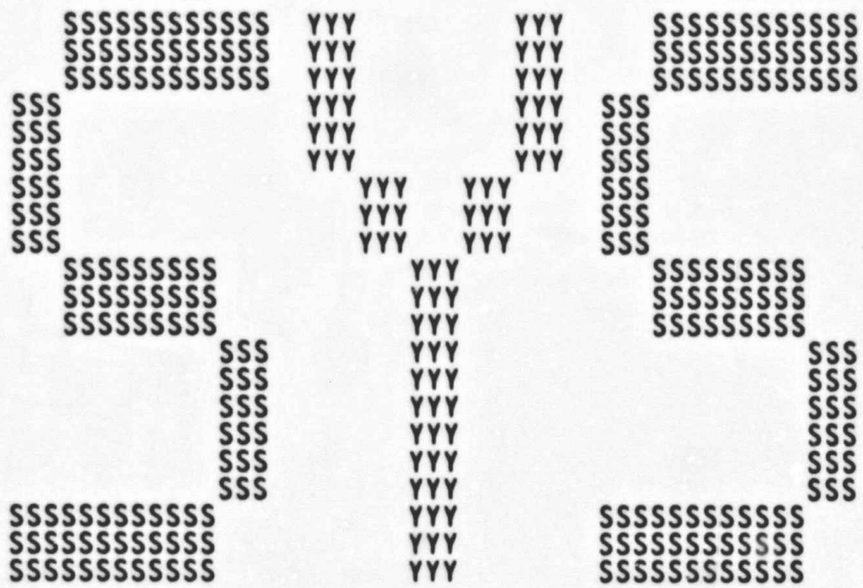
SSSSSSSSSSS YYY YYY SSSSSSSSSSS
SSSSSSSSSSS YYY YYY SSSSSSSSSSS
SSSSSSSSSSS YYY YYY SSSSSSSSSSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSSSSSSSS YYY SSSSSSSSS
SSSSSSSSS YYY SSSSSSSSS
SSSSSSSSS YYY SSSSSSSSS
SSS YYY SSS
SSS YYY SSS
SSS YYY SSS
SSS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

```
  SSSSSSSS  YY      YY   SSSSSSSS  PPPPPPPP    UU      UU  TTTTTTTTTT  MM      MM   SSSSSSSS    GGGGGGGG
  SSSSSSSS  YY      YY   SSSSSSSS  PPPPPPPP    UU      UU  TTTTTTTTTT  MM      MM   SSSSSSSS    GGGGGGGG
SS          YY      YY  SS         PP      PP  UU      UU      TT      MMMM  MMMM  SS          GG
SS          YY      YY  SS         PP      PP  UU      UU      TT      MMMM  MMMM  SS          GG
SS            YY  YY    SS         PP      PP  UU      UU      TT      MM  MM  MM  SS          GG
SS            YY  YY    SS         PP      PP  UU      UU      TT      MM  MM  MM  SS          GG
  SSSSSS        YY        SSSSSS    PPPPPPPP    UU      UU      TT      MM      MM    SSSSSS    GG
  SSSSSS        YY        SSSSSS    PPPPPPPP    UU      UU      TT      MM      MM    SSSSSS    GG
        SS      YY              SS  PP          UU      UU      TT      MM      MM          SS  GG  GGGGGG
        SS      YY              SS  PP          UU      UU      TT      MM      MM          SS  GG  GGGGGG
        SS      YY              SS  PP          UU      UU      TT      MM      MM          SS  GG      GG    ....
        SS      YY              SS  PP          UU      UU      TT      MM      MM          SS  GG      GG    ....
SSSSSSSS        YY      SSSSSSSS    PP          UUUUUUUUUU      TT      MM      MM  SSSSSSSS      GGGGGG     ....
SSSSSSSS        YY      SSSSSSSS    PP          UUUUUUUUUU      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  SYSPUTMSG - SYS$ERROR/SYS$OUTPUT Linked Message Routine
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 ;++
0000    29 ; FACILITY:  System Library
0000    30 ;
0000    31 ; ABSTRACT:
0000    32 ;
0000    33 ;       This utility routine sends one or more messages to SYS$ERROR and
0000    34 ;       SYS$OUTPUT.
0000    35 ;
0000    36 ; ENVIRONMENT:
0000    37 ;
0000    38 ; AUTHOR:  Ward Clark,  CREATION DATE:  5 December 1977
0000    39 ;
0000    40 ; REVISION HISTORY:
0000    41 ;
0000    42 ;       V03-002 JWT0135         Jim Teague              07-Sep-1983
0000    43 ;               ALWAYS call FAO -- don't bypass it if FAO argument
0000    44 ;               count is less than 2.  Carriage control does not
0000    45 ;               show up in the FAO argument count and is ignored.
0000    46 ;
0000    47 ;       V03-001 PCG0001         Peter George            23-May-1983
0000    48 ;               Add processing for "combine" message flag.  This bit directs
0000    49 ;               that the message flags specified in the system service call
0000    50 ;               be reduced by the default process flags.
0000    51 ;
0000    52 ;       V02-014 MLJ0064         Martin L. Jack          13-Dec-1981
0000    53 ;               Add ACTPRM parameter.
0000    54 ;
0000    55 ;       V02-013 KTA0022         Kerbey T. Altmann       10-Jun-1981
0000    56 ;               Add two new messges to the execption list. Also modify
0000    57 ;               the list so that PROCSTRT can use it.
```

```
0000     58 ;
0000     59 ;       V02-012 TMH0012         Tim Halvorsen           24-Feb-1981
0000     60 ;               Close SYS$OUTPUT and SYS$ERROR files after use.
0000     61 ;               If error detected in $FAO, output message w/o FAO.
0000     62 ;
0000     63 ;       V02-011 KTA0009         Kerbey T. Altmann       10-Feb-1981
0000     64 ;               Check length of argument list before accessing an
0000     65 ;               argument to protect against picking up junk.
0000     66 ;
0000     67 ;       010     TMH0008         Tim Halvorsen           31-Jan-1980
0000     68 ;               Increase buffer size to 255 bytes since the supervisor
0000     69 ;               stack size increased enough to handle the space.
0000     70 ;               If inhib_msg bit set in status code, completely ignore
0000     71 ;               message and its arguments.
0000     72 ;               Allow FAO call with leq 2 arguments for system or rms
0000     73 ;               messages since they do not have an FAO count longword.
0000     74 ;
0000     75 ;       009     TMH0007         Tim Halvorsen           26-Jan-1980
0000     76 ;               Fix so that FAO is called only if more than 2 arguments
0000     77 ;               specified, not one (since all msg sets with an FAO count
0000     78 ;               have at least 2 arguments).  Remove bypass of status=0
0000     79 ;               messages if the message is the primary message.
0000     80 ;
0000     81 ;       008     TMH0006         Tim Halvorsen           17-Jan-1980
0000     82 ;               Upcase the first character of the message if text only
0000     83 ;               and suppress null lines.
0000     84 ;
0000     85 ;       007     TMH0005         Tim Halvorsen           14-Jan-1980
0000     86 ;               Save registers r8,r9 over EXE$OPEN_MSG.  Also, always
0000     87 ;               clear r6 (facnam not inserted) on exit paths from facnam
0000     88 ;               processing code.
0000     89 ;
0000     90 ;       006     TMH0004         Tim Halvorsen           10-Jan-1980
0000     91 ;               Call EXE$OPEN_MSG only if message needs to be output
0000     92 ;               in order to reduce the total stack space required for
0000     93 ;               this routine by caller's not needing output (i.e. DCL).
0000     94 ;               Rewrite most of the GET_MODEL_MSG so that process msg
0000     95 ;               flags override if the facility name is given.  Also,
0000     96 ;               reduce the buffer size to 127.
0000     97 ;
0000     98 ;       005     TMH0003         Tim Halvorsen           02-Jan-1980
0000     99 ;               Ignore facility name if the facility bit is off in the
0000    100 ;               message flags argument.
0000    101 ;
0000    102 ;       004     TMH0002         Tim Halvorsen           29-Dec-1979
0000    103 ;               Fix increment delimiter insertion when facility name
0000    104 ;               supplied by caller and text only returned by GETMSG.
0000    105 ;
0000    106 ;       003     TMH0001         Tim Halvorsen           19-Dec-1979
0000    107 ;               Use default message flags from the control region (set
0000    108 ;               using the SET MESSAGE command).  Fix % handling when
0000    109 ;               prefixing a facility name so that the % returned from
0000    110 ;               GETMSG is overwritten with a dash (-).
0000    111 ;
0000    112 ;       02      RIH0038         Richard I. Hustvedt     07-Nov-1979
0000    113 ;               Add status codes for floating faults to list of exception
0000    114 ;               codes.
```

```
0000   115 ;
0000   116 ;--
```

```
                        0000    119          .SBTTL  Declarations
                        0000    120 ;
                        0000    121 ; MACROS:
                        0000    122 ;
                        0000    123
                        0000    124          .MACRO  $EXC_CODE  CODE,ARGS
                        0000    125          .BYTE   ARGS
                        0000    126          .WORD   CODE/8
                        0000    127          .ENDM   $EXC_CODE
                        0000    128
                        0000    129          .MACRO  $FORMAL ARGUMENT_LIST
                        0000    130 $$FORMAL = 0
                        0000    131          .IRP    ARGUMENT,<ARGUMENT_LIST>
                        0000    132 $$FORMAL = $$FORMAL+4
                        0000    133 ARGUMENT = $$FORMAL
                        0000    134          .ENDR
                        0000    135          .ENDM   $FORMAL
                        0000    136
                        0000    137          .MACRO  $LOCAL ARGUMENT_LIST
                        0000    138          .IRP    ARGUMENT,<ARGUMENT_LIST>
                        0000    139          $$LOCAL_ARG ARGUMENT
                        0000    140          .ENDR
                        0000    141          .ENDM   $LOCAL
                        0000    142
                        0000    143          .MACRO  $$LOCAL_ARG NAME,SIZE=4
                        0000    144          .IF     NDF,$$LOCAL_SIZE
                        0000    145 $$LOCAL_SIZE = 0
                        0000    146          .ENDC
                        0000    147 $$LOCAL_SIZE = $$LOCAL_SIZE+SIZE
                        0000    148 NAME = -$$LOCAL_SIZE
                        0000    149          .ENDM   $$LOCAL_ARG
                        0000    150
                        0000    151 ;
                        0000    152 ; EQUATED SYMBOLS:
                        0000    153 ;
                        0000    154
                00000000 0000   155 SS_ID = 0                                   ; VAX/VMS subsystem number
                00000001 0000   156 RMS_ID = 1                                  ; RMS subsystem number
                000000FF 0000   157 MODEL_BUFF_SIZE = 255                       ; Size of model message buffer
                000000FF 0000   158 MSG_BUFF_SIZE = 255                         ; Size of actual message buffer
                00000025 0000   159 PREFIX1 = ^A/%/                             ; Prefix on 1st message
                0000002D 0000   160 PREFIX2 = ^A/-/                             ; Prefix on subsequent messages
                        0000    161
                        0000    162                                             ; Define VAX/VMS symbols:
                        0000    163          $SSDEF                             ;    Define system status values
                        0000    164          $STSDEF                            ;    message code definitions
                        0000    165          $RMSDEF                            ;    RMS message codes
                        0000    166          $FABDEF                            ;    RMS FAB fields, masks and values
                        0000    167          $RABDEF                            ;    RMS RAB fields, masks and values
                        0000    168
                        0000    169 ;
                        0000    170 ; OWN STORAGE:
                        0000    171 ;
                        0000    172
                00000000 0000   173          .PSECT  YEXEPAGED
                        0000    174 EXE$EXCEPTABLE::
                     1C' 0000   175          .BYTE   EXCEPTION_COUNT            ; Define and initialize exception codes tabl
                                                                                ; Number of entries
```

```
                        0001    176 10$:     $EXC_CODE  SS$_ACCVIO,4          ;    Access violation - 4 arguments
00000003                0004    177 EXCEPTION_SIZE = .-10$                    ;    Length of a single table entry
                        0004    178          $EXC_CODE  SS$_MCHECK,2          ;    Machine check - 2 arguments
                        0007    179          $EXC_CODE  SS$_ASTFLT,6          ;    AST delivery stack fault - 6 arguments
                        000A    180          $EXC_CODE  SS$_BREAK,2           ;    Breakpoint fault - 2 arguments
                        000D    181          $EXC_CODE  SS$_CMODSUPR,3        ;    Change mode to supervisor trap - 3 args
                        0010    182          $EXC_CODE  SS$_CMODUSER,3        ;    Change mode to user trap - 3 arguments
                        0013    183          $EXC_CODE  SS$_COMPAT,3          ;    Compatibility mode fault - 3 arguments
                        0016    184          $EXC_CODE  SS$_OPCCUS,2          ;    Opcode reserved to user fault - 2 args
                        0019    185          $EXC_CODE  SS$_OPCDEC,2          ;    Opcode reserved to DEC fault - 2 args
                        001C    186          $EXC_CODE  SS$_PAGRDERR,4        ;    Page read error - 4 arguments
                        001F    187          $EXC_CODE  SS$_RADRMOD,2         ;    Reserved addressing fault - 2 arguments
                        0022    188          $EXC_CODE  SS$_ROPRAND,2         ;    Reserved operand fault - 2 arguments
                        0025    189          $EXC_CODE  SS$_SSFAIL,3          ;    System service failure - 3 arguments
                        0028    190          $EXC_CODE  SS$_TBIT,2            ;    TBIT pending trap - 2 arguments
                        002B    191          $EXC_CODE  SS$_DEBUG,2           ;    Debug trap - 2 arguments
                        002E    192          $EXC_CODE  SS$_ARTRES,2          ;    Arithmetic trap, reserved trap
                        0031    193          $EXC_CODE  SS$_INTOVF,2          ;    Arithmetic trap, integer overflow
                        0034    194          $EXC_CODE  SS$_INTDIV,2          ;    Arithmetic trap, integer divide by zero
                        0037    195          $EXC_CODE  SS$_FLTOVF,2          ;    Arithmetic trap, floating overflow
                        003A    196          $EXC_CODE  SS$_FLTDIV,2          ;    Arithmetic trap, floating/decimal divid
                        003D    197          $EXC_CODE  SS$_FLTUND,2          ;    Arithmetic trap, floating underflow
                        0040    198          $EXC_CODE  SS$_DECOVF,2          ;    Arithmetic trap, decimal overflow
                        0043    199          $EXC_CODE  SS$_SUBRNG,2          ;    Arithmetic trap, subscript out of range
                        0046    200          $EXC_CODE  SS$_FLTOVF_F,2        ;    Arithmetic fault, floating overflow
                        0049    201          $EXC_CODE  SS$_FLTDIV_F,2        ;    Arithmetic fault, floating/decimal divi
                        004C    202          $EXC_CODE  SS$_FLTUND_F,2        ;    Arithmetic fault, floating underflow
                        004F    203          $EXC_CODE  SS$_INHCHMR,3         ;    Inhibited CHMKernel trap - 3 arguments
                        0052    204          $EXC_CODE  SS$_INHCHME,3         ;    Inhibited CHMExecutive trap - 3 argumen
0000001C                0055    205 EXCEPTION_COUNT = <.-10$>/EXCEPTION_SIZE
```

```
0055   208          .SBTTL  SYS$PUTMSG - SYS$ERROR/SYS$OUTPUT message routine
0055   209
0055   210 ;++
0055   211 ; FUNCTIONAL DESCRIPTION:
0055   212 ;
0055   213 ;       This routine is a generalized VAX/VMS message output routine.  Messages
0055   214 ;       (which the caller references by message id) are sent to the SYS$OUTPUT
0055   215 ;       device.  Messages which have a severity different from 1 (normal) are also
0055   216 ;       sent to the SYS$ERROR device.
0055   217 ;
0055   218 ;       Since all user and utility routines are encouraged to "signal" error
0055   219 ;       conditions rather than writing error messages, this routine is
0055   220 ;       structured to be called from a signal handler.  It can, however, be
0055   221 ;       directly called by any routine which can construct a proper argument
0055   222 ;       list.
0055   223 ;
0055   224 ;       The primary (required) argument to this routine is the address of a
0055   225 ;       message argument vector (described below).  The second (optional)
0055   226 ;       argument is the address of a message action routine provided
0055   227 ;       by the caller.  This routine, if present, is called after the
0055   228 ;       standard processing for each message has been performed, but
0055   229 ;       before the message is actually written to the user.  The completion code
0055   230 ;       from the action routine indicates whether or not the message should
0055   231 ;       be sent to the user.  The third (optional) argument is the address
0055   232 ;       of a string descriptor which defines a facility name to be used in
0055   233 ;       the first message of a sequence.
0055   234 ;
0055   235 ;       The message argument vector has the following format:
0055   236 ;
0055   237 ;               a) total number of arguments (b - e)
0055   238 ;               b) message identifier
0055   239 ;               c) number of FAO arguments for the message
0055   240 ;               d) FAO argument(s)
0055   241 ;               e) repeat items b thru d as many times as necessary
0055   242 ;
0055   243 ;       This routine will process each "message set" (items b thru d) by calling
0055   244 ;       $GETMSG and $FAO and then outputting the completed message.  A simple
0055   245 ;       message (i.e., no FAO arguments and no linked message) would be items a,
0055   246 ;       b, f and g.
0055   247 ;
0055   248 ;       There are two special cases involving the message argument structure:
0055   249 ;
0055   250 ;               * an RMS message (STS value) is always immediately
0055   251 ;                 followed by the corresponding STV value.  This STV
0055   252 ;                 value will be used as an FAO argument or another
0055   253 ;                 message id, based on the RMS message number.
0055   254 ;
0055   255 ;               * a system exception message number (e.g., SS$_ARITH)
0055   256 ;                 is always immediately followed by associated
0055   257 ;                 exception values (from 2 to 6) which are treated as
0055   258 ;                 FAO arguments.  The number of arguments is
0055   259 ;                 determined from the message number.
0055   260 ;
0055   261 ; CALLING SEQUENCE:
0055   262 ;
0055   263 ;       CALL SYS$PUTMSG( MSG_ARGS_ADDR.rlu.ra
0055   264 ;                       ,ACTION_ADDR.ra.v
```

```
                     0055   265 ;                                  ,FAC_NAME_ADDR.rt.ds
                     0055   266 ;                                  ,ACTION_PARAM.rlu.v )
                     0055   267 ;
                     0055   268 ;       Note that this routine is actually invoked indirectly thru
                     0055   269 ;       use of the system vector.
                     0055   270 ;
                     0055   271 ; IMPLICIT INPUTS:
                     0055   272 ;
                     0055   273 ;       None
                     0055   274 ;
                     0055   275 ; IMPLICIT OUTPUTS:
                     0055   276 ;
                     0055   277 ;       None
                     0055   278 ;
                     0055   279 ; COMPLETION CODES:
                     0055   280 ;
                     0055   281 ;       SS$_NORMAL - Successful completion
                     0055   282 ;
                     0055   283 ; SIDE EFFECTS:
                     0055   284 ;
                     0055   285 ;       None
                     0055   286 ;
                     0055   287 ;--
                     0055   288
                     0055   289                 $FORMAL < -                     ; Define formal routine arguments:
                     0055   290 MSG_ARGS_ADDR, -                               ;    address of caller's message vector
                     0055   291 ACTION_ADDR, -                                 ;    address of caller's action routine
                     0055   292 FAC_NAME_ADDR, -                               ;    address of facility name descriptor
                     0055   293 ACTION_PARAM >                                 ;    parameter to caller's action routine
                     0055   294
                     0055   295 ;
                     0055   296 ; Define local (stack) variables
                     0055   297 ;
                     0055   298
                     0055   299                 $LOCAL  < -
                     0055   300 <GETMSG_VALUE>, -                             ; Message values returned by $GETMSG
                     0055   301 <MSG_FLAGS,2>, -                              ; Message flags currently selected
                     0055   302 <ARGCNT_LEFT,2>, -                            ; Total argument count left to process
                     0055   303 <FAO_CTL_DESC,8>, -                           ; FAO control string descriptor
                     0055   304 <FAO_OUT_DESC,8>, -                           ; FAO output buffer descriptor
                     0055   305 <SUB_MESSAGE,1>, -                           ; RMS sub-message indicator
                     0055   306 <SECONDARY_MSG,1>, -                         ; True if secondary error message
                     0055   307 <SAVE_REGS,8>, -                             ; Used to save r8,r9 over EXE$OPEN_MSG
                     0055   308 <MODEL_BUFFER,MODEL_BUFF_SIZE>, -            ; Model message buffer for SYS$GETMSG
                     0055   309 <MESSAGE_BUFFER,MSG_BUFF_SIZE> >             ; Actual message buffer
                     0055   310
                OFFC 0055   311         .ENTRY  EXE$PUTMSG,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
5E   FDE0 CE     9E 0057   312         MOVAB   -$$LOCAL_SIZE(SP),SP            ; Allocate space for local variables
           5B    D4 005C   313         CLRL    R11                             ; Mark FAB/RAB's not yet set up
  59   04 AC    D0 005E   314         MOVL    MSG_ARGS_ADDR(AP),R9            ; Get address of message argument list
                  0062   315         ASSUME  MSG_FLAGS EQ ARGCNT_LEFT+2
  F8 AD    89    D0 0062   316         MOVL    (R9)+,ARGCNT_LEFT(FP)           ; Save number of message vector arguments
                  0066   317                                                 ; and set default message flags
       E6 AD    94 0066   318         CLRB    SECONDARY_MSG(FP)               ; Clear secondary indicator
       E7 AD    94 0069   319         CLRB    SUB_MESSAGE(FP)                 ; Clear the sub-message indicator
                  006C   320
                  006C   321 ;
```

```
                                006C   322 ; Repeat the remaining portion of this routine for each
                                006C   323 ; message set provided by the caller.
                                006C   324 ;
                                006C   325
                                006C   326 TOP_OF_LOOP:
                58    01    D0  006C   327         MOVL    #1,R8                   ; Assume a single message argument.
             57     04 A9    9E  006F   328         MOVAB   4(R9),R7                ; Point to FAO argument count
          4E E7 AD     00    E4  0073   329         BBSC    #0,SUB_MESSAGE(FP),GET_MODEL_MSG ; If set, sub-message
                                0078   330
                                0078   331 ;
                                0078   332 ; Special system message setup.
                                0078   333 ;
                                0078   334
                                0078   335         .ENABL  LSB
                                0078   336
                                0078   337         ASSUME  RMS_ID EQ 1
                                0078   338         ASSUME  SS_ID EQ 0
                                0078   339
       01    69    0C    10    ED  0078   340         CMPZV   #STS$V_FAC_NO, -        ; Check the facility code portion
                                007D   341                 #STS$S_FAC_NO,(R9), -   ; of the current message code
                                007D   342                 #RMS_ID                 ; for an RMS id
                      28    18  007D   343         BGEQ    RMS_MESSAGE             ; If geq not system id
                                007F   344
             51    FF7D CF    9E  007F   345         MOVAB   EXE$EXCEPTABLE,R1       ; Point to the table of messages
                   50   81    9A  0084   346         MOVZBL  (R1)+,R0                ; Set loop count
                   52   81    9A  0087   347 10$:     MOVZBL  (R1)+,R2                ; Get number of arguments
                   53   81    3C  008A   348         MOVZWL  (R1)+,R3                ; Get next hardware exception code
                   0C   03    ED  008D   349         CMPZV   #STS$V_CODE,#STS$S_CODE,- ; Condition name match exception code?
                   53   69        0090   350                 (R9),R3
                        0E    13  0092   351         BEQL    20$                     ; Yes - jump to special setup.
                     F0 50    F5  0094   352         SOBGTR  R0,10$                  ; Any more entries to examine?
                  2B E6 AD    E9  0097   353         BLBC    SECONDARY_MSG(FP),GET_MODEL_MSG ; Skip zero bypass if primary
                        69    D5  009B   354         TSTL    (R9)                    ; Null message code? (status=0)
                        27    12  009D   355         BNEQ    GET_MODEL_MSG           ; If neq no
                     01A7     31  009F   356         BRW     END_OF_LOOP             ; Ignore secondary 0 status codes
                58      52    C0  00A2   357 20$:     ADDL    R2,R8                   ; Calculate actual number of FAO arguments
                        1F    11  00A5   358         BRB     GET_MODEL_MSG           ;
                                00A7   359
                                00A7   360 ;
                                00A7   361 ; Special RMS message setup.
                                00A7   362 ;
                                00A7   363
                                00A7   364 RMS_MESSAGE:
                        09    12  00A7   365         BNEQ    OTHER_MESSAGE           ; If neq not RMS id
                69      0E    E1  00A9   366         BBC     #RMS$V_STVSTATUS, -     ; Jump if the associated message
                        17        00AC   367                 (R9),30$                ; argument is not another message code.
                   E7 AD    96  00AD   368         INCB    SUB_MESSAGE(FP)         ; Indicate sub-message
                        14    11  00B0   369         BRB     GET_MODEL_MSG           ; Jump to continue normal processing.
                                00B2   370
                                00B2   371 ;
                                00B2   372 ; Standard (non-special) message setup.
                                00B2   373 ;
                                00B2   374
                                00B2   375 OTHER_MESSAGE:
          F8 AD     01    B1  00B2   376         CMPW    #1,ARGCNT_LEFT(FP)      ; Any more arguments to process?
                    0E    13  00B6   377         BEQL    GET_MODEL_MSG           ; If eql no
             58     87    A0  00B8   378         ADDW    (R7)+,R8                ; Calculate number of FAO arguments
```

```
                87   B5   00BB    379          TSTW    (R7)+                       ; Get message flags specified?
                05   13   00BD    380          BEQL    30$                         ; If eql no
      FA AD     FE A7   B0   00BF    381          MOVW    -2(R7),MSG_FLAGS(FP)        ; Save get message flags
                58   D6   00C4    382 30$:     INCL    R8                          ; Augment number by one
```

```
                                00C6    384          .DSABL  LSB
                                00C6    385
                                00C6    386 ;
                                00C6    387 ; Call $GETMSG to retrieve the model message text which corresponds
                                00C6    388 ; to the current message number.
                                00C6    389 ;
                                00C6    390
                                00C6    391 GET_MODEL_MSG:
                                00C6    392
                                00C6    393 ;
                                00C6    394 ;       If flags argument zero, then use process default flags.
                                00C6    395 ;       If combine bit is set, then reduce the flags argument by the
                                00C6    396 ;       default flags.
                                00C6    397 ;
      55    FA AD    3C        00C6    398          MOVZWL  MSG_FLAGS(FP),R5        ; Get user flags
                     09  12    00CA    399          BNEQ    2$                      ; Branch if non-zero
55    00000000'GF    9A        00CC    400          MOVZBL  G^CTL$GB_MSGMASK,R5     ; If zero, use process flags
                     14  11    00D3    401          BRB     5$                      ; Done processing flags
      10 55    04    E1        00D5    402 2$:      BBC     #4,R5,5$                ; Branch if no combine bit
50    00000000'GF    9A        00D9    403          MOVZBL  G^CTL$GB_MSGMASK,R0     ; Complement default flags
         50    50    D2        00E0    404          MCOML   R0,R0                   ;
         55    50    CA        00E3    405          BICL    R0,R5                   ; Clear the specified flags
         55    10    C8        00E6    406          BISL    #16,R5                  ; Reset the combine bit for $GETMSG
      FA AD    55    B0        00E9    407 5$:      MOVW    R5,MSG_FLAGS(FP)        ; Save final flags
                               00ED    408
   F0 AD    FF 8F    9A        00ED    409          MOVZBL  #MODEL_BUFF_SIZE, -     ; Setup the GETMSG buffer descriptor
                               00F2    410                  FAO_CTL_DESC(FP)       ; with the model buffer size
 F4 AD    FEDF CD    9E        00F2    411          MOVAB   MODEL_BUFFER(FP), -     ; and buffer address.
                               00F8    412                  FAO_CTL_DESC+4(FP)     ;
                               00F8    413 ;
                               00F8    414 ;       If facility message flag set and a facility name was specified,
                               00F8    415 ;       then put the facility name given into the buffer before calling GETMSG
                               00F8    416 ;
         49 E6 AD    E8        00F8    417          BLBS    SECONDARY_MSG(FP),15$   ; Branch if not first message
            03    6C  91        00FC    418          CMPB    (AP), #FAC_NAME_ADDR/4  ; Enough arguments?
                  44  1F        00FF    419          BLSSU   15$                     ; No, don't try to access
         56    0C AC    D0      0101    420          MOVL    FAC_NAME_ADDR(AP),R6    ; Any facility name descriptor?
                  3E  13        0105    421          BEQL    15$                     ; If eql not
      39 FA AD    03  E1        0107    422          BBC     #3,MSG_FLAGS(FP),15$    ; If facility bit off, ignore name
         F0 AD    66  A2        010C    423          SUBW    (R6),FAO_CTL_DESC(FP)   ; Put the remaining buffer length
               F0 AD  B7        0110    424          DECW    FAO_CTL_DESC(FP)        ; into the model buffer descriptor
                  30  15        0113    425          BLEQ    15$                     ; If leq buffer not large enough
         53    F4 AD    D0      0115    426          MOVL    FAO_CTL_DESC+4(FP),R3   ; Address of GETMSG buffer
            83    25  90        0119    427          MOVB    #PREFIXT,(R3)+          ; Insert leading percent sign
   63    04 B6    66  28        011C    428          MOVC    (R6),@4(R6),(R3)        ; Move the facility name to the buffer
   55    FA AD    08  8B        0121    429          BICB3   #^X8,MSG_FLAGS(FP),R5   ; Clear facility name from default flags
      56    66    01  A1        0126    430          ADDW3   #1,(R6),R6              ; Calculate real length of prefix
            54    2D  D0        012A    431          MOVL    #^A'-',R4               ; Set delim to stick over GETMSG result
01    55    04    00  ED        012D    432          CMPZV   #0,#4,R5,#1             ; Requesting only text from GETMSG?
                  0B  12        0132    433          BNEQ    10$                     ; Branch if not
            83    2C  90        0134    434          MOVB    #^A'.',(R3)+            ; If so, append facility/text delimiter
            54    20  90        0137    435          MOVB    #^A' ',R4               ; and set space as delimiter afterwards
                  56  D6        013A    436          INCL    R6                      ; increment prefix length
               F0 AD  D7        013C    437          DECL    FAO_CTL_DESC(FP)        ; and decrement buffer space left
         F4 AD    53  D0        013F    438 10$:     MOVL    R3,FAO_CTL_DESC+4(FP)   ; Point to next available space in buffer
                  02  11        0143    439          BRB     20$
                              0145    440
```

```
                        56   D4  0145   441 15$:    CLRL    R6                          ; Mark no facility name inserted
                                 0147   442
                                 0147   443 20$:    $GETMSG_S -                         ; Call $GETMSG with the following arguments:
                                 0147   444                (R9), -                   ;    message number
                                 0147   445                FAO_CTL_DESC(FP), -       ;    address of text length deposit area
                                 0147   446                FAO_CTL_DESC(FP), -       ;    address of model text buffer descriptor
                                 0147   447                R5, -                     ;    option bits (see above)
                                 0147   448                GETMSG_VALUE(FP)          ;    address of message value deposit area
                                 015B   449
                        56   D5  015B   450          TSTL    R6                          ; Was prefix supplied by caller?
                        09   13  015D   451          BEQL    35$                         ; branch if not
                  01    55   D1  015F   452          CMPL    R5,#1                       ; Did we ask only for text?
                        04   13  0162   453          BEQL    35$                         ; If so, there is no % in string
              F4 BD     54   90  0164   454          MOVB    R4,@FAO_CTL_DESC+4(FP)      ; Overwrite GETMSG % with delimiter
              F0 AD     56   A0  0168   455 35$:     ADDW    R6,FAO_CTL_DESC(FP)         ; Add in length of prefix
        F4 AD    FEDF  CD   9E  016C   456          MOVAB   MODEL_BUFFER(FP),FAO_CTL_DESC+4(FP) ; Reset to begining of buffer
                 F0 AD   B5   12  0172   457          TSTW    FAO_CTL_DESC(FP)            ; Null string?
                        03   12  0175   458          BNEQ    40$                         ; If not, continue
                     00CF   31  0177   459          BRW     END_OF_LOOP                 ; If null string, skip to next message
                                 017A   460 ;
                                 017A   461 ;        Upcase the first character if text only message
                                 017A   462 ;
  01    FA AD     04    00   ED  017A   463 40$:    CMPZV   #0,#4,MSG_FLAGS(FP),#1      ; Text only message?
                        14   12  0180   464          BNEQ    FINAL_MESSAGE               ; Branch if not
              50    F4 AD     D0  0182   465          MOVL    FAO_CTL_DESC+4(FP),R0       ; Get address of first character
              61 8F     60   91  0186   466          CMPB    (R0),#^A'a'                 ; Check lower bounds of lowercase range
                        0A   1F  018A   467          BLSSU   FINAL_MESSAGE               ; Branch if already upper case
              7A 8F     60   91  018C   468          CMPB    (R0),#^A'z'                 ; Check upper bounds of lowercase range
                        04   1A  0190   469          BGTRU   FINAL_MESSAGE               ; Branch if already upper case
              60    E0 8F     80  0192   470          ADDB    #^A'A'-^A'a',(R0)           ; Convert to upper case
```

```
                     0196   472
                     0196   473 ;
                     0196   474 ; Create the final output message by calling $FAOL to fillin the variable
                     0196   475 ; portions of the model message returned by $GETMSG, or simply move the
                     0196   476 ; model message to the output buffer.
                     0196   477 ;
                     0196   478
                     0196   479 FINAL_MESSAGE:
  E8 AD   FF 8F   9A 0196   480 5$:     MOVZBL  #MSG_BUFF_SIZE,FAO_OUT_DESC(FP) ; Set length of message buffer
EC AD   FDE0 CD   9E 019B   481         MOVAB   MESSAGE_BUFFER(FP),FAO_OUT_DESC+4(FP) ; Set address of buffer
                     01A1   482         $FAOL_S -                       ; Call $FAOL with the following arguments:
                     01A1   483                 FAO_CTL_DESC(FP), -     ;    addr of control msg string desc
                     01A1   484                 FAO_OUT_DESC(FP), -     ;    addr of msg size deposit area
                     01A1   485                 FAO_OUT_DESC(FP), -     ;    addr of msg buffer descriptor
                     01A1   486                 (R7)                    ;    addr of the FAO argument list, if any
        05 50   E8 01B3   487         BLBS    R0,20$                  ; Jump to add the message prefix.
                     01B6   488                                         ; If FAO failed, use original string
  E8 AD   F0 AD   7D 01B6   489 10$:    MOVQ    FAO_CTL_DESC(FP), -     ; Copy control buffer descriptor
                     01BB   490                 FAO_OUT_DESC(FP)        ;
08 E6 AD    00    E3 01BB   491 20$:    BBCS    #0,SECONDARY_MSG(FP),CALL_ACTION ; If clr, output first message
   04 55    03    E1 01C0   492         BBC     #3,R5,CALL_ACTION       ; If clr, suppress insertion on minus sign
   EC BD    2D    90 01C4   493         MOVB    #^A/-/,@FAO_OUT_DESC+4(FP) ; Insert leading minus sign
```

```
                        01C8   495
                        01C8   496 ;
                        01C8   497 ; Call the caller's action routine if one was provided.
                        01C8   498 ;
                        01C8   499
                        01C8   500 CALL_ACTION:
7D 69    1C    E0       01C8   501          BBS     #STS$V_INHIB_MSG,(R9),END_OF_LOOP ; ignore message if inhibited
   02    6C    91       01CC   502          CMPB    (AP), #ACTION_ADDR/4      ; Enough arguments?
         1A    1F       01CF   503          BLSSU   PUT_SYS$ERROR             ; No, don't try to access it
      08 AC    D5       01D1   504          TSTL    ACTION_ADDR(AP)           ; if action routine address is zero,
         15    13       01D4   505          BEQL    PUT_SYS$ERROR             ; bypass calling an action routine.
         00    DD       01D6   506          PUSHL   #0                        ; Push zero action parameter
   04    6C    91       01D8   507          CMPB    (AP), #ACTION_PARAM/4     ; Enough arguments?
         04    1F       01DB   508          BLSSU   25$                       ; No, don't try to access it
6E    10 AC    D0       01DD   509          MOVL    ACTION_PARAM(AP), (SP)    ; Copy user's parameter
      E8 AD    9F       01E1   510 25$:     PUSHAB  FAO_OUT_DESC(FP)          ; Push the address of message descriptor
08 BC    02    FB       01E4   511          CALLS   #2,@ACTION_ADDR(AP)       ; and call the caller's action routine.
      5E 50    E9       01E8   512          BLBC    R0,END_OF_LOOP            ; If lbc skip further output of message
                        01EB   513
                        01EB   514 ;
                        01EB   515 ; Send error messages to the SYS$ERROR device if this is not a success sequence
                        01EB   516 ;
                        01EB   517
                        01EB   518 PUT_SYS$ERROR:
         5B    D5       01EB   519          TSTL    R11                       ; Have FAB/RAB's been set up yet?
         0B    12       01ED   520          BNEQ    5$                        ; branch if all set from last iteration
DE AD    58    7D       01EF   521          MOVQ    R8,SAVE_REGS(FP)          ; Save registers
      FE0A'    30       01F3   522          BSBW    EXE$OPEN_MSG              ; Allocate/init FAB and RAB's on stack
58    DE AD    7D       01F6   523          MOVQ    SAVE_REGS(FP),R8          ; Restore registers
      50 69    D2       01FA   524 5$:      MCOML   (R9),R0                   ; Get complement of severity field
04 AB 01 1F    50       01FD   525          INSV    R0,#RAB$V_CCO,#1,RAB$L_ROP(R11) ; Cancel ^O if not success or info
         03    00       0203   526          CMPZV   #STS$V_SEVERITY,#STS$S_SEVERITY,- ; If severity field
         06    50       0206   527                  R0,#<^C<STS$K_SUCCESS>&STS$M_SEVERITY> ; is "success"
               23    13 0208   528          BEQL    10$                       ; then don't write SYS$ERROR
      22 AB    E8 AD    B0 020A   529          MOVW    FAO_OUT_DESC(FP),RAB$W_RSZ(R11) ; Set size of output message
      28 AB    EC AD    D0 020F   530          MOVL    FAO_OUT_DESC+4(FP),RAB$L_RBF(R11) ; Set address of output message
                        0214   531          $WAIT   RAB=(R11)                 ; Wait for any outstanding I/O
                        021D   532          $PUT    RAB=(R11)                 ; Send the message to SYS$ERROR.
                        0226   533
                        0226   534 ;
                        0226   535 ; Send the completed message to the SYS$OUTPUT device if different from 'SYS$ERROR'
                        0226   536 ;
                        0226   537
      02 AB    02 AA    B1 0226   538          CMPW    RAB$W_ISI(R10),RAB$W_ISI(R11) ; SYS$ERROR and SYS$OUTPUT same?
               1C    13 022B   539          BEQL    END_OF_LOOP               ; If eql yes
      22 AA    E8 AD    B0 022D   540 10$:     MOVW    FAO_OUT_DESC(FP),RAB$W_RSZ(R10) ; Set size of output message
      28 AA    EC AD    D0 0232   541          MOVL    FAO_OUT_DESC+4(FP),RAB$L_RBF(R10) ; Set address of output message
                        0237   542          $WAIT   RAB=(R10)                 ; Wait for any outstanding I/O
                        0240   543          $PUT    RAB=(R10)                 ; Send the message to SYS$OUTPUT.
                        0249   544
                        0249   545 ;
                        0249   546 ; Setup to process the next message, if any.
                        0249   547 ;       R8 = Number of longwords gobbled for this message
                        0249   548 ;
                        0249   549
                        0249   550 END_OF_LOOP:
   F8 AD    58    A2 0249   551          SUBW    R8,ARGCNT_LEFT(FP)        ; Calculate remaining arguments
```

```
                07   15  024D    552          BLEQ    RETURN                  ; If leq no more to process
            59  6948   DE  024F    553          MOVAL   (R9)[R8],R9             ; Get address of next message code
                FE16   31  0253    554          BRW     TOP_OF_LOOP             ; Loop until all messages have been
                           0256    555                                          ; processed.
                           0256    556 ;
                           0256    557 ; Close the message files
                           0256    558 ;
                           0256    559
                           0256    560 RETURN:
                FDA7'  30  0256    561          BSBW    EXE$CLOSE_MSG           ; Close the message files
                           0259    562
                           0259    563 ;
                           0259    564 ; Return to the caller.
                           0259    565 ;
                           0259    566
                50  01  D0  0259    567          MOVL    S^#SS$_NORMAL,R0        ; Return a normal completion code
                       04  025C    568          RET                             ; to the caller.
                           025D    569
                           025D    570          .END
```

```
$$.TMP1                 = 00000001
$$.TMP2                 = 0000006A
$$FORMAL                = 00000010
$$LOCAL_SIZE            = 00000220
ACTION_ADDR             = 00000008
ACTION_PARAM            = 00000010
ARGCNT_LEFT             = FFFFFFF8
CALL_ACTION               000001C8 R      02
CTL$GB_MSGMASK            ******** X      02
END_OF_LOOP               00000249 R      02
EXCEPTION_COUNT         = 0000001C
EXCEPTION_SIZE          = 00000003
EXE$CLOSE_MSG             ******** X      02
EXE$EXCEPTABLE            00000000 RG     02
EXE$OPEN_MSG              ******** X      02
EXE$PUTMSG                00000055 RG     02
FAC_NAME_ADDR           = 0000000C
FAO_CTL_DESC            = FFFFFFF0
FAO_OUT_DESC            = FFFFFFE8
FINAL_MESSAGE             00000196 R      02
GETMSG_VALUE            = FFFFFFFC
GET_MODEL_MSG             000000C6 R      02
MESSAGE_BUFFER          = FFFFFDE0
MODEL_BUFFER            = FFFFFEDF
MODEL_BUFF_SIZE         = 000000FF
MSG_ARGS_ADDR           = 00000004
MSG_BUFF_SIZE           = 000000FF
MSG_FLAGS               = FFFFFFFA
OTHER_MESSAGE             000000B2 R      02
PREFIX1                 = 00000025
PREFIX2                 = 0000002D
PUT_SYS$ERROR             000001EB R      02
RAB$L_RBF               = 00000028
RAB$L_ROP               = 00000004
RAB$V_CCO               = 0000001F
RAB$W_ISI               = 00000002
RAB$W_RSZ               = 00000022
RETURN                    00000256 R      02
RMS$V_STVSTATUS         = 0000000E
RMS_ID                  = 00000001
RMS_MESSAGE               000000A7 R      02
SAVE_REGS               = FFFFFFDE
SECONDARY_MSG           = FFFFFFE6
SS$_ACCVIO              = 0000000C
SS$_ARTRES              = 00000474
SS$_ASTFLT              = 0000040C
SS$_BREAK               = 00000414
SS$_CMODSUPR            = 0000041C
SS$_CMODUSER            = 00000424
SS$_COMPAT              = 0000042C
SS$_DEBUG               = 0000046C
SS$_DECOVF              = 000004A4
SS$_FLTDIV              = 00000494
SS$_FLTDIV_F            = 000004BC
SS$_FLTOVF              = 0000048C
SS$_FLTOVF_F            = 000004B4
SS$_FLTUND              = 0000049C
SS$_FLTUND_F            = 000004C4
SS$_INHCHME             = 000004D4
SS$_INHCHMK             = 000004CC
SS$_INTDIV              = 00000484
SS$_INTOVF              = 0000047C
SS$_MCHECK              = 000002BC
SS$_NORMAL              = 00000001
SS$_OPCCUS              = 00000434
SS$_OPCDEC              = 0000043C
SS$_PAGRDERR            = 00000444
SS$_RADRMOD             = 0000044C
SS$_ROPRAND             = 00000454
SS$_SSFAIL              = 0000045C
SS$_SUBRNG              = 000004AC
SS$_TBIT                = 00000464
SS_ID                   = 00000000
STS$K_SUCCESS           = 00000001
STS$M_SEVERITY          = 00000007
STS$S_CODE              = 0000000C
STS$S_FAC_NO            = 0000000C
STS$S_SEVERITY          = 00000003
STS$V_CODE              = 00000003
STS$V_FAC_NO            = 00000010
STS$V_INHIB_MSG         = 0000001C
STS$V_SEVERITY          = 00000000
SUB_MESSAGE             = FFFFFFE7
SYS$FAOL                  ********  GX    02
SYS$GETMSG                ********  GX    02
SYS$PUT                   ********  GX    02
SYS$WAIT                  ********  GX    02
TOP_OF_LOOP               0000006C R      02
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
YEXEPAGED	0000025D (605.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	35	00:00:00.10	00:00:00.56
Command processing	129	00:00:00.57	00:00:01.76
Pass 1	304	00:00:09.85	00:00:21.65
Symbol table sort	0	00:00:01.24	00:00:02.34
Pass 2	116	00:00:02.21	00:00:05.10
Symbol table output	12	00:00:00.09	00:00:00.09
Psect synopsis output	2	00:00:00.02	00:00:00.04
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	600	00:00:14.09	00:00:31.55

The working set limit was 1500 pages.
56719 bytes (111 pages) of virtual memory were used to buffer the intermediate code.
There were 50 pages of symbol table space allocated to hold 1002 non-local and 17 local symbols.
570 source lines were read in Pass 1, producing 17 object records in Pass 2.
24 pages of virtual memory were used to define 22 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SYS.OBJ]LIB.MLB;1	0
_255DUA28:[SYSLIB]STARLET.MLB;2	15
TOTALS (all libraries)	15

1120 GETS were required to define 15 macros.

There were no errors, warnings or information messages.

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

0387 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

SYSPURGWS
LIS

SYSPUTMSG
LIS

SYSPCNTRL
LIS

SYSQIOFDT
LIS

SYSQIOREQ
LIS

SYSRUNDWN
LIS

SYSRDBRES
LIS

SYSRTSLST
LIS