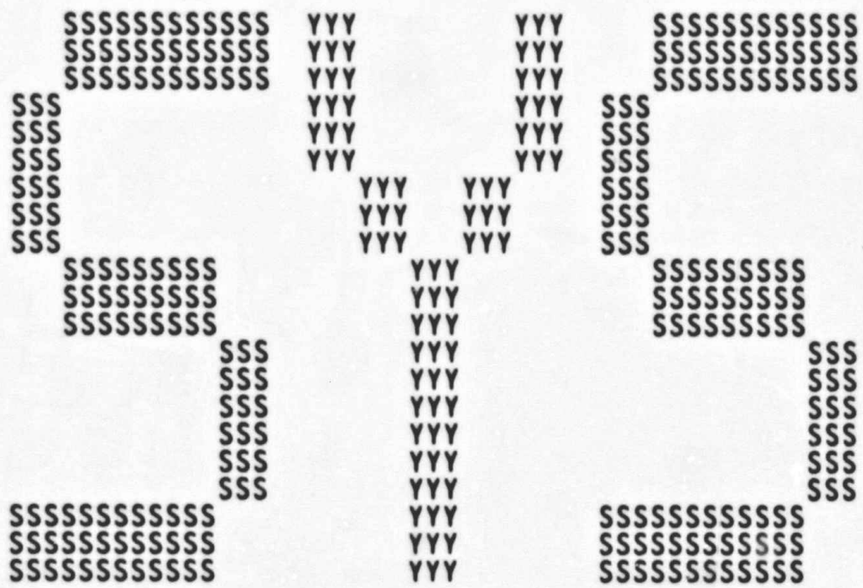
SSSSSSSSSSS YYY YYY SSSSSSSSSSS
SSSSSSSSSSS YYY YYY SSSSSSSSSSS
SSSSSSSSSSS YYY YYY SSSSSSSSSSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSSSSSSSS YYY SSSSSSSSS
SSSSSSSSS YYY SSSSSSSSS
SSSSSSSSS YYY SSSSSSSSS
SSS YYY SSS
SSS YYY SSS
SSS YYY SSS
SSS YYY SSS
SSS YYY SSS
SSS YYY SSS
SSSSSSSSSSS YYY SSSSSSSSSSS
SSSSSSSSSSS YYY SSSSSSSSSSS
SSSSSSSSSSS YYY SSSSSSSSSSS

```
  SSSSSSSS  YY      YY   SSSSSSSS    GGGGGGGG  EEEEEEEEEE  TTTTTTTTTT  MM      MM   SSSSSSSS    GGGGGGGG
  SSSSSSSS  YY      YY   SSSSSSSS    GGGGGGGG  EEEEEEEEEE  TTTTTTTTTT  MM      MM   SSSSSSSS    GGGGGGGG
SS          YY      YY  SS          GG          EE              TT      MMMM  MMMM  SS          GG
SS          YY      YY  SS          GG          EE              TT      MMMM  MMMM  SS          GG
SS            YY  YY    SS          GG          EE              TT      MM  MM  MM  SS          GG
SS            YY  YY    SS          GG          EE              TT      MM  MM  MM  SS          GG
  SSSSSS        YY        SSSSSS    GG          EEEEEEEE        TT      MM      MM    SSSSSS    GG
  SSSSSS        YY        SSSSSS    GG          EEEEEEEE        TT      MM      MM    SSSSSS    GG
        SS      YY              SS  GG  GGGGGG  EE              TT      MM      MM          SS  GG  GGGGGG
        SS      YY              SS  GG  GGGGGG  EE              TT      MM      MM          SS  GG  GGGGGG
        SS      YY              SS  GG      GG  EE              TT      MM      MM          SS  GG      GG    ....
      SS        YY              SS  GG      GG  EE              TT      MM      MM          SS  GG      GG    ....
SSSSSSSS        YY      SSSSSSSS      GGGGGG    EEEEEEEEEE      TT      MM      MM  SSSSSSSS      GGGGGG      ....
SSSSSSSS        YY      SSSSSSSS      GGGGGG    EEEEEEEEEE      TT      MM      MM  SSSSSSSS      GGGGGG      ....

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  SYSGETMSG - Get message text from message code
0000     2          .ident  'V04-000'
0000     3 ;
0000     4 ;*****************************************************************************
0000     5 ;*                                                                           *
0000     6 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     7 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     8 ;*  ALL RIGHTS RESERVED.                                                     *
0000     9 ;*                                                                           *
0000    10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    11 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    13 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    15 ;*  TRANSFERRED.                                                             *
0000    16 ;*                                                                           *
0000    17 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    18 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    19 ;*  CORPORATION.                                                             *
0000    20 ;*                                                                           *
0000    21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    23 ;*                                                                           *
0000    24 ;*                                                                           *
0000    25 ;*****************************************************************************
0000    26 ;
0000    27 ;
0000    28 ;  Author
0000    29 ;
0000    30 ;      Tim Halvorsen, Nov 1979
0000    31 ;
0000    32 ;  Modified by
0000    33 ;
0000    34 ;      V03-010 MSH0053         Michael S. Harvey       29-May-1984
0000    35 ;              Verify accessibility and length of argument list.
0000    36 ;
0000    37 ;      V03-009 LJK0278         Lawrence J. Kenah       8-May-1984
0000    38 ;              The change in LJK0275 is overkill. It is only necessary to
0000    39 ;              call an internal image activator subroutine to make sure
0000    40 ;              that message vectors hang around. Put code into separate
0000    41 ;              program section to ease word displacements.
0000    42 ;
0000    43 ;      V03-008 LJK0275         Lawrence J. Kenah       17-Apr-1984
0000    44 ;              Call SYS$IMGFIX after an indirect message section is mapped
0000    45 ;              to make sure that the message vectors are made permanent.
0000    46 ;
0000    47 ;      V03-007 BLS0257         Benn Schreiber          5-Jan-1984
0000    48 ;              If we fail to image activate a message file, do not
0000    49 ;              terminate the search.
0000    50 ;
0000    51 ;      V03-006 WMC0001         Wayne Cardoza           20-Sep-1983
0000    52 ;              Don't probe default message against user mode.
0000    53 ;
0000    54 ;      V03-005 PCG0001         Peter George            23-May-1983
0000    55 ;              Add processing for "combine" message flag.  This bit directs
0000    56 ;              that the message flags specified in the system service call
0000    57 ;              be reduced by the default process flags.
```

```
0000     58 ;
0000     59 ;       V03-004 DWT0040         David Thiel             14-Apr-1982
0000     60 ;               Temporarily remove check for fixup vector after
0000     61 ;               activating image.  This avoids problems with message
0000     62 ;               files linked as executables.
0000     63 ;
0000     64 ;       V03-003 DWT0037         David Thiel             5-Apr-1982
0000     65 ;               Return a message even if a bad message section is
0000     66 ;               found.
0000     67 ;
0000     68 ;       V03-002 DWT0035         David Thiel             31-Mar-1982
0000     69 ;               Thoroughly probe all accesses to message sections
0000     70 ;               against user mode.
0000     71 ;
0000     72 ;       V03-001 DWT0033         David Thiel             25-Mar-1982
0000     73 ;               Fix PROBEing problems and adapt to being an all-mode
0000     74 ;               service.  Fix attempt to write a possibly read-only
0000     75 ;               message section.  Start checking message sections.
0000     76 ;
0000     77 ;---
```

```
                        0000     79 ;
                        0000     80 ;       Define system definitions
                        0000     81 ;
                        0000     82
                        0000     83          $ssdef                          ; Define system status values
                        0000     84          $stsdef                         ; Define format of a message code
                        0000     85          $psldef                         ; Processor status longword definitions
                        0000     86          $plvdef                         ; Privileged vector format
                        0000     87          $mscdef                         ; Message section format
                        0000     88          $midxdef                        ; Message index format
                        0000     89          $mrecdef                        ; Message definition record
                        0000     90          $mfacdef                        ; Facility definition record
                        0000     91          $opdef                          ; Opcode definitions
                        0000     92          $iacdef                         ; Image activator flags
                        0000     93          $getmsgdef                      ; GETMSG symbols
                        0000     94
                        0000     95 ;
                        0000     96 ; Argument list offset definitions
                        0000     97 ; (5 arguments to system service)
                        0000     98 ;
                00000004 0000    99          msgid   = 4                     ; Message code to retrieve
                00000008 0000   100          msglen  = 8                     ; Address to store message length
                0000000C 0000   101          bufadr  = 12                    ; Address of buffer descriptor
                00000010 0000   102          flags   = 16                    ; Flags for selection of:
                00000000 0000   103           txtind = 0                     ; Include text portion
                00000001 0000   104           idind  = 1                     ; Include ID portion of message
                00000002 0000   105           sevind = 2                     ; Include severity portion
                00000003 0000   106           subind = 3                     ; Include subsystem name
                00000004 0000   107           combine = 4                    ; Combine specified and default flags
                00000014 0000   108          outadr  = 20                    ; Place to store longword with:
                00000000 0000   109           level  = 0                     ; Detail level of message
                00000001 0000   110           faoarg = 1                     ; Number of FAO arguments
                00000002 0000   111           user   = 2                     ; User defined value
                         0000   112
                         0000   113 ;
                         0000   114 ; Opcode definitions for code interpreter
                         0000   115 ;
                00000065 0000   116 at_r5_mode      = ^X65                      ; Mode is (r5)
                0000009F 0000   117 absolute_mode   = ^X9F                      ; Mode is @#
                00009F16 0000   118 jsb_absolute    = <absolute_mode@8> ! op$_jsb   ; Beginning of JSB @#...
                00006516 0000   119 jsb_r5          = <at_r5_mode@8> ! op$_jsb      ; JSB (R5)
                         0000   120 ;
                         0000   121 ; Local data
                         0000   122 ;
                         0000   123
                00000000 0000   124          .PSECT  YF$SYSGETMSG            ; Paged PSECT
                         0000   125 ;
                         0000   126 ; Default message if not found (ss$_msgnotfnd)
                         0000   127 ;
                         0000   128
             45 4D 41 4E 4F 4E 00' 0000   129 dfac:   .ascic  'NONAME'                ; Facility if not found
                            06    0000
                      47 53 4D 4F 4E  0007   130 dident: .ascii  'NOMSG'                 ; Ident if not found
                         00000005    000C   131 didentlen = .-dident
6D 75 6E 20 65 67 61 73 73 65 4D 00' 000C   132 dmsg:   .ascic  'Message number !XL'
             4C 58 21 20 72 65 62    0018
                               12    000C
```

```
                                001F    133
                00000050        001F    134 fakereclen = 80                             ; # bytes for fake message record
                                001F    135                                             ; if message not found
                                001F    136 inddefnam:
45 47 41 53 53 45 4D 24 53 59 53 00'    001F    137             .ascic  'SYS$MESSAGE:.EXE'      ; Default name string for IND
                   45 58 45 2E 3A       002B
                               10       001F
                                0030    138
                                0030    139 ;
                                0030    140 ; Severity table
                                0030    141 ;
                               57 0030  142 sevtab: .ascii  'W'                        ; Warning
                               53 0031  143         .ascii  'S'                        ; Success
                               45 0032  144         .ascii  'E'                        ; Error
                               49 0033  145         .ascii  'I'                        ; Informational
                               46 0034  146         .ascii  'F'                        ; Fatal error
                               3F 0035  147         .ascii  '?'
                               3F 0036  148         .ascii  '?'
                               3F 0037  149         .ascii  '?'
```

```
                        0038   151          .sbttl  Get message
                        0038   152 ;---
                        0038   153 ;
                        0038   154 ; This service provides the capability to retrieve message text from
                        0038   155 ; the system message file.
                        0038   156 ;
                        0038   157 ; Inputs:
                        0038   158 ;
                        0038   159 ;       msgid(ap)  = Message code of message to be retrieved.
                        0038   160 ;       msglen(ap) = Address to store length of result message.
                        0038   161 ;       bufadr(ap) = Address of message buffer descriptor.
                        0038   162 ;       flags(ap)  = Input flags indicating parts of message wanted.
                        0038   163 ;       outadr(ap) = Address to store message associated information.
                        0038   164 ;
                        0038   165 ; Outputs:
                        0038   166 ;
                        0038   167 ;       r0 = ss$_normal         - Normal completion.
                        0038   168 ;       r0 = ss$_accvio         - Access violation
                        0038   169 ;       r0 = ss$_msgnotfnd      - Message not in file, default message given
                        0038   170 ;       r0 = ss$_bufferovf      - Output buffer too small for message
                        0038   171 ;
                        0038   172 ; This is an all-mode service.  Therefore, arguments are not checked to
                        0038   173 ; prevent access violations; rather, an exception handler turns these into
                        0038   174 ; the appropriate return status.
                        0038   175 ; Further, this service does not assume valid message vectors.  Since
                        0038   176 ; message vector entries must be available to user mode calls to $GETMSG,
                        0038   177 ; all accesses to message vectors are probed against user mode.
                        0038   178 ;
                        0038   179 ;---
                        0038   180
                        0038   181
      50   0114 8F   3C  0038   182 insarg: movzwl  #ss$_insfarg,r0          ; not enough arguments specified
                     04 003D   183          ret
                        003E   184
                        003E   185 exe$getmsg::
                  0FFC  003E   186          .word   ^m<r2,r3,r4,r5,r6,r7,r8,r9,r10,r11>
                        0040   187
                        0040   188 ;
                        0040   189 ;       This is an all-mode service, so arguments are not PROBEd and access
                        0040   190 ;       violations can happen.
                        0040   191 ;
                        0040   192
   6D   00000000'GF   9E  0040   193          movab   g^exe$sigtoret,(fp)     ; Establish handler
                        0047   194
                        0047   195 ;
                        0047   196 ;       Verify accessibility of argument list. Also, verify that enough
                        0047   197 ;       arguments were specified.
                        0047   198 ;
                        0047   199          assume getmsg$_nargs lt 128
         05   6C   91  0047   200          cmpb    (ap),#getmsg$_nargs     ; Enough arguments specified?
                   EC 1F 004A   201          blssu   insarg                  ; If lssu, no, report error to user
           14   AC   D5  004C   202          tstl    <getmsg$_nargs*4>(ap)   ; Touch last argument, verify accessibility
                        004F   203
                        004F   204 ;
                        004F   205 ;       Preset section address for emit_facility in case no sections found.
                        004F   206 ;
                5B   D4  004F   207          clrl    r11                     ; Preset section address to zero
```

```
                                0051   208 ;
                                0051   209 ;       If the message is SS$_NORMAL and the severity is anything but 1,
                                0051   210 ;       then output NOMSG rather than "normal successful completion".
                                0051   211 ;
00   04 AC   19   03   ED       0051   212         cmpzv   #sts$v_cond_id,#sts$s_cond_id,msgid(ap),#0 ; SS$_NORMAL?
                    08   12     0057   213         bneq    5$                              ; branch if not
01   04 AC   03   00   ED       0059   214         cmpzv   #sts$v_severity,#sts$s_severity,msgid(ap),#sts$k_success
                    3E   12     005F   215         bneq    20$                             ; if not SS$_NORMAL(S), issue NOMSG
                                0061   216
                                0061   217 ;
                                0061   218 ;       Search all message sections in the current image
                                0061   219 ;
     50   00000000'GF   D0     0061   220 5$:     movl    g^ctl$gl_getmsg,r0              ; Address of message vector
          9F16 8F       80   B1 0068   221 6$:     cmpw    (r0)+,#jsb_absolute             ; Expect JSB @#
                        0C   12 006D   222         bneq    8$                              ; Something else
               58       80   D0 006F   223         movl    (r0)+,r8                        ; Address of JSB @# target
                        50   DD 0072   224         pushl   r0                              ; Save search context
                        70   10 0074   225         bsbb    search_vector                   ; Do search
                                0076   226                                                 ;  returns only if not found
                     50 8ED0    0076   227         popl    r0                              ; Get back simulated PC
                        ED   11 0079   228         brb     6$                              ; Iterate until RSB
                                007B   229
           05     FE A0   91   007B   230 8$:     cmpb    -2(r0),#op$_rsb                 ; RSB signals end of list
                        63   12 007F   231         bneq    badsec0                         ; Bad vector
                                0081   232
                                0081   233 ;
                                0081   234 ;       Search all process permanent message sections
                                0081   235 ;
     58   00000000'GF   D0     0081   236         movl    g^ctl$gl_ppmsg,r8               ; Get address of proc. perm. section
                        06   13 0088   237         beql    10$                             ; branch if none
           58   0E A8   9E   008A   238         movab   plv$l_msgdsp+6(r8),r8           ; Simulated PC
                        56   10 008E   239         bsbb    search_vector                   ; Search the vector
                                0090   240                                                 ; returns only if not found
                                0090   241 ;
                                0090   242 ;       Search the system-wide message section in system space
                                0090   243 ;
     58   00000000'GF   D0     0090   244 10$:     movl    g^exe$gl_sysmsg,r8              ; Get address of system-wide section
                        06   13 0097   245         beql    20$                             ; branch if none
           58   0E A8   9E   0099   246         movab   plv$l_msgdsp+6(r8),r8           ; Simulated PC
                        47   10 009D   247         bsbb    search_vector                   ; Search the vector
                                009F   248                                                 ; returns only if not found
                                009F   249 20$:
                                009F   250
                                009F   251 ;
                                009F   252 ;       The message was not found in any message sections.  Return
                                009F   253 ;       a default message which says that no message was found and
                                009F   254 ;       gives the actual message code.
                                009F   255 ;       R11 = address of message section containing facility name
                                009F   256 ;
           5E     B0 AE   9E   009F   257 nomsg:   movab   -fakereclen(sp),sp               ; Allocate fake mrec buffer
                59   5E   D0 00A3   258         movl    sp,r9                           ; r9 = Address of fake message record
                     02 A9   7C 00A6   259         clrq    mrec$b_type(r9)                 ; clear next 8 bytes of record
                09 A9   05   90 00A9   260         movb    #dident$len,mrec$b_ident$len(r9)  ; Set length of ident string
0A A9        FF55 CF   05   28 00AD   261         movc    #dident$len,dident,mrec$t_ident(r9) ; sets r3 for later use
                   10 A9   9F 00B4   262         pushab  mrec$c_fixed$len+dident$len+2(r9) ; Descriptor of text portion
                7E   40 8F   9A 00B7   263         movzbl  #fakereclen-mrec$c_fixed$len-dident$len-2,-(sp)
              51   FF4D CF   9E 00BB   264         movab   dmsg,r1
```

```
                50    81    9A   00C0    265          movzbl  (r1)+,r0
                7E    50    7D   00C3    266          movq    r0,-(sp)                ; Descriptor of FAO control string
                  04  AC    DD   00C6    267          pushl   msgid(ap)               ; Entire message code
                  0C  AE    9F   00C9    268          pushab  3*4(sp)                 ; Address to return length
                      6E    DD   00CC    269          pushl   (sp)                    ; Address of buffer descriptor
                  0C  AE    9F   00CE    270          pushab  3*4(sp)                 ; Address of default message string
      00000000'GF     06    FB   00D1    271          calls   #6,g^sys$fao            ; Create text of message
                83    6E    90   00D8    272          movb    (sp),(r3)+              ; Move count byte of message text
                                00DB    273                                          ; (actual text already there from fao)
                    00A5    30   00DB    274          bsbw    output_info             ; Output the information
                                00DE    275                                          ; using r11 as last section searched
             50   0621  8F    3C   00DE    276          movzwl  #ss$_msgnotfnd,r0       ; return 'not found' status (success)
                                04   00E3    277          ret                             ; Exit with status
                                00E4    278
                                00E4    279 badsec0:                                 ; Problem with message vector
                          22    11   00E4    280          brb     badsec                  ; Assist with long branch
```

```
                     00E6   282          .sbttl  search_vector, Search a given message vector
                     00E6   283 ;---
                     00E6   284 ;
                     00E6   285 ;       This routine is called by the dispatch code in a message
                     00E6   286 ;       vector.  This may be called multiple times if there is
                     00E6   287 ;       more than one image section in the program region.
                     00E6   288 ;
                     00E6   289 ; Inputs:
                     00E6   290 ;
                     00E6   291 ;       r8 = Address of message section JSB (R5)
                     00E6   292 ;       (sp) = Return address
                     00E6   293 ;
                     00E6   294 ; Outputs:
                     00E6   295 ;
                     00E6   296 ;       None
                     00E6   297 ;
                     00E6   298 ;       Return via RSB if not found, RET if found.
                     00E6   299 ;---
                     00E6   300
                     00E6   301 search_vector:
                     00E6   302          ifnord  #2,(r8),badsec,#psl$c_user ; branch if unable to read next address
6516 8F    88   B1  00EC   303          cmpw    (r8)+,#jsb_r5              ; Expect JSB (R5)
           15   12  00F1   304          bneq    badsec                     ; Bad vector
                     00F3   305 10$:     ifnord  #4,(r8),badsec,#psl$c_user ; branch if unable to read next address
     50    88   D0  00F9   306          movl    (r8)+,r0                   ; Get self-relative offset value
           09   13  00FC   307          beql    90$                        ; branch if end of list
5B   FC A840   9E  00FE   308          movab   -4(r8)[r0],r11             ; Get address of next section to search
           07   10  0103   309          bsbb    search_section             ; Search the section, RET if found
           EC   11  0105   310          brb     10$                        ; and keep going until end of list
           05   0107   311 90$:        rsb                                ; done with vector, message not found
                0108   312
                0108   313 ;
                0108   314 ;       Unable to read message vector address list
                0108   315 ;       or error activating message section image
                0108   316 ;
5B   D4  0108   317 badsec: clrl    r11                                    ; no message section
93   11  010A   318          brb     nomsg                                  ; output default message
```

```
                     010C   320          .sbttl  search_section, Search a given message section
                     010C   321 ;---
                     010C   322 ;
                     010C   323 ;       This subroutine searches a given message section for
                     010C   324 ;       the specified message code.  If the code is found, the
                     010C   325 ;       information is returned to the GETMSG caller and the
                     010C   326 ;       service is exited.  If the message is not found, return
                     010C   327 ;       via RSB.
                     010C   328 ;
                     010C   329 ; Inputs:
                     010C   330 ;
                     010C   331 ;       r11 = Address of message section to search
                     010C   332 ;
                     010C   333 ; Outputs:
                     010C   334 ;
                     010C   335 ;       None
                     010C   336 ;
                     010C   337 ;       RSB is message not found, RET if message found.
                     010C   338 ;---
                     010C   339          .enabl  lsb
                     010C   340
                     010C   341 search_section:
                     010C   342 ;
                     010C   343 ;       Verify validity of message section
                     010C   344 ;
                     010C   345          ifnord  #msc$c_length,(r11),badsec,#psl$c_user  ; Branch if cannot read head
        04D2 8F  02 AB  B1   0112   346          cmpw    msc$w_sanity(r11),#msc$c_sanity ; Check if sanity word matches
                    EE  12   0118   347          bneq    badsec                          ; branch if illegal
                     011A   348
                     011A   349          assume  msc$c_msg eq 0
                     011A   350          assume  msc$c_ind eq 1
                     011A   351          case    msc$b_type(r11),type=b,<-       ; case on section type
                     011A   352                  10$,-                           ; normal section
                     011A   353                  5$>                             ; indirect section
                        E4   11   0122   354          brb     badsec                          ; Illegal message section
                     0124   355
                     0124   356 ;
                     0124   357 ;       Process indirect message sections.  The message section
                     0124   358 ;       merely contains the name of the message file to be mapped
                     0124   359 ;       into user space.  If the secion has already been mapped,
                     0124   360 ;       then ignore the section completely.
                     0124   361 ;
                   01D9   30   0124   362 5$:      bsbw    map_indirect                    ; map the indirect file
                          05   0127   363          rsb
                     0128   364 ;
                     0128   365 ;       Process normal message sections.  Search the section
                     0128   366 ;       for the desired message code.
                     0128   367 ;
                     0128   368 10$:
                     0128   369
                     0128   370 ;
                     0128   371 ;       Compute the message number to be searched for in
                     0128   372 ;       the message sections
                     0128   373 ;
          57   04 AC   D0   0128   374          movl    msgid(ap),r7                    ; get message code
57   F0000007 8F   CA   012C   375          bicl    #sts$m_control!sts$m_severity,r7 ; clear severity (0 in index)
                     0133   376                                                  ; and control bits on top
```

```
                03 57    0F   E0  0133   377          bbs     #sts$v_fac_sp,r7,25$                    ; branch if facility specific
                   57    57   3C  0137   378          movzwl  r7,r7                                   ; if not, use fac. 0 for search
                                  013A   379 ;
                                  013A   380 ;       Verify the validity of the primary subindex
                                  013A   381 ;
                5A    08 AB    D0  013A   382 25$:     movl    msc$l_index_off(r11),r10                ; Get offset to primary index
                      5A   5B   C0  013E   383 30$:     addl    r11,r10                                 ; Get address of index
                                  0141   384           ifnord  #midx$c_length,(r10),badsec,#psl$c_user ; Must be user readable
                      56   6A   3C  0147   385           movzwl  midx$w_size(r10),r6                     ; Length of index
                      53   03   D0  014A   386           movl    #psl$c_user,r3                          ; specify access mode to maximize
                      51   56   D0  014D   387           movl    r6,r1                                   ; get length
                      50   5A   D0  0150   388           movl    r10,r0                                  ; get address of buffer
             00000000'EF       16  0153   389           jsb     exe$prober                              ; test for read access
                        AC 50   E9  0159   390           blbc    r0,badsec                               ; if lbc, no access
             7B 8F      02 AA   91  015C   391           cmpb    midx$b_sanity(r10),#midx$c_sanity       ; Check sanity byte
                           A5   12  0161   392           bneq    badsec                                  ; branch if not valid
                                  0163   393 ;
                                  0163   394 ;       Search the index for the message code
                                  0163   395 ;
                         0146   30  0163   396           bsbw    binary_search                           ; Perform binary search
                      04 51     E8  0166   397           blbs    r1,40$                                  ; branch if subindex pointer
                      07 50     E8  0169   398           blbs    r0,50$                                  ; branch if found
                                05  016C   399           rsb                                             ; exit, message not found
                                  016D   400 ;
                                  016D   401 ;       If offset points to a subindex, then go off to that
                                  016D   402 ;       subindex regardless of whether the key was matched or
                                  016D   403 ;       not, and search it as this must be a higher level index.
                                  016D   404 ;
             5A    51   01   CB  016D   405 40$:     bicl3   #1,r1,r10                               ; Compute offset to subindex
                           CB   11  0171   406           brb     30$                                     ; and start over
                                  0173   407 ;
                                  0173   408 ;       Key found - return data
                                  0173   409 ;
          59    5B   51   C1    0173   410 50$:     addl3   r1,r11,r9                               ; Address of MSG record
                                  0177   411           assume  mrec$b_identlen eq mrec$c_fixedlen
                                  0177   412           ifnord  #mrec$c_fixedlen+1,(r9),badsec,#psl$c_user      ; branch if not read
                      04   10   017D   413           bsbb    output_info                             ; Output the information
                50    01   D0   017F   414           movl    #1,r0                                   ; success
                                04   0182   415           ret
```

```
                     0183   417          .sbttl  output_info, Return information to caller's buffer
                     0183   418 ;---
                     0183   419 ;
                     0183   420 ;       This routine returns the information from a given
                     0183   421 ;       message record to the caller's buffer.
                     0183   422 ;
                     0183   423 ; Inputs:
                     0183   424 ;
                     0183   425 ;       r9 = Address of message record (fixed part already probed)
                     0183   426 ;       r11 = Address of message section
                     0183   427 ;
                     0183   428 ; Outputs:
                     0183   429 ;
                     0183   430 ;       Information is output
                     0183   431 ;---
                     0183   432          .enabl  lsb
                     0183   433
                     0183   434 output_info:
50   14 AC    D0     0183   435          movl    outadr(ap),r0                   ; Does user want assoc. info?
       04    13     0187   436          beql    60$                             ; branch if not
60   04 A9    D0     0189   437          movl    mrec$b_level(r9),(r0)           ; Move 4 byte fields
                     018D   438 ;
                     018D   439 ;       Initialize output buffer registers
                     018D   440 ;
50   0C AC    D0     018D   441 60$:     movl    bufadr(ap),r0                   ; Get address of descriptor
  56   60    3C     0191   442          movzwl  (r0),r6                         ; Get length of buffer
53   04 A0    D0     0194   443          movl    4(r0),r3                        ; Get address of buffer
                     0198   444 ;
                     0198   445 ;       If flags argument zero, then use process default flags.
                     0198   446 ;       If combine bit is set, then reduce the flags argument by the
                     0198   447 ;       default flags.
                     0198   448 ;
     57    10 AC    9A     0198   449 65$:     movzbl  flags(ap),r7            ; Get user flags
       09    12     019C   450          bneq    67$                             ; Branch if non-zero
57   00000000'GF    9A     019E   451          movzbl  g^ctl$gb_msgmask,r7     ; If zero, use process flags
       11    11     01A5   452          brb     68$                             ; Done processing flags
     0D 57    04     E1     01A7   453 67$:     bbc     #combine,r7,68$         ; Branch if no combine bit
50   00000000'GF    9A     01AB   454          movzbl  g^ctl$gb_msgmask,r0     ; Complement default flags
       50    50     D2     01B2   455          mcoml   r0,r0                   ;
       57    50     CA     01B5   456          bicl    r0,r7                   ; Clear the specified flags
                     01B8   457 ;
                     01B8   458 ;       Special case when only text indicator is set (all others off).
                     01B8   459 ;       In this case, do not return any leading punctuation.
                     01B8   460 ;
01   57   04   00     ED     01B8   461 68$:     cmpzv   #0,#4,r7,#1@txtind      ; Only wants text?
       45    13     01BD   462          beql    95$                             ; if so, skip all leading punc.
       50    25     9A     01BF   463          movzbl  #^a'%',r0               ; Output a leading '%'
             57    10     01C2   464          bsbb    put_char
                     01C4   465 ;
                     01C4   466 ;       Output facility name
                     01C4   467 ;
     08 57    03     E1     01C4   468          bbc     #subind,r7,70$          ; Branch if don't want facility
           008F    30     01C8   469          bsbw    emit_facility           ; Emit facility name
       50    2D     9A     01CB   470          movzbl  #^a'=',r0               ; Output a delimiter
             4B    10     01CE   471          bsbb    put_char
                     01D0   472 ;
                     01D0   473 ;       Output severity character
```

```
                                                L 5
                     01D0   474 ;
          13 57  02  E1  01D0   475 70$:    bbc     #sevind,r7,80$                  ; Branch if don't want severity
50  04 AC  03  00  EF  01D4   476           extzv   #sts$v_severity,#sts$s_severity,msgid(ap),r0 ; Get severity
   50  FE51 CF40  9A  01DA   477           movzbl  sevtab[r0],r0                   ; Get character corresp. to it
                39  10  01E0   478           bsbb    put_char                        ; and output it
             50  2D  9A  01E2   479           movzbl  #^a'-',r0                       ; Output a delimiter
                34  10  01E5   480           bsbb    put_char
                     01E7   481 ;
                     01E7   482 ;       Output message identification name
                     01E7   483 ;
          08 57  01  E1  01E7   484 80$:    bbc     #idind,r7,90$                   ; Branch if don't want ident
          51  09 A9  9E  01EB   485           movab   mrec$b_identlen(r9),r1          ; Address of ASCIC ident
                34  10  01EF   486           bsbb    put_string
                04  11  01F1   487           brb     92$                             ; skip adding and then removing
                     01F3   488                                                   ; the delimiter
                     01F3   489 ;
                     01F3   490 ;       Output the message text
                     01F3   491 ;
                53  D7  01F3   492 90$:    decl    r3                              ; remove previous delimiter
                56  D6  01F5   493           incl    r6
                     01F7   494 92$:    assume  txtind eq 0
          15 57  E9  01F7   495           blbc    r7,return_msglen                ; Branch if don't want text
             50  2C  9A  01FA   496           movzbl  #^a',',r0                       ; Output a comma
                1C  10  01FD   497           bsbb    put_char
             50  20  9A  01FF   498           movzbl  #^a' ',r0                       ; Output a space
                17  10  0202   499           bsbb    put_char
       51  09 A9  9A  0204   500 95$:    movzbl  mrec$b_identlen(r9),r1          ; Length of ident string
    51  0A A941  9E  0208   501           movab   mrec$t_ident(r9)[r1],r1         ; Address of ASCIC text
                16  10  020D   502           bsbb    put_string                      ; Output it
                     020F   503 ;
                     020F   504 ;       Return length of string in buffer (r6)
                     020F   505 ;
                     020F   506 return_msglen:
    51  08 AC  D0  020F   507           movl    msglen(ap),r1                   ; Result length desired?
                05  13  0213   508           beql    110$                            ; branch if not wanted
61  0C BC  56  A3  0215   509           subw3   r6,@bufadr(ap),(r1)             ; Return the length used
                05  021A   510 110$:    rsb
                     021B   511           .dsabl  lsb
```

```
                     021B   513          .sbttl  put_char, Output a single character
                     021B   514 ;---
                     021B   515 ;
                     021B   516 ;       Output a single character to the output buffer.
                     021B   517 ;
                     021B   518 ; Inputs:
                     021B   519 ;
                     021B   520 ;       r6 = Number of bytes left in buffer
                     021B   521 ;       r3 = Address of next available byte in buffer
                     021B   522 ;       r0 = Character to output
                     021B   523 ;
                     021B   524 ; Outputs:
                     021B   525 ;
                     021B   526 ;       r6 and r3 are updated.
                     021B   527 ;       If buffer overflows, return via RET
                     021B   528 ;---
                     021B   529
                     021B   530 put_char:
          56   D5    021B   531          tstl    r6                      ; Any room left in buffer?
          30   15    021D   532          bleq    bufovf                  ; branch if not
    83    50   90    021F   533          movb    r0,(r3)+                ; Insert character in buffer
          56   D7    0222   534          decl    r6                      ; and decrement space left
               05    0224   535          rsb
```

```
                     0225   537          .sbttl  put_string, Output a string
                     0225   538 ;---
                     0225   539 ;
                     0225   540 ;       Output a string to the output buffer.
                     0225   541 ;
                     0225   542 ; Inputs:
                     0225   543 ;
                     0225   544 ;       r11= Address of message section
                     0225   545 ;            0 implies default message and no probing
                     0225   546 ;       r6 = Number of bytes left in buffer
                     0225   547 ;       r3 = Address of next available byte in buffer
                     0225   548 ;       r1 = Address of counted string to be output
                     0225   549 ;
                     0225   550 ; Outputs:
                     0225   551 ;
                     0225   552 ;       r6 and r3 are updated.
                     0225   553 ;       If buffer overflows, return via RET
                     0225   554 ;---
                     0225   555
                     0225   556 put_string:
              5B D5  0225   557          tstl    r11
              06 13  0227   558          beql    10$                             ; Don't probe for the default message
                     0229   559          ifnord  #1,(r1),badsec1,#psl$c_user ; Count must be accessible
        50    81 9A  022F   560 10$:     movzbl  (r1)+,r0                        ; Get count field
              5B D5  0232   561          tstl    r11
              06 13  0234   562          beql    20$                             ; Don't probe for the default message
                     0236   563          ifnord  r0,(r1),badsec1,#psl$c_user ; Text must be accessible
        56    50 D1  023C   564 20$:     cmpl    r0,r6                           ; Enough room in buffer?
              08 14  023F   565          bgtr    50$                             ; branch if not
        56    50 C2  0241   566          subl    r0,r6                           ; Decrement space left
     63 61    50 28  0244   567          movc    r0,(r1),(r3)                    ; Move string into buffer
              05     0248   568          rsb
                     0249   569
     63 61    56 28  0249   570 50$:     movc    r6,(r1),(r3)                    ; overflow, only do what we can
              56 D4  024D   571          clrl    r6                              ; and decrement length used
                     024F   572
              BE 10  024F   573 bufovf: bsbb    return_msglen                   ; return string length
     50 0601  8F 3C  0251   574          movzwl  #ss$_bufferovf,r0               ; set error status
              04     0256   575          ret                                     ; and exit service
                     0257   576
                     0257   577 badsec1:                                        ; invalid message section
           FEAE      31 0257   578          brw     badsec                          ; long branch
```

```
                                                B 6
                     025A   580          .sbttl  emit_facility, Emit the facility name
                     025A   581 ;---
                     025A   582 ;
                     025A   583 ;       Output the facility name to the output buffer.
                     025A   584 ;
                     025A   585 ; Inputs:
                     025A   586 ;
                     025A   587 ;       msgid(ap) = Message code
                     025A   588 ;       r11 = Address of message section (0 if none)
                     025A   589 ;       r6/r3 = Descriptor of buffer space left
                     025A   590 ;
                     025A   591 ; Outputs:
                     025A   592 ;
                     025A   593 ;       r6/r3 are updated.
                     025A   594 ;       r0-r2 destroyed.
                     025A   595 ;       If buffer overflows, return via RET
                     025A   596 ;---
                     025A   597
                     025A   598 emit_facility:
            30   BB  025A   599          pushr   #^m<r4,r5>                      ; save registers
            5B   D5  025C   600          tstl    r11                             ; any section found yet?
            3B   13  025E   601          beql    20$                             ; branch if none at all
     50     5B   0C  AB C1  0260   602          addl3   msc$l_fac_off(r11),r11,r0 ; Address of facility table
     52     5B   04  AB C1  0265   603          addl3   msc$l_size(r11),r11,r2  ; Address of end of section
          0C     10  EF      026A   604          extzv   #sts$v_fac_no,#sts$s_fac_no,-   ; Get facility number of msg
          55   04 AC          026D   605                  msgid(ap),r5
                           0270   606          ifnord  #2,(r0),badsec1,#psl$c_user ; Must be user readable
            54   80  3C  0276   607          movzwl  (r0)+,r4                        ; Get size of table
            20   13  0279   608          beql    20$                             ; branch if empty table
            52   50  D1  027B   609 10$:     cmpl    r0,r2                           ; Exceeded end of section?
            1B   1E  027E   610          bgequ   20$                             ; branch if so
                     0280   611          assume  mfac$w_number le mfac$b_namelen
                     0280   612          ifnord  #mfac$b_namelen,(r0),badsec1,#psl$c_user ; Must be user readable
          60   55  B1  0286   613          cmpw    r5,mfac$w_number(r0)            ; Does number match?
            17   13  0289   614          beql    30$                             ; branch if match found
        51   02  A0  028B   615          movzbl  mfac$b_namelen(r0),r1           ; get length of facility name
        51   03  A1  028F   616          movab   mfac$t_name(r1),r1              ; get length of entire entry
          50   51  C0  0293   617          addl    r1,r0                           ; skip to next facility definition
          54   51  C2  0296   618          subl    r1,r4                           ; decrement length of table left
               E0  12  0299   619          bneq    10$                             ; branch if more to go
   51      FD61 CF  9E  029B   620 20$:     movab   w^dfac,r1                       ; Name not found - use default
               04  11  02A0   621          brb     40$
        51   02  A0  9E  02A2   622 30$:     movab   mfac$b_namelen(r0),r1           ; Address of ASCIC facility name
             FF7C  30  02A6   623 40$:     bsbw    put_string                      ; Output the string
               30  BA  02A9   624          popr    #^m<r4,r5>                      ; restore registers
                   05  02AB   625          rsb
```

```
                                02AC   627          .sbttl  binary_search, Perform binary search on index
                                02AC   628 ;---
                                02AC   629 ;
                                02AC   630 ;       Perform a binary search on the specified message index
                                02AC   631 ;       searching for the given message code.
                                02AC   632 ;
                                02AC   633 ; Inputs:
                                02AC   634 ;
                                02AC   635 ;       r10 = Address of message (sub)index
                                02AC   636 ;       r6 = Length of message (sub)index
                                02AC   637 ;       r7 = Desired message code
                                02AC   638 ;
                                02AC   639 ; Outputs:
                                02AC   640 ;
                                02AC   641 ;       r0 = Success/failure flag
                                02AC   642 ;       r1 = Offset to message definition record
                                02AC   643 ;               (actually, the contents of the second longword of the entry)
                                02AC   644 ;
                                02AC   645 ;       r2-r4 destroyed.
                                02AC   646 ;
                                02AC   647 ;       If the entry is not found, and the bottom bit of r1 is set, then
                                02AC   648 ;       the offset points to a subindex rather than the definition record
                                02AC   649 ;       and the search should be continued with that subindex.
                                02AC   650 ;---
                                02AC   651
                                02AC   652 binary_search:
                          55    DD    02AC   653          pushl   r5                          ; save register
                          52    D4    02AE   654          clrl    r2                          ; low limit = 0
                    53    56    10    C3    02B0   655          subl3   #midx$c_length+8,r6,r3  ; Compute index length minus overhead
                                      02B4   656                                              ; also minus 1 entry (the 8)
                53    53      FD    8F    78    02B4   657          ashl    #-3,r3,r3           ; Convert to number of entries - 1
                          55    53    D0    02B9   658          movl    r3,r5               ; save maximum entry #
                          51    D4    02BC   659          clrl    r1                          ; preset r1 to lbc in case of empty
                                      02BE   660                                              ; index, you don't get subindex ptr
                    54    53    52    C1    02BE   661 10$:     addl3   r2,r3,r4            ; average low and high bounds
                54    54      FF    8F    78    02C2   662          ashl    #-1,r4,r4           ; and divide by two
                          53    52    D1    02C7   663          cmpl    r2,r3               ; if low>high, end of search
                          19    14    02CA   664          bgtr    40$                 ; branch if end of search
                50    08  AA44    7D    02CC   665          movq    midx$c_entries(r10)[r4],r0      ; Get entire entry in r0/r1
                50    57    50    C3    02D1   666          subl3   r0,r7,r0            ; check if key matches
                                      02D5   667                                              ; save comparison result in r0
                          22    13    02D5   668          beql    50$                 ; branch if found
                          06    1A    02D7   669          bgtru   20$                 ; branch if in upper half, set new low
                    53    54    01    C3    02D9   670          subl3   #1,r4,r3            ; set new upper limit
                          DF    11    02DD   671          brb     10$                 ; and continue
                    52    54    01    C1    02DF   672 20$:     addl3   #1,r4,r2            ; set new lower limit
                          D9    11    02E3   673          brb     10$                 ; and continue
                                      02E5   674 ;
                                      02E5   675 ;       Key not found.  If the last comparison shows that
                                      02E5   676 ;       the desired key falls before the key we are sitting on,
                                      02E5   677 ;       then return the following key value.  We guarantee that
                                      02E5   678 ;       the entry we return will always have a key value greater
                                      02E5   679 ;       than the desired key.  This is because the index is structured
                                      02E5   680 ;       so that an index which points to subindices use a key value
                                      02E5   681 ;       of the highest key on that subindex.  In order to follow the
                                      02E5   682 ;       tree down successfully, we must return the subindex with the
                                      02E5   683 ;       higher key.
```

```
                02E5   684 ;
          50    D5 02E5   685 40$:    tstl    r0                          ; if desired key is GTR key tested
          0C    15 02E7   686         bleq    45$                         ; then execute following code
          54    D6 02E9   687         incl    r4                          ; use entry of maximal key value
       55 54    D1 02EB   688         cmpl    r4,r5                       ; off the end of the index?
          05    18 02EE   689         bgeq    45$                         ; if so, leave on last entry in index
50 08 AA44    7D 02F0   690         movq    midx$c_entries(r10)[r4],r0 ; return offset assoc. with entry
          50    D4 02F5   691 45$:    clrl    r0                          ; failure - key not found
                02F7   692                                               ; return with r1 of closest entry
          03    11 02F7   693         brb     90$
                02F9   694 ;
                02F9   695 ;       Key found - return success.
                02F9   696 ;
       50 01    D0 02F9   697 50$:    movl    #1,r0                       ; success - key found
       55 8ED0  02FC   698 90$:    popl    r5                          ; restore register
             05 02FF   699         rsb
```

```
                     0300   701          .sbttl  map_indirect, Map an indirect message file
                     0300   702 ;---
                     0300   703 ;
                     0300   704 ;       This routine processes an indirect message section.
                     0300   705 ;       It attempts to map the message file named in the
                     0300   706 ;       message section into user space and marks the section
                     0300   707 ;       processed so that the file will not be mapped twice.
                     0300   708 ;       When the message file is mapped via $IMGACT, the
                     0300   709 ;       addresses of the message sections contained in the
                     0300   710 ;       file will be added to the end of the message vector
                     0300   711 ;       list and will be processed later in the search.
                     0300   712 ;
                     0300   713 ; Inputs:
                     0300   714 ;
                     0300   715 ;       r11 = Address of message section
                     0300   716 ;
                     0300   717 ; Outputs:
                     0300   718 ;
                     0300   719 ;       None, return via RET if any fatal errors
                     0300   720 ;---
                     0300   721
                     0300   722 map_indirect:
                     0300   723          ifnowrt #msc$c_length,msc$b_flags(r11),70$,#psl$c_user  ; check access
59 01 AB    00   E2 0307   724          bbss    #msc$v_mapped,msc$b_flags(r11),40$ ; skip if already mapped
   53    08 AB   9E 030C   725          movab   msc$b_indnamlen(r11),r3 ; Address of ASCIC string
      52    83   9A 0310   726          movzbl  (r3)+,r2                ; Descriptor in r2/r3
   51    FD08 CF   9E 0313   727          movab   inddefnam,r1            ; address of default name string
      50    81   9A 0318   728          movzbl  (r1)+,r0                ; get length of string
   5E    FE00 CE   9E 031B   729          movab   -512(sp),sp             ; allocate image activator work area
      7E    50   7D 0320   730          movq    r0,-(sp)                ; put default descriptor on stack
      7E    52   7D 0323   731          movq    r2,-(sp)                ; put input descriptor on stack
            7E   7C 0326   732          clrq    -(sp)                   ; specify program region 0
      51    5E   D0 0328   733          movl    sp,r1                   ; get address of stack
                   032B   734          $imgact_s name=8(r1),-          ; merge image into address space
                   032B   735                  dflnam=16(r1),-         ; using default name string
                   032B   736                  inadr=(r1),-            ; using dummy inadr (to give region)
                   032B   737                  retadr=(r1),-           ; return address here also
                   032B   738                  hdrbuf=24(r1),-         ; address of work area
                   032B   739                  imgctl=#iac$m_merge!iac$m_expreg ; merge into address space
5E   0218 CE   9E 0345   740          movab   24+512(sp),sp           ; deallocate descriptors and work page
                   034A   741          $imgact_s       0,0,0,-
                   034A   742                  imgctl=#iac$m_setvector ; let image activator set vectors
            05     0365   743 40$:    rsb                             ; and exit
                   0366   744 ;
                   0366   745 ;       Error accessing mapped bit or mapping message file
                   0366   746 ;
                   0366   747 70$:
      FD9F   31    0366   748 80$:    brw     badsec                  ; exit with default message
                   0369   749
                   0369   750          .end
```

```
$$ARGS                  =  00000005
$$T1                    =  00000000
ABSOLUTE_MODE           =  0000009F
AT_R5_MODE              =  00000065
BADSEC                     00000108  R      02
BADSEC0                    000000E4  R      02
BADSEC1                    00000257  R      02
BINARY_SEARCH              000002AC  R      02
BUFADR                  =  0000000C
BUFOVF                     0000024F  R      02
COMBINE                 =  00000004
CTL$GB_MSGMASK             ********    X    02
CTL$GL_GETMSG              ********    X    02
CTL$GL_PPMSG               ********    X    02
DFAC                       00000000  R      02
DIDENT                     00000007  R      02
DIDENTLEN               =  00000005
DMSG                       0000000C  R      02
EMIT_FACILITY              0000025A  R      02
EXE$GETMSG                 0000003E  RG     02
EXE$GL_SYSMSG              ********    X    02
EXE$PROBER                 ********    X    02
EXE$SIGTORET               ********    X    02
FAKERECLEN              =  00000050
FAOARG                  =  00000001
FLAGS                   =  00000010
GETMSG$_BUFADR          =  0000000C
GETMSG$_FLAGS           =  00000010
GETMSG$_MSGID           =  00000004
GETMSG$_MSGLEN          =  00000008
GETMSG$_NARGS           =  00000005
GETMSG$_OUTADR          =  00000014
IAC$M_EXPREG            =  00000020
IAC$M_MERGE             =  00000010
IAC$M_SETVECTOR         =  00200000
IDIND                   =  00000001
INDDEFNAM                  0000001F  R      02
INSARG                     00000038  R      02
JSB_ABSOLUTE            =  00009F16
JSB_R5                  =  00006516
LEVEL                   =  00000000
MAP_INDIRECT               00000300  R      02
MFAC$B_NAMELEN          =  00000002
MFAC$T_NAME             =  00000003
MFAC$W_NUMBER           =  00000000
MIDX$B_SANITY           =  00000002
MIDX$C_ENTRIES          =  00000008
MIDX$C_LENGTH           =  00000008
MIDX$C_SANITY           =  0000007B
MIDX$W_SIZE             =  00000000
MREC$B_IDENTLEN         =  00000009
MREC$B_LEVEL            =  00000004
MREC$B_TYPE             =  00000002
MREC$C_FIXEDLEN         =  00000009
MREC$T_IDENT            =  0000000A
MSC$B_FLAGS             =  00000001
MSC$B_INDNAMLEN         =  00000008
MSC$B_TYPE              =  00000000
MSC$C_IND               =  00000001
MSC$C_LENGTH            =  00000028
MSC$C_MSG               =  00000000
MSC$C_SANITY            =  000004D2
MSC$L_FAC_OFF           =  0000000C
MSC$L_INDEX_OFF         =  00000008
MSC$L_SIZE              =  00000004
MSC$V_MAPPED            =  00000000
MSC$W_SANITY            =  00000002
MSGID                   =  00000004
MSGLEN                  =  00000008
NOMSG                      0000009F  R      02
OP$_JSB                 =  00000016
OP$_RSB                 =  00000005
OUTADR                  =  00000014
OUTPUT_INFO                00000183  R      02
PLV$L_MSGDSP            =  00000008
PSL$C_USER              =  00000003
PUT_CHAR                   0000021B  R      02
PUT_STRING                 00000225  R      02
RETURN_MSGLEN              0000020F  R      02
SEARCH_SECTION             0000010C  R      02
SEARCH_VECTOR              000000E6  R      02
SEVIND                  =  00000002
SEVTAB                     00000030  R      02
SS$_BUFFEROVF           =  00000601
SS$_INSFARG             =  00000114
SS$_MSGNOTFND           =  00000621
STS$K_SUCCESS           =  00000001
STS$M_CONTROL           =  F0000000
STS$M_SEVERITY          =  00000007
STS$S_COND_ID           =  00000019
STS$S_FAC_NO            =  0000000C
STS$S_SEVERITY          =  00000003
STS$V_COND_ID           =  00000003
STS$V_FAC_NO            =  00000010
STS$V_FAC_SP            =  0000000F
STS$V_SEVERITY          =  00000000
SUBIND                  =  00000003
SYS$FAO                    ********    X    02
SYS$IMGACT                 ********   GX    02
TXTIND                  =  00000000
USER                    =  00000002
```

**F

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
YF$SYSGETMSG	00000369 (873.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	37	00:00:00.05	00:00:00.61
Command processing	131	00:00:00.55	00:00:03.82
Pass 1	308	00:00:10.44	00:00:27.44
Symbol table sort	0	00:00:01.56	00:00:04.48
Pass 2	141	00:00:02.53	00:00:08.43
Symbol table output	13	00:00:00.11	00:00:00.23
Psect synopsis output	2	00:00:00.03	00:00:00.07
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	634	00:00:15.28	00:00:45.09

The working set limit was 1650 pages.
60646 bytes (119 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 1000 non-local and 41 local symbols.
750 source lines were read in Pass 1, producing 15 object records in Pass 2.
26 pages of virtual memory were used to define 25 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SYS.OBJ]LIB.MLB;1	7
_255DUA28:[SYSLIB]STARLET.MLB;2	15
TOTALS (all libraries)	22

1112 GETS were required to define 22 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:SYSGETMSG/OBJ=OBJ$:SYSGETMSG MSRC$:SYSGETMSG/UPDATE=(ENH$:SYSGETMSG)+EXECML$/LIB

SYSGETSYI
LIS

SYSGETPTI
LIS

SYSTMGFIX
LIS

SYSGETTIM
LIS

SYSGETLKI
LIS

SYSGETMSG
LIS

SYSIMGACT
LIS