```
MMM          MMM     SSSSSSSSSSS     GGGGGGGGGGG  FFFFFFFFFFFFFFF      IIIIIIIII     LLL
MMM          MMM     SSSSSSSSSSS     GGGGGGGGGGG  FFFFFFFFFFFFFFF      IIIIIIIII     LLL
MMM          MMM     SSSSSSSSSSS     GGGGGGGGGGG  FFFFFFFFFFFFFFF      IIIIIIIII     LLL
MMMMMM    MMMMMM  SSS             GGG             FFF                     III        LLL
MMMMMM    MMMMMM  SSS             GGG             FFF                     III        LLL
MMMMMM    MMMMMM  SSS             GGG             FFF                     III        LLL
MMM   MMM    MMM  SSS             GGG             FFF                     III        LLL
MMM   MMM    MMM  SSS             GGG             FFF                     III        LLL
MMM   MMM    MMM  SSS             GGG             FFF                     III        LLL
MMM          MMM     SSSSSSSSS    GGG             FFFFFFFFFFF             III        LLL
MMM          MMM     SSSSSSSSS    GGG             FFFFFFFFFFF             III        LLL
MMM          MMM     SSSSSSSSS    GGG             FFFFFFFFFFF             III        LLL
MMM          MMM             SSS  GGG   GGGGGGGGG  FFF                     III        LLL
MMM          MMM             SSS  GGG   GGGGGGGGG  FFF                     III        LLL
MMM          MMM             SSS  GGG   GGGGGGGGG  FFF                     III        LLL
MMM          MMM             SSS  GGG         GGG  FFF                     III        LLL
MMM          MMM             SSS  GGG         GGG  FFF                     III        LLL
MMM          MMM             SSS  GGG         GGG  FFF                     III        LLL
MMM          MMM  SSSSSSSSSSS     GGGGGGGGG    FFF                 IIIIIIIII     LLLLLLLLLLLLLLL
MMM          MMM  SSSSSSSSSSS     GGGGGGGGG    FFF                 IIIIIIIII     LLLLLLLLLLLLLLL
MMM          MMM  SSSSSSSSSSS     GGGGGGGGG    FFF                 IIIIIIIII     LLLLLLLLLLLLLLL
```

FILEID**OBJECT

```
  000000    BBBBBBBB            JJ  EEEEEEEEEE    CCCCCCCC  TTTTTTTTTT
  000000    BBBBBBBB            JJ  EEEEEEEEEE    CCCCCCCC  TTTTTTTTTT
00      00  BB      BB          JJ  EE          CC              TT
00      00  BB      BB          JJ  EE          CC              TT
00      00  BB      BB          JJ  EE          CC              TT
00      00  BB      BB          JJ  EE          CC              TT
00      00  BBBBBBBB            JJ  EEEEEEEE    CC              TT
00      00  BBBBBBBB            JJ  EEEEEEEE    CC              TT
00      00  BB      BB  JJ      JJ  EE          CC              TT
00      00  BB      BB  JJ      JJ  EE          CC              TT
00      00  BB      BB  JJ      JJ  EE          CC              TT          ....
00      00  BB      BB  JJ      JJ  EE          CC              TT          ....
  000000    BBBBBBBB      JJJJJJ    EEEEEEEEEE    CCCCCCCC      TT          ....
  000000    BBBBBBBB      JJJJJJ    EEEEEEEEEE    CCCCCCCC      TT          ....

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
```

```
;     1          0001  0 MODULE object (IDENT='V04-000') =
;     2          0002  0
;     3          0003  1 BEGIN
;     4          0004  1
;     5          0005  1 !
;     6          0006  1 !*****************************************************************************
;     7          0007  1 !*                                                                           *
;     8          0008  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;     9          0009  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
;    10          0010  1 !*  ALL RIGHTS RESERVED.                                                      *
;    11          0011  1 !*                                                                           *
;    12          0012  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
;    13          0013  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
;    14          0014  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;    15          0015  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;    16          0016  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;    17          0017  1 !*  TRANSFERRED.                                                              *
;    18          0018  1 !*                                                                           *
;    19          0019  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;    20          0020  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;    21          0021  1 !*  CORPORATION.                                                              *
;    22          0022  1 !*                                                                           *
;    23          0023  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
;    24          0024  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
;    25          0025  1 !*                                                                           *
;    26          0026  1 !*                                                                           *
;    27          0027  1 !*****************************************************************************
;    28          0028  1
;    29          0029  1 !++
;    30          0030  1 ! FACILITY:  The MESSAGE compiler
;    31          0031  1 !
;    32          0032  1 ! ABSTRACT:
;    33          0033  1 !
;    34          0034  1 !       This compiler translates a message definition language
;    35          0035  1 !       and produces object modules suitable for the $GETMSG
;    36          0036  1 !       system service.
;    37          0037  1 !
;    38          0038  1 ! ENVIRONMENT:
;    39          0039  1 !
;    40          0040  1 !       VAX/VMS operating system. unprivileged user mode,
;    41          0041  1 !
;    42          0042  1 ! AUTHOR:  Tim Halvorsen, Nov 1979
;    43          0043  1 !
;    44          0044  1 ! Modified by:
;    45          0045  1 !
;    46          0046  1 !       V03-004 GJA0096         Greg Awdziewicz         10-Aug-1984
;    47          0047  1 !               - Change version # written into object module to v04-00.
;    48          0048  1 !
;    49          0049  1 !       V03-003 GJA0079         Greg Awdziewicz         6-Apr-1984
;    50          0050  1 !               Initialize flags byte in Create_indirect.
;    51          0051  1 !
;    52          0052  1 !       V03-002 GJA0050         Greg Awdziewicz         12-Sep-1983
;    53          0053  1 !               Reference the filename descriptor as a byte block structure
;    54          0054  1 !               instead of as a longword vector.
;    55          0055  1 !
;    56          0056  1 !       001     JWT0024         Jim Teague      17-Mar-1982
;    57          0057  1 !               Corrects the bug which prevented pointer psects from being
```

```
;    58          0058  1 !               output when /notext is explicitly specified.
;    59          0059  1 !
;    60          0060  1 !--
```

```
;    62          0061  1 !
;    63          0062  1 ! Include files
;    64          0063  1 !
;    65          0064  1
;    66          0065  1 LIBRARY 'SYS$LIBRARY:STARLET';          . VMS common definitions
;    67          0066  1
;    68          0067  1 REQUIRE 'SRC$:MSG.REQ';                 . Message common definitions
```

```
;    70          0305  1 !
;    71          0306  1 ! Table of contents
;    72          0307  1 !
;    73          0308  1
;    74          0309  1 FORWARD ROUTINE
;    75          0310  1     output_object,                  ! Output object module
;    76          0311  1     create_section,                 ! Create message section
;    77          0312  1     create_indirect,                ! Create indirect message section
;    78          0313  1     output_mhd,                     ! Output module header record
;    79          0314  1     output_psects,                  ! Output psect definitions
;    80          0315  1     output_gsd,                     ! Output global symbol records
;    81          0316  1     output_section,                 ! Output message section
;    82          0317  1     output_eom,                     ! Output end of module record
;    83          0318  1     set_psect,                      ! Set to selected psect
;    84          0319  1     put_record,                     ! Put object record to file
;    85          0320  1     start_record,                   ! Start of GSD/TIR record
;    86          0321  1     put_entry,                      ! Output GSD/TIR entry
;    87          0322  1     end_record;                     ! End of GSD/TIR entries
;    88          0323  1
;    89          0324  1 !
;    90          0325  1 ! Macros
;    91          0326  1 !
;    92          0327  1
;    93          0328  1 MACRO
;    94        M 0329  1     perform(command) =
;    95        M 0330  1         BEGIN
;    96        M 0331  1         LOCAL status;
;    97        M 0332  1         status = command;
;    98        M 0333  1         IF NOT .status
;    99        M 0334  1         THEN
;   100        M 0335  1             BEGIN
;   101        M 0336  1             SIGNAL(.status);
;   102        M 0337  1             RETURN .status;
;   103        M 0338  1             END;
;   104          0339  1         END%;
;   105          0340  1
;   106          0341  1 !
;   107          0342  1 ! Literals
;   108          0343  1 !
;   109          0344  1
;   110          0345  1 LITERAL
;   111          0346  1     psect_flags = gps$m_pic OR gps$m_rel OR gps$m_rd OR gps$m_vec;
;   112          0347  1
;   113          0348  1 !
;   114          0349  1 ! Module storage
;   115          0350  1 !
;   116          0351  1
;   117          0352  1 OWN
;   118          0353  1     section:            REF BBLOCK,     ! Address of message section
;   119          0354  1     record_buffer:      BBLOCK[obj$c_maxrecsiz],        ! record buffer
;   120          0355  1     rec_filled;                         ! Space used in record_buffer so far
;   121          0356  1
;   122          0357  1 !
;   123          0358  1 ! External storage
;   124          0359  1 !
;   125          0360  1
;   126          0361  1 EXTERNAL
```

```
;  127     0362  1      object_fab:         BBLOCK,         ! Object module FAB
;  128     0363  1      object_rab:         BBLOCK,         ! Object module RAB
;  129     0364  1      cli_flags:          BITVECTOR,      ! CLI qualifier flags
;  130     0365  1      module_name:        VECTOR,         ! Name of object module
;  131     0366  1      filename_desc:      BLOCK [, BYTE], ! Descriptor of /FILE value
;  132     0367  1      facility_header,                    ! Facility list head
;  133     0368  1      message_header,                     ! Message list head
;  134     0369  1      symbol_header,                      ! Symbol list head
;  135     0370  1      num_messages,                       ! Number of messages defined
;  136     0371  1      msg_space,                          ! Total space used by MSG records
;  137     0372  1      num_facilities,                     ! Number of facilities defined
;  138     0373  1      fac_space,                          ! Total space needed for facility table
;  139     0374  1      version_num:        VECTOR,         ! Descriptor for version/ident
;  140     0375  1      num_files;                          ! Total files parsed
;  141     0376  1
;  142     0377  1  !
;  143     0378  1  ! External routines
;  144     0379  1  !
;  145     0380  1
;  146     0381  1  EXTERNAL ROUTINE
;  147     0382  1      get_module_name,                    ! Extract module name
;  148     0383  1      rms_error,                          ! Signal RMS-type error
;  149     0384  1      allocate;                           ! Allocate dynamic memory
```

```
;  151     0385  1 GLOBAL ROUTINE output_object =
;  152     0386  1
;  153     0387  1 !---
;  154     0388  1 !
;  155     0389  1 !       This routine generates the object module file from the
;  156     0390  1 !       message definition blocks generated by input parse routines.
;  157     0391  1 !
;  158     0392  1 ! Inputs:
;  159     0393  1 !
;  160     0394  1 !       object_fab/rab = FAB/RAB for object module file
;  161     0395  1 !       facility_header = Facility list head
;  162     0396  1 !       message_header = Message definition list head
;  163     0397  1 !
;  164     0398  1 ! Outputs:
;  165     0399  1 !
;  166     0400  1 !       The object module file
;  167     0401  1 !---
;  168     0402  1
;  169     0403  2 BEGIN
;  170     0404  2
;  171     0405  2 LOCAL
;  172     0406  2     status;                             ! status code
;  173     0407  2
;  174     0408  2 !
;  175     0409  2 !       Create the object module file
;  176     0410  2 !
;  177     0411  2
;  178     0412  2 status = $CREATE (FAB = object_fab);    ! Create the output file
;  179     0413  2 IF NOT .status                          ! If error detected,
;  180     0414  2 THEN
;  181     0415  3     BEGIN
;  182     0416  3     rms_error(emsg(openout),object_fab);        ! then signal the error
;  183     0417  3     RETURN .status;                     ! and return with status
;  184     0418  2     END;
;  185     0419  2
;  186     0420  2 status = $CONNECT (RAB = object_rab);   ! Connect to the object file
;  187     0421  2 IF NOT .status                          ! If error detected,
;  188     0422  2 THEN
;  189     0423  3     BEGIN
;  190     0424  3     rms_error(emsg(openout),object_fab,object_rab);     ! then signal it
;  191     0425  3     RETURN .status;                     ! and return with status
;  192     0426  2     END;
;  193     0427  2
;  194     0428  2 IF .num_files GTR 1                     ! If more than 1 input file,
;  195     0429  2 THEN
;  196     0430  2     get_module_name(object_fab);        ! Use object file name as module name
;  197     0431  2
;  198     0432  2 !
;  199     0433  2 !       Create the index message structure in dyramic storage.
;  200     0434  2 !
;  201     0435  2
;  202     0436  2 IF NOT .cli_flags [qual_file]           ! If /FILE not specified,
;  203     0437  2 THEN
;  204     0438  3     perform(create_section())           ! Create the message structure
;  205     0439  2 ELSE
;  206     0440  2     perform(create_indirect());         ! Else, create indirect section
;  207     0441  2
```

```
;  208     0442  2 !
;  209     0443  2 !       Output the module header record
;  210     0444  2 !
;  211     0445  2
;  212     0446  2 perform(output_mhd());                  ! Output the module header record
;  213     0447  2
;  214     0448  2 !
;  215     0449  2 !       Define the psects used for the message section.  If
;  216     0450  2 !       constructing an indirect message section, make the psects
;  217     0451  2 !       writable.
;  218     0452  2 !
;  219     0453  2
;  220     0454  2 IF NOT .cli_flags [qual_file]           ! If /FILE qualifier not specified,
;  221     0455  2 THEN
;  222     0456  3     perform(output_psects(psect_flags)) ! Make normal NOWRT
;  223     0457  2 ELSE
;  224     0458  2     perform(output_psects(psect_flags OR gps$m_wrt));   ! Make indirect WRT
;  225     0459  2
;  226     0460  2 !
;  227     0461  2 !       Output the GSD records for the symbols
;  228     0462  2 !
;  229     0463  2
;  230     0464  2 IF .cli_flags [qual_symbols]            ! If /SYMBOLS specified,
;  231     0465  2 THEN
;  232     0466  2     perform(output_gsd());              ! Output the GSD records
;  233     0467  2
;  234     0468  2 !
;  235     0469  2 !       Output the TIR records which describe the message structure
;  236     0470  2 !
;  237     0471  2
;  238     0472  2 IF .cli_flags [qual_text] OR .cli_flags [qual_file]  ! If /TEXT or /FILE
;  239     0473  2 THEN
;  240     0474  2     perform(output_section());          ! Output section TIR records
;  241     0475  2
;  242     0476  2 !
;  243     0477  2 !       Output the end-of-module record
;  244     0478  2 !
;  245     0479  2
;  246     0480  2 perform(output_eom());                  ! Output the EOM record
;  247     0481  2
;  248     0482  2 !
;  249     0483  2 !       Close the object module file
;  250     0484  2 !
;  251     0485  2
;  252     0486  2 status = $CLOSE(FAB = object_fab);      ! Close the file
;  253     0487  2 IF NOT .status                          ! If error,
;  254     0488  2 THEN
;  255     0489  3     BEGIN
;  256     0490  3     rms_error(emsg(closedel),object_fab); ! signal the error
;  257     0491  3     RETURN .status;                     ! and return with status
;  258     0492  2     END;
;  259     0493  2
;  260     0494  2 RETURN true;
;  261     0495  2
;  262     0496  1 END;
```

```
                                        .TITLE  OBJECT
                                        .IDENT  \V04-000\

                                        .PSECT  $OWN$,NOEXE,2

                                00000 SECTION:.BLKB   4
                                00004 RECORD_BUFFER:
                                        .BLKB   2048
                                00804 REC_FILLED:
                                        .BLKB   4

                                        .EXTRN  OBJECT_FAB, OBJECT_RAB
                                        .EXTRN  CLI_FLAGS, MODULE_NAME
                                        .EXTRN  FILENAME_DESC, FACILITY_HEADER
                                        .EXTRN  MESSAGE_HEADER, SYMBOL_HEADER
                                        .EXTRN  NUM_MESSAGES, MSG_SPACE
                                        .EXTRN  NUM_FACILITIES, FAC_SPACE
                                        .EXTRN  VERSION_NUM, NUM_FILES
                                        .EXTRN  GET_MODULE_NAME
                                        .EXTRN  RMS_ERROR, ALLOCATE
                                        .EXTRN  SYS$CREATE, SYS$CONNECT
                                        .EXTRN  SYS$CLOSE

                                        .PSECT  $CODE$,NOWRT,2

                                003C 00000      .ENTRY  OUTPUT_OBJECT, Save R2,R3,R4,R5         ; 0385
                        55      0000G   CF  9E  00002   MOVAB   CLI_FLAGS, R5                   ;
                        54      0000G   CF  9E  00007   MOVAB   OBJECT_FAB, R4                  ;
                                        54  DD  0000C   PUSHL   R4                              ; 0412
00000000G               00              01  FB  0000E   CALLS   #1, SYS$CREATE                 ;
                        53              50  D0  00015   MOVL    R0, STATUS                    ;
                        0B              53  E8  00018   BLBS    STATUS, 1$                    ; 0413
                                        54  DD  0001B   PUSHL   R4                              ; 0416
                            009710A4    8F  DD  0001D   PUSHL   #9900196                      ;
                                    00C0    31  00023   BRW     13$                           ;
                                0000G   CF  9F  00026 1$:   PUSHAB  OBJECT_RAB                  ; 0420
                    00000000G   00          01  FB  0002A   CALLS   #1, SYS$CONNECT           ;
                                53          50  D0  00031   MOVL    R0, STATUS                ;
                                14          53  E8  00034   BLBS    STATUS, 2$                ; 0421
                                0000G   CF  9F  00037   PUSHAB  OBJECT_RAB                  ; 0424
                                        54  DD  0003B   PUSHL   R4                          ;
                            009710A4    8F  DD  0003D   PUSHL   #9900196                    ;
                        0000G   CF          03  FB  00043   CALLS   #3, RMS_ERROR           ;
                                    00A0    31  00048   BRW     14$                         ; 0425
                                01      0000G   CF  D1  0004B 2$:   CMPL    NUM_FILES, #1   ; 0428
                                        07  15  00050   BLEQ    3$                          ;
                                        54  DD  00052   PUSHL   R4                          ; 0430
                        0000G   CF          01  FB  00054   CALLS   #1, GET_MODULE_NAME     ;
07                              65          02  E0  00059 3$:   BBS     #2, CLI_FLAGS, 4$   ; 0436
                        0000V   CF          00  FB  0005D   CALLS   #0, CREATE_SECTION      ; 0438
                                        05  11  00062   BRB     5$                          ;
                        0000V   CF          00  FB  00064 4$:   CALLS   #0, CREATE_INDIRECT ; 0440
                                52          50  D0  00069 5$:   MOVL    R0, STATUS          ;
                                53          52  E9  0006C   BLBC    STATUS, 11$           ;
                        0000V   CF          00  FB  0006F   CALLS   #0, OUTPUT_MHD          ; 0446
                                52          50  D0  00074   MOVL    R0, STATUS            ;
                                48          52  E9  00077   BLBC    STATUS, 11$           ;
```

```
                07              65              02  E0 0007A          BBS     #2, CLI_FLAGS, 6$                       ; 0454
                                7E      0289    8F  3C 0007E          MOVZWL  #649, -(SP)                             ; 0456
                                                05  11 00083          BRB     7$                                      ;
                                7E      0389    8F  3C 00085 6$:      MOVZWL  #905, -(SP)                             ; 0458
                        0000V   CF              01  FB 0008A 7$:      CALLS   #1, OUTPUT_PSECTS                       ;
                                52              50  D0 0008F          MOVL    R0, STATUS                              ;
                                2D              52  E9 00092          BLBC    STATUS, 11$                             ;
                0B              65              06  E1 00095          BBC     #6, CLI_FLAGS, 8$                       ; 0464
                        0000V   CF              00  FB 00099          CALLS   #0, OUTPUT_GSD                          ; 0466
                                52              50  D0 0009E          MOVL    R0, STATUS                              ;
                                1E              52  E9 000A1          BLBC    STATUS, 11$                             ;
                                                65  95 000A4 8$:      TSTB    CLI_FLAGS                               ; 0472
                                                04  19 000A6          BLSS    9$                                      ;
                0B              65              02  E1 000A8          BBC     #2, CLI_FLAGS, 10$                      ;
                        0000V   CF              00  FB 000AC 9$:      CALLS   #0, OUTPUT_SECTION                      ; 0474
                                52              50  D0 000B1          MOVL    R0, STATUS                              ;
                                0B              52  E9 000B4          BLBC    STATUS, 11$                             ;
                        0000V   CF              00  FB 000B7 10$:     CALLS   #0, OUTPUT_EOM                          ; 0480
                                52              50  D0 000BC          MOVL    R0, STATUS                              ;
                                0D              52  E8 000BF          BLBS    STATUS, 12$                             ;
                                                52  DD 000C2 11$:     PUSHL   STATUS                                  ;
            00000000G           00              01  FB 000C4          CALLS   #1, LIB$SIGNAL                          ;
                                50              52  D0 000CB          MOVL    STATUS, R0                              ;
                                                    04 000CE          RET                                             ;
                                                54  DD 000CF 12$:     PUSHL   R4                                      ; 0486
            00000000G           00              01  FB 000D1          CALLS   #1, SYS$CLOSE                           ;
                                53              50  D0 000D8          MOVL    R0, STATUS                              ;
                                11              53  E8 000DB          BLBS    STATUS, 15$                             ; 0487
                                                54  DD 000DE          PUSHL   R4                                      ; 0490
                                    0097121A    8F  DD 000E0          PUSHL   #9900570                                ;
                        0000G   CF              02  FB 000E6 13$:     CALLS   #2, RMS_ERROR                           ;
                                50              53  D0 000EB 14$:     MOVL    STATUS, R0                              ; 0491
                                                    04 000EE          RET                                             ;
                                50              01  D0 000EF 15$:     MOVL    #1, R0                                  ; 0494
                                                    04 000F2          RET                                             ; 0496

; Routine Size:  243 bytes,    Routine Base:  $CODE$ + 0000
```

```
;  264     0497  1 ROUTINE create_section =
;  265     0498  1
;  266     0499  1 !---
;  267     0500  1 !
;  268     0501  1 !       This routine creates the message section exactly as
;  269     0502  1 !       it will look when linked into the user image.  The
;  270     0503  1 !       resulting structure will be output as TIR records to
;  271     0504  1 !       the object module.
;  272     0505  1 !
;  273     0506  1 ! Inputs:
;  274     0507  1 !
;  275     0508  1 !       facility_header = List head for facility definitions
;  276     0509  1 !       message_header = List head for message definitions
;  277     0510  1 !       num_messages = Number of message definitions in the list
;  278     0511  1 !       msg_space = Space used by MSG records alone
;  279     0512  1 !       num_facilities = Number of facility definitions in the list
;  280     0513  1 !       fac_space = Space needed for facility table
;  281     0514  1 !
;  282     0515  1 ! Outputs:
;  283     0516  1 !
;  284     0517  1 !       r0 = status (unsignaled)
;  285     0518  1 !       section = Address of final message section
;  286     0519  1 !---
;  287     0520  1
;  288     0521  2 BEGIN
;  289     0522  2
;  290     0523  2 LITERAL
;  291     0524  2     bucket_size = 512,                  ! Size of each index bucket
;  292     0525  2     bucket_overhead = midx$c_entries,   ! Overhead in each bucket
;  293     0526  2     usable_size = bucket_size-bucket_overhead,  ! Useable space in each bucket
;  294     0527  2     entry_size = 8,                     ! Size of each  ndex entry
;  295     0528  2     entries_per_blk = usable_size/entry_size,
;  296     0529  2     max_1level = 3*bucket_size;         ! Maximum size of 1 level index
;  297     0530  2                                         ! before 2 level index will be created
;  298     0531  2
;  299     0532  2 LOCAL
;  300     0533  2     prim_index_len,                     ! Length of primary index
;  301     0534  2     sec_index_len,                      ! Length of secondary index (0 if none)
;  302     0535  2     sec_index_bkts,                     ! # buckets in secondary index
;  303     0536  2     section_size,                       ! Total size of message section
;  304     0537  2     prim_index:         REF VECTOR,     ! Current position in primary index
;  305     0538  2     bucket:             REF BBLOCK,     ! Address of current index bucket
;  306     0539  2     bucket_filled,                      ! Bytes allocated in current bucket
;  307     0540  2     data:               REF BBLOCK,     ! Address of next available data byte
;  308     0541  2     code:               REF BBLOCK,     ! Address of current CODE block
;  309     0542  2     fac:                REF BBLOCK,     ! Address of current FAC block
;  310     0543  2     status;                             ! Status code
;  311     0544  2
;  312     0545  2 MACRO
;  313   M 0546  2     round_div(i,j) =
;  314     0547  2         ((i/j) + ((i MOD j) NEQ 0))%; ! i/j rounded up if remainder GTR 0
;  315     0548  2
;  316     0549  2 IF .num_messages LEQ 0                  ! If no messages defined,
;  317     0550  2 THEN
;  318     0551  2     RETURN emsg(nomsgs);                ! then return with error
;  319     0552  2
;  320     0553  2 !
```

```
;  321          0554  2 !       Compute the amount of space needed to hold the index
;  322          0555  2 !       and text portions of the message section.
;  323          0556  2 !
;  324          0557  2
;  325          0558  2 prim_index_len = .num_messages * entry_size + bucket_overhead;
;  326          0559  2
;  327          0560  2 IF .prim_index_len GTR max_1level       ! If index too large,
;  328          0561  2 THEN
;  329          0562  3     BEGIN                               ! Setup a 2 level index
;  330          0563  3     sec_index_bkts = round_div(.num_messages*entry_size,usable_size);
;  331          0564  3
;  332          0565  3     sec_index_len = .num_messages * entry_size  ! Size of secondary index
;  333          0566  3         + .sec_index_bkts * bucket_overhead;    ! plus bucket overhead
;  334          0567  3
;  335          0568  3     prim_index_len = .sec_index_bkts * entry_size ! Set size of primary index
;  336          0569  3                    + bucket_overhead        ! plus bucket overhead
;  337          0570  3     END
;  338          0571  2 ELSE
;  339          0572  2     sec_index_len = 0;                  ! Else no secondary index (useless)
;  340          0573  2
;  341          0574  2 !
;  342          0575  2 !       Compute the amount of space needed for the entire message section
;  343          0576  2 !       including the facility table.
;  344          0577  2 !
;  345          0578  2
;  346          0579  2 section_size = msc$c_length + .prim_index_len + .sec_index_len
;  347          0580  2                  + .fac_space + .msg_space;
;  348          0581  2
;  349          0582  2 !
;  350          0583  2 !       Allocate enough space for the entire thing
;  351          0584  2 !
;  352          0585  2
;  353          0586  2 status = allocate(.section_size,section);! Allocate the space, signal error
;  354          0587  2 IF NOT .status                          ! If error detected,
;  355          0588  2 THEN
;  356          0589  2     RETURN .status;                     ! then return with status
;  357          0590  2
;  358          0591  2 !
;  359          0592  2 !       Construct the index and data portions of the section
;  360          0593  2 !
;  361          0594  2
;  362          0595  2 CH$FILL(0,msc$c_length,.section);       ! Zero the section header
;  363          0596  2 section [msc$b_type] = msc$c_msg;       ! Set type of section
;  364          0597  2 section [msc$w_sanity] = msc$c_sanity;  ! Set sanity check word
;  365          0598  2 section [msc$l_size] = .section_size;   ! Set size of entire section
;  366          0599  2 section [msc$l_index_off] = msc$c_length; ! Offset to primary index
;  367          0600  2 section [msc$l_text_off] = msc$c_length+.prim_index_len+.sec_index_len;
;  368          0601  2 section [msc$l_fac_off] = .section [msc$l_text_off] + .msg_space;
;  369          0602  2
;  370          0603  2 !
;  371          0604  2 !       Initialize the primary index
;  372          0605  2 !
;  373          0606  2
;  374          0607  3 BEGIN
;  375          0608  3 BIND
;  376          0609  3     prim_bucket = .section+msc$c_length: BBLOCK;        ! Address primary index
;  377          0610  3 prim_bucket [midx$w_size] = .prim_index_len;    ! Set length of index
```

```
;  378     0611  3 prim_bucket [midx$b_sanity] = midx$c_sanity;     ! Set sanity check byte
;  379     0612  3 prim_index = prim_bucket + midx$c_entries;              ! Position at entry 1
;  380     0613  2 END;
;  381     0614  2
;  382     0615  2 !
;  383     0616  2 !       Setup the secondary index pointers
;  384     0617  2 !
;  385     0618  2
;  386     0619  2 bucket = .section+msc$c_length + .prim_index_len;       ! Point at sec. index
;  387     0620  2 bucket_filled = 0;                          ! Nothing filled yet
;  388     0621  2
;  389     0622  2 code = .message_header;                     ! Point at first message definition
;  390     0623  2 data = .section + .section [msc$l_text_off];    ! Point at data area
;  391     0624  2
;  392     0625  2 !
;  393     0626  2 !       Loop thru each message definition storing the
;  394     0627  2 !       message definition record and the index entry.
;  395     0628  2 !
;  396     0629  2
;  397     0630  2 INCR i FROM 1 TO .num_messages              ! For each message definition,
;  398     0631  2 DO
;  399     0632  3     BEGIN
;  400     0633  3     BIND
;  401     0634  3         msg = code [code$c_msg,0,0,0]: BBLOCK;  ! Reference MSG block
;  402     0635  3     LOCAL
;  403     0636  3         sec_index:      REF VECTOR;     ! Address of secondary index entry
;  404     0637  3
;  405     0638  3     !
;  406     0639  3     !   Store the index to the record in the primary or secondary
;  407     0640  3     !   index depending on whether we're doing a 1 or 2 level index.
;  408     0641  3     !
;  409     0642  3
;  410     0643  3     IF .sec_index_len NEQ 0             ! If doing a 2 level index,
;  411     0644  3     THEN
;  412     0645  4         BEGIN
;  413     0646  4         IF .bucket_filled EQL 0         ! If at start of bucket,
;  414     0647  4         THEN
;  415     0648  5             BEGIN
;  416     0649  5             bucket [midx$w_size] = 0;           ! Preset size of bucket
;  417     0650  5             bucket [midx$b_sanity] = midx$c_sanity;     ! Set sanity check byte
;  418     0651  5             bucket_filled = bucket_overhead;    ! Account for space
;  419     0652  5             sec_index = .bucket + bucket_overhead;      ! Set to first entry
;  420     0653  4             END;
;  421     0654  4         END
;  422     0655  3     ELSE
;  423     0656  3         sec_index = .prim_index;        ! Else, store directly into primary
;  424     0657  3
;  425     0658  3     IF .sec_index GEQ .section + .section [msc$l_text_off] ! If overflow,
;  426     0659  3     THEN
;  427     0660  4         BEGIN
;  428     0661  4         SIGNAL(emsg(indexovfl));                ! Signal index area overflow
;  429     0662  4         RETURN emsg(indexovfl);
;  430     0663  3         END;
;  431     0664  3
;  432     0665  3     sec_index [0] = .code [code$l_number]       ! Insert message code in index
;  433     0666  3                     AND NOT sts$m_severity; ! but leave severity bits zero
;  434     0667  3     sec_index [1] = .data - .section;           ! Store offset to record
```

```
;  435     0668  3
;  436     0669  3      !
;  437     0670  3      !   Store message definition record into data area
;  438     0671  3      !
;  439     0672  3
;  440     0673  3      IF .data GEQ .section + .section [msc$l_fac_off] ! If overflow,
;  441     0674  3      THEN
;  442     0675  4          BEGIN
;  443     0676  4          SIGNAL(emsg(dataovfl));                 ! Signal data area overflow
;  444     0677  4          RETURN emsg(dataovfl);
;  445     0678  3          END;
;  446     0679  3
;  447     0680  3      CH$MOVE(.msg [mrec$w_size], msg, .data);    ! Store MSG record in data area
;  448     0681  3
;  449     0682  3      data = .data + .msg [mrec$w_size];          ! Skip to next available byte
;  450     0683  3
;  451     0684  3      !
;  452     0685  3      !   If doing 2 level index, increment space used in bucket.
;  453     0686  3      !   Check for secondary index bucket full.  If full, then
;  454     0687  3      !   put the highest key in the bucket into the primary index.
;  455     0688  3      !
;  456     0689  3
;  457     0690  3      IF .sec_index_len NEQ 0             ! If doing 2 level index,
;  458     0691  3      THEN
;  459     0692  4          BEGIN
;  460     0693  4          bucket_filled = .bucket_filled + entry_size;! Account for space used
;  461     0694  4
;  462     0695  4          IF .bucket_filled GTR bucket_size-entry_size ! If last entry in bucket,
;  463     0696  4              OR .i EQL .num_messages                  ! or last message
;  464     0697  4          THEN
;  465     0698  5              BEGIN
;  466     0699  5              bucket [midx$w_size] = .bucket_filled;       ! Set size of bucket
;  467     0700  5              IF .prim_index GEQ .section + .section [msc$l_index_off]
;  468     0701  5                                 + .prim_index_len ! If overflow,
;  469     0702  5              THEN
;  470     0703  6                  BEGIN
;  471     0704  6                  SIGNAL(emsg(indexovfl));        ! Signal index area overflow
;  472     0705  6                  RETURN emsg(indexovfl);
;  473     0706  5                  END;
;  474     0707  5              prim_index [0] = .sec_index [0];    ! Store highest entry in bucket
;  475     0708  6              prim_index [1] = (.bucket - .section) ! Store offset to bucket
;  476     0709  5                               OR 1;              ! with bottom bit set (subindex)
;  477     0710  5              prim_index = .prim_index + entry_size;      ! Set to next entry
;  478     0711  5              bucket = .bucket + bucket_size;             ! Set to next bucket
;  479     0712  5              bucket_filled = 0;                  ! reset usage counter
;  480     0713  5              END
;  481     0714  4          ELSE                                ! Else, if still more room in bucket,
;  482     0715  4              sec_index = .sec_index + entry_size;        ! Set to next entry
;  483     0716  4          END
;  484     0717  4
;  485     0718  4      !
;  486     0719  4      !   If only doing a 1 level index, then we need only to
;  487     0720  4      !   increment to the next position in the primary index.
;  488     0721  4      !
;  489     0722  4
;  490     0723  3      ELSE                                ! Else, if only doing primary index,
;  491     0724  3          prim_index = .prim_index + entry_size;  ! Set to next primary entry
```

```
;  492          0725  3
;  493          0726  3      code = .code [code$l_link];          ! and skip to next CODE block
;  494          0727  2      END;
;  495          0728  2
;  496          0729  2 !
;  497          0730  2 !       Generate the facility definition table in the section
;  498          0731  2 !
;  499          0732  2
;  500          0733  2 data [0,0,16,0] = .fac_space - 2;        ! Set size of entire table (excl. size)
;  501          0734  2 data = .data + 2;
;  502          0735  2
;  503          0736  2 fac = .facility_header;                  ! Start at first entry in list
;  504          0737  2 INCR i FROM 1 TO .num_facilities         ! For each facility definition,
;  505          0738  2 DO
;  506          0739  3     BEGIN
;  507          0740  3     IF .data GEQ .section + .section [msc$l_size] ! If overflow,
;  508          0741  3     THEN
;  509          0742  4         BEGIN
;  510          0743  4         SIGNAL(emsg(facovfl));          ! Signal facility table overflow
;  511          0744  4         RETURN emsg(facovfl);
;  512          0745  3         END;                                                              ; R
;  513          0746  3     data [mfac$w_number] = .fac [fac$w_number]; ! Store facility number
;  514          0747  3     data [mfac$b_namelen] = .fac [fac$b_namelen]; ! and facility name
;  515          0748  3     CH$MOVE(.fac [fac$b_namelen], fac [fac$t_name], data [mfac$t_name]);
;  516          0749  3     data = .data + $BYTEOFFSET(mfac$t_name) + .fac [fac$b_namelen];
;  517          0750  3     fac = .fac [fac$l_link];            ! Link to next in chain
;  518          0751  2     END;
;  519          0752  2
;  520          0753  2 RETURN true;
;  521          0754  2
;  522          0755  1 END;


                                                .EXTRN  MSG$_NOMSGS, MSG$_INDEXOVFL
                                                .EXTRN  MSG$_DATAOVFL, MSG$_FACOVFL

                                    OFFC 00000 CREATE_SECTION:
                                                .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11        ; 0497
                          5E                 14  C2 00002         SUBL2   #20, SP                     ;
                          51      0000G      CF  D0 00005         MOVL    NUM_MESSAGES, R1            ; 0549
                                             08  14 0000A         BGTR    1$                          ;
                          50 00000000G       8F  D0 0000C         MOVL    #MSG$_NOMSGS, R0            ; 0551
                                             04  00013           RET                                 ;
                59                51           03  78 00014 1$:  ASHL    #3, R1, PRIM_INDEX_LEN      ; 0558
                                  59           08  C0 00018       ADDL2   #8, PRIM_INDEX_LEN          ;
                    00000600      8F           59  D1 0001B       CMPL    PRIM_INDEX_LEN, #1536       ; 0560
                                               37  15 00022       BLEQ    3$                          ;
                53                51           03  78 00024       ASHL    #3, R1, R3                  ; 0563
                                  53 000001F8  8F  C6 00028       DIVL2   #504, R3                    ;
                50                51           03  78 0002F       ASHL    #3, R1, R0                  ;
          7E    00                50           01  7A 00033       EMUL    #1, R0, #0, -(SP)           ;
          50    50                8E 000001F8  8F  7B 00038       EDIV    #504, (SP)+, R0, R0         ;
                                               52  D4 00041       CLRL    R2                          ;
                                               50  D5 00043       TSTL    R0                          ;
                                               02  13 00045       BEQL    2$                          ;
                                               52  D6 00047       INCL    R2                          ;
```

```
                50              53          52  C1 00049 2$:    ADDL3   R2, R3, SEC_INDEX_BKTS                  ;
                                50          08  C4 0004D        MULL2   #8, R0                                 ; 0566
                        10      AE      6041    7E 00050        MOVAQ   (R0)[R1], SEC_INDEX_LEN                 ;
                                59    08    A0  9E 00055        MOVAB   8(R0), PRIM_INDEX_LEN                   ; 0569
                                            03  11 00059        BRB     4$                                      ; 0568
                                      10    AE  D4 0005B 3$:    CLRL    SEC_INDEX_LEN                           ; 0572
                58              59    10    AE  C1 0005E 4$:    ADDL3   SEC_INDEX_LEN, PRIM_INDEX_LEN, R8       ; 0579
                57              58  0000G   CF  C1 00063        ADDL3   FAC_SPACE, R8, R7                       ; 0580
                                57  0000G   CF  C0 00069        ADDL2   MSG_SPACE, R7                           ;
                                57          28  C0 0006E        ADDL2   #40, SECTION_SIZE                       ;
                                        0000' CF  9F 00071        PUSHAB  SECTION                                 ; 0586
                                            57  DD 00075        PUSHL   SECTION_SIZE                            ;
                        0000G   CF          02  FB 00077        CALLS   #2, ALLOCATE                            ;
                                01          50  E8 0007C        BLBS    STATUS, 5$                              ; 0587
                                            04  0007F          RET                                             ;
                                56      0000' CF  D0 00080 5$:  MOVL    SECTION, R6                             ; 0595
        28              00      6E          00  2C 00085        MOVC5   #0, (SP), #0, #40, (R6)                 ;
                                            66      0008A                                                       ;
                                            66  94  0008B        CLRB    (R6)                                    ; 0596
                        02  A6      04D2    8F  B0 0008D        MOVW    #1234, 2(R6)                            ; 0597
                        04  A6              57  D0 00093        MOVL    SECTION_SIZE, 4(R6)                     ; 0598
                        08  A6              28  D0 00097        MOVL    #40, 8(R6)                              ; 0599
                        10  A6        28    A8  9E 0009B        MOVAB   40(R8), 16(R6)                          ; 0600
                0C  A6  10  A6      0000G   CF  C1 000A0        ADDL3   MSG_SPACE, 16(R6), 12(R6)               ; 0601
                                57    28    A6  9E 000A8        MOVAB   40(R6), R7                              ; 0609
                                87          59  B0 000AC        MOVW    PRIM_INDEX_LEN, (R7)+                   ; 0610
                                87    7B    8F  90 000AF        MOVB    #123, (R7)+                             ; 0611
                                57          05  C0 000B3        ADDL2   #5, PRIM_INDEX                          ; 0612
                                58    28  A946  9E 000B6        MOVAB   40(PRIM_INDEX_LEN)[R6], BUCKET          ; 0619
                                5A      0000G   CF  D0 000BB        MOVL    MESSAGE_HEADER, CODE                    ; 0622
                5B              56        10    A6  C1 000C0        ADDL3   16(R6), R6, DATA                        ; 0623
                        0C      AE      0000G   CF  D0 000C5        MOVL    NUM_MESSAGES, 12(SP)                    ; 0630
                                            6E  7C 000CB        CLRQ    BUCKET_FILLED                           ; 0620
                                        00E1    31 000CD        BRW     16$                                     ; 0630
                                      08    AE  D4 000D0 6$:    CLRL    8(SP)                                   ; 0643
                                      10    AE  D5 000D3        TSTL    SEC_INDEX_LEN                           ;
                                            17  13 000D6        BEQL    7$                                      ;
                                      08    AE  D6 000D8        INCL    8(SP)                                   ;
                                            6E  D5 000DB        TSTL    BUCKET_FILLED                           ; 0646
                                            13  12 000DD        BNEQ    8$                                      ;
                                            68  B4 000DF        CLRW    (BUCKET)                                ; 0649
                        02  A8          7B    8F  90 000E1        MOVB    #123, 2(BUCKET)                         ; 0650
                                6E          08  D0 000E6        MOVL    #8, BUCKET_FILLED                       ; 0651
                                56          08  A8  9E 000E9        MOVAB   8(R8), SEC_INDEX                        ; 0652
                                            03  11 000ED        BRB     8$                                      ; 0643
                                56              57  D0 000EF 7$:    MOVL    PRIM_INDEX, SEC_INDEX                   ; 0656
                                50      0000' CF  D0 000F2 8$:    MOVL    SECTION, R0                             ; 0658
                                50        10  A0  C0 000F7        ADDL2   16(R0), R0                              ;
                                50              56  D1 000FB        CMPL    SEC_INDEX, R0                           ;
                                            15  19 000FE        BLSS    9$                                      ;
                                  00000000G     8F  DD 00100        PUSHL   #MSG$_INDEXOVFL                         ; 0661
                    00000000G   00              01  FB 00106        CALLS   #1, LIB$SIGNAL                          ;
                                50 00000000G    8F  D0 0010D        MOVL    #MSG$_INDEXOVFL, R0                     ; 0662
                                            04  00114        RET                                             ;
        66      04  A4              07  CB  00115 9$:    BICL3   #7, 4(CODE), (SEC_INDEX)                       ; 0666
                                50      0000' CF  D0 0011A        MOVL    SECTION, R0                             ; 0667
    04  A6                      5B          50  C3 0011F        SUBL3   R0, DATA, 4(SEC_INDEX)                  ;
```

```
                     50        0C  A0   C0 00124         ADDL2   12(R0), R0                          ; 0673
                     50            5B   D1 00128         CMPL    DATA, R0                            ;
                                   15   19 0012B         BLSS    10$                                 ;
                         00000000G 8F   DD 0012D         PUSHL   #MSG$_DATAOVFL                      ; 0676
             00000000G   00        01   FB 00133         CALLS   #1, LIB$SIGNAL                      ;
                     50  00000000G 8F   D0 0013A         MOVL    #MSG$_DATAOVFL, R0                  ; 0677
                                   04      00141         RET                                         ;
        6B      08   AA        08  AA   28 00142  10$:   MOVC3   8(CODE), 8(CODE), (DATA)            ; 0680
                     50        08  AA   3C 00148         MOVZWL  8(CODE), R0                         ; 0682
                     5B            50   C0 0014C         ADDL2   R0, DATA                            ;
                     58        08  AE   E9 0014F         BLBC    8(SP), 14$                          ; 0690
                     6E            08   C0 00153         ADDL2   #8, BUCKET_FILLED                   ; 0693
             000001F8    8F        6E   D1 00156         CMPL    BUCKET_FILLED, #504                 ; 0695
                                   08   14 0015D         BGTR    11$                                 ;
                 0000G   CF        04  AE   D1 0015F     CMPL    I, NUM_MESSAGES                     ; 0696
                                   3F   12 00165         BNEQ    13$                                 ;
                     68            6E   B0 00167  11$:   MOVW    BUCKET_FILLED, (BUCKET)             ; 0699
                     50      0000'  CF   D0 0016A        MOVL    SECTION, R0                         ; 0700
                     50        08  A0   C0 0016F         ADDL2   8(R0), R0                           ;
                     50            59   C0 00173         ADDL2   PRIM_INDEX_LEN, R0                  ; 0701
                     50            57   D1 00176         CMPL    PRIM_INDEX, R0                      ;
                                   15   19 00179         BLSS    12$                                 ;
                         00000000G 8F   DD 0017B         PUSHL   #MSG$_INDEXOVFL                     ; 0704
             00000000G   00        01   FB 00181         CALLS   #1, LIB$SIGNAL                      ;
                     50  00000000G 8F   D0 00188         MOVL    #MSG$_INDEXOVFL, R0                 ; 0705
                                   04      0018F         RET                                         ;
                     87            66   D0 00190  12$:   MOVL    (SEC_INDEX), (PRIM_INDEX)+          ; 0707
        56           58      0000'  CF   C3 00193        SUBL3   SECTION, BUCKET, R6                 ; 0708
        87           56            01   C9 00199         BISL3   #1, R6, (PRIM_INDEX)+               ; 0709
                     58      0200  C8   9E 0019D         MOVAB   512(R8), BUCKET                     ; 0711
                                   6E   D4 001A2         CLRL    BUCKET_FILLED                       ; 0712
                                   08   11 001A4         BRB     15$                                 ; 0695
                     56            08   C0 001A6  13$:   ADDL2   #8, SEC_INDEX                       ; 0715
                                   03   11 001A9         BRB     15$                                 ; 0690
                     57            08   C0 001AB  14$:   ADDL2   #8, PRIM_INDEX                      ; 0724
                     5A            6A   D0 001AE  15$:   MOVL    (CODE), CODE                        ; 0726
   FF17      04  AE  01        0C  AE   F1 001B1  16$:   ACBL    12(SP), #1, I, 6$                   ; 0630
        8B      0000G  CF          02   A3 001B9         SUBW3   #2, FAC_SPACE, (DATA)+              ; 0733
                     56      0000G  CF   D0 001BF        MOVL    FACILITY_HEADER, FAC                ; 0736
                     5A      0000G  CF   D0 001C4        MOVL    NUM_FACILITIES, R10                 ; 0737
                                   58   D4 001C9         CLRL    I                                   ;
                                   3D   11 001CB         BRB     19$                                 ;
                     50      0000'  CF   D0 001CD  17$:  MOVL    SECTION, R0                         ; 0740
                     50        04  A0   C0 001D2         ADDL2   4(R0), R0                           ;
                     50            5B   D1 001D6         CMPL    DATA, R0                            ;
                                   15   19 001D9         BLSS    18$                                 ;
                         00000000G 8F   DD 001DB         PUSHL   #MSG$_FACOVFL                       ; 0743
             00000000G   00        01   FB 001E1         CALLS   #1, LIB$SIGNAL                      ;
                     50  00000000G 8F   D0 001E8         MOVL    #MSG$_FACOVFL, R0                   ; 0744
                                   04      001EF         RET                                         ;
                     6B        04  A6   B0 001F0  18$:   MOVW    4(FAC), (DATA)                      ; 0746
                     57        06  A6   9A 001F4         MOVZBL  6(FAC), R7                          ; 0747
                02   AB            57   90 001F8         MOVB    R7, 2(DATA)                         ;
        03   AB  07  A6            57   28 001FC         MOVC3   R7, 7(FAC), 3(DATA)                 ; 0748
                     5B      03 A74B    9E 00202         MOVAB   3(R7)[DATA], DATA                   ; 0749
                     56            66   D0 00207         MOVL    (FAC), FAC                          ; 0750
        BF           58            5A   F3 0020A  19$:   AOBLEQ  R10, I, 17$                         ; 0737
```

```
                50          01  DO 0020E       MOVL    #1, R0                          ; 0753
                                04 00211       RET                                     ; 0755

; Routine Size:  530 bytes,    Routine Base:  $CODE$ + 00F3
```

```
;  524          0756  1 ROUTINE create_indirect =
;  525          0757  1
;  526          0758  1 !---
;  527          0759  1 !
;  528          0760  1 !       Create the indirect message section.  This section
;  529          0761  1 !       will merely contain the name of the message file which
;  530          0762  1 !       should be mapped at run-time by SYS$GETMSG.
;  531          0763  1 !
;  532          0764  1 ! Inputs:
;  533          0765  1 !
;  534          0766  1 !       filename_desc = Descriptor of /FILE value
;  535          0767  1 !
;  536          0768  1 ! Outputs:
;  537          0769  1 !
;  538          0770  1 !       r0 = status (unsignaled)
;  539          0771  1 !       section = Address of section
;  540          0772  1 !---
;  541          0773  1
;  542          0774  2 BEGIN
;  543          0775  2
;  544          0776  2 LOCAL
;  545          0777  2     status,                                 ! status code
;  546          0778  2     section_size;                           ! Size of section in bytes
;  547          0779  2
;  548          0780  2 section_size = $BYTEOFFSET(msc$t_indname) + .filename_desc [dsc$w_length];
;  549          0781  2
;  550          0782  2 status = allocate(.section_size,section); ! Allocate space for section
;  551          0783  2 IF NOT .status                            ! If error allocating space
;  552          0784  2 THEN
;  553          0785  2     RETURN .status;                         ! then return with status
;  554          0786  2
;  555          0787  2 section [msc$b_type] = msc$c_ind;        ! Set type to indirect
;  556          0788  2 section [msc$b_flags] = 0;               ! Initialize flags as all zero.
;  557          0789  2 section [msc$w_sanity] = msc$c_sanity;   ! Set sanity check word
;  558          0790  2 section [msc$l_size] = .section_size;    ! Set size of section
;  559          0791  2 section [msc$b_indnamlen] = .filename_desc [dsc$w_length];      ! Set length of name
;  560          0792  2 CH$MOVE(.filename_desc [dsc$w_length], .filename_desc [dsc$a_pointer], section [msc$t_indname]);
;  561          0793  2
;  562          0794  2 RETURN true;                             ! return with success
;  563          0795  2
;  564          0796  1 END;


                                007C 00000 CREATE_INDIRECT:
                                           .WORD   Save R2,R3,R4,R5,R6                                  ; 0756
                     56     0000G  CF  9E 00002         MOVAB   FILENAME_DESC, R6                       ;
                     52            66  3C 00007         MOVZWL  FILENAME_DESC, SECTION_SIZE              ; 0780
                     52            09  C0 0000A         ADDL2   #9, SECTION_SIZE                        ;
                           0000'    CF  9F 0000D         PUSHAB  SECTION                                 ; 0782
                                    52  DD 00011         PUSHL   SECTION_SIZE                            ;
                0000G  CF           02  FB 00013         CALLS   #2, ALLOCATE                            ;
                     1D             50  E9 00018         BLBC    STATUS, 1$                              ; 0783
                     50     0000'    CF  D0 0001B         MOVL    SECTION, R0                             ; 0787
                     60 04D20001     8F  D0 00020         MOVL    #80871425, (R0)                         ;
```

```
                    04   A0          52  D0 00027          MOVL    SECTION_SIZE, 4(R0)                          ; 0790
                    08   A0          66  90 0002B          MOVB    FILENAME_DESC, 8(R0)                         ; 0791
              09    A0   04   B6      66  28 0002F          MOVC3   FILENAME_DESC, @FILENAME_DESC+4, 9(R0)       ; 0792
                         50          01  D0 00035          MOVL    #1, R0                                       ; 0794
                                         04 00038 1$:      RET                                                  ; 0796

; Routine Size:  57 bytes,    Routine Base:  $CODE$ + 0305
```

```
;  566          0797  1 ROUTINE output_mhd =
;  567          0798  1
;  568          0799  1 !---
;  569          0800  1 !
;  570          0801  1 !       Output a module header record.
;  571          0802  1 !
;  572          0803  1 ! Inputs:
;  573          0804  1 !
;  574          0805  1 !       None
;  575          0806  1 !
;  576          0807  1 ! Outputs:
;  577          0808  1 !
;  578          0809  1 !       Module header record is output.
;  579          0810  1 !---
;  580          0811  1
;  581          0812  2 BEGIN
;  582          0813  2
;  583          0814  2 BIND
;  584          0815  2     lang_name = UPLIT('VAX-11 Message V04-00');
;  585          0816  2
;  586          0817  2 LITERAL
;  587          0818  2     lang_length = 21;
;  588          0819  2
;  589          0820  2 LOCAL
;  590          0821  2     rec:        BBLOCK[128],            ! Allocate record buffer
;  591          0822  2     offset,                             ! Offset to current spot in record
;  592          0823  2     bufdesc:    VECTOR[2];              ! General buffer descriptor
;  593          0824  2
;  594          0825  2 rec [obj$b_rectyp] = obj$c_hdr;         ! Set record type = HDR
;  595          0826  2 rec [mhd$b_hdrtyp] = mhd$c_mhd;         ! Main module header record
;  596          0827  2 rec [mhd$b_strlvl] = 0;                 ! Structure level 0
;  597          0828  2 rec [mhd$w_recsiz] = obj$c_maxrecsiz;   ! Maximum record size
;  598          0829  2
;  599          0830  2 rec [mhd$b_namlng] = .module_name [0];  ! Length of module name
;  600          0831  2 CH$MOVE(.module_name [0], .module_name[1], rec [mhd$t_name]);
;  601          0832  2 offset = $BYTEOFFSET(mhd$t_name) + .module_name [0];
;  602          0833  2
;  603          0834  2 IF .version_num [0]  EQL  0
;  604          0835  2 THEN
;  605          0836  2     !
;  606          0837  2     ! No .IDENT was specified
;  607          0838  2     !
;  608          0839  3     BEGIN
;  609          0840  3     rec [.offset,0,8,0] = 1;
;  610          0841  3     rec [.offset,8,8,0] = '0';
;  611          0842  3     offset = .offset + 2;
;  612          0843  3     END
;  613          0844  3
;  614          0845  2 ELSE
;  615          0846  2     !
;  616          0847  2     ! .IDENT was specified
;  617          0848  2     !
;  618          0849  3     BEGIN
;  619          0850  3     rec [.offset,0,8,0] = .version_num [0];
;  620          0851  3     offset = .offset + 1;
;  621          0852  3     CH$MOVE(.version_num [0],.version_num [1],rec [.offset,0,0,0]);
;  622          0853  3     offset = .offset + .version_num [0] ;
```

```
;  623          0854  3     version_num [0] = 0;
;  624          0855  2     END;
;  625          0856  2
;  626          0857  2 bufdesc [0] = 17;                       ! Length of date/time string
;  627          0858  2 bufdesc [1] = rec + .offset;            ! Address to put creation date/time
;  628          0859  2 $ASCTIM(TIMBUF=bufdesc);                ! Put creation date/time into buffer
;  629          0860  2 offset = .offset + 17;
;  630          0861  2
;  631          0862  2 bufdesc [1] = .bufdesc [1] + 17;        ! Address to put last patch date/time
;  632          0863  2 $ASCTIM(TIMBUF=bufdesc);                ! Put time of last patch into buffer
;  633          0864  2 offset = .offset + 17;
;  634          0865  2
;  635          0866  2 put_record (rec, .offset);             ! Output record
;  636          0867  2
;  637          0868  2 rec [mhd$b_hdrtyp] = mhd$c_lnm;        ! Language name and version
;  638          0869  2 CH$MOVE(lang_length, lang_name, rec + $BYTEOFFSET(mhd$b_hdrtyp)+1);
;  639          0870  2
;  640          0871  2 put_record(rec, $BYTEOFFSET(mhd$b_hdrtyp)+1+lang_length);
;  641          0872  2
;  642          0873  2 RETURN true;
;  643          0874  2
;  644          0875  1 END;


                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

20 65 67 61 73 73 65 4D 20 31 31 2D 58 41 56 00000 P.AAA:  .ASCII  \VAX-11 Message V04-00\<0><0><0>                ;
                  00 00 00 30 30 2D 34 30 56 0000F                                                              ;

                                                LANG_NAME=          P.AAA
                                                        .EXTRN  SYS$ASCTIM

                                                        .PSECT  $CODE$,NOWRT,2

                                        01FC 00000 OUTPUT_MHD:
                                                        .WORD   Save R2,R3,R4,R5,R6,R7,R8               ; 0797
                          58 00000000G   00   9E 00002          MOVAB   SYS$ASCTIM, R8                  ;
                          5E     FF78    CE   9E 00009          MOVAB   -136(SP), SP                    ;
                                  08    AE   B4 0000E          CLRW    REC                             ; 0825
                                  0A    AE   94 00011          CLRB    REC+2                           ; 0827
                    0B    AE      0800  8F   B0 00014          MOVW    #2048, REC+3                    ; 0828
                          56      0000G CF   D0 0001A          MOVL    MODULE_NAME, R6                 ; 0830
                    0D    AE            56   90 0001F          MOVB    R6, REC+5                       ;
OE    AE      0000G   DF              56   28 00023          MOVC3   R6, @MODULE_NAME+4, REC+6       ; 0831
                      56              06   C0 0002A          ADDL2   #6, OFFSET                      ; 0832
                      50        08 AE46   9E 0002D          MOVAB   REC[OFFSET], R0                 ; 0840
                      57      0000G CF   D0 00032          MOVL    VERSION_NUM, R7                 ; 0834
                                    0A    12 00037          BNEQ    1$                              ;
                      60      3001  8F   B0 00039          MOVW    #12289, (R0)                    ; 0840
                      56            02   C0 0003E          ADDL2   #2, OFFSET                      ; 0842
                                    14   11 00041          BRB     2$                              ; 0834
                      60            57   90 00043 1$:      MOVB    R7, (R0)                        ; 0850
                                    56   D6 00046          INCL    OFFSET                          ; 0851
08 AE46       0000G   DF            57   28 00048          MOVC3   R7, @VERSION_NUM+4, REC[OFFSET] ; 0852
                      56            57   C0 00050          ADDL2   R7, OFFSET                      ; 0853
                              0000G CF   D4 00053          CLRL    VERSION_NUM                     ; 0854
```

```
                                 6E              11  D0 00057 2$:    MOVL    #17, BUFDESC                            ; 0857
                           04    AE        08 AE46  9E 0005A         MOVAB   REC[OFFSET], BUFDESC+4                  ; 0858
                                              7E  7C 00060         CLRQ    -(SP)                                   ; 0859
                                           08 AE  9F 00062         PUSHAB  BUFDESC
                                              7E  D4 00065         CLRL    -(SP)
                                 68               04  FB 00067    CALLS   #4, SYS$ASCTIM
                                 56               11  C0 0006A    ADDL2   #17, OFFSET                             ; 0860
                           04    AE               11  C0 0006D    ADDL2   #17, BUFDESC+4                          ; 0862
                                              7E  7C 00071         CLRQ    -(SP)                                   ; 0863
                                           08 AE  9F 00073         PUSHAB  BUFDESC
                                              7E  D4 00076         CLRL    -(SP)
                                 68               04  FB 00078    CALLS   #4, SYS$ASCTIM
                                 56               11  C0 0007B    ADDL2   #17, OFFSET                             ; 0864
                                                  56  DD 0007E    PUSHL   OFFSET                                  ; 0866
                                           0C AE  9F 00080         PUSHAB  REC
                          0000V    CF              02  FB 00083    CALLS   #2, PUT_RECORD
                            09     AE              01  90 00088    MOVB    #1, REC+1                               ; 0868
                  0A AE     0000'  CF              15  28 0008C    MOVC3   #21, LANG_NAME, REC+2                   ; 0869
                                                  17  DD 00093    PUSHL   #23                                     ; 0871
                                           0C AE  9F 00095         PUSHAB  REC
                          0000V    CF              02  FB 00098    CALLS   #2, PUT_RECORD
                                 50               01  D0 0009D    MOVL    #1, R0                                  ; 0873
                                                      04 000A0    RET                                             ; 0875

; Routine Size:  161 bytes,    Routine Base:  $CODE$ + 033E
```

```
;  646          0876  1 ROUTINE output_eom =
;  647          0877  1
;  648          0878  1 !---
;  649          0879  1 !
;  650          0880  1 !       Output an end of module record
;  651          0881  1 !
;  652          0882  1 ! Inputs:
;  653          0883  1 !
;  654          0884  1 !       None
;  655          0885  1 !
;  656          0886  1 ! Outputs:
;  657          0887  1 !
;  658          0888  1 !       The record is output.
;  659          0889  1 !---
;  660          0890  1
;  661          0891  2 BEGIN
;  662          0892  2
;  663          0893  2 LOCAL
;  664          0894  2     rec:        BBLOCK[128];            ! Allocate record buffer
;  665          0895  2
;  666          0896  2 rec [obj$b_rectyp] = obj$c_eom;         ! Set record type = EOM
;  667          0897  2 rec [eom$b_comcod] = eom$c_success;     ! Set completion severity
;  668          0898  2
;  669          0899  2 put_record(rec, 2);                    ! Output record
;  670          0900  2
;  671          0901  2 RETURN true;
;  672          0902  2
;  673          0903  1 END;


                                0000 00000 OUTPUT_EOM:
                                           .WORD   Save nothing                    ; 0876
                     5E    80   AE  9E 00002        MOVAB   -128(SP), SP            ;
                     6E         03  B0 00006        MOVW    #3, REC                 ; 0896
                                02  DD 00009        PUSHL   #2                      ; 0899
                           04   AE  9F 0000B        PUSHAB  REC                     ;
               0000V CF         02  FB 0000E        CALLS   #2, PUT_RECORD          ;
                     50         01  D0 00013        MOVL    #1, R0                  ; 0901
                                    04 00016        RET                             ; 0903

; Routine Size:  23 bytes,    Routine Base:  $CODE$ + 03DF
```

```
;  675     0904  1 ROUTINE output_psects (psect_flags) =
;  676     0905  1
;  677     0906  1 !---
;  678     0907  1 !
;  679     0908  1 !       This routine outputs all the psect definitions
;  680     0909  1 !       for normal message sections.  Only the absolute
;  681     0910  1 !       psect is output if /NOTEXT is specified.
;  682     0911  1 !
;  683     0912  1 ! Inputs:
;  684     0913  1 !
;  685     0914  1 !       psect_flags = Psect attributes
;  686     0915  1 !
;  687     0916  1 ! Outputs:
;  688     0917  1 !
;  689     0918  1 !       The records are output.
;  690     0919  1 !---
;  691     0920  1
;  692     0921  2 BEGIN
;  693     0922  2
;  694     0923  2 LOCAL
;  695     0924  2     entry:      BBLOCK[129];            ! Entry buffer
;  696     0925  2
;  697     0926  2 start_record(obj$c_gsd);               ! Signal start of GSD record
;  698     0927  2
;  699     0928  2 !
;  700     0929  2 !       Define psects used in this object module
;  701     0930  2 !
;  702     0931  2
;  703     0932  2 entry [gps$b_gsdtyp] = gsd$c_psc;       ! Psect definition entry
;  704     0933  2 entry [gps$b_align] = 0;                ! Psect alignment (byte)
;  705     0934  2 entry [gps$w_flags] = gps$m_exe OR gps$m_rd OR gps$m_wrt; ! Psect attributes
;  706     0935  2 entry [gps$l_alloc] = 0;                ! Set size of section
;  707     0936  2 entry [gps$b_namlng] = 5;               ! Length of psect name
;  708     0937  2 CH$MOVE(5,UPLIT('$ABS$'),entry [gps$t_name]);   ! Set psect name
;  709     0938  2 put_entry(entry, $BYTEOFFSET(gps$t_name)+5);    ! Output entry
;  710     0939  2
;  711     0940  2 IF NOT .cli_flags [qual_text]  AND              ! If /NOTEXT specified,
;  712     0941  2    NOT .cli_flags [qual_file]                   !      and no pointers
;  713     0942  2 THEN
;  714     0943  3     BEGIN
;  715     0944  3     end_record();                       ! then flush absolute psect only
;  716     0945  3     RETURN true;                        ! and return
;  717     0946  2     END;
;  718     0947  2
;  719     0948  2 entry [gps$w_flags] = .psect_flags;     ! Psect attributes
;  720     0949  2 entry [gps$l_alloc] = .section [msc$l_size];    ! Set size of section
;  721     0950  2
;  722     0951  2 IF .cli_flags[qual_file]
;  723     0952  2 THEN
;  724     0953  3     BEGIN
;  725     0954  3     entry [gps$b_namlng] = 14;          ! Length of psect name
;  726     0955  3     CH$MOVE(14,UPLIT('MSGPTR$SECTION'),entry [gps$t_name]);! Set ptr psect name
;  727     0956  3     put_entry(entry, $BYTEOFFSET(gps$t_name)+14);       ! Output entry
;  728     0957  3     END
;  729     0958  2 ELSE
;  730     0959  3     BEGIN
;  731     0960  3     entry [gps$b_namlng] = 11;          ! Length of psect name
```

```
;  732          0961  3     CH$MOVE(11,UPLIT('MSG$SECTION'),entry [gps$t_name]);   ! Set psect name
;  733          0962  3     put_entry(entry, $BYTEOFFSET(gps$t_name)+11);          ! Output entry
;  734          0963  2     END;
;  735          0964  2
;  736          0965  2 entry [gps$b_align] = 2;                  ! Psect alignment (longword)
;  737          0966  2 entry [gps$w_flags] = .psect_flags OR gps$m_ovr; ! This psect is overlayed
;  738          0967  2 entry [gps$l_alloc] = 4*4;                ! Size of psect contribution
;  739          0968  2 entry [gps$b_namlng] = 15;                ! Length of psect name
;  740          0969  2
;  741          0970  2 IF .cli_flags[qual_file]
;  742          0971  2 THEN
;  743          0972  2     CH$MOVE(15,UPLIT('MSGPTR$AAAAAAAA'),entry [gps$t_name])! Set ptr psect name
;  744          0973  2 ELSE
;  745          0974  2     CH$MOVE(15,UPLIT('MSG$AAAAAAAAAAA'),entry [gps$t_name]); ! Set psect name
;  746          0975  2
;  747          0976  2 put_entry(entry, $BYTEOFFSET(gps$t_name)+15);   ! Output entry
;  748          0977  2
;  749          0978  2 entry [gps$w_flags] = .psect_flags;      ! Normal psect flags
;  750          0979  2 entry [gps$l_alloc] = 4;                ! Size of psect contribution
;  751          0980  2
;  752          0981  2 IF .cli_flags[qual_file]
;  753          0982  2 THEN
;  754          0983  2     CH$MOVE(15,UPLIT('MSGPTR$AAAAAAAB'),entry [gps$t_name])! Set ptr psect name
;  755          0984  2 ELSE
;  756          0985  2     CH$MOVE(15,UPLIT('MSG$AAAAAAAAAAB'),entry [gps$t_name]); ! Set psect name
;  757          0986  2
;  758          0987  2 put_entry(entry, $BYTEOFFSET(gps$t_name)+15);   ! Output entry
;  759          0988  2
;  760          0989  2 entry [gps$w_flags] = .psect_flags OR gps$m_ovr; ! This psect is overlayed
;  761          0990  2
;  762          0991  2 IF .cli_flags[qual_file]
;  763          0992  2 THEN
;  764          0993  2     CH$MOVE(15,UPLIT('MSGPTR$AAAAAAAC'),entry [gps$t_name])! Set ptr psect name
;  765          0994  2 ELSE
;  766          0995  2     CH$MOVE(15,UPLIT('MSG$AAAAAAAAAAC'),entry [gps$t_name]); ! Set psect name
;  767          0996  2
;  768          0997  2 put_entry(entry, $BYTEOFFSET(gps$t_name)+15);   ! Output entry
;  769          0998  2
;  770          0999  2 end_record();                              ! Flush rest of record buffer
;  771          1000  2
;  772          1001  2 RETURN true;
;  773          1002  2
;  774          1003  1 END;


                                                                    .PSECT  $PLIT$,NOWRT,NOEXE,2

                                                00 00 00 24 53 42 41 24 00018 P.AAB:  .ASCII  \$ABS$\<0><0><0>
00 4E 4F 49 54 43 45 53 24 52 54 50 47 53 4D 00020 P.AAC:  .ASCII  \MSGPTR$SECTION\<0><0>
                                                00 0002F
      00 4E 4F 49 54 43 45 53 24 47 53 4D 00030 P.AAD:  .ASCII  \MSG$SECTION\<0>
41 41 41 41 41 41 41 41 24 52 54 50 47 53 4D 0003C P.AAE:  .ASCII  \MSGPTR$AAAAAAAA\<0>
                                                00 0004B
41 41 41 41 41 41 41 41 41 41 24 47 53 4D 0004C P.AAF:  .ASCII  \MSG$AAAAAAAAAAA\<0>
                                                00 0005B
42 41 41 41 41 41 41 41 24 52 54 50 47 53 4D 0005C P.AAG:  .ASCII  \MSGPTR$AAAAAAAB\<0>
```

```
                                                                00  0006B
42  41  41  41  41  41  41  41  41  41  41  24  47  53          4D  0006C P.AAH:  .ASCII  \MSG$AAAAAAAAAAB\<0>                    ;
                                                                00  0007B                                                       ;
43  41  41  41  41  41  41  41  24  52  54  50  47  53          4D  0007C P.AAI:  .ASCII  \MSGPTR$AAAAAAAC\<0>                  ;
                                                                00  0008B                                                       ;
43  41  41  41  41  41  41  41  41  41  41  24  47  53          4D  0008C P.AAJ:  .ASCII  \MSG$AAAAAAAAAAC\<0>                  ;
                                                                00  0009B                                                       ;


                                                                                .PSECT  $CODE$,NOWRT,2

                                                            03FC 00000 OUTPUT_PSECTS:
                                                                                .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9            ; 0904
                                    59      0000V   CF  9E  00002               MOVAB   PUT_ENTRY, R9                           ;
                                    58      0000G   CF  9E  00007               MOVAB   CLI_FLAGS, R8                           ;
                                    57      0000'   CF  9E  0000C               MOVAB   P.AAB, R7                               ;
                                    5E      FF7C    CE  9E  00011               MOVAB   -132(SP), SP                            ;
                                                        01  DD  00016           PUSHL   #1                                      ; 0926
                                0000V   CF              01  FB  00018           CALLS   #1, START_RECORD                        ;
                                        6E  01C00000    8F  D0  0001D           MOVL    #29360128, ENTRY                        ; 0932
                                                04      AE  D4  00024           CLRL    ENTRY+4                                 ; 0935
                                    08      AE          05  90  00027           MOVB    #5, ENTRY+8                             ; 0936
                        09  AE              67          05  28  0002B           MOVC3   #5, P.AAB, ENTRY+9                      ; 093?
                                                        0E  DD  00030           PUSHL   #14                                     ; 0938
                                                04      AE  9F  00032           PUSHAB  ENTRY                                   ;
                                                69          02  FB  00035       CALLS   #2, PUT_ENTRY                           ;
                                                            68  95  00038       TSTB    CLI_FLAGS                               ; 0940
                                                            07  19  0003A       BLSS    1$                                      ;
                            03                  68          02  E0  0003C       BBS     #2, CLI_FLAGS, 1$                       ; 0941
                                                        00A3    31  00040       BRW     10$                                     ;
                                    02      AE      04  AC  B0  00043 1$:       MOVW    PSECT_FLAGS, ENTRY+2                    ; 0948
                                            50  0000' CF  D0  00048           MOVL    SECTION, R0                             ; 0949
                                    04      AE      04  A0  D0  0004D           MOVL    4(R0), ENTRY+4                          ;
                                0E          68          02  E1  00052       BBC     #2, CLI_FLAGS, 2$                       ; 0951
                                    08      AE          0E  90  00056           MOVB    #14, ENTRY+8                            ; 0954
                        09  AE              08  A7      0E  28  0005A           MOVC3   #14, P.AAC, ENTRY+9                     ; 0955
                                                        17  DD  00060           PUSHL   #23                                     ; 0956
                                                        0C  11  00062           BRB     3$                                      ;
                                    08      AE          0B  90  00064 2$:       MOVB    #11, ENTRY+8                            ; 0960
                        09  AE              18  A7      0B  28  00068           MOVC3   #11, P.AAD, ENTRY+9                     ; 0961
                                                        14  DD  0006E           PUSHL   #20                                     ; 0962
                                                04      AE  9F  00070 3$:       PUSHAB  ENTRY                                   ;
                                                69          02  FB  00073       CALLS   #2, PUT_ENTRY                           ;
                                    01      AE          02  90  00076           MOVB    #2, ENTRY+1                             ; 0965
                                56  04      AC          04  C9  0007A           BISL3   #4, PSECT_FLAGS, R6                     ; 0966
                                    02      AE          56  B0  0007F           MOVW    R6, ENTRY+2                             ;
                                    04      AE          10  D0  00083           MOVL    #16, ENTRY+4                            ; 0967
                                    08      AE          0F  90  00087           MOVB    #15, ENTRY+8                            ; 0968
                                08          68          02  E1  0008B           BBC     #2, CLI_FLAGS, 4$                       ; 0970
                        09  AE      24      A7          0F  28  0008F           MOVC3   #15, P.AAE, ENTRY+9                     ; 0972
                                                        06  11  00095           BRB     5$                                      ;
                        09  AE      34      A7          0F  28  00097 4$:       MOVC3   #15, P.AAF, ENTRY+9                     ; 0974
                                                        18  DD  0009D 5$:       PUSHL   #24                                     ; 0976
                                                04      AE  9F  0009F           PUSHAB  ENTRY                                   ;
                                                69          02  FB  000A2       CALLS   #2, PUT_ENTRY                           ;
                                    02      AE      04  AC  B0  000A5           MOVW    PSECT_FLAGS, ENTRY+2                    ; 0978
```

```
                  04   AE          04 D0 000AA         MOVL     #4, ENTRY+4                    ; 0979
            08         68          02 E1 000AE         BBC      #2, CLI_FLAGS, 6$              ; 0981
      09    AE    44   A7          0F 28 000B2         MOVC3    #15, P.AAG, ENTRY+9            ; 0983
                                   06 11 000B8         BRB      7$                             ;
      09    AE    54   A7          0F 28 000BA 6$:     MOVC3    #15, P.AAH, ENTRY+9            ; 0985
                                   18 DD 000C0 7$:     PUSHL    #24                            ; 0987
                            04     AE 9F 000C2         PUSHAB   ENTRY                          ;
                       69          02 FB 000C5         CALLS    #2, PUT_ENTRY                  ;
                  02   AE          56 B0 000C8         MOVW     R6, ENTRY+2                    ; 0989
            08         68          02 E1 000CC         BBC      #2, CLI_FLAGS, 8$              ; 0991
      09    AE    64   A7          0F 28 000D0         MOVC3    #15, P.AAI, ENTRY+9            ; 0993
                                   06 11 000D6         BRB      9$                             ;
      09    AE    74   A7          0F 28 000D8 8$:     MOVC3    #15, P.AAJ, ENTRY+9            ; 0995
                                   18 DD 000DE 9$:     PUSHL    #24                            ; 0997
                            04     AE 9F 000E0         PUSHAB   ENTRY                          ;
                       69          02 FB 000E3         CALLS    #2, PUT_ENTRY                  ;
               0000V   CF          00 FB 000E6 10$:    CALLS    #0, END_RECORD                 ; 0999
                       50          01 D0 000EB         MOVL     #1, R0                         ; 1001
                                      04 000EE         RET                                     ; 1003

; Routine Size:  239 bytes,    Routine Base:  $CODE$ + 03F6
```

```
;  776          1004  1 ROUTINE output_gsd =
;  777          1005  1
;  778          1006  1 !---
;  779          1007  1 !
;  780          1008  1 !       This routine outputs all the global symbol
;  781          1009  1 !       definitions to symbolically refer to the
;  782          1010  1 !       messages.
;  783          1011  1 !
;  784          1012  1 ! Inputs:
;  785          1013  1 !
;  786          1014  1 !       None
;  787          1015  1 !
;  788          1016  1 ! Outputs:
;  789          1017  1 !
;  790          1018  1 !       The records are output.
;  791          1019  1 !---
;  792          1020  1
;  793          1021  2 BEGIN
;  794          1022  2
;  795          1023  2 LOCAL
;  796          1024  2     entry:      BBLOCK[129],            ! Entry buffer
;  797          1025  2     ptr:        REF BBLOCK;             ! Address of current symbol entry
;  798          1026  2
;  799          1027  2 start_record(obj$c_gsd);               ! Signal start of GSD record
;  800          1028  2
;  801          1029  2 entry [sdf$b_gsdtyp] = gsd$c_sym;       ! Global symbol definition
;  802          1030  2 entry [sdf$b_datyp] = 0;               ! Data type undefined
;  803          1031  2 entry [sdf$w_flags] = gsy$m_def;       ! Definition rather than reference
;  804          1032  2 entry [sdf$b_psindx] = 0;              ! Owning psect number 0 ($ABS$)
;  805          1033  2
;  806          1034  2 ptr = .symbol_header;                 ! Point at first symbol entry
;  807          1035  2
;  808          1036  2 WHILE .ptr NEQ 0                      ! Until end of list
;  809          1037  2 DO
;  810          1038  3     BEGIN
;  811          1039  3     entry [sdf$l_value] = .ptr [sym$l_value];
;  812          1040  3     entry [sdf$b_namlng] = .ptr [sym$b_symlen];
;  813          1041  3     CH$MOVE(.ptr [sym$b_symlen], ptr [sym$t_symbol], entry [sdf$t_name]);
;  814          1042  3     put_entry(entry, $BYTEOFFSET(sdf$t_name)+.entry [sdf$b_namlng]);
;  815          1043  3     ptr = .ptr [sym$l_link];            ! Skip to next in chain
;  816          1044  2     END;
;  817          1045  2
;  818          1046  2 end_record();                          ! Flush rest of record buffer
;  819          1047  2
;  820          1048  2 RETURN true;
;  821          1049  2
;  822          1050  1 END;


                                007C 00000 OUTPUT_GSD:
                                           .WORD   Save R2,R3,R4,R5,R6                     ; 1004
                     5E     FF7C    CE  9E 00002        MOVAB   -132(SP), SP                    ;
                                    01  DD 00007        PUSHL   #1                              ; 1027
                0000V  CF           01  FB 00009        CALLS   #1, START_RECORD                ;
```

```
                        6E 00020001    8F   D0 0000E          MOVL    #131073, ENTRY                          ; 1029
                                 04    AE   94 00015          CLRB    ENTRY+4                                 ; 1032
                        56       0000G CF   D0 00018          MOVL    SYMBOL_HEADER, PTR                      ; 1034
                                       28   13 0001D 1$:      BEQL    2$                                      ; 1036
                     05 AE          04   A6   D0 0001F          MOVL    4(PTR), ENTRY+5                         ; 1039
                     09 AE          08   A6   90 00024          MOVB    8(PTR), ENTRY+9                         ; 1040
                        50          08   A6   9A 00029          MOVZBL  8(PTR), R0                              ; 1041
                0A AE      09       A6        50   28 0002D     MOVC3   R0, 9(PTR), ENTRY+10                    ;
                        7E          09   AE   9A 00033          MOVZBL  ENTRY+9, -(SP)                          ; 1042
                        6E             0A   C0 00037          ADDL2   #10, (SP)                               ;
                                    04 AE   9F 0003A          PUSHAB  ENTRY                                   ;
                   0000V CF             02   FB 0003D          CALLS   #2, PUT_ENTRY                           ;
                        56             66   D0 00042          MOVL    (PTR), PTR                              ; 1043
                                       D6   11 00045          BRB     1$                                      ; 1036
                   0000V CF             00   FB 00047 2$:      CALLS   #0, END_RECORD                          ; 1046
                        50             01   D0 0004C          MOVL    #1, R0                                  ; 1048
                                          04 0004F          RET                                             ; 1050

; Routine Size:  80 bytes,    Routine Base:  $CODE$ + 04E5
```

```
;  824     1051  1 ROUTINE output_section =
;  825     1052  1
;  826     1053  1 !---
;  827     1054  1 !
;  828     1055  1 !       This routine outputs the TIR records to the object
;  829     1056  1 !       module file to describe the message section.
;  830     1057  1 !
;  831     1058  1 ! Inputs:
;  832     1059  1 !
;  833     1060  1 !       None
;  834     1061  1 !
;  835     1062  1 ! Outputs:
;  836     1063  1 !
;  837     1064  1 !       The records are written.
;  838     1065  1 !---
;  839     1066  1
;  840     1067  2 BEGIN
;  841     1068  2
;  842     1069  3 LOCAL
;  843     1070  2     entry:      VECTOR[129,BYTE],       ! TIR command buffer
;  844     1071  2     position,                           ! Position within section
;  845     1072  2     bytes_left;                         ! Bytes left of section to output
;  846     1073  2
;  847     1074  2 start_record(obj$c_tir);                ! Signal start of TIR record
;  848     1075  2
;  849     1076  2 !
;  850     1077  2 !       Output the message section vector dispatcher (psect 1)
;  851     1078  2 !       The name of the psect is set so that it will be loaded
;  852     1079  2 !       in front of the message section list.
;  853     1080  2 !
;  854     1081  2
;  855     1082  2 set_psect(2,0);                         ! Set to psect 2
;  856     1083  2
;  857     1084  2 entry [0] = -16;                        ! Store immedate 16 bytes
;  858     1085  3 BEGIN
;  859     1086  3 LOCAL
;  860     1087  3     vector: REF BBLOCK;                 ! Address buffer as longwords
;  861     1088  3
;  862     1089  3 vector = entry [1];                     ! Address the start of the vector
;  863     1090  3 vector [plv$l_type] = plv$c_typ_msg;    ! Type of vector
;  864     1091  3 vector [plv$l_version] = 0;             ! (reserved)
;  865     1092  3 vector [plv$l_msgdsp] = 6;              ! Self-rel offset to dispatcher code
;  866     1093  3 vector [12,0,32,0] = %x'65160101';      ! NOP, NOP, JSB (R5) instructions
;  867     1094  2 END;
;  868     1095  2 put_entry(entry,17);                    ! Output TIR command
;  869     1096  2
;  870     1097  2 !
;  871     1098  2 !       Output the self-relative offset to the mssage section (psect 3).
;  872     1099  2 !       It will be contributed to the section list psect at
;  873     1100  2 !       the end - resulting in a list of section offsets.
;  874     1101  2 !       Each offset to the sections is relative to the start
;  875     1102  2 !       of the longword itself.
;  876     1103  2 !
;  877     1104  2
;  878     1105  2 set_psect(3,0);                         ! Set to psect 3
;  879     1106  2
;  880     1107  2 entry [0] = tir$c_sta_sb;               ! Stack signed byte
```

```
;  881          1108  2 entry [1] = -1;                         ! Sign extended -1
;  882          1109  2 entry [2] = tir$c_sta_pb;               ! Stack psect base plus offset
;  883          1110  2 entry [3] = 1;                          ! psect number of message section
;  884          1111  2 entry [4] = 0;                          ! start of psect
;  885          1112  2 entry [5] = tir$c_sto_pirr;             ! Store self-relative offset to psect 1
;  886          1113  2
;  887          1114  2 put_entry(entry,6);                     ! Output TIR command
;  888          1115  2
;  889          1116  2 !
;  890          1117  2 !       Output a zero list terminator (psect 4).
;  891          1118  2 !       This psect is an overlayed psect of length 4 which
;  892          1119  2 !       contains a zero.  The name of the psect is set
;  893          1120  2 !       so that it will be loaded following the message
;  894          1121  2 !       section list as a list terminator.
;  895          1122  2 !
;  896          1123  2
;  897          1124  2 set_psect(4,0);                         ! Set to psect 4
;  898          1125  2
;  899          1126  2 entry [0] = tir$c_sta_sb;               ! Stack signed byte
;  900          1127  2 entry [1] = 0;                          ! value = 0 (byte to store)
;  901          1128  2 entry [2] = tir$c_sta_sb;               ! Stack signed byte
;  902          1129  2 entry [3] = 4;                          ! value = 4 (repeat count)
;  903          1130  2 entry [4] = tir$c_sto_rb;               ! Store repeated byte (4 zero bytes)
;  904          1131  2
;  905          1132  2 put_entry(entry,5);                     ! Output TIR commands
;  906          1133  2
;  907          1134  2 !
;  908          1135  2 !       Output the message section text
;  909          1136  2 !
;  910          1137  2
;  911          1138  2 set_psect(1,0);                         ! Set to psect 1
;  912          1139  2
;  913          1140  2 position = .section;                    ! Start at beginning of section
;  914          1141  2 bytes_left = .section [msc$l_size];     ! Bytes left to output
;  915          1142  2
;  916          1143  2 WHILE .bytes_left GTR 0                 ! While stuff left to output,
;  917          1144  2 DO
;  918          1145  3     BEGIN
;  919          1146  3     LOCAL bytes;
;  920          1147  3
;  921          1148  3     bytes = MINU(.bytes_left,128);      ! Max 128 bytes per record
;  922          1149  3     entry [0] = -.bytes;                ! Number of bytes to output
;  923          1150  3     CH$MOVE(.bytes,.position,entry[1]); ! Move the bytes
;  924          1151  3     put_entry (entry, .bytes+1);        ! Output the TIR command
;  925          1152  3     position = .position + .bytes;      ! Skip past stuff output
;  926          1153  3     bytes_left = .bytes_left - .bytes;  ! Decrement amount remaining
;  927          1154  2     END;
;  928          1155  2
;  929          1156  2 end_record();                   ! Flush remaining record buffer
;  930          1157  2
;  931          1158  2 RETURN true;
;  932          1159  2
;  933          1160  1 END;
```

```
                                07FC 00000 OUTPUT_SECTION:
                                           .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10                ; 1051
                5A      0000V   CF  9E 00002       MOVAB   PUT_ENTRY, R10
                59      0000V   CF  9E 00007       MOVAB   SET_PSECT, R9
                5E      FF7C    CE  9E 0000C       MOVAB   -132(SP), SP
                                02  DD 00011       PUSHL   #2                                      ; 1074
          0000V CF              01  FB 00013       CALLS   #1, START_RECORD
                7E              02  7D 00018       MOVQ    #2, -(SP)                               ; 1082
                69              02  FB 0001B       CALLS   #2, SET_PSECT
                6E              10  8E 0001E       MNEGB   #16, ENTRY                              ; 1084
                50        01    AE  9E 00021       MOVAB   ENTRY+1, VECTOR                         ; 1089
                60              02  7D 00025       MOVQ    #2, (VECTOR)                            ; 1090
             08 A0              06  D0 00028       MOVL    #6, 8(VECTOR)                           ; 1092
             0C A0  65160101    8F  D0 0002C       MOVL    #1695940865, 12(VECTOR)                 ; 1093
                                11  DD 00034       PUSHL   #17                                     ; 1095
                          04    AE  9F 00036       PUSHAB  ENTRY
                6A              02  FB 00039       CALLS   #2, PUT_ENTRY
                7E              03  7D 0003C       MOVQ    #3, -(SP)                               ; 1105
                69              02  FB 0003F       CALLS   #2, SET_PSECT
                6E  0104FF01    8F  D0 00042       MOVL    #17104641, ENTRY                        ; 1107
             04 AE        2A00  8F  B0 00049       MOVW    #10752, ENTRY+4                         ; 1111
                                06  DD 0004F       PUSHL   #6                                      ; 1114
                          04    AE  9F 00051       PUSHAB  ENTRY
                6A              02  FB 00054       CALLS   #2, PUT_ENTRY
                7E              04  7D 00057       MOVQ    #4, -(SP)                               ; 1124
                69              02  FB 0005A       CALLS   #2, SET_PSECT
                6E  04010001    8F  D0 0005D       MOVL    #67174401, ENTRY                        ; 1126
             04 AE              27  90 00064       MOVB    #39, ENTRY+4                            ; 1130
                                05  DD 00068       PUSHL   #5                                      ; 1132
                          04    AE  9F 0006A       PUSHAB  ENTRY
                6A              02  FB 0006D       CALLS   #2, PUT_ENTRY
                7E              01  7D 00070       MOVQ    #1, -(SP)                               ; 1138
                69              02  FB 00073       CALLS   #2, SET_PSECT
                50      0000'   CF  D0 00076       MOVL    SECTION, R0                             ; 1140
                58              50  D0 0007B       MOVL    R0, POSITION
                56        04    A0  D0 0007E       MOVL    4(R0), BYTES_LEFT                       ; 1141
                                2C  15 00082 1$:    BLEQ    3$                                      ; 1143
                50              56  D0 00084       MOVL    BYTES_LEFT, R0                          ; 1148
                    00000080    8F  50  D1 00087    CMPL    R0, #128
                                04  1B 0008E       BLEQU   2$
                50        80    8F  9A 00090       MOVZBL  #128, R0
                57              50  D0 00094 2$:    MOVL    R0, BYTES
                6E              57  8E 00097       MNEGB   BYTES, ENTRY                            ; 1149
          01    AE              68  57  28 0009A    MOVC3   BYTES, (POSITION), ENTRY+1              ; 1150
                          01    A7  9F 0009F       PUSHAB  1(BYTES)                                ; 1151
                          04    AE  9F 000A2       PUSHAB  ENTRY
                6A              02  FB 000A5       CALLS   #2, PUT_ENTRY
                58              57  C0 000A8       ADDL2   BYTES, POSITION                         ; 1152
                56              57  C2 000AB       SUBL2   BYTES, BYTES_LEFT                       ; 1153
                                D2  11 000AE       BRB     1$                                      ; 1143
                    0000V   CF      00  FB 000B0 3$:    CALLS   #0, END_RECORD                  ; 1156
                50              01  D0 000B5       MOVL    #1, R0                                  ; 1158
                                    04 000B8       RET                                             ; 1160

; Routine Size:  185 bytes,    Routine Base:  $CODE$ + 0535
```

```
;  935          1161  1 ROUTINE set_psect(psect_number, offset) =
;  936          1162  1
;  937          1163  1 !---
;  938          1164  1 !
;  939          1165  1 !       Output an object module command to set the
;  940          1166  1 !       current psect to the specified value to
;  941          1167  1 !       signal the start of contributions to that
;  942          1168  1 !       psect.
;  943          1169  1 !
;  944          1170  1 ! Inputs:
;  945          1171  1 !
;  946          1172  1 !       psect_number = Psect number to set to
;  947          1173  1 !       offset = Offset into psect
;  948          1174  1 !
;  949          1175  1 ! Outputs:
;  950          1176  1 !
;  951          1177  1 !       None
;  952          1178  1 !---
;  953          1179  1
;  954          1180  2 BEGIN
;  955          1181  2 LOCAL
;  956          1182  2     entry:      VECTOR[4,BYTE];         ! Allocate buffer
;  957          1183  2
;  958          1184  2 entry [0] = tir$c_sta_pb;              ! Stack psect base
;  959          1185  2 entry [1] = .psect_number;             ! psect number
;  960          1186  2 entry [2] = .offset;                   ! offset into psect
;  961          1187  2 entry [3] = tir$c_ctl_setrb;           ! Set relocation base
;  962          1188  2 put_entry(entry,4);                   ! Output TIR command
;  963          1189  2
;  964          1190  2 RETURN true;
;  965          1191  2
;  966          1192  1 END;


                                        0000 00000 SET_PSECT:
                                                           .WORD   Save nothing                        ;  1161
                                5E              04  C2 00002       SUBL2   #4, SP                      ;
                                6E              04  90 00005       MOVB    #4, ENTRY                   ;  1184
                          01    AE        04    AC  90 00008       MOVB    PSECT_NUMBER, ENTRY+1       ;  1185
                          02    AE        08    AC  90 0000D       MOVB    OFFSET, ENTRY+2             ;  1186
                          03    AE        50    8F  90 00012       MOVB    #80, ENTRY+3                ;  1187
                                                  04  DD 00017       PUSHL   #4                          ;  1188
                                          04      AE  9F 00019       PUSHAB  ENTRY                       ;
                                0000V     CF            02  FB 0001C       CALLS   #2, PUT_ENTRY       ;
                                          50            01  D0 00021       MOVL    #1, R0              ;  1190
                                                        04  00024       RET                         ;  1192

; Routine Size:  37 bytes,    Routine Base:  $CODE$ + 05EE
```

```
;  968          1193  1 ROUTINE put_record (address, length) =
;  969          1194  1
;  970          1195  1 !---
;  971          1196  1 !
;  972          1197  1 !       Output a record to the object module file.
;  973          1198  1 !
;  974          1199  1 ! Inputs:
;  975          1200  1 !
;  976          1201  1 !       address = Address of record
;  977          1202  1 !       length = Length of record
;  978          1203  1 !
;  979          1204  1 ! Outputs:
;  980          1205  1 !
;  981          1206  1 !       None
;  982          1207  1 !---
;  983          1208  1
;  984          1209  2 BEGIN
;  985          1210  2
;  986          1211  2 LOCAL
;  987          1212  2     status;
;  988          1213  2
;  989          1214  2 object_rab [rab$w_rsz] = .length;
;  990          1215  2 object_rab [rab$l_rbf] = .address;
;  991          1216  2
;  992          1217  2 status = $PUT(RAB = object_rab);          ! Put the record to the file
;  993          1218  2 IF NOT .status                            ! If error detected,
;  994          1219  2 THEN
;  995          1220  3     BEGIN
;  996          1221  3     rms_error(emsg(writeerr),object_fab,object_rab);    ! signal error
;  997          1222  3     RETURN .status;                       ! and return with status
;  998          1223  2     END;
;  999          1224  2
; 1000          1225  2 RETURN true;
; 1001          1226  2
; 1002          1227  1 END;


                                                        .EXTRN  SYS$PUT

                                        000C 00000 PUT_RECORD:
                                                        .WORD   Save R2,R3                              ; 1193
                        53      0000G   CF  9E 00002            MOVAB   OBJECT_RAB, R3                  ;
                    22  A3        08   AC  B0 00007            MOVW    LENGTH, OBJECT_RAB+34           ; 1214
                    28  A3        04   AC  D0 0000C            MOVL    ADDRESS, OBJECT_RAB+40          ; 1215
                                       53  DD 00011            PUSHL   R3                              ; 1217
                000000000G  00             01  FB 00013        CALLS   #1, SYS$PUT                     ;
                        52                 50  D0 0001A        MOVL    R0, STATUS                      ;
                        15                 52  E8 0001D        BLBS    STATUS, 1$                      ; 1218
                                           53  DD 00020        PUSHL   R3                              ; 1221
                                0000G      CF  9F 00022        PUSHAB  OBJECT_FAB                      ;
                             009710D2      8F  DD 00026        PUSHL   #9900242                        ;
                    0000G   CF               03  FB 0002C    CALLS   #3, RMS_ERROR                   ;
                        50                 52  D0 00031        MOVL    STATUS, R0                      ; 1222
                                           04  00034          RET                                     ;
                        50                 01  D0 00035 1$:   MOVL    #1, R0                          ; 1225
                                           04  00038          RET                                     ; 1227
```

; Routine Size: 57 bytes,   Routine Base: $CODE$ + 0613

```
;  1004          1228  1 ROUTINE start_record (type) =
;  1005          1229  1
;  1006          1230  1 !---
;  1007          1231  1 !
;  1008          1232  1 !       This routine is called when a variable size record
;  1009          1233  1 !       (like GSD or TIR) is about to be output.
;  1010          1234  1 !
;  1011          1235  1 ! Inputs:
;  1012          1236  1 !
;  1013          1237  1 !       type = Type of variable record (GSD or TIR)
;  1014          1238  1 !
;  1015          1239  1 ! Outputs:
;  1016          1240  1 !
;  1017          1241  1 !       rec_filled = Address of next available byte in record
;  1018          1242  1 !---
;  1019          1243  1
;  1020          1244  2 BEGIN
;  1021          1245  2
;  1022          1246  2 record_buffer [obj$b_rectyp] = .type;          ! Set record type
;  1023          1247  2 rec_filled = 1;                               ! Set offset to beginning
;  1024          1248  2
;  1025          1249  2 RETURN true;
;  1026          1250  2
;  1027          1251  1 END;


                                0000 00000 START_RECORD:
                                                .WORD   Save nothing                    ;  1228
                0000'  CF       04      AC  90 00002    MOVB    TYPE, RECORD_BUFFER     ;  1246
                0000'  CF               01  D0 00008    MOVL    #1, REC_FILLED          ;  1247
                       50               01  D0 0000D    MOVL    #1, R0                  ;  1249
                                        04  00010       RET                             ;  1251

; Routine Size:  17 bytes,    Routine Base:  $CODE$ + 064C
```

```
;  1029          1252  1 ROUTINE put_entry (address, length) =
;  1030          1253  1
;  1031          1254  1 !---
;  1032          1255  1 !
;  1033          1256  1 !       This routine outputs a GSD/TIR entry into the current record
;  1034          1257  1 !       buffer.  The record buffer will be output automatically
;  1035          1258  1 !       when full.
;  1036          1259  1 !
;  1037          1260  1 ! Inputs:
;  1038          1261  1 !
;  1039          1262  1 !       address = Address of GSD entry buffer
;  1040          1263  1 !       length = Length of entry
;  1041          1264  1 !       rec_filled = Current offset into record buffer
;  1042          1265  1 !
;  1043          1266  1 ! Outputs:
;  1044          1267  1 !
;  1045          1268  1 !       rec_filled updated.
;  1046          1269  1 !---
;  1047          1270  1
;  1048          1271  2 BEGIN
;  1049          1272  2
;  1050          1273  2 IF (.rec_filled+.length) GTR obj$c_maxrecsiz    ! If record too large,
;  1051          1274  2 THEN
;  1052          1275  3     BEGIN
;  1053          1276  3     put_record(record_buffer,.rec_filled);       ! Output the record
;  1054          1277  3     rec_filled = 1;                              ! Start at beginning again
;  1055          1278  2     END;
;  1056          1279  2
;  1057          1280  2 CH$MOVE(.length, .address, record_buffer+.rec_filled); ! Move into record buffer
;  1058          1281  2
;  1059          1282  2 rec_filled = .rec_filled + .length;             ! Update buffer position
;  1060          1283  2
;  1061          1284  2 RETURN true;
;  1062          1285  2
;  1063          1286  1 END;


                                        007C 00000 PUT_ENTRY:
                                                           .WORD   Save R2,R3,R4,R5,R6                          ; 1252
                                 56     0000'  CF  9E 00002       MOVAB   REC_FILLED, R6
                           50    66       08   AC  C1 00007       ADDL3   LENGTH, REC_FILLED, R0               ; 1273
                     00000800    8F            50  D1 0000C       CMPL    R0, #2048
                                               0D  15 00013       BLEQ    1$
                                               66  DD 00015       PUSHL   REC_FILLED                           ; 1276
                                       F800    C6  9F 00017       PUSHAB  RECORD_BUFFER
                                 97    AF            02  FB 0001B       CALLS   #2, PUT_RECORD
                                       66            01  D0 0001F       MOVL    #1, REC_FILLED               ; 1277
                                       50      F800  C6  9E 00022 1$:   MOVAB   RECORD_BUFFER, R0            ; 1280
                           00 B640     04      BC    08  AC  28 00027       MOVC3   LENGTH, @ADDRESS, @REC_FILLED[R0]
                                       66      08    AC  C0 0002F       ADDL2   LENGTH, REC_FILLED           ; 1282
                                       50            01  D0 00033       MOVL    #1, R0                       ; 1284
                                                     04  00036       RET                                  ; 1286

; Routine Size:  55 bytes,    Routine Base:  $CODE$ + 065D
```

```
;  1065          1287  1 ROUTINE end_record =
;  1066          1288  1
;  1067          1289  1 !---
;  1068          1290  1 !
;  1069          1291  1 !       This routine is called to signal that all entries have
;  1070          1292  1 !       been output for now and to flush the current record buffer.
;  1071          1293  1 !
;  1072          1294  1 ! Inputs:
;  1073          1295  1 !
;  1074          1296  1 !       rec_filled = Amount left in buffer
;  1075          1297  1 !
;  1076          1298  1 ! Outputs:
;  1077          1299  1 !
;  1078          1300  1 !       None
;  1079          1301  1 !---
;  1080          1302  1
;  1081          1303  2 BEGIN
;  1082          1304  2
;  1083          1305  2 IF .rec_filled GTR 1                          ! If anything in buffer
;  1084          1306  2 THEN
;  1085          1307  2     put_record(record_buffer, .rec_filled);    ! then output record
;  1086          1308  2
;  1087          1309  2 RETURN true;
;  1088          1310  2
;  1089          1311  1 END;


                                0000 00000 END_RECORD:
                                           .WORD   Save nothing                    ;  1287
                     01   0000'  CF  D1 00002       CMPL    REC_FILLED, #1      ;  1305
                                  0D  15 00007       BLEQ    1$                  ;
                          0000'  CF  DD 00009       PUSHL   REC_FILLED          ;  1307
                          0000'  CF  9F 0000D       PUSHAB  RECORD_BUFFER       ;
                FF69      CF         02  FB 00011   CALLS   #2, PUT_RECORD      ;
                     50             01  D0 00016 1$: MOVL    #1, R0              ;  1309
                                    04  00019       RET                         ;  1311

; Routine Size:  26 bytes,    Routine Base:  $CODE$ + 0694
```

```
; 1091          1312  1 END
; 1092          1313  0 ELUDOM


                                                .EXTRN  LIB$SIGNAL

;                               PSECT SUMMARY

;       Name                    Bytes                   Attributes

;  $OWN$                        2056  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $CODE$                       1710  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $PLIT$                        156  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)


;                           Library Statistics

;                                    -------- Symbols --------      Pages       Processing
;       File                         Total    Loaded   Percent      Mapped      Time

;  _$255$DUA28:[SYSLIB]STARLET.L32;1   9776        95         0       581          00:01.0




;                                   COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:OBJECT/OBJ=OBJ$:OBJECT MSRC$:OBJECT/UPDATE=(ENH$:OBJECT)

; Size:          1710 code + 2212 data bytes
; Run Time:         00:32.1
; Elapsed Time:     01:30.6
; Lines/CPU Min:     2452
; Lexemes/CPU-Min: 20445
; Memory Used:  198 pages
; Compilation Complete
```

MDLGEN
LIS

SDLGEN
LIS

PARSE
LIS

MESSAGES
LIS

OBJECT
LIS