DAP$V_BRO EQ FAB$V_BRO
                        0133   369          ASSUME  DAP$V_APP EQ FAB$V_EXE  ; Map APP to PUT
                        0133   370
16 AA   42 A9   90      0133   371          MOVB    DAP$B_FAC(R9),FAB$B_FAC(R10)
05 16 AA    07  E5      0138   372          BBCC    #FAB$V_EXE,FAB$B_FAC(R10),10$
00 16 AA    00  E2      013D   373          BBSS    #FAB$V_PUT,FAB$B_FAC(R10),10$
                        0142   374
                        0142   375 ;
                        0142   376 ; Process the DAP SHR field.
                        0142   377 ;
                        0142   378
         52   D4        0142   379 10$:     CLRL    R2                      ; Clear RMS SHR bits
      51 43 A9   9A     0144   380          MOVZBL  DAP$B_SHR(R9),R1        ; Get DAP SHR bits
           30   13      0148   381          BEQL    20$                     ; Branch if no bits to map
                        014A   382          $MAPBIT DAP$V_SHRPUT,FAB$V_SHRPUT; Map SHRPUT bit
                        0152   383          $MAPBIT DAP$V_SHRGET,FAB$V_SHRGET; Map SHRGET bit
                        015A   384          $MAPBIT DAP$V_SHRDEL,FAB$V_SHRDEL; Map SHRDEL bit
```

```
                     0162   385          $MAPBIT DAP$V_SHRUPD,FAB$V_SHRUPD; Map SHRUPD bit
                     016A   386          $MAPBIT DAP$V_UPI,FAB$V_UPI      ;  Map UPI bit
                     0172   387          $MAPBIT DAP$V_NIL,FAB$V_NIL      ;  Map NIL bit
17 AA   52   90      017A   388 20$:     MOVB    R2,FAB$B_SHR(R10)        ; Update SHR field in FAB
                     017E   389
                     017E   390 ;
                     017E   391 ; Use the ACCFUNC field value as the next state table value.
                     017E   392 ;
                     017E   393
     40 A9   90      017E   394          MOVB    DAP$B_ACCFUNC(R9),-      ; Store new state transition value
     10 A8           0181   395                  FAL$B_VALUE(R8)          ;
        00   E1      0183   396          BBC     #DAP$V_LOAD,-            ; Branch if no system specific
0A 0084   C9         0185   397                  DAP$L_SSP_FLG(R9),30$    ;  function modifier found
                     0189   398          $SETBIT #FAB$V_SQO,FAB$L_FOP(R10)
                     018E   399                                           ; Force sequential file transfer
                     018E   400                                           ;  mode for efficiency
     FF 8F   90      018E   401          MOVB    #DAP$K_LOAD,-            ; Make new state transition value
     10 A8           0191   402                  FAL$B_VALUE(R8)          ;  the load image function
      03F6   31      0193   403 30$:     BRW     EXIT_SUCCESS             ; Exit state with success
```

```
                     0196   405          .SBTTL  FAL$DECODE_CTL
                     0196   406
                     0196   407 ;++
                     0196   408 ; Process the Control message which has been received and validated.
                     0196   409 ; Update the RAB with information from this message.
                     0196   410 ;--
                     0196   411
                     0196   412 FAL$DECODE_CTL::                          ; Entry point
                     0196   413
                     0196   414 ;+
                     0196   415 ; Save the DAP DISPLAY field for use later if we're not in a wildcard context.
                     0196   416 ; In wildcard file retrieval, for example, the DAP Access message is sent only
                     0196   417 ; once, thus FAL$W_DISPLAY must reflect the DISPLAY value from the Access
                     0196   418 ; message on subsequent file opens. Since the Control message functions of
                     0196   419 ; DISPLAY and EXTEND are not valid in a wildcard context (which require
                     0196   420 ; FAL$W_DISPLAY to be updated), this special check is an acceptible solution
                     0196   421 ; to a wildcard retrieval problem.
                     0196   422 ;-
                     0196   423
 68    0A    E0      0196   424          BBS     #FAL$V_WILD,(R8),-      ; Branch if wildcard operation
     05               0199   425                  RAC_FIELD               ;
   54 A9   B0        019A   426          MOVW    DAP$W_DISPLAY2(R9),-    ; Save display message bitmask in
   70 A8              019D   427                  FAL$W_DISPLAY(R8)       ;  FAL work area
                     019F   428
                     019F   429 ;+
                     019F   430 ; Process the DAP RAC field.
                     019F   431 ; In addition to normal RMS-32 RAC information, this field specifies whether
                     019F   432 ; the access is to be in:
                     019F   433 ;   (1) file transfer mode or record transfer mode
                     019F   434 ;   (2) block I/O mode or record I/O mode
                     019F   435 ;
                     019F   436 ; Note: If the RAC field is not present in the Control message, then the default
                     019F   437 ;       action is to use the previous value.
                     019F   438 ;-
                     019F   439
                     019F   440          ASSUME  DAP$K_SEQ_ACC  EQ 0
                     019F   441          ASSUME  DAP$K_KEY_ACC  EQ 1
                     019F   442          ASSUME  DAP$K_RFA_ACC  EQ 2
                     019F   443          ASSUME  DAP$K_SEQ_FILE EQ 3
                     019F   444          ASSUME  DAP$K_BLK_VBN  EQ 4
                     019F   445          ASSUME  DAP$K_BLK_FILE EQ 5
                     019F   446
                     019F   447 RAC_FIELD:                                ; Process record access field
     00    E0        019F   448          BBS     #DAP$V_RAC,-            ; Branch if RAC field was explicitly
06 44 A9             01A1   449                  DAP$W_CTLMENU(R9),10$    ;  specified
 01F7 C8    90       01A4   450          MOVB    FAL$B_RAC(R8),-         ; If not, use previous value saved in
   46 A9             01A8   451                  DAP$B_RAC(R9)           ;  FAL work area
   46 A9    90       01AA   452 10$:     MOVB    DAP$B_RAC(R9),-         ; Save currently specified value as
 01F7 C8             01AD   453                  FAL$B_RAC(R8)           ;  previous value for next-time-thru
                     01B0   454          $CASEB  SELECTOR=DAP$B_RAC(R9)- ; Dispatch on DAP record access mode:
                     01B0   455                  DISPL=<-                ;
                     01B0   456                      20$-                ;  Sequential record access
                     01B0   457                      20$-                ;  Random access by key value
                     01B0   458                      20$-                ;  Random access by RFA
                     01B0   459                      30$-                ;  Sequential file transfer
                     01B0   460                      40$-                ;  Block I/O access by VBN
                     01B0   461                      50$-                ;  Block I/O sequential file transfer
```

```
                     01B0   462                >                       ;
                     01C1   463 ;
                     01C1   464 ; Update the RAC field of the RAB unless block I/O mode is specified.
                     01C1   465 ; (RMS-32 ignores the RAC field on block I/O operations.)
                     01C1   466 ;
                     01C1   467 ; Also update the file transfer mode and block I/O flags as appropriate
                     01C1   468 ; for the access mode invoked.
                     01C1   469 ;
                     01C1   470
                     01C1   471          ASSUME  DAP$K_SEQ_ACC EQ RAB$C_SEQ
                     01C1   472          ASSUME  DAP$K_KEY_ACC EQ RAB$C_KEY
                     01C1   473          ASSUME  DAP$K_RFA_ACC EQ RAB$C_RFA
                     01C1   474
          46 A9   90 01C1   475 20$:     MOVB    DAP$B_RAC(R9),-         ; Store record access mode in RAB
          1E AB      01C4   476                 RAB$B_RAC(R11)          ;  (either SEQ, KEY, or RFA)
                     01C6   477          $CLRBIT #FAL$V_FTM,(R8)         ; Say record transfer mode
             08   11 01CA   478          BRB     35$                     ;
1E AB        00   90 01CC   479 30$:     MOVB    #RAB$C_SEQ,RAB$B_RAC(R11);Set record access mode to SEQ in RAB
                     01D0   480          $SETBIT #FAL$V_FTM,(R8)         ; Say file transfer mode
                     01D4   481 35$:     $CLRBIT #FAL$V_BLK_IO,(R8)      ; Say record I/O mode
             0E   11 01D8   482          BRB     ROP_FIELD               ;
                     01DA   483 40$:     $CLRBIT #FAL$V_FTM,(R8)         ; Say record transfer mode
             04   11 01DE   484          BRB     55$                     ;
                     01E0   485 50$:     $SETBIT #FAL$V_FTM,(R8)         ; Say file transfer mode
                     01E4   486 55$:     $SETBIT #FAL$V_BLK_IO,(R8)      ; Say block I/O mode
                     01E8   487
                     01E8   488 ;+
                     01E8   489 ; Process the DAP ROP field.
                     01E8   490 ;
                     01E8   491 ; Note: If the ROP field is not present in the Control message, then the default
                     01E8   492 ;       action is is to use the previous value.
                     01E8   493 ;-
                     01E8   494
                     01E8   495 ROP_FIELD:                              ; Process record options field
          03   E1 01E8   496          BBC     #DAP$V_ROP,-            ; Branch if ROP field was not explicitly
       44 A9      01EA   497                  DAP$W_CTLMENU(R9),-      ;  specified making previous ROP value
          07      01EC   498                  KRF_FIELD               ;  the current value
    51 50 A9   D0 01ED   499          MOVL    DAP$L_ROP(R9),R1        ; Get DAP ROP bits
        02F7   30 01F1   500          BSBW    MAP_ROP_FIELD           ; Update ROP options in RAB
                     01F4   501
                     01F4   502 ;+
                     01F4   503 ; Process the DAP KRF field.
                     01F4   504 ; This field is applicable only for indexed files.
                     01F4   505 ;
                     01F4   506 ; Note: If the KRF field is not present in the Control message, then the default
                     01F4   507 ;       action is to use the previous value.
                     01F4   508 ;-
                     01F4   509
                     01F4   510 KRF_FIELD:                              ; Process key of reference field
          02   E1 01F4   511          BBC     #DAP$V_KRF,-            ; Branch if KRF field was not explicitly
       44 A9      01F6   512                  DAP$W_CTLMENU(R9),-      ;  specified making previous KRF value
          05      01F8   513                  KEY_FIELD               ;  the current value
       47 A9   90 01F9   514          MOVB    DAP$B_KRF(R9),-         ; Update key of reference value in RAB
       35 AB      01FC   515                  RAB$B_KRF(R11)          ;  (meaningful only for indexed files)
                     01FE   516
                     01FE   517 ;+
                     01FE   518 ; Process the DAP KEY field.
```

```
                     01FE    519 ; Its format and content are context dependent:
                     01FE    520 ;   (1) for block I/O access, it contains the virtual block number for
                     01FE    521 ;       $READ/$WRITE, or the number of blocks for $SPACE.
                     01FE    522 ;  (2a) for sequential record access without the key limit option in force,
                     01FE    523 ;       this field is ignored because RMS will use its internally stored
                     01FE    524 ;       next-record-pointer to locate the record.
                     01FE    525 ;  (2b) for sequential record access of an indexed file with the key limit
                     01FE    526 ;       option set (i.e., ROP = LIM), it contains the key value string.
                     01FE    527 ;  (3a) for random access by key value for relative (or fixed length
                     01FE    528 ;       sequential) files, it contains the relative record number.
                     01FE    529 ;  (3b) for random access by key value for indexed files, it contains the
                     01FE    530 ;       key value string.
                     01FE    531 ;   (4) for random access by record file address, it contains the RFA value.
                     01FE    532 ;-
                     01FE    533
                     01FE    534 KEY_FIELD:                                  ; Process the key field
50   48 A9    7D     01FE    535         MOVQ    DAP$Q_KEY(R9),R0            ; <R0,R1> => descriptor of key field
47 68   09    E0     0202    536         BBS     #FAL$V_BLK_IO,(R8),50$      ; Branch if block I/O access
                     0206    537
                     0206    538         ASSUME  RAB$C_SEQ EQ 0
                     0206    539         ASSUME  RAB$C_KEY EQ 1
                     0206    540         ASSUME  RAB$C_RFA EQ 2
                     0206    541
                     0206    542         $CASEB  SELECTOR=RAB$B_RAC(R11)-; Dispatch on RMS record access mode:
                     0206    543                 BASE=#RAB$C_SEQ-        ;
                     0206    544                 DISPL=<-                ;
                     0206    545                     10$-                ;  Sequential record access
                     0206    546                     20$-                ;  Random access by key value
                     0206    547                     40$-                ;  Random access by RFA
                     0206    548                 >                       ;
      01    E0     0211    549 10$:    BBS     #DAP$V_KEY,-                ; Update key value only if KEY field
  1C 44 A9         0213    550                 DAP$W_CTLMENU(R9),30$       ;  was explicitly specified
        4D   11    0216    551                 BRB     90$                 ; All done with key field
20    1D AA    91   0218    552 20$:    CMPB    FAB$B_ORG(R10),#FAB$C_IDX;Branch if indexed file
        14   13    021C    553         BEQL    30$                         ; Fall thru if sequential or relative
                     021E    554
                     021E    555 ;
                     021E    556 ; Key field contains a relative record number (RRN).
                     021E    557 ; RMS requires that the RRN be a 4-byte unsigned integer value.
                     021E    558 ;
                     021E    559
   34 AB    04    90   021E    560         MOVB    #4,RAB$B_KSZ(R11)           ; Store size and address of buffer
      01FC C8    DE   0222    561         MOVAL   FAL$L_NUMBER(R8),-          ;  that will hold RRN value
         30 AB         0226    562                 RAB$L_KBF(R11)              ;  in KSZ/KBF fields of RAB
04   00   61    50    2C   0228    563         MOVC5   R0,(R1),#0,#4,-         ; Copy RRN value as a longword to
      01FC C8         022D    564                 FAL$L_NUMBER(R8)            ;  buffer in FAL work area
        22   11    0230    565         BRB     60$                         ; Join common code
                     0232    566
                     0232    567 ;
                     0232    568 ; Key field contains a key string.
                     0232    569 ;
                     0232    570
   34 AB    50    90   0232    571 30$:    MOVB    R0,RAB$B_KSZ(R11)           ; Store size and address of buffer
      0700 C8    DE   0236    572         MOVAL   FAL$T_KEYBUF(R8),-          ;  that will hold key string value
         30 AB         023A    573                 RAB$L_KBF(R11)              ;  ir KSZ/KBF fields of RAB
0700 C8   61    50    28   023C    574         MOVC3   R0,(R1),FAL$T_KEYBUF(R8); Copy string to buffer in FAL work area
        21   11    0242    575         BRB     90$                         ; All done with key field
```

```
                     0244    576
                     0244    577 ;
                     0244    578 ; Key field contains a record file address (RFA).
                     0244    579 ; RMS requires that the RFA be a 6-byte unsigned integer value.
                     0244    580 ;
                     0244    581
06   00   61   50    2C   0244    582 40$:    MOVC5   R0,(R1),#0,#6,-         ; Store RFA value directly in RFA field
                10 AB        0249    583             RAB$W_RFA(R11)          ;  of RAB (zero extended to 6-bytes)
                   07   11   024B    584             BRB     60$             ; Join common code
                     024D    585
                     024D    586 ;
                     024D    587 ; Key field contains virtual block number (VBN).
                     024D    588 ; RMS requires that the VBN be a 4-byte unsigned integer value.
                     024D    589 ;
                     024D    590
04   00   61   50    2C   024D    591 50$:    MOVC5   R0,(R1),#0,#4,-         ; Store VBN value directly in BKT field
                38 AB        0252    592             RAB$L_BKT(R11)          ;  of RAB (zero extended to longword)
                     0254    593
                     0254    594 ;
                     0254    595 ; Common code to verify that the length of the string in the DAP KEY field
                     0254    596 ; does not exceed the size of the buffer used to store the string.
                     0254    597 ;
                     0254    598
                   0F   1B   0254    599 60$:    BLEQU   90$                     ; Done if all SRC bytes are copied
                     0256    600                                             ;  (i.e., SRC size LEQU DST size)
                   81   95   0256    601 70$:    TSTB    (R1)+                   ; Error if any unmoved bytes are
                   05   12   0258    602             BNEQ    80$             ;  non-zero
                F9 50   F5   025A    603             SOBGTR  R0,70$          ; Continue until all extra bytes
                   06   11   025D    604             BRB     90$             ;  are checked
                 FD9E'  30   025F    605 80$:    BSBW    FAL$UNS_KEY             ; Return error in Status message
                 0324   31   0262    606             BRW     EXIT_FAILURE            ; Exit state with failure
                     0265    607
                     0265    608 ;
                     0265    609 ; Use the CTLFUNC field value as the next state table value.
                     0265    610 ;
                     0265    611
40 A9   90           0265    612 90$:    MOVB    DAP$B_CTLFUNC(R9),-     ; Store new state transition value
10 A8                0268    613             FAL$B_VALUE(R8)         ;
 031F   31           026A    614             BRW     EXIT_SUCCESS            ; Exit state with success
```

```
                     026D   616          .SBTTL  FAL$DECODE_CON
                     026D   617
                     026D   618 ;++
                     026D   619 ; Process the Continue Transfer message which has been received and validated.
                     026D   620 ;--
                     026D   621
                     026D   622 FAL$DECODE_CON::                         ; Entry point
                     026D   623
                     026D   624 ;
                     026D   625 ; Use the CONFUNC field value as the next state transition table value.
                     026D   626 ;
                     026D   627
40 A9    90          026D   628          MOVB    DAP$B_CONFUNC(R9),-     ; Store new state transition value
10 A8                0270   629                  FAL$B_VALUE(R8)         ;
 0317    31          0272   630          BRW     EXIT_SUCCESS            ; Exit state with success
```

```
                     0275   632          .SBTTL  FAL$DECODE_CMP
                     0275   633
                     0275   634 ;++
                     0275   635 ; Process the Access Complete message which has been received and validated.
                     0275   636 ; Update the FAB if necessary.
                     0275   637 ;--
                     0275   638
                     0275   639 FAL$DECODE_CMP::                          ; Entry point
                     0275   640
                     0275   641 ;
                     0275   642 ; Process the DAP FOP field.
                     0275   643 ; Do not update the FOP field in the FAB if this is a DAP DISCONNECT function
                     0275   644 ; or if no FOP field was included in the Access Complete message.
                     0275   645 ;
                     0275   646
          51  44 A9  D0 0275   647          MOVL    DAP$L_FOP2(R9),R1       ; Get DAP FOP bits
              03      13 0279   648          BEQL    10$                     ; Branch if no bits to map
            01C6      30 027B   649          BSBW    MAP_FOP_FIELD           ; Update FOP in FAB
                     027E   650
                     027E   651 ;
                     027E   652 ; Use the CMPFUNC field value as the next state table value.
                     027E   653 ;
                     027E   654
          40 A9     90 027E   655 10$:     MOVB    DAP$B_CMPFUNC(R9),-     ; Store new state transition value
          10 A8             0281   656                  FAL$B_VALUE(R8)         ;
            0306    31 0283   657          BRW     EXIT_SUCCESS            ; Exit state with success
```

```
                     0286   659          .SBTTL  FAL$DECODE_KEY
                     0286   660
                     0286   661 ;++
                     0286   662 ; Process the Key Definition message which has been received and validated.
                     0286   663 ; Update the KEYXAB (by key of reference) with information from this message.
                     0286   664 ;--
                     0286   665
                     0286   666 FAL$DECODE_KEY::                          ; Entry point
                     0286   667
                     0286   668 ;
                     0286   669 ; Initialize the appropriate Key Definition XAB (in the FAL work area) and
                     0286   670 ; process the DAP REF field.
                     0286   671 ;
                     0286   672
          56   6C A9   9A  0286   673          MOVZBL  DAP$B_REF(R9),R6        ; Get key of reference value
                FD73'  30  028A   674          BSBW    FAL$INIT_KEYXAB         ; On return R7 = address of XAB
               03 50   E8  028D   675          BLBS    R0,10$                  ; Branch on success
                02F6   31  0290   676          BRW     EXIT_FAILURE            ; Exit state with failure
                       0293   677                                          ; (i.e., ignore this DAP message)
                       0293   678
                       0293   679 ;
                       0293   680 ; Process the DAP KNM field.
                       0293   681 ;
                       0293   682
                       0293   683          ASSUME  FAL$K_KEYNAM EQ 32
                       0293   684
            64 A9      D5  0293   685 10$:     TSTL    DAP$Q_KNM(R9)           ; Branch if no key name string was
                 0B    13  0296   686          BEQL    20$                     ;  specified
            64 A9      2C  0298   687          MOVC5   DAP$Q_KNM(R9),-         ; Copy DAP key name string
            68 B9          029B   688                  @DAP$Q_KNM+4(R9),-      ;  to 32 byte key name buffer
      38 B7   20     00      029D   689                  #0,#32,@XAB$L_KNM(R7)   ;  with zero fill
                 03    11  02A1   690          BRB     30$                     ;
            38 A7      D4  02A3   691 20$:     CLRL    XAB$L_KNM(R7)           ; Zero key name buffer address
                       02A6   692
                       02A6   693 ;
                       02A6   694 ; Process the DAP FLG field.
                       02A6   695 ;
                       02A6   696
       51     48 A9   9A  02A6   697 30$:     MOVZBL  DAP$B_FLG(R9),R1        ; Get DAP FLG bits
                 52    D4  02AA   698          CLRL    R2                      ; Clear RMS FLG bits
                       02AC   699          $MAPBIT DAP$V_DUP,XAB$V_DUP     ; Map DUP bit
                       02B4   700          $MAPBIT DAP$V_CHG,XAB$V_CHG     ; Map CHG bit
                       02BC   701          $MAPBIT DAP$V_NUL_CHR,XAB$V_NUL ; Map NUL bit
       12 A7     52    90  02C4   702          MOVB    R2,XAB$B_FLG(R7)        ; Update FLG field in XAB
                       02C8   703
                       02C8   704 ;
                       02C8   705 ; Process the DAP DFL, IFL, NUL, IAN, LAN, DAN, and DTP fields.
                       02C8   706 ;
                       02C8   707
1C A7      44 A9      90  02C8   708          MOVB    DAP$W_DFL(R9),XAB$W_DFL(R7)
1A A7      46 A9      90  02CD   709          MOVB    DAP$W_IFL(R9),XAB$W_IFL(R7)
15 A7      6D A9      90  02D2   710          MOVB    DAP$B_NUL(R9),XAB$B_NUL(R7)
08 A7      6E A9      90  02D7   711          MOVB    DAP$B_IAN(R9),XAB$B_IAN(R7)
09 A7      6F A9      90  02DC   712          MOVB    DAP$B_LAN(R9),XAB$B_LAN(R7)
0A A7      70 A9      90  02E1   713          MOVB    DAP$B_DAN(R9),XAB$B_DAN(R7)
13 A7      71 A9      90  02E6   714          MOVB    DAP$B_DTP(R9),XAB$B_DTP(R7)
                       02EB   715
```

```
                     02EB   716 ;
                     02EB   717 ; Process the DAP NSG, POS, and SIZ fields.
                     02EB   718 ;
                     02EB   719 ; Note: FAL$DECODE_MSG guarantees that 0 < DAP$B_NSG < 9.
                     02EB   720 ;
                     02EB   721
   56    49 A9   9A  02EB   722         MOVZBL  DAP$B_NSG(R9),R6        ; Get # key segments
   14 A7    56   90  02EF   723         MOVB    R6,XAB$B_NSG(R7)        ; Update NSG field in XAB
   5C A9    56   28  02F3   724         MOVC3   R6,DAP$B_SIZ(R9),-      ; Copy 1 to 8 key size values
         2E A7        02F7   725                 XAB$B_SIZ(R7)           ;  to XAB
56    56    01   78  02F9   726         ASHL    #1,R6,R6                ; Double byte count
   4C A9    56   28  02FD   727         MOVC3   R6,DAP$W_POS(R9),-      ; Copy 1 to 8 key position values
         1E A7        0301   728                 XAB$W_POS(R7)           ;  to XAB
                     0303   729
                     0303   730 ;
                     0303   731 ; Ignore the DAP RVB, DVB, DBS, IBS, LVL, TKS, and MRL fields as these are not
                     0303   732 ; inputs to RMS.
                     0303   733 ;
                     0303   734 ; Finish paper work and exit.
                     0303   735 ;
                     0303   736
                     0303   737         $SETBIT #FAL$V_KEYXAB,FAL$W_RECEIVED(R8)
                     0308   738                                         ; Denote XAB to add to XAB chain
         0281    31  0308   739         BRW     EXIT_SUCCESS            ; Exit state with success
```

```
                     030B    741          .SBTTL  FAL$DECODE_ALL
                     030B    742
                     030B    743 ;++
                     030B    744 ; Process the Allocation message which has been received and validated.
                     030B    745 ; Update the ALLXAB (by AID) with information from this message.
                     030B    746 ;--
                     030B    747
                     030B    748 FAL$DECODE_ALL::                            ; Entry point
                     030B    749
                     030B    750 ;
                     030B    751 ; Initialize the appropriate Allocation XAB (in the FAL work area) and
                     030B    752 ; process the DAP AID field.
                     030B    753 ;
                     030B    754
          56  50 A9  9A 030B    755          MOVZBL  DAP$B_AID(R9),R6           ; Get area ID value
            FCEE'  30 030F    756          BSBW    FAL$INIT_ALLXAB            ; On return R7 = address of XAB
         03 50  E8 0312    757          BLBS    R0,10$                     ; Branch on success
          0271  31 0315    758          BRW     EXIT_FAILURE               ; Exit state with failure
                     0318    759                                             ; (i.e., ignore this DAP message)
                     0318    760
                     0318    761 ;
                     0318    762 ; Process the DAP ALN field.
                     0318    763 ;
                     0318    764
                     0318    765          ASSUME  DAP$K_ANY EQ 0
                     0318    766          ASSUME  DAP$K_CYL EQ XAB$C_CYL
                     0318    767          ASSUME  DAP$K_LBN EQ XAB$C_LBN
                     0318    768          ASSUME  DAP$K_VBN EQ XAB$C_VBN
                     0318    769
  09 A7  44 A9  90   0318    770 10$:     MOVB    DAP$B_ALN(R9),XAB$B_ALN(R7)
                     031D    771
                     031D    772 ;
                     031D    773 ; Process the DAP AOP field.
                     031D    774 ;
                     031D    775
     51  45 A9  9A   031D    776          MOVZBL  DAP$B_AOP(R9),R1           ; Get DAP AOP bits
            52  D4   0321    777          CLRL    R2                         ; Clear RMS AOP bits
                     0323    778          $MAPBIT DAP$V_HRD,XAB$V_HRD        ; Map HRD bit
                     032B    779          $MAPBIT DAP$V_CBT2,XAB$V_CBT       ; Map CBT bit
                     0333    780          $MAPBIT DAP$V_CTG2,XAB$V_CTG       ; Map CTG bit
                     033B    781          $MAPBIT DAP$V_ONC,XAB$V_ONC        ; Map ONC bit
  08 A7  52  90      0343    782          MOVB    R2,XAB$B_AOP(R7)           ; Update AOP field in XAB
                     0347    783
                     0347    784 ;
                     0347    785 ; Process the DAP VOL, LOC, ALQ, BKZ, and DEQ fields.
                     0347    786 ;
                     0347    787
  0A A7  42 A9  B0   0347    788          MOVW    DAP$W_VOL(R9),XAB$W_VOL(R7)
  0C A7  48 A9  D0   034C    789          MOVL    DAP$L_LOC(R9),XAB$L_LOC(R7)
  10 A7  4C A9  D0   0351    790          MOVL    DAP$L_ALQ2(R9),XAB$L_ALQ(R7)
  16 A7  51 A9  90   0356    791          MOVB    DAP$B_BKZ(R9),XAB$B_BKZ(R7)
  14 A7  52 A9  B0   035B    792          MOVW    DAP$W_DEQ2(R9),XAB$W_DEQ(R7)
                     0360    793
                     0360    794 ;
                     0360    795 ; Finish paper work and exit.
                     0360    796 ;
                     0360    797
```

```
                     0360   798          $SETBIT #FAL$V_ALLXAB,FAL$W_RECEIVED(R8)
                     0365   799                                  ; Denote XAB to add to XAB chain
0224      31         0365   800          BRW     EXIT_SUCCESS    ; Exit state with success
```

```
                     0368    802          .SBTTL  FAL$DECODE_TIM
                     0368    803
                     0368    804 ;++
                     0368    805 ; Process the Date and Time message which has been received and validated.
                     0368    806 ; Initialize both the DATXAB and RDTXAB and update them with information from
                     0368    807 ; this message. Other action routines will determine which of the two XABs to
                     0368    808 ; to use (or both) depending on the function that will be performed.
                     0368    809 ;--
                     0368    810
                     0368    811 FAL$DECODE_TIM::                        ; Entry point
                     0368    812
                     0368    813 ;
                     0368    814 ; Initialize and fill-in the Date and Time XAB.
                     0368    815 ;
                     0368    816
 FC95'  30           0368    817          BSBW    FAL$INIT_DATXAB        ; On return R7 = address of XAB
48 A9   7D           036B    818          MOVQ    DAP$Q_CDT(R9),-        ; Copy creation date and time
14 A7                036E    819                  XAB$Q_CDT(R7)          ;  binary value to XAB
50 A9   7D           0370    820          MOVQ    DAP$Q_RDT(R9),-        ; Copy revision date and time
0C A7                0373    821                  XAB$Q_RDT(R7)          ;  binary value to XAB
58 A9   7D           0375    822          MOVQ    DAP$Q_EDT(R9),-        ; Copy expiration date and time
1C A7                0378    823                  XAB$Q_EDT(R7)          ;  binary value to XAB
60 A9   7D           037A    824          MOVQ    DAP$Q_BDT(R9),-        ; Copy backup date and time
24 A7                037D    825                  XAB$Q_BDT(R7)          ;  binary value to XAB
42 A9   B0           037F    826          MOVW    DAP$W_RVN(R9),-        ; Store revision number value in XAB
08 A7                0382    827                  XAB$W_RVN(R7)          ;
                     0384    828
                     0384    829 ;
                     0384    830 ; Initialize and fill-in the Revision Date and Time XAB.
                     0384    831 ;
                     0384    832
 FC79'  30           0384    833          BSBW    FAL$INIT_RDTXAB        ; On return R7 = address of XAB
50 A9   7D           0387    834          MOVQ    DAP$Q_RDT(R9),-        ; Copy revision date and time
0C A7                038A    835                  XAB$Q_RDT(R7)          ;  binary value to XAB
42 A9   B0           038C    836          MOVW    DAP$W_RVN(R9),-        ; Store revision number value in XAB
08 A7                038F    837                  XAB$W_RVN(R7)          ;
                     0391    838
                     0391    839 ;
                     0391    840 ; Finish paper work and exit.
                     0391    841 ;
                     0391    842
72 A8    14  A8      0391    843          BISW2   #<<FAL$M_DATXAB>!-      ; Denote XABs to add to XAB chain
                     0395    844                    <FAL$M_RDTXAB>!-     ;
                     0395    845                  0>,FAL$W_RECEIVED(R8)  ;
    01F4     31      0395    846          BRW     EXIT_SUCCESS           ; Exit state with success
```

```
                        0398   848          .SBTTL  FAL$DECODE_PRO
                        0398   849
                        0398   850 ;++
                        0398   851 ; Process the Protection message which has been received and validated.
                        0398   852 ; Update the PROXAB with information from this message.
                        0398   853 ;--
                        0398   854
                        0398   855 FAL$DECODE_PRO::                          ; Entry point
                FC65'    30     0398   856          BSBW    FAL$INIT_PROXAB     ; On return R7 = address of XAB
              0C A7      D4     039B   857          CLRL    XAB$L_UIC(R7)       ; Initialize UIC and protection mask
08 A7      FFFF 8F       B0     039E   858          MOVW    #-1,XAB$W_PRO(R7)   ;  fields to [0,0] and -1. These mean
                        03A4   859                                             ;  use process UIC and default process
                        03A4   860                                             ;  protection in effect, respectively
                        03A4   861
                        03A4   862 ;
                        03A4   863 ; Process the DAP OWNER field.
                        03A4   864 ;
                        03A4   865
      54    48 A9       7D     03A4   866          MOVQ    DAP$Q_OWNER(R9),R4  ; Get descriptor of ASCII string
   5B 8F      65       91     03A8   867          CMPB    (R5),#^A\[\         ; Branch if string does not begin
                 4C     12     03AC   868          BNEQ    30$                 ;  with bracket
5D 8F      FF A544      91     03AE   869          CMPB    -1(R5)[R4],#^A\]\   ; Branch if string does not end
                 44     12     03B4   870          BNEQ    30$                 ;  with bracket
           54    02     C2     03B6   871          SUBL2   #2,R4               ; Discard brackets
                 55     D6     03B9   872          INCL    R5                  ;
      65   54    2C     3A     03BB   873          LOCC    #^A\,\,R4,(R5)      ; Locate group-member delimiter
                 39     13     03BF   874          BEQL    30$                 ; Branch on failure
      54   51    55     C3     03C1   875          SUBL3   R5,R1,R4            ; <R4,R5> => group string
                 50     D7     03C5   876          DECL    R0                  ; <R0,R1> => member string
                 51     D6     03C7   877          INCL    R1                  ;
                 7E     D4     03C9   878          CLRL    -(SP)               ; Allocate space from stack
                 5E     DD     03CB   879          PUSHL   SP                  ; Address of result
                 51     DD     03CD   880          PUSHL   R1                  ; Address of input string
                 50     DD     03CF   881          PUSHL   R0                  ; Size of input string
00000000'GF      03     FB     03D1   882          CALLS   #3,G^LIB$CVT_OTB    ; Convert octal string to binary
         1D      50     E9     03D8   883          BLBC    R0,20$              ; Branch on failure
      0C A7      6E     B0     03DB   884          MOVW    (SP),XAB$W_MBM(R7)  ; Update member UIC value in XAB
                 5E     DD     03DF   885          PUSHL   SP                  ; Address of result
                 55     DD     03E1   886          PUSHL   R5                  ; Address of input string
                 54     DD     03E3   887          PUSHL   R4                  ; Size of input string
00000000'GF      03     FB     03E5   888          CALLS   #3,G^LIB$CVT_OTB    ; Convert octal string to binary
         06      50     E9     03EC   889          BLBC    R0,10$              ; Branch on failure
      0E A7      6E     B0     03EF   890          MOVW    (SP),XAB$W_GRP(R7)  ; Update group UIC value in XAB
                 03     11     03F3   891          BRB     20$                 ; UIC has been successfully converted
           0C    A7     B4     03F5   892 10$:     CLRW    XAB$W_MBM(R7)       ; GRP is invalid, so also discard MBM
                 8E     D4     03F8   893 20$:     CLRL    (SP)+               ; Deallocate space from stack
                        03FA   894
                        03FA   895 ;
                        03FA   896 ; Process the DAP PROSYS, PROOWN, PROGRP, PROWLD fields.
                        03FA   897 ;
                        03FA   898
      40 A9      1E     B3     03FA   899 30$:     BITW    #<<DAP$M_PROSYS>!-  ; Use default file protection in effect
                        03FE   900                   <DAP$M_PROOWN>!-      ;  for the user process if all four
                        03FE   901                   <DAP$M_PROGRP>!-      ;  protection fields of the DAP
                        03FE   902                   <DAP$M_PROWLD>!-      ;  Protection message were defaulted
                        03FE   903                 0>,DAP$W_PROMENU(R9)    ;  (i.e., omitted from message)
                 1C     13     03FE   904          BEQL    40$                 ; Branch if no fields explicitly sent
```

```
                50   04   00    50 A9   F0  0400    905          INSV    DAP$W_PROSYS(R9),#0,#4,R0 ; Map system bits
                50   04   04    52 A9   F0  0406    906          INSV    DAP$W_PROOWN(R9),#4,#4,R0 ; Map owner bits
                50   04   08    54 A9   F0  040C    907          INSV    DAP$W_PROGRP(R9),#8,#4,R0 ; Map group bits
                50   04   0C    56 A9   F0  0412    908          INSV    DAP$W_PROWLD(R9),#12,#4,R0; Map world bits
                          08 A7    50   B0  0418    909          MOVW    R0,XAB$W_PRO(R7)          ; Update protection mask in XAB
                                                041C    910
                                                041C    911 ;
                                                041C    912 ; Finish paper work and exit.
                                                041C    913 ;
                                                041C    914
                                                041C    915 40$:    $SETBIT #FAL$V_PROXAB,FAL$W_RECEIVED(R8)
                                                0421    916                                    ; Denote XAB to add to XAB chain
                                     0168   31  0421    917          BRW     EXIT_SUCCESS       ; Exit state with success
```

```
                     0424   919          .SBTTL  FAL$DECODE_NAM
                     0424   920
                     0424   921 ;++
                     0424   922 ; Process the name message which has been received and validated.
                     0424   923 ; Update FAB2 with information from this message.
                     0424   924 ;
                     0424   925 ; NOTE: At this time, only a rename operation will cause a Name message to be
                     0424   926 ; returned by FAL.
                     0424   927 ;--
                     0424   928
                     0424   929 FAL$DECODE_NAM::                              ; Entry point
5A   0800 C8   DE    0424   930          MOVAL   FAL$L_FAB2(R8),R10          ; Put new filename FAB (FAB2) in R10
       44 A9   90    0429   931          MOVB    DAP$Q_NAMESPEC(R9),-        ; Store size of new filespec string
       34 AA         042C   932                  FAB$B_FNS(R10)              ;  in FAB2
       44 A9   28    042E   933          MOVC3   DAP$Q_NAMESPEC(R9),-        ; Copy the filespec string to buffer
       48 B9         0431   934                  @DAP$Q_NAMESPEC+4(R9),-     ;
       2C BA         0433   935                  @FAB$L_FNA(R10)             ;
  52   44 A9   7E    0435   936          MOVAQ   DAP$Q_NAMESPEC(R9),R2       ; Get address of filename descriptor
         FBC4'  30    0439   937          BSBW    FAL$LOG_REQNAM2             ; Log requested new name in print file
5A   0200 C8   DE    043C   938          MOVAL   FAL$L_FAB(R8),R10           ; Restore old filename FAB in R10
         0148  31    0441   939          BRW     EXIT_SUCCESS                ; Exit state with success
```

```
                     0444   941          .SBTTL  SUPPORT ROUTINES
                     0444   942
                     0444   943
                     0444   944          .SBTTL  MAP_FOP_FIELD
                     0444   945
                     0444   946 ;++
                     0444   947 ; This routine maps DAP FOP bits into RMS FOP bits and stores the result in
                     0444   948 ; the FOP field of the FAB.
                     0444   949 ;
                     0444   950 ; R1 contains the DAP bitmask on input.
                     0444   951 ; R2 is destroyed on output.
                     0444   952 ;--
                     0444   953
                     0444   954 MAP_FOP_FIELD:                              ; Entry point
          52    D4   0444   955          CLRL    R2                         ; Clear RMS FOP bits
          51    D5   0446   956          TSTL    R1                         ; Examine FOP bitmask
          03    12   0448   957          BNEQ    10$                        ; Begin mapping if any bits are set
        0090    31   044A   958          BRW     20$                        ; Branch if there are no bits to map
                     044D   959 10$:     $MAPBIT DAP$V_RWO,FAB$V_RWO        ; Map RWO bit
                     0455   960          $MAPBIT DAP$V_RWC,FAB$V_RWC        ; Map RWC bit
                     045D   961          $MAPBIT DAP$V_POS,FAB$V_POS        ; Map POS bit
                     0465   962          $MAPBIT DAP$V_CTG,FAB$V_CTG        ; Map CTG bit
                     046D   963          $MAPBIT DAP$V_SUP,FAB$V_SUP        ; Map SUP bit
                     0475   964          $MAPBIT DAP$V_NEF,FAB$V_NEF        ; Map NEF bit
                     047D   965          $MAPBIT DAP$V_TMP,FAB$V_TMP        ; Map TMP bit
                     0485   966          $MAPBIT DAP$V_TMD,FAB$V_TMD        ; Map TMD bit
                     048D   967          $MAPBIT DAP$V_DMO,FAB$V_DMO        ; Map DMO bit
                     0495   968          $MAPBIT DAP$V_WCK,FAB$V_WCK        ; Map WCK bit
                     049D   969          $MAPBIT DAP$V_RCK,FAB$V_RCK        ; Map RCK bit
                     04A5   970 ; ***** $MAPBIT DAP$V_CIF,FAB$V_CIF        ; Map CIF bit
                     04A5   971          $MAPBIT DAP$V_SQO,FAB$V_SQO        ; Map SQO bit
                     04AD   972          $MAPBIT DAP$V_MXV,FAB$V_MXV        ; Map MXV bit
                     04B5   973          $MAPBIT DAP$V_SPL,FAB$V_SPL        ; Map SPL bit
                     04BD   974          $MAPBIT DAP$V_SCF,FAB$V_SCF        ; Map SCF bit
                     04C5   975          $MAPBIT DAP$V_DLT,FAB$V_DLT        ; Map DLT bit
                     04CD   976          $MAPBIT DAP$V_CBT,FAB$V_CBT        ; Map CBT bit
                     04D5   977 ; ***** $MAPBIT DAP$V_DFW,FAB$V_DFW        ; Map DFW bit
                     04D5   978          $MAPBIT DAP$V_TEF,FAB$V_TEF        ; Map TEF bit
                     04DD   979 ;        $MAPBIT DAP$V_OFP,FAB$V_OFP        ; Map OFP bit
                     04DD   980                                             ; Note: this bit has no meaning here
                     04DD   981                                             ;  because only primary filespec
                     04DD   982                                             ;  is being given to RMS by FAL
  04 04 AA    18   E1   04DD   983 20$:     BBC     #FAB$V_NAM,FAB$L_FOP(R10),30$
                     04E2   984          $SETBIT #FAB$V_NAM,R2              ; Preserve state of NAM bit in FOP
     04 AA    52   D0   04E6   985 30$:     MOVL    R2,FAB$L_FOP(R10)          ; Update FOP field in FAB
                  05   04EA   986          RSB                                ; Exit
```

```
                     04EB   988          .SBTTL  MAP_ROP_FIELD
                     04EB   989
                     04EB   990 ;+
                     04EB   991 ; This routine maps DAP ROP bits into RMS ROP bits and stores the result in
                     04EB   992 ; the ROP field of the RAB.
                     04EB   993 ;
                     04EB   994 ; R1 contains the DAP bitmask on input.
                     04EB   995 ; R2 is destroyed on output.
                     04EB   996 ;-
                     04EB   997
                     04EB   998 MAP_ROP_FIELD:                          ; Entry point
       52    D4      04EB   999          CLRL    R2                     ; Clear RMS ROP bits
       51    D5      04ED  1000          TSTL    R1                     ; Examine ROP bitmask
       03    12      04EF  1001          BNEQ    10$                    ; Begin mapping if any bits are set
     0090    31      04F1  1002          BRW     20$                    ; Branch if there are no bits to map
                     04F4  1003 10$:     $MAPBIT DAP$V_EOF,RAB$V_EOF    ; Map EOF bit
                     04FC  1004          $MAPBIT DAP$V_FDL,RAB$V_FDL    ; Map FDL bit
                     0504  1005          $MAPBIT DAP$V_UIF,RAB$V_UIF    ; Map UIF bit
                     050C  1006          $MAPBIT DAP$V_LOA,RAB$V_LOA    ; Map LOA bit
                     0514  1007          $MAPBIT DAP$V_ULK,RAB$V_ULK    ; Map ULK bit
                     051C  1008          $MAPBIT DAP$V_TPT,RAB$V_TPT    ; Map TPT bit
                     0524  1009          $MAPBIT DAP$V_RAH,RAB$V_RAH    ; Map RAH bit
                     052C  1010          $MAPBIT DAP$V_WBH,RAB$V_WBH    ; Map WBH bit
                     0534  1011          $MAPBIT DAP$V_KGE,RAB$V_KGE    ; Map KGE bit
                     053C  1012          $MAPBIT DAP$V_KGT,RAB$V_KGT    ; Map KGT bit
                     0544  1013          $MAPBIT DAP$V_NLK,RAB$V_NLK    ; Map NLK bit
                     054C  1014          $MAPBIT DAP$V_RLK,RAB$V_RLK    ; Map RLK bit
                     0554  1015          $MAPBIT DAP$V_ROPBIO,RAB$V_BIO ; Map BIO bit
                     055C  1016          $MAPBIT DAP$V_LIM,RAB$V_LIM    ; Map LIM bit
                     0564  1017          $MAPBIT DAP$V_NXR,RAB$V_NXR    ; Map NXR bit
                     056C  1018          $MAPBIT DAP$V_ROPWAT,RAB$V_WAT ; Map WAT bit
                     0574  1019          $MAPBIT DAP$V_RRL,RAB$V_RRL    ; Map RRL bit
                     057C  1020          $MAPBIT DAP$V_REA,RAB$V_REA    ; Map REA bit
  04 AB      52    D0   0584  1021 20$:     MOVL    R2,RAB$L_ROP(R11)      ; Update ROP field in RAB
                05   0588  1022          RSB                            ; Exit
```

```
                     0589  1024          .SBTTL  STATE EXIT ROUTINES
                     0589  1025
                     0589  1026 ;++
                     0589  1027 ; Exit state with failure.
                     0589  1028 ;--
                     0589  1029
                     0589  1030 EXIT_FAILURE:                          ; Entry point
 50   D4             0589  1031          CLRL    R0                    ; Signal state transition failure
      05             058B  1032          RSB                           ; Exit to state table manager
                     058C  1033
                     058C  1034 ;++
                     058C  1035 ; Exit state with success.
                     058C  1036 ;--
                     058C  1037
                     058C  1038 EXIT_SUCCESS:                          ; Entry point
 50   01        D0  058C  1039          MOVL    #1,R0                 ; Signal state transition success
      05             058F  1040          RSB                           ; Exit to state table manager
                     0590  1041
                     0590  1042          .END                          ; End of module
```

```
$$COUNT                 =  00000003
DAP$B_ACCFUNC              00000040
DAP$B_ACCOPT               00000041
DAP$B_AID                  00000050
DAP$B_ALN                  00000044
DAP$B_AOP                  00000045
DAP$B_BITCNT               00000035
DAP$B_BKS                  00000050
DAP$B_BKZ                  00000051
DAP$B_BLKCNT               00000056
DAP$B_BSZ                  00000052
DAP$B_CMPFUNC              00000040
DAP$B_CONFUNC              00000040
DAP$B_CTLFUNC              00000040
DAP$B_DAN                  00000070
DAP$B_DATATYPE             00000044
DAP$B_DBS                  0000007C
DAP$B_DCODE_FID            00000019
DAP$B_DCODE_MAC            0000001B
DAP$B_DCODE_MSG            0000001A
DAP$B_DECVER               00000047
DAP$B_DTP                  00000071
DAP$B_ECONUM               00000045
DAP$B_FAC                  00000042
DAP$B_FILESYS              00000043
DAP$B_FLAGS                00000031
DAP$B_FLG                  00000048
DAP$B_FSZ                  00000051
DAP$B_IAN                  0000006E
DAP$B_IBS                  0000007D
DAP$B_KRF                  00000047
DAP$B_LAN                  0000006F
DAP$B_LEN256               00000034
DAP$B_LENGTH               00000033
DAP$B_LVL                  0000007E
DAP$B_NAMETYPE             00000040
DAP$B_NSG                  00000049
DAP$B_NUL                  0000006D
DAP$B_ORG                  00000045
DAP$B_OSTYPE               00000042
DAP$B_RAC                  00000046
DAP$B_RAT                  00000047
DAP$B_REF                  0000006C
DAP$B_RFM                  00000046
DAP$B_SHR                  00000043
DAP$B_SIZ                  0000005C
DAP$B_SIZ_TMP              0000004A
DAP$B_STREAMID             00000032
DAP$B_TKS                  0000007F
DAP$B_TYPE                 00000030
DAP$B_USRNUM               00000046
DAP$B_USRVER               00000048
DAP$B_VERNUM               00000044
DAP$B_X_FIELD              00000024
DAP$C_BLN                  000000C0
DAP$K_ANY               =  00000000
DAP$K_BLK_FILE          =  00000005
DAP$K_BLK_VBN           =  00000004
DAP$K_BLN                  000000C0
DAP$K_CNF_MSG           =  00000001
DAP$K_CYL               =  00000001
DAP$K_DECVER_V          =  00000004
DAP$K_ECONUM_V          =  00000000
DAP$K_FIX               =  00000001
DAP$K_IDX               =  00000020
DAP$K_KEY_ACC           =  00000001
DAP$K_LBN               =  00000002
DAP$K_LOAD              =  000000FF
DAP$K_REL               =  00000010
DAP$K_RFA_ACC           =  00000002
DAP$K_RMS32             =  00000003
DAP$K_SEQ               =  00000000
DAP$K_SEQ_ACC           =  00000000
DAP$K_SEQ_FILE          =  00000003
DAP$K_STG               =  00000000
DAP$K_STM               =  00000004
DAP$K_STMCR             =  00000006
DAP$K_STMLF             =  00000005
DAP$K_SYSCAP1_V         =  EFF67DF7
DAP$K_SYSCAP2_V         =  00001962
DAP$K_UDF               =  00000000
DAP$K_USRNUM_V          =  00000000
DAP$K_USRVER_V          =  00000000
DAP$K_VAR               =  00000002
DAP$K_VAXVMS            =  00000007
DAP$K_VBN               =  00000003
DAP$K_VERNUM_V          =  00000007
DAP$K_VFC               =  00000003
DAP$L_ALQ1                 0000004C
DAP$L_ALQ2                 0000004C
DAP$L_ATTMENU              00000040
DAP$L_CMWA                 00000030
DAP$L_CRC_RSLT             00000020
DAP$L_DCODE_STS            00000018
DAP$L_DEV                  00000068
DAP$L_DVB                  00000078
DAP$L_EBK                  00000078
DAP$L_FOP1                 00000064
DAP$L_FOP2                 00000044
DAP$L_HBK                  00000074
DAP$L_KEYMENU              00000040
DAP$L_LOC                  00000048
DAP$L_MRN                  00000058
DAP$L_MSG_MASK             0000001C
DAP$L_ROP                  00000050
DAP$L_RVB                  00000074
DAP$L_SBN                  0000007C
DAP$L_SSPWA                00000080
DAP$L_SSP_CAP              00000088
DAP$L_SSP_FLG              00000084
DAP$L_TEMP                 00000090
DAP$M_BITCNT            =  00000008
DAP$M_BLKCNT            =  00000040
DAP$M_CMPFMT            =  00000008
```

```
DAP$M_DFTSPEC           = 00000010
DAP$M_DMO               = 00002000
DAP$M_DSP_3NAM          = 00000200
DAP$M_EMBEDDED          = 00000010
DAP$M_GET               = 00000002
DAP$M_GO_NOGO           = 00000010
DAP$M_IMAGE             = 00000002
DAP$M_LOADIM            = 00000001
DAP$M_LSA               = 00000040
DAP$M_MACY11            = 00000080
DAP$M_MSE               = 00000010
DAP$M_PROGRP            = 00000008
DAP$M_PROOWN            = 00000004
DAP$M_PROSYS            = 00000002
DAP$M_PROWLD            = 00000010
DAP$M_SEGMENT           = 00000040
DAP$M_TMP1$             = 00000020
DAP$M_TMP2$             = 000000C0
DAP$M_TMP3$             = 00020000
DAP$M_TMP4$             = 01000000
DAP$M_TMP5$             = F0000000
DAP$M_ZERO              = 00000080
DAP$Q_ADT                 00000070
DAP$Q_BDT                 00000060
DAP$Q_CDT                 00000048
DAP$Q_DCODE_FLG           00000000
DAP$Q_EDT                 00000058
DAP$Q_FILESPEC            00000044
DAP$Q_KEY                 00000048
DAP$Q_KNM                 00000064
DAP$Q_MSG_BUF1            00000008
DAP$Q_MSG_BUF2            00000010
DAP$Q_NAMESPEC            00000044
DAP$Q_OWNER               00000048
DAP$Q_PASSWORD            00000050
DAP$Q_PDT                 00000068
DAP$Q_RDT                 00000050
DAP$Q_RUNSYS              0000005C
DAP$Q_SYSCAP              00000028
DAP$Q_SYSPEC              00000038
DAP$V_APP               = 00000007
DAP$V_BIGBLK            = 00000014
DAP$V_BIO               = 00000005
DAP$V_BLK               = 00000003
DAP$V_BRO               = 00000006
DAP$V_CBT               = 00000017
DAP$V_CBT2              = 00000002
DAP$V_CHG               = 00000001
DAP$V_CR                = 00000001
DAP$V_CTG               = 00000007
DAP$V_CTG2              = 00000001
DAP$V_DAPCRC            = 00000015
DAP$V_DEL               = 00000002
DAP$V_DLT               = 00000016
DAP$V_DMO               = 0000000D
DAP$V_DUP               = 00000000
DAP$V_EOF               = 00000000
DAP$V_FDL               = 00000001
DAP$V_FTN               = 00000000
DAP$V_GET               = 00000001
DAP$V_HRD               = 00000000
DAP$V_KEY               = 00000001
DAP$V_KGE               = 00000009
DAP$V_KGT               = 0000000A
DAP$V_KRF               = 00000002
DAP$V_LIM               = 0000000E
DAP$V_LOA               = 00000004
DAP$V_LOAD              = 00000000
DAP$V_MSGBLK            = 00000012
DAP$V_MXV               = 00000013
DAP$V_NEF               = 00000009
DAP$V_NIL               = 00000006
DAP$V_NLK               = 0000000B
DAP$V_NUL_CHR           = 00000002
DAP$V_NXR               = 0000000F
DAP$V_ONC               = 00000003
DAP$V_POS               = 00000003
DAP$V_PRN               = 00000002
DAP$V_PUT               = 00000000
DAP$V_RAC               = 00000000
DAP$V_RAH               = 00000007
DAP$V_RCK               = 0000000F
DAP$V_REA               = 00000012
DAP$V_RLK               = 0000000C
DAP$V_ROP               = 00000003
DAP$V_ROPBIO            = 0000000D
DAP$V_ROPWAT            = 00000010
DAP$V_RRL               = 00000011
DAP$V_RWC               = 00000001
DAP$V_RWO               = 00000000
DAP$V_SCF               = 00000015
DAP$V_SHRDEL            = 00000002
DAP$V_SHRGET            = 00000001
DAP$V_SHRPUT            = 00000000
DAP$V_SHRUPD            = 00000003
DAP$V_SPL               = 00000014
DAP$V_SQO               = 00000012
DAP$V_SUP               = 00000008
DAP$V_TEF               = 0000001A
DAP$V_TMD               = 0000000B
DAP$V_TMP               = 0000000A
DAP$V_TPT               = 00000006
DAP$V_TRN               = 00000004
DAP$V_UIF               = 00000002
DAP$V_ULK               = 00000005
DAP$V_UPD               = 00000003
DAP$V_UPI               = 00000005
DAP$V_VAXVMS            = 00000034
DAP$V_WBH               = 00000008
DAP$V_WCK               = 0000000E
DAP$W_ALLMENU             00000040
DAP$W_BLS                 00000048
DAP$W_BUFSIZ              00000040
DAP$W_CHECK               00000042
```

```
DAP$W_CTLMENU                   00000044
DAP$W_DEQ1                      00000054
DAP$W_DEQ2                      00000052
DAP$W_DFL                       00000044
DAP$W_DISPLAY1                  0000004C
DAP$W_DISPLAY2                  00000054
DAP$W_FFB                       00000072
DAP$W_IFL                       00000046
DAP$W_LRL                       00000070
DAP$W_MRL                       00000072
DAP$W_MRS                       0000004A
DAP$W_PARTNER                   00000006
DAP$W_POS                       0000004C
DAP$W_POS_TMP                   0000004A
DAP$W_PROGRP                    00000054
DAP$W_PROMENU                   00000040
DAP$W_PROOWN                    00000052
DAP$W_PROSYS                    00000050
DAP$W_PROWLD                    00000056
DAP$W_RVN                       00000042
DAP$W_SSP_MENU                  00000080
DAP$W_TIMENU                    00000040
DAP$W_VERSION                   00000004
DAP$W_VOL                       00000042
EXIT_FAILURE                    00000589 R      02
EXIT_SUCCESS                    0000058C R      02
FAB$B_BKS                     = 0000003E
FAB$B_FAC                     = 00000016
FAB$B_FNS                     = 00000034
FAB$B_FSZ                     = 0000003F
FAB$B_ORG                     = 0000001D
FAB$B_RAT                     = 0000001E
FAB$B_RFM                     = 0000001F
FAB$B_SHR                     = 00000017
FAB$C_FIX                     = 00000001
FAB$C_IDX                     = 00000020
FAB$C_REL                     = 00000010
FAB$C_SEQ                     = 00000000
FAB$C_STM                     = 00000004
FAB$C_STMCR                   = 00000006
FAB$C_STMLF                   = 00000005
FAB$C_UDF                     = 00000000
FAB$C_VAR                     = 00000002
FAB$C_VFC                     = 00000003
FAB$L_ALQ                     = 00000010
FAB$L_FNA                     = 0000002C
FAB$L_FOP                     = 00000004
FAB$L_MRN                     = 00000038
FAB$M_CR                      = 00000002
FAB$V_BIO                     = 00000005
FAB$V_BLK                     = 00000003
FAB$V_BRO                     = 00000006
FAB$V_CBT                     = 00000015
FAB$V_CR                      = 00000001
FAB$V_CTG                     = 00000014
FAB$V_DEL                     = 00000002
FAB$V_DLT                     = 0000000F
FAB$V_DMO                     = 0000000C
FAB$V_EXE                     = 00000007
FAB$V_FTN                     = 00000000
FAB$V_GET                     = 00000001
FAB$V_MXV                     = 00000001
FAB$V_NAM                     = 00000018
FAB$V_NEF                     = 0000000A
FAB$V_NIL                     = 00000005
FAB$V_POS                     = 00000008
FAB$V_PRN                     = 00000002
FAB$V_PUT                     = 00000000
FAB$V_RCK                     = 00000017
FAB$V_RWC                     = 0000000B
FAB$V_RWO                     = 00000007
FAB$V_SCF                     = 0000000E
FAB$V_SHRDEL                  = 00000002
FAB$V_SHRGET                  = 00000001
FAB$V_SHRPUT                  = 00000000
FAB$V_SHRUPD                  = 00000003
FAB$V_SPL                     = 0000000D
FAB$V_SQO                     = 00000006
FAB$V_SUP                     = 00000002
FAB$V_TEF                     = 0000001C
FAB$V_TMD                     = 00000004
FAB$V_TMP                     = 00000003
FAB$V_TRN                     = 00000004
FAB$V_UPD                     = 00000003
FAB$V_UPI                     = 00000006
FAB$V_WCK                     = 00000009
FAB$W_BLS                     = 0000003C
FAB$W_DEQ                     = 00000014
FAB$W_MRS                     = 00000036
FAL$BUILD_HEAD                  ********    X   02
FAL$BUILD_TAIL                  ********    X   02
FAL$B_ACCFUNC                   000001F6
FAL$B_ACCOPT                    000001F5
FAL$B_DATATYPE                  000001F4
FAL$B_DISABLE                   00000006
FAL$B_ENABLE                    00000005
FAL$B_LOGGING                   00000004
FAL$B_MISCOPT                   00000007
FAL$B_RAC                       000001F7
FAL$B_RBK_CACHE                 00000012
FAL$B_RCVBUFIDX                 00000011
FAL$B_VALUE                     00000010
FAL$CVT_BN8_EXT                 ********    X   02
FAL$C_WRKBLN                    00002000
FAL$DECODE_ACC                  0000010B RG     02
FAL$DECODE_ALL                  0000030B RG     02
FAL$DECODE_ATT                  000000A8 RG     02
FAL$DECODE_CMP                  00000275 RG     02
FAL$DECODE_CNF                  00000000 RG     02
FAL$DECODE_CON                  0000026D RG     02
FAL$DECODE_CTL                  00000196 RG     02
FAL$DECODE_KEY                  00000286 RG     02
FAL$DECODE_NAM                  00000424 RG     02
FAL$DECODE_PRO                  00000398 RG     02
```

```
FAL$DECODE_TIM          00000368 RG    02
FAL$INIT_ACLXAB         ******** X     02
FAL$INIT_DATXAB         ******** X     02
FAL$INIT_KEYXAB         ******** X     02
FAL$INIT_PROXAB         ******** X     02
FAL$INIT_RDTXAB         ******** X     02
FAL$K_KEYNAM          = 00000020
FAL$K_WRKBLN            00002000
FAL$LOG_REQNAM          ******** X     02
FAL$LOG_REQNAM2         ******** X     02
FAL$L_ACLXAB            00000C00
FAL$L_ALLXABINI         00000074
FAL$L_CHAIN_NXT         0000007C
FAL$L_DATXAB            00000320
FAL$L_FAB               00000200
FAL$L_FAB2              00000800
FAL$L_FHCXAB            000002F4
FAL$L_FOP               000001F8
FAL$L_KEYNAM            00001C00
FAL$L_KEYXAB            00001000
FAL$L_KEYXABINI         00000078
FAL$L_NAM               00000294
FAL$L_NAM2              00000850
FAL$L_NUMBER            000001FC
FAL$L_PROXAB            0000034C
FAL$L_RAB               00000250
FAL$L_RCVBUF            0000005C
FAL$L_RDTXAB            000003B0
FAL$L_RMS_PTR           0000006C
FAL$L_STB               000000C0
FAL$L_SUMXAB            000003A4
FAL$L_TEMP              000003F4
FAL$L_USE_SC1           000000A8
FAL$L_USE_SC2           000000AC
FAL$L_USE_VER           000000A4
FAL$M_DATXAB          = 00000004
FAL$M_RDTXAB          = 00000010
FAL$Q_BLD               00000050
FAL$Q_DIRNAME           00000088
FAL$Q_FALLOG            00000090
FAL$Q_FLG               00000000
FAL$Q_MBX               00000038
FAL$Q_MBXIOSB           00000030
FAL$Q_RCV               00000040
FAL$Q_RCVIOSB           00000020
FAL$Q_RMS               00000064
FAL$Q_STATE_CTX         00000008
FAL$Q_SYSNET            00000098
FAL$Q_TEMP              000003F8
FAL$Q_VOLNAME           00000080
FAL$Q_XMT               00000048
FAL$Q_XMTIOSB           00000028
FAL$TRANSMIT            ******** X     02
FAL$T_DAP               00000100
FAL$T_DIRNAME           00001F00
FAL$T_EXPAND            00000500
FAL$T_EXPAND2           00000A00
FAL$T_FALLOG            00001C00
FAL$T_FILESPEC          00000400
FAL$T_FILESPEC2         00000900
FAL$T_KEYBUF            00000700
FAL$T_MBXBUF            00001980
FAL$T_PRTBUF1           00001A00
FAL$T_PRTBUF2           00001B00
FAL$T_RESULT            00000600
FAL$T_RESULT2           00000B00
FAL$T_SYSNET            00001D00
FAL$T_VOLNAME           00001E00
FAL$UNS_KEY             ******** X     02
FAL$V_ACLXAB          = 00000001
FAL$V_ATT_MSG         = 00000001
FAL$V_BLK_IO          = 00000009
FAL$V_CNF_MSG         = 00000000
FAL$V_DIS_CRC         = 00000030
FAL$V_DIS_MBK         = 00000031
FAL$V_FTM             = 00000008
FAL$V_KEYXAB          = 00000000
FAL$V_LAST_MSG        = 00000018
FAL$V_PROXAB          = 00000003
FAL$V_USE_DBS         = 00000039
FAL$V_USE_SC1         = 0000003C
FAL$V_USE_SC2         = 0000003D
FAL$V_USE_SYS         = 0000003A
FAL$V_USE_VER         = 0000003B
FAL$V_WILD            = 0000000A
FAL$W_DAPBUFSIZ         0000001A
FAL$W_DISPLAY           00000070
FAL$W_LNKCHN            0000001C
FAL$W_MBXCHN            0000001E
FAL$W_QIOBUFSIZ         00000018
FAL$W_RECEIVED          00000072
FAL$W_USE_DBS           000000A0
FAL$W_USE_SYS           000000A2
KEY_FIELD               000001FE R     02
KRF_FIELD               000001F4 R     02
LIB$CVT_OTB             ******** X     02
MAP_FOP_FIELD           00000444 R     02
MAP_ROP_FIELD           000004EB R     02
RAB$B_KRF             = 00000035
RAB$B_KSZ             = 00000034
RAB$B_RAC             = 0000001E
RAB$C_KEY             = 00000001
RAB$C_RFA             = 00000002
RAB$C_SEQ             = 00000000
RAB$L_BKT             = 00000038
RAB$L_KBF             = 00000030
RAB$L_ROP             = 00000004
RAB$V_BIO             = 0000000B
RAB$V_EOF             = 00000C08
RAB$V_FDL             = 00000006
RAB$V_KGE             = 00000015
RAB$V_KGT             = 00000016
RAB$V_LIM             = 0000000E
RAB$V_LOA             = 0000000D
```

```
RAB$V_NLK                       =  00000014
RAB$V_NXR                       =  00000017
RAB$V_RAH                       =  00000009
RAB$V_REA                       =  00000002
RAB$V_RLK                       =  00000013
RAB$V_RRL                       =  00000003
RAB$V_TPT                       =  00000001
RAB$V_UIF                       =  00000004
RAB$V_ULK                       =  00000012
RAB$V_WAT                       =  00000011
RAB$V_WBH                       =  0000000A
RAB$W_RFA                       =  00000010
RAC_FIELD                          0000019F R      02
ROP_FIELD                          000001E8 R      02
SEND_CNF                           00000015 R      02
XAB$B_ALN                       =  00000009
XAB$B_AOP                       =  00000008
XAB$B_BKZ                       =  00000016
XAB$B_DAN                       =  0000000A
XAB$B_DTP                       =  00000013
XAB$B_FLG                       =  00000012
XAB$B_IAN                       =  00000008
XAB$B_LAN                       =  00000009
XAB$B_NSG                       =  00000014
XAB$B_NUL                       =  00000015
XAB$B_SIZ                       =  0000002E
XAB$C_CYL                       =  00000001
XAB$C_LBN                       =  00000002
XAB$C_VBN                       =  00000003
XAB$L_ALQ                       =  00000010
XAB$L_KNM                       =  00000038
XAB$L_LOC                       =  0000000C
XAB$L_UIC                       =  0000000C
XAB$Q_BDT                       =  00000024
XAB$Q_CDT                       =  00000014
XAB$Q_EDT                       =  0000001C
XAB$Q_RDT                       =  0000000C
XAB$V_CBT                       =  00000005
XAB$V_CHG                       =  00000001
XAB$V_CTG                       =  00000007
XAB$V_DUP                       =  00000000
XAB$V_HRD                       =  00000000
XAB$V_NUL                       =  00000002
XAB$V_ONC                       =  00000001
XAB$W_DEQ                       =  00000014
XAB$W_DFL                       =  0000001C
XAB$W_GRP                       =  0000000E
XAB$W_IFL                       =  0000001A
XAB$W_LRL                          ********     X    02
XAB$W_MBM                       =  0000000C
XAB$W_POS                       =  0000001E
XAB$W_PRO                       =  00000008
XAB$W_RVN                       =  00000008
XAB$W_VOL                       =  0000000A
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00002000 (8192.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
FAL$CODE	00000590 (1424.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	37	00:00:00.05	00:00:01.61
Command processing	142	00:00:00.36	00:00:02.39
Pass 1	450	00:00:13.74	00:00:51.94
Symbol table sort	0	00:00:01.65	00:00:04.98
Pass 2	192	00:00:02.86	00:00:10.29
Symbol table output	59	00:00:00.30	00:00:00.60
Psect synopsis output	1	00:00:00.03	00:00:00.73
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	883	00:00:19.00	00:01:12.55

The working set limit was 1950 pages.
111221 bytes (218 pages) of virtual memory were used to buffer the intermediate code.
There were 90 pages of symbol table space allocated to hold 1637 non-local and 110 local symbols.
1042 source lines were read in Pass 1, producing 17 object records in Pass 2.
36 pages of virtual memory were used to define 35 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[FAL.OBJ]FAL.MLB;1	20
_255DUA28:[SYSLIB]STARLET.MLB;2	12
TOTALS (all libraries)	32

2114 GETS were required to define 32 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:FALACTMSG/OBJ=OBJ$:FALACTMSG MSRC$:FALACTMSG/UPDATE=(ENH$:FALACTMSG)+LIB$:FAL/LIB
```

FALACTION
LIS

FALDAPCRC
LIS

FALMACROS
MAR

FALBLDXAB
LIS

FALBLDATT
LIS

FALBLDSTS
LIS

FALDEF
MDL

FALDAPIO
LIS

FALACTINI
LIS

FALACTMSG
LIS