



```

FFFFFFFFF      AAAAAA  LL      LL      000000      GGGGGGGG      GGGGGGGG      EEEEEEEEEE      RRRRRRRR
FFFFFFFFF      AAAAAA  LL      LL      000000      GGGGGGGG      GGGGGGGG      EEEEEEEEEE      RRRRRRRR
FF           AA      AA  LL      LL      00      00      GG      GG      GG      GG      EE      RR      RR
FF           AA      AA  LL      LL      00      00      GG      GG      GG      GG      EE      RR      RR
FF           AA      AA  LL      LL      00      00      GG      GG      GG      GG      EE      RR      RR
FF           AA      AA  LL      LL      00      00      GG      GG      GG      GG      EE      RR      RR
FFFFFFFFF      AA      AA  LL      LL      00      00      GG      GG      GG      GG      EEEEEEEEEE      RRRRRRRR
FFFFFFFFF      AA      AA  LL      LL      00      00      GG      GG      GG      GG      EEEEEEEEEE      RRRRRRRR
FF           AAAAAAAAAA LL      LL      00      00      GG      GGGGGG      GG      GGGGGG      EE      RR      RR
FF           AAAAAAAAAA LL      LL      00      00      GG      GGGGGG      GG      GGGGGG      EE      RR      RR
FF           AA      AA  LL      LL      00      00      GG      GG      GG      GG      EE      RR      RR
FF           AA      AA  LL      LL      00      00      GG      GG      GG      GG      EE      RR      RR
FF           AA      AA  LLLLLLLLLL LLLLLLLLLL 000000      GGGGGG      GGGGGG      EEEEEEEEEE      RR      RR
FF           AA      AA  LLLLLLLLLL LLLLLLLLLL 000000      GGGGGG      GGGGGG      EEEEEEEEEE      RR      RR

```

```

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
LLLLLLLLLLL IIIIII  SSSSSSSS
LLLLLLLLLLL IIIIII  SSSSSSSS

```

(2)	76	FAL LOGGING OPTIONS DESCRIPTION
(3)	248	DECLARATIONS
(4)	662	FAL\$PARSE_FAL\$LOG - PARSE FAL\$LOG STRING
(5)	871	FAL\$STATISTICS - COMPUTE AND PRINT STATISTICS
(6)	1197	FAL\$PRINT_FAO, FAL\$PRINT_FAO_ASTLEVEL
(7)	1272	FAL\$DISPLAY_MSG - DISPLAY MESSAGE BUFFER
(8)	1472	FAL\$LOG_QIO, FAL\$LOG_AST
(9)	1506	FAL\$LOG_REQNAM, FAL\$LOG_REQNAM2
(10)	1549	FAL\$LOG_RESNAM, FAL\$LOG_CLSMSG

```

0000 1      .TITLE FALLOGGER - FAL LOGGING ROUTINES
0000 2      .IDENT 'V04-000'
0000 3
0000 4
0000 5 *****
0000 6 *
0000 7 *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9 *  ALL RIGHTS RESERVED.
0000 10 *
0000 11 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12 *  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15 *  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16 *  TRANSFERRED.
0000 17 *
0000 18 *  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19 *  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20 *  CORPORATION.
0000 21 *
0000 22 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24 *
0000 25 *
0000 26 *****
0000 27
0000 28
0000 29 ++
0000 30 Facility: FAL (DECnet File Access Listener)
0000 31
0000 32 Abstract:
0000 33
0000 34 This module contains FAL logging routines and associated text.
0000 35
0000 36 Environment: VAX/VMS, user mode
0000 37
0000 38 Author: James A. Krycka,      Creation Date: 16-JUN-1977
0000 39
0000 40 Modified By:
0000 41
0000 42 V03-009 JAK0144      J A Krycka      11-APR-1984
0000 43 Use local symbols instead of global symbols wherever possible.
0000 44 Log CPU time with FALSLOG statistics display and optionally
0000 45 display internal counters.
0000 46
0000 47 V03-008 JAK0137      J A Krycka      12-MAR-1984
0000 48 Add FAL$PARSE_FAL$LOG routine to parse FAL logging options.
0000 49 Make minor changes to FAL logging display.
0000 50 Add description of FALSLOG options and use of FALSOUTPUT.
0000 51 Modifications to reflect macro name changes in FALMACROS.MAR.
0000 52
0000 53 V03-007 JAK0136      J A Krycka      07-MAR-1984
0000 54 Do not terminate FAL on error writing to print file with FAL
0000 55 logging information.
0000 56
0000 57 V03-006 JAK0107      J A Krycka      30-APR-1983

```

```
0000 58 : This fix supersedes V03-002.
0000 59 :
0000 60 : V03-005 JAK0105 J A Krycka 29-APR-1983
0000 61 : Make general enhancements to FAL logging display and log more
0000 62 : performance diagnostic information.
0000 63 :
0000 64 : V03-004 KRM0097 K Malik 06-Apr-1983
0000 65 : Add support for DAP V7.0 rename operation.
0000 66 :
0000 67 : V03-003 KRM0071 K Malik 23-Nov-1982
0000 68 : Modify FAL$LOG_RESNAM to support rename operation.
0000 69 :
0000 70 : 03-002 KRM0062 K Malik 08-Oct-1982
0000 71 : Modified FAL$PRINT_FA0 to loop on encountering a
0000 72 : Record Stream Active error, instead of quitting.
0000 73 :
0000 74 :--
```

```
0000 76      .SBTTL  FAL LOGGING OPTIONS DESCRIPTION
0000 77
0000 78      :++
0000 79
0000 80      : INTRODUCTION TO FAL LOGGING
0000 81
0000 82      : The following describes the format and use of the FAL logging options which
0000 83      : are specified via the logical names FAL$LOG and FAL$OUTPUT. These are normally
0000 84      : defined in the user's LOGIN.COM file, but they can be placed in a group or
0000 85      : system logical name table to affect a larger class of remote file accesses.
0000 86      : FAL$LOG conveys logging and control directives to FAL and FAL$OUTPUT is used
0000 87      : to specify the name of the log file to create (in place of SYSS$OUTPUT).
0000 88
0000 89      : NOTE: Use of the logical names FAL$LOG and FAL$OUTPUT by FAL is an UNSUPPORTED
0000 90      : feature intended as a diagnostic, debugging, and performance monitoring
0000 91      : tool for use by Digital. The format and function of these logical names
0000 92      : may change at any time, or perhaps not be used in the future.
0000 93
0000 94      : NOTE: Logging of information other than file name and statistics (parameter
0000 95      : bits 0, 1, and 5) can severely reduce data throughput!!!
0000 96
0000 97
0000 98      : SYNTAX RULES
0000 99
0000 100     : The primary function of the logical name FAL$LOG is to request the logging of
0000 101     : various types of information about the file operations performed by FAL. This
0000 102     : includes identifying each file accessed, displaying the Data Access Protocol
0000 103     : (DAP) messages exchanged, computing data throughput statistics, and logging
0000 104     : the logical link and mailbox QIO calls and the subsequent delivery of ASTs.
0000 105     : Logging operations are requested via the parameter bitmask value. A secondary
0000 106     : use of the logical name is to specify qualifiers that control various aspects
0000 107     : of FAL's operation such as determining buffer sizes or disabling features.
0000 108     : Currently, the format of the FAL$LOG options string is:
0000 109
0000 110     :     [parameter][/qualifier-1,....,qualifier-n]
0000 111
0000 112     : where each qualifier is of the form keyword=value (e.g., /BPM=20).
0000 113
0000 114     : The parameter and qualifiers are optional. However, the parameter if present
0000 115     : must precede any qualifiers. In addition, only the first three characters of
0000 116     : a qualifier keyword are examined to determine a match. Thus, /DISABLE=xx can
0000 117     : be abbreviated to /DIS=xx. Spaces and tabs are ignored and keywords can be
0000 118     : be entered using either uppercase or lowercase characters.
0000 119
0000 120
0000 121     : PARAMETER VALUES
0000 122
0000 123     : The parameter is a hexadecimal bitmask used to specify FAL logging options.
0000 124     : If this parameter is non-zero (indicating that FAL logging output will be
0000 125     : generated), then an attempt is made to translate the logical name SYSS$OUTPUT
0000 126     : prior to opening the log file. If FAL$OUTPUT is defined, then its equivalence
0000 127     : string is used as the file specification of the log file; otherwise logging
0000 128     : output is directed to SYSS$OUTPUT which normally points to the default network
0000 129     : log file named SYSS$LOGIN:NETSERVER.LOG. The bitmask definitions for the
0000 130     : parameter are as follows:
0000 131
0000 132     :     bit0 -- enable logging of file name and type of file access requested.
```

```
0000 133 : bit1 -- enable logging of data throughput and other performance
0000 134 : statistics.
0000 135 : bit2 -- enable logging of individual DAP messages as they are processed
0000 136 : from the input buffer or assembled in the output buffer
0000 137 : bit3 -- enable logging of DAP message packet and mailbox AST routine
0000 138 : completions.
0000 139 : bit4 -- enable logging of DAP message packet and mailbox QIO requests.
0000 140 : bit5 -- enable logging of internal counters.
0000 141 :
0000 142 :
0000 143 : QUALIFIER VALUES
0000 144 :
0000 145 : The following qualifiers are recognized where 'd' denotes a decimal digit and
0000 146 : 'x' denotes a hexadecimal digit:
0000 147 :
0000 148 : /DISABLE=xx (Disable FAL Options) where the bitmask value denotes:
0000 149 :
0000 150 : bit0 -- disable DAP level CRC checksum generation and comparison.
0000 151 : (Note that CRC checking will be automatically disabled if the
0000 152 : initiating node does not support DAP level CRC computation.)
0000 153 : bit1 -- disable DAP message blocking in both directions (i.e., transmit
0000 154 : each DAP message in a separate QIO system service call).
0000 155 : bit2 -- disable RMS multi-block caching to/from disk when block I/O
0000 156 : file transfer mode is in effect. This restores the pre-VMS V3.4
0000 157 : block I/O processing behavior of FAL where each DAP DATA message
0000 158 : resulted in one RMS $READ or $WRITE call to be executed. Note
0000 159 : also that selection of this option eliminates one MOVCS copy
0000 160 : of the data in memory at the expense of greatly increasing the
0000 161 : number of RMS I/O operations performed during a file transfer.
0000 162 : bit3 -- disable door-man's (or manual) routing (i.e., have FAL reject
0000 163 : any file specification it receives that contains a node name).
0000 164 : bits4-7 are undefined.
0000 165 :
0000 166 : /ENABLE=xx (Enable FAL Options) where the bitmask value denotes:
0000 167 :
0000 168 : bits0-7 are undefined.
0000 169 :
0000 170 : /BPM=dddd (Bytes per Message) this is the maximum number of bytes per DAP
0000 171 : message to display (used only if parameter bit2 is set). The default
0000 172 : value is 20 bytes per message.
0000 173 :
0000 174 : /BPL=dd (Bytes per Line) this is the maximum number of bytes per line to
0000 175 : display when dumping a DAP message (used only if parameter bit2 is set).
0000 176 : The default value is 20 bytes per line.
0000 177 :
0000 178 : /RBK_CACHE=ddd (RMS Multi-block Cache Size) this controls the number of disk
0000 179 : blocks per RMS $READ or $WRITE call to transfer when block I/O file
0000 180 : transfer mode is selected (if bit2 of the /DISABLE option is set, this
0000 181 : option is ignored). The number can be from 1 to 127. The default is 64.
0000 182 :
0000 183 : /DBS=dddd (DAP Buffer Size) requests FAL to send this value in the DAP
0000 184 : Configuration message for the <BUFSIZ> field.
0000 185 :
0000 186 : /SYSTEM_ID=xxxx (System Identification) requests FAL to send this value in
0000 187 : the DAP Configuration message for the <FILESYS><OSTYPE> fields (the
0000 188 : OSTYPE field is the low order byte of the value).
0000 189 :
```

```
0000 190 : /VERSION=xxxxxxx (DAP Version Number) requests FAL to send this value in
0000 191 : the DAP Configuration message for the <DECVER><USRNUM><ECONUM><VERNUM>
0000 192 : fields (the VERNUM field is the low order byte of the value).
0000 193 :
0000 194 : /SC1=xxxxxxx (System Capabilities Part 1) requests FAL to send this value in
0000 195 : the DAP Configuration message for bits <31-00> of the <SYSCAP> field.
0000 196 :
0000 197 : /SC2=xxxxxxx (System Capabilities Part 2) requests FAL to send this value in
0000 198 : the DAP Configuration message for bits <63-32> of the <SYSCAP> field.
0000 199 :
0000 200 : Note that any qualifier that cannot be interpreted or that contains an invalid
0000 201 : value is ignored and a parse error message is written to the log file.
0000 202 :
0000 203 :
0000 204 : EXAMPLES
0000 205 :
0000 206 : The following DCL commands illustrate how FAL logging options may be setup
0000 207 : in one's LOGIN.COM file.
0000 208 :
0000 209 :     $ DEFINE FAL$LOG 1
0000 210 :
0000 211 : The above command enables the logging of file name and type of access in
0000 212 : the default network log file NETSERVER.LOG.
0000 213 :
0000 214 :     $ DEFINE FAL$LOG 3
0000 215 :     $ DEFINE FAL$OUTPUT FAL.LOG
0000 216 :
0000 217 : This requests the logging of file name, type of access, and data throughput
0000 218 : statistics in SYSS$LOGIN:FAL.LOG.
0000 219 :
0000 220 :     $ DEFINE FAL$LOG "3/RBK CACHE=16/DBS=1056"
0000 221 :     $ DEFINE FAL$OUTPUT work_disk:[testing]statistics.star_to_galaxy
0000 222 :
0000 223 : The above definitions are used to gather data throughput statistics in the
0000 224 : specified log file while altering buffer sizes.
0000 225 :
0000 226 :     $ DEFINE FAL$LOG "7/bpm=80"
0000 227 :
0000 228 : This definition causes the first 80 bytes of each DAP message to be dumped and
0000 229 : file identification and statistics to be displayed in the log file.
0000 230 :
0000 231 :     $ DEFINE FAL$LOG 7_50
0000 232 :
0000 233 : Same as the previous example, except the VMS V3.n parameter format of xx_yyyy
0000 234 : is used where yyyy is the number of bytes per DAP message to display expressed
0000 235 : as a hexadecimal value.
0000 236 :
0000 237 :     $ DEFINE FAL$LOG "/DISABLE=8"
0000 238 :
0000 239 : This disables poor-man's routing which prevents users from using FAL as a
0000 240 : pass-through object on this node.
0000 241 :
0000 242 :     $DEFINE FAL$LOG 2F
0000 243 :
0000 244 : This enables all FAL logging options excluding qualifier control options.
0000 245 :
0000 246 :--
```



```

0000 248      .SBTTL  DECLARATIONS
00000000 249      .PSECT  FAL$DATA_LOGGER SHR,NOEXE,RD,WRT,QUAD
0000 250
0000 251      ;
0000 252      ; Include Files:
0000 253      ;
0000 254
0000 255      $DAPHDRDEF      ; Define DAP message header
0000 256      $DAPACCDEF     ; Define DAP Access message
0000 257      $DAPCTLDEF    ; Define DAP Control message
0000 258      $FABDEF       ; Define File Access Block symbols
0000 259      $FALSTBDEF    ; Define Statistics Block symbols
0000 260      $FALWRKDEF    ; Define FAL Work Area symbols
0000 261      $JPIDEF      ; Define Job/Process Information symbols
0000 262      $NAMDEF       ; Define Name Block symbols
0000 263      $RMSDEF      ; Define RMS completion codes
0000 264
0000 265      ;
0000 266      ; Macros:
0000 267      ;
0000 268      ;     None
0000 269      ;
0000 270      ; Equated Symbols:
0000 271      ;
0000 272
0000003D 0000 273 EQUALSIGN      = ^X3D      ; ASCII code for equal sign
0000002F 0000 274 SLASH          = ^X2F      ; ASCII code for slash
00000020 0000 275 SPACE          = ^X20      ; ASCII code for space
00000009 0000 276 TAB            = ^X09      ; ASCII code for horizontal tab
0000005F 0000 277 UNDERSCORE     = ^X5F      ; ASCII code for underscore
0000 278
0000 279      ASSUME  FAL$Q_FLG EQ 0
0000 280
0000 281      ;
0000 282      ; Own Storage:
0000 283      ;
0000 284
0000 285      ;
0000 286      ; Data structures for the print file:
0000 287      ;
0000 288
0000 289      .ALIGN  LONG      ; Required for FABs and RABs
0000 290 FAL$PRTFAB::      ; File Access Block
0000 291      $FAB      FAC=PUT-  ; Put access
0000 292      FOP=SQO-    ; Sequential-only access
0000 293      RAT=CR-     ; Carriage control
0000 294      FNA=FALLST_PRTNAM- ; File name string address
0000 295      FNS=FALLSS_PRTNAM  ; File name string size
0050 296 FAL$PRTRAB::    ; Record Access Block
0050 297      $RAB      FAB=FAL$PRTFAB- ; Address of associated FAB
0050 298      RBF=0-     ; Record buffer address--t.b.s. later
0050 299      RSZ=0     ; Record buffer size--t.b.s. later
0094 300 FAL$GQ_PRTBUF1:: ; Output string descriptor for FAO
0000009C 0094 301      .BLKQ  1      ; when called from non-AST-level code
009C 302 FAL$GQ_PRTBUF2:: ; Output string descriptor for FAO
000000A4 009C 303      .BLKQ  1      ; when called from AST-level code
00A4 304 FAL$GW_PRTLEN1:: ; Formatted message length from FAO

```

```

000000A6 00A4 305          .BLKW 1
00A6 306 FAL$GW_PRTLEN2:::
000000A8 00A6 307          .BLKW 1
00A8 308 FALL$T_PRTNAM:
54 55 50 54 55 4F 24 53 59 53 00A8 309          .ASCII \SYS$OUTPUT\
0000000A 00B2 310 FALL$S_PRTNAM=-FALL$T_PRTNAM
00B2 311
00B2 312
00B2 313 : Time related storage:
00B2 314
00B2 315
00B2 316          .ALIGN LONG
00B4 317 FAL$GQ_TIME0:::
000000BC 00B4 318          .BLKW 1
00BC 319 FAL$GQ_TIME1:::
000000C4 00BC 320          .BLKW 1
00C4 321
00C4 322
00C4 323 : Item list parameter block for the $GETJPI System Service calls:
00C4 324
00C4 325
00C4 326          .ALIGN LONG
00C4 327 FAL$GETJPI_LST0:::
0004 00C4 328          .WORD 4
0407 00C6 329          .WORD JPIS_CPUTIM
00000120' 00C8 330          .LONG FALL$SL_CPUTIM0
00000000 00CC 331          .LONG 0
0004 00D0 332          .WORD 4
040C 00D2 333          .WORD JPIS_BUFIO
00000128' 00D4 334          .LONG FALL$SL_BUFIO0
00000000 00D8 335          .LONG 0
0004 00DC 336          .WORD 4
040B 00DE 337          .WORD JPIS_DIRIO
00000130' 00E0 338          .LONG FALL$SL_DIRIO0
00000000 00E4 339          .LONG 0
00000000 00E8 340          .LONG 0
00EC 341
00EC 342 FAL$GETJPI_LST1:::
0004 00EC 343          .WORD 4
0407 00EE 344          .WORD JPIS_CPUTIM
00000124' 00F0 345          .LONG FALL$SL_CPUTIM1
00000000 00F4 346          .LONG 0
0004 00F8 347          .WORD 4
040C 00FA 348          .WORD JPIS_BUFIO
0000012C' 00FC 349          .LONG FALL$SL_BUFIO1
00000000 0100 350          .LONG 0
0004 0104 351          .WORD 4
040B 0106 352          .WORD JPIS_DIRIO
00000134' 0108 353          .LONG FALL$SL_DIRIO1
00000000 010C 354          .LONG 0
0004 0110 355          .WORD 4
0201 0112 356          .WORD JPIS_WSPEAK
00000138' 0114 357          .LONG FALL$SL_WSPEAK
00000000 0118 358          .LONG 0
00000000 011C 359          .LONG 0
0120 360
0120 361 FAL$GQ_CPUTIME:

```

```

: Formatted message length from FAO
: Print device name
: Print device name length
:
: Required for FABs and RABs
: Time of link connect
: Time of link disconnect
:
:
: Start of parameter block
: Length of buffer
: Type of info to return
: Address of buffer
: Don't return length of value
: Length of buffer
: Type of info to return
: Address of buffer
: Don't return length of value
: Length of buffer
: Type of info to return
: Address of buffer
: Don't return length of value
: End of parameter block
:
: Start of parameter block
: Length of buffer
: Type of info to return
: Address of buffer
: Don't return length of value
: Length of buffer
: Type of info to return
: Address of buffer
: Don't return length of value
: Length of buffer
: Type of info to return
: Address of buffer
: Don't return length of value
: End of parameter block
: Total CPU time in delta format

```

```

00000124 0120 362 FALL$SL_CPUTIMO: ; (overlays next two longwords)
00000128 0124 363 .BLKL 1 ; Accumulated CPU time in 10 millisecond
0000012C 0128 364 .BLKL 1 ; units is returned here
00000130 012C 365 FALL$SL_CPUTIM1: ; Accumulated CPU time in 10 millisecond
00000134 0130 366 .BLKL 1 ; units is returned here
00000138 0134 367 FALL$SL_BUFIO0: ; Accumulated buffered I/O operations
0000013C 0138 368 .BLKL 1 ; count is returned here
00000140 0140 369 FALL$SL_BUFIO1: ; Accumulated buffered I/O operations
00000144 0144 370 .BLKL 1 ; count is returned here
00000148 0148 371 FALL$SL_DIRIO0: ; Accumulated direct I/O operations
0000014C 014C 372 .BLKL 1 ; count is returned here
00000150 0150 373 FALL$SL_DIRIO1: ; Accumulated direct I/O operations
00000154 0154 374 .BLKL 1 ; count is returned here
00000158 0158 375 FALL$SL_WSPEAK: ; Peak working set size
0000015C 015C 376 .BLKL 1 ;
00000160 0160 377 ;
00000164 0164 378 ;
00000168 0168 379 ; Flags to control logging printout:
0000016C 016C 380 ;
00000170 0170 381 ;
00000174 0174 382 FALL$W_PERMSG: ; Maximum # bytes per message to log
00000178 0178 383 .LONG FALS$K_DFLT_BPM ; Default value
0000017C 017C 384 FALL$W_PERLINE: ; Maximum # bytes per line to log
00000180 0180 385 .LONG FALS$K_DFLT_BPL ; Default value
00000184 0184 386 ;
00000188 0188 387 ;
0000018C 018C 388 ; Miscellaneous counter storage:
00000190 0190 389 ;
00000194 0194 390 ;
00000198 0198 391 FALS$GL_RECVWAIT:: ; Receive QIO wait count
00000200 0200 392 .LONG 0 ; Initialize to zero
00000204 0204 393 FALS$GL_XMITWAIT:: ; Transmit QIO wait count
00000208 0208 394 .LONG 0 ; Initialize to zero
00000212 0212 395 FALS$GL_READWAIT:: ; RMS FTM READ wait count
00000216 0216 396 .LONG 0 ; Initialize to zero
00000220 0220 397 FALS$GL_WRITWAIT:: ; RMS FTM WRITE wait count
00000224 0224 398 .LONG 0 ; Initialize to zero
00000228 0228 399 FALS$GL_COUNTER1:: ; Miscellaneous counter 1
00000232 0232 400 .LONG 0 ; Initialize to zero
00000236 0236 401 FALS$GL_COUNTER2:: ; Miscellaneous counter 2
00000240 0240 402 .LONG 0 ; Initialize to zero
00000244 0244 403 FALS$GL_COUNTER3:: ; Miscellaneous counter 3
00000248 0248 404 .LONG 0 ; Initialize to zero
00000252 0252 405 FALS$GL_COUNTER4:: ; Miscellaneous counter 4
00000256 0256 406 .LONG 0 ; Initialize to zero
00000260 0260 407 ;
00000264 0264 408 ;
00000268 0268 409 ; FAO related descriptor blocks with text:
00000272 0272 410 ;
00000276 0276 411 ;
00000280 0280 412 FALL$Q_MBXNAM: ; Name of associated mailbox
00000284 0284 413 $QBLOCK TEXT=<<_!AC!UW:>>
00000288 0288 414 FALS$Q_UIC: ; File owner UIC string
00000292 0292 415 $QBLOCK TEXT=<<[!30W,!30W]>>
00000296 0296 416 FALS$Q_CALLER: ; Requestor ID message
00000300 0300 417 $QBLOCK TEXT=<!!/!56*=/!/-
00000304 0304 418 <FAL !AC started execution on !23%D!/>-

```

```

0187 419 < with SYSSNET = !AS and!/>-
0187 420 < with FAL$LOG = !AS!/>-
0187 421 >
01EC 422 FAL$GQ_PARSERR: ; Parse error message
01EC 423 $QBLOCK TEXT=<<***** ERROR PARSING FAL$LOG COMMAND *****!/>>
0223 424 FAL$GQ_HEADER: ; Header line for DAP message logging
0223 425 $QBLOCK TEXT=<-
0223 426 <For DAP message dump read byte stream from right to left!/>-
0223 427 >
0265 428 FAL$GQ_LINKUP: ; Link established message
0265 429 $QBLOCK TEXT=<<Logical link was established on !23%D!/>>
0295 430 FALL$Q_REQNAM: ; Requested file name message
0295 431 $QBLOCK TEXT=<-
0295 432 <Requested file access operation: !AC!/>-
0295 433 <Specified file: !AS>-
0295 434 >
02D6 435 FALL$Q_REQNAM2: ; Requested new file name message
02D6 436 $QBLOCK TEXT=<<Change name to: !AS>>
02F1 437 FALL$Q_RESNAM: ; Resultant file name message
02F1 438 $QBLOCK TEXT=<<Resultant file: !AD>>
030C 439 FAL$GQ_STATUS: ; Status code message
030C 440 $QBLOCK TEXT=<<DAP status code of !XW generated>>
0334 441 FALL$Q_CLOSE: ; File closed message
0334 442 $QBLOCK TEXT=<<File access was terminated with !AC set on close>>
036C 443 FAL$GQ_MBXMSG: ; Mailbox message type
036C 444 $QBLOCK TEXT=<<Mailbox message type !XW received>>
0395 445 FAL$GQ_LINKDOWN: ; Link terminated message
0395 446 $QBLOCK TEXT=<<!/Logical link was terminated on !23%D>>
03C5 447 FALL$Q_CONNTIME: ; Link connect time message
03C5 448 $QBLOCK TEXT=<!/>-
03C5 449 <Total connect time for logical link was !%D!/>-
03C5 450 <Total CPU time used for connection was !%D>-
03C5 451 >
0427 452 FALL$Q_STAT1: ; Statistics message part 1
0427 453 $QBLOCK TEXT=<!/>-
0427 454 <File Access Statistics for RECV-Side XMIT-Side Composite!/>-
0427 455 <!26*- !9*- !9*- !9*-!/>-
0427 456 <# DAP Message QIO Calls !3(10UL)!/>-
0427 457 <# DAP Messages Exchanged !3(10UL)>-
0427 458 >
04C7 459 FALL$Q_STAT2: ; Statistics message part 2
04C7 460 $QBLOCK TEXT=<-
04C7 461 <# User Records/Blocks !3(10UL)!/>-
04C7 462 <# Bytes of User Data !3(10UL)!/>-
04C7 463 <# Bytes in DAP Layer !3(10UL)>-
04C7 464 >
0539 465 FALL$Q_STAT3: ; Statistics message part 3
0539 466 $QBLOCK TEXT=<-
0539 467 <User Data Throughput (bps)!3(10UL)!/>-
0539 468 <DAP Layer Throughput (bps)!3(10UL)>-
0539 469 >
0587 470 FALL$Q_STAT4: ; Statistics message part 4
0587 471 $QBLOCK TEXT=<-
0587 472 <Average Record/Block Size !3(10UL)>-
0587 473 >
05B1 474 FALL$Q_STAT5: ; Statistics message part 5
05B1 475 $QBLOCK TEXT=<-

```

```

05B1 476 <% User Data in DAP Layer !7UB.!1UB%!7UB.!1UB%!7UB.!1UB%!/>-
05B1 477 <!26*- !9*- !9*- !9*->-
05B1 478 >
0607 479 FALLSQ_STAT6: ; Statistics message part 6
0607 480 $QBLOCK TEXT=<!/=-
0607 481 <Negotiated DAP buffer size = !UW bytes!/>-
0607 482 <Buffered I/O count during connection = !UL!/>-
0607 483 <Direct I/O count during connection = !UL!/>-
0607 484 <Peak working set size for process = !UL pages>-
0607 485 >
06BE 486 FALSQ_INTCTR: ; Internal counter information
06BE 487 $QBLOCK TEXT=<!/=-
06BE 488 <Total RECV_WAIT = !UL and XMIT_WAIT not kept!/>-
06BE 489 <Total READ_WAIT not kept and WRIT_WAIT = !UL!/>-
06BE 490 <COUNTER1 = !UL and COUNTER2 = !UL!/>-
06BE 491 <COUNTER3 = !UL and COUNTER4 = !UL>-
06BE 492 >
0768 493 FALSQ_EXIT: ; Exit message
0768 494 $QBLOCK TEXT=<<!/FAL terminated execution on !%D!;!56*=!/>>
079F 495 FALLSQ_LOGMSG: ; Print DAP message (first line)
079F 496 $QBLOCK TEXT=<<!AC !AC !AC!1(6UW) -!#(3XB)>>
07C2 497 FALLSQ_LOGMSG2: ; Print DAP message (continuation lines)
07C2 498 $QBLOCK TEXT=<<!19* -!#(3XB)>>
07D7 499 FALLSQ_LOGQIO: ; Print QIO message
07D7 500 $QBLOCK TEXT=<<!11XT !AC QIO issued>>
07F3 501 FALLSQ_LOGAST: ; Print AST message
07F3 502 $QBLOCK TEXT=<<!11XT !AC AST delivered!1(6UW) bytes>>
081F 503
081F 504 ;
081F 505 ; Additional text stored in counted ASCII strings:
081F 506 ;
081F 507 ;
20 65 76 69 65 63 65 52 00' 081F 508 FALSQT_RCVQIO: ; Text for LOGAST and LOGQIO
081F 509 .ASCIC \Receive \ ;
74 69 6D 73 6E 61 72 54 00' 0828 510 FALSQT_XMTQIO: ; Text for LOGAST and LOGQIO
0828 511 .ASCIC \Transmit\ ;
20 78 6F 62 6C 69 61 4D 00' 0831 512 FALSQT_MBXQIO: ; Text for LOGAST and LOGQIO
0831 513 .ASCIC \Mailbox \ ;
2D 2D 2D 3C 00' 083A 514 FALSQT_ENCODE: ; Text for LOGMSG
083A 515 .ASCIC \<---\ ;
3E 2D 2D 2D 00' 083F 516 FALSQT_DECODE: ; Text for LOGMSG
083F 517 .ASCIC \<--->\ ;
67 73 6D 00' 0844 518 FALLST_MSG: ; Text for LOGMSG
0844 519 .ASCIC \msg\ ;
0848 520
0848 521 ;
0848 522 ; The following counted ASCII strings are arranged in an array where each string
0848 523 ; must begin one quadword apart to accomodate indexed addressing into the array.
0848 524 ;
0848 525
0848 526 .ALIGN LONG

```

```

73 74 69 62 20 6F 6E 00' 0848 527 FALLSQ_FOP OPT: ; Text for FOP options on close
07 0848 528 .ASCIC \no bits\ ; No options selected
74 69 62 20 4C 50 53 00' 0850 529 .ASCIC \SPL bit\ ; SPL bit set
07 0850
74 69 62 20 46 43 53 00' 0858 530 .ASCIC \SCF bit\ ; SCF bit set
07 0858
4C 50 53 21 46 43 53 00' 0860 531 .ASCIC \SCF!SPL\ ; SCF and SPL bits set ==> SCF
07 0860
74 69 62 20 54 4C 44 00' 0868 532 .ASCIC \DLT bit\ ; DLT bit set
07 0868
54 4C 44 21 4C 50 53 00' 0870 533 .ASCIC \SPL!DLT\ ; SPL and DLT bits set
07 0870
54 4C 44 21 46 43 53 00' 0878 534 .ASCIC \SCF!DLT\ ; SCF and DLT bits set
07 0878
74 70 6F 20 6C 6C 61 00' 0880 535 .ASCIC \all opt\ ; All three bits set ==> SCF and DLT
07 0880
0888 536
0888 537 ;
0888 538 ; The following counted ASCII strings are arranged in an array where each string
0888 539 ; must begin one octaword apart to accomodate indexed addressing into the array.
0888 540 ;
0888 541 ;
0888 542 .ALIGN LONG
0888 543 FALLST_ACCFUNC: ; Access function code text
71 65 72 20 64 69 6C 61 76 6E 49 00' 0888 544 .ASCIC \Invalid request\ ; Invalid function code
74 73 65 75 0894
0F 0888
20 20 65 6C 69 66 20 6E 65 70 4F 00' 0898 545 .ASCIC \Open file \ ; OPEN
20 20 20 20 08A4
0F 0898
65 6C 69 66 20 65 74 61 65 72 43 00' 08A8 546 .ASCIC \Create file \ ; CREATE
20 20 20 20 08B4
0F 08A8
65 6C 69 66 20 65 6D 61 6E 65 52 00' 08B8 547 .ASCIC \Rename file \ ; RENAME
20 20 20 20 08C4
0F 08B8
20 65 6C 69 66 20 65 73 61 72 45 00' 08C8 548 .ASCIC \Erase file \ ; ERASE
20 20 20 20 08D4
0F 08C8
71 65 72 20 64 69 6C 61 76 6E 49 00' 08D8 549 .ASCIC \Invalid request\ ; Reserved
74 73 65 75 08E4
0F 08D8
4C 20 79 72 6F 74 63 65 72 69 44 00' 08E8 550 .ASCIC \Directory List \ ; DIRECTORY-LIST
20 74 73 69 08F4
0F 08E8
62 75 73 28 20 65 74 61 65 72 43 00' 08F8 551 .ASCIC \Create (submit)\ ; SUBMIT
29 74 69 6D 0904
0F 08F8
75 63 65 78 65 28 20 6E 65 70 4F 00' 0908 552 .ASCIC \Open (execute) \ ; EXECUTE
20 29 65 74 0914
0F 0908
0918 553
0918 554 ;
0918 555 ; The following counted ASCII strings are arranged in an array where each string
0918 556 ; must begin one longword apart to accomodate indexed addressing into the array.
0918 557 ;

```

```

0918 558
0918 559 .ALIGN LONG
0918 560 FALL$SL_MSGTYPE:
3F 3F 3F 00' 0918 561 .ASCIC \???\ ; DAP message type text
03 0918 ; Unknown message type
46 4E 43 00' 091C 562 .ASCIC \CNF\ ; Configuration message
03 091C ;
54 54 41 00' 0920 563 .ASCIC \ATT\ ; Attributes message
03 0920 ;
43 43 41 00' 0924 564 .ASCIC \ACC\ ; Access message
03 0924 ;
4C 54 43 00' 0928 565 .ASCIC \CTL\ ; Control message
03 0928 ;
4E 4F 43 00' 092C 566 .ASCIC \CON\ ; Continue Transfer message
03 092C ;
4B 43 41 00' 0930 567 .ASCIC \ACK\ ; Acknowledge message
03 0930 ;
50 4D 43 00' 0934 568 .ASCIC \CMP\ ; Access Complete message
03 0934 ;
54 41 44 00' 0938 569 .ASCIC \DATA\ ; Data message
03 0938 ;
53 54 53 00' 093C 570 .ASCIC \STS\ ; Status message
03 093C ;
59 45 4B 00' 0940 571 .ASCIC \KEY\ ; Key Definition message
03 0940 ;
4C 4C 41 00' 0944 572 .ASCIC \ALL\ ; Allocation message
03 0944 ;
4D 55 53 00' 0948 573 .ASCIC \SUM\ ; Summary message
03 0948 ;
4D 49 54 00' 094C 574 .ASCIC \TIM\ ; Date and Time message
03 094C ;
4F 52 50 00' 0950 575 .ASCIC \PRO\ ; Protection message
03 0950 ;
4D 41 4E 00' 0954 576 .ASCIC \NAM\ ; Name message
03 0954 ;
0958 577
0958 578 ;
0958 579 ; The following counted ASCII strings are arranged in an array where each string
0958 580 ; must begin one longword apart to accomodate indexed addressing into the array.
0958 581 ;
0958 582
0958 583 .ALIGN LONG
0958 584 FALL$SL_ACCFUNC:
3F 3F 3F 00' 0958 585 .ASCIC \???\ ; Access function code text
03 0958 ; Unknown function code
4E 50 4F 00' 095C 586 .ASCIC \OPN\ ; OPEN
03 095C ;
45 52 43 00' 0960 587 .ASCIC \CRE\ ; CREATE
03 0960 ;
4E 45 52 00' 0964 588 .ASCIC \REN\ ; RENAME
03 0964 ;
41 52 45 00' 0968 589 .ASCIC \ERA\ ; ERASE
03 0968 ;
76 73 72 00' 096C 590 .ASCIC \rsv\ ; Reserved
03 096C ;
52 49 44 00' 0970 591 .ASCIC \DIR\ ; DIRECTORY-LIST
03 0970

```

```

42 55 53 00' 0974 592 .ASCIC \SUB\ ; SUBMIT
03 0974
45 58 45 00' 0978 593 .ASCIC \EXE\ ; EXECUTE
03 0978
097C 594
097C 595 ;
097C 596 ; The following counted ASCII strings are arranged in an array where each string
097C 597 ; must begin one longword apart to accomodate indexed addressing into the array.
097C 598 ;
097C 599 ;
097C 600 .ALIGN LONG
097C 601 FALL$L_CTLFUNC: ; Control function code text
3F 3F 3F 00' 097C 602 .ASCIC \???\ ; Unknown function code
03 097C
54 45 47 00' 0980 603 .ASCIC \GET\ ; GET or READ
03 0980
4E 4F 43 00' 0984 604 .ASCIC \CON\ ; CONNECT
03 0984
44 50 55 00' 0988 605 .ASCIC \UPD\ ; UPDATE
03 0988
54 55 50 00' 098C 606 .ASCIC \PUT\ ; PUT or WRITE
03 098C
4C 45 44 00' 0990 607 .ASCIC \DEL\ ; DELETE
03 0990
57 45 52 00' 0994 608 .ASCIC \REW\ ; REWIND
03 0994
55 52 54 00' 0998 609 .ASCIC \TRU\ ; TRUNCATE
03 0998
44 4F 4D 00' 099C 610 .ASCIC \MOD\ ; MODIFY (reserved)
03 099C
4C 45 52 00' 09A0 611 .ASCIC \REL\ ; RELEASE
03 09A0
45 52 46 00' 09A4 612 .ASCIC \FRE\ ; FREE
03 09A4
42 54 58 00' 09A8 613 .ASCIC \XTB\ ; EXTEND-BEGIN
03 09A8
55 4C 46 00' 09AC 614 .ASCIC \FLU\ ; FLUSH
03 09AC
56 58 4E 00' 09B0 615 .ASCIC \NXV\ ; NEXT VOLUME (reserved)
03 09B0
44 4E 46 00' 09B4 616 .ASCIC \FND\ ; FIND
03 09B4
45 54 58 00' 09B8 617 .ASCIC \XTE\ ; EXTEND-END
03 09B8
50 53 44 00' 09BC 618 .ASCIC \DSP\ ; DISPLAY
03 09BC
46 50 53 00' 09C0 619 .ASCIC \SPF\ ; SPACE FORWARD
03 09C0
42 46 53 00' 09C4 620 .ASCIC \SFB\ ; SPACE BACKWARD
03 09C4
09C8 621 ; Add strings for new codes here and
09C8 622 ; update the following constant
00000012 09C8 623 CTLFUNC_ALT_DEF=18 ; Constant to add to function code value
09C8 624 ; to index into alternate definitions
09C8 625 ; (for GET/READ = 1 and PUT/WRITE = 4)
41 45 52 00' 09C8 626 .ASCIC \REA\ ; READ (redefiniton of GET code)
03 09C8

```



```

4E 4F 43 00' 09CC 627 .ASCIC \CON\ ; CONNECT
      03 09CC
44 50 55 00' 09D0 628 .ASCIC \UPD\ ; UPDATE
      03 09D0
54 52 57 00' 09D4 629 .ASCIC \WRT\ ; WRITE (redefiniton of PUT code)
      03 09D4
      09D8 630
      09D8 631 ;
      09D8 632 ; The following counted ASCII strings are arranged in an array where each string
      09D8 633 ; must begin one longword apart to accomodate indexed addressing into the array.
      09D8 634 ;
      09D8 635 ;
      09D8 636 .ALIGN LONG
      09D8 637 FALL$$_CONFUNC: ; Continue Transfer function code text
      09D8 638 .ASCIC \???\ ; Unknown function code
3F 3F 3F 00' 09DC 639 .ASCIC \TRY\ ; TRY AGAIN
      03 09DC
52 48 53 00' 09E0 640 .ASCIC \SKR\ ; SKIP to next record
      03 09E0
4F 42 41 00' 09E4 641 .ASCIC \ABO\ ; ABORT
      03 09E4
53 45 52 00' 09E8 642 .ASCIC \RES\ ; RESUME
      03 09E8
      09EC 643
      09EC 644 ;
      09EC 645 ; The following counted ASCII strings are arranged in an array where each string
      09EC 646 ; must begin one longword apart to accomodate indexed addressing into the array.
      09EC 647 ;
      09EC 648 ;
      09EC 649 .ALIGN LONG
      09EC 650 FALL$$_CMPFUNC: ; Access Complete function code text
      09EC 651 .ASCIC \???\ ; Unknown function code
      03 09EC
53 4C 43 00' 09F0 652 .ASCIC \CLS\ ; CLOSE
      03 09F0
      0000002 09F4 653 RESPONSE_CODE=2 ; Value of response function code
50 53 52 00' 09F4 654 .ASCIC \RSP\ ; RESPONSE
      03 09F4
54 53 52 00' 09F8 655 .ASCIC \RST\ ; RESET
      03 09F8
43 53 44 00' 09FC 656 .ASCIC \DSC\ ; DISCONNECT
      03 09FC
46 48 53 00' 0A00 657 .ASCIC \SKF\ ; SKIP to next file
      03 0A00
42 48 43 00' 0A04 658 .ASCIC \CHB\ ; CHANGE-BEGIN
      03 0A04
45 48 43 00' 0A08 659 .ASCIC \CHE\ ; CHANGE-END
      03 0A08
52 45 54 00' 0A0C 660 .ASCIC \TER\ ; TERMINATE operation
      03 0A0C

```

```

00000000 662      .SBTTL FAL$PARSE_FAL$LOG - PARSE FAL$LOG STRING
00000000 663      .PSECT FAL$CODE_LOGGER NOSHR,EXE,RD,NOWRT,BYTE
00000000 664
00000000 665      :++
00000000 666      : Functional Description:
00000000 667      :
00000000 668      : FAL$PARSE_FAL$LOG parses the equivalence string from the translation
00000000 669      : of FAL$LOG and stores the results in the FAL work area and/or global
00000000 670      : storage depending on the options specified in the string.
00000000 671      :
00000000 672      : Calling Sequence:
00000000 673      :
00000000 674      :     BSBW     FAL$PARSE_FAL$LOG
00000000 675      :
00000000 676      : Input Parameters:
00000000 677      :
00000000 678      :     None
00000000 679      :
00000000 680      : Implicit Inputs:
00000000 681      :
00000000 682      :     FAL$Q_FALLOG
00000000 683      :
00000000 684      : Output Parameters:
00000000 685      :
00000000 686      :     R0-R6   Destroyed
00000000 687      :
00000000 688      : Implicit Outputs:
00000000 689      :
00000000 690      :     FAL$V_PARSE_ERR, FAL$V_DBS, FAL$V_SYS, FAL$V_VER, FAL$V_SC1, FAL$V_SC2
00000000 691      :
00000000 692      : Completion Codes:
00000000 693      :
00000000 694      :     None
00000000 695      :
00000000 696      : Side Effects:
00000000 697      :
00000000 698      :     None
00000000 699      :
00000000 700      :--
00000000 701
00000000 702 FAL$PARSE_FAL$LOG::
54 0090 C8 7D 0000 703      MOVQ   FAL$Q_FALLOG(R8),R4      ; Entry point
                                TSTL   R4      ; <R4,R5> = FAL$LOG equivalence string
                                BEQL   PARSE_NEXT      ; Branch if this is a null string
53 55 D0 0009 706      MOVL   R5,R3      ; Make copy of string address
51 85 90 000C 707 10$:  MOVB   (R5)+,R1      ; Get next character
51 20 91 000F 708      CMPB   #SPACE,R1      ; Skip over if this is a space
                                BEQL   20$      ;
51 09 91 0014 710      CMPB   #TAB,R1      ; Skip over if this is a tab
                                BEQL   20$      ;
83 51 90 0019 712      MOVB   1,(R3)+      ; Move character to buffer
0090 C8 53 0094 C8 C3 001C 713 20$: SOBGTR R4,10$      ; Branch if not end-of-string
                                SUBL3   ^L$Q_FALLOG+4(R8),R3,- ; Update FAL options string to reflect
                                ^L$Q_FALLOG(R8)      ; size of compressed string
54 0090 C8 7D 0027 716      MOVQ   FAL$Q_FALLOG(R8),R4      ; <R4,R5> = compressed FAL$LOG string
                                002C 717
                                002C 718 ;

```

```

002C 719 ; Obtain the parameter string or next qualifier string (divided into keyword
002C 720 ; and value substrings).
002C 721 ;
002C 722 ;
002C 723 PARSE_NEXT: ; Parse parameter or next qualifier
54 D5 002C 724 TSTL R4 ; Check for end-of-string and branch
01 12 002E 725 BNEQ 10$ ; if there is more string to parse
05 0030 726 RSB ; Exit from routine -----
65 2F 91 0031 727 10$: CMPB #SLASH,(R5) ; Is this start of a qualifier string?
04 12 0034 728 BNEQ 20$ ; No, it is the parameter string
54 D7 0036 729 DECL R4 ; Yes, skip over slash by adjusting
55 D6 0038 730 INCL R5 ; string descriptor
65 54 2F 3A 003A 731 20$: LOCC #SLASH,R4,(R5) ; Find end of parameter or qualifier str
52 54 50 C3 003E 732 SUBL3 R0,R4,R2 ; <R2,R3> = parameter or qualifier desc
53 53 55 D0 0042 733 MOVL R5,R3 ; string less leading slash character
54 50 7D 0045 734 MOVQ R0,R4 ; <R4,R5> = rest of string to parse desc
53 0094 C8 D1 0048 735 CMPL FAL$Q_FALLOG+4(R8),R3 ; Determine if parameter or qualifier
63 52 0F 13 004D 736 BEQL PROCESS_PARAMETER ; Branch if this is the parameter
2C 13 0053 738 BEQL #EQUALSIGN,R2,(R3) ; Determine whether or not keyword has
52 50 C2 0055 739 SUBL2 R0,R2 ; associated value string
50 D7 0058 740 DECL R0 ; <R2,R3> = qualifier keyword descriptor
51 D6 005A 741 INCL R1 ; Skip over equal sign character
23 11 005C 742 BRB PROCESS_QUALIFIER ; <R0,R1> = qualifier value descriptor
005E 743 ; Process qualifier
005E 744 ;
005E 745 ; Process the parameter bitmask which specifies FAL logging options.
005E 746 ;
005E 747 ; Note that the FAL V3.0 format of xx_yyyy is supported for compatibility where
005E 748 ; yyyy is a hexadecimal value for bytes per message to display (now /BPM=dddd).
005E 749 ;
005E 750 ;
005E 751 PROCESS_PARAMETER: ; Process FAL logging bitmask parameter
63 52 5F 8F 3A 005E 752 LOCC #UNDERSCORE,R2,(R3) ; Check for possible underline delimiter
0F 13 0063 753 BEQL 10$ ; Branch if no underline character found
52 50 C2 0065 754 SUBL2 R0,R2 ; <R2,R3> = parameter string descriptor
50 D7 0068 755 DECL R0 ; Skip over underline character
51 D6 006A 756 INCL R1 ; <R0,R1> = bytes per message descriptor
0123 30 006C 757 BSBW CONVERT_HEX_STRING ; Convert string to binary value
013C'CF 51 B0 006F 758 MOVW R1,W^FACL$W_PERMSG ; Save number of bytes per message value
50 52 7D 0074 759 10$: MOVQ R2,R0 ; <R0,R1> = parameter bitmask descriptor
0118 30 0077 760 BSBW CONVERT_HEX_STRING ; Convert string to binary value
04 AB 51 90 007A 761 MOVB R1,FAL$B_LOGGING(R8) ; Store logging flags in FAL work area
FFAB 31 007E 762 BRW PARSE_NEXT ; Parse next qualifier
0081 763 ;
0081 764 ;
0081 765 ; Process the qualifier keyword and its associated value string. The value may
0081 766 ; be either a decimal or a hexadecimal integer depending on the keyword.
0081 767 ;
0081 768 ;
0081 769 PROCESS_QUALIFIER: ; Process qualifier (keyword/value str)
56 FF A3 20202000 8F CB 0081 770 BICL3 #^X20202000,-1(R3),R6 ; Move slash and first three characters
008A 771 ; of qualifier to register upcased
56 4D50422F 8F D1 008A 772 10$: CMPL #^A\BPM\,R6 ; Check for BPM=dddd (zero is valid)
0B 12 0091 773 BNEQ 20$ ;
00EB 30 0093 774 BSBW CONVERT_DEC_STRING ; Convert string to binary value
013C'CF 51 B0 0096 775 MOVW R1,W^FACL$W_PERMSG ; Store bytes per message value

```

```

FF8E 31 009B 776 BRW PARSE_NEXT ; Parse next qualifier
      009E 777
56 4C50422F 8F D1 009E 778 20$: CMPL #^A\BPL\R6 ; Check for /BPL=dd
      OF 12 00A5 779 BNEQ 30$
      00D7 30 00A7 780 BSBW CONVERT_DEC_STRING ; Convert string to binary value
      51 B5 00AA 781 TSTW R1 ; This must be a non-zero value
      27 13 00AC 782 BEQL 35$ ; Branch if zero
0140'CF 51 B0 00AE 783 MOVW R1,W^FALL$W_PERLINE ; Store bytes per line value
      FF76 31 00B3 784 BRW PARSE_NEXT ; Parse next qualifier
      00B6 785
56 4B42522F 8F D1 00B6 786 30$: CMPL #^A\RBK\R6 ; Check for /RBK_CACHE=ddd
      1D 12 00BD 787 BNEQ 40$
      00BF 30 00BF 788 BSBW CONVERT_DEC_STRING ; Convert string to binary value
      01 51 B1 00C2 789 CMPW R1,#FAL$K_MIN_RBK ; Branch if specified value is less
      OE 1F 00C5 790 BLSSU 35$ ; than minimum allowed
007F 8F 51 B1 00C7 791 CMPW R1,#FAL$K_MAX_RBK ; Branch if specified value is greater
      07 1A 00CC 792 BGTRU 35$ ; than maximum allowed
      12 A8 51 90 00CE 793 MOVW R1,FAL$B_RBK_CACHE(R8) ; Store RMS cache block value
      FF57 31 00D2 794 BRW PARSE_NEXT ; Parse next qualifier
      FF50 31 00D5 795 35$: $SETBIT #FAL$V_PARSE_ERR,(R8) ; Denote invalid value
      00D9 796 BRW PARSE_NEXT ; Parse next qualifier
      00DC 797
56 5349442F 8F D1 00DC 798 40$: CMPL #^A\DIS\R6 ; Check for /DISABLE=xx
      OA 12 00E3 799 BNEQ 50$
      00AA 30 00E5 800 BSBW CONVERT_HEX_STRING ; Convert string to binary value
      06 A8 51 90 00E8 801 MOVW R1,FAL$B_DISABLE(R8) ; Store bitmask in FAL work area
      FF3D 31 00EC 802 BRW PARSE_NEXT ; Parse next qualifier
      00EF 803
56 414E452F 8F D1 00EF 804 50$: CMPL #^A\ENA\R6 ; Check for /ENABLE=xx
      OA 12 00F6 805 BNEQ 60$
      0097 30 00F8 806 BSBW CONVERT_HEX_STRING ; Convert string to binary value
      05 A8 51 90 00FB 807 MOVW R1,FAL$B_ENABLE(R8) ; Store bitmask in FAL work area
      FF2A 31 00FF 808 BRW PARSE_NEXT ; Parse next qualifier
      0102 809
56 5342442F 8F D1 0102 810 60$: CMPL #^A\DBS\R6 ; Check for /DBS=dddd
      OF 12 0109 811 BNEQ 70$
      0073 30 010B 812 BSBW CONVERT_DEC_STRING ; Convert string to binary value
      00A0 C8 51 B0 010E 813 MOVW R1,FAL$Q_USE_DBS(R8) ; Store buffer size in FAL work area
      FF12 31 0113 814 $SETBIT #FAL$V_USE_DBS,(R8) ; Denote DBS present
      0117 815 BRW PARSE_NEXT ; Parse next qualifier
      011A 816
56 5359532F 8F D1 011A 817 70$: CMPL #^A\SYS\R6 ; Check for /SYSTEM=xxxx
      OF 12 0121 818 BNEQ 80$
      006C 30 0123 819 BSBW CONVERT_HEX_STRING ; Convert string to binary value
      00A2 C8 51 B0 0126 820 MOVW R1,FAL$Q_USE_SYS(R8) ; Store bitmask in FAL work area
      FEFA 31 012B 821 $SETBIT #FAL$V_USE_SYS,(R8) ; Denote SYS present
      012F 822 BRW PARSE_NEXT ; Parse next qualifier
      0132 823
56 5245562F 8F D1 0132 824 80$: CMPL #^A\VER\R6 ; Check for /VERSION=xxxxxxxx
      OF 12 0139 825 BNEQ 90$
      0054 30 013B 826 BSBW CONVERT_HEX_STRING ; Convert string to binary value
      00A4 C8 51 D0 013E 827 MOVW R1,FAL$C_USE_VER(R8) ; Store bitmask in FAL work area
      FEE2 31 0143 828 $SETBIT #FAL$V_USE_VER,(R8) ; Denote VER present
      0147 829 BRW PARSE_NEXT ; Parse next qualifier
      014A 830
56 3143532F 8F D1 014A 831 90$: CMPL #^A\SC1\R6 ; Check for /SC1=xxxxxxxx
      OF 12 0151 832 BNEQ 100$

```

```

00AB C8 003C 30 0153 833      BSBW  CONVERT HEX STRING      ; Convert string to binary value
                    51  D0 0156 834      MOVL  R1,FAL$C_USE_SC1(R8)      ; Store bitmask in FAL work area
                    FECA 31 015B 835      $$SETBIT #FAL$V_USE_SC1,(R8)    ; Denote SC1 present
                    015F 836      BRW   PARSE_NEXT                    ; Parse next qualifier
                    0162 837
56  3243532F 8F D1 0162 838 100$:  CMPL  #^A\ /SC2\,R6                ; Check for /SC2=xxxxxxx
                    OF  12 0169 839      BNEQ  999$                      ;
                    0024 30 016B 840      BSBW  CONVERT HEX STRING      ; Convert string to binary value
00AC C8  51  D0 016E 841      MOVL  R1,FAL$C_USE_SC2(R8)      ; Store bitmask in FAL work area
                    0173 842      $$SETBIT #FAL$V_USE_SC2,(R8)    ; Denote SC2 present
                    FEB2 31 0177 843      BRW   PARSE_NEXT                    ; Parse next qualifier
                    017A 844 999$:  $$SETBIT #FAL$V_PARSE_ERR,(R8)    ; Denote invalid qualifier
                    FEAB 31 017E 845      BRW   PARSE_NEXT                    ; Parse next qualifier
                    0181 846
                    0181 847      ;+
                    0181 848      ; These routines convert either a decimal or a hexadecimal number to an unsigned
                    0181 849      ; binary value which is returned in R1.
                    0181 850      ; -
                    0181 851
                    0181 852 CONVERT_DEC_STRING:                      ; Convert decimal string to binary
03F4 C8  DF 0181 853      PUSHAL FAL$L_TEMP(R8)                ; Address to place converted result
                    51  DD 0185 854      PUSHL  R1                    ; Address of input string
                    00000000'GF 50 DD 0187 855      PUSHL  R0                    ; Size of input string
                    OF  11 0189 856      CALLS  #3,G^LIB$CVT_DTB          ; Perform the conversion
                    0192 857      BRB   CONVERT_COMMON                ; Join common code
                    03F4 C8  DF 0192 858 CONVERT_HEX_STRING:          ; Convert hexadecimal string to binary
                    51  DD 0196 859      PUSHAL FAL$L_TEMP(R8)                ; Address to place converted result
                    00000000'GF 50 DD 0198 860      PUSHL  R1                    ; Address of input string
                    01  BA 01AA 861      PUSHL  R0                    ; Size of input string
                    FE79 31 01B0 862      CALLS  #3,G^LIB$CVT_HTB          ; Perform the conversion
                    06 50  E9 01A1 863 CONVERT_COMMON:                ; Common conversion code
                    51 03F4 C8  D0 01A4 864      BLBC  R0,10$                    ; Branch on failure
                    01  BA 01AA 865      MOVL  FAL$L_TEMP(R8),R1          ; Return resultant binary value in R1
                    01AC 866      RSB   ; Exit
                    01  BA 01AA 867 10$:  POPR  #^M<R0>                ; Discard return address
                    01AC 868      $$SETBIT #FAL$V_PARSE_ERR,(R8)    ; Denote conversion failure
                    FE79 31 01B0 869      BRW   PARSE_NEXT                    ; Parse next qualifier

```

```

000001B3 871      .SBTTL FAL$STATISTICS - COMPUTE AND PRINT STATISTICS
01B3 872      .PSECT FAL$CODE_LOGGER NOSHR,EXE,RD,NOWRT,BYTE
01B3 873
01B3 874      :++
01B3 875      : Functional Description:
01B3 876      :
01B3 877      :     FAL$STATISTICS computes and prints statistics to the print file.
01B3 878      :
01B3 879      : Calling Sequence:
01B3 880      :
01B3 881      :     BSBW    FAL$STATISTICS
01B3 882      :
01B3 883      : Input Parameters:
01B3 884      :
01B3 885      :     None
01B3 886      :
01B3 887      : Implicit Inputs:
01B3 888      :
01B3 889      :     None
01B3 890      :
01B3 891      : Output Parameters:
01B3 892      :
01B3 893      :     R0-R7   Destroyed
01B3 894      :     R9-R11  Destroyed
01B3 895      :
01B3 896      : Implicit Outputs:
01B3 897      :
01B3 898      :     None
01B3 899      :
01B3 900      : Completion Codes:
01B3 901      :
01B3 902      :     None
01B3 903      :
01B3 904      : Side Effects:
01B3 905      :
01B3 906      :     None
01B3 907      :
01B3 908      :--
01B3 909
57 00C0 C8 DE 01B3 910 FAL$STATISTICS.:      ; Entry point
01B3 911      MOVAL  FAL$SL_STB(R8),R7      ; Get address of statistics block
01B8 912
01B8 913
01B8 914      : Compute total link connect time and store it in 64-bit delta time format.
01B8 915      : The starting and ending times are initially in 64-bit absolute time format, so
01B8 916      : delta_time = -(end_time - start_time) = (start_time - end_time).
01B8 917      :
01B8 918
00BC'CF C2 01B8 919      SUBL2  W^FAL$GQ_TIME1,-      ; Double precision subtraction of
00B4'CF 01BC 920      W^FAL$GQ_TIME0      ; ending time from starting time
00C0'CF D9 01BF 921      SBWC   W^FAL$GQ_TIME1+4,-      ; to obtain negative difference and
00B8'CF 01C3 922      W^FAL$GQ_TIME0+4      ; store in 64-bit delta format
01C6 923
01C6 924      :
01C6 925      : Compute total CPU time used while the logical link connection was in effect
01C6 926      : and store the result in 64-bit delta time format in 100 nanosecond units.
01C6 927      : The starting and ending CPU times are initially in 32-bit binary format in

```

0120'CF 50

0124'CF 50  
0120'CF 8F  
000186A0

```

01C6 928 : 10 millisecond units (or hundredths of a second). Therefore, the algorithm is
01C6 929 : delta_time = (start_units - end_units) * 100000.
01C6 930 :
01C6 931 :
01C6 932 :
01CA 933 :
01CE 934 :
01D8 935 :
01D8 936 :
01DD 937 :
01DD 938 :
01DD 939 : Write total link connect time and total CPU time used to the print file.
01DD 940 :
01DD 941 :
01DD 942 :
01DD 943 :
01DD 944 :
01DD 945 :
01DD 946 :
01DD 947 :
01FC 948 :
01FF 949 :
0202 950 :
0202 951 :
0202 952 : Total the event counters.
0202 953 :
0202 954 :
0202 955 :
0204 956 :
0207 957 :
020A 958 :
020D 959 :
0210 960 :
0213 961 :
0216 962 :
0219 963 :
021C 964 :
021F 965 :
021F 966 :
021F 967 : Write statistics header and counters to the print file.
021F 968 :
021F 969 :
021F 970 :
021F 971 :
021F 972 :
021F 973 :
021F 974 :
021F 975 :
021F 976 :
021F 977 :
021F 978 :
021F 979 :
0241 980 :
0244 981 :
0247 982 :
0247 983 :
0247 984 :

```

52 14 67 A7 C1  
53 04 18 A7 C1  
54 08 1C A7 C1  
55 0C 20 A7 C1  
56 10 24 A7 C1

```

SUBL3 W^FALL$Q_CPUTIME,- : Subtract ending time ticks from
W^FALL$Q_CPUTIME,R0 : starting time ticks to obtain
MULL3 #100000,R0,- : negative difference, then multiply
W^FALL$Q_CPUTIME : by 100000 and store result as a
MNEGL #1,W^FALL$Q_CPUTIME+4 : 64-bit delta time

```

```

$FAO_S- : Format the message
CTRSTR=W^FALL$Q_CONNTIME- : Address of FAO control string
OUTLEN=W^FALL$GW_PRTLEN1- : Address to receive string length
OUTBUF=W^FALL$GQ_PRTBUF1- : Address of buffer descriptor
P1=#FALL$GQ_TIME0- : Address of delta connect time
P2=#FALL$Q_CPUTIME : Address of delta CPU time used
$CHECK_STATUS : Check status code
BSBW FAL$PRINT_FAO : Print message

```

```

ADDL3 FAL$Q_RCV_PKT(R7),- : Total DAP message packets
FAL$Q_XMT_PKT(R7),R2 :
ADDL3 FAL$Q_RCV_MSG(R7),- : Total DAP messages
FAL$Q_XMT_MSG(R7),R3 :
ADDL3 FAL$Q_RCV_DAT(R7),- : Total user records/blocks
FAL$Q_XMT_DAT(R7),R4 :
ADDL3 FAL$Q_RCV_USR(R7),- : Total bytes of user data
FAL$Q_XMT_USR(R7),R5 :
ADDL3 FAL$Q_RCV_LNK(R7),- : Total bytes of link data
FAL$Q_XMT_LNK(R7),R6 :

```

```

$FAO_S- : Format the message
CTRSTR=W^FALL$Q_STAT1- : Address of FAO control string
OUTLEN=W^FALL$GW_PRTLEN1- : Address to receive string length
OUTBUF=W^FALL$GQ_PRTBUF1- : Address of buffer descriptor
P1=FAL$Q_RCV_PKT(R7)- : # DAP message packets received
P2=FAL$Q_XMT_PKT(R7)- : # DAP message packets transmitted
P3=R2- : # DAP messages packets exchanged
P4=FAL$Q_RCV_MSG(R7)- : # DAP messages received
P5=FAL$Q_XMT_MSG(R7)- : # DAP messages transmitted
P6=R3 : # DAP messages exchanged
$CHECK_STATUS : Check status code
BSBW FAL$PRINT_FAO : Print message
$FAO_S- : Format the message
CTRSTR=W^FALL$Q_STAT2- : Address of FAO control string
OUTLEN=W^FALL$GW_PRTLEN1- : Address to receive string length

```









```

52 012C'CF      03C8 1156      W^FALL$SL_BUFIO1,R2      :
0130'CF      C3 03CC 1157      SUBL3 W^FALL$SL_DIRIO0,-  : Find total direct I/O count
53 0134'CF      03D0 1158      W^FALL$SL_DIRIO1,R3      :
                                $FAO_S-  : Format the message
                                03D4 1159      CTRSTR=W^FALL$Q_STAT6-  : Address of control string
                                03D4 1160      OUTLEN=W^FALL$GW_PRTLEN1-: Address of receive string length
                                03D4 1161      OUTBUF=W^FALL$GQ_PRTBUF1-: Address of buffer to put string
                                03D4 1162      P1=R1-  : Negotiated DAP buffer size
                                03D4 1163      P2=R2-  : Total buffered I/O count
                                03D4 1164      P3=R3-  : Total direct I/O count
                                03D4 1165      P4=W^FALL$SL_WSPEAK     : Peak working set size
                                03F1 1166      $CHECK_STATUS          : Check status code
001E 30 03F4 1168      BSBW - FAL$PRINT_FAO    : Print message
      05 03F7 1169      RSB                    : Exit
      03F8 1170
      03F8 1171 :++
      03F8 1172 : This routine computes a percentage in xx.y format, given an integer divisor
      03F8 1173 : and an integer dividend.
      03F8 1174 :
      03F8 1175 : On input:
      03F8 1176 :
      03F8 1177 :     R0 = Divisor
      03F8 1178 :     R1 = Dividend
      03F8 1179 :
      03F8 1180 : On output:
      03F8 1181 :
      03F8 1182 :     R0 = xx part
      03F8 1183 :     R1 = y part
      03F8 1184 :--
      03F8 1185
      03F8 1186 PERCENTAGE:      : Entry point
50 50 4E 03F8 1187      CVTLF  R0,R0      : Put divisor in floating point format
      10 13 03FB 1188      BEQL   10$      : Branch if division by zero
51 0000457A 8F 44 0400 1189      CVTLF  R1,R1      : Put dividend in floating point format
      51 50 46 0407 1191      MULF2  #1000,R1  : Compute percentage x 10
      50 51 48 040A 1192      DIVF2  R0,R1      :
      51 51 D4 040D 1193 10$:  CVTRFL  R1,R0      : Round result and store as integer
51 50 50 0A 7B 040F 1194      CLRL   R1        : Prepare for double precision division
      05 0414 1195      EDIV   #10,R0,R0,R1  : Split result into xx.y format
      RSB                    : Exit

```

```

00000415 1197      .SBTTL  FAL$PRINT_FAO, FAL$PRINT_FAO_ASTLEVEL
0415 1198      .PSECT  FAL$CODE_[O]GGER NOSHR,EXE,RD,NOWRT,BYTE
0415 1199
0415 1200      :++
0415 1201      : Functional Description:
0415 1202      :
0415 1203      : FAL$PRINT_FAO outputs the buffer formatted by FAO to the print file
0415 1204      : and is called from non-AST-level code.
0415 1205      :
0415 1206      : FAL$PRINT_FAO_ASTLEVEL performs the same function, except it is called
0415 1207      : from AST-level code.
0415 1208      :
0415 1209      : Calling Sequence:
0415 1210      :
0415 1211      :     BSBW  FAL$PRINT_FAO
0415 1212      :     BSBW  FAL$PRINT_FAO_ASTLEVEL
0415 1213      :
0415 1214      : Input Parameters:
0415 1215      :
0415 1216      :     None
0415 1217      :
0415 1218      : Implicit Inputs:
0415 1219      :
0415 1220      :     FAL$GQ_PRTBUF1
0415 1221      :     FAL$GQ_PRTBUF2
0415 1222      :     FAL$GW_PRTLEN1
0415 1223      :     FAL$GW_PRTLEN2
0415 1224      :
0415 1225      : Output Parameters:
0415 1226      :
0415 1227      :     R0-R1  Destroyed
0415 1228      :
0415 1229      : Implicit Outputs:
0415 1230      :
0415 1231      :     FAL$PRTRAB is updated
0415 1232      :
0415 1233      : Completion Codes:
0415 1234      :
0415 1235      :     None
0415 1236      :
0415 1237      : Side Effects:
0415 1238      :
0415 1239      :     None
0415 1240      :
0415 1241      :--
0415 1242
0415 1243 FAL$PRINT_FAO::
51 0050'CF  DE 0415 1244      MOVAL  W^FAL$PRTRAB,R1      : Entry point
0098'CF  DO 041A 1245      MOVL   W^FAL$GQ_PRTBUF1+4,-  : Get address of print RAB
      28 A1 041E 1246      RABSL  RBF(R1)           : Update buffer address in RAB
00A4'CF  BO 0420 1247      MOVW  W^FAL$GW_PRTLEN1,-  : Update buffer size in RAB
      22 A1 0424 1248      RABSW  RSZ(R1)
0426 1249      $PUT   RAB=RT           : Write the record
042F 1250      $CHECK STATUS       : Check completion code
00A6'CF  B5 0432 1251      TSTW  W^FAL$GW_PRTLEN2    : Check for AST level PUT failure
      06 13 0436 1252      BEQL  10$                : because of PUT above in progress
0154'CF  D6 0438 1253      INCL  W^FAL$GL_COUNTER1  : Count occurrence of race condition

```

```

01 10 043C 1254 BSBF FAL$PRINT_FAO_ASTLEVEL ; Write AST level message now
05 043E 1255 10$: RSB ; Exit
043F 1256
043F 1257 FAL$PRINT_FAO_ASTLEVEL:: ; Entry point
51 0050'CF DE 043F 1258 MOVAL W^FAL$PRTRAB,R1 ; Get address of print RAB
00A0'CF D0 0444 1259 MOVL W^FAL$GQ_PRTBUF2+4,- ; Update buffer address in RAB
28 A1 0448 1260 RAB$R RBF(R1) ;
00A6'CF B0 044A 1261 MOVW W^FAL$GW_PRTLEN2,- ; Update buffer size in RAB
22 A1 044E 1262 RAB$W RSZ(R1) ;
848C 8F 50 B1 0450 1263 $PUT RAB=RT ; Write the record
8F 0E 13 0459 1264 CMPW RO,#<RMSS_BUSY&^XFFFF> ; Record stream busy? (new status code
07 13 045E 1265 BEQL 10$ ; If so, exit to let other PUT finish
82DA 8F 50 B1 0460 1266 CMPW RO,#<RMSS_RSA&^XFFFF> ; Record stream active error?
07 13 0465 1267 BEQL 10$ ; If so, exit to let other PUT finish
0467 1268 $CHECK_STATUS ; Check completion code
00A6'CF B4 046A 1269 CLRW W^FAL$GW_PRTLEN2 ; Declare AST level PUT complete
05 046E 1270 10$: RSB ; Exit

```

```

0000 046F 1272      .SBTTL  FAL$DISPLAY_MSG - DISPLAY MESSAGE BUFFER
      046F 1273      .PSECT  FAL$CODE_LOGGER NOSHR,EXE,RD,NOWRT,BYTE
      046F 1274
      046F 1275      :++
      046F 1276      : Functional Description:
      046F 1277      :
      046F 1278      :     FAL$DISPLAY formats the specified DAP message and outputs it to the
      046F 1279      :     print file. Note that for incoming messages, this routine requires
      046F 1280      :     that FAL$DECODE_MSG has already parsed the message and updated the
      046F 1281      :     DAP control block.
      046F 1282      :
      046F 1283      : Calling Sequence:
      046F 1284      :
      046F 1285      :     Call   #3,FAL$DISPLAY_MSG
      046F 1286      :
      046F 1287      : Input Parameters:
      046F 1288      :
      046F 1289      :     4(AP)  Address of header text (counted ASCII string) for message
      046F 1290      :     8(AP)  Size of the DAP message in bytes
      046F 1291      :     12(AP) Address of the DAP message
      046F 1292      :     R9     Address of DAP control block (for incoming messages)
      046F 1293      :
      046F 1294      : Implicit Inputs:
      046F 1295      :
      046F 1296      :     FALL$W_PERMSG
      046F 1297      :     FALL$W_PERLINE
      046F 1298      :
      046F 1299      : Output Parameters:
      046F 1300      :
      046F 1301      :     R0-R1  Destroyed
      046F 1302      :
      046F 1303      : Implicit Outputs:
      046F 1304      :
      046F 1305      :     None
      046F 1306      :
      046F 1307      : Completion Codes:
      046F 1308      :
      046F 1309      :     None
      046F 1310      :
      046F 1311      : Side Effects:
      046F 1312      :
      046F 1313      :     None
      046F 1314      :
      046F 1315      : --
003C 046F 1316      .ENTRY  FAL$DISPLAY_MSG,^M<R2,R3,R4,R5> ; Entry point
      0471 1318      :
      0471 1319      :
      0471 1320      : Determine total number of bytes to display of the DAP message and the number
      0471 1321      : of bytes to display on the first line of the printout.
      0471 1322      :
      0471 1323      :
      50 013C'CF 3C 0471 1324      MOVZWL  W^FALL$W_PERMSG,R0      ; Get max # bytes to display per message
      55 0140'CF 3C 0476 1325      MOVZWL  W^FALL$W_PERLINE,R5     ; Get max # bytes to display per line
      50 08 AC  D1 047B 1326      CMLPL  8(AP),R0               ; Is message size GEQ max count?
      50 08 AC  1E 047F 1327      BGEQU  10$                    ; Yes
      50 08 AC  D0 0481 1328      MOVL   8(AP),R0               ; No, use actual message size
  
```

```

53 50 D0 0485 1329 10$: MOVL R0,R3 ; Save count
55 50 D1 0488 1330 ; Branch if message will fit on one
   03 1B 048B 1331 ; line
50 55 D0 048D 1332 ; Specify count for this line
53 50 C2 0490 1333 20$: MOVL R5,R0 ; Determine count remaining after this
   0493 1334 ; line
51 50 D0 0493 1335 ; Save count for this line
   0A 13 0496 1336 ; Branch on zero length message
   0498 1337
   0498 1338
   0498 1339 ; Construct parameter list on the stack for use by $FAOL routine.
   0498 1340
   0498 1341 ; R0=R1 # bytes to print on first line
   0498 1342 ; R3 Total # bytes to display for message
   0498 1343
   0498 1344
52 0C AC D0 0498 1345 ; MOVL 12(AP),R2 ; Get address of message buffer
7E 82 9A 049C 1346 30$: MOVZBL (R2)+, -(SP) ; Put each character in list
   FA 50 F5 049F 1347 ; SOBGTR R0,30$ ; Continue until done
   51 DD 04A2 1348 40$: PUSHL R1 ; Put # bytes to convert in list
   08 AC DD 04A4 1349 ; PUSHL 8(AP) ; Put actual message size in list
50 0C BC 9A 04A7 1350 ; MOVZBL @12(AP),R0 ; Get first byte of message (type field)
   0070 30 04AB 1351 ; BSBW GET_FUNCTION ; Get function code
   54 DD 04AE 1352 ; PUSHL R4 ; Put address of string in list
   08 AC D5 04B0 1353 ; TSTL 8(AP) ; Branch on zero length message
   05 13 04B3 1354 ; BEQL 50$
   OF 50 91 04B5 1355 ; CMPB R0,#DAP$K_NAM_MSG ; Bounds check message type value
   02 1B 04B8 1356 ; BLEQU 60$ ; Branch if within range
   50 D4 04BA 1357 50$: CLRL R0 ; Treat as reserved
0918'CF40 DF 04BC 1358 60$: PUSHAL W^FALL$$_MSGTYPE[R0] ; Put address of message type text
   04C1 1359 ; in list
   04 AC DD 04C1 1360 ; PUSHL 4(AP) ; Put address of header text for message
   04C4 1361 ; in list
51 5E D0 04C4 1362 ; MOVL SP,R1 ; Get address of FAOL parameter list
   04C7 1363
   04C7 1364 ;
   04C7 1365 ; Format and print first line of message.
   04C7 1366 ;
   04C7 1367 ;
   04C7 1368 ; $FAOL_S- ; Format the message
   04C7 1369 ; CTRSTR=W^FALL$$_LOGMSG- ; Address of FAOL control string
   04C7 1370 ; OUTLEN=W^FALL$$_PRTLEN1- ; Address to receive string length
   04C7 1371 ; OUTBUF=W^FALL$$_PRTBUF1- ; Address of buffer descriptor
   04C7 1372 ; PRMLST=(R1) ; Address of parameter list
   04DC 1373 ; $CHECK_STATUS ; Check status code
   FF33 30 04DF 1374 ; BSBW FAL$PRINT_FAO ; Print message
   04E2 1375 ;
   04E2 1376 ;
   04E2 1377 ; Print additional lines as required to display the entire DAP message.
   04E2 1378 ;
   04E2 1379 ;
   04E2 1380 DISPLAY_LOOP: ;
50 53 D0 04E2 1381 ; MOVL R3,R0 ; Get # bytes remaining to display
   36 13 04E5 1382 ; BEQL DISPLAY_EXIT ; Branch if none
55 50 D1 04E7 1383 ; CMPL R0,R5 ; Branch if message will fit on one
   03 1B 04EA 1384 ; BLEQU 10$ ; line
50 55 D0 04EC 1385 ; MOVL R5,R0 ; Specify count for this line
  
```

```

53 50 C2 04EF 1386 10$:  SUBL2  R0,R3          ; Determine count remaining after this
                                04F2 1387          ; line
51 50 D0 04F2 1388          ; Save count for this line
7E 82 9A 04F5 1389 20$:  MOVL  R0,R1          ; Put each character in list
    FA 50 F5 04F8 1390      SOBGTR R0,20$      ; Continue until done
    51 DD 04FB 1391          ; Put # bytes to convert in list
51 5E D0 04FD 1392          ; Get address of FAOL parameter list
                                0500 1393          ;
                                0500 1394          ;
                                0500 1395          ; Format and print next line of message.
                                0500 1396          ;
                                0500 1397          ;
                                0500 1398          ;
                                0500 1399          ; $FAOL_S-
                                0500 1400          ; CTRSTR=W^FALL$Q_LOGMSG2-; Format the message
                                0500 1401          ; OUTLEN=W^FAL$GW_PRTLEN1-; Address of FAOL control string
                                0500 1402          ; OUTBUF=W^FAL$GQ_PRTBUF1-; Address to receive string length
                                0500 1403          ; PRMLST=(R1)                ; Address of buffer descriptor
                                0515 1404          ; $CHECK_STATUS            ; Address of parameter list
FEFA 30 0518 1404          ; Check status code
    CS 11 051B 1405          ; BSBW  FAL$PRINT FAO      ; Print message
                                051D 1406          ; BRB  DISPLAY_LOOP
                                051D 1407          ;
                                051D 1408          ; The "RET" instruction will adjust SP so that the parameter list for $FAOL
                                051D 1409          ; that was constructed on the stack is eliminated.
                                051D 1410          ;
                                051D 1411          ;
                                04 051D 1412          ; DISPLAY_EXIT:
                                051D 1413          ; RET                      ; Exit
                                051E 1414          ;
                                051E 1415          ; +
                                051E 1416          ; This routine returns the address of a counted text string that describes the
                                051E 1417          ; function code (if any) of the parsed DAP message.
                                051E 1418          ;
                                051E 1419          ; Inputs:
                                051E 1420          ; R0      Message type value
                                051E 1421          ; R9      Address of DAP control block
                                051E 1422          ; Outputs:
                                051E 1423          ; R1      Destroyed
                                051E 1424          ; R4      Address of counted ASCII text string
                                051E 1425          ; -
                                051E 1426          ;
00000040 051E 1427          ; FUNCTION_CODE=DAP$B_CTLFUNC ; Offset of function code fields in DAP
                                051E 1428          ; control block (same for all messages)
                                051E 1429          ; GET_FUNCTION:
                                051E 1430          ; CMPL  4(AP),#FAL$GT_DECODE ; Entry point
0000083F '8F 04 AC D1 051E 1430          ; BEQL  10$                  ; Branch if this is an incoming message
                                0526 1431          ; CMPB  R0,#DAP$K_CMP_MSG    ; that has just been parsed
07 50 91 0528 1432          ; BNEQ  99$                  ; Check for outgoing Access Complete
                                052B 1433          ; MOVL  #RESPONSE_CODE,R1   ; message
51 02 D0 052D 1434          ; BRB  55$                   ; It must be a response fuction code
                                0530 1435          ; 10$:  $CASEB              ; Join common code
                                0532 1436          ; SELECTOR=R0-              ; Get DAP message type code
                                0532 1437          ; BASE=#DAP$K_ACC_MSG-
                                0532 1438          ; DISPL=<-
                                0532 1439          ; 20$-
                                0532 1440          ; 30$-
                                0532 1441          ; 40$-
                                0532 1442          ; 99$-
                                ; Message type:
                                ; Access Complete
                                ; Control
                                ; Continue Transfer
                                ; Acknowledge

```



```

0532 1443          50$-          : Access Complete
0532 1444          >          :
54 0844'CF 9E 0540 1445 99$: MOVAB W^FALL$T_MSG,R4 : Get address of counted string
05 0545 1446 RSB : Exit
0546 1447
0546 1448 :
0546 1449 : Get address of text string based on message type and function code.
0546 1450 :
0546 1451 :
54 51 40 A9 9A 0546 1452 20$: MOVZBL FUNCTION_CODE(R9),R1 : Get Access function code
0958'CF41 DE 054A 1453 MOVAL W^FALL$T_ACCFUNC[R1],R4 : Get address of counted string
05 0550 1454 RSB : Exit
51 40 A9 9A 0551 1455 30$: MOVZBL FUNCTION_CODE(R9),R1 : Get Control function code
0E 13 0555 1456 BEQL 35$ : Screen out illegal code
46 A9 91 0557 1457 CMPB DAP$B_RAC(R9),- : Branch if not block I/O mode; else
04 08 1F 055A 1458 #DAP$R_BLK_VBN : convert 'GET' into 'READ' string
04 51 91 055B 1459 BLSSU 35$ : and 'PUT' into 'WRITE' string
03 1A 055D 1460 CMPB R1,#DAP$K_PUT_WRITE : if function code is within bounds
51 12 C0 0562 1462 ADDL2 #CTLFUNC_ALT_DEF,R1 : Branch if not GET, CON, UPD, or PUT
54 097C'CF41 DE 0565 1463 35$: MOVAL W^FALL$T_CTLFUNC[R1],R4 : Index into alternate definition table
05 056B 1464 RSB : Get address of counted string
54 51 40 A9 9A 056C 1465 40$: MOVZBL FUNCTION_CODE(R9),R1 : Get Continue Transfer function code
09DB'CF41 DE 0570 1466 MOVAL W^FALL$T_CONFUNC[R1],R4 : Get address of counted string
05 0576 1467 RSB : Exit
54 51 40 A9 9A 0577 1468 50$: MOVZBL FUNCTION_CODE(R9),R1 : Get Access Complete function code
09EC'CF41 DE 057B 1469 55$: MOVAL W^FALL$T_CMPFUNC[R1],R4 : Get address of counted string
05 0581 1470 RSB : Exit

```

```
00000582 1472      .SBTTL FAL$LOG_QIO, FAL$LOG_AST
0582 1473      .PSECT FAL$CODE_LOGGER NOSHR,EXE,RD,NOWRT,BYTE
0582 1474
0582 1475      :++
0582 1476      : Write QIO posted message to the print file.
0582 1477      :--
0582 1478
0582 1479 FAL$LOG_QIO::      ; Entry point
0582 1480      $FAO_S-      ; Format the message
0582 1481      CTRSTR=W^FALLSQ_LOGQIO- ; Address of FAO control string
0582 1482      OUTLEN=W^FAL$GW_PRTLEN1- ; Address to receive string length
0582 1483      OUTBUF=W^FAL$GQ_PRTBUF1- ; Address of buffer descriptor
0582 1484      P1=#0-      ; Use current time of day
0582 1485      P2=R1      ; Address of counted string
0599 1486      $CHECK_STATUS      ; Check status code
FE76 30 059C 1487      BSBW FAL$PRINT_FAO      ; Print message
059F 1488      RSB      ; Exit
05A0 1489
05A0 1490      :++
05A0 1491      : Write AST delivered message to the print file.
05A0 1492      :--
05A0 1493
05A0 1494 FAL$LOG_AST::      ; Entry point
05A0 1495      $FAO_S-      ; Format the message
05A0 1496      CTRSTR=W^FALLSQ_LOGAST- ; Address of FAO control string
05A0 1497      OUTLEN=W^FAL$GW_PRTLEN2- ; Address of receive string length
05A0 1498      OUTBUF=W^FAL$GQ_PRTBUF2- ; Address of buffer descriptor
05A0 1499      P1=#0-      ; Use current time of day
05A0 1500      P2=R1      ; Address of counted string
05A0 1501      P3=R0      ; # bytes in DAP message packet
05B9 1502      $CHECK_STATUS      ; Check status code
FE80 30 05BC 1503      BSBW FAL$PRINT_FAO_ASTLEVEL ; Print message
05BF 1504      RSB      ; Exit
```

```

0000 05C0 1506      .SBTTL FAL$LOG_REQNAM, FAL$LOG_REQNAM2
      05C0 1507      .PSECT FAL$CODE_LOGGER NOSHR,EXE,RD,NOWRT,BYTE
      05C0 1508
      05C0 1509      :++
      05C0 1510      : Write requested new file Name message to the print file.
      05C0 1511      :
      05C0 1512      : Inputs:
      05C0 1513      :   R1      Access function code
      05C0 1514      :   R2      Address of filespec string descriptor
      05C0 1515      :--
      05C0 1516
      05C0 1517 FAL$LOG_REQNAM::      : Entry point
      05C0 1518   BBC      #FAL$V_LOG_NAM,(R8),10$      : Branch if logging disabled
      05C4 1519   ASHL     #1,R1,R1      : Multiply by 2 for octaword index
      05C8 1520   MOVAQ    W^FALL$T_ACCFUNC[R1],R1      : Get address of counted string
      05CE 1521   $FAO_S-      : Format the message
      05CE 1522   CTRSTR=W^FALL$Q_REQNAM-      : Address of FAO control string
      05CE 1523   OUTLEN=W^FAL$GW_PRTLEN1-      : Address to receive string length
      05CE 1524   OUTBUF=W^FAL$GQ_PRTBUF1-      : Address of buffer descriptor
      05CE 1525   P1=R1-      : Address of counted string
      05CE 1526   P2=R2      : Address of filespec string descriptor
      05E5 1527   $CHECK_STATUS      : Check status code
      FE2A 30 05E8 1528   BSBW     FAL$PRINT_FAO      : Print message
      05EB 1529 10$:   RSB      : Exit
      05EC 1530
      05EC 1531      :++
      05EC 1532      : Write requested new file Name message to the print file.
      05EC 1533      :
      05EC 1534      : Inputs:
      05EC 1535      :   R2      Address of filespec string descriptor
      05EC 1536      :--
      05EC 1537
      1B 68 20 E1 05EC 1538 FAL$LOG_REQNAM2::      : Entry point
      05EC 1539   BBC      #FAL$V_LOG_NAM,(R8),10$      : Branch if logging disabled
      05F0 1540   $FAO_S-      : Format the message
      05F0 1541   CTRSTR=W^FALL$Q_REQNAM2-      : Address of FAO control string
      05F0 1542   OUTLEN=W^FAL$GW_PRTLEN1-      : Address to receive string length
      05F0 1543   OUTBUF=W^FAL$GQ_PRTBUF1-      : Address of buffer descriptor
      05F0 1544   P1=R2      : Address of filespec string descriptor
      FE0A 30 0605 1545   $CHECK_STATUS      : Check status code
      0608 1546   BSBW     FAL$PRINT_FAO      : Print message
      060B 1547 10$:   RSB      : Exit

```

```

060C 1549      .SBTTL FALSLOG_RESNAM, FALSLOG_CLSMMSG
060C 1550      :++
060C 1551      : Write resultant file Name message to the print file.
060C 1552      :--
060C 1553
060C 1554 FALSLOG_RESNAM::      : Entry point
35 68 20      E1 060C 1555      BBC      #FALS$V_LOG_NAM,(R8),10$ : Branch if logging disabled
50 0297 C8     9A 0610 1556      MOVZBL  FALS$L_NAM+NAM$B_RSL(R8),R0 : Get resultant string length
51 0298 C8     D0 0615 1557      MOVL   FALS$L_NAM+NAM$L_RSA(R8),R1 : Get resultant string address
  OA 68 0B     E1 061A 1558      BBC      #FALS$V_NEWNAM,(R8),5$ : Branch if not the 2nd (new) Name
50 0853 C8     9A 061E 1559      : message of rename operation
51 0854 C8     D0 061E 1560      MOVZBL  FALS$L_NAM2+NAM$B_RSL(R8),R0 : Get new resultant string length
0623 1561      MOVL   FALS$L_NAM2+NAM$L_RSA(R8),R1 : Get new resultant string address
0628 1562 5$:  $FAO_S-      : Format the message
0628 1563      CTRSTR=W^FALL$Q_RESNAM-      : Address of FAO control string
0628 1564      OUTLEN=W^FAL$GW_PRTLEN1-      : Address to receive string length
0628 1565      OUTBUF=W^FAL$GQ_PRTBUF1-      : Address of buffer descriptor
0628 1566      P1=R0-      : Resultant string length
0628 1567      P2=R1      : Resultant string address
FDD0 30 063F 1568      $CHECK_STATUS      : Check status code
      05 0642 1569      BSBW   FALS$PRINT_FAO      : Print message
0645 1570 10$:  RSB      : Exit
0646 1571
0646 1572      :++
0646 1573      : Write file close message with selected FOP options to the print file.
0646 1574      :
0646 1575      : Inputs:
0646 1576      :      R0      FOP field of the FAB
0646 1577      :--
0646 1578
0646 1579 FALSLOG_CLSMMSG::      : Entry point
0646 1580
0646 1581      ASSUME  FABS$V_SPL+1 EQ FABS$V_SCF
0646 1582      ASSUME  FABS$V_SCF+1 EQ FABS$V_DLT
0646 1583
51 26 68 20     E1 0646 1584      BBC      #FALS$V_LOG_NAM,(R8),10$ : Branch if logging disabled
51 51 03 0D     EF 064A 1585      EXTZV  #FABS$V_SPL,#3,R1,R1      : Extract DLT!SCF!SPL bits
51 0848'CF41   7E 064F 1586      MOVAQ  W^FALL$Q_FOP_OPT[R1],R1 : Get appropriate counted string
0655 1587      $FAO_S-      : Format the message
0655 1588      CTRSTR=W^FALL$Q_CLOSE-      : Address of FAO control string
0655 1589      OUTLEN=W^FAL$GW_PRTLEN1-      : Address to receive string length
0655 1590      OUTBUF=W^FAL$GQ_PRTBUF1-      : Address of buffer descriptor
0655 1591      P1=R1      : Address of counted string
FDA5 30 066A 1592      $CHECK_STATUS      : Check status code
      05 066D 1593      BSBW   FALS$PRINT_FAO      : Print message
0670 1594 10$:  RSB      : Exit
0671 1595
0671 1596      .END      : End of module

```

\$\$TAB	= 00000050	R	01	FABSL_FOP	= 00000004		
\$\$TABEND	= 00000094	R	01	FABSV_CHAN_MODE	= 00000002		
\$\$TMP	= 00000000			FABSV_CR	= 00000001		
\$\$TMP1	= 00000001			FABSV_DLT	= 0000000F		
\$\$TMP2	= 00000051			FABSV_FILE_MODE	= 00000004		
\$\$COUNT	= 00000005			FABSV_LNM_MODE	= 00000000		
\$\$T2	= 00000004			FABSV_PUT	= 00000000		
AVERAGE	00000325	R	03	FABSV_SCF	= 0000000E		
CONVERT_COMMON	000001A1	R	03	FABSV_SPL	= 0000000D		
CONVERT_DEC_STRING	00000181	R	03	FABSV_SQO	= 00000006		
CONVERT_HEX_STRING	00000192	R	03	FABSW_GBC	= 00000048		
CTLFUNC_ALT_DEF	= 00000012			FALSB_ACCFUNC	000001F6		
DAPSB_ACCFUNC	00000040			FALSB_ACCOPT	000001F5		
DAPSB_ACCOPT	00000041			FALSB_DATATYPE	000001F4		
DAPSB_BITCNT	00000035			FALSB_DISABLE	00000006		
DAPSB_BLKCNT	00000056			FALSB_ENABLE	00000005		
DAPSB_CTLFUNC	00000040			FALSB_LOGGING	00000004		
DAPSB_FAC	00000042			FALSB_MISCOPT	00000007		
DAPSB_FLAGS	00000031			FALSB_RAC	000001F7		
DAPSB_KRF	00000047			FALSB_RBK_CACHE	00000012		
DAPSB_LEN256	00000034			FALSB_RCVBUFIDX	00000011		
DAPSB_LENGTH	00000033			FALSB_VALUE	00000010		
DAPSB_RAC	00000046			FALSCHECK_STATUS	*****	X	03
DAPSB_SHR	00000043			FALSC_STBBLN	00000040		
DAPSB_STREAMID	00000032			FALSC_WRKBLN	00002000		
DAPSB_TYPE	00000030			FALSDISPLAY_MSG	0000046F	RG	03
DAPSK_ACC_MSG	= 00000003			FALSGETJPI_ESTO	000000C4	RG	01
DAPSK_BLK_VBN	= 00000004			FALSGETJPI_LST1	000000EC	RG	01
DAPSK_CMP_MSG	= 00000007			FALSGL_COUNTER1	00000154	RG	01
DAPSK_NAM_MSG	= 0000000F			FALSGL_COUNTER2	00000158	RG	01
DAPSK_PUT_WRITE	= 00000004			FALSGL_COUNTER3	0000015C	RG	01
DAPSK_SEQ_ACC	= 00000000			FALSGL_COUNTER4	00000160	RG	01
DAPSL_R0P	= 00000050			FALSGL_READWAIT	0000014C	RG	01
DAPSM_BITCNT	= 00000008			FALSGL_RECVWAIT	00000144	RG	01
DAPSM_BLKCNT	= 00000040			FALSGL_WRITWAIT	00000150	RG	01
DAPSM_DSP_3NAM	= 00000200			FALSGL_XMITWAIT	00000148	RG	01
DAPSM_GET	= 00000002			FALSGQ_CALLER	00000187	RG	01
DAPSM_GO_NOGO	= 00000010			FALSGQ_EXIT	00000768	RG	01
DAPSM_MSE	= 00000010			FALSGQ_HEADER	00000223	RG	01
DAPSM_SEGMENT	= 00000040			FALSGQ_INTCNTR	000006BE	RG	01
DAPSM_TMP1\$	= 00000008			FALSGQ_LINKDOWN	00000395	RG	01
DAPSM_TMP2\$	= FFF80000			FALSGQ_LINKUP	00000265	RG	01
DAPSQ_FILESPEC	00000044			FALSGQ_MBXMSG	0000036C	RG	01
DAPSQ_KEY	00000048			FALSGQ_PARSERR	000001EC	RG	01
DAPSQ_PASSWORD	00000050			FALSGQ_PRTBUF1	00000094	RG	01
DAPSQ_SYSPEC	00000038			FALSGQ_PRTBUF2	0000009C	RG	01
DAPSW_CTLMENU	00000044			FALSGQ_STATUS	0000030C	RG	01
DAPSW_DISPLAY1	0000004C			FALSGQ_TIME0	00000084	RG	01
DAPSW_DISPLAY2	00000054			FALSGQ_TIME1	0000008C	RG	01
DISPLAY_EXIT	0000051D	R	03	FALSGQ_UIC	00000174	RG	01
DISPLAY_LOOP	000004E2	R	03	FALSGT_DECODE	0000083F	RG	01
EQUALSIGN	= 0000003D			FALSGT_ENCODE	0000083A	RG	01
FABSC_BID	= 00000003			FALSGT_MBXQIO	00000831	RG	01
FABSC_BLN	= 00000050			FALSGT_RCVQIO	0000081F	RG	01
FABSC_SEQ	= 00000000			FALSGT_XMTQIO	00000828	RG	01
FABSC_VAR	= 00000002			FALSGW_PRTLEN1	000000A4	RG	01
FABSL_ALQ	= 00000010			FALSGW_PRTLEN2	000000A6	RG	01

FALLOGGER  
Symbol table

- FAL LOGGING ROUTINES

F 11

16-SEP-1984 01:44:38 VAX/VMS Macro V04-00  
5-SEP-1984 01:17:06 [FAL.SRC]FALLOGGER.MAR;1

Page 35  
(10)

FA  
VO

FALSK_DFLT_BPL	= 00000014			FALSQ_MBXIOSB	00000030		
FALSK_DFLT_BPM	= 00000014			FALSQ_RCV	00000040		
FALSK_MAX_RBK	= 0000007F			FALSQ_RCVIOSB	00000020		
FALSK_MIN_RBK	= 00000001			FALSQ_RMS	00000064		
FALSK_STBBLN	00000040			FALSQ_STATE_CTX	00000008		
FALSK_WRKBLN	00002000			FALSQ_SYSNET	00000098		
FALSLOG_AST	000005A0	RG	03	FALSQ_TEMP	000003F8		
FALSLOG_CLMSG	00000646	RG	03	FALSQ_VOLNAME	00000080		
FALSLOG_QIO	00000582	RG	03	FALSQ_XMT	00000048		
FALSLOG_REQNAM	000005C0	RG	03	FALSQ_XMTIOSB	00000028		
FALSLOG_REQNAM2	000005EC	RG	03	FALSSTATISTICS	000001B3	RG	03
FALSLOG_RESNAM	0000060C	RG	03	FALST_DAP	00000100		
FALSL_ACLXAB	00000C00			FALST_DIRNAME	00001F00		
FALSL_ALLXABINI	00000074			FALST_EXPAND	00000500		
FALSL_CHAIN_NXT	0000007C			FALST_EXPAND2	00000A00		
FALSL_DATXAB	00000320			FALST_FALLOG	00001C00		
FALSL_FAB	000002C0			FALST_FILESPEC	00000400		
FALSL_FAB2	00000800			FALST_FILESPEC2	00000900		
FALSL_FHCXAB	000002F4			FALST_KEYBUF	00000700		
FALSL_FOP	000001F8			FALST_MBXBUF	00001980		
FALSL_KEYNAM	00001C00			FALST_PRTBUF1	00001A00		
FALSL_KEYXAB	00001000			FALST_PRTBUF2	00001B00		
FALSL_KEYXABINI	00000078			FALST_RESULT	00000600		
FALSL_NAM	00000294			FALST_RESULT2	00000B00		
FALSL_NAM2	00000850			FALST_SYSNET	00001D00		
FALSL_NUMBER	000001FC			FALST_VOLNAME	00001E00		
FALSL_PROXAB	0000034C			FALSV_LOG_NAM	= 00000020		
FALSL_RAB	00000250			FALSV_NEWNAM	= 0000000B		
FALSL_RCVBUF	0000005C			FALSV_PARSE_ERR	= 00000038		
FALSL_RCV_DAT	00000008			FALSV_USE_DBS	= 00000039		
FALSL_RCV_LNK	00000010			FALSV_USE_SC1	= 0000003C		
FALSL_RCV_MSG	00000004			FALSV_USE_SC2	= 0000003D		
FALSL_RCV_PKT	00000000			FALSV_USE_SYS	= 0000003A		
FALSL_RCV_USR	0000000C			FALSV_USE_VER	= 0000003B		
FALSL_RDTXAB	000003B0			FALSW_DAPBUFSIZ	0000001A		
FALSL_RMS_PTR	0000006C			FALSW_DISPLAY	00000070		
FALSL_STB	000000C0			FALSW_LNKCHN	0000001C		
FALSL_SUMXAB	000003A4			FALSW_MBXCHN	0000001E		
FALSL_TEMP	000003F4			FALSW_QIOBUFSIZ	00000018		
FALSL_USE_SC1	000000A8			FALSW_RECEIVED	00000072		
FALSL_USE_SC2	000000AC			FALSW_USE_DBS	000000A0		
FALSL_USE_VER	000000A4			FALSW_USE_SYS	000000A2		
FALSL_XMT_DAT	0000001C			FALLSE_ACCFUNC	00000958	R	01
FALSL_XMT_LNK	00000024			FALLSL_BUF100	00000128	R	01
FALSL_XMT_MSG	00000018			FALLSL_BUF101	0000012C	R	01
FALSL_XMT_PKT	00000014			FALLSL_CMPFUNC	000009EC	R	01
FALSL_XMT_USR	00000020			FALLSL_CONFUNC	000009D8	R	01
FALSPARSE_FALSLOG	00000000	RG	03	FALLSL_CPUTIMO	00000120	R	01
FALSPRINT_FAO	00000415	RG	03	FALLSL_CPUTIM1	00000124	R	01
FALSPRINT_FAO_ASTLEVEL	0000043F	RG	03	FALLSL_CTLFUNC	0000097C	R	01
FALSPRTFAB	00000000	RG	01	FALLSL_DIR100	00000130	R	01
FALSPRTRAB	00000050	RG	01	FALLSL_DIR101	00000134	R	01
FALSQ_BLD	00000050			FALLSL_MSGTYPE	00000918	R	01
FALSQ_DIRNAME	00000088			FALLSL_WSPKAK	00000138	R	01
FALSQ_FALLOG	00000090			FALLSQ_CLOSE	00000334	R	01
FALSQ_FLG	00000000			FALLSQ_CONNTIME	000003C5	R	01
FALSQ_MBX	00000038			FALLSQ_CPUTIME	00000120	R	01

FALLOGGER  
Symbol table

- FAL LOGGING ROUTINES

G 11

16-SEP-1984 01:44:38  
5-SEP-1984 01:17:06

VAX/VMS Macro V04-00  
[FAL.SRC]FALLOGGER.MAR;1

Page 36  
(10)

FA  
VO

FALLSQ_FOP_OPT	00000848	R	01
FALLSQ_LOGAST	000007F3	R	01
FALLSQ_LOGMSG	0000079F	R	01
FALLSQ_LOGMSG2	000007C2	R	01
FALLSQ_LOGQIO	000007D7	R	01
FALLSQ_MBXNAM	00000164	R	01
FALLSQ_REQNAM	00000295	R	01
FALLSQ_REQNAM2	000002D6	R	01
FALLSQ_RESNAM	000002F1	R	01
FALLSQ_STAT1	00000427	R	01
FALLSQ_STAT2	000004C7	R	01
FALLSQ_STAT3	00000539	R	01
FALLSQ_STAT4	00000587	R	01
FALLSQ_STAT5	000005B1	R	01
FALLSQ_STAT6	00000607	R	01
FALLSQ_PRTNAM	= 0000000A		
FALLST_ACCFUNC	00000888	R	01
FALLST_MSG	00000844	R	01
FALLST_PRTNAM	000000A8	R	01
FALLSW_PERLINE	00000140	R	01
FALLSW_PERMSG	0000013C	R	01
FUNCTION_CODE	= 00000040		
GET_FUNCTION	0000051E	R	03
JPI\$BUFIO	= 0000040C		
JPI\$CPUTIM	= 00000407		
JPI\$DIRIO	= 0000040B		
JPI\$WSPEAK	= 00000201		
LIBSCVT_DTB	*****	X	03
LIBSCVT_HTB	*****	X	03
NAM\$B_RSL	= 00000003		
NAM\$S_RSA	= 00000004		
PARSE_NEXT	0000002C	R	03
PERCENTAGE	000003F8	R	03
PROCESS_PARAMETER	0000005E	R	03
PROCESS_QUALIFIER	00000081	R	03
RAB\$B_RAC	= 0000001E		
RAB\$C_BID	= 00000001		
RAB\$C_BLN	= 00000044		
RAB\$C_SEQ	= 00000000		
RAB\$L_CTX	= 00000018		
RAB\$L_RBF	= 00000028		
RAB\$L_ROP	= 00000004		
RAB\$W_RSZ	= 00000022		
RESPONSE_CODE	= 00000002		
RMS\$BUSY	= 0001848C		
RMS\$RSA	= 000182DA		
SLASH	= 0000002F		
SPACE	= 00000020		
SYSS\$FAO	*****	X	03
SYSS\$FAOL	*****	GX	03
SYSS\$NUMTIM	*****	GX	03
SYSS\$PUT	*****	GX	03
TAB	= 00000009		
UNDERSCORE	= 0000005F		

-----  
! 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_LOGGER	00000A10 ( 2576.)	01 ( 1.)	NOPIC USR CON REL LCL SHR NOEXE RD WRT NOVEC QUAD
\$AB\$\$	00002000 ( 8192.)	02 ( 2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
FAL\$CODE_LOGGER	00000671 ( 1649.)	03 ( 3.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE

-----  
! Performance indicators !  
-----

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.07	00:00:01.22
Command processing	113	00:00:00.37	00:00:02.17
Pass 1	449	00:00:12.67	00:00:43.61
Symbol table sort	0	00:00:01.28	00:00:07.83
Pass 2	306	00:00:03.60	00:00:12.87
Symbol table output	33	00:00:00.20	00:00:00.51
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	934	00:00:18.21	00:01:08.23

The working set limit was 1950 pages.  
111489 bytes (218 pages) of virtual memory were used to buffer the intermediate code.  
There were 80 pages of symbol table space allocated to hold 1359 non-local and 106 local symbols.  
1596 source lines were read in Pass 1, producing 33 object records in Pass 2.  
39 pages of virtual memory were used to define 32 macros.

-----  
! Macro library statistics !  
-----

Macro library name	Macros defined
_\$255\$DUA28:[FAL.OBJ]FAL.MLB;1	10
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	19
TOTALS (all libraries)	29

1718 GETS were required to define 29 macros.

There were no errors, warnings or information messages.

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



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

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

The image displays a grid of 144 small terminal window screenshots, arranged in 12 rows and 12 columns. Each window shows a different VAX/VMS command-line interface, likely representing various system utilities or their outputs. The windows are densely packed and cover most of the page area. Some larger, more prominent windows are visible, including:

- FALDECODE LIS** (top left)
- FALRMSDAP LIS** (middle right)
- FALSTATE LIS** (middle right)
- FALMAIN LIS** (middle right)
- FALCODE LIS** (middle left)
- FALLOGGER LIS** (middle left)
- FDL** (middle right)
- CREATEFDL MAP** (middle right)