


```

FFFFFFFFF  AAAAAA  LL  DDDDDDD  AAAAAA  PPPPPPP  IIIII  000000
FFFFFFFFF  AAAAAA  LL  DDDDDDD  AAAAAA  PPPPPPP  IIIII  000000
FF        AA      AA  LL  DD      DD  AA      AA  PP      PP
FF        AA      AA  LL  DD      DD  AA      AA  PP      PP
FF        AA      AA  LL  DD      DD  AA      AA  PP      PP
FF        AA      AA  LL  DD      DD  AA      AA  PP      PP
FFFFFFFFF  AA      AA  LL  DD      DD  AA      AA  PPPPPPP  PP
FFFFFFFFF  AA      AA  LL  DD      DD  AA      AA  PPPPPPP  PP
FF        AAAAAA  LL  DD      DD  AAAAAA  AA  PP
FF        AAAAAA  LL  DD      DD  AAAAAA  AA  PP
FF        AA      AA  LL  DD      DD  AA      AA  PP
FF        AA      AA  LL  DD      DD  AA      AA  PP
FF        AA      AA  LL  DD      DD  AA      AA  PP
FF        AA      AA  LLLLLLLLL  DDDDDDD  AA      AA  PP
FF        AA      AA  LLLLLLLLL  DDDDDDD  AA      AA  PP

LL          IIIII  SSSSSSS
LL          IIIII  SSSSSSS
LL          II     SS
LL          II     SS
LL          II     SS
LL          II     SS
LL          II     SSSSSS
LL          II     SSSSSS
LL          II     SS
LL          II     SS
LL          II     SS
LL          II     SS
LLLLLLLLLL IIIII  SSSSSSS
LLLLLLLLLL IIIII  SSSSSSS

```


(2)	60	DECLARATIONS
(3)	90	FAL\$RECEIVE - READ AND PARSE DAP MESSAGE
(4)	161	FAL\$RECEIVE_PKT - RECEIVE DAP PACKET
(5)	229	FAL\$RCV_QIO - QUEUE LINK RECEIVE
(6)	309	FAL\$RCV_AST - LINK RECEIVE AST
(7)	367	FAL\$TRANSMIT - SEND DAP MESSAGE
(8)	533	FAL\$TRANSMIT_PKT - SEND DAP PACKET
(9)	586	FAL\$XMT_QIO - QUEUE LINK TRANSMIT
(10)	655	FAL\$XMT_AST - LINK TRANSMIT AST
(11)	713	FAL\$RECEIVE_MBX - READ AND PARSE INTERRUPT
(12)	796	FAL\$MBX_RCV_QIO - QUEUE MAILBOX RECEIVE
(13)	865	FAL\$MBX_RCV_AST - MAILBOX RECEIVE AST


```
0000 1      .TITLE FALDAPIO - DAP MESSAGE I/O ROUTINES
0000 2      .IDENT 'V04-000'
0000 3
0000 4
0000 5 *****
0000 6
0000 7      *
0000 8      *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 9      *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 10     *  ALL RIGHTS RESERVED.
0000 11     *
0000 12     *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 13     *  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 14     *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 15     *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 16     *  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 17     *  TRANSFERRED.
0000 18     *
0000 19     *  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 20     *  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 21     *  CORPORATION.
0000 22     *
0000 23     *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 24     *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
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
```


FALDAP10
V04-000

- DAP MESSAGE I/O ROUTINES

B 1

16-SEP-1984 01:41:37 VAX/VMS Macro V04-00
5-SEP-1984 01:16:43 [FAL.SRC]FALDAP10.MAR;1

Page 2
(1)

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 symbols
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
```



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

08 A9 B5 0000 135
07 12 0003 136
24 10 0005 137
40 A8 7D 0007 138
08 A9 000A 139

0000*CF 59 DD 000C 145
01 FB 000E 146


```
0013 147
0013 148 :
0013 149 : Update counter and log the DAP message.
0013 150 :
0013 151 :
00C4 C8 D6 0013 152 INCL FALS$_STB+FALS$_RCV_MSG(R8)
0017 153
0F 68 22 E1 0017 154 BBC #FALS$V LOG_MSG,(R8),20$ : Count DAP message for logging
14 A9 DD 001B 155 PUSHL DAPSQ_MSG_BUF2+4(R9) : Branch if logging disabled
10 A9 DD 001E 156 PUSHL DAPSQ_MSG_BUF2(R9) : Address of DAP message just decoded
0000'CF 03 9F 0021 157 PUSHAB W^FALS$GT DECODE : Size of DAP message just decoded
0000'CF 03 FB 0025 158 CALLS #3,W^FALS$DISPLAY_MSG : Address of counted ASCII string
05 002A 159 20$: RSB : Print the message
: Exit
```



```
0000002B 161 .SBTTL FALS$RECEIVE_PKT - RECEIVE DAP PACKET
002B 162 .PSECT FALS$CODE NOSHR,EXE,RD,NOWRT,BYTE
002B 163
002B 164 :++
002B 165 : Functional Description:
002B 166 :
002B 167 : FALS$RECEIVE_PKT reads the next DAP message packet from partner process.
002B 168 :
002B 169 : Calling Sequence:
002B 170 :
002B 171 : BSBW FALS$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 : FALS$Q_RCV
002B 192 :
002B 193 : Completion Codes:
002B 194 :
002B 195 : None
002B 196 :
002B 197 : Side Effects:
002B 198 :
002B 199 : FAL terminates execution on encountering a logical link QIO error.
002B 200 :
002B 201 :--
002B 202
002B 203 FALS$RECEIVE_PKT::
10 68 11 E0 002B 204 BBS #FALS$V_RCVAST,(R8),10$ ; Entry Point
0000'CF D6 002F 205 INCL W^FALS$GL_RECVWAIT ; Branch if receive has completed
0033 206 $WAITFR_S- ; Increment receive wait counter
0033 207 -EFN=#FALS$K_RCVEFN ; Otherwise wait for it to complete
003C 208 $CHECK SS ; Check status code and exit on failure
003F 209 10$: $CLRBIT #FALS$V_RCVQIO,(R8) ; Clear receive QIO outstanding flag
0043 210 $CLRBIT #FALS$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 FALS$B_RCVBUFIDX(R8),R0 ; Get index of buffer
22 A8 3C 004B 217 MOVZWL FALS$Q_RCVIOSB+2(R8),- ; Store message size in receive
```



```

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

```



```
00000059 229      .SBTTL FALSRCV_QIO - QUEUE LINK RECEIVE
0059      230      .PSECT FAL$CODE          NOSHR,EXE,RD,NOWRT,BYTE
0059      231
0059      232      :++
0059      233      : Functional Description:
0059      234      :
0059      235      :     FALSRCV_QIO issues a QIO read request with an AST for the logical link.
0059      236      :
0059      237      : Calling Sequence:
0059      238      :
0059      239      :     BSBW    FALSRCV_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 FALSRCV_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 :
005D      278      BBC      #FAL$V_LOG_QIO,(R8),10$ ; Branch if logging disabled
0061      279      MOVAB   W^FAL$GT_RCVQIO,R1 ; Get address of counted string
0066      280      BSBW    FAL$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
```

08 68 24 E1
51 0000'CF 9E
FF97' 30


```
50 11 A8 9A 0069 286 10$: MOVZBL FALS$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,#<FALS$K_RCVBUFCNT-1> ; Is it out of range?
      02 1B 0072 289      BLEQU 20$ ; Branch if not
      50 D4 0074 290      CLRL R0 ; Reset it
11 A8 50 90 0076 291 20$: MOVB R0,FALS$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 D0 007A 297      MOVL FALS$L_RCVBUF(R8)[R0],R1 ; Get address of receive buffer
      007F 298      $QIO_S EFN=#FALS$K_RCVEFN- ; Issue read with AST
      007F 299      CHAN=FALS$W_LNKCHN(R8)- ;
      007F 300      FUNC=#IOS$ READVBLK- ;
      007F 301      IOSB=FALS$Q_RCVIOSB(R8)- ;
      007F 302      ASTADR=W^FALS$RCV_AST- ;
      007F 303      ASTPRM=R8- ; FAL work area address
      007F 304      P1=(R1)- ; Receive buffer address
      007F 305      P2=FALS$W_DAPBUFSIZ(R8) ; Receive buffer size
      00A1 306      $CHECK_SS ; Check status code and exit on failure
      05 00A4 307      RSB ; Exit
```



```
000000A5 309      .SBTTL  FAL$RCV_AST - LINK RECEIVE AST
000000A5 310      .PSECT  FAL$CODE      NOSHR,EXE,RD,NOWRT,BYTE
000000A5 311
000000A5 312      :++
000000A5 313      : Functional Description:
000000A5 314      :
000000A5 315      : FAL$RCV_AST processes a receive AST for the logical link.
000000A5 316      :
000000A5 317      : Calling Sequence:
000000A5 318      :
000000A5 319      : Call    #5,FAL$RCV_AST  (invoked by VAX/VMS as an AST)
000000A5 320      :
000000A5 321      : Input Parameters:
000000A5 322      :
000000A5 323      : 4(AP)  Address of FAL work area
000000A5 324      :
000000A5 325      : Implicit Inputs:
000000A5 326      :
000000A5 327      : None
000000A5 328      :
000000A5 329      : Output Parameters:
000000A5 330      :
000000A5 331      : None
000000A5 332      :
000000A5 333      : Implicit Outputs:
000000A5 334      :
000000A5 335      : FAL$V_RCVAST
000000A5 336      :
000000A5 337      : Completion Codes:
000000A5 338      :
000000A5 339      : None
000000A5 340      :
000000A5 341      : Side Effects:
000000A5 342      :
000000A5 343      : FAL terminates execution on encountering a logical link QIO error.
000000A5 344      :
000000A5 345      :--
000000A5 346
000000A5 347      .ENTRY  FAL$RCV_AST,^M<R7,R8>  ; Entry point from executive
58 04 AC 0180 00A7 348      MOVL    4(AP),R8      ; Get address of FAL work area
000000AB 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
000000B3 351
000000B3 352      :
000000B3 353      : Update counters and log delivery of receive AST.
000000B3 354      :
000000B3 355      :
50 22 A8 3C 00B3 356      MOVZWL  FAL$Q_RCVIOSB+2(R8),R0 ; Get message size
57 00C0 C8 DE 00B7 357      MOVAL    FAL$S_STB(R8),R7 ; Get address of statistics block
000000B3 358      INCL    FAL$S_RCV_PKT(R7) ; Increment RCV message packet count
10 A7 50 C0 00BE 359      ADDL2    R0,FAL$S_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
000000CE 363      RET ; Exit
50 20 A8 3C 00CF 364      MOVZWL  FAL$Q_RCVIOSB(R8),R0 ; Save error code
000000D3 365      $CHECK_SS ; Fatal error--terminate image
```



```
00D6 367 .SBTTL FAL$TRANSMIT - SEND DAP MESSAGE
000000D6 368 .PSECT FAL$CODE NOSHR,EXE,RD,NOWRT,BYTE
00D6 369
00D6 370 :++
00D6 371 : Functional Description:
00D6 372 :
00D6 373 : FAL$TRANSMIT writes the next DAP message to the partner process.
00D6 374 : DAP outbound message blocking is supported.
00D6 375 :
00D6 376 : Calling Sequence:
00D6 377 :
00D6 378 : BSBW FAL$TRANSMIT
00D6 379 :
00D6 380 : Input Parameters:
00D6 381 :
00D6 382 : R8 Address of FAL work area
00D6 383 : R9 Address of DAP control block
00D6 384 : R10 Address of FAB
00D6 385 : R11 Address of RAB
00D6 386 :
00D6 387 : Implicit Inputs:
00D6 388 :
00D6 389 : DAP$V_LENGTH
00D6 390 : DAP$V_MSGBLK
00D6 391 : FAL$Q_BLD
00D6 392 : FAL$Q_XMT
00D6 393 : FAL$W_DAPBUFSIZ
00D6 394 : FAL$V_LAST_MSG
00D6 395 : Flags field of the DAP message
00D6 396 :
00D6 397 : Output Parameters:
00D6 398 :
00D6 399 : R0 Completion code
00D6 400 : R1 Destroyed
00D6 401 :
00D6 402 : Implicit Outputs:
00D6 403 :
00D6 404 : FAL$Q_XMT
00D6 405 : FAL$V_LAST_MSG cleared
00D6 406 :
00D6 407 : Completion Codes:
00D6 408 :
00D6 409 : None
00D6 410 :
00D6 411 : Side Effects:
00D6 412 :
00D6 413 : FAL terminates execution on encountering a logical link QIO error.
00D6 414 :
00D6 415 : --
00D6 416 :
00D6 417 FAL$TRANSMIT:: ; Entry Point
3C BB 00D6 418 PUSHF #^M<R2,R3,R4,R5> ; Save registers
00D8 419
00D8 420 :
00D8 421 : Update counter and log the DAP message.
00D8 422 :
00D8 423 :
```



```
00D8 C8 D6 00D8 424 INCL FAL$ _STB+FAL$ _XMT_MSG(R8)
OF 68 22 E1 00DC 425
54 A8 DD 00E0 426 BBC #FAL$V LOG_MSG,(R8),10$ ; Count this message for logging
50 A8 DD 00E3 427 PUSHF FAL$Q_BLD+4(R8) ; Branch if logging disabled
0000'CF 9F 00E6 428 PUSHF FAL$Q_BLD(R8) ; Address of DAP message
0000'CF 03 FB 00EA 429 PUSHAB W^FAL$GT_ENCODE ; Size of DAP message
0000'CF 03 FB 00EA 430 CALLS #3,W^FAL$DISPLAY_MSG ; Address of counted ASCII string
00EF 431 ; Print the message
00EF 432
00EF 433 : Does partner support message blocking?
00EF 434
00EF 435
57 28 12 E1 00EF 436 10$: BBC #DAP$V MSGBLK,- ; Branch if message blocking is
00F1 437 DAP$Q_SYSCAP(R9),50$ ; disabled
00F4 438
00F4 439 : Are there any messages in XMT buffer waiting to go?
00F4 440
00F4 441
00F4 442
48 A8 D5 00F4 443 TSTL FAL$Q_XMT(R8) ; Branch if there are no blocked DAP
18 13 00F7 444 BEQL 20$ ; messages in buffer
00F9 445
00F9 446 : Will new message fit in XMT buffer with other messages?
00F9 447
00F9 448
00F9 449
50 50 A8 C1 00F9 450 ADDL3 FAL$Q_BLD(R8),- ; Compute total # bytes if new message
50 48 A8 00FC 451 FAL$Q_XMT(R8),R0 ; were added to transmit buffer
51 1A A8 3C 00FF 452 MOVZWL FAL$W_DAPBUFSIZ(R8),R1 ; Prepare for longword compare as size
51 50 D1 0103 453 ; in R0 may slightly exceed 64K
28 1B 0106 454 CMPL R0,R1 ; Branch if new message can be blocked
0108 455 BLEQU 30$ ; with the others
0108 456
0108 457 : Flush XMT buffer to make room for new message.
0108 458
0108 459
0108 460
54 48 A8 7D 0108 461 MOVQ FAL$Q_XMT(R8),R4 ; Put buffer descriptor in <R4,R5>
4A 10 010C 462 BSBB FAL$TRANSMIT_PKT ; Send the blocked messages
48 A8 D4 010E 463 : ***** BLBC R0,EXIT ; Branch on failure (no status returned)
0111 464 CLRL FAL$Q_XMT(R8) ; Initialize byte count in descriptor
0111 465
0111 466 : Determine whether new message should be sent now or deferred (buffered).
0111 467
0111 468
0111 469
35 54 09 E1 0111 470 20$: BBC #<DAP$V_LENGTH+8>,- ; Branch if this message does not allow
31 68 B8 0113 471 @FAL$Q_BLD+4(R8),50$ ; additional blocking (implicit flush)
18 E0 0116 472 BBS #FAL$V_LAST_MSG,(R8),50$ ; Branch if this is last message of
011A 473 ; sequence (explicit flush)
011A 474
011A 475 : Move new message to XMT buffer, or simply map message if BLD and XMT buffers
011A 476 : are aligned.
011A 477
011A 478
50 A8 D0 011A 479
011A 480 MOVL FAL$Q_BLD(R8),- ; Update byte count in descriptor
```



```
48 A8      011D  481      CMPL  FAL$Q_XMT(R8)      ;
54 A8      011F  482      FAL$Q_BLD+4(R8),-      ; Eliminate move of data if BLD and XMT
4C A8      0122  483      FAL$Q_XMT+4(R8)      ; buffer addresses coincide
2B      13  0124  484      SUC      ;
50 A8      0126  485      MOVCL  FAL$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$Q_XMT(R8)      ; Update byte count in descriptor
0137  496      ;
0137  497      ;
0137  498      ; Determine whether new message should be sent now or deferred (buffered).
0137  499      ;
0137  500      ;
04 54 B8    09  E1  0137  501      BBC   #<DAP$V_LENGTH+8>,-      ; Branch if this message does not allow
11 68      18  E1  0139  502      @FAL$Q_BLD+4(R8),40$      ; additional blocking (implicit flush)
013C  503      BBC   #FAL$V_LAST_MSG,(R8),SUC      ; Fall thru if this is last message of
0140  504      ; sequence (explicit flush)
0140  505      ;
0140  506      ;
0140  507      ; Send blocked messages from XMT buffer.
0140  508      ;
0140  509      ;
54 48 A8    7D  10  0140  510 40$:   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$Q_XMT(R8)      ; Initialize byte count in descriptor
06      11  0149  513      BRB      EXIT      ;
014B  514      ;
014B  515      ;
014B  516      ; Send new message from BLD buffer.
014B  517      ;
014B  518      ;
54 50 A8    7D  10  014B  519 50$:   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      05  0157  531      RSB      ; Exit
```



```
00000158 533      .SBTTL FAL$TRANSMIT_PKT - SEND DAP PACKET
0158      534      .PSECT FAL$CODE      NOSHR,EXE,RD,NOWRT,BYTE
0158      535
0158      536      ;++
0158      537      ; Functional Description:
0158      538
0158      539      ; FAL$TRANSMIT_PKT sends the specified packet of DAP messages to the
0158      540      ; partner process.
0158      541
0158      542      ; Calling Sequence:
0158      543
0158      544      ; BSBW      FAL$TRANSMIT_PKT
0158      545
0158      546      ; Input Parameters:
0158      547
0158      548      ; R4-R5      Descriptor of message packet to transmit
0158      549      ; R8        Address of FAL work area
0158      550      ; R9        Address of DAP control block
0158      551      ; R10       Address of FAB
0158      552      ; R11       Address of RAB
0158      553
0158      554      ; Implicit Inputs:
0158      555
0158      556      ; None
0158      557
0158      558      ; Output Parameters:
0158      559
0158      560      ; R0        Completion code
0158      561      ; R1        Destroyed
0158      562
0158      563      ; Implicit Outputs:
0158      564
0158      565      ; None
0158      566
0158      567      ; Completion Codes:
0158      568
0158      569      ; None
0158      570
0158      571      ; Side Effects:
0158      572
0158      573      ; FAL terminates execution on encountering a logical link QIO error.
0158      574
0158      575      ;--
0158      576
15 10 0158 577 FAL$TRANSMIT_PKT::
0158      578      BSBW      FAL$XMT_QIO      ; Entry Point
0158      579      $WAITFR_S-      ; Send packet to partner
0158      580      -EFN=#FAL$K_XMTEFN      ; Wait for I/O to complete
0158      581      $CHECK_SS      ; Check status code and exit on failure
0158      582      $CLRBIT #FAL$V_XMTQIO,(R8) ; Clear transmit QIO outstanding flag
0158      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
0173 636      BBC      #FAL$V_LOG_QIO,(R8),10$      ; Branch if logging disabled
0177 637      MOVAB  W^FAL$GT_XMTQIO,R1      ; Get address of counted string
017C 638      BSBW      FAL$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      ;
```

08 68 24 E1
51 0000*CF 9E
FE81* 30


```
017F 643
017F 644 10$: $QIO_S EFN=#FAL$K_XMTEFN- ; Issue write with AST
017F 645 CHAN=FAL$W_LNKCHN(R8)- ;
017F 646 FUNC=#IOS_WRITEVBLK- ;
017F 647 IOSB=FAL$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
05 01A0 652 $CHECK_SS ; Check status code and exit on failure
01A3 653 RSB ; Exit
```



```
000001A4 655      .SBTTL FAL$XMT_AST - LINK TRANSMIT AST
01A4 656      .PSECT FAL$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$V_XMTAST
01A4 682      ;
01A4 683      ; Completion Codes:
01A4 684      ;
01A4 685      ; None
01A4 686      ;
01A4 687      ; Side Effects:
01A4 688      ;
01A4 689      ; FAL terminates execution on encountering a logical link QIO error.
01A4 690      ;
01A4 691      ;--
01A4 692
58 04 AC 0180 01A4 693      .ENTRY FAL$XMT_AST,^M<R7,R8> ; Entry point from executive
1D 28 A8 E9 01A6 694      MOVL 4(AP),R8 ; Get address of FAL work area
01AA 695      $SETBIT #FAL$V_XMTAST,(R8) ; Signal transmit AST delivered
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$Q_XMTIOSB+2(R8),R0 ; Save # bytes transmitted
57 00C0 C8 DE 01B6 703      MOVAL FAL$S_STB(R8),R7 ; Get address of statistics block
14 A7 D6 01BB 704      INCL FAL$S_XMT_PKT(R7) ; Increment XMT message packet count
24 A7 50 C0 01BE 705      ADDL2 R0,FAL$S_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
```



```
000001D6 713      .SBTTL FALS$RECEIVE_MBX - READ AND PARSE INTERRUPT
01D6 714      .PSECT FALS$CODE      NOSHR,EXE,RD,NOWRT,BYTE
01D6 715
01D6 716      :++
01D6 717      : Functional Description:
01D6 718      :
01D6 719      : FALS$RECEIVE_MBX returns the next DAP message sent as an interrupt
01D6 720      : message by the partner process. It is returned as a decoded message
01D6 721      : via the DAP control block.
01D6 722      :
01D6 723      : Calling Sequence:
01D6 724      :
01D6 725      :     BSBW      FALS$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      :     FALS$Q_MBX
01D6 737      :
01D6 738      : Output Parameters:
01D6 739      :
01D6 740      :     R0-R1     Destroyed
01D6 741      :
01D6 742      : Implicit Outputs:
01D6 743      :
01D6 744      :     DAP control block is updated to reflect message parse.
01D6 745      :
01D6 746      : Completion Codes:
01D6 747      :
01D6 748      :     DAPS$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
0C 68 15 E0 01D6 756 FALS$RECEIVE_MBX::      : Entry Point
01D6 757      BBS-      #FALS$V_MBXAST,(R8),10$      : Branch if receive has completed
01DA 758      $WAITFR_S-      : Otherwise wait for it to complete
01DA 759      -EFN=#FALS$K_MBXEFN      :
01E3 760      $CHECK SS      : Check status code and exit on failure
01E6 761 10$:      $CLRBIT #FALS$V_MBXQIO,(R8)      : Clear mailbox QIO outstanding flag
01EA 762      $CLRBIT #FALS$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
      OF 68 22 E1 0200 789      BBC   #FAL$V_LOG_MSG,(R8),20$ ; Branch if logging disabled
      14 A9 DD 0204 790      PUSHL  DAP$Q_MSG_BUF2+4(R9) ; Address of DAP message just decoded
      10 A9 DD 0207 791      PUSHL  DAP$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
```

```
00000214 796      .SBTTL FAL$MBX_RCV_QIO - QUEUE MAILBOX RECEIVE
0214 797      .PSECT FAL$CODE      NOSHR,EXE,RD,NOWRT,BYTE
0214 798
0214 799      ;++
0214 800      ; Functional Description:
0214 801      ;
0214 802      ; FAL$MBX_RCV_QIO issues a QIO read request with an AST for the
0214 803      ; mailbox associated with the logical link.
0214 804
0214 805      ; Calling Sequence:
0214 806      ;
0214 807      ; BSBW      FAL$MBX_RCV_QIO
0214 808
0214 809      ; Input Parameters:
0214 810      ;
0214 811      ; R8      Address of FAL work area
0214 812      ; R9      Address of DAP control block
0214 813      ; R10     Address of FAB
0214 814      ; R11     Address of RAB
0214 815
0214 816      ; Implicit Inputs:
0214 817      ;
0214 818      ; None
0214 819
0214 820      ; Output Parameters:
0214 821      ;
0214 822      ; R0      Completion code
0214 823      ; R1      Destroyed
0214 824
0214 825      ; Implicit Outputs:
0214 826      ;
0214 827      ; None
0214 828
0214 829      ; Completion Codes:
0214 830      ;
0214 831      ; None
0214 832
0214 833      ; Side Effects:
0214 834      ;
0214 835      ; FAL terminates execution on encountering a mailbox read error.
0214 836
0214 837      ;--
0214 838
0214 839 FAL$MBX_RCV_QIO::      ; Entry Point
0214 840      $SETBIT #FAL$V_MBXQIO,(R8)      ; Signal mailbox QIO outstanding
0218 841
0218 842      ;
0218 843      ; Log our intent to post a mailbox receive QIO.
0218 844      ;
0218 845
0218 846      B5C      #FAL$V_LOG_QIO,(R8),10$ ; Branch if logging disabled
021C 847      MOVAB W^FAL$GT_MBXQIO,R1      ; Get address of counted string
0221 848      BSBW      FAL$LOG_QIO      ; Log QIO posted to print file
0224 849
0224 850      ;
0224 851      ; Issue a read with an AST for the mailbox.
0224 852      ;
```

08 68 24 E1
51 0000'CF 9E
FDDC' 30


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

```
0000024F 865      .SBTTL FAL$MBX_RCV_AST - MAILBOX RECEIVE AST
024F 866      .PSECT FAL$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
024F 911      :.ENTRY FAL$MBX_RCV_AST,^M<R7,R8> ; Entry point from executive
58 04 AC 0180 0251 912      MOVL 4(AP),R8 ; Get address of FAL work area
07 30 A8 E8 0255 913      $SETBIT #FAL$V_MBXAST,(R8) ; Signal mailbox AST delivered
50 20 A8 3C 0259 914      BLBS FAL$Q_MBXIOSB(R8),10$ ; Branch on success
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$: BBC #FALS$ LOG MSG (R8),20$ ; Branch if logging disabled
0268 923 $FAO_S CTRSTR=W^FALS$Q_MBXMSG- ; Format the message
0268 924 OUTLEN=W^FALS$Q_PRTLEN2- ;
0268 925 OUTBUF=W^FALS$Q_PRTBUF2- ;
0268 926 P1=FALS$T_MBXBUF(R8) ; First four bytes of mailbox message
027F 927 $CHECK_STATUS ; Check status code
FD7B' 30 0282 928 BSBW FALS$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$: MOVAB FALS$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 FALS$TERMINATE- ; Disconnect abort
028A 953 IGNORE- ; Connect confirm
028A 954 IGNORE- ; Connect initiate
028A 955 FALS$TERMINATE- ; Synchronous disconnect
028A 956 FALS$TERMINATE- ; Partner exited prematurely
028A 957 INTERRUPT- ; Interrupt message
028A 958 FALS$TERMINATE- ; Partner no longer accessible
028A 959 FALS$TERMINATE- ; NSP protocol error
028A 960 IGNORE- ; Connect reject
028A 961 FALS$TERMINATE- ; Thirdparty disconnect
028A 962 IGNORE- ; Connect timeout
028A 963 IGNORE- ; Network shutting down
028A 964 > ;
02A6 965 ;
02A6 966 ; Ignore unexpected mailbox message.
02A6 967 ;
02A6 968 ;
02A6 969 ;
02A6 970 IGNORE: ; Discard junk mail
02A6 971 CLRL R0 ; Make this a null message
38 A8 50 D4 02A8 972 MOVQ R0,FALS$Q_MBX(R8) ; Store descriptor of interrupt data
50 7D 04 02AC 973 RET ; Exit
02AD 974 ;
02AD 975 ;
02AD 976 ; An interrupt message has been received.
02AD 977 ;
02AD 978 ;
```

```

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


FALDPIO
Symbol table

- DAP MESSAGE I/O ROUTINES

L 2

16-SEP-1984 01:41:37 VAX/VMS Macro V04-00
5-SEP-1984 01:16:43 [FAL.SRC]FALDPIO.MAR;1

Page 25
(13)

\$\$COUNT	= 0000000C	FAL\$C_WRKBLN	00002000		
\$\$T1	= 00000000	FAL\$DECODE_MSG	*****	X	02
\$\$T2	= 00000004	FAL\$DISPLAY_MSG	*****	X	02
DAP\$B_BITCNT	00000035	FAL\$GL_COUNTER2	*****	X	02
DAP\$B_DCODE_FID	00000019	FAL\$GL_RECVWAIT	*****	X	02
DAP\$B_DCODE_MAC	0000001B	FAL\$GQ_MBXMSG	*****	X	02
DAP\$B_DCODE_MSG	0000001A	FAL\$GQ_PRTBUF2	*****	X	02
DAP\$B_DECVER	00000047	FAL\$GT_DECODE	*****	X	02
DAP\$B_ECONUM	00000045	FAL\$GT_ENCODE	*****	X	02
DAP\$B_FILESYS	00000043	FAL\$GT_MBXQIO	*****	X	02
DAP\$B_FLAGS	00000031	FAL\$GT_RCVQIO	*****	X	02
DAP\$B_LEN256	00000034	FAL\$GT_XMTQIO	*****	X	02
DAP\$B_LENGTH	00000033	FAL\$GW_PRTLEN2	*****	X	02
DAP\$B_OSTYPE	00000042	FAL\$K_MBXBUF	= 00000040		
DAP\$B_STREAMID	00000032	FAL\$K_MBXEFN	= 00000003		
DAP\$B_TYPE	00000030	FAL\$K_RCVBUF CNT	= 00000002		
DAP\$B_USRNUM	00000046	FAL\$K_RCVEFN	= 00000001		
DAP\$B_USRVER	00000048	FAL\$K_STBBLN	00000040		
DAP\$B_VERNUM	00000044	FAL\$K_WRKBLN	00002000		
DAP\$B_X_FIELD	00000024	FAL\$K_XMTEFN	= 00000002		
DAP\$C_BCN	000000C0	FAL\$LOG_AST	*****	X	02
DAP\$K_BLN	000000C0	FAL\$LOG_QIO	*****	X	02
DAP\$L_CMWA	00000030	FAL\$L_ACLXAB	000000C0		
DAP\$L_CRC_RSLT	00000020	FAL\$L_ALLXABINI	00000074		
DAP\$L_DCODE_STS	00000018	FAL\$L_CHAIN_NXT	0000007C		
DAP\$L_MSG_MASK	0000001C	FAL\$L_DATXAB	00000320		
DAP\$L_SSPWA	00000080	FAL\$L_FAB	00000200		
DAP\$L_TEMP	00000090	FAL\$L_FAB2	00000800		
DAP\$M_BITCNT	= 00000008	FAL\$L_FHCXAB	000002F4		
DAP\$M_SEGMENT	= 00000040	FAL\$L_FOP	000001F8		
DAP\$M_TMP1\$	= 00000010	FAL\$L_KEYNAM	00001C00		
DAP\$M_TMP2\$	= 00000080	FAL\$L_KEYXAB	00001000		
DAP\$Q_DCODE_FLG	00000000	FAL\$L_KEYXABINI	00000078		
DAP\$Q_MSG_BUF1	00000008	FAL\$L_NAM	00000294		
DAP\$Q_MSG_BUF2	00000010	FAL\$L_NAM2	00000850		
DAP\$Q_SYSCAP	00000028	FAL\$L_NUMBER	000001FC		
DAP\$Q_SYSPEC	00000038	FAL\$L_PROXAB	0000034C		
DAP\$V_LENGTH	= 00000001	FAL\$L_RAB	00000250		
DAP\$V_MSGBLK	= 00000012	FAL\$L_RCVBUF	0000005C		
DAP\$W_BUFSIZ	00000040	FAL\$L_RCV_DAT	00000008		
DAP\$W_PARTNER	00000006	FAL\$L_RCV_LNK	00000010		
DAP\$W_VERSION	00000004	FAL\$L_RCV_MSG	00000004		
EXIT	00000151 R	FAL\$L_RCV_PKT	00000000		
FAL\$B_ACCFUNC	000001F6	FAL\$L_RCV_USR	0000000C		
FAL\$B_ACCOPT	000001F5	FAL\$L_RDTXAB	000003B0		
FAL\$B_DATATYPE	000001F4	FAL\$L_RMS_PTR	0000006C		
FAL\$B_DISABLE	00000006	FAL\$L_STB	000000C0		
FAL\$B_ENABLE	00000005	FAL\$L_SUMXAB	000003A4		
FAL\$B_LOGGING	00000004	FAL\$L_TEMP	000003F4		
FAL\$B_MISCOPT	00000007	FAL\$L_USE_SC1	000000A8		
FAL\$B_RAC	000001F7	FAL\$L_USE_SC2	000000AC		
FAL\$B_RBK_CACHE	00000012	FAL\$L_USE_VER	000000A4		
FAL\$B_RCVBUFIDX	00000011	FAL\$L_XMT_DAT	0000001C		
FAL\$B_VALUE	00000010	FAL\$L_XMT_LNK	00000024		
FAL\$CHECK_SS	***** X	FAL\$L_XMT_MSG	00000018		
FAL\$CHECK_STATUS	***** X	FAL\$L_XMT_PKT	00000014		
FAL\$C_STBBLN	00000040	FAL\$L_XMT_USR	00000020		

FALDAPIO
Symbol table

- DAP MESSAGE I/O ROUTINES

M 2

16-SEP-1984 01:41:37 VAX/VMS Macro V04-00
5-SEP-1984 01:16:43 [FAL.SRC]FALDAPIO.MAR;1

Page 26
(13)

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
IOS_READVBLK	= 00000031		
IOS_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
SYSS\$FAO	*****	X	02
SYSS\$QIO	*****	GX	02
SYSS\$WAITFR	*****	GX	02

FA
VO

+-----+
! 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
\$255\$DUA28:[FAL.OBJ]FAL.MLB;1	11
\$255\$DUA28:[SYSLIB]STARLET.MLB;2	12
TOTALS (all libraries)	23

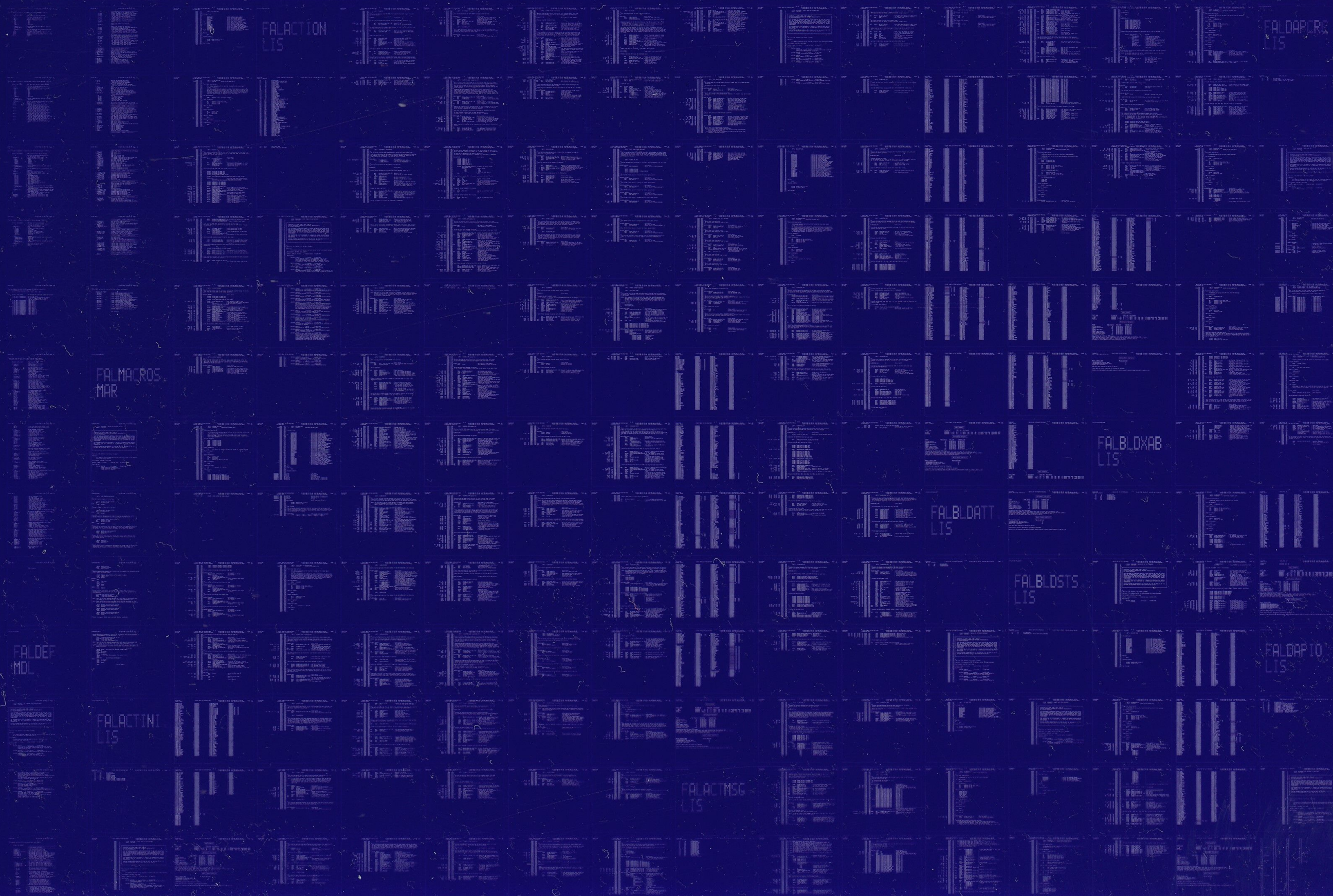
1125 GETS were required to define 23 macros.

There were no errors, warnings or information messages.

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

0174 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



0175 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

