```
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
```

FILEID**FALENCODE

```
FFFFFFFFFF    AAAAAA    LL          EEEEEEEEEE  NN      NN    CCCCCCCC    OOOOOO    DDDDDDDD    EEEEEEEEEE
FFFFFFFFFF    AAAAAA    LL          EEEEEEEEEE  NN      NN    CCCCCCCC    OOOOOO    DDDDDDDD    EEEEEEEEEE
FF           AA    AA   LL          EE          NN      NN  CC          OO      OO  DD      DD  EE
FF           AA    AA   LL          EE          NN      NN  CC          OO      OO  DD      DD  EE
FF           AA    AA   LL          EE          NNNN    NN  CC          OO      OO  DD      DD  EE
FF           AA    AA   LL          EE          NNNN    NN  CC          OO      OO  DD      DD  EE
FFFFFFFF     AA    AA   LL          EEEEEEEE    NN  NN  NN  CC          OO      OO  DD      DD  EEEEEEEE
FFFFFFFF     AA    AA   LL          EEEEEEEE    NN  NN  NN  CC          OO      OO  DD      DD  EEEEEEEE
FF           AAAAAAAAAA LL          EE          NN    NNNN  CC          OO      OO  DD      DD  EE
FF           AAAAAAAAAA LL          EE          NN    NNNN  CC          OO      OO  DD      DD  EE
FF           AA    AA   LL          EE          NN      NN  CC          OO      OO  DD      DD  EE                ....
FF           AA    AA   LL          EE          NN      NN  CC          OO      OO  DD      DD  EE                ....
FF           AA    AA   LLLLLLLLLL  EEEEEEEEEE  NN      NN    CCCCCCCC    OOOOOO    DDDDDDDD    EEEEEEEEEE        ....
FF           AA    AA   LLLLLLLLLL  EEEEEEEEEE  NN      NN    CCCCCCCC    OOOOOO    DDDDDDDD    EEEEEEEEEE        ....

LL             IIIIII      SSSSSSSS
LL             IIIIII      SSSSSSSS
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  FALENCODE - ENCODE DAP MESSAGE
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 support routines that encode (build) portions of
0000    35 ;       A DAP message. Included are routines to build a message header and to
0000    36 ;       convert binary data to extensible or image format.
0000    37 ;
0000    38 ; Environment: VAX/VMS, user mode
0000    39 ;
0000    40 ; Author: James A. Krycka,      Creation Date:  16-JUN-1977
0000    41 ;
0000    42 ; Modified By:
0000    43 ;
0000    44 ;       V03-001 JAK0145         J A Krycka      12-APR-1984
0000    45 ;                Track changes in DAP message building algorithm.
0000    46 ;
0000    47 ;--
```

```
0000     49          .SBTTL  DECLARATIONS
0000     50
0000     51 ;
0000     52 ; Include Files:
0000     53 ;
0000     54
0000     55          $DAPHDRDEF                      ; Define DAP message header
0000     56          $DAPCNFDEF                      ; Define DAP Configuration message
0000     57          $FALWRKDEF                      ; Define FAL Work Area symbols
0000     58
0000     59 ;
0000     60 ; Macros:
0000     61 ;
0000     62 ;        None
0000     63 ;
0000     64 ; Equated Symbols:
0000     65 ;
0000     66
0000     67          ASSUME  FAL$Q_FLG EQ 0
0000     68
0000     69 ;
0000     70 ; Own Storage:
0000     71 ;
0000     72 ;        None
```

```
                        74          .SBTTL  FAL$BUILD_HEAD - BUILD DAP MESSAGE HEADER
00000000                75          .PSECT  FAL$CODE        NOSHR,EXE,RD,NOWRT,BYTE
                        76
                        77 ;++
                        78 ; Functional Description:
                        79 ;
                        80 ;       FAL$BUILD_HEAD obtains a buffer and constructs a DAP message header
                        81 ;       in it. FAL$BUILD_TAIL is a companion routine that is called after the
                        82 ;       message body has been formed to update the length value in the header.
                        83 ;       Both 1-byte and 2-byte length fields are supported for DAP message
                        84 ;       blocking.
                        85 ;
                        86 ;       Note that the algorithm for building the header does not support the
                        87 ;       use of optional fields such as STREAMID or SYSPEC.
                        88 ;
                        89 ; Calling Sequence:
                        90 ;
                        91 ;       BSBW    FAL$BUILD_HEAD
                        92 ;
                        93 ; Input Parameters:
                        94 ;
                        95 ;       R0      DAP message type value
                        96 ;       R8      Address of FAL work area
                        97 ;       R9      Address of DAP control block
                        98 ;
                        99 ; Implicit Inputs:
                       100 ;
                       101 ;       DAP$V_BIGBLK
                       102 ;       DAP$V_MSGBLK
                       103 ;       FAL$Q_XMT
                       104 ;       FAL$V_LAST_MSG
                       105 ;
                       106 ; Output Parameters:
                       107 ;
                       108 ;       R3      Address of next byte available for message body
                       109 ;
                       110 ; Implicit Outputs:
                       111 ;
                       112 ;       FAL$Q_BLD
                       113 ;       DAP message header is placed in the message buffer.
                       114 ;
                       115 ; Completion Codes:
                       116 ;
                       117 ;       None
                       118 ;
                       119 ; Side Effects:
                       120 ;
                       121 ;       None
                       122 ;
                       123 ;--
```

```
                     0000   125 ;++
                     0000   126 ; On exit from FAL$BUILD_HEAD the incomplete message header will be either
                     0000   127 ; 2, 3, or 4 bytes long in one of the following three formats:
                     0000   128 ;
                     0000   129 ;       TYPE=msg#       TYPE=msg#       TYPE=8
                     0000   130 ;       FLAGS=0         FLAGS=2         FLAGS=6
                     0000   131 ;                       LENGTH=0        LENGTH=0
                     0000   132 ;                                       LEN256=0
                     0000   133 ;
                     0000   134 ; Note: The four byte format containing the extended length field is used only
                     0000   135 ;       for a DAP Data message.
                     0000   136 ;--
                     0000   137
                     0000   138         ASSUME  DAP$V_STREAMID+1 EQ DAP$V_LENGTH
                     0000   139         ASSUME  DAP$V_LENGTH+1 EQ DAP$V_LEN256
                     0000   140
                     0000   141 FAL$BUILD_HEAD::                        ; Entry point
      48 A8    C1    0000   142         ADDL3   FAL$Q_XMT(R8),-         ; Compute address of first unused byte
53    4C A8          0003   143                 FAL$Q_XMT+4(R8),R3       ;  in transmit buffer and return in R3
54 A8    53    D0    0006   144         MOVL    R3,FAL$Q_BLD+4(R8)      ; Update build message descriptor
   83    50    90    000A   145         MOVB    R0,(R3)+                ; Store DAP message type value
18 68    18    E0    000D   146         BBS     #FAL$V_LAST_MSG,(R8),20$; Branch if this will be last message
                     0011   147                                         ;  to block, thus requiring no length
                     0011   148                                         ;  field in header
      08    50    91    0011   149         CMPB    R0,#DAP$K_DAT_MSG       ; Branch if this is not a Data message
            0B    12    0014   150         BNEQ    10$                     ;  (because only a Data message is
                     0016   151                                         ;  potentially longer than 255 bytes)
            14    E1    0016   152         BBC     #DAP$V_BIGBLK,-         ; Branch if 2-byte length field is
   06 28 A9          0018   153                 DAP$Q_SYSCAP(R9),10$     ;  not supported by partner
    83    06    90    001B   154         MOVB    #<<DAP$M_LENGTH>!-      ; Store FLAGS field indicating that
                     001E   155                   <DAP$M_LEN256>!-       ;  an extended length field will be
                     001E   156                   0>,(R3)+              ;  included
            83    B4    001E   157         CLRW    (R3)+                   ; Reserve space for 2-byte length value
                     0020   158                                         ;  (to be filled in by FAL$BUILD_TAIL)
            05    0020   159         RSB                             ; Exit
            12    E1    0021   160 10$:   BBC     #DAP$V_MSGBLK,-         ; Branch if 1-byte length field is
   03 28 A9          0023   161                 DAP$Q_SYSCAP(R9),20$     ;  not supported by partner
    83    02    90    0026   162         MOVB    #DAP$M_LENGTH,(R3)+     ; Store FLAGS field
            83    94    0029   163 20$:   CLRB    (R3)+                   ; Reserve space for 1-byte length value
                     002B   164                                         ;  (to be filled in by FAL$BUILD_TAIL)
                     002B   165                                         ;  OR overwrite FLAGS field with zero
                     002B   166                                         ;  if message blocking is not being used
            05    002B   167         RSB                             ; Exit
```

```
                        .SBTTL  FAL$BUILD_TAIL - COMPLETE DAP MESSAGE HEADER
0000002C                .PSECT  FAL$CODE        NOSHR,EXE,RD,NOWRT,BYTE

                ;++
                ; Functional Description:
                ;
                ;       FAL$BUILD_TAIL completes construction of the DAP message header that
                ;       FAL$BUILD_HEAD initiated by updating the length value in the header.
                ;       Both 1-byte and 2-byte length fields are supported for DAP message
                ;       blocking.
                ;
                ;       Note that the algorithm for building the header does not support the
                ;       use of optional fields such as STREAMID or SYSPEC.
                ;
                ;       Note also that the LENGTH field will be converted to a STREAMID field
                ;       with a value of zero if a 1-byte length field was allocated by
                ;       FAL$BUILD_HEAD, but the message body turned out to be more than 255
                ;       bytes long which cannot be represented in a single byte field.
                ;
                ; Calling Sequence:
                ;
                ;       BSBW    FAL$BUILD_TAIL
                ;
                ; Input Parameters:
                ;
                ;       R3      Address of last byte of message + 1
                ;       R8      Address of FAL work area
                ;
                ; Implicit Inputs:
                ;
                ;       FAL$Q_BLD
                ;       DAP message header
                ;
                ; Output Parameters:
                ;
                ;       R0-R1   Destroyed
                ;
                ; Implicit Outputs:
                ;
                ;       FAL$Q_BLD
                ;       LENGTH and LEN256 fields (if present) in the header are updated.
                ;
                ; Completion Codes:
                ;
                ;       None
                ;
                ; Side Effects:
                ;
                ;       LENGTH field may be converted to a STREAMID field with a value of zero.
                ;
                ;--
```

```
                     002C    221 ;++
                     002C    222 ; On exit from FAL$BUILD_TAIL the completed message header will be either
                     002C    223 ; 2, 3, or 4 bytes long in one of the following four formats:
                     002C    224 ;
                     002C    225 ;       TYPE=msg#       TYPE=msg#       TYPE=8          TYPE=8
                     002C    226 ;       FLAGS=0         FLAGS=2         FLAGS=6         FLAGS=1
                     002C    227 ;                       LENGTH=size     LENGTH=size0    STREAMID=0
                     002C    228 ;                                       LEN256=size1
                     002C    229 ;
                     002C    230 ; Note: The four byte format containing the extended length field is used only
                     002C    231 ;       for a DAP Data message.
                     002C    232 ;--
                     002C    233
                     002C    234         ASSUME  DAP$V_STREAMID+1 EQ DAP$V_LENGTH
                     002C    235         ASSUME  DAP$V_LENGTH+1 EQ DAP$V_LEN256
                     002C    236
                     002C    237 FAL$BUILD_TAIL::                            ; Entry point
50 A8   53    04 A8  C3    002C    238         SOBL3   FAL$Q_BLD+4(R8),R3,-    ; Compute size of DAP message and
                     0032    239                 FAL$Q_BLD(R8)               ;  update build descriptor
        50    50 A8  7D    0032    240         MOVQ    FAL$Q_BLD(R8),R0        ; Put message descriptor in <R0,R1>
                 0A E1    0036    241         BBC     #<DAP$V_LEN256+8>,-     ; Branch if 2-byte length field was
                06 61            0038    242                 (R1),10$            ;  not allocated in message header
   02 A1   50    04 A3    003A    243         SUBW3   #4,R0,2(R1)             ; Compute size of message body and
                           003F    244                                         ;  store value in <LEN256,LENGTH> field
                     05    003F    245         RSB                             ; Exit
                 09 E1    0040    246 10$:    BBC     #<DAP$V_LENGTH+8>,-     ; Branch if 1-byte length field was
               13 61              0042    247                 (R1),30$            ;  not allocated in message header
           50    03 A2    0044    248         SUBW2   #3,R0                   ; Compute size of message body
      00FF 8F    50 B1    0047    249         CMPW    R0,#255                 ; Branch if length of message body
                 05 1A    004C    250         BGTRU   20$                     ;  will not fit in LENGTH field
     02 A1   50    90    004E    251         MOVB    R0,2(R1)                ; Update LENGTH field
                     05    0052    252         RSB                             ; Exit
   01 A1   01    90    0053    253 20$:    MOVB    #DAP$M_STREAMID,1(R1)   ; Rewrite FLAGS field (converting
                           0057    254                                         ;  LENGTH field into STREAMID field)
                     05    0057    255 30$:    RSB                             ; Exit
```

```
                     0058   257          .SBTTL  FAL$CVT_BN4_EXT - CONVERT BINARY TO EXTENSIBLE
                00000058   258          .PSECT  FAL$CODE        NOSHR,EXE,RD,NOWRT,BYTE
                     0058   259
                     0058   260 ;++
                     0058   261 ; Functional Description:
                     0058   262 ;
                     0058   263 ;       FAL$CVT_BN4_EXT converts an unsigned longword value to an extensible
                     0058   264 ;       field format and stores the result in a minimal number of bytes.
                     0058   265 ;
                     0058   266 ; Calling Sequence:
                     0058   267 ;
                     0058   268 ;       BSBW    FAL$CVT_BN4_EXT
                     0058   269 ;
                     0058   270 ; Input Parameters:
                     0058   271 ;
                     0058   272 ;       R1      Binary value to convert and store
                     0058   273 ;       R3      Address of next byte in buffer to store result
                     0058   274 ;
                     0058   275 ; Implicit Inputs:
                     0058   276 ;
                     0058   277 ;       None
                     0058   278 ;
                     0058   279 ; Output Parameters:
                     0058   280 ;
                     0058   281 ;       R1      Zeroed
                     0058   282 ;       R3      Address of last byte of result + 1
                     0058   283 ;
                     0058   284 ; Implicit Outputs:
                     0058   285 ;
                     0058   286 ;       None
                     0058   287 ;
                     0058   288 ; Completion Codes:
                     0058   289 ;
                     0058   290 ;       None
                     0058   291 ;
                     0058   292 ; Side Effects:
                     0058   293 ;
                     0058   294 ;       None
                     0058   295 ;
                     0058   296 ;--
                     0058   297
                     0058   298 FAL$CVT_BN4_EXT::                    ; Entry point
          83   51    90 0058   299          MOVB    R1,(R3)+         ; Copy 7 bits to DST byte--the high
                     005B   300                                      ;  bit will be corrected later
     51    7F 8F    8A 005B   301          BICB2   #^X7F,R1         ; Discard SRC bits just copied
51   51    F9 8F    9C 005F   302          ROTL    #-7,R1,R1        ; Move next 7 bits into place
               07    13 0064   303          BEQL    10$              ; All done if remaining SRC bits
                     0066   304                                      ;  are zero
  FF A3    80 8F    88 0066   305          BISB2   #^X80,-1(R3)     ; Set extensible bit in DST byte
               EB    11 006B   306          BRB     FAL$CVT_BN4_EXT  ;  and process next byte
                     05 006D   307 10$:     RSB                      ; Exit
```

```
                        .SBTTL  FAL$CVT_BN8_EXT - CONVERT BINARY TO EXTENSIBLE
                        .PSECT  FAL$CODE        NOSHR,EXE,RD,NOWRT,BYTE

                ;++
                ; Functional Description:
                ;
                ;       FAL$CVT_BN8_EXT converts an unsigned quadword value to an extensible
                ;       field format and stores the result in a minimal number of bytes.
                ;
                ;       Note that only source bits 00-62 are used; bit 63 is ignored.
                ;
                ; Calling Sequence:
                ;
                ;       BSBW    FAL$CVT_BN8_EXT
                ;
                ; Input Parameters:
                ;
                ;       R1      Binary value to convert and store (low order bits)
                ;       R2      Binary value to convert and store (high order bits)
                ;       R3      Address of next byte in buffer to store result
                ;
                ; Implicit Inputs:
                ;
                ;       None
                ;
                ; Output Parameters:
                ;
                ;       R1-R2   Zeroed
                ;       R3      Address of last byte of result + 1
                ;
                ; Implicit Outputs:
                ;
                ;       None
                ;
                ; Completion Codes:
                ;
                ;       None
                ;
                ; Side Effects:
                ;
                ;       None
                ;
                ;--
                ;
FAL$CVT_BN8_EXT::                       ; Entry point
                        $CLRBIT #31,R2  ; Clear high order bit so that zero
                ;                       ;  will always propogate on shift
83  51  90      10$:    MOVB    R1,(R3)+ ; Copy 7 bits to DST byte--the high
                ;                       ;  bit will be corrected later
    51  7F 8F   8A      BICB2   #^X7F,R1 ; Discard SRC bits just copied
51  51  F9 8F   79      ASHQ    #-7,R1,R1 ; Move next 7 bits into place
        07      13      BEQL    20$     ; All done if remaining SRC bits
                ;                       ;  are zero
FF A3   80 8F   88      BISB2   #^X80,-1(R3) ; Set extensible bit in DST byte
        EB      11      BRB     10$     ;  and process next byte
                05      RSB             ; Exit
```

```
                     0088   366          .SBTTL  FAL$CVT_BN4_IMG - CONVERT BINARY TO IMAGE
                00000088   367          .PSECT  FAL$CODE        NOSHR,EXE,RD,NOWRT,BYTE
                     0088   368
                     0088   369 ;++
                     0088   370 ; Functional Description:
                     0088   371 ;
                     0088   372 ;       FAL$CVT_BN4_IMG converts an unsigned longword value to an image
                     0088   373 ;       field format (counted binary string) and stores the result in a
                     0088   374 ;       minimal number of bytes.
                     0088   375 ;
                     0088   376 ; Calling Sequence:
                     0088   377 ;
                     0088   378 ;       BSBW    FAL$CVT_BN4_IMG
                     0088   379 ;
                     0088   380 ; Input Parameters:
                     0088   381 ;
                     0088   382 ;       R1      Binary value to convert and store
                     0088   383 ;       R3      Address of next byte in buffer to store result
                     0088   384 ;
                     0088   385 ; Implicit Inputs:
                     0088   386 ;
                     0088   387 ;       None
                     0088   388 ;
                     0088   389 ; Output Parameters:
                     0088   390 ;
                     0088   391 ;       R1      Zeroed
                     0088   392 ;       R2      Destroyed
                     0088   393 ;       R3      Address of last byte of result + 1
                     0088   394 ;
                     0088   395 ; Implicit Outputs:
                     0088   396 ;
                     0088   397 ;       None
                     0088   398 ;
                     0088   399 ; Completion Codes:
                     0088   400 ;
                     0088   401 ;       None
                     0088   402 ;
                     0088   403 ; Side Effects:
                     0088   404 ;
                     0088   405 ;       None
                     0088   406 ;
                     0088   407 ;--
                     0088   408
                     0088   409 FAL$CVT_BN4_IMG::                    ; Entry point
           52   53  D0 0088   410          MOVL    R3,R2               ; Save address of DST count byte
                83  94 008B   411          CLRB    (R3)+               ; Zero DST count byte
                51  D5 008D   412          TSTL    R1                  ; Test value to convert
                0E  13 008F   413          BEQL    20$                 ; All done if value is zero
           83   51  90 0091   414 10$:     MOVB    R1,(R3)+            ; Copy next byte to DST
                62  96 0094   415          INCB    (R2)                ; Increment byte counter
                51  94 0096   416          CLRB    R1                  ; Discard byte just copied
      51   51   F8 8F 9C 0098   417          ROTL    #-8,R1,R1           ; Move next byte into place
                F2  12 009D   418          BNEQ    10$                 ; There is more to do if any
                    009F   419                                          ;  remaining bits are non-zero
                05  009F   420 20$:     RSB                         ; Exit
                    00A0   421
                    00A0   422          .END                        ; End of module
```

```
DAP$B_BITCNT        00000035                FAL$L_KEYXABINI  00000078
DAP$B_DECVER        00000047                FAL$L_NAM        00000294
DAP$B_ECONUM        00000045                FAL$L_NAM2       00000850
DAP$B_FILESYS       00000043                FAL$L_NUMBER     000001FC
DAP$B_FLAGS         00000031                FAL$L_PROXAB     0000034C
DAP$B_LEN256        00000034                FAL$L_RAB        00000250
DAP$B_LENGTH        00000033                FAL$L_RCVBUF     0000005C
DAP$B_OSTYPE        00000042                FAL$L_RDTXAB     000003B0
DAP$B_STREAMID      00000032                FAL$L_RMS_PTR    0000006C
DAP$B_TYPE          00000030                FAL$L_STB        000000C0
DAP$B_USRNUM        00000046                FAL$L_SUMXAB     000003A4
DAP$B_USRVER        00000048                FAL$L_TEMP       000003F4
DAP$B_VERNUM        00000044                FAL$L_USE_SC1    000000A8
DAP$K_DAT_MSG   =   00000008                FAL$L_USE_SC2    000000AC
DAP$M_BITCNT    =   00000008                FAL$L_USE_VER    000000A4
DAP$M_LEN256    =   00000004                FAL$Q_BLD        00000050
DAP$M_LENGTH    =   00000002                FAL$Q_DIRNAME    00000088
DAP$M_SEGMENT   =   00000040                FAL$Q_FALLOG     00000090
DAP$M_STREAMID  =   00000001                FAL$Q_FLG        00000000
DAP$M_TMP1$     =   00000010                FAL$Q_MBX        00000038
DAP$M_TMP2$     =   00000080                FAL$Q_MBXIOSB    00000030
DAP$Q_SYSCAP        00000028                FAL$Q_RCV        00000040
DAP$Q_SYSPEC        00000038                FAL$Q_RCVIOSB    00000020
DAP$V_BIGBLK    =   00000014                FAL$Q_RMS        00000064
DAP$V_LEN256    =   00000002                FAL$Q_STATE_CTX  00000008
DAP$V_LENGTH    =   00000001                FAL$Q_SYSNET     00000098
DAP$V_MSGBLK    =   00000012                FAL$Q_TEMP       000003F8
DAP$V_STREAMID  =   00000000                FAL$Q_VOLNAME    00000080
DAP$W_BUFSIZ        00000040                FAL$Q_XMT        00000048
FAL$BUILD_HEAD      00000000 RG     02      FAL$Q_XMTIOSB    00000028
FAL$BUILD_TAIL      0000002C RG     02      FAL$T_DAP        00000100
FAL$B_ACCFUNC       000001F6                FAL$T_DIRNAME    00001F00
FAL$B_ACCOPT        000001F5                FAL$T_EXPAND     00000500
FAL$B_DATATYPE      000001F4                FAL$T_EXPAND2    00000A00
FAL$B_DISABLE       00000006                FAL$T_FALLOG     00001C00
FAL$B_ENABLE        00000005                FAL$T_FILESPEC   00000400
FAL$B_LOGGING       00000004                FAL$T_FILESPEC2  00000900
FAL$B_MISCOPT       00000007                FAL$T_KEYBUF     00000700
FAL$B_RAC           000001F7                FAL$T_MBXBUF     00001980
FAL$B_RBK_CACHE     00000012                FAL$T_PRTBUF1    00001A00
FAL$B_RCVBUFIDX     00000011                FAL$T_PRTBUF2    00001B00
FAL$B_VALUE         00000010                FAL$T_RESULT     00000600
FAL$CVT_BN4_EXT     00000058 RG     02      FAL$T_RESULT2    00000B00
FAL$CVT_BN4_IMG     00000088 RG     02      FAL$T_SYSNET     00001D00
FAL$CVT_BN8_EXT     0000006E RG     02      FAL$T_VOLNAME    00001E00
FAL$C_WRKBLR        00002000                FAL$V_LAST_MSG  = 00000018
FAL$K_WRKBLN        00002000                FAL$W_DAPBUFSIZ  0000001A
FAL$L_ALLXAB        00000C00                FAL$W_DISPLAY    00000070
FAL$L_ALLXABINI     00000074                FAL$W_LNKCHN     0000001C
FAL$L_CHAIN_NXT     0000007C                FAL$W_MBXCHN     0000001E
FAL$L_DATXAB        00000320                FAL$W_QIOBUFSIZ  00000018
FAL$L_FAB           00000200                FAL$W_RECEIVED   00000072
FAL$L_FAB2          00000800                FAL$W_USE_DBS    000000A0
FAL$L_FHCXAB        000002F4                FAL$W_USE_SYS    000000A2
FAL$L_FOP           000001F8
FAL$L_KEYNAM        00001C00
FAL$L_KEYXAB        00001000
```

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	000000A0 (160.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	30	00:00:00.02	00:00:02.56
Command processing	112	00:00:00.37	00:00:01.59
Pass 1	201	00:00:03.28	00:00:16.05
Symbol table sort	0	00:00:00.34	00:00:01.25
Pass 2	80	00:00:00.82	00:00:04.73
Symbol table output	11	00:00:00.06	00:00:00.06
Psect synopsis output	2	00:00:00.01	00:00:00.01
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	438	00:00:04.90	00:00:26.82

The working set limit was 1350 pages.
23892 bytes (47 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 374 non-local and 11 local symbols.
422 source lines were read in Pass 1, producing 13 object records in Pass 2.
12 pages of virtual memory were used to define 11 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[FAL.OBJ]FAL.MLB;1	4
_255DUA28:[SYSLIB]STARLET.MLB;2	4
TOTALS (all libraries)	8

497 GETS were required to define 8 macros.

There were no errors, warnings or information messages.

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

FALDECODE
LIS

FALRMSDAP
LIS

FALSTATE
LIS

FDL

CREATEFDL
MAP

FALENCODE
LIS

FALLOGGER
LIS

FALMAIN
LIS