```
LLL              IIIIIIIII   BBBBBBBBBBBB   RRRRRRRRRRRR        AAAAAAAAA     RRRRRRRRRRRR
LLL              IIIIIIIII   BBBBBBBBBBBB   RRRRRRRRRRRR        AAAAAAAAA     RRRRRRRRRRRR
LLL              IIIIIIIII   BBBBBBBBBBBB   RRRRRRRRRRRR        AAAAAAAAA     RRRRRRRRRRRR
LLL                 III      BBB       BBB  RRR         RRR  AAA         AAA  RRR         RRR
LLL                 III      BBB       BBB  RRR         RRR  AAA         AAA  RRR         RRR
LLL                 III      BBB       BBB  RRR         RRR  AAA         AAA  RRR         RRR
LLL                 III      BBB       BBB  RRR         RRR  AAA         AAA  RRR         RRR
LLL                 III      BBB       BBB  RRR         RRR  AAA         AAA  RRR         RRR
LLL                 III      BBB       BBB  RRR         RRR  AAA         AAA  RRR         RRR
LLL                 III      BBBBBBBBBBBB   RRRRRRRRRRRR     AAA         AAA  RRRRRRRRRRRR
LLL                 III      BBBBBBBBBBBB   RRRRRRRRRRRR     AAA         AAA  RRRRRRRRRRRR
LLL                 III      BBBBBBBBBBBB   RRRRRRRRRRRR     AAA         AAA  RRRRRRRRRRRR
LLL                 III      BBB       BBB  RRR    RRR       AAAAAAAAAAAAAAA  RRR    RRR
LLL                 III      BBB       BBB  RRR    RRR       AAAAAAAAAAAAAAA  RRR    RRR
LLL                 III      BBB       BBB  RRR    RRR       AAAAAAAAAAAAAAA  RRR    RRR
LLL                 III      BBB       BBB  RRR       RRR    AAA         AAA  RRR       RRR
LLL                 III      BBB       BBB  RRR       RRR    AAA         AAA  RRR       RRR
LLL                 III      BBB       BBB  RRR       RRR    AAA         AAA  RRR       RRR
LLLLLLLLLLLLLLL  IIIIIIIII   BBBBBBBBBBBB   RRR         RRR  AAA         AAA  RRR         RRR
LLLLLLLLLLLLLLL  IIIIIIIII   BBBBBBBBBBBB   RRR         RRR  AAA         AAA  RRR         RRR
LLLLLLLLLLLLLLL  IIIIIIIII   BBBBBBBBBBBB   RRR         RRR  AAA         AAA  RRR         RRR
```

```
_$2

Val
---
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
```

```
IIIIII    NN      NN  PPPPPPPP    UU      UU  TTTTTTTTTT  MM      MM    AAAAAA      CCCCCCCC
IIIIII    NN      NN  PPPPPPPP    UU      UU  TTTTTTTTTT  MM      MM    AAAAAA      CCCCCCCC
  II      NN      NN  PP      PP  UU      UU      TT      MMMM  MMMM  AA      AA  CC
  II      NN      NN  PP      PP  UU      UU      TT      MMMM  MMMM  AA      AA  CC
  II      NNNN    NN  PP      PP  UU      UU      TT      MM  MM  MM  AA      AA  CC
  II      NNNN    NN  PP      PP  UU      UU      TT      MM  MM  MM  AA      AA  CC
  II      NN  NN  NN  PPPPPPPP    UU      UU      TT      MM      MM  AA      AA  CC
  II      NN  NN  NN  PPPPPPPP    UU      UU      TT      MM      MM  AA      AA  CC
  II      NN    NNNN  PP          UU      UU      TT      MM      MM  AAAAAAAAAA  CC
  II      NN    NNNN  PP          UU      UU      TT      MM      MM  AAAAAAAAAA  CC
  II      NN      NN  PP          UU      UU      TT      MM      MM  AA      AA  CC                ....
  II      NN      NN  PP          UU      UU      TT      MM      MM  AA      AA  CC                ....
IIIIII    NN      NN  PP          UUUUUUUUUU      TT      MM      MM  AA      AA    CCCCCCCC      ....
IIIIII    NN      NN  PP          UUUUUUUUUU      TT      MM      MM  AA      AA    CCCCCCCC      ....

LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
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 lib_inputmac (                       ! Get next macro input line
;     2          0002  0                 LANGUAGE (BLISS32),
;     3          0003  0                 IDENT = 'V04-000'
;     4          0004  0                 ) =
;     5          0005  1 BEGIN
;     6          0006  1
;     7          0007  1 !
;     8          0008  1 !**********************************************************************************
;     9          0009  1 !*                                                                                *
;    10          0010  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                       *
;    11          0011  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                         *
;    12          0012  1 !*  ALL RIGHTS RESERVED.                                                           *
;    13          0013  1 !*                                                                                *
;    14          0014  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED         *
;    15          0015  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE        *
;    16          0016  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER        *
;    17          0017  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY        *
;    18          0018  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY        *
;    19          0019  1 !*  TRANSFERRED.                                                                   *
;    20          0020  1 !*                                                                                *
;    21          0021  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE        *
;    22          0022  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT        *
;    23          0023  1 !*  CORPORATION.                                                                   *
;    24          0024  1 !*                                                                                *
;    25          0025  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS        *
;    26          0026  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                        *
;    27          0027  1 !*                                                                                *
;    28          0028  1 !*                                                                                *
;    29          0029  1 !**********************************************************************************
;    30          0030  1
;    31          0031  1 !++
;    32          0032  1 !
;    33          0033  1 ! FACILITY:  Library command processor
;    34          0034  1 !
;    35          0035  1 ! ABSTRACT:
;    36          0036  1 !
;    37          0037  1 !       The VAX/VMS librarian is invoked by DCL to process the LIBRARY
;    38          0038  1 !       command.  It utilizes the librarian procedure set to perform
;    39          0039  1 !       the actual modifications to the library.
;    40          0040  1 !
;    41          0041  1 ! ENVIRONMENT:
;    42          0042  1 !
;    43          0043  1 !       VAX native, user mode.
;    44          0044  1 !
;    45          0045  1 !--
;    46          0046  1 !
;    47          0047  1 !
;    48          0048  1 ! AUTHOR:  Benn Schreiber,       CREATION DATE:  22-June-1979
;    49          0049  1 !
;    50          0050  1 ! MODIFIED BY:
;    51          0051  1 !
;    52          0052  1 !       V02-008         RPG44341        Bob Grosso      02-Mar-1982
;    53          0053  1 !               Fix routine scan_word to continue processing after
;    54          0054  1 !               a label is encountered, and correct the macro name
;    55          0055  1 !               printing on all the messages that get the macro name
;    56          0056  1 !               from macnamptrtbl.
;    57          0057  1 !
```

```
;    58          0058  1 !       V02-007         RPG0047         Bob Grosso      7-Aug-1981
;    59          0059  1 !                 lib$gl_ctlmsk now a quadword
;    60          0060  1 !
;    61          0061  1 !       V02-006         RPG0046         Bob Grosso      21-Jul-1981
;    62          0062  1 !                 Check macro level in setmacroname.
;    63          0063  1 !
;    64          0064  1 !       V02-005         RPG0036         Bob Grosso      25-Jun-1981
;    65          0065  1 !                 Continue inserting macros after an Lbr$_dupkey error.
;    66          0066  1 !
;    67          0067  1 !       V02-004         RPG0035         Bob Grosso      22-Apr-1981
;    68          0068  1 !                 Record module names for library update history
;    69          0069  1 !
;    70          0070  1 !       V02-003 BLS0029 Benn Schreiber          23-Dec-1980
;    71          0071  1 !                 Convert messages to message compiler
;    72          0072  1 !
;    73          0073  1 !       V02-002         Benn Schreiber          28-May-1980
;    74          0074  1 !                 Correct scan_word to not look past end of line.
;    75          0075  1 !--
;    76          0076  1
;    77          0077  1
```

```
;    79          0078  1 LIBRARY
;    80          0079  1         'SYS$LIBRARY:STARLET.L32';
;    81          0080  1 REQUIRE
;    82          0081  1         'PREFIX';
;    83          0265  1 REQUIRE
;    84          0266  1         'LIBDEF';
;    85          0554  1 REQUIRE
;    86          0555  1         'LBRDEF';
;    87          1146  1
;    88          1147  1 EXTERNAL ROUTINE
;    89          1148  1     get_record,                              !Read next input record
;    90          1149  1     lib_log_op,                              !Log insert operation
;    91          1150  1     lib_log_upd,                             !Log module names for Library History
;    92          1151  1     lib_get_zmem,                            !Allocate memory
;    93          1152  1     lib_free_mem,                            !Deallocate memory
;    94          1153  1     lbr$delete_key : ADDRESSING_MODE (GENERAL), !Delete key from index
;    95          1154  1     lbr$delete_data : ADDRESSING_MODE (GENERAL),!Delete data
;    96          1155  1     lbr$put_record : ADDRESSING_MODE (GENERAL), !Write record to library
;    97          1156  1     lbr$lookup_key : ADDRESSING_MODE (GENERAL), !Lookup key in library
;    98          1157  1     lbr$insert_key : ADDRESSING_MODE (GENERAL), !Insert key in library
;    99          1158  1     lbr$replace_key : ADDRESSING_MODE (GENERAL),!Replace or insert key
;   100          1159  1     lbr$put_end : ADDRESSING_MODE (GENERAL);    !Finish writing to library
;   101          1160  1
;   102          1161  1 EXTERNAL
;   103          1162  1     lbr$gl_rmsstv : ADDRESSING_MODE (GENERAL),  !RMS STV from librarian
;   104          1163  1     lib$gl_ctlmsk : BLOCK [2],
;   105          1164  1     lib$gl_keysize,                          !Max length of keys
;   106          1165  1     lib$gl_libfdb : REF BBLOCK,
;   107          1166  1     lib$gl_inpfdb : REF BBLOCK,
;   108          1167  1     lib$gl_libctl;
;   109          1168  1
;   110          1169  1 EXTERNAL LITERAL
;   111          1170  1     lib$_nomacfound,                         !No macro def found
;   112          1171  1     lib$_nestlevel,                          !Nesting level exceeded
;   113          1172  1     lib$_nomtchendr,                         !No matching .endr
;   114          1173  1     lib$_toomnyendr,                         !Too many .endr's
;   115          1174  1     lib$_inserterr,                          !Insert error
;   116          1175  1     lib$_deldaterr,                          !Delete data error
;   117          1176  1     lib$_endwrngmac,                         !Ends wrong macro
;   118          1177  1     lib$_replaced,                           !Module replaced
;   119          1178  1     lib$_inserted,                           !Module inserted
;   120          1179  1     lib$_nomtchendm,                         !No matching .endm
;   121          1180  1     lib$_macnamlng,                          !Macro name length illegal
;   122          1181  1     lib$_dupmodule,                          !Duplicate module
;   123          1182  1     lib$_dupmod;                             !Duplicate module
;   124          1183  1
;   125          1184  1 FORWARD ROUTINE
;   126          1185  1     setmacroname,
;   127          1186  1     checkendmac,
;   128          1187  1     scan_line,
;   129          1188  1     scan_word,
;   130          1189  1     skip_blanks,
;   131          1190  1     skip_blnk_bkwds,
;   132          1191  1     symbol_char,
;   133          1192  1     elim_trail_blnk,
;   134          1193  1     make_upper_case,
;   135          1194  1     lookup_keyword;
```

```
;  136          1195  1
;  137          1196  1 OWN
;  138          1197  1     bufdesc : BBLOCK [dsc$c_s_bln],          ! Descriptor for current line
;  139          1198  1     token1desc : BBLOCK [dsc$c_s_bln],       ! String descriptor for first token
;  140          1199  1     token2desc : BBLOCK [dsc$c_s_bln],       ! String descriptor for second token
;  141          1200  1     curchar,                                 ! Current character
;  142          1201  1     dupseen,                                 ! Flag set to skip duplicate module
;  143          1202  1     tokenindex,                              ! Index for first token
;  144          1203  1     lineptr,                                 ! Current line pointer
;  145          1204  1     endptr,                                  ! Pointer to past end of line
;  146          1205  1     nestinglevel,                            ! Current nesting level
;  147          1206  1     reptnestlevel,                           ! nesting level for .rept/.endr
;  148          1207  1     macrorfa : BBLOCK [rfa$c_length],        ! RFA of macro module header
;  149          1208  1     macnamptrtbl : REF BBLOCK,               ! Pointer to macro descriptor table
;  150          1209  1     macro_names : descriptor ('.MACRO'),
;  151          1210  1     repeat_name : descriptor ('.REPEAT'),
;  152          1211  1     rept_name   : descriptor ('.REPT'),
;  153          1212  1     irp_name    : descriptor ('.IRP'),
;  154          1213  1     irpc_name   : descriptor ('.IRPC'),
;  155          1214  1     ending_names: descriptor ('.ENDM'),
;  156          1215  1     endr_name   : descriptor ('.ENDR'),
;  157          1216  1     warn_name   : descriptor ('.WARN'),
;  158          1217  1     error_name  : descriptor ('.ERROR'),
;  159          1218  1     print_name  : descriptor ('.PRINT'),
;  160          1219  1     end_of_list : LONG INITIAL (0);
;  161          1220  1 LITERAL
;  162          1221  1     key_macro = 0,                           ! Must parallel order of ascii names above
;  163          1222  1     key_repeat = 1,
;  164          1223  1     key_rept = 2,
;  165          1224  1     key_irp = 3,
;  166          1225  1     key_irpc = 4,
;  167          1226  1     key_endm = 5,
;  168          1227  1     key_endr = 6,
;  169          1228  1     key_warn = 7,
;  170          1229  1     key_error = 8,
;  171          1230  1     key_print = 9,
;  172          1231  1     lib$c_maxnest = lbr$c_pagesize/dsc$c_s_bln; ! Max nesting level
;  173          1232  1
;  174          1233  1 BIND
;  175          1234  1     token1len = token1desc [dsc$w_length] : WORD,
;  176          1235  1     token1ptr = token1desc [dsc$a_pointer],
;  177          1236  1     token2len = token2desc [dsc$w_length] : WORD,
;  178          1237  1     token2ptr = token2desc [dsc$a_pointer],
;  179          1238  1     linelen = bufdesc [dsc$w_length] : WORD,
;  180          1239  1     lineaddr = bufdesc [dsc$a_pointer];
```

```
;  182          1240  1 GLOBAL ROUTINE lib_input_mac =
;  183          1241  2 BEGIN
;  184          1242  2 !
;  185          1243  2 ! This routine reads macro source files, extracts the macro definitions
;  186          1244  2 ! contained in them, and inserts them into the macro library.
;  187          1245  2 !
;  188          1246  2
;  189          1247  2 ROUTINE put_record (linedesc, rfa) =
;  190          1248  3 BEGIN
;  191          1249  3
;  192          1250  3 !++
;  193          1251  3 ! Local routine to call lbr$put_record
;  194          1252  3 !
;  195          1253  3 !--
;  196          1254  3
;  197          1255  3 IF NOT .dupseen
;  198          1256  3 THEN
;  199        P 1257  3     rms_perform (lbr$put_record (lib$gl_libctl, .linedesc, .rfa)
;  200          1258  3                  lib$_writeerr, .lbr$gl_rmsstv, 1, lib$gl_libfdb [fdb$l_namdesc]);
;  201          1259  3 RETURN true
;  202          1260  2 END;

                                                        .TITLE  LIB_INPUTMAC
                                                        .IDENT  \V04-000\

                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

                00  00  4F  52  43  41  4D  2E  00000 P.AAA:  .ASCII  \.MACRO\<0><0>                ;
                00  54  41  45  50  45  52  2E  00008 P.AAB:  .ASCII  \.REPEAT\<0>                 ;
                00  00  00  54  50  45  52  2E  00010 P.AAC:  .ASCII  \.REPT\<0><0><0>             ;
                                50  52  49  2E  00018 P.AAD:  .ASCII  \.IRP\                      ;
                00  00  00  43  50  52  49  2E  0001C P.AAE:  .ASCII  \.IRPC\<0><0><0>           ;
                00  00  00  4D  44  4E  45  2E  00024 P.AAF:  .ASCII  \.ENDM\<0><0><0>           ;
                00  00  00  52  44  4E  45  2E  0002C P.AAG:  .ASCII  \.ENDR\<0><0><0>           ;
                00  00  00  4E  52  41  57  2E  00034 P.AAH:  .ASCII  \.WARN\<0><0><0>           ;
                00  00  52  4F  52  52  45  2E  0003C P.AAI:  .ASCII  \.ERROR\<0><0>             ;
                00  00  54  4E  49  52  50  2E  00044 P.AAJ:  .ASCII  \.PRINT\<0><0>             ;

                                                        .PSECT  $OWN$,NOEXE,2

                                                00000 BUFDESC:.BLKB   8
                                                00008 TOKEN1DESC:
                                                              .BLKB   8
                                                00010 TOKEN2DESC:
                                                              .BLKB   8
                                                00018 CURCHAR:.BLKB   4
                                                0001C DUPSEEN:.BLKB   4
                                                00020 TOKENINDEX:
                                                              .BLKB   4
                                                00024 LINEPTR:.BLKB   4
                                                00028 ENDPTR: .BLKB   4
                                                0002C NESTINGLEVEL:
                                                              .BLKB   4
                                                00030 REPTNESTLEVEL:
                                                              .BLKB   4
                                                00034 MACRORFA:
```

```
                                .BLKB   6
                0003A           .BLKB   2
                0003C MACNAMPTRTBL:
                                .BLKB   4
00000006        00040 MACRO_NAMES:
                                .LONG   6
00000000'       00044           .ADDRESS P.AAA                          ;
00000007        00048 REPEAT_NAME:
                                .LONG   7
00000000'       0004C           .ADDRESS P.AAB                          ;
00000005        00050 REPT_NAME:
                                .LONG   5
00000000'       00054           .ADDRESS P.AAC                          ;
00000004        00058 IRP_NAME:
                                .LONG   4
00000000'       0005C           .ADDRESS P.AAD                          ;
00000005        00060 IRPC_NAME:
                                .LONG   5
00000000'       00064           .ADDRESS P.AAE                          ;
00000005        00068 ENDING_NAMES:
                                .LONG   5
00000000'       0006C           .ADDRESS P.AAF                          ;
00000005        00070 ENDR_NAME:
                                .LONG   5
00000000'       00074           .ADDRESS P.AAG                          ;
00000005        00078 WARN_NAME:
                                .LONG   5
00000000'       0007C           .ADDRESS P.AAH                          ;
00000006        00080 ERROR_NAME:
                                .LONG   6
00000000'       00084           .ADDRESS P.AAI                          ;
00000006        00088 PRINT_NAME:
                                .LONG   6
00000000'       0008C           .ADDRESS P.AAJ                          ;
00000000        00090 END_OF_LIST:
                                .LONG   0                               ;

                TOKEN1LEN=              TOKEN1DESC
                TOKEN1PTR=              TOKEN1DESC+4
                TOKEN2LEN=              TOKEN2DESC
                TOKEN2PTR=              TOKEN2DESC+4
                LINELEN=                BUFDESC
                LINEADDR=               BUFDESC+4
                        .EXTRN  GET_RECORD, LIB_LOG_OP
                        .EXTRN  LIB_LOG_UPD, LIB_GET_ZMEM
                        .EXTRN  LIB_FREE_MEM, LBR$DELETE_KEY
                        .EXTRN  LBR$DELETE_DATA
                        .EXTRN  LBR$PUT_RECORD, LBR$LOOKUP_KEY
                        .EXTRN  LBR$INSERT_KEY, LBR$REPLACE_KEY
                        .EXTRN  LBR$PUT_END, LBR$GL_RMSSTV
                        .EXTRN  LIB$GL_CTLMSK, LIB$GL_KEYSIZE
                        .EXTRN  LIB$GL_LIBFDB, LIB$GL_INPFDB
                        .EXTRN  LIB$GL_LIBCTL, LIB$_NOMACFOUND
                        .EXTRN  LIB$_NESTLEVEL, LIB$_NOMTCHENDR
                        .EXTRN  LIB$_TOOMNYENDR
                        .EXTRN  LIB$_INSERTERR, LIB$_DELDATERR
                        .EXTRN  LIB$_ENDWRNGMAC
```

```
                                        .EXTRN  LIB$_REPLACED, LIB$_INSERTED
                                        .EXTRN  LIB$_NOMTCHENDM
                                        .EXTRN  LIB$_MACNAMLNG, LIB$_DUPMODULE
                                        .EXTRN  LIB$_DUPMOD

                                        .PSECT  $CODE$,NOWRT,2

                                0000 00000 PUT_RECORD:
                                        .WORD   Save nothing                                    ; 1247
                          2F      0000' CF  E8 00002        BLBS    DUPSEEN, 1$               ; 1255
                          7E        04  AC  7D 00007        MOVQ    LINEDESC, -(SP)           ; 1258
                                  0000G CF  9F 0000B        PUSHAB  LIB$GL_LIBCTL             ;
              00000000G   00            03  FB 0000F        CALLS   #3, LBR$PUT_RECORD        ;
                          1D            50  E8 00016        BLBS    STATUS, 1$                ;
                             00000000G  00  DD 00019        PUSHL   LBR$GL_RMSSTV             ;
                                        50  DD 0001F        PUSHL   STATUS                    ;
                    7E      0000G CF        10  C1 00021    ADDL3   #16, LIB$GL_LIBFDB, -(SP) ;
                                        01  DD 00027        PUSHL   #1                        ;
                             008610D2   8F  DD 00029        PUSHL   #8786130                  ;
              00000000G   00            05  FB 0002F        CALLS   #5, LIB$SIGNAL            ;
                          50            01  D0 00036 1$:    MOVL    #1, R0                    ; 1259
                                        04  00039        RET                                   ; 1260

; Routine Size:  58 bytes,    Routine Base:  $CODE$ + 0000

;  203          1261  2
;  204          1262  2 ROUTINE put_end =
;  205          1263  3 BEGIN
;  206          1264  3
;  207          1265  3 !++
;  208          1266  3 !
;  209          1267  3 ! Write end of module record
;  210          1268  3 !
;  211          1269  3 !--
;  212          1270  3
;  213          1271  3 IF NOT .dupseen
;  214          1272  3 THEN
;  215        P 1273  3     rms_perform (lbr$put_end (lib$gl_libctl),
;  216          1274  3                  lib$_writeerr, .lbr$gl_rmsstv, 1, lib$gl_libfdb [fdb$l_namdesc]);
;  217          1275  3 RETURN true
;  218          1276  2 END;



                                0000 00000 PUT_END:.WORD    Save nothing                        ; 1262
                          2B      0000' CF  E8 00002        BLBS    DUPSEEN, 1$               ; 1271
                                  0000G CF  9F 00007        PUSHAB  LIB$GL_LIBCTL             ; 1274
              00000000G   00            01  FB 0000B        CALLS   #1, LBR$PUT_END           ;
                          1D            50  E8 00012        BLBS    STATUS, 1$                ;
                             00000000G  00  DD 00015        PUSHL   LBR$GL_RMSSTV             ;
                                        50  DD 0001B        PUSHL   STATUS                    ;
                    7E      0000G CF        10  C1 0001D    ADDL3   #16, LIB$GL_LIBFDB, -(SP) ;
                                        01  DD 00023        PUSHL   #1                        ;
                             008610D2   8F  DD 00025        PUSHL   #8786130                  ;
```

```
                     00000000G  00          05  FB 0002B          CALLS   #5, LIB$SIGNAL                                    ;
                                50          01  D0 00032 1$:      MOVL    #1, R0                                          ; 1275
                                            04 00035              RET                                                     ; 1276

; Routine Size:  54 bytes,    Routine Base:  $CODE$ + 003A

;  219          1277  2
;  220          1278  2 !
;  221          1279  2 ! Main body of lib_input_mac
;  222          1280  2 !
;  223          1281  2
;  224          1282  2 LOCAL
;  225          1283  2     deltxtrfa : BBLOCK [rfa$c_length],
;  226          1284  2     found_one,
;  227          1285  2     status,
;  228          1286  2     replacing,
;  229          1287  2     get_status,
;  230          1288  2     stop_flag;
;  231          1289  2 BIND
;  232          1290  2     libdesc = lib$gl_libfdb [fdb$l_namdesc] : BBLOCK,
;  233          1291  2     inpdesc = lib$gl_inpfdb [fdb$l_namdesc] : BBLOCK;
;  234          1292  2
;  235          1293  2 found_one = false;
;  236          1294  2 dupseen = false;
;  237          1295  2 CH$FILL (0, rfa$c_length, macrorfa);
;  238          1296  2 !
;  239          1297  2 ! Allocate macro name descriptor table if needed
;  240          1298  2 !
;  241          1299  2 IF .macnamptrtbl EQL 0
;  242          1300  2 THEN perform (lib_get_zmem (lbr$c_pagesize, macnamptrtbl));
;  243          1301  2 !
;  244          1302  2 ! Loop reading whole input file until end of file
;  245          1303  2 !
;  246          1304  2 WHILE true                                                    !Until eof
;  247          1305  3 DO BEGIN
;  248          1306  3 !
;  249          1307  3 ! Look for ".MACRO"
;  250          1308  3 !
;  251          1309  4 WHILE ((get_status = get_record (bufdesc)) NEQ rms$_eof) !Until .MACRO found
;  252          1310  4 DO BEGIN
;  253          1311  4     IF .linelen NEQ 0                                       !If non-null line
;  254          1312  4     AND scan_line ()                                        !and something interesting
;  255          1313  4     AND .tokenindex EQL key_macro                           ! and it is a .MACRO
;  256          1314  4         THEN EXITLOOP;
;  257          1315  3     END;
;  258          1316  3 IF .get_status EQL rms$_eof
;  259          1317  3 THEN IF .found_one
;  260          1318  3     THEN EXITLOOP
;  261          1319  4     ELSE BEGIN
;  262          1320  4         SIGNAL (lib$_nomacfound, 1, inpdesc);   !Otherwise done
;  263          1321  4         RETURN lib$_nomacfound;
;  264          1322  3         END;
;  265          1323  3
;  266          1324  3 replacing = false;                                         !Not replacing yet
;  267          1325  3 nestinglevel = 1;                                          !Nesting level initially 1
;  268          1326  3 reptnestlevel = 0;
```

```
;   269          1327  3 found_one = true;                                   !.MACRO has been found
;   270          1328  3 perform (setmacroname ());                         !Save the macro name away
;   271          1329  3 put_record (bufdesc, macrorfa);                    !Write the record
;   272          1330  3
;   273          1331  3 stop_flag = false;
;   274          1332  3 !
;   275          1333  3 ! Read and write records until the matching .ENDM is seen
;   276          1334  3 !
;   277          1335  4 DO BEGIN
;   278          1336  4     tokenindex = -1;
;   279          1337  4     get_status = get_record (bufdesc);
;   280          1338  4     IF .get_status EQL rms$_eof
;   281          1339  4         THEN EXITLOOP;
;   282          1340  4
;   283          1341  4     IF .linelen NEQ 0                                 !non-null line
;   284          1342  4     THEN IF scan_line ()                              !and something interesting there
;   285          1343  4     THEN CASE .tokenindex FROM key_macro TO key_print OF
;   286          1344  4         SET
;   287          1345  4
;   288          1346  4     [key_macro] :
;   289          1347  5                 BEGIN
;   290          1348  5                 nestinglevel = .nestinglevel + 1;
;   291          1349  5                 IF .nestinglevel GEQU lib$c_maxnest
;   292          1350  5                 THEN
;   293          1351  6                     BEGIN
;   294          1352  6                     BIND
;   295          1353  6                         macro_nam = .macnamptrtbl [dsc$a_pointer];      ! locates a counted ASCII string
;   296          1354  6                     SIGNAL (lib$_nestlevel, 2, macro_nam, inpdesc);
;   297          1355  6                     EXITLOOP;
;   298          1356  5                     END;
;   299          1357  5                 IF NOT setmacroname ()
;   300          1358  5                 THEN EXITLOOP;
;   301          1359  4                 END;
;   302          1360  4
;   303          1361  4     [key_repeat, key_rept, key_irp, key_irpc] :
;   304          1362  4                 reptnestlevel = .reptnestlevel + 1;
;   305          1363  4
;   306          1364  4     [key_endm] :
;   307          1365  5                 BEGIN
;   308          1366  5                 IF .token2len EQL 0                           !If no macro name
;   309          1367  5                   AND .reptnestlevel GTRU 0                   ! and still in a repeat
;   310          1368  5                 THEN reptnestlevel = .reptnestlevel -1        ! then assume its .endm on a repeat (should
;   311          1369  5                 ELSE
;   312          1370  6                     BEGIN
;   313          1371  6                     checkendmac ();
;   314          1372  6                     nestinglevel = .nestinglevel - 1;
;   315          1373  5                     END;
;   316          1374  4                 END;
;   317          1375  4
;   318          1376  4     [key_endr] :
;   319          1377  5                 BEGIN
;   320          1378  5                 reptnestlevel = .reptnestlevel - 1;
;   321          1379  4                 END;
;   322          1380  4
;   323          1381  4     [INRANGE] : true;
;   324          1382  4
;   325          1383  4         TES;
```

```
;  326     1384  4
;  327     1385  4     IF .nestinglevel EQL 0
;  328     1386  4     THEN
;  329     1387  5         BEGIN
;  330     1388  5         stop_flag = true;
;  331     1389  5         IF .reptnestlevel GTR 0
;  332     1390  5         THEN
;  333     1391  6             BEGIN
;  334     1392  6             BIND
;  335     1393  6                 macro_nam = .macnamptrtbl [dsc$a_pointer];      ! locates a counted ASCII string
;  336     1394  6             SIGNAL (lib$_nomtchendr, 3, .reptnestlevel, macro_nam, inpdesc)
;  337     1395  6             END
;  338     1396  5         ELSE
;  339     1397  5             IF .reptnestlevel LSS 0
;  340     1398  5             THEN
;  341     1399  6                 BEGIN
;  342     1400  6                 BIND
;  343     1401  6                     macro_nam = .macnamptrtbl [dsc$a_pointer];  ! locates a counted ASCII string
;  344     1402  6                 SIGNAL (lib$_toomnyendr, 2, macro_nam, inpdesc);
;  345     1403  5                 END;
;  346     1404  4         END;
;  347     1405  4 !
;  348     1406  4 ! Squeeze out trailing blanks and comments if /SQUEEZE and line is non-zero
;  349     1407  4 ! and the line is not .ERROR, .WARN or .PRINT (which contain semicolons
;  350     1408  4 ! as part of the syntax).
;  351     1409  4 !
;  352     1410  4     IF .lib$gl_ctlmsk [lib$v_squeeze]
;  353     1411  4     AND .linelen GTRU 0
;  354     1412  6     AND NOT ((.tokenindex GEQU key_warn)
;  355     1413  5                 AND (.tokenindex LEQU key_print))
;  356     1414  5         THEN BEGIN
;  357     1415  5             elim_trail_blnk ();
;  358     1416  5             IF .linelen NEQ 0                   !If line left after squeezing
;  359     1417  5                 THEN IF NOT put_record (bufdesc, macrorfa)
;  360     1418  5                         THEN EXITLOOP;
;  361     1419  5             END
;  362     1420  4         ELSE IF NOT put_record (bufdesc, macrorfa)
;  363     1421  4                 THEN EXITLOOP;
;  364     1422  4
;  365     1423  4     END
;  366     1424  3     UNTIL .stop_flag;
;  367     1425  3
;  368     1426  3 IF .stop_flag
;  369     1427  3 THEN
;  370     1428  4     BEGIN
;  371     1429  4     BIND
;  372     1430  4         macrodesc = .macnamptrtbl : BBLOCK;
;  373     1431  4     IF .dupseen
;  374     1432  4     THEN
;  375     1433  4         !
;  376     1434  4         !   If a duplicate was seen, then then skip the insert_key call
;  377     1435  4         !   and reset the dupseen flag.
;  378     1436  4         !
;  379     1437  4         dupseen = false
;  380     1438  4     ELSE
;  381     1439  5         BEGIN           ! proceed with normal insertion/replace
;  382     1440  5         put_end ();                             !Write end of module record
```

```
;  383          1441  5
;  384          1442  5          macrodesc [dsc$a_pointer] = .macrodesc [dsc$a_pointer] + 1;
;  385          1443  5          IF .lib$gl_ctlmsk [lib$v_replace]                              !If requested to replace
;  386          1444  5          THEN
;  387          1445  6              BEGIN
;  388          1446  6              replacing = lbr$lookup_key (lib$gl_libctl, .macnamptrtbl, deltxtrfa);
;  389        P 1447  6              rms_perform (lbr$replace_key (lib$gl_libctl, .macnamptrtbl, deltxtrfa, macrorfa),
;  390          1448  6                           lib$_inserterr, .lbr$gl_rmsstv, 2, .macnamptrtbl, libdesc);
;  391          1449  6              IF .replacing                                              !If we are replacing
;  392        P 1450  6                  THEN rms_perform (lbr$delete_data (lib$gl_libctl, deltxtrfa),   ! then delete old text
;  393          1451  6                                    lib$_deldaterr, .lbr$gl_rmsstv, 1, libdesc);
;  394          1452  6              END
;  395          1453  5          ELSE
;  396          1454  6              BEGIN
;  397        P 1455  6              rms_perform (lbr$insert_key (lib$gl_libctl, .macnamptrtbl, macrorfa),
;  398          1456  6                           lib$_inserterr, .lbr$gl_rmsstv, 2, .macnamptrtbl, libdesc );
;  399          1457  5              END;
;  400          1458  5
;  401          1459  6          lib_log_upd ( (IF .replacing THEN lhe$c_replaced
;  402          1460  5                         ELSE lhe$c_inserted), .macnamptrtbl );
;  403          1461  6          lib_log_op ((IF .replacing THEN lib$_replaced
;  404          1462  5                       ELSE lib$_inserted), .macnamptrtbl, .lib$gl_libfdb);
;  405          1463  4          END;
;  406          1464  4      macrodesc [dsc$a_pointer] = .macrodesc [dsc$a_pointer] - 1;
;  407          1465  4      END
;  408          1466  4
;  409          1467  3  ELSE
;  410          1468  4      BEGIN
;  411          1469  4      BIND
;  412          1470  4          macro_nam = .macnamptrtbl [dsc$a_pointer],      ! locates a counted ASCII string
;  413          1471  4          macrodesc = .macnamptrtbl : BBLOCK;
;  414          1472  4
;  415          1473  4      macrodesc [dsc$a_pointer] = .macrodesc [dsc$a_pointer] + 1;
;  416          1474  4      IF .nestinglevel GTRU 0
;  417          1475  4      THEN
;  418          1476  4          SIGNAL (lib$_nomtchendm, 2, macro_nam, libdesc);
;  419          1477  4      IF .macrorfa NEQ 0                                                 !Need to clean up?
;  420          1478  4      THEN
;  421          1479  5          BEGIN
;  422          1480  5          put_end ();                                                    !Write end of module
;  423          1481  5          lbr$delete_data (lib$gl_libctl, macrorfa);
;  424          1482  4          END;
;  425          1483  4      macrodesc [dsc$a_pointer] = .macrodesc [dsc$a_pointer] + 1;
;  426          1484  4      EXITLOOP;                                                          !And end this file now.
;  427          1485  3      END;
;  428          1486  3  CH$FILL (0, rfa$c_length, macrorfa);
;  429          1487  2  END;                                                                  !Of loop reading source
;  430          1488  2
;  431          1489  2  !
;  432          1490  2  ! Deallocate the macro name descriptor table
;  433          1491  2  !
;  434          1492  2  INCRU i FROM 0 TO lib$c_maxnest-1
;  435          1493  3  DO BEGIN
;  436          1494  3      BIND
;  437          1495  3          curdesc = macnamptrtbl [.i * dsc$c_s_bln,0,0,0] : BBLOCK [dsc$c_s_bln];
;  438          1496  3
;  439          1497  3      IF .curdesc [dsc$a_pointer] NEQ 0
```

```
;  440          1498  3     THEN lib_free_mem (lbr$c_maxkeylen+1, .curdesc [dsc$a_pointer]);
;  441          1499  2     END;
;  442          1500  2
;  443          1501  2 lib_free_mem (lbr$c_pagesize, .macnamptrtbl);
;  444          1502  2 macnamptrtbl = 0;
;  445          1503  2
;  446          1504  2 RETURN true
;  447          1505  1 END;                                !Of lib_inputmac


                                                  0FFC 00000          .ENTRY  LIB_INPUT_MAC, Save R2,R3,R4,R5,R6,R7,R8,-      ; 1240
                                                                              R9,R10,R11
                                                  08   C2 00002          SUBL2   #8, SP                                          ;
                                    57     0000G  CF   10   C1 00005          ADDL3   #16, LIB$GL_LIBFDB, R7                          ; 1290
                                    56     0000G  CF   10   C1 0000B          ADDL3   #16, LIB$GL_INPFDB, R6                          ; 1291
                                                  5B   D4 00011          CLRL    FOUND_ONE                                       ; 1293
                                                0000'  CF   D4 00013          CLRL    DUPSEEN                                         ; 1294
                              06                00     6E   00   2C 00017          MOVC5   #0, (SP), #0, #6, MACRORFA              ; 1295
                                                0000'  CF        0001C
                                                0000'  CF   D5 0001F          TSTL    MACNAMPTRTBL                                    ; 1299
                                                            11   12 00023          BNEQ    1$
                                                0000'  CF   9F 00025          PUSHAB  MACNAMPTRTBL                                    ; 1300
                                                       7E   0200   8F   3C 00029          MOVZWL  #512, -(SP)
                                                 0000G CF        02   FB 0002E          CALLS   #2, LIB_GET_ZMEM
                                                       60        50   E9 00033          BLBC    STATUS, 5$
                                                0000'  CF   9F 00036 1$:    PUSHAB  BUFDESC                                         ; 1309
                                                 0000G CF        01   FB 0003A          CALLS   #1, GET_RECORD
                                                       59        50   D0 0003F          MOVL    R0, GET_STATUS
                                            0001827A   8F             59   D1 00042          CMPL    GET_STATUS, #98938
                                                                       14   13 00049          BEQL    2$
                                                0000'  CF   B5 0004B          TSTW    LINELEN                                         ; 1311
                                                            E5   13 0004F          BEQL    1$
                                             0000V CF             00   FB 00051          CALLS   #0, SCAN_LINE                           ; 1312
                                                   DD             50   E9 00056          BLBC    R0, 1$
                                                0000'  CF   D5 00059          TSTL    TOKENINDEX                                      ; 1313
                                                            D7   12 0005D          BNEQ    1$
                                        0001827A   8F             59   D1 0005F 2$:    CMPL    GET_STATUS, #98938                      ; 1316
                                                                  1F   12 00066          BNEQ    4$
                                                   03             5B   E9 00068          BLBC    FOUND_ONE, 3$                           ; 1317
                                                                 02A1   31 0006B          BRW     37$
                                                                  56   DD 0006E 3$:    PUSHL   R6                                      ; 1320
                                                                  01   DD 00070          PUSHL   #1
                                                00000000G         8F   DD 00072          PUSHL   #LIB$_NOMACFOUND
                                    00000000G   00                03   FB 00078          CALLS   #3, LIB$SIGNAL
                                                   50 00000000G   8F   D0 0007F          MOVL    #LIB$_NOMACFOUND, R0            ; 1321
                                                                       04   00086          RET
                                                                  5A   D4 00087 4$:    CLRL    REPLACING                               ; 1324
                                             0000' CF             01   7D 00089          MOVQ    #1, NESTINGLEVEL                        ; 1325
                                                   5B             01   D0 0008E          MOVL    #1, FOUND_ONE                           ; 1327
                                             0000V CF             00   FB 00091          CALLS   #0, SETMACRONAME                        ; 1328
                                                   01             50   E8 00096 5$:    BLBS    STATUS, 6$
                                                                       04   00099          RET
                                                0000'  CF   9F 0009A 6$:    PUSHAB  MACRORFA                                        ; 1329
                                                0000'  CF   9F 0009E          PUSHAB  BUFDESC                                         ;
```

```
                    FEE9   CF              02  FB 000A2          CALLS   #2, PUT_RECORD                                   ;
                                            58  D4 000A7          CLRL    STOP_FLAG                                       ; 1331
                    0000'  CF               01  CE 000A9 7$:     MNEGL   #1, TOKENINDEX                                  ; 1336
                                      0000'  CF  9F 000AE          PUSHAB  BUFDESC                                         ; 1337
                    0000G  CF               01  FB 000B2          CALLS   #1, GET_RECORD                                  ;
                           59               50  D0 000B7          MOVL    R0, GET_STATUS                                  ;
                0001827A   8F               59  D1 000BA          CMPL    GET_STATUS, #98938                              ; 1338
                                            56  13 000C1          BEQL    11$                                             ;
                                      0000' CF  B5 000C3          TSTW    LINELEN                                         ; 1341
                                            74  13 000C7          BEQL    16$                                             ;
                    0000V  CF               00  FB 000C9          CALLS   #0, SCAN_LINE                                   ; 1342
                           6C               50  E9 000CE          BLBC    R0, 16$                                         ;
                        09              0000' CF  CF 000D1          CASEL   TOKENINDEX, #0, #9                              ; 1343
            0045            0045            0014      000D7 8$:     .WORD   9$-8$,-                                         ;
            0066            0062            0045      000DF                 12$-8$,-                                        ;
                            0066            0066      000E7                 12$-8$,-                                        ;
                                                                            12$-8$,-                                        ;
                                                                            12$-8$,-                                        ;
                                                                            13$-8$,-                                        ;
                                                                            15$-8$,-                                        ;
                                                                            16$-8$,-                                        ;
                                                                            16$-8$,-                                        ;
                                                                            16$-8$                                          ;
                                      0000' CF  D6 000EB 9$:     INCL    NESTINGLEVEL                                    ; 1348
                                   3F 0000' CF  D1 000EF          CMPL    NESTINGLEVEL, #63                               ; 1349
                                            1B  1B 000F4          BLEQU   10$                                             ;
                                   50 0000' CF  D0 000F6          MOVL    MACNAMPTRTBL, R0                                ; 1353
                                            56  DD 000FB          PUSHL   R6                                              ; 1354
                                         04 A0  DD 000FD          PUSHL   4(R0)                                           ;
                                            02  DD 00100          PUSHL   #2                                              ;
                                00000000G   8F  DD 00102          PUSHL   #LIB$_NESTLEVEL                                 ;
                00000000G       00              04  FB 00108      CALLS   #4, LIB$SIGNAL                                  ;
                                                08  11 0010F      BRB     11$                                             ; 1353
                    0000V  CF               00  FB 00111 10$:     CALLS   #0, SETMACRONAME                                ; 1357
                           24               50  E8 00116          BLBS    R0, 16$                                         ;
                                         00A4   31 00119 11$:     BRW     22$                                             ; 1358
                                      0000' CF  D6 0011C 12$:     INCL    REPTNESTLEVEL                                   ; 1362
                                            1B  11 00120          BRB     16$                                             ;
                                      0000' CF  B5 00122 13$:     TSTW    TOKEN2LEN                                       ; 1366
                                            06  12 00126          BNEQ    14$                                             ;
                                      0000' CF  D5 00128          TSTL    REPTNESTLEVEL                                   ; 1367
                                            0B  12 0012C          BNEQ    15$                                             ;
                    0000V  CF               00  FB 0012E 14$:     CALLS   #0, CHECKENDMAC                                 ; 1371
                                      0000' CF  D7 00133          DECL    NESTINGLEVEL                                    ; 1372
                                            04  11 00137          BRB     16$                                             ; 1343
                                      0000' CF  D7 00139 15$:     DECL    REPTNESTLEVEL                                   ; 1378
                                      0000' CF  D5 0013D 16$:     TSTL    NESTINGLEVEL                                    ; 1385
                                            42  12 00141          BNEQ    18$                                             ;
                                   58       01  D0 00143          MOVL    #1, STOP_FLAG                                   ; 1388
                                   51 0000' CF  D0 00146          MOVL    REPTNESTLEVEL, R1                               ; 1389
                                            1D  15 0014B          BLEQ    17$                                             ;
                                   50 0000' CF  D0 0014D          MOVL    MACNAMPTRTBL, R0                                ; 1393
                                            56  DD 00152          PUSHL   R6                                              ; 1394
                                         04 A0  DD 00154          PUSHL   4(R0)                                           ;
                                            51  DD 00157          PUSHL   R1                                              ;
                                            03  DD 00159          PUSHL   #3                                              ;
                                00000000G   8F  DD 0015B          PUSHL   #LIB$_NOMTCHENDR                                ;
```

```
                00000000G  00              05  FB 00161          CALLS   #5, LIB$SIGNAL                                ;
                                            1B  11 00168          BRB     18$                                         ;
                                            19  18 0016A 17$:     BGEQ    18$                                         ; 1397
                           50       0000'   CF  D0 0016C          MOVL    MACNAMPTRTBL, R0                            ; 1401
                                            56  DD 00171          PUSHL   R6                                          ; 1402
                                      04    A0  DD 00173          PUSHL   4(R0)                                       ;
                                            02  DD 00176          PUSHL   #2                                          ;
                                00000000G   8F  DD 00178          PUSHL   #LIB$_TOOMNYENDR                            ;
                00000000G  00               04  FB 0017E          CALLS   #4, LIB$SIGNAL                              ;
             1F      0000G  CF              03  E1 00185 18$:     BBC     #3, LIB$GL_CTLMSK+2, 20$                    ; 1410
                                 0000'      CF  B5 0018B          TSTW    LINELEN                                     ; 1411
                                            19  13 0018F          BEQL    20$                                         ;
                           07       0000'   CF  D1 00191          CMPL    TOKENINDEX, #7                              ; 1412
                                            07  1F 00196          BLSSU   19$                                         ;
                           09       0000'   CF  D1 00198          CMPL    TOKENINDEX, #9                              ; 1413
                                            0B  1B 0019D          BLEQU   20$                                         ;
                      0000V  CF             00  FB 0019F 19$:     CALLS   #0, ELIM_TRAIL_BLNK                         ; 1415
                                 0000'      CF  B5 001A4          TSTW    LINELEN                                     ; 1416
                                            10  13 001A8          BEQL    21$                                         ;
                                 0000'      CF  9F 001AA 20$:     PUSHAB  MACRORFA                                    ; 1420
                                 0000'      CF  9F 001AE          PUSHAB  BUFDESC                                     ;
                         FDD9    CF             02  FB 001B2          CALLS   #2, PUT_RECORD                              ;
                                 06             50  E9 001B7          BLBC    R0, 22$                                     ;
                                 03             58  E8 001BA 21$:     BLBS    STOP_FLAG, 22$                              ; 1424
                                           FEE9  31  001BD          BRW     7$                                          ;
                                 52        0000' CF  D0 001C0 22$:     MOVL    MACNAMPTRTBL, R2                            ; 1430
                                 03             58  E8 001C5          BLBS    STOP_FLAG, 23$                              ; 1426
                                          00FA   31  001C8          BRW     33$                                         ;
                                 07        0000' CF  E9 001CB 23$:     BLBC    DUPSEEN, 24$                                ; 1431
                                 0000'          CF  D4 001D0          CLRL    DUPSEEN                                     ; 1437
                                          00E9   31  001D4          BRW     32$                                         ;
                         FDEE    CF             00  FB 001D7 24$:     CALLS   #0, PUT_END                                 ; 1440
                                      04        A2  D6 001DC          INCL    4(R2)                                       ; 1442
                      78      0000G  CF             05  E1 001DF          BBC     #5, LIB$GL_CTLMSK+1, 26$                    ; 1443
                                            5E  DD 001E5          PUSHL   SP                                          ; 1446
                                 0000'      CF  DD 001E7          PUSHL   MACNAMPTRTBL                                ;
                                 0000G      CF  9F 001EB          PUSHAB  LIB$GL_LIBCTL                               ;
                00000000G  00               03  FB 001EF          CALLS   #3, LBR$LOOKUP_KEY                          ;
                           5A               50  D0 001F6          MOVL    R0, REPLACING                               ;
                                 0000'      CF  9F 001F9          PUSHAB  MACRORFA                                    ; 1448
                                      04    AE  9F 001FD          PUSHAB  DELTXTRFA                                   ;
                                 0000'      CF  DD 00200          PUSHL   MACNAMPTRTBL                                ;
                                 0000G      CF  9F 00204          PUSHAB  LIB$GL_LIBCTL                               ;
                00000000G  00               04  FB 00208          CALLS   #4, LBR$REPLACE_KEY                         ;
                           1D               50  E8 0020F          BLBS    STATUS, 25$                                 ;
                                00000000G   00  DD 00212          PUSHL   LBR$GL_RMSSTV                               ;
                                            50  DD 00218          PUSHL   STATUS                                      ;
                                            57  DD 0021A          PUSHL   R7                                          ;
                                 0000'      CF  DD 0021C          PUSHL   MACNAMPTRTBL                                ;
                                            02  DD 00220          PUSHL   #2                                          ;
                                00000000G   8F  DD 00222          PUSHL   #LIB$_INSERTERR                             ;
                00000000G  00               06  FB 00228          CALLS   #6, LIB$SIGNAL                              ;
                           5E               5A  E9 0022F 25$:     BLBC    REPLACING, 27$                              ; 1449
                                            5E  DD 00232          PUSHL   SP                                          ; 1451
                                 0000G      CF  9F 00234          PUSHAB  LIB$GL_LIBCTL                               ;
                00000000G  00               02  FB 00238          CALLS   #2, LBR$DELETE_DATA                         ;
                           4E               50  E8 0023F          BLBS    STATUS, 27$                                 ;
```

```
                    00000000G  00   DD 00242          PUSHL   LBR$GL_RMSSTV                                ;
                               50   DD 00248          PUSHL   STATUS                                      ;
                               57   DD 0024A          PUSHL   R7                                          ;
                               01   DD 0024C          PUSHL   #1                                          ;
                    00000000G  8F   DD 0024E          PUSHL   #LIB$_DELDATERR                             ;
00000000G  00                  05   FB 00254          CALLS   #5, LIB$SIGNAL                              ;
                               33   11 0025B          BRB     27$                                         ; 1443
                      0000'    CF   9F 0025D 26$:     PUSHAB  MACRORFA                                    ; 1456
                      0000'    CF   DD 00261          PUSHL   MACNAMPTRTBL                                ;
                      0000G    CF   9F 00265          PUSHAB  LIB$GL_LIBCTL                               ;
00000000G  00                  03   FB 00269          CALLS   #3, LBR$INSERT_KEY                          ;
           1D                  50   E8 00270          BLBS    STATUS, 27$                                 ;
                    00000000G  00   DD 00273          PUSHL   LBR$GL_RMSSTV                               ;
                               50   DD 00279          PUSHL   STATUS                                      ;
                               57   DD 0027B          PUSHL   R7                                          ;
                      0000'    CF   DD 0027D          PUSHL   MACNAMPTRTBL                                ;
                               02   DD 00281          PUSHL   #2                                          ;
                    00000000G  8F   DD 00283          PUSHL   #LIB$_INSERTERR                             ;
00000000G  00                  06   FB 00289          CALLS   #6, LIB$SIGNAL                              ;
                      0000'    CF   DD 00290 27$:     PUSHL   MACNAMPTRTBL                                ; 1460
           04                  5A   E9 00294          BLBC    REPLACING, 28$                              ; 1459
                               03   DD 00297          PUSHL   #3                                          ;
                               02   11 00299          BRB     29$                                         ;
                               02   DD 0029B 28$:     PUSHL   #2                                          ;
     0000G CF                  02   FB 0029D 29$:     CALLS   #2, LIB_LOG_UPD                             ;
                      0000G    CF   DD 002A2          PUSHL   LIB$GL_LIBFDB                               ; 1462
                      0000'    CF   DD 002A6          PUSHL   MACNAMPTRTBL                                ;
           08                  5A   E9 002AA          BLBC    REPLACING, 30$                              ;
                    00000000G  8F   DD 002AD          PUSHL   #LIB$_REPLACED                              ; 1461
                               06   11 002B3          BRB     31$                                         ;
                    00000000G  8F   DD 002B5 30$:     PUSHL   #LIB$_INSERTED                              ;
     0000G CF                  03   FB 002BB 31$:     CALLS   #3, LIB_LOG_OP                              ;
                         04    A2   D7 002C0 32$:     DECL    4(R2)                                       ; 1464
                               3F   11 002C3          BRB     36$                                         ; 1426
                   50    04    A2   D0 002C5 33$:     MOVL    4(R2), R0                                   ; 1470
                         04    A2   D6 002C9          INCL    4(R2)                                       ; 1473
                      0000'    CF   D5 002CC          TSTL    NESTINGLEVEL                                ; 1474
                               13   13 002D0          BEQL    34$                                         ;
                      0081     8F   BB 002D2          PUSHR   #^M<R0,R7>                                  ; 1476
                               02   DD 002D6          PUSHL   #2                                          ;
                    00000000G  8F   DD 002D8          PUSHL   #LIB$_NOMTCHENDM                            ;
                00000000G  00       04   FB 002DE     CALLS   #4, LIB$SIGNAL                              ;
                      0000'    CF   D5 002E5 34$:     TSTL    MACRORFA                                    ; 1477
                               14   13 002E9          BEQL    35$                                         ;
                   FCDA      CF   00   FB 002EB      CALLS   #0, PUT_END                                 ; 1480
                      0000'    CF   9F 002F0          PUSHAB  MACRORFA                                    ; 1481
                      0000G    CF   9F 002F4          PUSHAB  LIB$GL_LIBCTL                               ;
                00000000G  00       02   FB 002F8     CALLS   #2, LBR$DELETE_DATA                         ;
                         04    A2   D6 002FF 35$:     INCL    4(R2)                                       ; 1483
                               0B   11 00302          BRB     37$                                         ; 1471
06              00         6E   00   2C 00304 36$:     MOVC5   #0, (SP), #0, #6, MACRORFA                  ; 1486
                      0000'    CF   00309                                                                 ;
                         FD27      31   0030C          BRW     1$                                          ; 1304
                               52   D4 0030F 37$:     CLRL    I                                           ; 1492
                   50  0000'DF42   7E 00311 38$:     MOVAQ   @MACNAMPTRTBL[I], R0                        ; 1495
                         04    A0   D5 00317          TSTL    4(R0)                                       ; 1497
                               0C   13 0031A          BEQL    39$                                         ;
```

```
                04   A0  DD 0031C          PUSHL   4(R0)                                   ; 1498
             7E 81   8F  9A 0031F          MOVZBL  #129, -(SP)                             ;
      0000G  CF      02  FB 00323          CALLS   #2, LIB_FREE_MEM                        ;
                     52  D6 00328 39$:     INCL    I                                       ; 1492
             3F      52  D1 0032A          CMPL    I, #63                                  ;
                     E2  1B 0032D          BLEQU   38$                                     ;
           0000'     CF  DD 0032F          PUSHL   MACNAMPTRTBL                            ; 1501
             7E 0200 8F  3C 00333          MOVZWL  #512, -(SP)                             ;
      0000G  CF      02  FB 00338          CALLS   #2, LIB_FREE_MEM                        ;
           0000'     CF  D4 0033D          CLRL    MACNAMPTRTBL                            ; 1502
             50      01  D0 00341          MOVL    #1, R0                                  ; 1504
                     04  00344          RET                                             ; 1505

; Routine Size:  837 bytes,    Routine Base:  $CODE$ + 0070
```

```
;  449          1506  1 ROUTINE scan_line =
;  450          1507  2 BEGIN
;  451          1508  2 !
;  452          1509  2 ! This routine scans the line and determines if the line contains any
;  453          1510  2 ! significant keyword and, if so, also attempts to scan the macro name,
;  454          1511  2 ! if any.
;  455          1512  2
;  456          1513  2 ROUTINE do_scan_line =
;  457          1514  3 BEGIN
;  458          1515  3 LOCAL
;  459          1516  3         lastchar;                                   ! Last character
;  460          1517  3
;  461          1518  3 IF NOT skip_blanks () THEN RETURN false;           !If line all blank or comment, done
;  462          1519  3 token1ptr = .lineptr;                              !Point to start of first token
;  463          1520  3 token1len = scan_word ();                          !Scan to end of word and get length
;  464          1521  3 IF .token1len EQL 0 THEN RETURN false;             !If no word, return false
;  465          1522  3 lastchar = .curchar;                              !Remember the character past
;  466          1523  3                                                   ! call to skip_blanks
;  467          1524  3 IF skip_blanks ()                                  !If not end of line now,
;  468          1525  3 THEN
;  469          1526  4     BEGIN
;  470          1527  4     IF .lastchar EQL %ASCII':'                    ! but on a label
;  471          1528  4     THEN
;  472          1529  5         BEGIN
;  473          1530  5         lineptr = .lineptr - 1;                   ! back up one because subsequent call to skip_blanks will sw
;  474          1531  5         RETURN do_scan_line ();                   ! then rescan what is left
;  475          1532  5         END
;  476          1533  4     ELSE
;  477          1534  4         IF .curchar EQL %ASCII'='                 !If an assignment
;  478          1535  4         THEN RETURN false;                        ! then all done
;  479          1536  4     END
;  480          1537  3 ELSE RETURN lookup_keyword (token1desc, macro_names); !Nothing left
;  481          1538  3                                                   ! on line, see if .endm/.endr
;  482          1539  3 token2ptr = .lineptr;
;  483          1540  3 token2len = scan_word ();
;  484          1541  3 RETURN lookup_keyword (token1desc, macro_names); !Lookup name and return
;  485          1542  2 END;                                !Of do_scan_line


                                000C 00000 DO_SCAN_LINE:
                                           .WORD   Save R2,R3                                  ; 1513
                    53      0000'  CF  9E 00002    MOVAB   LINEPTR, R3                         ;
          0000V     CF             00  FB 00007    CALLS   #0, SKIP_BLANKS                     ; 1518
                    46             50  E9 0000C    BLBC    R0, 3$                              ;
            E8      A3             63  D0 0000F    MOVL    LINEPTR, TOKEN1PTR                  ; 1519
          0000V     CF             00  FB 00013    CALLS   #0, SCAN_WORD                       ; 1520
            E4      A3             50  B0 00018    MOVW    R0, TOKEN1LEN                       ;
                                   37  13 0001C    BEQL    3$                                  ; 1521
                    52        F4   A3  D0 0001E    MOVL    CURCHAR, LASTCHAR                   ; 1522
          0000V     CF             00  FB 00022    CALLS   #0, SKIP_BLANKS                     ; 1524
                    1F             50  E9 00027    BLBC    R0, 2$                              ;
                    3A             52  D1 0002A    CMPL    LASTCHAR, #58                       ; 1527
                                   07  12 0002D    BNEQ    1$                                  ;
                                   63  D7 0002F    DECL    LINEPTR                             ; 1530
```

```
                CB   AF               00  FB 00031        CALLS   #0, DO_SCAN_LINE                      ; 1531
                                      04  00035           RET                                           ;
                     3D        F4     A3  D1 00036 1$:    CMPL    CURCHAR, #61                          ; 1534
                                      19  13 0003A        BEQL    3$                                    ;
                FO   A3               63  D0 0003C        MOVL    LINEPTR, TOKEN2PTR                    ; 1539
             0000V   CF               00  FB 00040        CALLS   #0, SCAN_WORD                         ; 1540
                EC   A3               50  B0 00045        MOVW    R0, TOKEN2LEN                         ;
                               1C     A3  9F 00049 2$:    PUSHAB  MACRO_NAMES                           ; 1541
                               E4     A3  9F 0004C        PUSHAB  TOKENTDESC                            ;
             0000V   CF               02  FB 0004F        CALLS   #2, LOOKUP_KEYWORD                    ;
                                      04  00054           RET                                           ;
                                      50  D4 00055 3$:    CLRL    R0                                    ; 1542
                                      04  00057           RET                                           ;

; Routine Size:  88 bytes,    Routine Base:  $CODE$ + 03B5

;  486          1543  2 !
;  487          1544  2 ! Main body of scan_line
;  488          1545  2 !
;  489          1546  2 lineptr = .lineaddr - 1;                        !Init moving line pointer
;  490          1547  2 endptr = .lineaddr + .linelen;
;  491          1548  2 token1len = 0;
;  492          1549  2 token2len = 0;
;  493          1550  2 RETURN do_scan_line ()
;  494          1551  1 END;                           !Of scan_line


                                   0004 00000 SCAN_LINE:
                                                          .WORD   Save R2                               ; 1506
                           52   0000'     CF  9E 00002    MOVAB   LINEADDR, R2                          ;
             20   A2       62             01  C3 00007    SUBL3   #1, LINEADDR, LINEPTR                 ; 1546
                           50        FC   A2  3C 0000C    MOVZWL  LINELEN, R0                           ; 1547
             24   A2       62             50  C1 00010    ADDL3   R0, LINEADDR, ENDPTR                  ;
                                     04   A2  B4 00015    CLRW    TOKEN1LEN                             ; 1548
                                     0C   A2  B4 00018    CLRW    TOKEN2LEN                             ; 1549
                      89   AF             00  FB 0001B    CALLS   #0, DO_SCAN_LINE                      ; 1550
                                          04  0001F       RET                                           ; 1551

; Routine Size:  32 bytes,    Routine Base:  $CODE$ + 040D
```

```
;  496          1552  1 ROUTINE scan_word =
;  497          1553  2 BEGIN
;  498          1554  2 !
;  499          1555  2 ! This routine returns the length of the word which is pointed to currently
;  500          1556  2 ! by lineptr and advances lineptr to the character past the end of the word.
;  501          1557  2 !
;  502          1558  2 LOCAL
;  503          1559  2     startptr;
;  504          1560  2
;  505          1561  2 startptr = .lineptr;
;  506          1562  2 WHILE CH$DIFF (.endptr, .lineptr + 1) GTR 0
;  507          1563  3 DO BEGIN
;  508          1564  3     curchar = CH$A_RCHAR (lineptr);              !next character
;  509          1565  3     IF NOT symbol_char () THEN RETURN .lineptr - .startptr;
;  510          1566  2     END;
;  511          1567  2 RETURN .lineptr + 1 - .startptr;
;  512          1568  1 END;


                                        000C 00000 SCAN_WORD:
                                                   .WORD   Save R2,R3                          ; 1552
                    53     0000'  CF  9E 00002             MOVAB   LINEPTR, R3                 ;
                    52            63  D0 00007             MOVL    LINEPTR, STARTPTR           ; 1561
                    50            63  01  C1 0000A 1$:     ADDL3   #1, LINEPTR, R0             ; 1562
                    50       04   A3  D1 0000E             CMPL    ENDPTR, R0                  ;
                                  14  15 00012             BLEQ    2$                          ;
                                  63  D6 00014             INCL    LINEPTR                     ; 1564
                          F4      A3      00  B3  9A 00016 MOVZBL  @LINEPTR, CURCHAR           ;
                        0000V     CF          00  FB 0001B CALLS   #0, SYMBOL_CHAR             ; 1565
                                  E7          50  E8 00020 BLBS    R0, 1$                      ;
                50                63          52  C3 00023 SUBL3   STARTPTR, LINEPTR, R0       ;
                                              04  00027     RET                                 ;
                50                63          52  C3 00028 2$:    SUBL3   STARTPTR, LINEPTR, R0 ; 1567
                                              50  D6 0002C     INCL    R0                      ;
                                              04  0002E     RET                                 ; 1568

; Routine Size:  47 bytes,    Routine Base:  $CODE$ + 042D
```

```
;  514          1569  1 ROUTINE skip_blanks =
;  515          1570  2 BEGIN
;  516          1571  2 !
;  517          1572  2 ! This routine skips blanks and tabs in the input line.
;  518          1573  2 ! Returns true if skipped to non-blank, non-tab character.
;  519          1574  2 ! Returns false if skipped to semi-colon or end-of-line.
;  520          1575  2 !
;  521          1576  2 WHILE CH$DIFF (.endptr, .lineptr + 1) GTR 0              ! More input line?
;  522          1577  3 DO BEGIN
;  523          1578  3     curchar = CH$A_RCHAR (lineptr);                    ! Read next character
;  524          1579  3     IF .curchar EQL %ASCII';' THEN RETURN false;        !Return false if comment
;  525          1580  3     IF .curchar NEQ %ASCII' ' AND .curchar NEQ %ASCII'	' !If character is not space/tab
;  526          1581  3       THEN RETURN true;
;  527          1582  2     END;
;  528          1583  2 RETURN false                                           !Return false for end of line
;  529          1584  1 END;                                 !Of skip_blanks


                                0004 00000 SKIP_BLANKS:
                                                .WORD   Save R2                                 ; 1569
                     52  0000'   CF  9E 00002         MOVAB   LINEPTR, R2
                     51          62  D0 00007         MOVL    LINEPTR, R1                        ; 1576
                     50    01    A1  9E 0000A 1$:     MOVAB   1(R1), R0
                     50    04    A2  D1 0000E         CMPL    ENDPTR, R0
                                 20  15 00012         BLEQ    2$
                                 62  D6 00014         INCL    LINEPTR                            ; 1578
                     51          62  D0 00016         MOVL    LINEPTR, R1
                F4   A2          61  9A 00019         MOVZBL  (R1), CURCHAR
                     50    F4    A2  D0 0001D         MOVL    CURCHAR, R0                        ; 1579
                     3B          50  D1 00021         CMPL    R0, #59
                                 0E  13 00024         BEQL    2$
                     20          50  D1 00026         CMPL    R0, #32                            ; 1580
                                 DF  13 00029         BEQL    1$
                     09          50  D1 0002B         CMPL    R0, #9
                                 DA  13 0002E         BEQL    1$
                     50          01  D0 00030         MOVL    #1, R0                             ; 1581
                                     04 00033         RET
                                 50  D4 00034 2$:     CLRL    R0                                 ; 1584
                                     04 00036         RET

; Routine Size:  55 bytes,    Routine Base:  $CODE$ + 045C
```

```
;  531          1585  1 ROUTINE symbol_char =
;  532          1586  2 BEGIN
;  533          1587  2 !
;  534          1588  2 ! This routine returns true if curchar is a character that may be
;  535          1589  2 ! in a symbol, and false if not.
;  536          1590  2 !
;  537          1591  2 OWN
;  538          1592  2     symbolics : BBLOCK [68] INITIAL                          !68 to pad to full word
;  539          1593  2                 ('ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789.$_');
;  540          1594  2
;  541          1595  2 IF CH$FAIL (CH$FIND_CH (65, symbolics, .curchar))
;  542          1596  2 THEN RETURN false
;  543          1597  2 ELSE RETURN true
;  544          1598  1 END;                                 !Of symbol_char


                                                            .PSECT  $OWN$,NOEXE,2

4F  4E  4D  4C  4B  4A  49  48  47  46  45  44  43  42  41  00094 SYMBOLICS:
                                                            .ASCII  \ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmn\  ;
64  63  62  61  5A  59  58  57  56  55  54  53  52  51  50  000A3
                    6E  6D  6C  6B  6A  69  68  67  66  65  000B2
32  31  30  7A  79  78  77  76  75  74  73  72  71  70  6F  000BC  .ASCII  \opqrstuvwxyz0123456789.$_\<0><0><0>          ;
        00  00  00  5F  24  2E  39  38  37  36  35  34  33  000CB                                                       ;


                                                            .PSECT  $CODE$,NOWRT,2

                                                0000 00000 SYMBOL_CHAR:
                                                            .WORD   Save nothing                                    ; 1585
            0000'  CF       0041    8F      0000'  CF  3A 00002     LOCC    CURCHAR, #65, SYMBOLICS                 ; 1595
                                                   02  12 0000C     BNEQ    1$                                      ;
                                                   51  D4 0000E     CLRL    R1                                      ;
                                                   51  D5 00010 1$: TSTL    R1                                      ;
                                                   03  12 00012     BNEQ    2$                                      ;
                                                   50  D4 00014     CLRL    R0                                      ; 1597
                                                       04 00016     RET                                             ;
                                        50              01  D0 00017 2$: MOVL    #1, R0                              ;
                                                       04 0001A     RET                                             ; 1598

; Routine Size:  27 bytes,    Routine Base:  $CODE$ + 0493
```

```
;  546          1599  1 ROUTINE lookup_keyword (tokendesc, tableaddr) =
;  547          1600  2 BEGIN
;  548          1601  2 !
;  549          1602  2 ! This routine looks up the token described by tokenptr and tokenlen
;  550          1603  2 ! in the vector of string descriptors pointed to by tableaddr.
;  551          1604  2 !
;  552          1605  2 ! Returns true with tokenindex set up if found, false if not.
;  553          1606  2 !
;  554          1607  2 MAP
;  555          1608  2     tokendesc : REF BBLOCK,
;  556          1609  2     tableaddr : REF BBLOCK;
;  557          1610  2
;  558          1611  2 LOCAL
;  559          1612  2     upcasename : VECTOR [lbr$c_pagesize,BYTE],
;  560          1613  2     i;
;  561          1614  2
;  562          1615  2 IF .tokendesc [dsc$w_length] EQL 0 THEN RETURN false;
;  563          1616  2 make_upper_case (.tokendesc, upcasename);                  !upper case the name
;  564          1617  2 i = 0;
;  565          1618  2 WHILE .tableaddr [.i * dsc$c_s_bln,0,16,0] NEQ 0
;  566          1619  3 DO BEGIN
;  567          1620  3     BIND
;  568          1621  3         curdesc = tableaddr [.i * dsc$c_s_bln,0,0,0] : BBLOCK [dsc$c_s_bln];
;  569          1622  3
;  570          1623  3     IF CH$EQL (.tokendesc [dsc$w_length], upcasename, .curdesc [dsc$w_length],
;  571          1624  3                .curdesc [dsc$a_pointer])
;  572          1625  4         THEN BEGIN
;  573          1626  4             tokenindex = .i;
;  574          1627  4             RETURN true;
;  575          1628  4             END
;  576          1629  3         ELSE i = .i + 1;
;  577          1630  2     END;
;  578          1631  2 RETURN false                                    !Not found
;  579          1632  1 END;                                  !Of lookup_keyword



                                        003C 00000 LOOKUP_KEYWORD:
                                                   .WORD   Save R2,R3,R4,R5                    ; 1599
                                5E      FE00   CE  9E 00002         MOVAB   -512(SP), SP
                                55        04   AC  D0 00007         MOVL    TOKENDESC, R5      ; 1615
                                               65  B5 0000B         TSTW    (R5)
                                               2A  13 0000D         BEQL    3$
                                          4020 8F  BB 0000F         PUSHR   #^M<R5,SP>         ; 1616
                                0000V   CF     02  FB 00013         CALLS   #2, MAKE_UPPER_CASE
                                               54  D4 00018         CLRL    I                  ; 1617
                                50        08 BC44  7E 0001A 1$:     MOVAQ   @TABLEADDR[I], R0  ; 1618
                                               60  B5 0001F         TSTW    (R0)
                                               16  13 00021         BEQL    3$
        60              00              6E     65  2D 00023         CMPC5   (R5), UPCASENAME, #0, (R0), @4(R0) ; 1623
                                          04   B0      00028
                                               09  12 0002A         BNEQ    2$
                                0000'   CF     54  D0 0002C         MOVL    I, TOKENINDEX      ; 1626
                                50             01  D0 00031         MOVL    #1, R0             ; 1627
                                                   04 00034         RET
```

```
                54  D6 00035 2$:    INCL    I                                       ; 1629
                E1  11 00037         BRB     1$                                      ; 1618
                50  D4 00039 3$:    CLRL    R0                                      ; 1632
                    04 0003B         RET                                             ;

; Routine Size:  60 bytes,    Routine Base:  $CODE$ + 04AE
```

```
;  581          1633  1 ROUTINE make_upper_case (idesc, oname) =
;  582          1634  2 BEGIN
;  583          1635  2 !
;  584          1636  2 ! This routine upper cases iname.
;  585          1637  2 !
;  586          1638  2 MAP
;  587          1639  2     idesc : REF BBLOCK,
;  588          1640  2     oname : REF VECTOR [,BYTE];
;  589          1641  2 BIND
;  590          1642  2     namlen = idesc [dsc$w_length] : WORD,
;  591          1643  2     iname = idesc [dsc$a_pointer] : REF VECTOR [,BYTE];
;  592          1644  2
;  593          1645  2 IF .namlen GTRU 0
;  594          1646  2 THEN INCRU i FROM 0 TO .namlen-1
;  595          1647  2 DO IF .iname [.i] GEQU %ASCII'a'                    !copy name and convert to upper case
;  596          1648  2    AND .iname [.i] LEQU %ASCII'z'
;  597          1649  3    THEN oname [.i] = .iname [.i] - (%ASCII'a' - %ASCII'A')
;  598          1650  2    ELSE oname [.i] = .iname [.i];
;  599          1651  2 RETURN true
;  600          1652  1 END;


                                001C 00000 MAKE_UPPER_CASE:
                                           .WORD   Save R2,R3,R4                    ; 1633
                53      04   AC            04  C1 00002        ADDL3   #4, IDESC, R3    ; 1643
                                        04  BC  B5 00007        TSTW    @IDESC           ; 1645
                                            30  13 0000A        BEQL    5$               ;
                                    54      04  BC  3C 0000C    MOVZWL  @IDESC, R4       ; 1646
                                            54  D7 00010        DECL    R4               ;
                                            50  D4 00012        CLRL    I                ; 1649
                                            21  11 00014        BRB     4$               ;
                52              50      08   AC  C1 00016 1$:   ADDL3   ONAME, I, R2     ;
                                51      00 B340  9A 0001B        MOVZBL  @0(R3)[I], R1    ; 1647
                            61  8F              51  91 00020    CMPB    R1, #97          ;
                                            0C  1F 00024        BLSSU   2$               ;
                            7A  8F              51  91 00026    CMPB    R1, #122         ; 1648
                                            06  1A 0002A        BGTRU   2$               ;
                62              51              20  83 0002C    SUBB3   #32, R1, (R2)    ; 1649
                                            03  11 00030        BRB     3$               ;
                                62              51  90 00032 2$:   MOVB    R1, (R2)         ; 1650
                                            50  D6 00035 3$:   INCL    I                ; 1647
                                54              50  D1 00037 4$:   CMPL    I, R4            ;
                                            DA  1B 0003A        BLEQU   1$               ;
                                50              01  D0 0003C 5$:   MOVL    #1, R0           ; 1651
                                                04 0003F        RET                      ; 1652

; Routine Size:  64 bytes,    Routine Base:  $CODE$ + 04EA
```

```
;  602          1653  1 ROUTINE elim_trail_blnk =
;  603          1654  2 BEGIN
;  604          1655  2 !
;  605          1656  2 ! Eliminate trailing blanks from the line
;  606          1657  2 !
;  607          1658  2 lineptr = .endptr - 1;
;  608          1659  2 skip_blnk_bkwds ();
;  609          1660  2 linelen = CH$DIFF (.lineptr, .lineaddr) + 1;
;  610          1661  2 WHILE CH$DIFF (.lineptr, .lineaddr) GEQ 0
;  611          1662  2 DO IF (curchar = CH$RCHAR (.lineptr)) NEQ %ASCII';'
;  612          1663  2 THEN lineptr = CH$PLUS (.lineptr, -1)
;  613          1664  3 ELSE BEGIN
;  614          1665  3     linelen = CH$DIFF (.lineptr, .lineaddr);
;  615          1666  3     EXITLOOP;
;  616          1667  2     END;
;  617          1668  2
;  618          1669  2 lineptr = .lineaddr + .linelen - 1;
;  619          1670  2 skip_blnk_bkwds ();
;  620          1671  2 linelen = CH$DIFF (.lineptr, .lineaddr) + 1;
;  621          1672  2 RETURN true
;  622          1673  1 END;                                !Of elim_trail_blnk


                                        0004 00000 ELIM_TRAIL_BLNK:
                                                   .WORD   Save R2                                  ; 1653
                        52      0000'   CF  9E 00002       MOVAB   LINEPTR, R2                      ;
              62        04      A2          01  C3 00007   SUBL3   #1, ENDPTR, LINEPTR              ; 1658
                    0000V       CF          00  FB 0000C   CALLS   #0, SKIP_BLNK_BKWDS              ; 1659
              50            62      E0      A2  C3 00011   SUBL3   LINEADDR, LINEPTR, R0           ; 1660
          DC  A2                50          01  A1 00016   ADDW3   #1, R0, LINELEN                 ;
                                51          62  D0 0001B 1$:MOVL    LINEPTR, R1                  ; 1661
                            E0  A2              51  D1 0001E   CMPL    R1, LINEADDR                 ;
                                                16  19 00022   BLSS    3$                           ;
                                50              61  9A 00024   MOVZBL  (R1), R0                     ; 1662
                            F4  A2              50  D0 00027   MOVL    R0, CURCHAR                  ;
                                3B              50  91 0002B   CMPB    R0, #59                      ;
                                                04  13 0002E   BEQL    2$                           ;
                                                62  D7 00030   DECL    LINEPTR                      ; 1663
                                                E7  11 00032   BRB     1$                           ;
          DC  A2                51      E0      A2  A3 00034 2$:SUBW3   LINEADDR, R1, LINELEN       ; 1665
                                50      DC      A2  3C 0003A 3$:MOVZWL  LINELEN, R0                 ; 1669
                                50      E0      A2  C0 0003E   ADDL2   LINEADDR, R0                 ;
                                62      FF      A0  9E 00042   MOVAB   -1(R0), LINEPTR              ;
                        0000V   CF              00  FB 00046   CALLS   #0, SKIP_BLNK_BKWDS          ; 1670
              50            62      E0      A2  C3 0004B   SUBL3   LINEADDR, LINEPTR, R0           ; 1671
          DC  A2                50              01  A1 00050   ADDW3   #1, R0, LINELEN             ;
                                50              01  D0 00055   MOVL    #1, R0                       ; 1672
                                                04  00058   RET                                     ; 1673

; Routine Size:  89 bytes,    Routine Base:  $CODE$ + 052A
```

```
;  624          1674  1 ROUTINE skip_blnk_bkwds =
;  625          1675  2 BEGIN
;  626          1676  2 !
;  627          1677  2 ! This routine skips blanks in an input line backwards.
;  628          1678  2 !
;  629          1679  2 WHILE CH$DIFF (.lineptr, .lineaddr) GEQ 0
;  630          1680  2 DO IF (curchar = CH$RCHAR (.lineptr)) EQL %ASCII' '
;  631          1681  2         OR .curchar EQL %ASCII'	'
;  632          1682  2     THEN lineptr = CH$PLUS (.lineptr, -1)
;  633          1683  2     ELSE RETURN true;
;  634          1684  2 RETURN true
;  635          1685  1 END;                                !Of skip_blnk_bkwds


                                        0004 00000 SKIP_BLNK_BKWDS:
                                                   .WORD   Save R2                          ; 1674
                    52      0000'   CF  9E 00002           MOVAB   LINEPTR, R2
                E0  A2              62  D1 00007 1$:        CMPL    LINEPTR, LINEADDR        ; 1679
                                    17  19 0000B           BLSS    3$
                    50        00    B2  9A 0000D           MOVZBL  @LINEPTR, R0             ; 1680
                F4  A2              50  D0 00011           MOVL    R0, CURCHAR
                    20              50  91 00015           CMPB    R0, #32
                                    06  13 00018           BEQL    2$
                    09        F4    A2  D1 0001A           CMPL    CURCHAR, #9              ; 1681
                                    04  12 0001E           BNEQ    3$
                                    62  D7 00020 2$:        DECL    LINEPTR                  ; 1682
                                    E3  11 00022           BRB     1$
                    50              01  D0 00024 3$:        MOVL    #1, R0                   ; 1684
                                        04 00027           RET                              ; 1685

; Routine Size:  40 bytes,    Routine Base:  $CODE$ + 0583
```

```
;  637          1686  1 ROUTINE setmacroname =
;  638          1687  2 BEGIN
;  639          1688  2 !
;  640          1689  2 ! This routine converts the macro name to upper case
;  641          1690  2 ! and saves it for later checking on the .ENDM and for
;  642          1691  2 ! entering the macro name into the library.
;  643          1692  2 !
;  644          1693  2 BIND
;  645          1694  2     inpdesc = lib$gl_inpfdb [fdb$l_namdesc] : BBLOCK,
;  646          1695  2     macrodesc = macnamptrtbl [(.nestinglevel - 1) * dsc$c_s_bln,0,0,0] : BBLOCK;
;  647          1696  2
;  648          1697  2 IF .token2len GTRU .lib$gl_keysize
;  649          1698  3 THEN BEGIN
;  650          1699  3     SIGNAL (lib$_macnamlng, 2, token2desc, inpdesc);
;  651          1700  3     RETURN lib$_macnamlng;
;  652          1701  2     END;
;  653          1702  2
;  654          1703  2 IF .macrodesc [dsc$a_pointer] EQL 0
;  655          1704  2 THEN perform (lib_get_zmem (lbr$c_maxkeylen+1, macrodesc [dsc$a_pointer]));
;  656          1705  2 macrodesc [dsc$a_pointer] = .macrodesc [dsc$a_pointer] + 1;
;  657          1706  2 make_upper_case (token2desc, .macrodesc [dsc$a_pointer]);
;  658          1707  2 macrodesc [dsc$w_length] = .token2len;
;  659          1708  3 BEGIN
;  660          1709  3     BIND
;  661          1710  3         namlen = .macrodesc [dsc$a_pointer]-1 : VECTOR [,BYTE]; !Name first byte (length)
;  662          1711  3
;  663          1712  3     namlen [0] = .macrodesc [dsc$w_length];                              !Set length into name
;  664          1713  2     END;
;  665          1714  2
;  666          1715  2 IF .nestinglevel EQL 1
;  667          1716  2 THEN
;  668          1717  3     BEGIN
;  669          1718  3     IF NOT .lib$gl_ctlmsk [lib$v_replace]                 !if not replacing
;  670          1719  3         AND lbr$lookup_key (lib$gl_libctl, macrodesc, macrorfa)
;  671          1720  3     THEN
;  672          1721  4         BEGIN
;  673          1722  4         SIGNAL (lib$_dupmodule, 3, .macrodesc [dsc$a_pointer] - 1, lib$gl_inpfdb [fdb$l_namdesc],
;  674          1723  4                 lib$gl_libfdb [fdb$l_namdesc]);
;  675          1724  4         dupseen = true;
;  676          1725  3         END;
;  677          1726  2     END;
;  678          1727  2
;  679          1728  2 macrodesc [dsc$a_pointer] = .macrodesc [dsc$a_pointer] - 1;
;  680          1729  2 RETURN true
;  681          1730  1 END;


                                        007C 00000 SETMACRONAME:
                                                   .WORD   Save R2,R3,R4,R5,R6                          ; 1686
                          56 00000000G   00   9E 00002     MOVAB   LIB$SIGNAL, R6                       ;
                          55 00000000G   8F   D0 00009     MOVL    #LIB$_MACNAMLNG, R5                  ;
                          54      0000'  CF   9E 00010     MOVAB   TOKEN2LEN, R4                        ;
                    51    0000G  CF         10   C1 00015     ADDL3   #16, LIB$GL_INPFDB, R1               ; 1694
                          50       1C   A4   D0 0001B     MOVL    NESTINGLEVEL, R0                     ; 1695
```

```
                        53        2C B440   7E 0001F          MOVAQ   @MACNAMPTRTBL[R0], R3                                  ;
                        53              08  C2 00024          SUBL2   #8, R3                                                  ;
    0000G  CF   64              10        00  ED 00027          CMPZV   #0, #16, TOKEN2LEN, LIB$GL_KEYSIZE                      ; 1697
                                          0F  1B 0002E          BLEQU   1$                                                      ;
                                          51  DD 00030          PUSHL   R1                                                      ; 1699
                                          54  DD 00032          PUSHL   R4
                                          02  DD 00034          PUSHL   #2
                                          55  DD 00036          PUSHL   R5
                                  66      04  FB 00038          CALLS   #4, LIB$SIGNAL                                          ;
                                  50      55  D0 0003B          MOVL    R5, R0                                                  ; 1700
                                          04  00 0003E          RET                                                             ;
                                  52  04  A3  9E 0003F  1$:     MOVAB   4(R3), R2                                               ; 1703
                                          62  D5 00043          TSTL    (R2)                                                    ;
                                          0E  12 00045          BNEQ    2$                                                      ;
                                          52  DD 00047          PUSHL   R2                                                      ; 1704
                                  7E  81  8F  9A 00049          MOVZBL  #129, -(SP)                                             ;
                0000G  CF                 02  FB 0004D          CALLS   #2, LIB_GET_ZMEM                                        ;
                                  58      50  E9 00052          BLBC    STATUS, 4$                                              ;
                                          62  D6 00055  2$:     INCL    (R2)                                                    ; 1705
                                          62  DD 00057          PUSHL   (R2)                                                    ; 1706
                                          54  DD 00059          PUSHL   R4                                                      ;
                FEDF    CF                02  FB 0005B          CALLS   #2, MAKE_UPPER_CASE                                     ;
                                  63      64  B0 00060          MOVW    TOKEN2LEN, (R3)                                         ; 1707
                50              62        01  C3 00063          SUBL3   #1, (R2), R0                                            ; 1710
                                  60      63  90 00067          MOVB    (R3), (R0)                                              ; 1712
                                  01  1C  A4  D1 0006A          CMPL    NESTINGLEVEL, #1                                        ; 1715
                                          38  12 0006E          BNEQ    3$                                                      ;
                32      0000G   CF        05  E0 00070          BBS     #5, LIB$GL_CTLMSK+1, 3$                                 ; 1718
                                      24  A4  9F 00076          PUSHAB  MACRORFA                                                ; 1719
                                          53  DD 00079          PUSHL   R3                                                      ;
                                0000G     CF  9F 0007B          PUSHAB  LIB$GL_LIBCTL                                           ;
            00000000G   00                03  FB 0007F          CALLS   #3, LBR$LOOKUP_KEY                                      ;
                                  1F      50  E9 00086          BLBC    R0, 3$                                                  ;
            7E      0000G   CF            10  C1 00089          ADDL3   #16, LIB$GL_LIBFDB, -(SP)                               ; 1723
            7E      0000G   CF            10  C1 0008F          ADDL3   #16, LIB$GL_INPFDB, -(SP)                               ; 1722
            7E              62            01  C3 00095          SUBL3   #1, (R2), -(SP)                                         ;
                                          03  DD 00099          PUSHL   #3                                                      ; 1723
                          00000000G       8F  DD 0009B          PUSHL   #LIB$_DUPMODULE                                         ;
                                  66      05  FB 000A1          CALLS   #5, LIB$SIGNAL                                          ;
                            0C    A4            01  D0 000A4          MOVL    #1, DUPSEEN                                             ; 1724
                                                62  D7 000A8  3$:     DECL    (R2)                                                    ; 1728
                                  50            01  D0 000AA          MOVL    #1, R0                                                  ; 1729
                                                04  00 000AD  4$:     RET                                                             ; 1730

; Routine Size:  174 bytes,    Routine Base:  $CODE$ + 05AB
```

```
;  683          1731  1 ROUTINE checkendmac =
;  684          1732  2 BEGIN
;  685          1733  2 !
;  686          1734  2 ! This routine checks that the name specified on the .ENDM
;  687          1735  2 ! matches what is expected.
;  688          1736  2 !
;  689          1737  2 BIND
;  690          1738  2     inpdesc = lib$gl_inpfdb [fdb$l_namdesc] : BBLOCK,
;  691          1739  2     macrodesc = macnamptrtbl [(.nestinglevel - 1) * dsc$c_s_bln,0,0,0] : BBLOCK;
;  692          1740  2
;  693          1741  2 LOCAL
;  694          1742  2     endname : VECTOR [lbr$c_maxkeylen,BYTE];
;  695          1743  2
;  696          1744  2 IF .token2len NEQ 0
;  697          1745  3 THEN BEGIN
;  698          1746  3     IF .token2len GTRU .lib$gl_keysize
;  699          1747  3     THEN SIGNAL (lib$_macnamlng, 2, token2desc, inpdesc);
;  700          1748  3     make_upper_case (token2desc, endname);
;  701          1749  3     IF NOT CH$EQL (.token2len, endname, .macrodesc [dsc$w_length],
;  702          1750  3                    .macrodesc [dsc$a_pointer] + 1)
;  703          1751  4     THEN BEGIN
;  704          1752  4         macrodesc [dsc$a_pointer] = .macrodesc [dsc$a_pointer] + 1;
;  705          1753  4         SIGNAL (lib$_endwrngmac, 3, token2desc, macrodesc, inpdesc);
;  706          1754  4         macrodesc [dsc$a_pointer] = .macrodesc [dsc$a_pointer] - 1;
;  707          1755  3         END;
;  708          1756  2     END;
;  709          1757  2
;  710          1758  2 RETURN true
;  711          1759  1 END;


                                00FC 00000 CHECKENDMAC:
                                           .WORD   Save R2,R3,R4,R5,R6,R7                          ; 1731
                    57 00000000G   00   9E 00002          MOVAB   LIB$SIGNAL, R7                  ;
                    56     0000'   CF   9E 00009          MOVAB   TOKEN2LEN, R6                   ;
                    5E        80   AE   9E 0000E          MOVAB   -128(SP), SP                    ;
              55    0000G   CF        10   C1 00012       ADDL3   #16, LIB$GL_INPFDB, R5          ; 1738
                    50        1C   A6   D0 00018          MOVL    NESTINGLEVEL, R0                ; 1739
                    54        2C B640   7E 0001C          MOVAQ   @MACNAMPTRTBL[R0], R4           ;
                    54           08   C2 00021          SUBL2   #8, R4                          ;
                    50              66   3C 00024       MOVZWL  TOKEN2LEN, R0                   ; 1744
                                    41   13 00027       BEQL    2$                              ;
                 0000G   CF         50   D1 00029       CMPL    R0, LIB$GL_KEYSIZE              ; 1746
                                    0F   1B 0002E       BLEQU   1$                              ;
                                    55   DD 00030       PUSHL   R5                              ; 1747
                                    56   DD 00032       PUSHL   R6                              ;
                                    02   DD 00034       PUSHL   #2                              ;
                    00000000G       8F   DD 00036       PUSHL   #LIB$_MACNAMLNG                 ;
                    67              04   FB 0003C       CALLS   #4, LIB$SIGNAL                  ;
                          4040      8F   BB 0003F 1$:    PUSHR   #^M<R6,SP>                      ; 1748
                 FE49   CF         02   FB 00043       CALLS   #2, MAKE_UPPER_CASE             ;
                    50        04   A4   D0 00048       MOVL    4(R4), R0                       ; 1750
              64    00        6E        66   2D 0004C       CMPC5   TOKEN2LEN, ENDNAME, #0, (R4), 1(R0) ; 1749
                              01   A0        00051                                             ;
```

```
                                     15  13 00053          BEQL    2$                                                  ;
                                  04 A4  D6 00055          INCL    4(R4)                                               ; 1752
                                     30  BB 00058          PUSHR   #^M<R4,R5>                                          ; 1753
                                     56  DD 0005A          PUSHL   R6                                                  ;
                                     03  DD 0005C          PUSHL   #3                                                  ;
                           00000000G 8F  DD 0005E          PUSHL   #LIB$_ENDWRNGMAC                                    ;
                                  67 05  FB 00064          CALLS   #5, LIB$SIGNAL                                      ;
                                  04 A4  D7 00067          DECL    4(R4)                                               ; 1754
                                  50 01  D0 0006A 2$:      MOVL    #1, R0                                              ; 1758
                                     04  00 0006D          RET                                                         ; 1759

; Routine Size:  110 bytes,    Routine Base:  $CODE$ + 0659

;  712          1760  1
;  713          1761  1 END
;  714          1762  0 ELUDOM


                                                          .EXTRN  LIB$SIGNAL

;                                PSECT SUMMARY
;
;      Name                      Bytes                          Attributes
;
;  $OWN$                           216  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $PLIT$                           76  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $CODE$                         1735  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]STARLET.L32;1      9776        26         0       581          00:01.0



;                                     COMMAND QUALIFIERS

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

; Size:         1735 code + 292 data bytes
; Run Time:       00:36.9
; Elapsed Time:   01:07.9
; Lines/CPU Min:   2863
; Lexemes/CPU-Min: 30580
; Memory Used:  314 pages
```

```
; Compilation Complete
```

GETCMD
LIS

INPUTOBJ
LIS

INPUTTXT
LIS

LIBRARIAN
LIS

INPUTMAC
LIS

INPUTHLP
LIS