H 5

```
MMM         MMM     SSSSSSSSSSSS     GGGGGGGGGGGG  FFFFFFFFFFFFFFF     IIIIIIIII     LLL
MMM         MMM     SSSSSSSSSSSS     GGGGGGGGGGGG  FFFFFFFFFFFFFFF     IIIIIIIII     LLL
MMM         MMM     SSSSSSSSSSSS     GGGGGGGGGGGG  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              FFFFFFFFFFFF           III        LLL
MMM         MMM     SSSSSSSSS     GGG              FFFFFFFFFFFF           III        LLL
MMM         MMM     SSSSSSSSS     GGG              FFFFFFFFFFFF           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  SSSSSSSSSSSS        GGGGGGGGG     FFF                 IIIIIIIII     LLLLLLLLLLLLLLL
MMM         MMM  SSSSSSSSSSSS        GGGGGGGGG     FFF                 IIIIIIIII     LLLLLLLLLLLLLLL
MMM         MMM  SSSSSSSSSSSS        GGGGGGGGG     FFF                 IIIIIIIII     LLLLLLLLLLLLLLL
```

FILEID**OBJECT

N 4

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

OBJE
V04-

```
MODULE object (IDENT='V04-000') =

BEGIN

!
!****************************************************************************
!*                                                                          *
!*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
!*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
!*  ALL RIGHTS RESERVED.                                                    *
!*                                                                          *
!*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
!*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
!*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
!*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
!*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
!*  TRANSFERRED.                                                            *
!*                                                                          *
!*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
!*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
!*  CORPORATION.                                                            *
!*                                                                          *
!*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
!*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
!*                                                                          *
!*                                                                          *
!****************************************************************************

!++
! FACILITY:  The MESSAGE compiler
!
! ABSTRACT:
!
!       This compiler translates a message definition language
!       and produces object modules suitable for the $GETMSG
!       system service.
!
! ENVIRONMENT:
!
!       VAX/VMS operating system. unprivileged user mode,
!
! AUTHOR:  Tim Halvorsen, Nov 1979
!
! Modified by:
!
!       V03-004 GJA0096         Greg Awdziewicz         10-Aug-1984
!               - Change version # written into object module to v04-00.
!
!       V03-003 GJA0079         Greg Awdziewicz         6-Apr-1984
!               Initialize flags byte in Create_indirect.
!
!       V03-002 GJA0050         Greg Awdziewicz         12-Sep-1983
!               Reference the filename descriptor as a byte block structure
!               instead of as a longword vector.
!
!       001     JWT0024         Jim Teague      17-Mar-1982
!               Corrects the bug which prevented pointer psects from being
```

```
1 !                     output when /notext is explicitly specified.
1 !
1 !--
```

```
1 !
1 ! Include files
1 !
1
1 LIBRARY 'SYS$LIBRARY:STARLET';           ! VMS common definitions
1
1 REQUIRE 'SRC$:MSG.REQ';                  ! Message common definitions
```

```
  1 !
  1 ! Table of contents
  1 !
  1
  1 FORWARD ROUTINE
  1     output_object,               ! Output object module
  1     create_section,              ! Create message section
  1     create_indirect,             ! Create indirect message section
  1     output_mhd,                  ! Output module header record
  1     output_psects,               ! Output psect definitions
  1     output_gsd,                  ! Output global symbol records
  1     output_section,              ! Output message section
  1     output_eom,                  ! Output end of module record
  1     set_psect,                   ! Set to selected psect
  1     put_record,                  ! Put object record to file
  1     start_record,                ! Start of GSD/TIR record
  1     put_entry,                   ! Output GSD/TIR entry
  1     end_record;                  ! End of GSD/TIR entries
  1
  1 !
  1 ! Macros
  1 !
  1
  1 MACRO
M 1     perform(command) =
M 1         BEGIN
M 1         LOCAL status;
M 1         status = command;
M 1         IF NOT .status
M 1         THEN
M 1             BEGIN
M 1             SIGNAL(.status);
M 1             RETURN .status;
M 1             END;
  1         END%;
  1
  1 !
  1 ! Literals
  1 !
  1
  1 LITERAL
  1     psect_flags = gps$m_pic OR gps$m_rel OR gps$m_rd OR gps$m_vec;
  1
  1 !
  1 ! Module storage
  1 !
  1
  1 OWN
  1     section:          REF BBLOCK,     ! Address of message section
  1     record_buffer:    BBLOCK[obj$c_maxrecsiz],      ! record buffer
  1     rec_filled;                       ! Space used in record_buffer so far
  1
  1 !
  1 ! External storage
  1 !
  1
  1 EXTERNAL
```

```
    object_fab:         BBLOCK,             ! Object module FAB
    object_rab:         BBLOCK,             ! Object module RAB
    cli_flags:          BITVECTOR,          ! CLI qualifier flags
    module_name:        VECTOR,             ! Name of object module
    filename_desc:      BLOCK [, BYTE],     ! Descriptor of /FILE value
    facility_header,                        ! Facility list head
    message_header,                         ! Message list head
    symbol_header,                          ! Symbol list head
    num_messages,                           ! Number of messages defined
    msg_space,                              ! Total space used by MSG records
    num_facilities,                         ! Number of facilities defined
    fac_space,                              ! Total space needed for facility table
    version_num:        VECTOR,             ! Descriptor for version/ident
    num_files;                              ! Total files parsed

!
! External routines
!

EXTERNAL ROUTINE
    get_module_name,                        ! Extract module name
    rms_error,                              ! Signal RMS-type error
    allocate;                               ! Allocate dynamic memory
```

```
GLOBAL ROUTINE output_object =

!---
!
!       This routine generates the object module file from the
!       message definition blocks generated by input parse routines.
!
! Inputs:
!
!       object_fab/rab = FAB/RAB for object module file
!       facility_header = Facility list head
!       message_header = Message definition list head
!
! Outputs:
!
!       The object module file
!---

BEGIN

LOCAL
    status;                             ! status code

!
!       Create the object module file
!

status = $CREATE (FAB = object_fab);    ! Create the output file
IF NOT .status                          ! If error detected,
THEN
    BEGIN
    rms_error(emsg(openout),object_fab);        ! then signal the error
    RETURN .status;                     ! and return with status
    END;

status = $CONNECT (RAB = object_rab);   ! Connect to the object file
IF NOT .status                          ! If error detected,
THEN
    BEGIN
    rms_error(emsg(openout),object_fab,object_rab);     ! then signal it
    RETURN .status;                     ! and return with status
    END;

IF .num_files GTR 1                     ! If more than 1 input file,
THEN
    get_module_name(object_fab);        ! Use object file name as module name

!
!       Create the index message structure in dynamic storage.
!

IF NOT .cli_flags [qual_file]           ! If /FILE not specified,
THEN
    perform(create_section())           ! Create the message structure
ELSE
    perform(create_indirect());         ! Else, create indirect section

```

```
2 !
2 !      Output the module header record
2 !
2
2 perform(output_mhd());                         ! Output the module header record
2
2 !
2 !      Define the psects used for the message section.  If
2 !      constructing an indirect message section, make the psects
2 !      writable.
2 !
2
2 IF NOT .cli_flags [qual_file]                  ! If /FILE qualifier not specified,
2 THEN
3     perform(output_psects(psect_flags)) ! Make normal NOWRT
2 ELSE
2     perform(output_psects(psect_flags OR gps$m_wrt));    ! Make indirect WRT
2
2 !
2 !      Output the GSD records for the symbols
2 !
2
2 IF .cli_flags [qual_symbols]                   ! If /SYMBOLS specified,
2 THEN
2     perform(output_gsd());                     ! Output the GSD records
2
2 !
2 !      Output the TIR records which describe the message structure
2 !
2
2 IF .cli_flags [qual_text] OR .cli_flags [qual_file]  ! If /TEXT or /FILE
2 THEN
2     perform(output_section());                 ! Output section TIR records
2
2 !
2 !      Output the end-of-module record
2 !
2
2 perform(output_eom());                         ! Output the EOM record
2
2 !
2 !      Close the object module file
2 !
2
2 status = $CLOSE(FAB = object_fab);             ! Close the file
2 IF NOT .status                                 ! If error,
2 THEN
3     BEGIN
3     rms_error(emsg(closedel),object_fab);  ! signal the error
3     RETURN .status;                        ! and return with status
2     END;
2
2 RETURN true;
2
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
```

```
ROUTINE create_section =

!---
!
!       This routine creates the message section exactly as
!       it will look when linked into the user image.  The
!       resulting structure will be output as TIR records to
!       the object module.
!
! Inputs:
!
!       facility_header = List head for facility definitions
!       message_header = List head for message definitions
!       num_messages = Number of message definitions in the list
!       msg_space = Space used by MSG records alone
!       num_facilities = Number of facility definitions in the list
!       fac_space = Space needed for facility table
!
! Outputs:
!
!       r0 = status (unsignaled)
!       section = Address of final message section
!---

BEGIN

LITERAL
    bucket_size = 512,                      ! Size of each index bucket
    bucket_overhead = midx$c_entries,       ! Overhead in each bucket
    usable_size = bucket_size-bucket_overhead,  ! Useable space in each bucket
    entry_size = 8,                         ! Size of each index entry
    entries_per_blk = usable_size/entry_size,
    max_1level = 3*bucket_size;             ! Maximum size of 1 level index
                                            ! before 2 level index will be created

LOCAL
    prim_index_len,                         ! Length of primary index
    sec_index_len,                          ! Length of secondary index (0 if none)
    sec_index_bkts,                         ! # buckets in secondary index
    section_size,                           ! Total size of message section
    prim_index:         REF VECTOR,         ! Current position in primary index
    bucket:             REF BBLOCK,         ! Address of current index bucket
    bucket_filled,                          ! Bytes allocated in current bucket
    data:               REF BBLOCK,         ! Address of next available data byte
    code:               REF BBLOCK,         ! Address of current CODE block
    fac:                REF BBLOCK,         ! Address of current FAC block
    status;                                 ! Status code

MACRO
    round_div(i,j) =
        ((i/j) + ((i MOD j) NEQ 0))%; ! i/j rounded up if remainder GTR 0

IF .num_messages LEQ 0                      ! If no messages defined,
THEN
    RETURN emsg(nomsgs);                    ! then return with error

!
```

```
!
!       Compute the amount of space needed to hold the index
!       and text portions of the message section.
!

prim_index_len = .num_messages * entry_size + bucket_overhead;

IF .prim_index_len GTR max_1level         ! If index too large,
THEN
    BEGIN                                 ! Setup a 2 level index
    sec_index_bkts = round_div(.num_messages*entry_size,usable_size);

    sec_index_len = .num_messages * entry_size  ! Size of secondary index
        + .sec_index_bkts * bucket_overhead;    ! plus bucket overhead

    prim_index_len = .sec_index_bkts * entry_size ! Set size of primary index
                    + bucket_overhead        ! plus bucket overhead
    END
ELSE
    sec_index_len = 0;                    ! Else no secondary index (useless)

!
!       Compute the amount of space needed for the entire message section
!       including the facility table.
!

section_size = msc$c_length + .prim_index_len + .sec_index_len
             + .fac_space + .msg_space;

!
!       Allocate enough space for the entire thing
!

status = allocate(.section_size,section);! Allocate the space, signal error
IF NOT .status                           ! If error detected,
THEN
    RETURN .status;                      ! then return with status

!
!       Construct the index and data portions of the section
!

CH$FILL(0,msc$c_length,.section);        ! Zero the section header
section [msc$b_type] = msc$c_msg;        ! Set type of section
section [msc$w_sanity] = msc$c_sanity;   ! Set sanity check word
section [msc$l_size] = .section_size;    ! Set size of entire section
section [msc$l_index_off] = msc$c_length; ! Offset to primary index
section [msc$l_text_off] = msc$c_length+.prim_index_len+.sec_index_len;
section [msc$l_fac_off] = .section [msc$l_text_off] + .msg_space;

!
!       Initialize the primary index
!

BEGIN
BIND
    prim_bucket = .section+msc$c_length: BBLOCK;        ! Address primary index
prim_bucket [midx$w_size] = .prim_index_len;    ! Set length of index
```

```
3 prim_bucket [midx$b_sanity] = midx$c_sanity;     ! Set sanity check byte
3 prim_index = prim_bucket + midx$c_entries;                ! Position at entry 1
2 END;
2
2 !
2 !       Setup the secondary index pointers
2 !
2
2 bucket = .section+msc$c_length + .prim_index_len;        ! Point at sec. index
2 bucket_filled = 0;                        ! Nothing filled yet
2
2 code = .message_header;                   ! Point at first message definition
2 data = .section + .section [msc$l_text_off];    ! Point at data area
2
2 !
2 !       Loop thru each message definition storing the
2 !       message definition record and the index entry.
2 !
2
2 INCR i FROM 1 TO .num_messages           ! For each message definition,
2 DO
3     BEGIN
3     BIND
3         msg = code [code$c_msg,0,0,0]: BBLOCK;  ! Reference MSG block
3     LOCAL
3         sec_index:      REF VECTOR;      ! Address of secondary index entry
3
3     !
3     !   Store the index to the record in the primary or secondary
3     !   index depending on whether we're doing a 1 or 2 level index.
3     !
3
3     IF .sec_index_len NEQ 0               ! If doing a 2 level index,
3     THEN
4         BEGIN
4         IF .bucket_filled EQL 0           ! If at start of bucket,
4         THEN
5             BEGIN
5             bucket [midx$w_size] = 0;             ! Preset size of bucket
5             bucket [midx$b_sanity] = midx$c_sanity;     ! Set sanity check byte
5             bucket_filled = bucket_overhead;    ! Account for space
5             sec_index = .bucket + bucket_overhead;      ! Set to first entry
4             END;
4         END
3     ELSE
3         sec_index = .prim_index;          ! Else, store directly into primary
3
3     IF .sec_index GEQ .section + .section [msc$l_text_off] ! If overflow,
3     THEN
4         BEGIN
4         SIGNAL(emsg(indexovfl));                  ! Signal index area overflow
4         RETURN emsg(indexovfl);
3         END;
3
3     sec_index [0] = .code [code$l_number]        ! Insert message code in index
3                     AND NOT sts$m_severity;  ! but leave severity bits zero
3     sec_index [1] = .data - .section;           ! Store offset to record
```

```
3
3   !
3   !   Store message definition record into data area
3   !
3
3   IF .data GEQ .section + .section [msc$l_fac_off] ! If overflow,
3   THEN
4       BEGIN
4       SIGNAL(emsg(dataovfl));                     ! Signal data area overflow
4       RETURN emsg(dataovfl);
3       END;
3
3   CH$MOVE(.msg [mrec$w_size], msg, .data);        ! Store MSG record in data area
3
3   data = .data + .msg [mrec$w_size];              ! Skip to next available byte
3
3   !
3   !   If doing 2 level index, increment space used in bucket.
3   !   Check for secondary index bucket full.  If full, then
3   !   put the highest key in the bucket into the primary index.
3   !
3
3   IF .sec_index_len NEQ 0             ! If doing 2 level index,
3   THEN
4       BEGIN
4       bucket_filled = .bucket_filled + entry_size;! Account for space used
4
4       IF .bucket_filled GTR bucket_size-entry_size ! If last entry in bucket,
4           OR .i EQL .num_messages                  ! or last message
4       THEN
5           BEGIN
5           bucket [midx$w_size] = .bucket_filled;       ! Set size of bucket
5           IF .prim_index GEQ .section + .section [msc$l_index_off]
5                           + .prim_index_len ! If overflow,
5           THEN
6               BEGIN
6               SIGNAL(emsg(indexovfl));          ! Signal index area overflow
6               RETURN emsg(indexovfl);
5               END;
5           prim_index [0] = .sec_index [0];      ! Store highest entry in bucket
6           prim_index [1] = (.bucket - .section) ! Store offset to bucket
5                       OR 1;                     ! with bottom bit set (subindex)
5           prim_index = .prim_index + entry_size;        ! Set to next entry
5           bucket = .bucket + bucket_size;               ! Set to next bucket
5           bucket_filled = 0;                    ! reset usage counter
5           END
4       ELSE                            ! Else, if still more room in bucket,
4           sec_index = .sec_index + entry_size;         ! Set to next entry
4       END
4
4   !
4   !   If only doing a 1 level index, then we need only to
4   !   increment to the next position in the primary index.
4   !
4
3   ELSE                                ! Else, if only doing primary index,
3       prim_index = .prim_index + entry_size;  ! Set to next primary entry
```

```
3
3       code = .code [code$l_link];           ! and skip to next CODE block
2       END;
2
2 !
2 !         Generate the facility definition table in the section
2 !
2
2 data [0,0,16,0] = .fac_space - 2;          ! Set size of entire table (excl. size)
2 data = .data + 2;
2
2 fac = .facility_header;                    ! Start at first entry in list
2 INCR i FROM 1 TO .num_facilities           ! For each facility definition,
2 DO
3       BEGIN
3       IF .data GEQ .section + .section [msc$l_size] ! If overflow,
3       THEN
4           BEGIN
4           SIGNAL(emsg(facovfl));           ! Signal facility table overflow
4           RETURN emsg(facovfl);
3           END;
3       data [mfac$w_number] = .fac [fac$w_number]; ! Store facility number
3       data [mfac$b_namelen] = .fac [fac$b_namelen]; ! and facility name
3       CH$MOVE(.fac [fac$b_namelen], fac [fac$t_name], data [mfac$t_name]);
3       data = .data + $BYTEOFFSET(mfac$t_name) + .fac [fac$b_namelen];
3       fac = .fac [fac$l_link];             ! Link to next in chain
2       END;
2
2 RETURN true;
2
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$_INDEXOVFL, R0                       ; 0662
                                  04 00114        RET                                               ;
              66    04  AA    07  CB 00115 9$:    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  D0 0020E         MOVL    #1, R0                          ; 0753
                                 04 00211         RET                                     ; 0755

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

```
1 ROUTINE create_indirect =
1
1 !---
1 !
1 !       Create the indirect message section.  This section
1 !       will merely contain the name of the message file which
1 !       should be mapped at run-time by SYS$GETMSG.
1 !
1 ! Inputs:
1 !
1 !       filename_desc = Descriptor of /FILE value
1 !
1 ! Outputs:
1 !
1 !       r0 = status (unsignaled)
1 !       section = Address of section
1 !---
1
2 BEGIN
2
2 LOCAL
2     status,                                 ! status code
2     section_size;                           ! Size of section in bytes
2
2 section_size = $BYTEOFFSET(msc$t_indname) + .filename_desc [dsc$w_length];
2
2 status = allocate(.section_size,section);  ! Allocate space for section
2 IF NOT .status                              ! If error allocating space
2 THEN
2     RETURN .status;                         ! then return with status
2
2 section [msc$b_type] = msc$c_ind;           ! Set type to indirect
2 section [msc$b_flags] = 0;                  ! Initialize flags as all zero.
2 section [msc$w_sanity] = msc$c_sanity;      ! Set sanity check word
2 section [msc$l_size] = .section_size;       ! Set size of section
2 section [msc$b_indnamlen] = .filename_desc [dsc$w_length];       ! Set length of name
2 CH$MOVE(.filename_desc [dsc$w_length], .filename_desc [dsc$a_pointer], section [msc$t_indname]);
2
2 RETURN true;                                ! return with success
2
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
```

OBJ
V04
; R

```
ROUTINE output_mhd =

!---
!
!       Output a module header record.
!
! Inputs:
!
!       None
!
! Outputs:
!
!       Module header record is output.
!---

BEGIN

BIND
    lang_name = UPLIT('VAX-11 Message V04-00');

LITERAL
    lang_length = 21;

LOCAL
    rec:        BBLOCK[128],                ! Allocate record buffer
    offset,                                 ! Offset to current spot in record
    bufdesc:    VECTOR[2];                  ! General buffer descriptor

rec [obj$b_rectyp] = obj$c_hdr;             ! Set record type = HDR
rec [mhd$b_hdrtyp] = mhd$c_mhd;             ! Main module header record
rec [mhd$b_strlvl] = 0;                     ! Structure level 0
rec [mhd$w_recsiz] = obj$c_maxrecsiz;       ! Maximum record size

rec [mhd$b_namlng] = .module_name [0];  ! Length of module name
CH$MOVE(.module_name [0], .module_name[1], rec [mhd$t_name]);
offset = $BYTEOFFSET(mhd$t_name) + .module_name [0];

IF .version_num [0]  EQL  0
THEN
    !
    ! No .IDENT was specified
    !
    BEGIN
    rec [.offset,0,8,0] = 1;
    rec [.offset,8,8,0] = '0';
    offset = .offset + 2;
    END

ELSE
    !
    ! .IDENT was specified
    !
    BEGIN
    rec [.offset,0,8,0] = .version_num [0];
    offset = .offset + 1;
    CH$MOVE(.version_num [0],.version_num [1],rec [.offset,0,0,0]);
    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$ASCTIM, R8                     ;
                 5E      FF78   CE   9E 00009                 MOVAB   -136(SP), SP                       ;
                           08   AE   B4 0000E                 CLRW    REC                                ; 0825
                           0A   AE   94 00011                 CLRB    REC+2                              ; 0827
           0B    AE      0800   8F   B0 00014                 MOVW    #2048, REC+3                       ; 0828
                 56     0000G   CF   D0 0001A                 MOVL    MODULE_NAME, R6                    ; 0830
           0D    AE             56   90 0001F                 MOVB    R6, REC+5                          ;
 0E   AE  0000G  DF             56   28 00023                 MOVC3   R6, @MODULE_NAME+4, REC+6          ; 0831
                 56             06   C0 0002A                 ADDL2   #6, OFFSET                         ; 0832
                 50       08 AE46    9E 0002D                 MOVAB   REC[OFFSET], R0                    ; 0840
                 57     0000G   CF   D0 00032                 MOVL    VERSION_NUM, R7                    ; 0834
                                0A   12 00037                 BNEQ    1$                                 ;
                 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$:   MOVL    #17, BUFDESC                        ; 0857
          04      AE     08 AE46 9E 00005A       MOVAB   REC[OFFSET], BUFDESC+4              ; 0858
                               7E 7C 00060       CLRQ    -(SP)                               ; 0859
                        08     AE 9F 00062       PUSHAB  BUFDESC
                               7E D4 00065       CLRL    -(SP)
                  68           04 FB 00067       CALLS   #4, SYS$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
```

```
1 ROUTINE output_eom =
1
1 !---
1 !
1 !       Output an end of module record
1 !
1 ! Inputs:
1 !
1 !       None
1 !
1 ! Outputs:
1 !
1 !       The record is output.
1 !---
1
2 BEGIN
2
2 LOCAL
2     rec:        BBLOCK[128];            ! Allocate record buffer
2
2 rec [obj$b_rectyp] = obj$c_eom;         ! Set record type = EOM
2 rec [eom$b_comcod] = eom$c_success;     ! Set completion severity
2
2 put_record(rec, 2);                     ! Output record
2
2 RETURN true;
2
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
```

```
1 ROUTINE output_psects (psect_flags) =
1
1 !---
1 !
1 !       This routine outputs all the psect definitions
1 !       for normal message sections.  Only the absolute
1 !       psect is output if /NOTEXT is specified.
1 !
1 ! Inputs:
1 !
1 !       psect_flags = Psect attributes
1 !
1 ! Outputs:
1 !
1 !       The records are output.
1 !---
1
2 BEGIN
2
2 LOCAL
2     entry:      BBLOCK[129];              ! Entry buffer
2
2 start_record(obj$c_gsd);                  ! Signal start of GSD record
2
2 !
2 !       Define psects used in this object module
2 !
2
2 entry [gps$b_gsdtyp] = gsd$c_psc;         ! Psect definition entry
2 entry [gps$b_align] = 0;                  ! Psect alignment (byte)
2 entry [gps$w_flags] = gps$m_exe OR gps$m_rd OR gps$m_wrt; ! Psect attributes
2 entry [gps$l_alloc] = 0;                  ! Set size of section
2 entry [gps$b_namlng] = 5;                 ! Length of psect name
2 CH$MOVE(5,UPLIT('$ABS$'),entry [gps$t_name]);   ! Set psect name
2 put_entry(entry, $BYTEOFFSET(gps$t_name)+5);    ! Output entry
2
2 IF NOT .cli_flags [qual_text]  AND             ! If /NOTEXT specified,
2    NOT .cli_flags [qual_file]                  !       and no pointers
2 THEN
3     BEGIN
3     end_record();                         ! then flush absolute psect only
3     RETURN true;                          ! and return
2     END;
2
2 entry [gps$w_flags] = .psect_flags;       ! Psect attributes
2 entry [gps$l_alloc] = .section [msc$l_size];    ! Set size of section
2
2 IF .cli_flags[qual_file]
2 THEN
3     BEGIN
3     entry [gps$b_namlng] = 14;            ! Length of psect name
3     CH$MOVE(14,UPLIT('MSGPTR$SECTION'),entry [gps$t_name]);! Set ptr psect name
3     put_entry(entry, $BYTEOFFSET(gps$t_name)+14);      ! Output entry
3     END
2 ELSE
3     BEGIN
3     entry [gps$b_namlng] = 11;            ! Length of psect name
```

```
3       CH$MOVE(11,UPLIT('MSG$SECTION'),entry [gps$t_name]);    ! Set psect name
3       put_entry(entry, $BYTEOFFSET(gps$t_name)+11);          ! Output entry
2       END;
2
2  entry [gps$b_align] = 2;                          ! Psect alignment (longword)
2  entry [gps$w_flags] = .psect_flags OR gps$m_ovr; ! This psect is overlayed
2  entry [gps$l_alloc] = 4*4;                        ! Size of psect contribution
2  entry [gps$b_namlng] = 15;                        ! Length of psect name
2
2  IF .cli_flags[qual_file]
2  THEN
2      CH$MOVE(15,UPLIT('MSGPTR$AAAAAAAA'),entry [gps$t_name])! Set ptr psect name
2  ELSE
2      CH$MOVE(15,UPLIT('MSG$AAAAAAAAAAA'),entry [gps$t_name]); ! Set psect name
2
2  put_entry(entry, $BYTEOFFSET(gps$t_name)+15);    ! Output entry
2
2  entry [gps$w_flags] = .psect_flags;          ! Normal psect flags
2  entry [gps$l_alloc] = 4;                     ! Size of psect contribution
2
2  IF .cli_flags[qual_file]
2  THEN
2      CH$MOVE(15,UPLIT('MSGPTR$AAAAAAAB'),entry [gps$t_name])! Set ptr psect name
2  ELSE
2      CH$MOVE(15,UPLIT('MSG$AAAAAAAAAAB'),entry [gps$t_name]); ! Set psect name
2
2  put_entry(entry, $BYTEOFFSET(gps$t_name)+15);    ! Output entry
2
2  entry [gps$w_flags] = .psect_flags OR gps$m_ovr; ! This psect is overlayed
2
2  IF .cli_flags[qual_file]
2  THEN
2      CH$MOVE(15,UPLIT('MSGPTR$AAAAAAAC'),entry [gps$t_name])! Set ptr psect name
2  ELSE
2      CH$MOVE(15,UPLIT('MSG$AAAAAAAAAAC'),entry [gps$t_name]); ! Set psect name
2
2  put_entry(entry, $BYTEOFFSET(gps$t_name)+15);    ! Output entry
2
2  end_record();                                ! Flush rest of record buffer
2
2  RETURN true;
2
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 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                  ; 0937
                                  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
```

```
1 ROUTINE output_gsd =
1
1 !---
1 !
1 !       This routine outputs all the global symbol
1 !       definitions to symbolically refer to the
1 !       messages.
1 !
1 ! Inputs:
1 !
1 !       None
1 !
1 ! Outputs:
1 !
1 !       The records are output.
1 !---
1
2 BEGIN
2
2 LOCAL
2     entry:      BBLOCK[129],              ! Entry buffer
2     ptr:        REF BBLOCK;               ! Address of current symbol entry
2
2 start_record(obj$c_gsd);                  ! Signal start of GSD record
2
2 entry [sdf$b_gsdtyp] = gsd$c_sym;         ! Global symbol definition
2 entry [sdf$b_datyp] = 0;                  ! Data type undefined
2 entry [sdf$w_flags] = gsy$m_def;          ! Definition rather than reference
2 entry [sdf$b_psindx] = 0;                 ! Owning psect number 0 ($ABS$)
2
2 ptr = .symbol_header;                     ! Point at first symbol entry
2
2 WHILE .ptr NEQ 0                          ! Until end of list
2 DO
3     BEGIN
3     entry [sdf$l_value] = .ptr [sym$l_value];
3     entry [sdf$b_namlng] = .ptr [sym$b_symlen];
3     CH$MOVE(.ptr [sym$b_symlen], ptr [sym$t_symbol], entry [sdf$t_name]);
3     put_entry(entry, $BYTEOFFSET(sdf$t_name)+.entry [sdf$b_namlng]);
3     ptr = .ptr [sym$l_link];              ! Skip to next in chain
2     END;
2
2 end_record();                             ! Flush rest of record buffer
2
2 RETURN true;
2
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
```

```
1 ROUTINE output_section =
1
1 !---
1 !
1 !       This routine outputs the TIR records to the object
1 !       module file to describe the message section.
1 !
1 ! Inputs:
1 !
1 !       None
1 !
1 ! Outputs:
1 !
1 !       The records are written.
1 !---
1
2 BEGIN
2
2 LOCAL
2     entry:      VECTOR[129,BYTE],       ! TIR command buffer
2     position,                           ! Position within section
2     bytes_left;                         ! Bytes left of section to output
2
2 start_record(obj$c_tir);                ! Signal start of TIR record
2
2 !
2 !       Output the message section vector dispatcher (psect 1)
2 !       The name of the psect is set so that it will be loaded
2 !       in front of the message section list.
2 !
2
2 set_psect(2,0);                         ! Set to psect 2
2
2 entry [0] = -16;                        ! Store immedate 16 bytes
3 BEGIN
3 LOCAL
3     vector: REF BBLOCK;                 ! Address buffer as longwords
3
3 vector = entry [1];                     ! Address the start of the vector
3 vector [plv$l_type] = plv$c_typ_msg;    ! Type of vector
3 vector [plv$l_version] = 0;             ! (reserved)
3 vector [plv$l_msgdsp] = 6;              ! Self-rel offset to dispatcher code
3 vector [12,0,32,0] = %x'65160101';      ! NOP, NOP, JSB (R5) instructions
2 END;
2 put_entry(entry,17);                    ! Output TIR command
2
2 !
2 !       Output the self-relative offset to the mssage section (psect 3).
2 !       It will be contributed to the section list psect at
2 !       the end - resulting in a list of section offsets.
2 !       Each offset to the sections is relative to the start
2 !       of the longword itself.
2 !
2
2 set_psect(3,0);                         ! Set to psect 3
2
2 entry [0] = tir$c_sta_sb;               ! Stack signed byte
```

```
entry [1] = -1;                              ! Sign extended -1
entry [2] = tir$c_sta_pb;                    ! Stack psect base plus offset
entry [3] = 1;                               ! psect number of message section
entry [4] = 0;                               ! start of psect
entry [5] = tir$c_sto_pirr;                  ! Store self-relative offset to psect 1

put_entry(entry,6);                          ! Output TIR command

!
!       Output a zero list terminator (psect 4).
!       This psect is an overlayed psect of length 4 which
!       contains a zero.  The name of the psect is set
!       so that it will be loaded following the message
!       section list as a list terminator.
!

set_psect(4,0);                              ! Set to psect 4

entry [0] = tir$c_sta_sb;                    ! Stack signed byte
entry [1] = 0;                               ! value = 0 (byte to store)
entry [2] = tir$c_sta_sb;                    ! Stack signed byte
entry [3] = 4;                               ! value = 4 (repeat count)
entry [4] = tir$c_sto_rb;                    ! Store repeated byte (4 zero bytes)

put_entry(entry,5);                          ! Output TIR commands

!
!       Output the message section text
!

set_psect(1,0);                              ! Set to psect 1

position = .section;                         ! Start at beginning of section
bytes_left = .section [msc$l_size];          ! Bytes left to output

WHILE .bytes_left GTR 0                      ! While stuff left to output,
DO
    BEGIN
    LOCAL bytes;

    bytes = MINU(.bytes_left,128);           ! Max 128 bytes per record
    entry [0] = -.bytes;                     ! Number of bytes to output
    CH$MOVE(.bytes,.position,entry[1]);      ! Move the bytes
    put_entry (entry, .bytes+1);             ! Output the TIR command
    position = .position + .bytes;           ! Skip past stuff output
    bytes_left = .bytes_left - .bytes;       ! Decrement amount remaining
    END;

end_record();                        ! Flush remaining record buffer

RETURN true;

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
           00000000G  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

; R

```
1 ROUTINE start_record (type) =
1
1 !---
1 !
1 !       This routine is called when a variable size record
1 !       (like GSD or TIR) is about to be output.
1 !
1 ! Inputs:
1 !
1 !       type = Type of variable record (GSD or TIR)
1 !
1 ! Outputs:
1 !
1 !       rec_filled = Address of next available byte in record
1 !---
1
2 BEGIN
2
2 record_buffer [obj$b_rectyp] = .type;        ! Set record type
2 rec_filled = 1;                              ! Set offset to beginning
2
2 RETURN true;
2
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
```

```
ROUTINE put_entry (address, length) =

!---
!
!       This routine outputs a GSD/TIR entry into the current record
!       buffer.  The record buffer will be output automatically
!       when full.
!
! Inputs:
!
!       address = Address of GSD entry buffer
!       length = Length of entry
!       rec_filled = Current offset into record buffer
!
! Outputs:
!
!       rec_filled updated.
!---

BEGIN

IF (.rec_filled+.length) GTR obj$c_maxrecsiz     ! If record too large,
THEN
    BEGIN
    put_record(record_buffer,.rec_filled);       ! Output the record
    rec_filled = 1;                              ! Start at beginning again
    END;

CH$MOVE(.length, .address, record_buffer+.rec_filled); ! Move into record buffer

rec_filled = .rec_filled + .length;              ! Update buffer position

RETURN true;

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