```
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**FALDAPIO

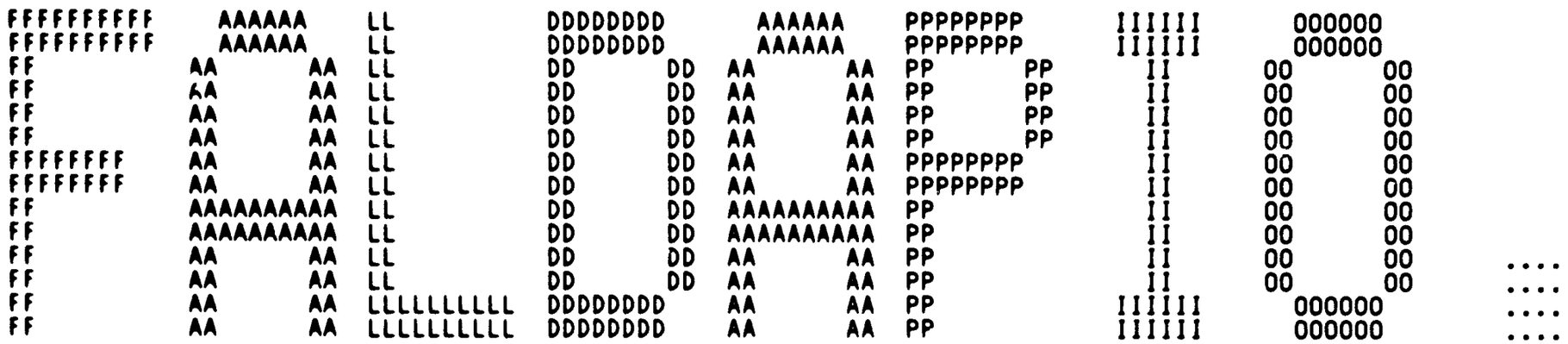

Table of contents

```
0000     1          .TITLE  FALDAPIO - DAP MESSAGE I/O 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 is responsible for receiving and transmitting DAP messages
0000    35 ;       and for receiving messages in the mailbox associated with the logical
0000    36 ;       link.
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 ;       V03-005 JAK0145         J A Krycka      12-APR-1984
0000    45 ;               Revise DAP message blocking algorithm in FAL$TRANSMIT to
0000    46 ;               eliminate an extra move of the data and rename the message
0000    47 ;               descriptors for clarity.
0000    48 ;
0000    49 ;       V03-004 JAK0139         J A Krycka      02-APR-1984
0000    50 ;               Support DAP buffer size up to 65535 bytes.
0000    51 ;
0000    52 ;       V03-003 JAK0133         J A Krycka      20-MAR-1984
0000    53 ;               Change a signed compare to an unsigned compare in FAL$TRANSMIT.
0000    54 ;
0000    55 ;       V03-002 JAK0137         J A Krycka      12-MAR-1984
0000    56 ;               Minor cleanup.
0000    57 ;
```

```
                        0000     58 ;--
```

```
0000     60          .SBTTL  DECLARATIONS
0000     61
0000     62 ;
0000     63 ; Include Files:
0000     64 ;
0000     65
0000     66          $DAPPLGDEF                      ; Define DAP prologue symbols
0000     67          $DAPHDRDEF                      ; Define DAP message header
0000     68          $DAPCNFDEF                      ; Define DAP Configuration message
0000     69          $FALWRKDEF                      ; Define FAL Work Area symobls
0000     70          $FALSTBDEF                      ; Define Statistics Block symbols
0000     71          $IODEF                          ; Define QIO function codes
0000     72          $MSGDEF                         ; Define mailbox message ID codes
0000     73
0000     74 ;
0000     75 ; Macros:
0000     76 ;
0000     77 ;        None
0000     78 ;
0000     79 ; Equated Symbols:
0000     80 ;
0000     81
0000     82          ASSUME  DAP$Q_DCODE_FLG EQ 0
0000     83          ASSUME  FAL$Q_FLG EQ 0
0000     84
0000     85 ;
0000     86 ; Own Storage:
0000     87 ;
0000     88 ;        None
```

```
                     90          .SBTTL  FAL$RECEIVE - READ AND PARSE DAP MESSAGE
00000000             91          .PSECT  FAL$CODE        NOSHR,EXE,RD,NOWRT,BYTE
                     92
                     93 ;++
                     94 ; Functional Description:
                     95 ;
                     96 ;       FAL$RECEIVE returns the next DAP message read from the partner process
                     97 ;       as a decoded message via the DAP control block.
                     98 ;       DAP inbound message blocking is supported.
                     99 ;
                    100 ; Calling Sequence:
                    101 ;
                    102 ;       BSBW    FAL$RECEIVE
                    103 ;
                    104 ; Input Parameters:
                    105 ;
                    106 ;       R8      Address of FAL work area
                    107 ;       R9      Address of DAP control block
                    108 ;       R10     Address of FAB
                    109 ;       R11     Address of RAB
                    110 ;
                    111 ; Implicit Inputs:
                    112 ;
                    113 ;       DAP$Q_MSG_BUF1
                    114 ;
                    115 ; Output Parameters:
                    116 ;
                    117 ;       R0-R1   Destroyed
                    118 ;
                    119 ; Implicit Outputs:
                    120 ;
                    121 ;       DAP$Q_MSG_BUF1 and DAP$Q_MSG_BUF2 plus
                    122 ;       DAP control block is updated to reflect message parse
                    123 ;
                    124 ; Completion Codes:
                    125 ;
                    126 ;       DAP$Q_DCODE_STS
                    127 ;
                    128 ; Side Effects:
                    129 ;
                    130 ;       None
                    131 ;
                    132 ;--
                    133
                    134 FAL$RECEIVE::                                   ; Entry Point
 08 A9    B5        135         TSTW    DAP$Q_MSG_BUF1(R9)             ; Is there a blocked message in buffer?
    07    12        136         BNEQ    10$                            ; Branch if yes
    24    10        137         BSBB    FAL$RECEIVE_PKT                ; Read next DAP message packet
 40 A8    7D        138         MOVQ    FAL$Q_RCV(R8),-                ; Copy message descriptor to
 08 A9              139                 DAP$Q_MSG_BUF1(R9)             ;  DAP message descriptor
                    140
                    141 ;
                    142 ; Parse next DAP message.
                    143 ;
                    144
    59    DD        145 10$:    PUSHL   R9                             ; Push address of DAP control block
0000'CF   01    FB  146         CALLS   #1,W^FAL$DECODE_MSG            ; Parse the message
```

```
                     0013   147
                     0013   148 ;
                     0013   149 ; Update counter and log the DAP message.
                     0013   150 ;
                     0013   151
      00C4 C8   D6   0013   152          INCL    FAL$L_STB+FAL$L_RCV_MSG(R8)
                     0017   153                                          ; Count DAP message for logging
   0F 68    22   E1   0017   154          BBC     #FAL$V_LOG_MSG,(R8),20$ ; Branch if logging disabled
        14 A9   DD   001B   155          PUSHL   DAP$Q_MSG_BUF2+4(R9)    ; Adddress of DAP message just decoded
        10 A9   DD   001E   156          PUSHL   DAP$Q_MSG_BUF2(R9)      ; Size of DAP message just decoded
      0000'CF   9F   0021   157          PUSHAB  W^FAL$GT_DECODE         ; Address of counted ASCII string
0000'CF    03   FB   0025   158          CALLS   #3,W^FAL$DISPLAY_MSG    ; Print the message
                05   002A   159 20$:     RSB                             ; Exit
```

```
                     002B   161          .SBTTL  FAL$RECEIVE_PKT - RECEIVE DAP PACKET
                0000002B    162          .PSECT  FAL$CODE        NOSHR,EXE,RD,NOWRT,BYTE
                     002B   163
                     002B   164 ;++
                     002B   165 ; Functional Description:
                     002B   166 ;
                     002B   167 ;       FAL$RECEIVE_PKT reads the next DAP message packet from partner process.
                     002B   168 ;
                     002B   169 ; Calling Sequence:
                     002B   170 ;
                     002B   171 ;       BSBW    FAL$RECEIVE_PKT
                     002B   172 ;
                     002B   173 ; Input Parameters:
                     002B   174 ;
                     002B   175 ;       R8      Address of FAL work area
                     002B   176 ;       R9      Address of DAP control block
                     002B   177 ;       R10     Address of FAB
                     002B   178 ;       R11     Address of RAB
                     002B   179 ;
                     002B   180 ; Implicit Inputs:
                     002B   181 ;
                     002B   182 ;       None
                     002B   183 ;
                     002B   184 ; Output Parameters:
                     002B   185 ;
                     002B   186 ;       R0      Completion code
                     002B   187 ;       R1      Destroyed
                     002B   188 ;
                     002B   189 ; Implicit Outputs:
                     002B   190 ;
                     002B   191 ;       FAL$Q_RCV
                     002B   192 ;
                     002B   193 ; Completion Codes:
                     002B   194 ;
                     002B   195 ;       None
                     002B   196 ;
                     002B   197 ; Side Effects:
                     002B   198 ;
                     002B   199 ;       FAL terminates execution on encountering a logical link QIO error.
                     002B   200 ;
                     002B   201 ;--
                     002B   202
                     002B   203 FAL$RECEIVE_PKT::                            ; Entry Point
  10 68    11    E0  002B   204          BBS     #FAL$V_RCVAST,(R8),10$      ; Branch if receive has completed
     0000'CF    D6  002F   205          INCL    W^FAL$GL_RECVWAIT           ; Increment receive wait counter
                     0033   206          $WAITFR_S-                          ; Otherwise wait for it to complete
                     0033   207                  EFN=#FAL$K_RCVEFN           ;
                     003C   208          $CHECK_SS                           ; Check status code and exit on failure
                     003F   209 10$:     $CLRBIT #FAL$V_RCVQIO,(R8)          ; Clear receive QIO outstanding flag
                     0043   210          $CLRBIT #FAL$V_RCVAST,(R8)          ; Clear receive AST delivered flag
                     0047   211
                     0047   212 ;
                     0047   213 ; Form receive descriptor pointing to the DAP message packet just read.
                     0047   214 ;
                     0047   215
  50    11 A8    9A  0047   216          MOVZBL  FAL$B_RCVBUFIDX(R8),R0      ; Get index of buffer
        22 A8    3C  004B   217          MOVZWL  FAL$Q_RCVIOSB+2(R8),-       ; Store message size in receive
```

```
                40 A8        004E   218                     FAL$Q_RCV(R8)                   ;  descriptor
            44 A8   5C A840  D0 0050   219             MOVL    FAL$L_RCVBUF(R8)[R0],-          ; Store buffer address in receive
                             0056   220                     FAL$Q_RCV+4(R8)                 ;  descriptor
                             0056   221
                             0056   222 ;
                             0056   223 ; Post a receive to replenish the one just completed.
                             0056   224 ;
                             0056   225
                      01     10 0056   226             BSBB    FAL$RCV_QIO                     ; Issue a read with an AST
                             05 0058   227             RSB                                     ; Exit
```

```
                                                H 1
     0059   229          .SBTTL  FAL$RCV_QIO - QUEUE LINK RECEIVE
0 0000059   230          .PSECT  FAL$CODE         NOSHR,EXE,RD,NOWRT,BYTE
     0059   231
     0059   232 ;++
     0059   233 ; Functional Description:
     0059   234 ;
     0059   235 ;       FAL$RCV_QIO issues a QIO read request with an AST for the logical link.
     0059   236 ;
     0059   237 ; Calling Sequence:
     0059   238 ;
     0059   239 ;       BSBW    FAL$RCV_QIO
     0059   240 ;
     0059   241 ; Input Parameters:
     0059   242 ;
     0059   243 ;       R8      Address of FAL work area
     0059   244 ;       R9      Address of DAP control block
     0059   245 ;       R10     Address of FAB
     0059   246 ;       R11     Address of RAB
     0059   247 ;
     0059   248 ; Implicit Inputs:
     0059   249 ;
     0059   250 ;       None
     0059   251 ;
     0059   252 ; Output Parameters:
     0059   253 ;
     0059   254 ;       R0      Completion code
     0059   255 ;       R1      Destroyed
     0059   256 ;
     0059   257 ; Implicit Outputs:
     0059   258 ;
     0059   259 ;       None
     0059   260 ;
     0059   261 ; Completion Codes:
     0059   262 ;
     0059   263 ;       None
     0059   264 ;
     0059   265 ; Side Effects:
     0059   266 ;
     0059   267 ;       FAL terminates execution on encountering a logical link QIO error.
     0059   268 ;
     0059   269 ;--
     0059   270
     0059   271 FAL$RCV_QIO::                               ; Entry Point
     0059   272          $SETBIT #FAL$V_RCVQIO,(R8)         ; Signal receive QIO outstanding
     005D   273
     005D   274 ;
     005D   275 ; Log our intent to post a receive QIO.
     005D   276 ;
     005D   277
  08 68   24   E1  005D   278          BBC     #FAL$V_LOG_QIO,(R8),10$ ; Branch if logging disabled
51 0000'CF   9E  0061   279          MOVAB   W^FAL$GT_RCVQIO,R1      ; Get address of counted string
      FF97'  30  0066   280          BSBW    FAL$LOG_QIO             ; Log QIO posted to print file
     0069   281
     0069   282 ;
     0069   283 ; Determine next receive buffer to use and update receive buffer index.
     0069   284 ;
     0069   285
```

```
                50   11 A8    9A  0069    286 10$:    MOVZBL  FAL$B_RCVBUFIDX(R8),R0  ; Get index of last buffer used
                        50    D6  006D    287         INCL    R0                      ; Increment it
                   01   50    D1  006F    288         CMPL    R0,#<FAL$K_RCVBUFCNT-1> ; Is it out of range?
                        02    1B  0072    289         BLEQU   20$                     ; Branch if not
                        50    D4  0074    290         CLRL    R0                      ; Reset it
                11 A8   50    90  0076    291 20$:    MOVB    R0,FAL$B_RCVBUFIDX(R8)  ; Store new index value
                                  007A    292
                                  007A    293 ;
                                  007A    294 ; Issue a read with an AST for the logical link.
                                  007A    295 ;
                                  007A    296
           51   5C A840  DO        007A    297         MOVL    FAL$L_RCVBUF(R8)[R0],R1 ; Get address of receive buffer
                                  007F    298         $QIO_S  EFN=#FAL$K_RCVEFN-      ; Issue read with AST
                                  007F    299                 CHAN=FAL$W_LNKCHN(R8)-  ;
                                  007F    300                 FUNC=#IO$_READVBLK-     ;
                                  007F    301                 IOSB=FAL$Q_RCVIOSB(R8)- ;
                                  007F    302                 ASTADR=W^FAL$RCV_AST-   ;
                                  007F    303                 ASTPRM=R8-              ; FAL work area address
                                  007F    304                 P1=(R1)-                ; Receive buffer address
                                  007F    305                 P2=FAL$W_DAPBUFSIZ(R8)  ; Receive buffer size
                                  00A1    306         $CHECK_SS                       ; Check status code and exit on failure
                              05  00A4    307         RSB                             ; Exit
```

```
                     00A5   309          .SBTTL  FAL$RCV_AST - LINK RECEIVE AST
                000000A5    310          .PSECT  FAL$CODE        NOSHR,EXE,RD,NOWRT,BYTE
                     00A5   311
                     00A5   312 ;++
                     00A5   313 ; Functional Description:
                     00A5   314 ;
                     00A5   315 ;       FAL$RCV_AST processes a receive AST for the logical link.
                     00A5   316 ;
                     00A5   317 ; Calling Sequence:
                     00A5   318 ;
                     00A5   319 ;       Call    #5,FAL$RCV_AST  (invoked by VAX/VMS as an AST)
                     00A5   320 ;
                     00A5   321 ; Input Parameters:
                     00A5   322 ;
                     00A5   323 ;       4(AP)   Address of FAL work area
                     00A5   324 ;
                     00A5   325 ; Implicit Inputs:
                     00A5   326 ;
                     00A5   327 ;       None
                     00A5   328 ;
                     00A5   329 ; Output Parameters:
                     00A5   330 ;
                     00A5   331 ;       None
                     00A5   332 ;
                     00A5   333 ; Implicit Outputs:
                     00A5   334 ;
                     00A5   335 ;       FAL$V_RCVAST
                     00A5   336 ;
                     00A5   337 ; Completion Codes:
                     00A5   338 ;
                     00A5   339 ;       None
                     00A5   340 ;
                     00A5   341 ; Side Effects:
                     00A5   342 ;
                     00A5   343 ;       FAL terminates execution on encountering a logical link QIO error.
                     00A5   344 ;
                     00A5   345 ;--
                     00A5   346
                0180 00A5   347          .ENTRY  FAL$RCV_AST,^M<R7,R8>   ; Entry point from executive
          58  04 AC   D0   00A7   348          MOVL    4(AP),R8                ; Get address of FAL work area
                     00AB   349          $SETBIT #FAL$V_RCVAST,(R8)      ; Signal receive AST delivered
            1C 20 A8   E9   00AF   350          BLBC    FAL$Q_RCVIOSB(R8),20$   ; Branch on receive failure
                     00B3   351
                     00B3   352 ;
                     00B3   353 ; Update counters and log delivery of receive AST.
                     00B3   354 ;
                     00B3   355
          50   22 A8   3C   00B3   356          MOVZWL  FAL$Q_RCVIOSB+2(R8),R0  ; Get message size
        57   00C0 C8   DE   00B7   357          MOVAL   FAL$L_STB(R8),R7        ; Get address of statistics block
                67   D6   00BC   358          INCL    FAL$L_RCV_PKT(R7)       ; Increment RCV message packet count
          10 A7   50   C0   00BE   359          ADDL2   R0,FAL$L_RCV_LNK(R7)    ; Update RCV message byte count
          08 68   23   E1   00C2   360          BBC     #FAL$V_LOG_AST,(R8),10$ ; Branch if logging disabled
        51   0000'CF   9E   00C6   361          MOVAB   W^FAL$GT_RCVQIO,R1      ; Get address of counted string
                FF32'  30   00CB   362          BSBW    FAL$LOG_AST             ; Log AST delivered to print file
                       04   00CE   363 10$:     RET                             ; Exit
          50   20 A8   3C   00CF   364 20$:     MOVZWL  FAL$Q_RCVIOSB(R8),R0    ; Save error code
                     00D3   365          $CHECK_SS                       ; Fatal error--terminate image
```

```
                        367          .SBTTL  FAL$TRANSMIT - SEND DAP MESSAGE
                        368          .PSECT  FAL$CODE        NOSHR,EXE,RD,NOWRT,BYTE
                        369
                        370 ;++
                        371 ; Functional Description:
                        372 ;
                        373 ;       FAL$TRANSMIT writes the next DAP message to the partner process.
                        374 ;       DAP outbound message blocking is supported.
                        375 ;
                        376 ; Calling Sequence:
                        377 ;
                        378 ;       BSBW    FAL$TRANSMIT
                        379 ;
                        380 ; Input Parameters:
                        381 ;
                        382 ;       R8      Address of FAL work area
                        383 ;       R9      Address of DAP control block
                        384 ;       R10     Address of FAB
                        385 ;       R11     Address of RAB
                        386 ;
                        387 ; Implicit Inputs:
                        388 ;
                        389 ;       DAP$V_LENGTH
                        390 ;       DAP$V_MSGBLK
                        391 ;       FAL$Q_BLD
                        392 ;       FAL$Q_XMT
                        393 ;       FAL$W_DAPBUFSIZ
                        394 ;       FAL$V_LAST_MSG
                        395 ;       Flags field of the DAP message
                        396 ;
                        397 ; Output Parameters:
                        398 ;
                        399 ;       R0      Completion code
                        400 ;       R1      Destroyed
                        401 ;
                        402 ; Implicit Outputs:
                        403 ;
                        404 ;       FAL$Q_XMT
                        405 ;       FAL$V_LAST_MSG cleared
                        406 ;
                        407 ; Completion Codes:
                        408 ;
                        409 ;       None
                        410 ;
                        411 ; Side Effects:
                        412 ;
                        413 ;       FAL terminates execution on encountering a logical link QIO error.
                        414 ;
                        415 ;--
                        416
                        417 FAL$TRANSMIT::                          ; Entry Point
3C   BB                 418          PUSHR   #^M<R2,R3,R4,R5>       ; Save registers
                        419
                        420 ;
                        421 ; Update counter and log the DAP message.
                        422 ;
                        423
```

```
      00D8 C8   D6 00D8         INCL    FAL$L_STB+FAL$L_XMT_MSG(R8)
                   00DC                                            ; Count this message for logging
     0F 68   22 E1 00DC         BBC     #FAL$V_LOG_MSG,(R8),10$ ; Branch if logging disabled
        54 A8   DD 00E0         PUSHL   FAL$Q_BLD+4(R8)          ; Address of DAP message
        50 A8   DD 00E3         PUSHL   FAL$Q_BLD(R8)            ; Size of DAP message
      0000'CF   9F 00E6         PUSHAB  W^FAL$GT_ENCODE          ; Address of counted ASCII string
  0000'CF    03 FB 00EA         CALLS   #3,W^FAL$DISPLAY_MSG     ; Print the message
                   00EF
                   00EF ;
                   00EF ; Does partner support message blocking?
                   00EF ;
                   00EF
             12 E1 00EF 10$:    BBC     #DAP$V_MSGBLK,-          ; Branch if message blocking is
     57 28 A9      00F1                 DAP$Q_SYSCAP(R9),50$     ;  disabled
                   00F4
                   00F4 ;
                   00F4 ; Are there any messages in XMT buffer waiting to go?
                   00F4 ;
                   00F4
        48 A8   D5 00F4         TSTL    FAL$Q_XMT(R8)            ; Branch if there are no blocked DAP
           18   13 00F7         BEQL    20$                      ;  messages in buffer
                   00F9
                   00F9 ;
                   00F9 ; Will new message fit in XMT buffer with other messages?
                   00F9 ;
                   00F9
        50 A8   C1 00F9         ADDL3   FAL$Q_BLD(R8),-          ; Compute total # bytes if new message
  50    48 A8      00FC                 FAL$Q_XMT(R8),R0         ;  were added to transmit buffer
  51    1A A8   3C 00FF         MOVZWL  FAL$W_DAPBUFSIZ(R8),R1   ; Prepare for longword compare as size
                   0103                                          ;  in R0 may slightly exceed 64K
     51    50   D1 0103         CMPL    R0,R1                    ; Branch if new message can be blocked
           2B   1B 0106         BLEQU   30$                      ;  with the others
                   0108
                   0108 ;
                   0108 ; Flush XMT buffer to make room for new message.
                   0108 ;
                   0108
  54    48 A8   7D 0108         MOVQ    FAL$Q_XMT(R8),R4         ; Put buffer descriptor in <R4,R5>
           4A   10 010C         BSBB    FAL$TRANSMIT_PKT         ; Send the blocked messages
                   010E ; ***** BLBC    R0,EXIT                  ; Branch on failure (no status returned)
        48 A8   D4 010E         CLRL    FAL$Q_XMT(R8)            ; Initialize byte count in descriptor
                   0111
                   0111 ;
                   0111 ; Determine whether new message should be sent now or deferred (bufferred).
                   0111 ;
                   0111
           09   E1 0111 20$:    BBC     #<DAP$V_LENGTH+8>,-      ; Branch if this message does not allow
     35 54 B8      0113                 @FAL$Q_BLD+4(R8),50$     ;  additional blocking (implicit flush)
  31 68    18   E0 0116         BBS     #FAL$V_LAST_MSG,(R8),50$; Branch if this is last message of
                   011A                                          ;  sequence (explicit flush)
                   011A
                   011A ;
                   011A ; Move new message to XMT buffer, or simply map message if BLD and XMT buffers
                   011A ; are aligned.
                   011A ;
                   011A
        50 A8   D0 011A         MOVL    FAL$Q_BLD(R8),-          ; Update byte count in descriptor
```

```
                48 A8        011D   481                     FAL$Q_XMT(R8)                   ;
                54 A8    D1  011F   482             CMPL    FAL$Q_BLD+4(R8),-               ; Eliminate move of data if BLD and XMT
                4C A8        0122   483                     FAL$Q_XMT+4(R8)                 ;  buffer addresses coincide
                   2B    13  0124   484             BEQL    SUC                             ;
                50 A8    28  0126   485             MOVC3   FAL$Q_BLD(R8),-                 ; Copy the message
                54 B8        0129   486                     @FAL$Q_BLD+4(R8),-              ;
                4C B8        012B   487                     @FAL$Q_XMT+4(R8)                ;
           0000'CF       D6  012D   488             INCL    W^FAL$GL_COUNTER2               ; Update message copy counter
                1E       11  0131   489             BRB     SUC                             ;
                             0133   490
                             0133   491 ;
                             0133   492 ; Append new message to XMT buffer which already abuts XMT buffer.
                             0133   493 ;
                             0133   494
    48 A8       50       D0  0133   495 30$:        MOVL    R0,FAL$Q_XMT(R8)                ; Update byte count in descriptor
                             0137   496
                             0137   497 ;
                             0137   498 ; Determine whether new message should be sent now or deferred (bufferred).
                             0137   499 ;
                             0137   500
                09           E1  0137   501             BBC     #<DAP$V_LENGTH+8>,-             ; Branch if this message does not allow
              04 54 B8           0139   502                     @FAL$Q_BLD+4(R8),40$            ;  additional blocking (implicit flush)
          11 68    18           E1  013C   503             BBC     #FAL$V_LAST_MSG,(R8),SUC; Fall thru if this is last message of
                             0140   504                                                     ;  sequence (explicit flush)
                             0140   505
                             0140   506 ;
                             0140   507 ; Send blocked messages from XMT buffer.
                             0140   508 ;
                             0140   509
    54      48 A8       7D  0140   510 40$:        MOVQ    FAL$Q_XMT(R8),R4                ; Put buffer descriptor in <R4,R5>
                12       10  0144   511             BSBB    FAL$TRANSMIT_PKT                ; Send the blocked messages
                48 A8    D4  0146   512             CLRL    FAL$Q_XMT(R8)                   ; Initialize byte count in descriptor
                06       11  0149   513             BRB     EXIT                            ;
                             014B   514
                             014B   515 ;
                             014B   516 ; Send new message from BLD buffer.
                             014B   517 ;
                             014B   518
    54      50 A8       7D  014B   519 50$:        MOVQ    FAL$Q_BLD(R8),R4                ; Put buffer descriptor in <R4,R5>
                07       10  014F   520             BSBB    FAL$TRANSMIT_PKT                ; Send the message
                             0151   521 ; ***** BRB     EXIT                            ;
                             0151   522
                             0151   523 ;
                             0151   524 ; Return to caller.
                             0151   525 ;
                             0151   526
                             0151   527 SUC:                                            ; No status is returned by this routine
                             0151   528 ; ***** MOVL    #1,R0                           ; Return success
                             0151   529 EXIT:   $CLRBIT #FAL$V_LAST_MSG,(R8)            ; Enable message blocking
                   3C    BA  0155   530             POPR    #^M<R2,R3,R4,R5>                ; Restore registers
                         05  0157   531             RSB                                     ; Exit
```

```
                                     N 1
                                .SBTTL  FAL$TRANSMIT_PKT - SEND DAP PACKET
00000158                        .PSECT  FAL$CODE        NOSHR,EXE,RD,NOWRT,BYTE

                        ;++
                        ; Functional Description:
                        ;
                        ;       FAL$TRANSMIT_PKT sends the specified packet of DAP messages to the
                        ;       partner process.
                        ;
                        ; Calling Sequence:
                        ;
                        ;       BSBW    FAL$TRANSMIT_PKT
                        ;
                        ; Input Parameters:
                        ;
                        ;       R4-R5   Descriptor of message packet to transmit
                        ;       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      Completion code
                        ;       R1      Destroyed
                        ;
                        ; Implicit Outputs:
                        ;
                        ;       None
                        ;
                        ; Completion Codes:
                        ;
                        ;       None
                        ;
                        ; Side Effects:
                        ;
                        ;       FAL terminates execution on encountering a logical link QIO error.
                        ;
                        ;--

                        FAL$TRANSMIT_PKT::                      ; Entry Point
15  10  0158  578               BSBB    FAL$XMT_QIO             ; Send packet to partner
            015A  579               $WAITFR_S-                  ; Wait for I/O to complete
            015A  580                       EFN=#FAL$K_XMTEFN   ;
            0163  581               $CHECK_SS                   ; Check status code and exit on failure
            0166  582               $CLRBIT #FAL$V_XMTQIO,(R8)  ; Clear transmit QIO outstanding flag
            016A  583               $CLRBIT #FAL$V_XMTAST,(R8)  ; Clear transmit AST delivered flag
    05      016E  584               RSB                         ; Exit
```

```
                        016F    586          .SBTTL  FAL$XMT_QIO - QUEUE LINK TRANSMIT
                 0000016F    587          .PSECT  FAL$CODE        NOSHR,EXE,RD,NOWRT,BYTE
                        016F    588
                        016F    589 ;++
                        016F    590 ; Functional Description:
                        016F    591 ;
                        016F    592 ;       FAL$XMT_QIO issues a QIO write request with an AST for the logical link.
                        016F    593 ;
                        016F    594 ; Calling Sequence:
                        016F    595 ;
                        016F    596 ;       BSBW    FAL$XMT_QIO
                        016F    597 ;
                        016F    598 ; Input Parameters:
                        016F    599 ;
                        016F    600 ;       R4-R5   Descriptor of message packet to transmit
                        016F    601 ;       R8      Address of FAL work area
                        016F    602 ;       R9      Address of DAP control block
                        016F    603 ;       R10     Address of FAB
                        016F    604 ;       R11     Address of RAB
                        016F    605 ;
                        016F    606 ; Implicit Inputs:
                        016F    607 ;
                        016F    608 ;       None
                        016F    609 ;
                        016F    610 ; Output Parameters:
                        016F    611 ;
                        016F    612 ;       R0      Completion code
                        016F    613 ;       R1      Destroyed
                        016F    614 ;
                        016F    615 ; Implicit Outputs:
                        016F    616 ;
                        016F    617 ;       None
                        016F    618 ;
                        016F    619 ; Completion Codes:
                        016F    620 ;
                        016F    621 ;       None
                        016F    622 ;
                        016F    623 ; Side Effects:
                        016F    624 ;
                        016F    625 ;       FAL terminates execution on encountering a logical link QIO error.
                        016F    626 ;
                        016F    627 ;--
                        016F    628
                        016F    629 FAL$XMT_QIO::                               ; Entry Point
                        016F    630         $SETBIT #FAL$V_XMTQIO,(R8)          ; Signal transmit QIO outstanding
                        0173    631
                        0173    632 ;
                        0173    633 ; Log out intent to post a transmit QIO.
                        0173    634 ;
                        0173    635
      08 68    24    E1   0173    636         BBC     #FAL$V_LOG_QIO,(R8),10$ ; Branch if logging disabled
51    0000'CF    9E    0177    637         MOVAB   W^FAL$GT_XMTQIO,R1          ; Get address of counted string
          FE81'    30    017C    638         BSBW    FAL$LOG_QIO             ; Log QIO posted to print file
                        017F    639
                        017F    640 ;
                        017F    641 ; Issue a write with a AST for the logical link.
                        017F    642 ;
```

```
                017F    643
                017F    644 10$:    $QIO_S  EFN=#FAL$K_XMTEFN-       ; Issue write with AST
                017F    645                 CHAN=FAL$W_LNKCHN(R8)-  ;
                017F    646                 FUNC=#IO$_WRITEVBLK-    ;
                017F    647                 IOSB=FAL$Q_XMTIOSB(R8)- ;
                017F    648                 ASTADR=W^FAL$XMT_AST-   ;
                017F    649                 ASTPRM=R8-              ; FAL work area address
                017F    650                 P1=(R5)-                ; Transmit buffer address
                017F    651                 P2=R4                   ; Transmit buffer size
                01A0    652         $CHECK_SS                       ; Check status code and exit on failure
     05         01A3    653         RSB                             ; Exit
```

```
                     01A4   655          .SBTTL  FAL$XMT_AST - LINK TRANSMIT AST
                000001A4   656          .PSECT  FAL$CODE         NOSHR,EXE,RD,NOWRT,BYTE
                     01A4   657
                     01A4   658 ;++
                     01A4   659 ; Functional Description:
                     01A4   660 ;
                     01A4   661 ;       FAL$XMT_AST processes a transmit AST for the logical link.
                     01A4   662 ;
                     01A4   663 ; Calling Sequence:
                     01A4   664 ;
                     01A4   665 ;       Call    #5,FAL$XMT_AST  (invoked by VAX/VMS as an AST)
                     01A4   666 ;
                     01A4   667 ; Input Parameters:
                     01A4   668 ;
                     01A4   669 ;       4(AP)   Address of FAL work area
                     01A4   670 ;
                     01A4   671 ; Implicit Inputs:
                     01A4   672 ;
                     01A4   673 ;       None
                     01A4   674 ;
                     01A4   675 ; Output Parameters:
                     01A4   676 ;
                     01A4   677 ;       None
                     01A4   678 ;
                     01A4   679 ; Implicit Outputs:
                     01A4   680 ;
                     01A4   681 ;       FAL$V_XMTAST
                     01A4   682 ;
                     01A4   683 ; Completion Codes:
                     01A4   684 ;
                     01A4   685 ;       None
                     01A4   686 ;
                     01A4   687 ; Side Effects:
                     01A4   688 ;
                     01A4   689 ;       FAL terminates execution on encountering a logical link QIO error.
                     01A4   690 ;
                     01A4   691 ;--
                     01A4   692
                0180 01A4   693          .ENTRY  FAL$XMT_AST,^M<R7,R8>   ; Entry point from executive
    58   04 AC   D0 01A6   694          MOVL    4(AP),R8                ; Get address of FAL work area
                     01AA   695          $SETBIT #FAL$V_XMTAST,(R8)      ; Signal transmit AST delivered
      1D 28 A8   E9 01AE   696          BLBC    FAL$Q_XMTIOSB(R8),20$   ; Branch on transmit failure
                     01B2   697
                     01B2   698 ;
                     01B2   699 ; Update counters and log delivery of transmit AST.
                     01B2   700 ;
                     01B2   701
    50   2A A8   3C 01B2   702          MOVZWL  FAL$Q_XMTIOSB+2(R8),R0  ; Save # bytes transmitted
57  00C0 C8   DE 01B6   703          MOVAL   FAL$L_STB(R8),R7        ; Get address of statistics block
       14 A7   D6 01BB   704          INCL    FAL$L_XMT_PKT(R7)       ; Increment XMT message packet count
  24 A7   50   C0 01BE   705          ADDL2   R0,FAL$L_XMT_LNK(R7)    ; Update XMT message byte count
  08 68   23   E1 01C2   706          BBC     #FAL$V_LOG_AST,(R8),10$ ; Branch if logging disabled
51  0000'CF   9E 01C6   707          MOVAB   W^FAL$GT_XMTQIO,R1      ; Get address of counted string
        FE32'  30 01CB   708          BSBW    FAL$LOG_AST             ; Log AST delivered to print file
               04 01CE   709 10$:     RET                             ; Exit
    50   28 A8   3C 01CF   710 20$:     MOVZWL  FAL$Q_XMTIOSB(R8),R0    ; Save error code
                     01D3   711          $CHECK_SS                       ; Fatal error--terminate image
```

```
                     01D6   713          .SBTTL  FAL$RECEIVE_MBX - READ AND PARSE INTERRUPT
                     01D6   714          .PSECT  FAL$CODE         NOSHR,EXE,RD,NOWRT,BYTE
                     01D6   715
                     01D6   716 ;++
                     01D6   717 ; Functional Description:
                     01D6   718 ;
                     01D6   719 ;       FAL$RECEIVE_MBX returns the next DAP message sent as an interrupt
                     01D6   720 ;       message by the partner process. It is returned as a decoded message
                     01D6   721 ;       via the DAP control block.
                     01D6   722 ;
                     01D6   723 ; Calling Sequence:
                     01D6   724 ;
                     01D6   725 ;       BSBW    FAL$RECEIVE_MBX
                     01D6   726 ;
                     01D6   727 ; Input Parameters:
                     01D6   728 ;
                     01D6   729 ;       R8      Address of FAL work area
                     01D6   730 ;       R9      Address of DAP control block
                     01D6   731 ;       R10     Address of FAB
                     01D6   732 ;       R11     Address of RAB
                     01D6   733 ;
                     01D6   734 ; Implicit Inputs:
                     01D6   735 ;
                     01D6   736 ;       FAL$Q_MBX
                     01D6   737 ;
                     01D6   738 ; Output Parameters:
                     01D6   739 ;
                     01D6   740 ;       R0-R1   Destroyed
                     01D6   741 ;
                     01D6   742 ; Implicit Outputs:
                     01D6   743 ;
                     01D6   744 ;       DAP control block is updated to reflect message parse.
                     01D6   745 ;
                     01D6   746 ; Completion Codes:
                     01D6   747 ;
                     01D6   748 ;       DAP$Q_DCODE_STS
                     01D6   749 ;
                     01D6   750 ; Side Effects:
                     01D6   751 ;
                     01D6   752 ;       FAL terminates execution on encountering a mailbox read error.
                     01D6   753 ;
                     01D6   754 ;--
                     01D6   755
                     01D6   756 FAL$RECEIVE_MBX::                            ; Entry Point
 0C 68   15   E0     01D6   757          BBS     #FAL$V_MBXAST,(R8),10$     ; Branch if receive has completed
                     01DA   758          $WAITFR_S-                         ; Otherwise wait for it to complete
                     01DA   759                  EFN=#FAL$K_MBXEFN          ;
                     01E3   760          $CHECK_SS                          ; Check status code and exit on failure
                     01E6   761 10$:     $CLRBIT #FAL$V_MBXQIO,(R8)         ; Clear mailbox QIO outstanding flag
                     01EA   762          $CLRBIT #FAL$V_MBXAST,(R8)         ; Clear mailbox AST delivered flag
                     01EE   763
                     01EE   764 ;
                     01EE   765 ; Post a receive to replenish the one just completed.
                     01EE   766 ; Note: We run the risk of receiving another mailbox message immediately
                     01EE   767 ;       which may overwrite the current message before we can process it.
                     01EE   768 ;       This would be a very very unlikely occurrence from a timing point of
                     01EE   769 ;       view, and the partner process would be misbehaving badly!
```

```
                     01EE    770 ;
                     01EE    771
            24   10  01EE    772          BSBB    FAL$MBX_RCV_QIO          ; Issue a read with an AST
                     01F0    773
                     01F0    774 ;
                     01F0    775 ; Parse next DAP message.
                     01F0    776 ;
                     01F0    777
        38 A8    7D  01F0    778          MOVQ    FAL$Q_MBX(R8),-          ; Copy mailbox message descriptor to
        08 A9        01F3    779                  DAP$Q_MSG_BUF1(R9)       ;  DAP message descriptor
           59    DD  01F5    780          PUSHL   R9                       ; Push address of DAP control block
0000'CF    01    FB  01F7    781          CALLS   #1,W^FAL$DECODE_MSG      ; Parse the message
                     01FC    782
                     01FC    783 ;
                     01FC    784 ; Update counter and log the DAP message.
                     01FC    785 ;
                     01FC    786
     00C4 C8    D6  01FC    787          INCL    FAL$L_STB+FAL$L_RCV_MSG(R8)
                     0200    788                                           ; Count DAP message for logging
  0F 68    22    E1  0200    789          BBC     #FAL$V_LOG_MSG,(R8),20$  ; Branch if logging disabled
        14 A9    DD  0204    790          PUSHL   DAP$Q_MSG_BUF2+4(R9)     ; Adddress of DAP message just decoded
        10 A9    DD  0207    791          PUSHL   DAP$Q_MSG_BUF2(R9)       ; Size of DAP message just decoded
     0000'CF    9F  020A    792          PUSHAB  W^FAL$GT_DECODE          ; Address of counted ASCII string
0000'CF    03    FB  020E    793          CALLS   #3,W^FAL$DISPLAY_MSG     ; Print the message
                 05  0213    794 20$:     RSB                              ; Exit
```

```
                     796          .SBTTL  FAL$MBX_RCV_QIO - QUEUE MAILBOX RECEIVE
00000214             797          .PSECT  FAL$CODE        NOSHR,EXE,RD,NOWRT,BYTE
                     798
                     799 ;++
                     800 ; Functional Description:
                     801 ;
                     802 ;       FAL$MBX_RCV_QIO issues a QIO read request with an AST for the
                     803 ;       mailbox associated with the logical link.
                     804 ;
                     805 ; Calling Sequence:
                     806 ;
                     807 ;       BSBW    FAL$MBX_RCV_QIO
                     808 ;
                     809 ; Input Parameters:
                     810 ;
                     811 ;       R8      Address of FAL work area
                     812 ;       R9      Address of DAP control block
                     813 ;       R10     Address of FAB
                     814 ;       R11     Address of RAB
                     815 ;
                     816 ; Implicit Inputs:
                     817 ;
                     818 ;       None
                     819 ;
                     820 ; Output Parameters:
                     821 ;
                     822 ;       R0      Completion code
                     823 ;       R1      Destroyed
                     824 ;
                     825 ; Implicit Outputs:
                     826 ;
                     827 ;       None
                     828 ;
                     829 ; Completion Codes:
                     830 ;
                     831 ;       None
                     832 ;
                     833 ; Side Effects:
                     834 ;
                     835 ;       FAL terminates execution on encountering a mailbox read error.
                     836 ;
                     837 ;--
                     838
                     839 FAL$MBX_RCV_QIO::                             ; Entry Point
                     840         $SETBIT #FAL$V_MBXQIO,(R8)          ; Signal mailbox QIO outstanding
                     841
                     842 ;
                     843 ; Log our intent to post a mailbox receive QIO.
                     844 ;
                     845
   08 68    24   E1  846         BBC     #FAL$V_LOG_QIO,(R8),10$ ; Branch if logging disabled
51    0000'CF   9E  847         MOVAB   W^FAL$GT_MBXQIO,R1      ; Get address of counted string
        FDDC'   30  848         BSBW    FAL$LOG_QIO             ; Log QIO posted to print file
                    849
                    850 ;
                    851 ; Issue a read with an AST for the mailbox.
                    852 ;
```

```
                0224    853
                0224    854 10$:    $QIO_S  EFN=#FAL$K_MBXEFN-          ; Issue read with AST
                0224    855                 CHAN=FAL$W_MBXCHN(R8)-      ;
                0224    856                 FUNC=#IO$_READVBLK-         ;
                0224    857                 IOSB=FAL$Q_MBXIOSB(R8)-     ;
                0224    858                 ASTADR=W^FAL$MBX_RCV_AST-
                0224    859                 ASTPRM=R8-                  ; FAL work area address
                0224    860                 P1=FAL$T_MBXBUF(R8)-        ; Mailbox buffer address
                0224    861                 P2=#FAL$K_MBXBUF            ; Mailbox buffer size
                024B    862         $CHECK_SS                           ; Check status code and exit on failure
     05         024E    863         RSB                                 ; Exit
```

```
                     024F    865          .SBTTL  FAL$MBX_RCV_AST - MAILBOX RECEIVE AST
                0000024F    866          .PSECT  FAL$CODE        NOSHR,EXE,RD,NOWRT,BYTE
                     024F    867
                     024F    868 ;++
                     024F    869 ; Functional Description:
                     024F    870 ;
                     024F    871 ;       FAL$MBX_RCV_AST processes an AST for the mailbox associated with the
                     024F    872 ;       logical link. Depending on the type of message, one of three actions is
                     024F    873 ;       performed:
                     024F    874 ;       (1) if it is an interrupt message, a descriptor is formed to point to
                     024F    875 ;           the DAP message.
                     024F    876 ;       (2) if the message indicates that the link has been broken, FAL is
                     024F    877 ;           terminated.
                     024F    878 ;       (3) if it is an extraneous message, it is simply discarded.
                     024F    879 ;
                     024F    880 ; Calling Sequence:
                     024F    881 ;
                     024F    882 ;       Call    #5,FAL$MBX_RCV_AST      (invoked by VAX/VMS as an AST)
                     024F    883 ;
                     024F    884 ; Input Parameters:
                     024F    885 ;
                     024F    886 ;       4(AP)   Address of FAL work area
                     024F    887 ;
                     024F    888 ; Implicit Inputs:
                     024F    889 ;
                     024F    890 ;       None
                     024F    891 ;
                     024F    892 ; Output Parameters:
                     024F    893 ;
                     024F    894 ;       None
                     024F    895 ;
                     024F    896 ; Implicit Outputs:
                     024F    897 ;
                     024F    898 ;       FAL$Q_MBX
                     024F    899 ;       FAL$V_MBXAST
                     024F    900 ;
                     024F    901 ; Completion Codes:
                     024F    902 ;
                     024F    903 ;       None
                     024F    904 ;
                     024F    905 ; Side Effects:
                     024F    906 ;
                     024F    907 ;       FAL terminates execution on encountering a mailbox read error.
                     024F    908 ;
                     024F    909 ;--
                     024F    910
                 0180 024F    911          .ENTRY  FAL$MBX_RCV_AST,^M<R7,R8> ; Entry point from executive
58    04 AC    D0  0251    912          MOVL    4(AP),R8                  ; Get address of FAL work area
                     0255    913          $SETBIT #FAL$V_MBXAST,(R8)        ; Signal mailbox AST delivered
  07 30 A8    E8  0259    914          BLBS    FAL$Q_MBXIOSB(R8),10$     ; Branch on success
50    20 A8    3C  025D    915          MOVZWL  FAL$Q_RCVIOSB(R8),R0      ; Save error code
                     0261    916          $CHECK_SS                         ; Fatal error--terminate image
                     0264    917
                     0264    918 ;
                     0264    919 ; Log the mailbox message type code.
                     0264    920 ;
                     0264    921
```

```
                 1D 68    22    E1  0264    922 10$:    BBC     #FAL$V_LOG_MSG,(R8),20$ ; Branch if logging disabled
                                    0268    923         $FAO_S  CTRSTR=W^FAL$GQ_MBXMSG- ; Format the message
                                    0268    924                 OUTLEN=W^FAL$GW_PRTLEN2-;
                                    0268    925                 OUTBUF=W^FAL$GQ_PRTBUF2-;
                                    0268    926                 P1=FAL$T_MBXBUF(R8)     ; First four bytes of mailbox message
                                    027F    927         $CHECK_STATUS                   ; Check status code
                       FD7B'        30  0282    928         BSBW    FAL$PRINT_FAO_ASTLEVEL  ; Print message
                                        0285    929
                                        0285    930 ;
                                        0285    931 ; Take appropriate action depending on the mailbox message type.
                                        0285    932 ;
                                        0285    933
                 51    1980 C8    9E  0285    934 20$:    MOVAB   FAL$T_MBXBUF(R8),R1     ; Get address of mailbox buffer
                                        028A    935
                                        028A    936         ASSUME  MSG$_ABORT EQ 48
                                        028A    937         ASSUME  MSG$_CONFIRM EQ 49
                                        028A    938         ASSUME  MSG$_CONNECT EQ 50
                                        028A    939         ASSUME  MSG$_DISCON EQ 51
                                        028A    940         ASSUME  MSG$_EXIT EQ 52
                                        028A    941         ASSUME  MSG$_INTMSG EQ 53
                                        028A    942         ASSUME  MSG$_PATHLOST EQ 54
                                        028A    943         ASSUME  MSG$_PROTOCOL EQ 55
                                        028A    944         ASSUME  MSG$_REJECT EQ 56
                                        028A    945         ASSUME  MSG$_THIRDPARTY EQ 57
                                        028A    946         ASSUME  MSG$_TIMEOUT EQ 58
                                        028A    947         ASSUME  MSG$_NETSHUT EQ 59
                                        028A    948
                                        028A    949         $CASEW  SELECTOR=(R1)+,-        ; Mailbox message type:
                                        028A    950                 BASE=#MSG$_ABORT-       ;
                                        028A    951                 DISPL=<-                ;
                                        028A    952                     FAL$TERMINATE-      ;  Disconnect abort
                                        028A    953                     IGNORE-             ;  Connect confirm
                                        028A    954                     IGNORE-             ;  Connect initiate
                                        028A    955                     FAL$TERMINATE-      ;  Synchronous disconnect
                                        028A    956                     FAL$TERMINATE-      ;  Partner exited prematurely
                                        028A    957                     INTERRUPT-          ;  Interrupt message
                                        028A    958                     FAL$TERMINATE-      ;  Partner no longer accessible
                                        028A    959                     FAL$TERMINATE-      ;  NSP protocol error
                                        028A    960                     IGNORE-             ;  Connect reject
                                        028A    961                     FAL$TERMINATE-      ;  Thirdparty disconnect
                                        028A    962                     IGNORE-             ;  Connect timeout
                                        028A    963                     IGNORE-             ;  Network shutting down
                                        028A    964                 >                       ;
                                        02A6    965
                                        02A6    966 ;
                                        02A6    967 ; Ignore unexpected mailbox message.
                                        02A6    968 ;
                                        02A6    969
                                        02A6    970 IGNORE:                                 ; Discard junk mail
                       50        D4     02A6    971         CLRL    R0                      ; Make this a null message
                 38 A8 50        7D     02A8    972         MOVQ    R0,FAL$Q_MBX(R8)        ; Store descriptor of interrupt data
                                04      02AC    973         RET                             ; Exit
                                        02AD    974
                                        02AD    975 ;
                                        02AD    976 ; An interrupt message has been received.
                                        02AD    977 ;
                                        02AD    978
```

```
                                02AD   979 INTERRUPT:                                ;
                          81    B5  02AD   980          TSTW    (R1)+                 ; Skip over unit #
                     50   81    9A  02AF   981          MOVZBL  (R1)+,R0              ; Skip over device name stored
                     51   50    C0  02B2   982          ADDL2   R0,R1                 ;  as a counted ASCII string
                     50   81    9A  02B5   983          MOVZBL  (R1)+,R0              ; Get size of interrupt data field
                38 A8    50    7D  02B8   984          MOVQ    R0,FAL$Q_MBX(R8)      ; Store descriptor of interrupt data
                                02BC   985
                                02BC   986 ;
                                02BC   987 ; Update counters and log delivery of mailbox AST containing a DAP interrupt
                                02BC   988 ; message.
                                02BC   989 ;
                                02BC   990
                57   00C0 C8    DE  02BC   991          MOVAL   FAL$L_STB(R8),R7      ; Get address of statistics block
                           67    D6  02C1   992          INCL    FAL$L_RCV_PKT(R7)     ; Increment RCV message packet count
                  10 A7    50    C0  02C3   993          ADDL2   R0,FAL$L_RCV_LNK(R7)  ; Update RCV message byte count
                  08 68    23    E1  02C7   994          BBC     #FAL$V_LOG_AST,(R8),30$ ; Branch if logging disabled
                51   0000'CF    9E  02CB   995          MOVAB   W^FAL$GT_MBXQIO,R1    ; Get address of counted string
                        FD2D'    30  02D0   996          BSBW    FAL$LOG_AST           ; Log AST delivered to print file
                                04  02D3   997 30$:      RET                           ; Exit
                                02D4   998
                                02D4   999          .END                          ; End of module
```

```
$$COUNT                 =  0000000C
$$T1                    =  00000000
$$T2                    =  00000004
DAP$B_BITCNT               00000035
DAP$B_DCODE_FID            00000019
DAP$B_DCODE_MAC            0000001B
DAP$B_DCODE_MSG            0000001A
DAP$B_DECVER               00000047
DAP$B_ECONUM               00000045
DAP$B_FILESYS              00000043
DAP$B_FLAGS                00000031
DAP$B_LEN256               00000034
DAP$B_LENGTH               00000033
DAP$B_OSTYPE               00000042
DAP$B_STREAMID             00000032
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$K_BLN                  000000C0
DAP$L_CMWA                 00000030
DAP$L_CRC_RSLT             00000020
DAP$L_DCODE_STS            00000018
DAP$L_MSG_MASK             0000001C
DAP$L_SSPWA                00000080
DAP$L_TEMP                 00000090
DAP$M_BITCNT            =  00000008
DAP$M_SEGMENT           =  00000040
DAP$M_TMP1$             =  00000010
DAP$M_TMP2$             =  00000080
DAP$Q_DCODE_FLG            00000000
DAP$Q_MSG_BUF1             00000008
DAP$Q_MSG_BUF2             00000010
DAP$Q_SYSCAP               00000028
DAP$Q_SYSPEC               00000038
DAP$V_LENGTH            =  00000001
DAP$V_MSGBLK            =  00000012
DAP$W_BUFSIZ               00000040
DAP$W_PARTNER              00000006
DAP$W_VERSION              00000004
EXIT                       00000151 R      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$B_VALUE                00000010
FAL$CHECK_SS               ********   X    02
FAL$CHECK_STATUS           ********   X    02
FAL$C_STBBLN               00000040
FAL$C_WRKBLN               00002000
FAL$DECODE_MSG             ********   X    02
FAL$DISPLAY_MSG            ********   X    02
FAL$GL_COUNTER2            ********   X    02
FAL$GL_RECVWAIT            ********   X    02
FAL$GQ_MBXMSG              ********   X    02
FAL$GQ_PRTBUF2             ********   X    02
FAL$GT_DECODE              ********   X    02
FAL$GT_ENCODE              ********   X    02
FAL$GT_MBXQIO              ********   X    02
FAL$GT_RCVQIO              ********   X    02
FAL$GT_XMTQIO              ********   X    02
FAL$GW_PRTLEN2             ********   X    02
FAL$K_MBXBUF            =  00000040
FAL$K_MBXEFN            =  00000003
FAL$K_RCVBUFCNT         =  00000002
FAL$K_RCVEFN            =  00000001
FAL$K_STBBLN               00000040
FAL$K_WRKBLN               00002000
FAL$K_XMTEFN            =  00000002
FAL$LOG_AST                ********   X    02
FAL$LOG_QIO                ********   X    02
FAL$L_ALLXAB               00000C00
FAL$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_RCV_DAT              00000008
FAL$L_RCV_LNK              00000010
FAL$L_RCV_MSG              00000004
FAL$L_RCV_PKT              00000000
FAL$L_RCV_USR              0000000C
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$L_XMT_DAT              0000001C
FAL$L_XMT_LNK              00000024
FAL$L_XMT_MSG              00000018
FAL$L_XMT_PKT              00000014
FAL$L_XMT_USR              00000020
```

```
FAL$MBX_RCV_AST                 0000024F RG    02
FAL$MBX_RCV_QIO                 00000214 RG    02
FAL$PRINT_FAO_ASTLEVEL          ******** X     02
FAL$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$RCV_AST                     000000A5 RG    02
FAL$RCV_QIO                     00000059 RG    02
FAL$RECEIVE                     00000000 RG    02
FAL$RECEIVE_MBX                 000001D6 RG    02
FAL$RECEIVE_PKT                 0000002B RG    02
FAL$TERMINATE                   ******** X     02
FAL$TRANSMIT                    000000D6 RG    02
FAL$TRANSMIT_PKT                00000158 RG    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$T_VOLNAME                   00001E00
FAL$V_LAST_MSG                = 00000018
FAL$V_LOG_AST                 = 00000023
FAL$V_LOG_MSG                 = 00000022
FAL$V_LOG_QIO                 = 00000024
FAL$V_MBXAST                  = 00000015
FAL$V_MBXQIO                  = 00000014
FAL$V_RCVAST                  = 00000011
FAL$V_RCVQIO                  = 00000010
FAL$V_XMTAST                  = 00000013
FAL$V_XMTQIO                  = 00000012
FAL$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$W_USE_SYS                   000000A2
FAL$XMT_AST                     000001A4 RG    02
FAL$XMT_QIO                     0000016F RG    02
IGNORE                          000002A6 R     02
INTERRUPT                       000002AD R     02
IO$_READVBLK                  = 00000031
IO$_WRITEVBLK                 = 00000030
MSG$_ABORT                    = 00000030
MSG$_CONFIRM                  = 00000031
MSG$_CONNECT                  = 00000032
MSG$_DISCON                   = 00000033
MSG$_EXIT                     = 00000034
MSG$_INTMSG                   = 00000035
MSG$_NETSHUT                  = 0000003B
MSG$_PATHLOST                 = 00000036
MSG$_PROTOCOL                 = 00000037
MSG$_REJECT                   = 00000038
MSG$_THIRDPARTY               = 00000039
MSG$_TIMEOUT                  = 0000003A
SUC                             00000151 R     02
SYS$FAO                         ********   X   02
SYS$QIO                         ******** GX    02
SYS$WAITFR                      ******** GX    02
```

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	000002D4 (724.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	34	00:00:00.05	00:00:02.26
Command processing	114	00:00:00.34	00:00:02.80
Pass 1	308	00:00:07.46	00:00:31.64
Symbol table sort	0	00:00:01.01	00:00:03.61
Pass 2	169	00:00:01.87	00:00:07.39
Symbol table output	24	00:00:00.12	00:00:01.67
Psect synopsis output	1	00:00:00.01	00:00:00.01
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	652	00:00:10.86	00:00:49.39

The working set limit was 1500 pages.
60593 bytes (119 pages) of virtual memory were used to buffer the intermediate code.
There were 50 pages of symbol table space allocated to hold 908 non-local and 35 local symbols.
999 source lines were read in Pass 1, producing 23 object records in Pass 2.
28 pages of virtual memory were used to define 26 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[FAL.OBJ]FAL.MLB;1	11
_255DUA28:[SYSLIB]STARLET.MLB;2	12
TOTALS (all libraries)	23

1125 GETS were required to define 23 macros.

There were no errors, warnings or information messages.

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

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

FALACTION
LIS

FALDAPCRC
LIS

FALMACROS
MAR

FALBLDXAB
LIS

FALBLDATT
LIS

FALBLDSTS
LIS

FALDEF
MDL

FALDAPIO
LIS

FALACTINI
LIS

FALACTMSG
LIS

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

FALDECODE
LIS

FALRMSDAP
LIS

FALSTATE
LIS

FDL

CREATEFDL
MAP

FALENCODE
LIS

FALLOGGER
LIS

FALMAIN
LIS