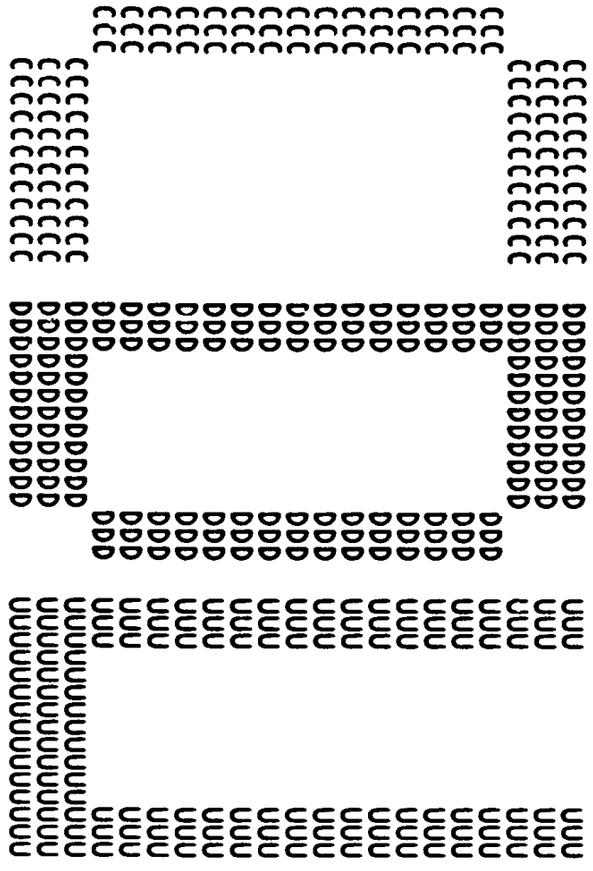

```
  SSSSSSSS  YY      YY  MM      MM  BBBBBBBB     000000    LL            SSSSSSSS
  SSSSSSSS  YY      YY  MM      MM  BBBBBBBB     000000    LL            SSSSSSSS
SS          YY      YY  MMMM  MMMM  BB      BB  00    00  LL          SS
SS          YY      YY  MMMM  MMMM  BB      BB  00    00  LL          SS
SS            YY  YY    MM  MM  MM  BB      BB  00    00  LL          SS
SS            YY  YY    MM  MM  MM  BB      BB  00    00  LL          SS
  SSSSSS        YY      MM      MM  BBBBBBBB    00    00  LL            SSSSSS
  SSSSSS        YY      MM      MM  BBBBBBBB    00    00  LL            SSSSSS
        SS      YY      MM      MM  BB      BB  00    00  LL                  SS
        SS      YY      MM      MM  BB      BB  00    00  LL                  SS
        SS      YY      MM      MM  BB      BB  00    00  LL                  SS      ....
        SS      YY      MM      MM  BB      BB  00    00  LL                  SS      ....
SSSSSSSS        YY      MM      MM  BBBBBBBB     000000    LLLLLLLLLL  SSSSSSSS      ....
SSSSSSSS        YY      MM      MM  BBBBBBBB     000000    LLLLLLLLLL  SSSSSSSS      ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

```
;     1          0001  0 MODULE symbols          (IDENT='V04-000',
;     2          0002  0                         ADDRESSING_MODE(EXTERNAL=GENERAL))
;     3          0003  1 = BEGIN
;     4          0004  1
;     5          0005  1 !*****************************************************************************
;     6          0006  1 !*                                                                           *
;     7          0007  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;     8          0008  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
;     9          0009  1 !*  ALL RIGHTS RESERVED.                                                     *
;    10          0010  1 !*                                                                           *
;    11          0011  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
;    12          0012  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    13          0013  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;    14          0014  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;    15          0015  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;    16          0016  1 !*  TRANSFERRED.                                                             *
;    17          0017  1 !*                                                                           *
;    18          0018  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;    19          0019  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;    20          0020  1 !*  CORPORATION.                                                             *
;    21          0021  1 !*                                                                           *
;    22          0022  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
;    23          0023  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
;    24          0024  1 !*                                                                           *
;    25          0025  1 !*                                                                           *
;    26          0026  1 !*****************************************************************************
;    27          0027  1
;    28          0028  1 !++
;    29          0029  1 ! Facility:     Command Definition Utility, Symbol Table Module
;    30          0030  1 !
;    31          0031  1 ! Abstract:     This module contains the routines necessary to create a
;    32          0032  1 !               symbol table file for use by the old CLI interface.
;    33          0033  1 !               Symbols are generated for each verb, syntax, or type
;    34          0034  1 !               definition which includes a PREFIX clause.  These symbols
;    35          0035  1 !               specify the number of each qualifier or keyword defined.
;    36          0036  1 !
;    37          0037  1 ! Environment:  Standard CDU environment.
;    38          0038  1 !
;    39          0039  1 ! Author:       Paul C. Anagnostopoulos
;    40          0040  1 ! Creation:     16 February 1983
;    41          0041  1 !
;    42          0042  1 ! Modifications:
;    43          0043  1 !--
;    44          0044  1
;    45          0045  1
;    46          0046  1 library 'sys$library:lib';
;    47          0047  1 require 'clitabdef';
;    48          0372  1 require 'cdureq';
```

```
;    50          0786  1 !       T A B L E   O F   C O N T E N T S
;    51          0787  1 !       ---------   ---   -------------
;    52          0788  1
;    53          0789  1 forward routine
;    54          0790  1         cdu$write_symbol_table_file: novalue,
;    55          0791  1         cdu$close_symbol_table_file: novalue,
;    56          0792  1         write_header_records: novalue,
;    57          0793  1         write_psect_record: novalue,
;    58          0794  1         write_symbol_records: novalue,
;    59          0795  1         write_eom_record: novalue;
;    60          0796  1
;    61          0797  1
;    62          0798  1 !       E X T E R N A L   R E F E R E N C E S
;    63          0799  1 !       -------------   -----------------
;    64          0800  1
;    65          0801  1 external routine
;    66          0802  1         cdu$lookup_child,
;    67          0803  1         cdu$report_rms_error,
;    68          0804  1         cli$get_value,
;    69          0805  1         lib$free_vm,
;    70          0806  1         lib$get_vm;
;    71          0807  1
;    72          0808  1 external
;    73          0809  1         cdu$facility_string: descriptor,
;    74          0810  1         cdu$gl_root_node: ref node;
;    75          0811  1
;    76        P 0812  1 $shr_msgdef(cdu,17,local,
;    77        P 0813  1         (openout,severe),
;    78        P 0814  1         (writeerr,severe)
;    79          0815  1         );
```

```
;   81          0816  1 !       S Y M B O L   T A B L E   F I L E   C O N T R O L   B L O C K S
;   82          0817  1 !       -----------   ---------   -------   -------------   -----------
;   83          0818  1
;   84          0819  1 ! The following items define the RMS control blocks needed to create and
;   85          0820  1 ! write the symbol table file.
;   86          0821  1
;   87          0822  1 own
;   88          0823  1         stb_esa: block[nam$c_maxrss,byte],
;   89          0824  1         stb_rsa: block[nam$c_maxrss,byte],
;   90        P 0825  1         stb_nam: $nam(
;   91        P 0826  1                         esa=stb_esa,
;   92        P 0827  1                         ess=%allocation(stb_esa),
;   93        P 0828  1                         rsa=stb_rsa,
;   94        P 0829  1                         rss=%allocation(stb_rsa)
;   95          0830  1                         ),
;   96          0831  1
;   97          0832  1         dbuffer(stb_spec,nam$c_maxrss),
;   98        P 0833  1         stb_fab: $fab(
;   99        P 0834  1                         dnm='.STB',
;  100        P 0835  1                         fna=stb_spec+8,
;  101        P 0836  1                         fns=%allocation(stb_spec)-8,
;  102        P 0837  1                         fac=put,
;  103        P 0838  1                         fop=<sqo,nam,ofp>,
;  104        P 0839  1                         nam=stb_nam,
;  105        P 0840  1                         org=seq,
;  106        P 0841  1                         rat=cr,
;  107        P 0842  1                         rfm=var
;  108          0843  1                         ),
;  109          0844  1
;  110        P 0845  1         stb_rab: $rab(
;  111        P 0846  1                         fab=stb_fab,
;  112        P 0847  1                         rac=seq,
;  113        P 0848  1                         rop=wbh
;  114          0849  1                         );
```

```
;  116          0850  1 !++
;  117          0851  1 ! Description:  This routine is called after a CLD file has been compiled
;  118          0852  1 !                into its intermediate representation.  We must determine
;  119          0853  1 !                all of the symbols that need to be added to the symbol
;  120          0854  1 !                table file and write them.
;  121          0855  1 !
;  122          0856  1 !                Upon first call, we create the symbol table file and write
;  123          0857  1 !                the records needed to start it off.
;  124          0858  1 !
;  125          0859  1 ! Parameters:   None.
;  126          0860  1 !
;  127          0861  1 ! Returns:      Nothing.
;  128          0862  1 !
;  129          0863  1 ! Notes:
;  130          0864  1 !--
;  131          0865  1
;  132          0866  1 GLOBAL ROUTINE cdu$write_symbol_table_file      : novalue
;  133          0867  2 = BEGIN
;  134          0868  2
;  135          0869  2 local
;  136          0870  2         status: long;
;  137          0871  2
;  138          0872  2
;  139          0873  2 ! If we haven't yet created the symbol table file, let's do it.
;  140          0874  2
;  141          0875  3 if .stb_fab[fab$w_ifi] eqlu 0 then (
;  142          0876  3
;  143          0877  3         ! Get the value specified on the /SYMBOLS qualifier to use as
;  144          0878  3         ! the spec for the symbol table file.
;  145          0879  3
;  146          0880  3         cli$get_value(dtext('SYMBOLS'),stb_spec);
;  147          0881  3
;  148          0882  3         ! Create and connect to the symbol table file.  Any errors are fatal.
;  149          0883  3
;  150          0884  3         status = $create(fab=stb_fab);
;  151          0885  3         if not .status then
;  152          0886  3                 cdu$report_rms_error(msg(cdu$_openout),stb_fab);
;  153          0887  3         status = $connect(rab=stb_rab);
;  154          0888  3         if not .status then
;  155          0889  3                 cdu$report_rms_error(msg(cdu$_openout),stb_rab);
;  156          0890  3
;  157          0891  3         ! Write the header records.
;  158          0892  3
;  159          0893  3         write_header_records();
;  160          0894  3
;  161          0895  3         ! Write the absolute psect definition record.
;  162          0896  3
;  163          0897  3         write_psect_record();
;  164          0898  2 );
;  165          0899  2
;  166          0900  2 ! Now we can write the symbol definition records.
;  167          0901  2
;  168          0902  2 write_symbol_records();
;  169          0903  2
;  170          0904  2 return;
;  171          0905  2
;  172          0906  1 END;
```

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

                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

                                42  54  53  2E  00000 P.AAA:  .ASCII  \.STB\
                00  53  4C  4F  42  4D  59  53  00004 P.AAC:  .ASCII  \SYMBOLS\<0>
                                  010E0007  0000C P.AAB:  .LONG   17694727
                                  00000000' 00010         .ADDRESS P.AAC

                                                        .PSECT  $OWN$,NOEXE,2

                                        00000 STB_ESA:.BLKB   255
                                        000FF         .BLKB   1
                                        00100 STB_RSA:.BLKB   255
                                        001FF         .BLKB   1
                                     02 00200 STB_NAM:.BYTE   2
                                     60 00201         .BYTE   96
                                     FF 00202         .BYTE   -1
                                     00 00203         .BYTE   0
                                  00000000' 00204         .ADDRESS STB_RSA
                                     00 00208         .BYTE   0
                                     00 00209         .BYTE   0
                                     FF 0020A         .BYTE   -1
                                     00 0020B         .BYTE   0
                                  00000000' 0020C         .ADDRESS STB_ESA
                                  00000000  00210         .LONG   0
                                      0000# 00214         .WORD   0[8]
                                      0000# 00224         .WORD   0[3]
                                      0000# 0022A         .WORD   0[3]
                                  00000000  00230         .LONG   0
                                  00000000  00234         .LONG   0
                                     00 00238         .BYTE   0
                                     00 00239         .BYTE   0
                                     00 0023A         .BYTE   0
                                     00 0023B         .BYTE   0
                                     00 0023C         .BYTE   0
                                     00 0023D         .BYTE   0
                                     00# 0023E         .BYTE   0[2]
                                  00000000  00240         .LONG   0
                                  00000000  00244         .LONG   0
                                  00000000  00248         .LONG   0
                                  00000000  0024C         .LONG   0
                                  00000000  00250         .LONG   0
                                  00000000  00254         .LONG   0
                                  00000000# 00258         .LONG   0[2]
                                      00FF  00260 STB_SPEC:
                                                        .WORD   255
                                    00  00  00262         .BYTE   0, 0
                                  00000000' 00264         .ADDRESS STB_SPEC+8
                                        00268         .BLKB   255
                                        00367         .BLKB   1
                                     03 00368 STB_FAB:.BYTE   3
                                     50 00369         .BYTE   80
                                   0000 0036A         .WORD   0
```

```
21000040  0036C             .LONG   553648192
00000000  00370             .LONG   0
00000000  00374             .LONG   0
00000000  00378             .LONG   0
     0000  0037C             .WORD   0
       01  0037E             .BYTE   1
       00  0037F             .BYTE   0
00000000  00380             .LONG   0
       00  00384             .BYTE   0
       00  00385             .BYTE   0
       02  00386             .BYTE   2
       02  00387             .BYTE   2
00000000  00388             .LONG   0
00000000  0038C             .LONG   0
00000000' 00390             .ADDRESS STB_NAM
00000000' 00394             .ADDRESS STB_SPEC+8
00000000' 00398             .ADDRESS P.AAA
       FF  0039C             .BYTE   -1
       04  0039D             .BYTE   4
     0000  0039E             .WORD   0
00000000  003A0             .LONG   0
     0000  003A4             .WORD   0
       00  003A6             .BYTE   0
       00  003A7             .BYTE   0
00000000  003A8             .LONG   0
00000000  003AC             .LONG   0
     0000  003B0             .WORD   0
       00  003B2             .BYTE   0
       00  003B3             .BYTE   0
00000000  003B4             .LONG   0
       01  003B8 STB_RAB:    .BYTE   1
       44  003B9             .BYTE   68
     0000  003BA             .WORD   0
00000400  003BC             .LONG   1024
00000000  003C0             .LONG   0
00000000  003C4             .LONG   0
     0000# 003C8             .WORD   0[3]
     0000  003CE             .WORD   0
00000000  003D0             .LONG   0
     0000  003D4             .WORD   0
       00  003D6             .BYTE   0
       00  003D7             .BYTE   0
     0000  003D8             .WORD   0
     0000  003DA             .WORD   0
00000000  003DC             .LONG   0
00000000  003E0             .LONG   0
00000000  003E4             .LONG   0
00000000  003E8             .LONG   0
       00  003EC             .BYTE   0
       00  003ED             .BYTE   0
       00  003EE             .BYTE   0
       00  003EF             .BYTE   0
00000000  003F0             .LONG   0
00000000' 003F4             .ADDRESS STB_FAB
00000000  003F8             .LONG   0

                            .EXTRN  CDU$LOOKUP_CHILD
```

```
                                        .EXTRN  CDU$REPORT_RMS_ERROR
                                        .EXTRN  CLI$GET_VALUE, LIB$FREE_VM
                                        .EXTRN  LIB$GET_VM, CDU$FACILITY_STRING
                                        .EXTRN  CDU$GL_ROOT_NODE
                                        .EXTRN  SYS$CREATE, SYS$CONNECT

                                        .PSECT  $CODE$,NOWRT,2

                    001C 00000          .ENTRY  CDU$WRITE_SYMBOL_TABLE_FILE, Save R2,R3,R4  ; 0866
    54 00000000G    00   9E 00002        MOVAB   CDU$REPORT_RMS_ERROR, R4                    ;
    53      0000'   CF   9E 00009        MOVAB   STB_FAB, R3                                 ;
            02      A3   B5 0000E        TSTW    STB_FAB+2                                   ; 0875
                    4F   12 00011        BNEQ    3$                                          ;
            FEF8    C3   9F 00013        PUSHAB  STB_SPEC                                    ; 0880
            0000'   CF   9F 00017        PUSHAB  P.AAB                                       ;
00000000G   00           02   FB 0001B    CALLS   #2, CLI$GET_VALUE                           ;
                         53   DD 00022    PUSHL   R3                                          ; 0884
00000000G   00           01   FB 00024    CALLS   #1, SYS$CREATE                              ;
            52           50   D0 0002B    MOVL    R0, STATUS                                  ;
            0B           52   E8 0002E    BLBS    STATUS, 1$                                  ; 0885
                         53   DD 00031    PUSHL   R3                                          ; 0886
        001110A4         8F   DD 00033    PUSHL   #1118372                                    ;
            64           02   FB 00039    CALLS   #2, CDU$REPORT_RMS_ERROR                    ;
                50       A3   9F 0003C 1$:    PUSHAB  STB_RAB                                 ; 0887
00000000G   00           01   FB 0003F    CALLS   #1, SYS$CONNECT                             ;
            52           50   D0 00046    MOVL    R0, STATUS                                  ;
            0C           52   E8 00049    BLBS    STATUS, 2$                                  ; 0888
                50       A3   9F 0004C    PUSHAB  STB_RAB                                     ; 0889
        001110A4         8F   DD 0004F    PUSHL   #1118372                                    ;
            64           02   FB 00055    CALLS   #2, CDU$REPORT_RMS_ERROR                    ;
  0000V     CF           00   FB 00058 2$:    CALLS   #0, WRITE_HEADER_RECORDS                ; 0893
  0000V     CF           00   FB 0005D    CALLS   #0, WRITE_PSECT_RECORD                      ; 0897
  0000V     CF           00   FB 00062 3$:    CALLS   #0, WRITE_SYMBOL_RECORDS                ; 0902
                              04 00067    RET                                                 ; 0906

; Routine Size:  104 bytes,    Routine Base:  $CODE$ + 0000
```

```
;  174          0907  1 !++
;  175          0908  1 ! Description:  This routine is called when all CLDs have been processed and
;  176          0909  1 !                we are done writing the symbols.  We write the
;  177          0910  1 !                end-of-module record in the symbol table file.
;  178          0911  1 !
;  179          0912  1 ! Parameters:   None.
;  180          0913  1 !
;  181          0914  1 ! Returns:      Nothing.
;  182          0915  1 !
;  183          0916  1 ! Notes:
;  184          0917  1 !--
;  185          0918  1
;  186          0919  1 GLOBAL ROUTINE cdu$close_symbol_table_file      : novalue
;  187          0920  2 = BEGIN
;  188          0921  2
;  189          0922  2
;  190          0923  2 ! Write the end-of-module record.
;  191          0924  2
;  192          0925  2 write_eom_record();
;  193          0926  2
;  194          0927  2 return;
;  195          0928  2
;  196          0929  1 END;


                                0000 00000          .ENTRY  CDU$CLOSE_SYMBOL_TABLE_FILE, Save nothing   ; 0919
                     0000V  CF    00 FB 00002       CALLS   #0, WRITE_EOM_RECORD                        ; 0925
                                04 00007          RET                                                 ; 0929

; Routine Size:  8 bytes,    Routine Base:  $CODE$ + 0068
```

```
;  198          0930  1 !++
;  199          0931  1 ! Description:  This routine is responsible for writing the header records
;  200          0932  1 !               in the symbol table file.  We write the mandatory module
;  201          0933  1 !               record, along with a language name record.
;  202          0934  1 !
;  203          0935  1 ! Parameters:   None.
;  204          0936  1 !
;  205          0937  1 ! Returns:      Nothing.
;  206          0938  1 !
;  207          0939  1 ! Notes:
;  208          0940  1 !--
;  209          0941  1
;  210          0942  1 ROUTINE write_header_records    : novalue
;  211          0943  2 = BEGIN
;  212          0944  2
;  213          0945  2 local
;  214          0946  2         status: long,
;  215          0947  2         hdr: block[256,byte],
;  216          0948  2         variable_ptr: pointer,
;  217          0949  2         child: ref node,
;  218          0950  2         work_dsc: descriptor;
;  219          0951  2
;  220          0952  2
;  221          0953  2 ! Set up the fixed portion of a module header record.
;  222          0954  2
;  223          0955  2 hdr[obj$b_rectyp] = obj$c_hdr;
;  224          0956  2 hdr[mhd$b_hdrtyp] = mhd$c_mhd;
;  225          0957  2 hdr[mhd$b_strlvl] = obj$c_strlvl;
;  226          0958  2 hdr[mhd$w_recsiz] = obj$c_maxrecsiz;
;  227          0959  2
;  228          0960  2 ! Now we want to include the module name.  If there is a MODULE statement
;  229          0961  2 ! in the CLD, use it.  Otherwise use the name of the symbol table file.  While
;  230          0962  2 ! we're at it, set up a pointer to the next available byte in the header.
;  231          0963  2
;  232          0964  2 child = cdu$lookup_child(.cdu$gl_root_node,node_k_module);
;  233          0965  3 if .child neqa 0 then (
;  234          0966  3         ch$move(1+.child[node_b_text_length],child[node_b_text_length], hdr[mhd$b_namlng]);
;  235          0967  3         variable_ptr = hdr[mhd$t_name] + .child[node_b_text_length];
;  236          0968  3 ) else (
;  237          0969  3         hdr[mhd$b_namlng] = .stb_nam[nam$b_name];
;  238          0970  3         ch$move(.stb_nam[nam$b_name],.stb_nam[nam$l_name], hdr[mhd$t_name]);
;  239          0971  3         variable_ptr = hdr[mhd$t_name] + .stb_nam[nam$b_name];
;  240          0972  2 );
;  241          0973  2
;  242          0974  2 ! Now we want to include the module ident string.  If there is an IDENT
;  243          0975  2 ! statement, then use it.  Otherwise use a string of "0-0".
;  244          0976  2
;  245          0977  2 child = cdu$lookup_child(.cdu$gl_root_node,node_k_ident);
;  246          0978  3 if .child neqa 0 then (
;  247          0979  3         ch$move(1+.child[node_b_text_length],child[node_b_text_length], .variable_ptr);
;  248          0980  3         variable_ptr = .variable_ptr + 1+.child[node_b_text_length];
;  249          0981  3 ) else (
;  250          0982  3         ch$move(4,ctext('0-0'), .variable_ptr);
;  251          0983  3         variable_ptr = .variable_ptr + 4;
;  252          0984  2 );
;  253          0985  2
;  254          0986  2 ! Finally, we want to include the current date and time.
```

```
;  255          0987  2
;  256          0988  2 build_descriptor(work_dsc,17,.variable_ptr);
;  257          0989  2 status = $asctim(timbuf=work_dsc);
;  258          0990  2 check(.status, .status);
;  259          0991  2 variable_ptr = .variable_ptr + 17;
;  260          0992  2
;  261          0993  2 ! Write the module header into the symbol table file.  Any error is fatal.
;  262          0994  2
;  263          0995  2 stb_rab[rab$l_rbf] = hdr;
;  264          0996  2 stb_rab[rab$w_rsz] = .variable_ptr - hdr;
;  265          0997  2 status = $put(rab=stb_rab);
;  266          0998  2 if not .status then
;  267          0999  2         cdu$report_rms_error(msg(cdu$_writeerr),stb_rab);
;  268          1000  2
;  269          1001  2 ! Set up the fixed portion of a language name record.
;  270          1002  2
;  271          1003  2 hdr[obj$b_rectyp] = obj$c_hdr;
;  272          1004  2 hdr[mhd$b_hdrtyp] = mhd$c_lnm;
;  273          1005  2
;  274          1006  2 ! Move in our language name.
;  275          1007  2
;  276          1008  2 ch$move(.cdu$facility_string[len],.cdu$facility_string[ptr], hdr + 2);
;  277          1009  2
;  278          1010  2 ! Write the language name record in the symbol table file.
;  279          1011  2
;  280          1012  2 stb_rab[rab$w_rsz] = 2 + .cdu$facility_string[len];
;  281          1013  2 status = $put(rab=stb_rab);
;  282          1014  2 if not .status then
;  283          1015  2         cdu$report_rms_error(msg(cdu$_writeerr),stb_rab);
;  284          1016  2
;  285          1017  2 return;
;  286          1018  2
;  287          1019  1 END;


                                                .PSECT  $PLIT$,NOWRT,NOEXE,2

                        30  2D  30  03  00014 P.AAD:  .ASCII  <3>\0-0\                                    ;

                                                .EXTRN  SYS$ASCTIM, SYS$PUT

                                                .PSECT  $CODE$,NOWRT,2

                                    0FFC 00000 WRITE_HEADER_RECORDS:
                                                .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11        ; 0942
                    5B 00000000G    00   9E 00002          MOVAB   SYS$PUT, R11                    ;
                    5A 00000000G    00   9E 00009          MOVAB   CDU$LOOKUP_CHILD, R10           ;
                    59     0000'    CF   9E 00010          MOVAB   STB_RAB, R9                     ;
                    5E     FEF8     CE   9E 00015          MOVAB   -264(SP), SP                    ;
                             08     AE   B4 0001A          CLRW    HDR                             ; 0955
                             0A     AE   94 0001D          CLRB    HDR+2                           ; 0957
                0B   AE      0800   8F   B0 00020          MOVW    #2048, HDR+3                    ; 0958
                                    03   DD 00026          PUSHL   #3                              ; 0964
                        00000000G   00   DD 00028          PUSHL   CDU$GL_ROOT_NODE                ;
                    6A              02   FB 0002E          CALLS   #2, CDU$LOOKUP_CHILD            ;
                    57              50   D0 00031          MOVL    R0, CHILD                       ;
```

```
                                19  13 00034        BEQL    1$                                      ; 0965
               50       10      A7  9A 00036        MOVZBL  16(CHILD), R0                           ; 0966
                                50  D6 0003A        INCL    R0                                      ;
 0D    AE       10      A7      50  28 0003C        MOVC3   R0, 16(CHILD), HDR+5                    ;
                        50  0E  AE  9E 00042        MOVAB   HDR+6, R0                               ; 0967
                        58  10  A7  9A 00046        MOVZBL  16(CHILD), VARIABLE_PTR                 ;
                        58      50  C0 0004A        ADDL2   R0, VARIABLE_PTR                        ;
                                15  11 0004D        BRB     2$                                      ; 0965
                        56 FE83 C9  9A 0004F  1$:   MOVZBL  STB_NAM+59, R6                          ; 0969
                0D      AE      56  90 00054        MOVB    R6, HDR+5                               ;
 0E    AE     FE94      D9      56  28 00058        MOVC3   R6, @STB_NAM+76, HDR+6                  ; 0970
                        58   0E AE46 9E 0005F       MOVAB   HDR+6[R6], VARIABLE_PTR                 ; 0971
                                02  DD 00064  2$:   PUSHL   #2                                      ; 0977
                       00000000G 00 DD 00066        PUSHL   CDU$GL_ROOT_NODE                        ;
                        6A      02  FB 0006C        CALLS   #2, CDU$LOOKUP_CHILD                    ;
                        57      50  D0 0006F        MOVL    R0, CHILD                               ;
                                14  13 00072        BEQL    3$                                      ; 0978
                        56  10  A7  9A 00074        MOVZBL  16(CHILD), R6                           ; 0979
                        50  01  A6  9E 00078        MOVAB   1(R6), R0                               ;
       68       10      A7      50  28 0007C        MOVC3   R0, 16(CHILD), (VARIABLE_PTR)           ;
                        58   01 A648 9E 00081       MOVAB   1(R6)[VARIABLE_PTR], VARIABLE_PTR       ; 0980
                                05  11 00086        BRB     4$                                      ; 0978
                        88 0000' CF D0 00088  3$:   MOVL    P.AAD, (VARIABLE_PTR)+                  ; 0982
                        6E      11  D0 0008D  4$:   MOVL    #17, WORK_DSC                           ; 0988
                04      AE      58  D0 00090        MOVL    VARIABLE_PTR, WORK_DSC+4                ;
                                7E  7C 00094        CLRQ    -(SP)                                   ; 0989
                            08  AE  9F 00096        PUSHAB  WORK_DSC                                ;
                                7E  D4 00099        CLRL    -(SP)                                   ;
         00000000G      00      04  FB 0009B        CALLS   #4, SYS$ASCTIM                          ;
                        57      50  D0 000A2        MOVL    R0, STATUS                              ;
                        09      57  E8 000A5        BLBS    STATUS, 5$                              ; 0990
                                57  DD 000A8        PUSHL   STATUS                                  ;
         00000000G      00      01  FB 000AA        CALLS   #1, LIB$SIGNAL                          ;
                        58      11  C0 000B1  5$:   ADDL2   #17, VARIABLE_PTR                       ; 0991
                28      A9  08  AE  9E 000B4        MOVAB   HDR, STB_RAB+40                         ; 0995
                        50  08  AE  9E 000B9        MOVAB   HDR, R0                                 ; 0996
 22    A9               58      50  A3 000BD        SUBW3   R0, VARIABLE_PTR, STB_RAB+34            ;
                                59  DD 000C2        PUSHL   R9                                      ; 0997
                        6B      01  FB 000C4        CALLS   #1, SYS$PUT                             ;
                        57      50  D0 000C7        MOVL    R0, STATUS                              ;
                        0F      57  E8 000CA        BLBS    STATUS, 6$                              ; 0998
                                59  DD 000CD        PUSHL   R9                                      ; 0999
                       001110D4 8F  DD 000CF        PUSHL   #1118420                                ;
         00000000G      00      02  FB 000D5        CALLS   #2, CDU$REPORT_RMS_ERROR                ;
                08      AE 0100 8F  B0 000DC  6$:   MOVW    #256, HDR                               ; 1003
                        56 00000000G 00 3C 000E2    MOVZWL  CDU$FACILITY_STRING, R6                 ; 1008
                        50 00000000G 00 D0 000E9    MOVL    CDU$FACILITY_STRING+4, R0               ;
 0A    AE               60      56  28 000F0        MOVC3   R6, (R0), HDR+2                         ;
 22    A9               56      02  A1 000F5        ADDW3   #2, R6, STB_RAB+34                      ; 1012
                                59  DD 000FA        PUSHL   R9                                      ; 1013
                        6B      01  FB 000FC        CALLS   #1, SYS$PUT                             ;
                        57      50  D0 000FF        MOVL    R0, STATUS                              ;
                        0F      57  E8 00102        BLBS    STATUS, 7$                              ; 1014
                                59  DD 00105        PUSHL   R9                                      ; 1015
                       001110D4 8F  DD 00107        PUSHL   #1118420                                ;
         00000000G      00      02  FB 0010D        CALLS   #2, CDU$REPORT_RMS_ERROR                ;
                                    04 00114  7$:   RET                                             ; 1019
```

```
; Routine Size:  277 bytes,    Routine Base:  $CODE$ + 0070
```

```
;  289          1020  1 !++
;  290          1021  1 ! Description:  This routine is responsible for writing the absolute psect
;  291          1022  1 !                definition, which is needed so that all the symbols can
;  292          1023  1 !                reside in it.
;  293          1024  1 !
;  294          1025  1 ! Parameters:   None.
;  295          1026  1 !
;  296          1027  1 ! Returns:      Nothing.
;  297          1028  1 !
;  298          1029  1 ! Notes:
;  299          1030  1 !--
;  300          1031  1
;  301          1032  1 ROUTINE write_psect_record              : novalue
;  302          1033  2 = BEGIN
;  303          1034  2
;  304          1035  2 local
;  305          1036  2         status: long,
;  306          1037  2         gsd: block[256,byte];
;  307          1038  2
;  308          1039  2 bind
;  309          1040  2         gsd_psc = gsd + 1: block[,byte];
;  310          1041  2
;  311          1042  2
;  312          1043  2 ! Set up the fixed portion of the psect record.
;  313          1044  2
;  314          1045  2 gsd[obj$b_rectyp] = obj$c_gsd;
;  315          1046  2 gsd_psc[gps$b_gsdtyp] = gsd$c_psc;
;  316          1047  2 gsd_psc[gps$b_align] = 0;
;  317          1048  2 gsd_psc[gps$w_flags] = gps$m_rd + gps$m_wrt + gps$m_exe;
;  318          1049  2 gsd_psc[gps$l_alloc] = 0;
;  319          1050  2
;  320          1051  2 ! Now we want the psect name.
;  321          1052  2
;  322          1053  3 begin
;  323          1054  3 bind
;  324          1055  3         name = ctext('$ABS$'): vector[,byte];
;  325          1056  3
;  326          1057  3 ch$move(1+.name[0],name[0], gsd_psc[gps$b_namlng]);
;  327          1058  2 end;
;  328          1059  2
;  329          1060  2 ! Write the psect definition record into the symbol table file.
;  330          1061  2 ! Errors are fatal.
;  331          1062  2
;  332          1063  2 stb_rab[rab$l_rbf] = gsd;
;  333          1064  2 stb_rab[rab$w_rsz] = 1 + 8 + 1+.gsd_psc[gps$b_namlng];
;  334          1065  2 status = $put(rab=stb_rab);
;  335          1066  2 if not .status then
;  336          1067  2         cdu$report_rms_error(msg(cdu$_writeerr),stb_rab);
;  337          1068  2
;  338          1069  2 return;
;  339          1070  2
;  340          1071  1 END;


                                                                .PSECT  $PLIT$,NOWRT,NOEXE,2
```

```
                    24  53  42  41  24  05  00018 P.AAE:  .ASCII  <5>\$ABS$\                                ;

                                                NAME=               P.AAE


                                                        .PSECT  $CODE$,NOWRT,2

                                            007C 00000 WRITE_PSECT_RECORD:
                                                        .WORD   Save R2,R3,R4,R5,R6                       ; 1032
                            56      0000'   CF  9E 00002       MOVAB   STB_RAB+34, R6                    ;
                            5E      FF00    CE  9E 00007       MOVAB   -256(SP), SP                      ;
                            6E              01  B0 0000C       MOVW    #1, GSD                           ; 1045
                              02            AE  94 0000F       CLRB    GSD_PSC+1                         ; 1047
                        03  AE      01C0    8F  B0 00012       MOVW    #448, GSD_PSC+2                   ; 1048
                              05            AE  D4 00018       CLRL    GSD_PSC+4                         ; 1049
                            50      0000'   CF  9A 0001B       MOVZBL  NAME, R0                          ; 1057
                                            50  D6 00020       INCL    R0                                ;
                09  AE      0000'   CF      50  28 00022       MOVC3   R0, NAME, GSD_PSC+8               ;
                          06    A6          6E  9E 00029       MOVAB   GSD, STB_RAB+40                   ; 1063
                          66      09    AE  9B 0002D       MOVZBW  GSD_PSC+8, STB_RAB+34             ; 1064
                          66            0A  A0 00031       ADDW2   #10, STB_RAB+34                   ;
                                DE      A6  9F 00034       PUSHAB  STB_RAB                           ; 1065
                00000000G   00          01  FB 00037       CALLS   #1, SYS$PUT                       ;
                            10          50  E8 0003E       BLBS    STATUS, 1$                        ; 1066
                                DE      A6  9F 00041       PUSHAB  STB_RAB                           ; 1067
                          001110D4      8F  DD 00044       PUSHL   #1118420                          ;
                00000000G   00          02  FB 0004A       CALLS   #2, CDU$REPORT_RMS_ERROR          ;
                                            04 00051 1$:    RET                                       ; 1071

; Routine Size:  82 bytes,    Routine Base:  $CODE$ + 0185
```

```
;  342          1072  1 !++
;  343          1073  1 ! Description:  This routine is responsible for writing out the symbol
;  344          1074  1 !               definition records.  This is done by traversing the
;  345          1075  1 !               intermediate representation tree to locate any verb,
;  346          1076  1 !               syntax, or type definitions that specify a PREFIX clause.
;  347          1077  1 !               A symbol is then generated for every subordinate qualifier
;  348          1078  1 !               or keyword, but not for parameters.
;  349          1079  1 !
;  350          1080  1 ! Parameters:   None.
;  351          1081  1 !
;  352          1082  1 ! Returns:      Nothing.
;  353          1083  1 !
;  354          1084  1 ! Notes:
;  355          1085  1 !--
;  356          1036  1
;  357          1087  1 ROUTINE write_symbol_records            : novalue
;  358          1088  2 = BEGIN
;  359          1089  2
;  360          1090  2 local
;  361          1091  2         status: long,
;  362          1092  2         gsd: block[256,byte],
;  363          1093  2         definition: ref node,
;  364          1094  2         prefix: ref node,
;  365          1095  2         entity_type: long,
;  366          1096  2         entity_number: long,
;  367          1097  2         entity: ref node;
;  368          1098  2
;  369          1099  2 bind
;  370          1100  2         gsd_sym = gsd + 1: block[,byte];
;  371          1101  2
;  372          1102  2
;  373          1103  2 ! Scan the intermediate representation tree, looking for definitions.
;  374          1104  2
;  375        P 1105  2 scan_children(cdu$gl_root_node,definition,
;  376        P 1106  2
;  377        P 1107  2         ! If we have a definition, then determine whether that definition
;  378        P 1108  2         ! includes a PREFIX clause.
;  379        P 1109  2
;  380        P 1110  2         if (prefix = cdu$lookup_child(.definition,node_k_prefix)) neqa 0 then (
;  381        P 1111  2
;  382        P 1112  2                 ! If we have a prefix clause, then we want to generate symbols
;  383        P 1113  2                 ! for this definition.  If a verb or syntax, then we want
;  384        P 1114  2                 ! symbols for the qualifiers.  If a type, symbols for the
;  385        P 1115  2                 ! keywords.
;  386        P 1116  2
;  387        P 1117  2                 entity_type =   (if .definition[node_w_type] eqlu node_k_define_type then
;  388        P 1118  2                                         node_k_keyword
;  389        P 1119  2                                 else
;  390        P 1120  2                                         node_k_qualifier);
;  391        P 1121  2
;  392        P 1122  2                 ! Scan the children of the definition node looking for the
;  393        P 1123  2                 ! entities that need symbols defined.  As we go, we will count
;  394        P 1124  2                 ! the relevent entities.
;  395        P 1125  2
;  396        P 1126  2                 entity_number = 0;
;  397        P 1127  2                 scan_children(definition,entity,
;  398        P 1128  2
```

```
;  399          P 1129  2 if .entity[node_w_type] eqlu .entity_type then (
;  400          P 1130  2
;  401          P 1131  2         ! Increment the entity number.
;  402          P 1132  2
;  403          P 1133  2         increment(entity_number);
;  404          P 1134  2
;  405          P 1135  2         ! We have an entity for which a symbol must be
;  406          P 1136  2         ! generated.  Begin by building the fixed
;  407          P 1137  2         ! portion of a GSD symbol definition record.
;  408          P 1138  2         ! The symbol value is the entity number.
;  409          P 1139  2
;  410          P 1140  2         gsd[obj$b_rectyp] = obj$c_gsd;
;  411          P 1141  2         gsd_sym[sdf$b_gsdtyp] = gsd$c_sym;
;  412          P 1142  2         gsd_sym[sdf$b_datyp] = 0;
;  413          P 1143  2         gsd_sym[sdf$w_flags] = gsy$m_def;
;  414          P 1144  2         gsd_sym[sdf$b_psindx] = 0;
;  415          P 1145  2         gsd_sym[sdf$l_value] = .entity_number;
;  416          P 1146  2
;  417          P 1147  2         ! Now we can build the symbol name out of the
;  418          P 1148  2         ! prefix and the first four characters of the
;  419          P 1149  2         ! entity name.
;  420          P 1150  2
;  421          P 1151  2         gsd_sym[sdf$b_namlng] = .prefix[node_b_text_length] +
;  422          P 1152  2                                 minu(.entity[node_b_text_length],4);
;  423          P 1153  2         ch$copy(.prefix[node_b_text_length],prefix[node_t_text],
;  424          P 1154  2                 minu(.entity[node_b_text_length],4),entity[node_t_text],
;  425          P 1155  2                 ' ',.gsd_sym[sdf$b_namlng],gsd_sym[sdf$t_name]);
;  426          P 1156  2
;  427          P 1157  2         ! Write the record into the symbol table file.
;  428          P 1158  2         ! Any error is fatal.
;  429          P 1159  2
;  430          P 1160  2         stb_rab[rab$l_rbf] = gsd;
;  431          P 1161  2         stb_rab[rab$w_rsz] = 1 + 9 + 1+.gsd_sym[sdf$b_namlng];
;  432          P 1162  2         status = $put(rab=stb_rab);
;  433          P 1163  2         if not .status then
;  434          P 1164  2                 cdu$report_rms_error(msg(cdu$_writeerr),stb_rab);
;  435          P 1165  2                                 );
;  436          P 1166  2                         );
;  437          P 1167  2                 );
;  438            1168  2 );
;  439            1169  2
;  440            1170  2 return;
;  441            1171  2
;  442            1172  1 END;


                        0FFC 00000 WRITE_SYMBOL_RECORDS:
                                   .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11          ; 1087
5E      FEF0    CE  9E 00002       MOVAB   -272(SP), SP
50 00000000G    00  D0 00007       MOVL    CDU$GL_ROOT_NODE, R0                         ; 1168
5A        08    A0  D0 0000E       MOVL    8(R0), DEFINITION
                01  12 00012 1$:  BNEQ    2$
                    04 00014       RET
                0F  DD 00015 2$:   PUSHL   #15
```

```
                                   5A  DD 00017        PUSHL   DEFINITION                                   ;
               00000000G  00       02  FB 00019        CALLS   #2, CDU$LOOKUP_CHILD                         ;
                          59       50  D0 00020        MOVL    R0, PREFIX                                   ;
                                   16  13 00023        BEQL    5$                                           ;
                          06       6A  B1 00025        CMPW    (DEFINITION), #6                             ;
                                   06  12 00028        BNEQ    3$                                           ;
                    04    AE       18  D0 0002A        MOVL    #24, ENTITY_TYPE                             ;
                                   04  11 0002E        BRB     4$                                           ;
                    04    AE       10  D0 00030 3$:    MOVL    #16, ENTITY_TYPE                             ;
                              08   AE  D4 00034 4$:    CLRL    ENTITY_NUMBER                                ;
                          56  08   AA  D0 00037        MOVL    8(DEFINITION), ENTITY                        ;
                                   03  12 0003B 5$:    BNEQ    6$                                           ;
                                 0095  31 0003D        BRW     11$                                          ;
   04   AE        66      10       00  ED 00040 6$:    CMPZV   #0, #16, (ENTITY), ENTITY_TYPE               ;
                                   03  13 00046        BEQL    7$                                           ;
                                 0083  31 00048        BRW     10$                                          ;
                              08   AE  D6 0004B 7$:    INCL    ENTITY_NUMBER                                ;
                    10    AE    0101   8F  B0 0004E    MOVW    #257, GSD                                    ;
                              12   AE  94 00054        CLRB    GSD_SYM+1                                    ;
                    13    AE       02  B0 00057        MOVW    #2, GSD_SYM+2                                ;
                              15   AE  94 0005B        CLRB    GSD_SYM+4                                    ;
                    16    AE  08   AE  D0 0005E        MOVL    ENTITY_NUMBER, GSD_SYM+5                     ;
                          57  10   A6  9A 00063        MOVZBL  16(ENTITY), R7                               ;
                          04       57  91 00067        CMPB    R7, #4                                       ;
                                   03  1B 0006A        BLEQU   8$                                           ;
                          57       04  D0 0006C        MOVL    #4, R7                                       ;
          1A    AE        10  A9   57  81 0006F 8$:    ADDB3   R7, 16(PREFIX), GSD_SYM+9                    ;
                          0C  AE  10   A9  9A 00075    MOVZBL  16(PREFIX), 12(SP)                           ;
                          5B  1A   AE  9A 0007A        MOVZBL  GSD_SYM+9, R11                               ;
                          58  1B   AE  9E 0007E        MOVAB   GSD_SYM+10, R8                               ;
     5B           20      11  A9  0C   AE  2C 00082    MOVC5   12(SP), 17(PREFIX), #32, R11, (R8)           ;
                                   68     00089                                                             ;
                                   0F  18 0008A        BGEQ    9$                                           ;
                          58  0C   AE  C0 0008C        ADDL2   12(SP), R8                                   ;
                          5B  0C   AE  C2 00090        SUBL2   12(SP), R11                                  ;
     5B           20      11  A6   57  2C 00094        MOVC5   R7, 17(ENTITY), #32, R11, (R8)               ;
                                   68     0009A                                                             ;
                 0000'    CF  10   AE  9E 0009B 9$:    MOVAB   GSD, STB_RAB+40                              ;
                 0000'    CF  1A   AE  9B 000A1        MOVZBW  GSD_SYM+9, STB_RAB+34                        ;
                 0000'    CF       0B  A0 000A7        ADDW2   #11, STB_RAB+34                              ;
                              0000'    CF  9F 000AC    PUSHAB  STB_RAB                                      ;
               00000000G  00       01  FB 000B0        CALLS   #1, SYS$PUT                                  ;
                          6E       50  D0 000B7        MOVL    R0, STATUS                                   ;
                          11       6E  E8 000BA        BLBS    STATUS, 10$                                  ;
                              0000'    CF  9F 000BD    PUSHAB  STB_RAB                                      ;
                           001110D4    8F  DD 000C1    PUSHL   #1118420                                     ;
               00000000G  00       02  FB 000C7        CALLS   #2, CDU$REPORT_RMS_ERROR                     ;
                          56  04   A6  D0 000CE 10$:   MOVL    4(ENTITY), ENTITY                            ;
                                 FF66  31 000D2        BRW     5$                                           ;
                          5A  04   AA  D0 000D5 11$:   MOVL    4(DEFINITION), DEFINITION                    ;
                                 FF36  31 000D9        BRW     1$                                           ;
                                   04     000DC        RET                                                  ; 1172

; Routine Size:  221 bytes,    Routine Base:  $CODE$ + 01D7
```

```
;  444          1173  1 !++
;  445          1174  1 ! Description:  This routine is responsible for writing the end-of-module
;  446          1175  1 !               record at the end of the symbol table file.
;  447          1176  1 !
;  448          1177  1 ! Parameters:   None.
;  449          1178  1 !
;  450          1179  1 ! Returns:      Nothing.
;  451          1180  1 !
;  452          1181  1 ! Notes:
;  453          1182  1 !--
;  454          1183  1
;  455          1184  1 ROUTINE write_eom_record        : novalue
;  456          1185  2 = BEGIN
;  457          1186  2
;  458          1187  2 local
;  459          1188  2         status: long,
;  460          1189  2         eom: block[256,byte];
;  461          1190  2
;  462          1191  2 ! Format the end-of-module record.
;  463          1192  2
;  464          1193  2 eom[obj$b_rectyp] = obj$c_eom;
;  465          1194  2 eom[eom$b_comcod] = 0;
;  466          1195  2
;  467          1196  2 ! Write the record.  All errors are fatal.
;  468          1197  2
;  469          1198  2 stb_rab[rab$l_rbf] = eom;
;  470          1199  2 stb_rab[rab$w_rsz] = 2;
;  471          1200  2 status = $put(rab=stb_rab);
;  472          1201  2 if not .status then
;  473          1202  2         cdu$report_rms_error(msg(cdu$_writeerr),stb_rab);
;  474          1203  2
;  475          1204  2 return;
;  476          1205  2
;  477          1206  1 END;


                                        0004 00000 WRITE_EOM_RECORD:
                                                   .WORD   Save R2                                          ; 1184
                            52    0000'   CF  9E 00002          MOVAB   STB_RAB, R2                        ;
                            5E    FF00    CE  9E 00007          MOVAB   -256(SP), SP                       ;
                            6E            03  B0 0000C          MOVW    #3, EOM                            ; 1193
                       28   A2            6E  9E 0000F          MOVAB   EOM, STB_RAB+40                    ; 1198
                       22   A2            02  B0 00013          MOVW    #2, STB_RAB+34                     ; 1199
                                          52  DD 00017          PUSHL   R2                                 ; 1200
                  00000000G  00           01  FB 00019          CALLS   #1, SYS$PUT                        ;
                             0F           50  E8 00020          BLBS    STATUS, 1$                         ; 1201
                                          52  DD 00023          PUSHL   R2                                 ; 1202
                                001110D4  8F  DD 00025          PUSHL   #1118420                           ;
                  00000000G  00           02  FB 0002B          CALLS   #2, CDU$REPORT_RMS_ERROR           ;
                                          04  00032 1$:         RET                                        ; 1206

; Routine Size:  51 bytes,    Routine Base:  $CODE$ + 02B4
```

```
;  478          1207  1
;  479          1208  1 END
;  480          1209  0 ELUDOM


                                                        .EXTRN  LIB$SIGNAL

;                               PSECT SUMMARY

;       Name                    Bytes                   Attributes

;  $OWN$                        1020  NOVEC,  WPT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $PLIT$                         30  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $CODE$                        743  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)


;                           Library Statistics

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

;  _$255$DUA28:[SYSLIB]LIB.L32;1     18619        85         0      1000          00:01.9


;                                   COMMAND QUALIFIERS

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

; Size:          743 code + 1050 data bytes
; Run Time:        00:22.9
; Elapsed Time:    00:56.6
; Lines/CPU Min:    3167
; Lexemes/CPU-Min: 30372
; Memory Used:  203 pages
; Compilation Complete
```

SYMBOLS
LIS

NODES
LIS

OBJECT
LIS

PARSE1
LIS

PARSE3
LIS

ROUTINES
LIS

LISTING
LIS

MAIN
LIS

TABLE
LIS

PARSE2
LIS