```
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
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**FALMAIN

```
FFFFFFFFFF    AAAAAA    LL          MM      MM    AAAAAA     IIIIII    NN      NN
FFFFFFFFFF    AAAAAA    LL          MM      MM    AAAAAA     IIIIII    NN      NN
FF           AA    AA   LL          MMMM  MMMM   AA    AA     II      NN      NN
FF           AA    AA   LL          MMMM  MMMM   AA    AA     II      NN      NN
FF           AA    AA   LL          MM  MM  MM   AA    AA     II      NNNN    NN
FF           AA    AA   LL          MM  MM  MM   AA    AA     II      NNNN    NN
FFFFFFFF     AA    AA   LL          MM      MM   AA    AA     II      NN  NN  NN
FFFFFFFF     AA    AA   LL          MM      MM   AA    AA     II      NN  NN  NN
FF           AAAAAAAAAA LL          MM      MM   AAAAAAAAAA     II      NN    NNNN
FF           AAAAAAAAAA LL          MM      MM   AAAAAAAAAA     II      NN    NNNN
FF           AA    AA   LL          MM      MM   AA    AA     II      NN      NN      ....
FF           AA    AA   LL          MM      MM   AA    AA     II      NN      NN      ....
FF           AA    AA   LLLLLLLLLL  MM      MM   AA    AA     IIIIII  NN      NN      ....
FF           AA    AA   LLLLLLLLLL  MM      MM   AA    AA     IIIIII  NN      NN      ....

LL             IIIIII      SSSSSSSS
LL             IIIIII      SSSSSSSS
LL               II      SS
LL               II      SS
LL               II      SS
LL               II      SS
LL               II        SSSSSS
LL               II        SSSSSS
LL               II              SS
LL               II              SS
LL               II              SS
LL               II              SS
LLLLLLLLLL     IIIIII    SSSSSSSS
LLLLLLLLLL     IIIIII    SSSSSSSS
```

Table of contents

```
0000     1          .TITLE  FALMAIN - FAL MAINLINE
0000     2          .IDENT  'V04-000'                ; Also change FAL$GT_VERSION!!!
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  ;       FAL is the DECnet-VAX file access server program, known as DECnet
0000    35  ;       object type 17 (decimal). Its purpose is to provide access to files
0000    36  ;       and unit record devices on a VMS node on behalf of processes executing
0000    37  ;       on any node of the DECnet communications network. FAL uses the Data
0000    38  ;       Access Protocol (DAP) to communicate with the requesting process and it
0000    39  ;       accesses the specified file (or device) through RMS calls.
0000    40  ;
0000    41  ; Environment:
0000    42  ;
0000    43  ;       The FAL image executes in the context of a process created by NETACP.
0000    44  ;       It runs in user mode and requires NETMBX and TMPMBX privilege.
0000    45  ;
0000    46  ; Author: James A. Krycka,      Creation Date:  16-JUN-1977
0000    47  ;
0000    48  ; Modified By:
0000    49  ;
0000    50  ;       V03-016 JAK0146         J A Krycka      31-AUG-1984
0000    51  ;               Update FAL$GT_VERSION for FAL logging.
0000    52  ;
0000    53  ;       V03-015 JAK0145         J A Krycka      12-APR-1984
0000    54  ;               Track changes in DAP message building algorithm.
0000    55  ;
0000    56  ;       V03-014 JAK0140         J A Krycka      03-APR-1984
0000    57  ;               Remove buffered I/O byte limit quota (BYTLM) as the basis for
```

```
0000     58 ;          determining the largest DAP buffer size that FAL is willing to
0000     59 ;          support.
0000     60 ;
0000     61 ;      V03-013 JAK0137         J A Krycka      12-MAR-1984
0000     62 ;          Continuation of V03-012 to support qualifier options in FAL$LOG
0000     63 ;          string to control RMS multi-buffer cache size, DAP buffer size,
0000     64 ;          and to alter Configuration message information to aid testing.
0000     65 ;          Also, conditionally translate FAL$OUTPUT to obtain user provided
0000     66 ;          output file specification.
0000     67 ;          Modifications to reflect macro name changes in FALMACROS.MAR.
0000     68 ;
0000     69 ;      V03-012 JAK0136         J A Krycka      07-MAR-1984
0000     70 ;          Rename CHECK_RMS to $CHECK_STATUS and revise it so that FAL does
0000     71 ;          not terminate on error during output of FAL logging information.
0000     72 ;          Modifications to reflect changes made to layout of $FALWRKDEF.
0000     73 ;
0000     74 ;      V03-011 JAK0129         J A Krycka      11-JAN-1984
0000     75 ;      V03-010 JAK0124         J A Krycka      06-SEP-1983
0000     76 ;      V03-009 JAK0118         J A Krycka      29-JUL-1983
0000     77 ;      V03-008 JAK0113         J A Krycka      22-JUN-1983
0000     78 ;      V03-007 JAK0107         J A Krycka      30-APR-1983
0000     79 ;      V03-006 JAK0105         J A Krycka      29-APR-1983
0000     80 ;          Update FAL$GT_VERSION for FAL logging.
0000     81 ;
0000     82 ;      V03-005 JAK0104         J A Krycka      06-APR-1983
0000     83 ;          Allocate RMS block I/O buffers for use during file transfer
0000     84 ;          mode.
0000     85 ;
0000     86 ;      V03-004 KRM0086         K Malik         23-Mar-1983
0000     87 ;          Update FAL$GT_VERSION for support of STMLF and STMCR file
0000     88 ;          formats and definition and setting of DAP$V_GEQ_V70 bit.
0000     89 ;
0000     90 ;      V03-003 KRM0072         K Malik         23-Nov-1982
0000     91 ;          Change FAL$B_LOGFLG to FAL$L_LOGFLG.
0000     92 ;
0000     93 ;      V03-002 JAK0101         J A Krycka      09-OCT-1982
0000     94 ;          Deduct extra buffered I/O quota before calculating DAP
0000     95 ;          buffer size to prevent FAL from exceeding its BYTLM quota
0000     96 ;          and thereby causing it to hang. This problem only occurs
0000     97 ;          if the requestor agrees to use FAL's buffer size and the
0000     98 ;          FAL process has a sufficiently low BYTLM quota.
0000     99 ;
0000    100 ;--
```

```
                        0000    102          .SBTTL  DECLARATIONS
                    00000000    103          .PSECT  FAL$DATA        SHR,NOEXE,RD,WRT,LONG
                        0000    104
                        0000    105 ;
                        0000    106 ; Include Files:
                        0000    107 ;
                        0000    108
                        0000    109          $DAPCNFDEF                     ; Define DAP Configuration message
                        0000    110          $FALWRKDEF                     ; Define FAL Work Area symbols
                        0000    111          $IODEF                         ; Define QIO function codes
                        0000    112          $SSDEF                         ; Define System Service status codes
                        0000    113
                        0000    114 ;
                        0000    115 ; Macros:
                        0000    116 ;
                        0000    117 ;        None
                        0000    118 ;
                        0000    119 ; Equated Symbols:
                        0000    120 ;
                        0000    121
                        0000    122          ASSUME  FAL$Q_FLG EQ 0
                        0000    123
                        0000    124 ;
                        0000    125 ; Own Storage:
                        0000    126 ;
                        0000    127 ; Text stored as counted ASCII strings:
                        0000    128 ;
                        0000    129
                        0000    130 FAL$GT_VERSION::                        ; FAL version number string used by
30 30 30 2D 34 30 56 00' 0000   131          .ASCIC  \V04-000\              ;  FALLOGGER--same as module ident #
                      07 0000
                        0008    132
                        0008    133 ;
                        0008    134 ; Storage of status code to report on image exit:
                        0008    135 ;
                        0008    136
                        0008    137 FAL$GL_EXIT::                           ; Status code
00000001                0008    138          .LONG   SS$_NORMAL             ; Initialize with success code
                        000C    139
                        000C    140 ;
                        000C    141 ; Device name and logical name descriptor blocks with text:
                        000C    142 ;
                        000C    143
                        000C    144 FAL$GQ_LNKNAM::                         ; Device name descriptor block
                        000C    145          $QBLOCK TEXT=<_NET0:>          ;  for the link
                        001A    146 FAL$GQ_SYSNAM::                         ; Logical name descriptor block
                        001A    147          $QBLOCK TEXT=<SYS$NET>         ;  for SYS$NET
                        0029    148 FAL$GQ_LOGNAM::                         ; Logical name descriptor block
                        0029    149          $QBLOCK TEXT=<FAL$LOG>         ;  for FAL$LOG
                        0038    150 FAL$GQ_OUTPUT::                         ; Logical name descriptor block
                        0038    151          $QBLOCK TEXT=<FAL$OUTPUT>      ;  for FAL$OUTPUT
                        004A    152 FAL$GQ_WILDSPEC::                       ; Wildcard file name, type, and version
                        004A    153          $QBLOCK TEXT=<<*.*;*>>         ;  (default) file specification string
```

```
                        .SBTTL  FAL$START - MAINLINE
00000000                .PSECT  FAL$CODE        NOSHR,EXE,RD,NOWRT,BYTE

                ;++
                ; Functional Description:
                ;
                ;       FAL$START is responsible for the initialization and termination of FAL.
                ;       These activities include:
                ;
                ;       (1) allocation of the FAL work area and several control blocks within
                ;       (2) translation of FAL$LOG, conditional translation of FAL$OUTPUT, and
                ;           conditional creation of a file for logging requested information
                ;       (3) creation of a control/information path to NETACP via NET0
                ;       (4) translation of SYS$NET and construction of an NCB
                ;       (5) completion of the logical link and optional logging startup data
                ;       (6) allocation of transmit and receive buffers
                ;       (7) posting reads and transferring control to the state table manager
                ;       (8) terminating the image and exiting to VMS
                ;
                ;       Once a logical link has been established between FAL and its requestor,
                ;       control is passed to FAL$STATE, the state transition table manager,
                ;       which controls the operation of FAL.
                ;
                ; Calling Sequence:
                ;
                ;       Normally, FAL is run from SYS$SYSTEM:FAL.COM, a batch procedure that
                ;       is executed when an inbound connect request for object type 17 (decimal)
                ;       is received by this node.
                ;
                ; Input Parameters:
                ;
                ;       None
                ;
                ; Implicit Inputs:
                ;
                ;       Logical name 'FAL$LOG'
                ;       Logical name 'FAL$OUTPUT'
                ;       Logical name 'SYS$NET'
                ;
                ; Output Parameters:
                ;
                ;       None
                ;
                ; Implicit Outputs:
                ;
                ;       None
                ;
                ; Completion Codes:
                ;
                ;       Standard RMS completion codes
                ;
                ; Side Effects:
                ;
                ;       None
                ;
                ;--
```

```
                 0000  0000   212          .ENTRY  FAL$START,^M<>          ; Entry point from executive
                       0002   213
                       0002   214 ;+
                       0002   215 ; Allocate space for the FAL work area (defined by $FALWRKDEF). Within this
                       0002   216 ; work area space is reserved for several other structures. These include the
                       0002   217 ; DAP, STB, FAB, RAB, NAM, and XAB control blocks.
                       0002   218 ;-
                       0002   219
      51    10   D0    0002   220          MOVL    #<FAL$K_WRKBLN+511/512>,R1 ; Request required # pages
         03A6   30    0005   221          BSBW    FAL$GETPAGE                ; Expand program region
      58    52   D0    0008   222          MOVL    R2,R8                      ; Get address of FAL work area
59    0100 C8   9E    000B   223          MOVAB   FAL$T_DAP(R8),R9           ; Get address of DAP control block
5A    0200 C8   DE    0010   224          MOVAL   FAL$L_FAB(R8),R10          ; Get address of FAB
5B    0250 C8   DE    0015   225          MOVAL   FAL$L_RAB(R8),R11          ; Get address of RAB
         68   7C    001A   226          CLRQ    FAL$Q_FLG(R8)              ; Initialize FAL work area status flags
        40 8F   90    001C   227          MOVB    #FAL$K_DFLT_RBK,-          ; Initialize RMS multi-block cache size
        12 A8         001F   228                  FAL$B_RBK_CACHE(R8)        ;  used only for block mode transfers
                      0021   229                                             ;  (FAL$LOG may override this value)
```

```
                     0021   231 ;+
                     0021   232 ; Translate the logical name FAL$LOG to determine what type of information
                     0021   233 ; (if any) is to be logged in the print file. Also, other debugging options
                     0021   234 ; and program control options may be specified via this mechanism.
                     0021   235 ;
                     0021   236 ; Note that the definition of FAL$LOG as a logical name is strictly optional.
                     0021   237 ;-
                     0021   238
                     0021   239 TRANSLATE_FAL$LOG:                          ; Continuation of mainline
    57   0090 C8   7E 0021   240         MOVAQ   FAL$Q_FALLOG(R8),R7         ; Get address of descriptor
    67   00FF 8F   3C 0026   241         MOVZWL  #FAL$K_FALLOG,(R7)          ; Store buffer size
04  A7   1C00 C8   9E 002B   242         MOVAB   FAL$T_FALLOG(R8),4(R7)      ; Store buffer address
                     0031   243         $TRNLOG_S-                          ; Translate 'FAL$LOG'
                     0031   244                 LOGNAM=W^FAL$GQ_LOGNAM-     ;  Address of logical name descriptor
                     0031   245                 RSLLEN=(R7)-                ;  Update size directly in descriptor
                     0031   246                 RSLBUF=(R7)                 ;  Put equivalence string in buffer
  0629 8F    50   B1 0046   247         CMPW    R0,#<SS$_NOTRAN&^XFFFF>     ; Did logical name translate?
               0B   13 004B   248         BEQL    10$                         ; Branch if it was not defined
                     004D   249         $CHECK_STATUS                       ; Check status code
           FFAD'   30 0050   250         BSBW    FAL$PARSE_FAL$LOG           ; Process FAL$LOG options
        04   A8   95 0053   251         TSTB    FAL$B_LOGGING(R8)           ; Were any output options selected?
               03   12 0056   252         BNEQ    TRANSLATE_FAL$OUTPUT        ; Yes, determine where to log output
             0078   31 0058   253 10$:    BRW     ASSIGN_CHANNEL              ; No, skip remaining log initialization
```

```
                     005B   255 ;+
                     005B   256 ; Translate the logical name FAL$OUTPUT to determine if an explicit print file
                     005B   257 ; specification has been provided. The logical name translation is performed
                     005B   258 ; only if FAL logging options have been requested that generate log information.
                     005B   259 ;
                     005B   260 ; Note that the definition of FAL$OUTPUT as a logical name is strictly optional.
                     005B   261 ;-
                     005B   262
                     005B   263 TRANSLATE_FAL$OUTPUT:                        ; Conditionally translate logical name
    57   03F8 C8   7E 005B   264         MOVAQ   FAL$Q_TEMP(R8),R7           ; Get address of descriptor
    67   00FF 8F   3C 0060   265         MOVZWL  #FAL$R_FILESPEC,(R7)        ; Store buffer size
04  A7   0400 C8   9E 0065   266         MOVAB   FAL$T_FILESPEC(R8),4(R7)    ; Store buffer address
                     006B   267         $TRNLOG_S-                          ; Translate 'FAL$OUTPUT'
                     006B   268                 LOGNAM=W^FAL$GQ_OUTPUT-     ;  Address of logical name descriptor
                     006B   269                 RSLLEN=(R7)-                ;  Update size directly in descriptor
                     006B   270                 RSLBUF=(R7)                 ;  Put equivalence string in buffer
  0629 8F    50   B1 0080   271         CMPW    R0,#<SS$_NOTRAN&^XFFFF>     ; Did logical name translate?
               17   13 0085   272         BEQL    CREATE_PRINT_FILE           ; Branch if it was not defined
                     0087   273         $CHECK_STATUS                       ; Check status code
         03F8 C8   95 008A   274         TSTB    FAL$Q_TEMP(R8)              ; Branch if translation yielded a null
               0E   13 008E   275         BEQL    CREATE_PRINT_FILE           ;  string
         03F8 C8   90 0090   276         MOVB    FAL$Q_TEMP(R8),-            ; Update primary file specification
         0000'CF            0094   277                 W^FAL$PRTFAB+FAB$B_FNS ;  name size and string address in
         03FC C8   D0 0097   278         MOVL    FAL$Q_TEMP+4(R8),-          ;  print file FAB overriding default
         0000'CF            009B   279                 W^FAL$PRTFAB+FAB$L_FNA ;  string of 'SYS$OUTPUT
```

```
                        009E   281 ;+
                        009E   282 ; Create the print file and initialize print buffer descriptors if the display
                        009E   283 ; bitmask parameter in the equivalence string for FAL$LOG was non-zero.
                        009E   284 ;
                        009E   285 ; Note: Usually, SYS$OUTPUT will refer to the network log file (NETSERVER.LOG).
                        009E   286 ;       However, the user may direct logging output to an explicit log file
                        009E   287 ;       via the logical name FAL$OUTPUT; or he may have redefined SYS$OUTPUT
                        009E   288 ;       to point to a file.
                        009E   289 ;-
                        009E   290
                        009E   291 CREATE_PRINT_FILE:                          ; Continuation of mainline
                        009E   292         $CREATE FAB=W^FAL$PRTFAB            ; Create the file
                        00A9   293         $CHECK_STATUS                       ; Check completion code
                        00AC   294         $CONNECT RAB=W^FAL$PRTRAB           ; Establish record stream
                        00B7   295         $CHECK_STATUS                       ; Check completion code
                        00BA   296
                        00BA   297 ;
                        00BA   298 ; Initialize print buffer descriptors used by both FAL system service calls
                        00BA   299 ; to fill the buffers and subsequent RMS calls to write the buffers.
                        00BA   300 ;
                        00BA   301
51  0000'CF   7E        00BA   302         MOVAQ   W^FAL$GQ_PRTBUF1,R1         ; Get address of first descriptor
81  0100 8F   3C        00BF   303         MOVZWL  #FAL$K_PRTBUF,(R1)+         ; Initialize descriptor of primary
81  1A00 C8   9E        00C4   304         MOVAB   FAL$T_PRTBUF1(R8),(R1)+     ;  (non-AST-level) print buffer
81  0100 8F   3C        00C9   305         MOVZWL  #FAL$K_PRTBUF,(R1)+         ; Initialize descriptor of secondary
81  1B00 C8   9E        00CE   306         MOVAB   FAL$T_PRTBUF2(R8),(R1)+     ;  (AST-level) print buffer
```

```
                        00D3   308 ;+
                        00D3   309 ; Create a temporary mailbox and assign a channel to it. Then assign a channel
                        00D3   310 ; to the pseudo device _NET0 and associate the temporary mailbox with this
                        00D3   311 ; channel.
                        00D3   312 ;
                        00D3   313 ; Note: Assigning a channel to _NET0 does not confirm the logical link--it
                        00D3   314 ;       simply creates a control/information path to NETACP in preparation for
                        00D3   315 ;       non-transparent network I/O. A QIO access function must be issued to
                        00D3   316 ;       complete the logical link.
                        00D3   317 ;-
                        00D3   318
                        00D3   319 ASSIGN_CHANNEL:                                  ; Continuation of mainline
    51    03F8 C8   9E  00D3   320         MOVAB   FAL$Q_TEMP(R8),R1                ; Get address of scratch area
    61    0040 8F   3C  00D8   321         MOVZWL  #FAL$K_MBXBUF,(R1)               ; Store mailbox buffer size parameter
 04 A1    0080 8F   3C  00DD   322         MOVZWL  #FAL$K_MBXQUOTA,4(R1)            ; Store mailbox buffer quota parameter
            1E A8   3F  00E3   323         PUSHAW  FAL$W_MBXCHN(R8)                 ; Address to return mailbox channel #
            1C A8   3F  00E6   324         PUSHAW  FAL$W_LNKCHN(R8)                 ; Address to return device channel #
            04 A1   DF  00E9   325         PUSHAL  4(R1)                            ; Address of mailbox buffer quota
               61   DF  00EC   326         PUSHAL  (R1)                             ; Address of mailbox buffer size
        000C'CF     7F  00EE   327         PUSHAQ  W^FAL$GQ_LNKNAM                  ; Address of device name descriptor
00000000'GF    05   FB  00F2   328         CALLS   #5,G^LIB$ASN_WTH_MBX             ; Assign a channel to _NET0 and
                        00F9   329                                                  ;  associate a temporary mailbox with it
                        00F9   330         $CHECK_SS                                ; Check status code and exit on failure
```

```
                        00FC    332 ;+
                        00FC    333 ; Translate the logical name 'SYS$NET'. It's equivalence string is defined by
                        00FC    334 ; VMS to be the entire network connect block (NCB) which is required to gain
                        00FC    335 ; non-transparent access to the network.
                        00FC    336 ;
                        00FC    337 ; The general format of the equivalence string is:
                        00FC    338 ;
                        00FC    339 ;       nodespec::"objecttype=taskid/{binary_data}"
                        00FC    340 ;
                        00FC    341 ; However, the string displayed in the print file will have the binary data
                        00FC    342 ; truncated as shown below:
                        00FC    343 ;
                        00FC    344 ;       nodespec::"objecttype=taskid"
                        00FC    345 ;
                        00FC    346 ; Finally, the string put into the NCB for a connect accept without userdata
                        00FC    347 ; will include only the first two bytes of the binary data past the slash:
                        00FC    348 ;
                        00FC    349 ;       nodespec::"objecttype=taskid/{two_bytes_of_binary_data}"
                        00FC    350 ;-
                        00FC    351
                        00FC    352 TRANS_SYS$NET:                              ; Continuation of mainline
    57  0098 C8  7E     00FC    353         MOVAQ   FAL$Q_SYSNET(R8),R7         ; Get address of output descriptor
    67  00FF 8F  3C     0101    354         MOVZWL  #FAL$R_SYSNET,(R7)          ; Store buffer size
04  A7  1D00 C8  9E     0106    355         MOVAB   FAL$T_SYSNET(R8),4(R7)      ; Store buffer address
                        010C    356         $TRNLOG_S-                          ; Translate 'SYS$NET'
                        010C    357                 LOGNAM=W^FAL$GQ_SYSNAM-     ;  Address of logical name descriptor
                        010C    358                 RSLLEN=(R7)-                ;  Update size directly in descriptor
                        010C    359                 RSLBUF=(R7)                 ;  Put equivalence string in buffer
  0629  8F    50  B1    0121    360         CMPW    R0,#<SS$_NOTRAN&^XFFFF>     ; Branch if logical name is not defined
                2B  13    0126    361         BEQL    20$                         ;
                        0128    362         $CHECK_SS                           ; Check status code and exit on failure
                        012B    363
                        012B    364 ;
                        012B    365 ; Extract the name of the partner node and save it for use by FAL$CRC_LOGERR.
                        012B    366 ;
                        012B    367
        52  67  7D      012B    368         MOVQ    (R7),R2                     ; Make writable copy of descriptor
    63  52  3A  3A       012E    369         LOCC    #^A/:/,R2,(R3)              ; Find the colon
    52  51  53  C3       0132    370         SUBL3   R3,R1,R2                    ; Compute the nodename length
      63    5F  8F  91    0136    371         CMPB    #^A/_/,(R3)                 ; Nodename start with an underscore?
                04  12    013A    372         BNEQ    10$                         ; No
                52  D7    013C    373         DECL    R2                          ; Yes, lower the count
                53  D6    013E    374         INCL    R3                          ; and bump pointer past underscore
    54  0000'CF  9E       0140    375 10$:    MOVAB   W^FAL$GT_NODENAME,R4        ; Get address of buffer to put nodename
        84  52  90         0145    376         MOVB    R2,(R4)+                    ; Move the count into the buffer
    64  63  52  28         0148    377         MOVC3   R2,(R3),(R4)                ; Put the nodename in buffer
                        014C    378
                        014C    379 ;
                        014C    380 ; Locate the slash preceeding the binary data and save its address.
                        014C    381 ;
                        014C    382
  04 B7  67  2F  3A       014C    383         LOCC    #^A\/\,(R7),@4(R7)          ; Find taskid delimiter
                0A  12    0151    384         BNEQ    30$                         ; Branch if slash found
0008'CF  0154  8F  3C     0153    385 20$:    MOVZWL  #SS$_IVLOGNAM,W^FAL$GL_EXIT ; Declare error
            0155  31      015A    386         BRW     FAL$TERMINATE               ; Terminate image execution
        52  51  D0         015D    387 30$:    MOVL    R1,R2                       ; Save address of slash
                        0160    388
```

```
                                0160   389 ;
                                0160   390 ; Write to the print file: FAL version number, a time stamp, the identity of
                                0160   391 ; the requesting process, and the equivalence string for FAL$LOG.
                                0160   392 ;
                                0160   393
                     04 A8   95 0160   394         TSTB    FAL$B_LOGGING(R8)       ; Branch if FAL logging is disabled
                        6C   13 0163   395         BEQL    COMPLETE_LOGICAL_LINK   ;
                  81    22   90 0165   396         MOVB    #^A\"\,(R1)+            ; Overlay slash to truncate string
              67  51    04 A7   C3 0168   397         SUBL3   4(R7),R1,(R7)           ; Update string size in descriptor
                 50   0090 C8   7E 016D   398         MOVAQ   FAL$Q_FALLOG(R8),R0     ; Get address of string descriptor
                                0172   399         $FAO_S-                         ; Format the message
                                0172   400                 CTRSTR=W^FAL$GQ_CALLER- ;  Address of control string
                                0172   401                 OUTLEN=W^FAL$GW_PRTLEN1-;  Address of receive string length
                                0172   402                 OUTBUF=W^FAL$GQ_PRTBUF1-;  Address of buffer to put string
                                0172   403                 P1=#FAL$GT_VERSION-     ;  FAL version number string address
                                0172   404                 P2=#0-                  ;  Use current date and time
                                0172   405                 P3=R7-                  ;  Address of requestor string desc
                                0172   406                 P4=R0                   ;  Address of FAL$LOG string descriptor
                                0191   407         $CHECK_STATUS                   ; Check status code
                     FE69'   30 0194   408         BSBW    FAL$PRINT_FAO           ; Print message
                                0197   409
                                0197   410 ;
                                0197   411 ; Write an error message if FAL$LOG options string was not successfully parsed.
                                0197   412 ;
                                0197   413
                  19 68   38   E1 0197   414         BBC     #FAL$V_PARSE_ERR,(R8),40$;Branch if no parse error was detected
                                019B   415         $FAO_S-                         ; Format the message
                                019B   416                 CTRSTR=W^FAL$GQ_PARSERR-;  Address of control string
                                019B   417                 OUTLEN=W^FAL$GW_PRTLEN1-;  Address of receive string length
                                019B   418                 OUTBUF=W^FAL$GQ_PRTBUF1 ;  Address of buffer to put string
                                01AE   419         $CHECK_STATUS                   ; Check status code
                     FE4C'   30 01B1   420         BSBW    FAL$PRINT_FAO           ; Print message
                                01B4   421
                                01B4   422 ;
                                01B4   423 ; Write and informational message if DAP messages are to be logged.
                                01B4   424 ;
                                01B4   425
              68    22   E1   01B4   426 40$:     BBC     #FAL$V_LOG_MSG,(R8),-   ; Branch if logging of DAP messages is
                 19             01B7   427                 COMPLETE_LOGICAL_LINK   ;  disabled
                                01B8   428         $FAO_S-                         ; Format the message
                                01B8   429                 CTRSTR=W^FAL$GQ_HEADER- ;  Address of control string
                                01B8   430                 OUTLEN=W^FAL$GW_PRTLEN1-;  Address of receive string length
                                01B8   431                 OUTBUF=W^FAL$GQ_PRTBUF1 ;  Address of buffer to put string
                                01CB   432         $CHECK_STATUS                   ; Check status code
                FE2F'   30 01CE   433         BSBW    FAL$PRINT_FAO           ; Print message
```

```
                        01D1    435 ;+
                        01D1    436 ; Respond to the connect initiate with a connect accept (without userdata)
                        01D1    437 ; to complete the logical link.
                        01D1    438 ;
                        01D1    439 ; Note: The equivalence string from the translation of SYS$NET will be shortened
                        01D1    440 ;       to remove unwanted userdata two bytes past the slash character so that
                        01D1    441 ;       this modified string can be used as the Network Connect Block (NCB) on
                        01D1    442 ;       the QIO connect accept request!!
                        01D1    443 ;-
                        01D1    444
                        01D1    445 COMPLETE_LOGICAL_LINK:                  ; Continuation of mainline
              82  2F  90  01D1    446         MOVB    #^A\/\,(R2)+            ; Restore slash to string
                  82  B5  01D4    447         TSTW    (R2)+                   ; Skip over first 2 bytes of binary data
              82  22  90  01D6    448         MOVB    #^A\"\,(R2)+            ; Terminate string here to make an NCB
          67  52  04 A7  C3  01D9    449         SUBL3   4(R7),R2,(R7)           ; Update string size in NCB descriptor
                        01DE    450         $QIOW_S-                        ; Issue connect accept to complete link
                        01DE    451                 EFN=#FAL$K_XMTEFN-      ;  Event flag number
                        01DE    452                 CHAN=FAL$W_LNKCHN(R8)-  ;  Channel number
                        01DE    453                 FUNC=#IO$_ACCESS!IO$M_ACCESS- ; Function code
                        01DE    454                 IOSB=FAL$Q_XMTIOSB(R8)- ;  Address of I/O status block
                        01DE    455                 P1=0-                   ;  Must be zero
                        01DE    456                 P2=R7                   ;  Address of NCB descriptor
                        01FD    457         $CHECK_SS                       ; Check status code and exit on failure
                        0200    458
                        0200    459 ;
                        0200    460 ; Write link established message to the print file.
                        C200    461 ;
                        02J0    462
              68  21  E1  0200    463         BBC     #FAL$V_LOG_STA,(R8),-   ; Branch if logging of statistics is
                  45        0203    464                 ALLOCATE_BUFFERS        ;  disabled
                        0204    465         $GETJPIW_S-                     ; Get selected job/process information
                        0204    466                 ITMLST=W^FAL$GETJPI_LST0; Address of item list
                        0219    467         $CHECK_STATUS                   ; Check status code
                        021C    468         $GETTIM_S-                      ; Get current date and time
                        021C    469                 TIMADR=W^FAL$GQ_TIME0   ;  Address to receive time value
                        0227    470         $CHECK_STATUS                   ; Check status code
                        022A    471         $FAO_S-                         ; Format the message
                        022A    472                 CTRSTR=W^FAL$GQ_LINKUP- ;  Address of control string
                        022A    473                 OUTLEN=W^FAL$GW_PRTLEN1-;  Address of receive string length
                        022A    474                 OUTBUF=W^FAL$GQ_PRTBUF1-;  Address of buffer to put string
                        022A    475                 P1=#FAL$GQ_TIME0        ;  Address of absolute date and time
                        0243    476         $CHECK_STATUS                   ; Check status code
          FDB7'  30     0246    477         BSBW    FAL$PRINT_FAO           ; Print message
```

```
                     0249   479 ;+
                     0249   480 ; Allocate space for two receive buffers and one double-length transmit buffer.
                     0249   481 ;  he receive buffers will be used alternately to obtain incoming DAP messages.
                     0249   482 ; The transmit buffer will be used for both building new DAP messages (BLD
                     0249   483 ; descriptor) and for concatenating DAP messages together (XMT descriptor)
                     0249   484 ; before sending them. The BLD area will overlay the XMT area and may overflow
                     0249   485 ; into the second part of the transmit buffer area, thus a double-length buffer
                     0249   486 ; area will be allocated.
                     0249   487 ;
                     0249   488 ; Also, allocate RMS block I/O buffers for use during file transfer mode.
                     0249   489 ;-
                     0249   490
                     0249   491 ALLOCATE_BUFFERS:                              ; Allocate logical link I/O buffers
54   7FFF 8F    3C   0249   492         MOVZWL  #FAL$K_MAXBUFSIZ,R4            ; Get supported DAP buffer size
  18 A8    54   B0   024E   493         MOVW    R4,FAL$W_QIOBUFSIZ(R8)         ; Save supported buffer size for use
                     0252   494                                                ;  in building Configuration message
  1A A8    54   B0   0252   495         MOVW    R4,FAL$W_DAPBUFSIZ(R8)         ; Initialize DAP buffer size for use by
                     0256   496                                                ;  FAL$RCV_QIO (it will be revised after
                     0256   497                                                ;  exchange of Configuration messages)
                     0256   498
                     0256   499 ;
                     0256   500 ; Allocate space for two receive buffers and one double-length transmit buffer
                     0256   501 ; based on the largest QIO that FAL is prepared to support. Then put size and
                     0256   502 ; address information about them in the FAL work area. Note that all buffers
                     0256   503 ; will start on a page boundary and that the transmit buffer is the size of two
                     0256   504 ; QIO buffers to provide an overflow area potentially used while building a
                     0256   505 ; when there are concatenated messages already in the transmit buffer.
                     0256   506 ;
                     0256   507
  05 68    39   E1   0256   508         BBC     #FAL$V_USE_DBS,(R8),10$ ; Branch to use calculated buffer size
54   00A0  C8   3C   025A   509         MOVZWL  FAL$W_USE_DBS(R8),R4    ; Override with user specified value
54 000001FF  8F   C0   025F   510 10$:  ADDL2   #511,R4                 ; Compute # of pages required for
 54   54    F7 8F   78   0266   511         ASHL    #-9,R4,R4               ;  each buffer
                  04   C5   026B   512         MULL3   #<FAL$K_RCVBUFCNT+1+1>,-; Compute total # of pages required
           51   54         026D   513                 R4,R1                   ;  for receive and transmit buffers
              013C   30   026F   514         BSBW    FAL$GETPAGE             ; Expand program region
     55   54    09   78   0272   515         ASHL    #9,R4,R5                ; Compute # bytes per buffer
          50    02   D0   0276   516         MOVL    #FAL$K_RCVBUFCNT,R0     ; Get number of receive buffers
       51   5C A8   DE   0279   517         MOVAL   FAL$L_RCVBUF(R8),R1     ; Get address of buffer address table
          81    52   D0   027D   518 20$:  MOVL    R2,(R1)+                ; Store receive buffer address
          52    55   C0   0280   519         ADDL2   R5,R2                   ; Compute next buffer address
            F7  50   F5   0283   520         SOBGTR  R0,20$                  ; Branch if more to do
            48 A8   D4   0286   521         CLRL    FAL$Q_XMT(R8)           ; Initialize transmit buffer descriptor
     4C A8    52   D0   0289   522         MOVL    R2,FAL$Q_XMT+4(R8)      ; Store transmit buffer address
                     028D   523
                     028D   524 ;
                     028D   525 ; Allocate space for RMS multi-block buffer for use during file transfer mode
                     028D   526 ; to issue large block I/O requests to the local RMS.
                     028D   527 ;
                     028D   528
       51   12 A8   9A   028D   529         MOVZBL  FAL$B_RBK_CACHE(R8),R1  ; Get number of pages to allocate
     54   51    09   78   0291   530         ASHL    #9,R1,R4                ; Calculate # bytes in buffer
              0116   30   0295   531         BSBW    FAL$GETPAGE             ; Expand program region
       64 A8    54   D0   0298   532         MOVL    R4,FAL$Q_RMS(R8)        ; Store RMS buffer size in descriptor
       68 A8    52   D0   029C   533         MOVL    R2,FAL$Q_RMS+4(R8)      ; Store RMS buffer addr in descriptor
       6C A8    52   D0   02A0   534         MOVL    R2,FAL$L_RMS_PTR(R8)    ; Initialize next byte pointer
```

```
                     02A4    536 ;+
                     02A4    537 ; Post read requests to the mailbox and logical link, then transfer control
                     02A4    538 ; to the state table manager.
                     02A4    539 ;-
                     02A4    540
                     02A4    541 END_OF_SETUP:                           ; Continuation of mainline
                     02A4    542
                     02A4    543 ;
                     02A4    544 ; Issue a read with an AST for the associated mailbox so that FAL will be
                     02A4    545 ; notified of a change in the status of the link such as partner exited or
                     02A4    546 ; of the receipt of an interrupt message.
                     02A4    547 ;
                     02A4    548 ; Note: Each time that a read completes, another read will be issued to keep
                     02A4    549 ;       one read request outstanding.
                     02A4    550 ;
                     02A4    551
          FD59'  30  02A4    552         BSBW    FAL$MBX_RCV_QIO         ; Issue mailbox read with AST
                     02A7    553
                     02A7    554 ;
                     02A7    555 ; Issue a read with an AST for the logical link.
                     02A7    556 ;
                     02A7    557 ; Note: Each time that a read completes, another read will be issued to keep
                     02A7    558 ;       one read request outstanding.
                     02A7    559 ;
                     02A7    560
          FD56'  30  02A7    561         BSBW    FAL$RCV_QIO             ; Issue link read with AST
                     02AA    562
                     02AA    563 ;
                     02AA    564 ; Transfer control to the state transiton table mananger.
                     02AA    565 ;
                     02AA    566
     52 0000'CF  9E  02AA    567         MOVAB   W^FAL$STATE_TABLE,R2    ; Get address of state transition table
        FD4E'  30  02AF    568         BSBW    FAL$STATE               ; Give control to state table manager
```

```
                        02B2    570          .SBTTL  FAL$TERMINATE -- NORMAL IMAGE EXIT
                        02B2    571
                        02B2    572 ;++
                        02B2    573 ; Normal image exit to VMS. Transfer control here to unconditionally terminate
                        02B2    574 ; FAL in a graceful manner.
                        02B2    575 ;--
                        02B2    576
                        02B2    577 FAL$TERMINATE::                             ; Control point
                        02B2    578          $SETBIT #FAL$V_TERMINATE,(R8)      ; Signal image termination in progress
                        02B6    579
                        02B6    580 ;
                        02B6    581 ; Write link terminated message and statistics messages to the print file.
                        02B6    582 ;
                        02B6    583
 4C 68    21    E1      02B6    584          BBC     #FAL$V_LOG_STA,(R8),10$ ; Branch if logging disabled
 48 68    00    E1      02BA    585          BBC     #FAL$V_CNF_MSG,(R8),10$ ; Branch if no messages received
                        02BE    586          $GETTIM_S-                      ; Get current date and time
                        02BE    587                  TIMADR=W^FAL$GQ_TIME1   ;  Address to receive time value
                        02C9    588          $CHECK_STATUS                   ; Check status code
                        02CC    589          $GETJPIW_S-                     ; Get selected job/process information
                        02CC    590                  ITMLST=W^FAL$GETJPI_LST1;  Address of item list
                        02E1    591          $CHECK_STATUS                   ; Check status code
                        02E4    592          $FAO_S-                         ; Format the message
                        02E4    593                  CTRSTR=W^FAL$GQ_LINKDOWN-; Address of control string
                        02E4    594                  OUTLEN=W^FAL$GW_PRTLEN1-;  Address of receive string length
                        02E4    595                  OUTBUF=W^FAL$GQ_PRTBUF1-;  Address of buffer to put string
                        02E4    596                  P1=#FAL$GQ_TIME1        ;  Address of absolute date and time
                        02FD    597          $CHECK_STATUS                   ; Check status code
       FCFD'    30      0300    598          BSBW    FAL$PRINT_FAO           ; Print message
       FCFA'    30      0303    599          BSBW    FAL$STATISTICS          ; Compute and print statistics
                        0306    600
                        0306    601 ;
                        0306    602 ; Write internal counters to the print file.
                        0306    603 ;
                        0306    604
 31 68    25    E1      0306    605 10$:     BBC     #FAL$V_LOG_CNT,(R8),20$ ; Branch if internal counters not wanted
                        030A    606          $FAO_S-                         ; Format the message
                        030A    607                  CTRSTR=W^FAL$GQ_INTCNTR-; Address of control string
                        030A    608                  OUTLEN=W^FAL$GW_PRTLEN1-;  Address of receive string length
                        030A    609                  OUTBUF=W^FAL$GQ_PRTBUF1-;  Address of buffer to put string
                        030A    610                  P1=W^FAL$GL_RECVWAIT-   ;  Receive QIO wait count
                        030A    611                  P2=W^FAL$GL_WRITWAIT-   ;  RMS FTM WRITE wait count
                        030A    612                  P3=W^FAL$GL_COUNTER1-   ;  Miscellaneous counter 1
                        030A    613                  P4=W^FAL$GL_COUNTER2-   ;  Miscellaneous counter 2
                        030A    614                  P5=W^FAL$GL_COUNTER3-   ;  Miscellaneous counter 3
                        030A    615                  P6=W^FAL$GL_COUNTER4    ;  Miscellaneous counter 4
                        0335    616          $CHECK_STATUS                   ; Check status code
       FCC5'    30      0338    617          BSBW    FAL$PRINT_FAO           ; Print message
                        033B    618
                        033B    619 ;
                        033B    620 ; Write exit message to the print file.
                        033B    621 ;
                        033B    622
    04 A8    95      033B    623 20$:     TSTB    FAL$B_LOGGING(R8)       ; Branch if FAL logging is disabled
       29    13      033E    624          BEQL    IMAGE_EXIT              ;
                        0340    625          $FAO_S-                         ; Format the message
                        0340    626                  CTRSTR=W^FAL$GQ_EXIT-   ;  Address of control string
```

```
                0340   627                     OUTLEN=W^FAL$GW_PRTLEN1-;  Address of receive string length
                0340   628                     OUTBUF=W^FAL$GQ_PRTBUF1-;  Address of buffer to put string
                0340   629                     P1=#0                      ;  Use current date and time
                0355   630             $CHECK_STATUS                      ; Check status code
FCA5'    30     0358   631             BSBW    FAL$PRINT_FAO              ; Print message
                035B   632
                035B   633 ;
                035B   634 ; Close the print file.
                035B   635 ;
                035B   636
                035B   637             $CLOSE  FAB=W^FAL$PRTFAB           ; Close the print file
                0366   638             $CHECK_STATUS                      ; Check completion code
                0369   639
                0369   640 ;+
                0369   641 ; Quick exit to VMS (or continuation of FAL$TERMINATE). Transfer control here
                0369   642 ; to abruptly terminate FAL.
                0369   643 ;
                0369   644 ; Note: Image rundown will deassign all channels (which will break the logical
                0369   645 ;       link with partner).
                0369   646 ;-
                0369   647
                0369   648 IMAGE_EXIT:                                     ; Control point
                0369   649             $EXIT_S CODE=W^FAL$GL_EXIT         ; Exit to VMS with status code specified
```

```
                        0374   651          .SBTTL  FAL$CHECK_SS - CHECK SYSTEM SERVICE STATUS CODE
                  00000374   652          .PSECT  FAL$CODE         NOSHR,EXE,RD,NOWRT,BYTE
                        0374   653
                        0374   654 ;++
                        0374   655 ; Functional Description:
                        0374   656 ;
                        0374   657 ;       FAL$CHECK_SS checks the status code in R0 following a System Service
                        0374   658 ;       call. If failure is indicated after filtering out allowable error
                        0374   659 ;       conditions, the image is terminated with R0 as the exit completion code.
                        0374   660 ;
                        0374   661 ; Calling Sequence:
                        0374   662 ;
                        0374   663 ;       BSBW    FAL$CHECK_SS
                        0374   664 ;
                        0374   665 ; Input Parameters:
                        0374   666 ;
                        0374   667 ;       R0      System Service status code
                        0374   668 ;
                        0374   669 ; Implicit Inputs:
                        0374   670 ;
                        0374   671 ;       FAL$GL_EXIT
                        0374   672 ;
                        0374   673 ; Output Parameters:
                        0374   674 ;
                        0374   675 ;       None
                        0374   676 ;
                        0374   677 ; Implicit Outputs:
                        0374   678 ;
                        0374   679 ;       FAL$GL_EXIT
                        0374   680 ;
                        0374   681 ; Completion Codes:
                        0374   682 ;
                        0374   683 ;       None
                        0374   684 ;
                        0374   685 ; Side Effects:
                        0374   686 ;
                        0374   687 ;       If the System Service indicates failure after filtering out allowable
                        0374   688 ;       error conditions, the image is terminated with R0 on input as the exit
                        0374   689 ;       completion code.
                        0374   690 ;
                        0374   691 ;--
                        0374   692
                        0374   693 FAL$CHECK_SS::                                ; Entry point
              01 50   E9  0374   694          BLBC    R0,10$                    ; Was System Service successful?
                      05  0377   695          RSB                               ; Yes, exit
                          0378   696                                            ; No, terminate execution ...
        1D 0008'CF    E9  0378   697 10$:     BLBC    W^FAL$GL_EXIT,30$         ; Branch if an error was previously
                          037D   698                                            ;  detected to prevent possible loop
    20E4 8F     50    B1  037D   699          CMPW    R0,#SS$_LINKABORT         ; Do not report these error codes as
                  13    13  0382   700          BEQL    20$                       ;  the partner process tells FAL to
    20EC 8F     5C    B1  0384   701          CMPW    R0,#SS$_LINKDISCON        ;  terminate by breaking the logical
                  0C    13  0389   702          BEQL    20$                       ;  link (i.e., there is no DAP directive
    20F4 8F     50    B1  038B   703          CMPW    R0,#SS$_LINKEXIT          ;  to tell FAL to terminate)
                  05    13  0390   704          BEQL    20$                       ;
    0008'CF     50    D0  0392   705          MOVL    R0,W^FAL$GL_EXIT          ; Store error code for use at image exit
                FF18    31  0397   706 20$:     BRW     FAL$TERMINATE             ; Take graceful termination path
                FFCC    31  039A   707 30$:     BRW     IMAGE_EXIT                ; Take quick termination path
```

```
                     039D   709          .SBTTL  FAL$CHECK_STATUS - CHECK STATUS CODE (NON-FATAL)
                0000039D   710          .PSECT  FAL$CODE        NOSHR,EXE,RD,NOWRT,BYTE
                     039D   711
                     039D   712 ;++
                     039D   713 ; Functional Description:
                     039D   714 ;
                     039D   715 ;       FAL$CHECK_STATUS checks the status code in R0 following an RMS or
                     039D   716 ;       System Service call. This routine should be used to check the status
                     039D   717 ;       of a FAL logging operation. Unlike FAL$CHECK_SS, this routine does not
                     039D   718 ;       terminate FAL on detecting an error. Instead it disables FAL logging
                     039D   719 ;       and stores the status code for subsequent reporting when FAL terminates
                     039D   720 ;       normally. Thus, a FAL logging failure will not disrupt the remote file
                     039D   721 ;       access operation in progress.
                     039D   722 ;
                     039D   723 ; Calling Sequence:
                     039D   724 ;
                     039D   725 ;       BSBW    FAL$CHECK_STATUS
                     039D   726 ;
                     039D   727 ; Input Parameters:
                     039D   728 ;
                     039D   729 ;       R0      System Service status code or RMS completion code
                     039D   730 ;       R8      Address of FAL work area
                     039D   731 ;
                     039D   732 ; Implicit Inputs:
                     039D   733 ;
                     039D   734 ;       FAL$GL_EXIT
                     039D   735 ;
                     039D   736 ; Output Parameters:
                     039D   737 ;
                     039D   738 ;       None
                     039D   739 ;
                     039D   740 ; Implicit Outputs:
                     039D   741 ;
                     039D   742 ;       FAL$GL_EXIT
                     039D   743 ;       FAL$B_LOGGING
                     039D   744 ;
                     039D   745 ; Completion Codes:
                     039D   746 ;
                     039D   747 ;       None
                     039D   748 ;
                     039D   749 ; Side Effects:
                     039D   750 ;
                     039D   751 ;       If the status code indicates failure, then FAL logging is disabled and
                     039D   752 ;       the failure code is saved for display on image exit.
                     039D   753 ;
                     039D   754 ;--
                     039D   755
                     039D   756 FAL$CHECK_STATUS::                          ; Entry point
          0D 50   E8 039D   757          BLBS    R0,10$                     ; Was service call successful?
          04 A8   94 03A0   758          CLRB    FAL$B_LOGGING(R8)          ; No, disable further logging output
05 0008'CF        E9 03A3   759          BLBC    W^FAL$GL_EXIT,10$          ; Store error code in R0 for use at
0008'CF   50      D0 03A8   760          MOVL    R0,W^FAL$GL_EXIT           ;  image exit unless an error was
                     03AD   761                                             ;  previously detected
                     05 03AD   762 10$:     RSB                            ; Exit
```

```
                     03AE    764          .SBTTL  FAL$GETPAGE - EXPAND PROGARM REGION
                000003AE    765          .PSECT  FAL$CODE        NOSHR,EXE,RD,NOWRT,BYTE
                     03AE    766
                     03AE    767 ;++
                     03AE    768 ; Functional Description:
                     03AE    769 ;
                     03AE    770 ;       FAL$GETPAGE expands the program region (P0) by the number of pages
                     03AE    771 ;       specified. These are demand zero pages.
                     03AE    772 ;
                     03AE    773 ; Calling Sequence:
                     03AE    774 ;
                     03AE    775 ;       BSBW    FAL$GETPAGE
                     03AE    776 ;
                     03AE    777 ; Input Parameters:
                     03AE    778 ;
                     03AE    779 ;       R1      # of pages to allocate
                     03AE    780 ;
                     03AE    781 ; Implicit Inputs:
                     03AE    782 ;
                     03AE    783 ;       None
                     03AE    784 ;
                     03AE    785 ; Output Parameters:
                     03AE    786 ;
                     03AE    787 ;       R0-R1   Destroyed
                     03AE    788 ;       R2      Address of first byte allocated
                     03AE    789 ;       R3      Address of last byte allocated
                     03AE    790 ;
                     03AE    791 ; Implicit Outputs:
                     03AE    792 ;
                     03AE    793 ;       None
                     03AE    794 ;
                     03AE    795 ; Completion Codes:
                     03AE    796 ;
                     03AE    797 ;       None
                     03AE    798 ;
                     03AE    799 ; Side Effects:
                     03AE    800 ;
                     03AE    801 ;       The virtual address space of the image is expanded.
                     03AE    802 ;
                     03AE    803 ;--
                     03AE    804
                     03AE    805 FAL$GETPAGE::                           ; Entry point
          7E    7C 03AE    806          CLRQ    -(SP)                   ; Form array to receive address bounds
     50   5E    D0 03B0    807          MOVL    SP,R0                   ;  and save its address
                     03B3    808          $EXPREG_S-                      ; Expand program region
                     03B3    809                  PAGCNT=R1-              ;  # pages desired in P0 space
                     03B3    810                  RETADR=(R0)             ;  Address to receive lo & hi addresses
                     03C2    811          $CHECK_SS                       ; Check status code and exit on failure
     52   8E    7D 03C5    812          MOVQ    (SP)+,R2                ; Store first and last addresses
                05 03C8    813          RSB                             ; Exit
                     03C9    814
                     03C9    815          .END    FAL$START               ; Image transfer address
```

```
$$.TMP1                 = 00000001
$$.TMP2                 = 000000CF
$$T1                    = 00000000
$$T2                    = 00000004
ALLOCATE_BUFFERS          00000249 R      03
ASSIGN_CHANNEL            000000D3 R      03
COMPLETE_LOGICAL_LINK     000001D1 R      03
CREATE_PRINT_FILE         0000009E R      03
DAP$B_DECVER              00000047
DAP$B_ECONUM              00000045
DAP$B_FILESYS             00000043
DAP$B_OSTYPE              00000042
DAP$B_USRNUM              00000046
DAP$B_USRVER              00000048
DAP$B_VERNUM              00000044
DAP$Q_SYSCAP              00000028
DAP$W_BUFSIZ              00000040
END_OF_SETUP              000002A4 R      03
FAB$B_FNS                 ********    X   03
FAB$L_FNA                 ********    X   03
FAL$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$CHECK_SS              00000374 RG     03
FAL$CHECK_STATUS          0000039D RG     03
FAL$C_WRKBLN              00002000
FAL$GETJPI_LST0           ********    X   03
FAL$GETJPI_LST1           ********    X   03
FAL$GETPAGE               000003AE RG     03
FAL$GL_COUNTER1           ********    X   03
FAL$GL_COUNTER2           ********    X   03
FAL$GL_COUNTER3           ********    X   03
FAL$GL_COUNTER4           ********    X   03
FAL$GL_EXIT               00000008 RG     01
FAL$GL_RECVWAIT           ********    X   03
FAL$GL_WRITWAIT           ********    X   03
FAL$GQ_CALLER             ********    X   03
FAL$GQ_EXIT               ********    X   03
FAL$GQ_HEADER             ********    X   03
FAL$GQ_INTCNTR            ********    X   03
FAL$GQ_LINKDOWN           ********    X   03
FAL$GQ_LINKUP             ********    X   03
FAL$GQ_LNKNAM             0000000C RG     01
FAL$GQ_LOGNAM             00000029 RG     01
FAL$GQ_OUTPUT             00000038 RG     01
FAL$GQ_PARSERR            ********    X   03
FAL$GQ_PRTBUF1            ********    X   03
FAL$GQ_SYSNAM             0000001A RG     01
FAL$GQ_TIME0              ********    X   03
FAL$GQ_TIME1              ********    X   03
FAL$GQ_WILDSPEC           0000004A RG     01
FAL$GT_NODENAME           ********    X   03
FAL$GT_VERSION            00000000 RG     01
FAL$GW_PRTLEN1            ********    X   03
FAL$K_DFLT_RBK          = 00000040
FAL$K_FALLOG            = 000000FF
FAL$K_FILESPEC          = 000000FF
FAL$K_MAXBUFSIZ         = 00007FFF
FAL$K_MBXBUF            = 00000040
FAL$K_MBXQUOTA          = 00000080
FAL$K_PRTBUF            = 00000100
FAL$K_RCVBUFCNT         = 00000002
FAL$K_SYSNET            = 000000FF
FAL$K_WRKBLN              00002000
FAL$K_XMTEFN            = 00000002
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_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$L_USE_VER             000000A4
FAL$MBX_RCV_QIO           ********    X   03
FAL$PARSE_FAL$LOG         ********    X   03
FAL$PRINT_FAO             ********    X   03
FAL$PRTFAB                ********    X   03
FAL$PRTRAB                ********    X   03
FAL$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$RCV_QIO                     ********    X   03
FAL$START                       00000000 RG     03
FAL$STATE                       ********    X   03
FAL$STATE_TABLE                 ********    X   03
FAL$STATISTICS                  ********    X   03
FAL$TERMINATE                   000002B2 RG     03
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_CNF_MSG                 = 00000000
FAL$V_LOG_CNT                 = 00000025
FAL$V_LOG_MSG                 = 00000022
FAL$V_LOG_STA                 = 00000021
FAL$V_PARSE_ERR               = 00000038
FAL$V_TERMINATE               = 0000000C
FAL$V_USE_DBS                 = 00000039
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
IMAGE_EXIT                      00000369 R      03
IO$M_ACCESS                   = 00000040
IO$_ACCESS                    = 00000032
LIB$ASN_WTH_MBX                 ********    X   03
SS$_IVLOGNAM                  = 00000154
SS$_LINKABORT                 = 000020E4
SS$_LINKDISCON                = 000020EC
SS$_LINKEXIT                  = 000020F4
SS$_NORMAL                    = 00000001
SS$_NOTRAN                    = 00000629
SYS$CLOSE                       ********    GX  03
SYS$CONNECT                     ********    GX  03
SYS$CREATE                      ********    GX  03
SYS$EXIT                        ********    GX  03
SYS$EXPREG                      ********    GX  03
SYS$FAO                         ********     X  03
SYS$GETJPIW                     ********    GX  03
SYS$GETTIM                      ********    GX  03
SYS$QIOW                        ********    GX  03
SYS$TRNLOG                      ********    GX  03
TRANSLATE_FAL$LOG               00000021 R      03
TRANSLATE_FAL$OUTPUT            0000005B R      03
TRANS_SYS$NET                   000000FC R      03
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
FAL$DATA	00000057 (87.)	01 (1.)	NOPIC USR CON REL LCL SHR NOEXE RD WRT NOVEC LONG
ABS	00002000 (8192.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
FAL$CODE	000003C9 (969.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	36	00:00:00.04	00:00:01.58
Command processing	128	00:00:00.36	00:00:02.20
Pass 1	366	00:00:09.06	00:00:26.71
Symbol table sort	0	00:00:01.20	00:00:04.58
Pass 2	160	00:00:02.07	00:00:07.63
Symbol table output	22	00:00:00.13	00:00:00.27
Psect synopsis output	2	00:00:00.01	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	716	00:00:12.87	00:00:42.99

The working set limit was 1650 pages.
73909 bytes (145 pages) of virtual memory were used to buffer the intermediate code.
There were 70 pages of symbol table space allocated to hold 1167 non-local and 24 local symbols.
815 source lines were read in Pass 1, producing 24 object records in Pass 2.
33 pages of virtual memory were used to define 30 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[FAL.OBJ]FAL.MLB;1	6
_255DUA28:[SYSLIB]STARLET.MLB;2	21
TOTALS (all libraries)	27

1413 GETS were required to define 27 macros.

There were no errors, warnings or information messages.

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

FALDECODE
LIS

FALRMSDAP
LIS

FALSTATE
LIS

FDL

CREATEFDL
MAP

FALENCODE
LIS

FALLOGGER
LIS

FALMAIN
LIS