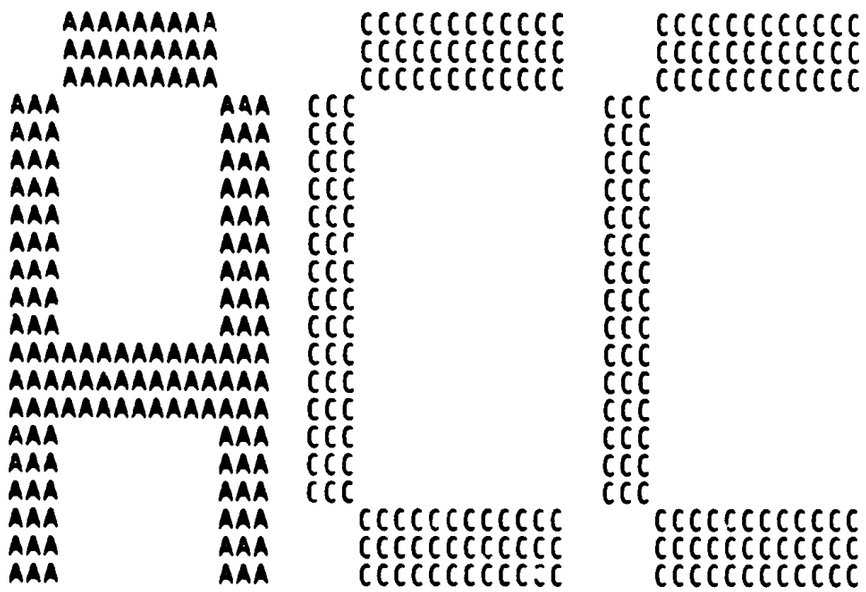
TITLE1**AH-Bi13A-BE
TITLE2**VAX/VMS V4.0
FICHENUMBER**1



```
**FILE**ID**SYMBOLS                                    E  5
```

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

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

```
;     1          0001  0 MODULE symbols (IDENT = 'V04-000') =
;     2          0002  1 BEGIN
;     3          0003  1
;     4          0004  1 !
;     5          0005  1 !*****************************************************************************
;     6          0006  1 !*                                                                           *
;     7          0007  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
;     8          0008  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
;     9          0009  1 !*  ALL RIGHTS RESERVED.                                                     *
;    10          0010  1 !*                                                                           *
;    11          0011  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
;    12          0012  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    13          0013  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;    14          0014  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;    15          0015  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;    16          0016  1 !*  TRANSFERRED.                                                             *
;    17          0017  1 !*                                                                           *
;    18          0018  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;    19          0019  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;    20          0020  1 !*  CORPORATION.                                                             *
;    21          0021  1 !*                                                                           *
;    22          0022  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
;    23          0023  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
;    24          0024  1 !*                                                                           *
;    25          0025  1 !*                                                                           *
;    26          0026  1 !*****************************************************************************
;    27          0027  1 !
;    28          0028  1
;    29          0029  1 !++
;    30          0030  1 ! FACILITY:  Command language editor
;    31          0031  1 !
;    32          0032  1 ! ABSTRACT:
;    33          0033  1 !
;    34          0034  1 !       This facility is used to enhance the command language
;    35          0035  1 !       and allow user-written commands to be available in the
;    36          0036  1 !       language.
;    37          0037  1 !
;    38          0038  1 ! ENVIRONMENT:
;    39          0039  1 !
;    40          0040  1 !       VAX/VMS operating system. unprivileged user mode,
;    41          0041  1 !
;    42          0042  1 ! AUTHOR:  Tim Halvorsen, Feb 1980
;    43          0043  1 !
;    44          0044  1 ! Modified by:
;    45          0045  1 !
;    46          0046  1 !       V03-001 DAS0001         David Solomon   03-Jul-1984
;    47          0047  1 !               Return success if trying to add a duplicate symbol (SPR 55578).
;    48          0048  1 !
;    49          0049  1 !       V002    DWT0006         David W. Thiel  10-Dec-1981
;    50          0050  1 !               Fix find_record to fail if asked for (n+1)st record.
;    51          0051  1 !
;    52          0052  1 !       V001    TMH0001         Tim Halvorsen   28-Mar-1981
;    53          0053  1 !               Clear upper word of descriptor passed to scan_symbols
;    54          0054  1 !               action routine, in case it uses the entire longword as
;    55          0055  1 !               the length rather than the lower word.
;    56          0056  1 !--
;    57          0057  1
```

```
;    58          0058  1 !
;    59          0059  1 ! Include files
;    60          0060  1 !
;    61          0061  1
;    62          0062  1 LIBRARY 'SYS$LIBRARY:STARLET';          ! VAX/VMS common definitions
;    63          0063  1
;    64          0064  1 !** REQUIRE 'SRC$:CLEDEF';                      ! Utility definitions
;    65          0065  1 !---
;    66          0066  1 !
;    67          0067  1 !       Require file for all modules in the command language editor
;    68          0068  1 !
;    69          0069  1 ! IDENT V02-001
;    70          0070  1 !
;    71          0071  1 !---
;    72          0072  1
;    73          0073  1 !
;    74          0074  1 !*****************************************************************************
;    75          0075  1 !*                                                                           *
;    76          0076  1 !*  COPYRIGHT (c) 1978, 1980, 1982 BY                                        *
;    77          0077  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
;    78          0078  1 !*  ALL RIGHTS RESERVED.                                                      *
;    79          0079  1 !*                                                                           *
;    80          0080  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
;    81          0081  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
;    82          0082  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
;    83          0083  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
;    84          0084  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
;    85          0085  1 !*  TRANSFERRED.                                                              *
;    86          0086  1 !*                                                                           *
;    87          0087  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
;    88          0088  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT     *
;    89          0089  1 !*  CORPORATION.                                                              *
;    90          0090  1 !*                                                                           *
;    91          0091  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
;    92          0092  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
;    93          0093  1 !*                                                                           *
;    94          0094  1 !*                                                                           *
;    95          0095  1 !*****************************************************************************
;    96          0096  1 !
;    97          0097  1
;    98          0098  1 !++
;    99          0099  1 ! FACILITY:  Command language editor
;   100          0100  1 !
;   101          0101  1 ! ABSTRACT:
;   102          0102  1 !
;   103          0103  1 !       This is the common require file for all modules in the
;   104          0104  1 !       command language editor.
;   105          0105  1 !
;   106          0106  1 ! ENVIRONMENT:
;   107          0107  1 !
;   108          0108  1 !       VAX/VMS operating system, unprivileged user mode utility,
;   109          0109  1 !       operates at non-AST level.
;   110          0110  1 !
;   111          0111  1 ! AUTHOR:  Tim Halvorsen,  Feb 1980
;   112          0112  1 !
;   113          0113  1 ! MODIFIED BY:
;   114          0114  1 !
```

```
;  115          0115  1 !       V02-001 BLS0089          Benn Schreiber          16-Oct-1981
;  116          0116  1 !               Add badvalue shared message
;  117          0117  1 !----
```

```
;  119          0118  1
;  120          0119  1 !
;  121          0120  1 !       Define commonly used BLISS definitions
;  122          0121  1 !
;  123          0122  1
;  124          0123  1 !** REQUIRE 'LIB$:UTILDEF';                    ! Commonly used BLISS definitions
;  125          0124  1 !---
;  126          0125  1 !
;  127          0126  1 !       Commonly used definitions for VMS modules written in BLISS
;  128          0127  1 !
;  129          0128  1 ! Version 'V03-000'
;  130          0129  1 !
;  131          0130  1 !****************************************************************************
;  132          0131  1 !*                                                                          *
;  133          0132  1 !*  COPYRIGHT (c) 1978, 1980, 1982 BY                                       *
;  134          0133  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
;  135          0134  1 !*  ALL RIGHTS RESERVED.                                                     *
;  136          0135  1 !*                                                                          *
;  137          0136  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
;  138          0137  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;  139          0138  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;  140          0139  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;  141          0140  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;  142          0141  1 !*  TRANSFERRED.                                                             *
;  143          0142  1 !*                                                                          *
;  144          0143  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;  145          0144  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;  146          0145  1 !*  CORPORATION.                                                             *
;  147          0146  1 !*                                                                          *
;  148          0147  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
;  149          0148  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
;  150          0149  1 !*                                                                          *
;  151          0150  1 !*                                                                          *
;  152          0151  1 !****************************************************************************
;  153          0152  1
;  154          0153  1 !++
;  155          0154  1 ! ABSTRACT:
;  156          0155  1 !
;  157          0156  1 !       This is the common require file for any module written
;  158          0157  1 !       in BLISS
;  159          0158  1 !
;  160          0159  1 ! ENVIRONMENT:
;  161          0160  1 !
;  162          0161  1 !       VAX/VMS operating system.
;  163          0162  1 !
;  164          0163  1 ! AUTHOR:  Tim Halvorsen,  Feb 1980
;  165          0164  1 !
;  166          0165  1 ! MODIFIED BY:
;  167          0166  1 !
;  168          0167  1 !----
```

```
;  170          0168  1 !
;  171          0169  1 !       Equated symbols
;  172          0170  1 !
;  173          0171  1
;  174          0172  1 LITERAL
;  175          0173  1     true        = 1,                    ! boolean true
;  176          0174  1     false       = 0,                    ! boolean false
;  177          0175  1     ok          = 1,                    ! success return code
;  178          0176  1     error       = 2,                    ! error return code
;  179          0177  1     quad        = 8;                    ! quadword allocation definition
;  180          0178  1
;  181          0179  1 !
;  182          0180  1 !       Define structure type for VMS structures
;  183          0181  1 !
;  184          0182  1
;  185          0183  1 STRUCTURE
;  186          0184  1     bblock [o, p, s, e; n] =
;  187          0185  1                 [n]
;  188          0186  1                 (bblock+o)<p,s,e>;
;  189          0187  1
;  190          0188  1 MACRO
;  191        M 0189  1     descriptor [] =             ! Generate a static string descriptor
;  192        M 0190  1         UPLIT (%CHARCOUNT (%STRING (%REMAINING)),
;  193          0191  1                UPLIT BYTE (%STRING (%REMAINING))) %;
;  194          0192  1
;  195          0193  1 MACRO
;  196        M 0194  1     own_desciptor [] =          ! Generate the actual static string descriptor
;  197        M 0195  1         BBLOCK [8] INITIAL(%CHARCOUNT(%STRING(%REMAINING)),
;  198          0196  1                            UPLIT BYTE (%STRING(%REMAINING))) %;
;  199          0197  1
;  200          0198  1 MACRO
;  201        M 0199  1     return_if_error(command) =
;  202        M 0200  1         BEGIN
;  203        M 0201  1         LOCAL
;  204        M 0202  1             status;
;  205        M 0203  1
;  206        M 0204  1         status = command;
;  207        M 0205  1         IF NOT .status
;  208        M 0206  1         THEN
;  209        M 0207  1             RETURN .status;
;  210          0208  1         END%;
;  211          0209  1
;  212          0210  1 MACRO
;  213        M 0211  1     signal_if_error(command) =
;  214        M 0212  1         BEGIN
;  215        M 0213  1         LOCAL
;  216        M 0214  1             status;
;  217        M 0215  1
;  218        M 0216  1         status = command;
;  219        M 0217  1         IF NOT .status
;  220        M 0218  1         THEN
;  221        M 0219  1             BEGIN
;  222        M 0220  1             SIGNAL(.status);
;  223        M 0221  1             RETURN .status;
;  224        M 0222  1             END;
;  225          0223  1         END%;
;  226          0224  1
```

```
;  227          0225  1 !
;  228          0226  1 ! Macro to implement a function (f) of the message severity level that
;  229          0227  1 ! maps the various severity levels such that arithmetic comparisions of the
;  230          0228  1 ! mapped values (  f(severity) )  yield a order of precedence that is
;  231          0229  1 ! intuivitvely acceptable:
;  232          0230  1 !
;  233          0231  1 !
;  234          0232  1 !                   ERROR NAME      OLDVAL      NEWVAL
;  235          0233  1 !
;  236          0234  1 !                   F(SUCCESS)         1    -->    0
;  237          0235  1 !                   F(INFORMATIONAL)   3    -->    2
;  238          0236  1 !                   F(WARNING)         0    -->    3
;  239          0237  1 !                   F(ERROR)           2    -->    5
;  240          0238  1 !                   F(SEVERE_ERROR)    4    -->    7
;  241          0239  1 !
;  242          0240  1
;  243          0241  1 MACRO
;  244        M 0242  1     severity_level (status) =
;  245        M 0243  1         BEGIN
;  246        M 0244  1         LOCAL code: BBLOCK [LONG];
;  247        M 0245  1         code = status;
;  248        M 0246  1         .code [sts$v_severity] - (4 * .code [sts$v_success]) + 3
;  249          0247  1         END%;
;  250          0248  1
;  251          0249  1 MACRO
;  252        M 0250  1     cli$external(prefix) =
;  253        M 0251  1         %IF %DECLARED(%QUOTE %QUOTE cli$prefix)
;  254        M 0252  1                 %THEN UNDECLARE %QUOTE %QUOTE cli$prefix; %FI
;  255        M 0253  1         MACRO cli$prefix = prefix %QUOTE %;
;  256        M 0254  1         EXTERNAL LITERAL
;  257          0255  1             cli$external_loop(%REMAINING)%,
;  258          0256  1
;  259        M 0257  1     cli$external_loop[name] =
;  260          0258  1         %NAME(cli$prefix,name): UNSIGNED(8)%;
;  261          0259  1
;  262          0260  1 MACRO
;  263        M 0261  1     $external_literal(symbol) =
;  264        M 0262  1         BEGIN
;  265        M 0263  1         %IF NOT %DECLARED(symbol) %THEN EXTERNAL LITERAL symbol
;  266        M 0264  1         %IF %LENGTH GTR 1 %THEN : %REMAINING %FI; %FI
;  267        M 0265  1         symbol
;  268          0266  1         END%;
;  269          0267  1
;  270          0268  1 MACRO
;  271        M 0269  1     $fab_dev(dev_bit) =         ! Access FAB$L_DEV bits of FAB block
;  272        M 0270  1         $BYTEOFFSET(fab$l_dev),
;  273          0271  1         $BITPOSITION(%NAME('dev$v_',dev_bit)),1,0%;
;  274          0272  1
;  275          0273  1 !
;  276          0274  1 ! $SHR_MESSAGES - a macro which defines facility-specific message codes
;  277          0275  1 !       which are based on the system-wide shared message codes.
;  278          0276  1 !
;  279          0277  1 !       $SHR_MESSAGES( name, code, (msg,severity), ... )
;  280          0278  1 !
;  281          0279  1 !       where:
;  282          0280  1 !           "name" is the name of the facility (e.g., COPY)
;  283          0281  1 !           "code" is the corresponding facility code (e.g., 103)
```

```
;  284          0282  1 !           "msg" is the name of the shared message (e.g., BEGIN)
;  285          0283  1 !           "severity" is the desired message severity (e.g., 1, 0, 2)
;  286          0284  1 !
;  287          0285  1
;  288          0286  1 MACRO
;  289        M 0287  1     $SHR_MESSAGES( FACILITY_NAME, FACILITY_CODE ) =
;  290        M 0288  1         LITERAL
;  291          0289  1         SHR$MSG_IDS( FACILITY_NAME, FACILITY_CODE, %REMAINING ); %,
;  292          0290  1
;  293        M 0291  1     SHR$MSG_IDS( FACILITY_NAME, FACILITY_CODE) [ VALUE ] =
;  294          0292  1         SHR$MSG_CALC( FACILITY_NAME, FACILITY_CODE, %REMOVE(VALUE) ) %,
;  295          0293  1
;  296        M 0294  1     SHR$MSG_CALC( FACILITY_NAME, FACILITY_CODE, MSG_ID, SEVERITY ) =
;  297        M 0295  1         %NAME(FACILITY_NAME,'$',MSG_ID) = %NAME('SHR$_',MSG_ID) + FACILITY_CODE*65536 +
;  298        M 0296  1             %IF %DECLARED(%NAME('STS$K_',SEVERITY))
;  299        M 0297  1                 %THEN %NAME('STS$K_',SEVERITY)
;  300          0298  1                 %ELSE SEVERITY %FI %;
```

```
;  302     0299  1
;  303     0300  1 !** REQUIRE 'LIB$:CLIDEF.B32';          ! CLI command table definitions
;  304     0301  1 !
;  305     0302  1 !       Command language interpreter command table structures
;  306     0303  1 !
;  307     0304  1 !  IDENT        V03-003
;  308     0305  1 !
;  309     0306  1 !
;  310     0307  1 !
;  311     0308  1 !*****************************************************************************
;  312     0309  1 !*                                                                           *
;  313     0310  1 !*  COPYRIGHT (c) 1978, 1980, 1982 BY                                        *
;  314     0311  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
;  315     0312  1 !*  ALL RIGHTS RESERVED.                                                      *
;  316     0313  1 !*                                                                           *
;  317     0314  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
;  318     0315  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;  319     0316  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER  *
;  320     0317  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
;  321     0318  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY  *
;  322     0319  1 !*  TRANSFERRED.                                                              *
;  323     0320  1 !*                                                                           *
;  324     0321  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;  325     0322  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;  326     0323  1 !*  CORPORATION.                                                              *
;  327     0324  1 !*                                                                           *
;  328     0325  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
;  329     0326  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
;  330     0327  1 !*                                                                           *
;  331     0328  1 !*                                                                           *
;  332     0329  1 !*****************************************************************************
;  333     0330  1 !
;  334     0331  1
;  335     0332  1 !++
;  336     0333  1 ! FACILITY:  DCL & MCR Command language interpreters
;  337     0334  1 !
;  338     0335  1 ! ABSTRACT:
;  339     0336  1 !
;  340     0337  1 !       These are the command table structure definitions
;  341     0338  1 !       which describe the generic command table format used
;  342     0339  1 !       by the DCL and MCR command interpreters.
;  343     0340  1 !
;  344     0341  1 ! ENVIRONMENT:
;  345     0342  1 !
;  346     0343  1 !       VAX/VMS operating system. supervisor mode.
;  347     0344  1 !
;  348     0345  1 ! AUTHOR:  Tim Halvorsen, Feb 1980
;  349     0346  1 !
;  350     0347  1 ! Modified by:
;  351     0348  1 !
;  352     0349  1 !       V03-003 PCG0005         Peter George    22-Nov-1982
;  353     0350  1 !               Add INT_W_PMPTLEN and INT_L_PMPTADDR and remove
;  354     0351  1 !               INT_L_PROMPT.
;  355     0352  1 !
;  356     0353  1 !       V03-002 PCG0004         Peter George    18-Oct-1982
;  357     0354  1 !               Add VEC_C_PROMPTMAX, INT_L_PROMPT, and ENT_V_SPELL.
;  358     0355  1 !
```

```
;  359          0356  1  !       V03-001 PCG0003          Peter George    15-Jul-1982
;  360          0357  1  !               Add INT data structure for CLI$INTERFACE routines.
;  361          0358  1  !--
```

```
;  362          0359  1
;  363          0360  1
;  364          0361  1  !
;  365          0362  1  !       Note that the term "SRO" stands for self-relative offset.
;  366          0363  1  !       The actual address is computed by adding the signed contents
;  367          0364  1  !       of the field to the address of the structure.
;  368          0365  1  !
;  369          0366  1  !       If the offset is zero, then there is no associated data.
;  370          0367  1  !
```

```
;  371          0368  1
;  372          0369  1
;  373          0370  1
;  374          0371  1  !
;  375          0372  1  !       DEFINE VECTOR AT FRONT OF COMMAND TABLES DESCRIBING
;  376          0373  1  !       OFFSETS INTO THE SECTION FOR VARIOUS TABLES.
;  377          0374  1  !
;  378          0375  1
;  379          0376  1  !...$VECDEF
;  380          0377  1  MACRO           VEC_L_IMAGETBL  = 0,0,32,0%;            ! OFFSET TO IMAGE TABLE
;  381          0378  1  MACRO           VEC_L_PROMPTBL  = 4,0,32,0%;            ! OFFSET TO PROMPT TABLE
;  382          0379  1  MACRO           VEC_L_QUALTBL   = 8,0,32,0%;            ! OFFSET TO QUALIFIER TABLE
;  383          0380  1  MACRO           VEC_L_VERBTBL   = 12,0,32,0%;           ! OFFSET TO BUILT-IN VERB TABLE
;  384          0381  1  MACRO           VEC_L_VERBEND   = 16,0,32,0%;           ! OFFSET TO END OF VERBTBL
;  385          0382  1  MACRO           VEC_L_USRCMD    = 20,0,32,0%;           ! OFFSET TO USER VERB TABLE
;  386          0383  1  MACRO           VEC_L_USREND    = 24,0,32,0%;           ! OFFSET TO END OF USER VERB TABLE
;  387          0384  1  MACRO           VEC_L_COMDPTR   = 28,0,32,0%;           ! OFFSET TO BUILT-IN POINTER TABLE
;  388          0385  1  MACRO           VEC_L_USERPTR   = 32,0,32,0%;           ! OFFSET TO USER POINTER TABLE
;  389          0386  1  MACRO           VEC_L_FREE      = 36,0,32,0%;                   ! OFFSET TO NEXT FREE BYTE
;  390          0387  1  MACRO           VEC_B_STRLVL    = 40,0,8,0%;            ! TABLE STRUCTURE LEVEL
;  391          0388  1  LITERAL
;  392        P 0389  1  $EQULST (VEC_C_,GBL,0,1
;  393        P 0390  1          ,(STRLVL,5)                          ! CURRENT STRUCTURE LEVEL
;  394          0391  1          );
;  395          0392  1  MACRO           VEC_B_PROMPTMAX = 41,0,8,0%;            ! MAXIMUM SIZE OF ANY PROMPT STRING
;  396          0393  1  LITERAL         VEC_C_LENGTH3   = 42;
;  397          0394  1  LITERAL         VEC_K_LENGTH3   = 42;                           ! LENGTH OF STR LEVEL 3 AND BEFORE VEC
;  398          0395  1  MACRO           VEC_B_CLI       = 42,0,8,0%;            ! CLI TYPE
;  399          0396  1  LITERAL
;  400        P 0397  1  $EQULST (VEC_C_,GBL,0,1
;  401        P 0398  1          ,(DCL,0)                             ! TABLES ARE FOR DCL
;  402        P 0399  1          ,(MCR,1)                             ! TABLES ARE FOR MCR
;  403          0400  1          );
;  404          0401  1  MACRO           VEC_W_SIZE      = 44,0,16,0%;           ! SIZE IN BYTES OF VECTOR AREA
;  405          0402  1  LITERAL         VEC_C_LENGTH    = 60;
;  406          0403  1  LITERAL         VEC_K_LENGTH    = 60;                   ! LENGTH OF VECTOR AREA
;  407          0404  1  LITERAL         VEC_C_PROMPTMAX = 32;           ! MAXIMUM SIZE OF ANY PROMPT STRING
;  408          0405  1
```

```
;  409     0406  1
;  410     0407  1
;  411     0408  1 !
;  412     0409  1 !       DEFINE COMMAND DESCRIPTOR BLOCK
;  413     0410  1 !
;  414     0411  1
;  415     0412  1 !...$CMDDEF
;  416     0413  1 MACRO           CMD_B_SIZE      = 0,0,8,0%;             ! SIZE OF COMMAND DESCRIPTOR BLOCK
;  417     0414  1 MACRO           CMD_B_VERBTYP   = 1,0,8,0%;     ! VERB GENERIC TYPE
;  418     0415  1 MACRO           CMD_B_PARMCNT   = 2,0,8,0%;     ! MIN/MAX PARAMETER COUNTS
;  419     0416  1
;  420     0417  1 MACRO           CMD_V_MINPARM   = 2,0,4,0%;     ! MINIMUM NUMBER OF PARAMETERS REQUIRED
;  421     0418  1 MACRO           CMD_V_MAXPARM   = 2,4,4,0%;     ! MAXIMUM NUMBER OF PARAMETERS ALLOWED
;  422     0419  1
;  423     0420  1 MACRO           CMD_B_FLAGS     = 3,0,8,0%;             ! COMMAND FLAGS
;  424     0421  1
;  425     0422  1 MACRO           CMD_V_ABREV     = 3,0,1,0%;             ! COMMAND MAY BE ABBREVIATED NON-UNIQUELY
;  426     0423  1 LITERAL         CMD_M_ABREV     = 1^1 - 1^0;
;  427     0424  1                                 ! TO A SINGLE CHARACTER
;  428     0425  1 MACRO           CMD_V_NOSTAT    = 3,1,1,0%;             ! COMMAND DOES NOT RETURN VALID STATUS
;  429     0426  1 LITERAL         CMD_M_NOSTAT    = 1^2 - 1^1;
;  430     0427  1 MACRO           CMD_V_FOREIGN   = 3,2,1,0%;             ! FOREIGN COMMAND - NO PARSING IS DONE
;  431     0428  1 LITERAL         CMD_M_FOREIGN   = 1^3 - 1^2;
;  432     0429  1 MACRO           CMD_V_IMMED     = 3,3,1,0%;             ! COMMAND IS IMMEDIATELY DISPATCHED W/O PARSING
;  433     0430  1 LITERAL         CMD_M_IMMED     = 1^4 - 1^3;
;  434     0431  1 MACRO           CMD_V_MCRPARSE  = 3,4,1,0%;     ! COMMAND IS MCR STYLE COMMAND (OUT=IN)
;  435     0432  1 LITERAL         CMD_M_MCRPARSE  = 1^5 - 1^4;
;  436     0433  1                                 ! (THIS FLAG ONLY EXAMINED BY MCR CLI)
;  437     0434  1
;  438     0435  1 MACRO           CMD_W_IMAGE     = 4,0,16,1%;    ! SRO TO ASCIC IMAGE NAME
;  439     0436  1 MACRO           CMD_W_QUALS     = 6,0,16,1%;    ! SRO TO FIRST NONPOSITIONAL ENTITY
;  440     0437  1 MACRO           CMD_W_PARMS     = 8,0,16,1%;    ! SRO TO FIRST POSITIONAL ENTITY
;  441     0438  1 MACRO           CMD_W_OUTPUTS   = 10,0,16,1%;   ! SRO TO LIST OF "OUTPUT" ENTITIES
;  442     0439  1 MACRO           CMD_W_MUTEXSET  = 12,0,16,1%;   ! SRO TO MUTUAL EXCLUSION SET
;  443     0440  1 MACRO           CMD_W_IMPSET    = 14,0,16,1%;   ! SRO TO IMPLICATION SET
;  444     0441  1 LITERAL         CMD_C_LENGTH    = 16;
;  445     0442  1 LITERAL         CMD_K_LENGTH    = 16;                   ! LENGTH OF FIXED PORTION
;  446     0443  1 !
;  447     0444  1 ! OUTPUT LIST FORMAT:
;  448     0445  1 !
;  449     0446  1 ! FIRST BYTE CONTAINS COUNT OF ENTRIES IN LIST EACH ENTRY IS ONE BYTE,
;  450     0447  1 ! SIGNED, DESCRIBING THAT "OUTPUT NUMBER".  NEGATIVE VALUES INDICATE THE
;  451     0448  1 ! OUTPUT IS A PARAMETER AND THE ABS(VALUE) IS THE PARAMETER NUMBER.
;  452     0449  1 ! POSITIVE VALUES INDICATE THE OUTPUT IS A QUALIFIER AND THE VALUE IS A
;  453     0450  1 ! QUALIFIER NUMBER.
;  454     0451  1 !
;  455     0452  1 ! QUAL IS (0:MAXQUALS-1),PARM IS (MAXQUALS:255)
;  456     0453  1 !
;  457     0454  1 LITERAL         CMD_C_MAXPARMS  = 8;            ! MAXIMUM POSSIBLE PARAMETERS
;  458     0455  1 LITERAL         CMD_C_MAXQUALS  = 248;          ! MAXIMUM POSSIBLE QUALIFIERS (256-8)
;  459     0456  1
```

```
;  460          0457  1
;  461          0458  1
;  462          0459  1  !
;  463          0460  1  !       DEFINE ENTITY DESCRIPTOR BLOCK
;  464          0461  1  !
;  465          0462  1
;  466          0463  1  !...$ENTDEF
;  467          0464  1  MACRO          ENT_B_NEXT      = 0,0,8,0%;             ! OFFSET TO NEXT BLOCK IN CHAIN
;  468          0465  1  MACRO          ENT_B_SIZE      = 1,0,8,0%;             ! SIZE OF THIS BLOCK IN BYTES
;  469          0466  1  MACRO          ENT_B_TYPE      = 2,0,8,0%;
;  470          0467  1  LITERAL
;  471        P 0468  1  $EQULST (ENT_C_,GBL,0,1
;  472        P 0469  1          ,(PARAMETER,)            ! POSITIONAL VALUE DELIMITED BY SPACE
;  473        P 0470  1          ,(QUALIFIER,)            ! NONPOSITIONAL KEYWORD DELIMITED BY SLASH
;  474          0471  1          );
;  475          0472  1  MACRO          ENT_B_VALTYPE   = 3,0,8,0%;     ! TYPE OF VALUE
;  476          0473  1  LITERAL
;  477        P 0474  1  $EQULST (ENT_C_,GBL,1,1                  ! (STARTING AT #1)
;  478        P 0475  1          ,(INFILE,)                       ! INPUT FILE SPECIFICATION
;  479        P 0476  1          ,(OUTFILE,)                      ! OUTPUT FILE SPECIFICATION
;  480        P 0477  1          ,(NUMBER,)                       ! DECIMAL INTEGER NUMBER
;  481        P 0478  1          ,(PRIVILEGE,)            ! PRIVILEGE KEYWORD
;  482        P 0479  1          ,(DATETIME,)             ! DATE/TIME SPECIFICATION
;  483        P 0480  1          ,(PROTECTION,)           ! PROTECTION SPECIFICATION
;  484        P 0481  1          ,(PROCESS,)                      ! PROCESS NAME
;  485        P 0482  1          ,(INLOG,)                        ! INPUT LOGICAL NAME
;  486        P 0483  1          ,(OUTLOG,)                       ! OUTPUT LOGICAL NAME
;  487        P 0484  1          ,(INSYM,)                        ! INPUT SYMBOL NAME
;  488        P 0485  1          ,(OUTSYM,)                       ! OUTPUT SYMBOL NAME
;  489        P 0486  1          ,(NODE,)                         ! NODE NAME
;  490        P 0487  1          ,(DEVICE,)                       ! NODE/DEVICE NAME
;  491        P 0488  1          ,(DIR,)                          ! NODE/DEVICE/DIRECTORY NAME
;  492        P 0489  1          ,(UIC,)                          ! UIC SPECIFICATION
;  493        P 0490  1          ,(RESTOFLINE,)           ! GOBBLE REST OF COMMAND STRING AS VALUE
;  494          0491  1          );
;  495          0492  1
;  496          0493  1  MACRO          ENT_W_NAME      = 4,0,16,1%;    ! SRO TO ASCIC ENTITY NAME (USER SPELLING)
;  497          0494  1  MACRO          ENT_W_NUMBER    = 4,0,16,0%;    ! OR, PARAMETER NUMBER (POSITIONAL)
;  498          0495  1  MACRO          ENT_W_LABEL     = 6,0,16,1%;    ! SRO TO ASCIC ENTITY LABEL (FOR PGM USE)
;  499          0496  1  MACRO          ENT_W_DEFVAL    = 8,0,16,1%;    ! SRO TO ASCIC DEFAULT VALUE
;  500          0497  1  MACRO          ENT_W_SYNTAX    = 10,0,16,1%;   ! SRO TO SYNTAX LIST
;  501          0498  1  MACRO          ENT_W_KEYWORDS  = 12,0,16,1%;   ! SRO TO VALUE KEYWORD LIST
;  502          0499  1                                  ! IF ZERO, ALL VALUES ARE LEGAL
;  503          0500  1  MACRO          ENT_W_PROMPT    = 14,0,16,1%;   ! SRO TO VALUE PROMPT
;  504          0501  1  MACRO          ENT_L_FLAGS     = 16,0,32,0%;           ! ENTITY FLAGS
;  505          0502  1
;  506          0503  1  MACRO          ENT_V_FILE      = 16,0,1,0%;            ! VALUE IS FILE SPECIFICATION
;  507          0504  1  LITERAL        ENT_M_FILE      = 1^1 - 1^0;
;  508          0505  1  MACRO          ENT_V_VAL       = 16,1,1,0%;            ! CAN HAVE A VALUE
;  509          0506  1  LITERAL        ENT_M_VAL       = 1^2 - 1^1;
;  510          0507  1  MACRO          ENT_V_NEG       = 16,2,1,0%;            ! VALUE CAN BE NEGATED
;  511          0508  1  LITERAL        ENT_M_NEG       = 1^3 - 1^2;
;  512          0509  1  MACRO          ENT_V_DEFTRUE   = 16,3,1,0%;            ! TRUE BY DEFAULT
;  513          0510  1  LITERAL        ENT_M_DEFTRUE   = 1^4 - 1^3;
;  514          0511  1  MACRO          ENT_V_BATDEF    = 16,4,1,0%;            ! PRESENT BY DEFAULT IF BATCH JOB
;  515          0512  1  LITERAL        ENT_M_BATDEF    = 1^5 - 1^4;
;  516          0513  1  MACRO          ENT_V_VALREQ    = 16,5,1,0%;            ! VALUE IS REQUIRED
```

```
;  517          0514  1  LITERAL       ENT_M_VALREQ    = 1^6 - 1^5;
;  518          0515  1  MACRO         ENT_V_LIST      = 16,6,1,0%;            ! COMMA-SEPARATED LIST OF VALUES ALLOWED
;  519          0516  1  LITERAL       ENT_M_LIST      = 1^7 - 1^6;
;  520          0517  1  MACRO         ENT_V_CONCAT    = 16,7,1,0%;            ! CONCATENATED VALUES ALLOWED
;  521          0518  1  LITERAL       ENT_M_CONCAT    = 1^8 - 1^7;
;  522          0519  1  MACRO         ENT_V_IMPCAT    = 16,8,1,0%;            ! VALUES ARE IMPLICITLY CONCATENATED
;  523          0520  1  LITERAL       ENT_M_IMPCAT    = 1^9 - 1^8;
;  524          0521  1  MACRO         ENT_V_VERB      = 16,9,1,0%;            ! QUALIFIER CAN APPEAR ON COMMAND VERB
;  525          0522  1  LITERAL       ENT_M_VERB      = 1^10 - 1^9;
;  526          0523  1  MACRO         ENT_V_PARM      = 16,10,1,0%;           ! QUALIFIER CAN APPEAR ON PARAMETER
;  527          0524  1  LITERAL       ENT_M_PARM      = 1^11 - 1^10;
;  528          0525  1  MACRO         ENT_V_MCROPTDLM = 16,11,1,0%;   ! VALUE DELIMITER IS OPTIONAL (MCR)
;  529          0526  1  LITERAL       ENT_M_MCROPTDLM = 1^12 - 1^11;
;  530          0527  1  MACRO         ENT_V_MCRIGNORE = 16,12,1,0%;   ! IGNORE THIS ENTITY BLOCK (MCR)
;  531          0528  1  LITERAL       ENT_M_MCRIGNORE = 1^13 - 1^12;
;  532          0529  1  MACRO         ENT_V_SPELL     = 16,13,1,0%;           ! ONLY CHECK FIRST FOUR CHARS OF KEYWORD VALUES
;  533          0530  1  LITERAL       ENT_M_SPELL     = 1^14 - 1^13;
;  534          0531  1
;  535          0532  1  LITERAL       ENT_C_LENGTH    = 20;
;  536          0533  1  LITERAL       ENT_K_LENGTH    = 20;                   ! LENGTH OF FIXED LENGTH PORTION
;  537          0534  1
```

```
;  538          0535  1
;  539          0536  1
;  540          0537  1  !
;  541          0538  1  !       DEFINE CHANGE LIST STRUCTURE
;  542          0539  1  !
;  543          0540  1
;  544          0541  1  !...$CHGDEF
;  545          0542  1  MACRO           CHG_B_SIZE      = 0,0,8,0%;             ! SIZE OF CHANGE LIST BLOCK
;  546          0543  1  MACRO           CHG_B_FLAGS     = 1,0,8,0%;             ! FLAGS
;  547          0544  1
;  548          0545  1  MACRO           CHG_V_IMAGE     = 1,0,1,0%;             ! IMAGE CHANGE
;  549          0546  1  LITERAL         CHG_M_IMAGE     = 1^1 - 1^0;
;  550          0547  1  MACRO           CHG_V_PARMS     = 1,1,1,0%;             ! PARAMETER(S) CHANGE
;  551          0548  1  LITERAL         CHG_M_PARMS     = 1^2 - 1^1;
;  552          0549  1  MACRO           CHG_V_QUALS     = 1,2,1,0%;             ! QUALIFIER(S) CHANGE
;  553          0550  1  LITERAL         CHG_M_QUALS     = 1^3 - 1^2;
;  554          0551  1  MACRO           CHG_V_MCRIGNORE = 1,3,1,0%;     ! IGNORE IF CLI IS MCR
;  555          0552  1  LITERAL         CHG_M_MCRIGNORE = 1^4 - 1^3;
;  556          0553  1
;  557          0554  1  MACRO           CHG_W_IMAGE     = 2,0,16,1%;    ! SRO TO NEW IMAGE
;  558          0555  1  MACRO           CHG_B_PARMCNT   = 4,0,8,0%;     ! MIN/MAX PARAMETER COUNTS
;  559          0556  1
;  560          0557  1  MACRO           CHG_V_MINPARM   = 4,0,4,0%;     ! MINIMUM NUMBER OF PARAMETERS REQUIRED
;  561          0558  1  MACRO           CHG_V_MAXPARM   = 4,4,4,0%;     ! MAXIMUM NUMBER OF PARAMETERS ALLOWED
;  562          0559  1
;  563          0560  1  MACRO           CHG_W_PARMS     = 5,0,16,1%;    ! SRO TO FIRST PARAMETER DESCRIPTOR
;  564          0561  1  MACRO           CHG_W_QUALS     = 7,0,16,1%;    ! SRO TO FIRST QUALIFIER DESCRIPTOR
;  565          0562  1  LITERAL         CHG_C_LENGTH    = 9;
;  566          0563  1  LITERAL         CHG_K_LENGTH    = 9;
;  567          0564  1
```

```
;  568          0565  1
;  569          0566  1
;  570          0567  1  !
;  571          0568  1  !       DEFINE IND STRUCTURE FOR CLI$INTERFACE ROUTINES
;  572          0569  1  !
;  573          0570  1
;  574          0571  1  !...$INTDEF
;  575          0572  1
;  576          0573  1  MACRO          INT_B_TYPE      = 0,0,8,0%;             ! TYPE OF REQUEST
;  577          0574  1  MACRO          INT_L_TABLES    = 4,0,32,0%;            ! ADDRESS OF COMMAND TABLES
;  578          0575  1
;  579          0576  1  MACRO          INT_W_ENTLEN    = 8,0,16,0%;    ! LENGTH OF ENTITY NAME
;  580          0577  1  MACRO          INT_L_ENTADDR   = 12,0,32,0%;           ! ADDRESS OF ENTITY NAME
;  581          0578  1
;  582          0579  1  MACRO          INT_W_RETLEN    = 8,0,16,0%;    ! LENGTH OF RETURN VALUE
;  583          0580  1  MACRO          INT_L_RETADDR   = 12,0,32,0%;           ! ADDRESS OF RETURN VALUE
;  584          0581  1  MACRO          INT_L_GETVM     = 16,0,32,0%;           ! ADDRESS OF LIB$GET_VM ROUTINE
;  585          0582  1  MACRO          INT_L_FREEVM    = 20,0,32,0%;           ! ADDRESS OF LIB$FREE_VM ROUTINE
;  586          0583  1  MACRO          INT_L_LIST      = 24,0,32,0%;           ! ADDRESS OF AUXILIARY ARGUMENT LIST
;  587          0584  1  LITERAL        INT_C_LENGTH    = 28;
;  588          0585  1  LITERAL        INT_K_LENGTH    = 28;
;  589          0586  1
;  590          0587  1  MACRO          INT_L_LISTLEN   = 0,0,32,0%;            ! LENGTH OF AUXILIARY ARGUMENT LIST
;  591          0588  1  MACRO          INT_L_PROMPTRTN = 4,0,32,0%;    ! ADDRESS OF PROMPT ROUTINE
;  592          0589  1  MACRO          INT_L_CONTINRTN = 8,0,32,0%;    ! ADDRESS OF CONTINUATION ROUTINE
;  593          0590  1  MACRO          INT_W_PMPTLEN   = 12,0,16,0%;   ! LENGTH OF PROMPT STRING
;  594          (591  1  MACRO          INT_L_PMPTADDR  = 16,0,32,0%;   ! ADDRESS OF PROMPT STRING
;  595          0592  1
;  596          0593  1  !
;  597          0594  1  !       Define shared message codes.
;  598          0595  1  !
;  599          0596  1
;  600        P 0597  1  $SHR_MESSAGES(msg,155,
;  601        P 0598  1          (badvalue,severe),                   ! Bad value for keyword
;  602        P 0599  1          (syntax,severe),                     ! Syntax error
;  603        P 0600  1          (openin,error),                      ! Unable to access input file
;  604        P 0601  1          (openout,severe),                    ! Unable to access output file
;  605        P 0602  1          (readerr,severe),                    ! Error reading input file
;  606        P 0603  1          (closedel,error),                    ! Unable to close input file
;  607        P 0604  1          (searchfail,error),                  ! Error searching for file
;  608        P 0605  1          (writeerr,error),                    ! Error writing file
;  609          0606  1          );
;  610          0607  1
;  611          0608  1  MACRO
;  612        M 0609  1      emsg(ident) =
;  613        M 0610  1          BEGIN
;  614        M 0611  1          %IF NOT %DECLARED(%NAME('msg$_',ident))
;  615        M 0612  1                %THEN EXTERNAL LITERAL %NAME('msg$_',ident); %FI
;  616        M 0613  1          %NAME('msg$_',ident)
;  617          0614  1          END%;
;  618          0615  1
;  619          0616  1  !** REQUIRE 'LIB$:SYMDEF';                      ! Symbol table definitions
;  620          0617  1  !-----
;  621          0618  1  !
;  622          0619  1  !       Symbol table management structures
;  623          0620  1  !
;  624          0621  1  ! IDENT V02-000
```

```
;  625     0622  1  !
;  626     0623  1  !-----
;  627     0624  1
;  628     0625  1  !
;  629     0626  1  !**************************************************************************
;  630     0627  1  !*                                                                        *
;  631     0628  1  !*  COPYRIGHT (c) 1978, 1980, 1982 BY                                     *
;  632     0629  1  !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                 *
;  633     0630  1  !*  ALL RIGHTS RESERVED.                                                   *
;  634     0631  1  !*                                                                        *
;  635     0632  1  !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
;  636     0633  1  !*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE         *
;  637     0634  1  !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER    *
;  638     0635  1  !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
;  639     0636  1  !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY     *
;  640     0637  1  !*  TRANSFERRED.                                                           *
;  641     0638  1  !*                                                                        *
;  642     0639  1  !*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE   *
;  643     0640  1  !*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT       *
;  644     0641  1  !*  CORPORATION.                                                           *
;  645     0642  1  !*                                                                        *
;  646     0643  1  !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS    *
;  647     0644  1  !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                *
;  648     0645  1  !*                                                                        *
;  649     0646  1  !*                                                                        *
;  650     0647  1  !**************************************************************************
;  651     0648  1  !
;  652     0649  1
;  653     0650  1
;  654     0651  1  !
;  655     0652  1  !       Define symbol table entry
;  656     0653  1  !
;  657     0654  1  !       (Length is SYM$C_FIXEDLEN + .SYM$B_SYMLEN)
;  658     0655  1  !
;  659     0656  1
;  660     0657  1  !...$SYMDEF
;  661     0658  1
;  662     0659  1  MACRO           SYM$L_LINK      = 0,0,32,0%;            ! LINK TO NEXT IN CHAIN
;  663     0660  1  MACRO           SYM$L_VALUE     = 4,0,32,0%;            ! VALUE OF SYMBOL
;  664     0661  1  MACRO           SYM$B_SYMLEN    = 8,0,8,0%;     ! LENGTH OF SYMBOL NAME
;  665     0662  1  LITERAL         SYM$C_FIXEDLEN  = 9;
;  666     0663  1  LITERAL         SYM$K_FIXEDLEN  = 9;            ! LENGTH OF FIXED PORTION OF ENTRY
;  667     0664  1  MACRO           SYM$T_SYMBOL    = 9,0,8,0%;     ! SYMBOL NAME
```

```
;  669          0665  1 !
;  670          0666  1 ! Table of contents
;  671          0667  1 !
;  672          0668  1
;  673          0669  1 FORWARD ROUTINE
;  674          0670  1     add_record,                     ! Add record to linked list
;  675          0671  1     find_record,                    ! Find record by number
;  676          0672  1     delete_list,                    ! Deallocate entire record list
;  677          0673  1     add_symbol,                     ! Add symbol to symbol table
;  678          0674  1     lookup_symbol,                  ! Lookup symbol in symbol table
;  679          0675  1     lookup_value,                   ! Lookup value in symbol table
;  680          0676  1     scan_symbols,                   ! Scan all symbols in symbol table
;  681          0677  1     delete_symbol,                  ! Delete symbol from symbol table
;  682          0678  1     delete_symbols,                 ! Delete entire symbol table
;  683          0679  1     allocate,                       ! Allocate dynamic storage
;  684          0680  1     deallocate;                     ! Deallocate dynamic storage
;  685          0681  1
;  686          0682  1 !
;  687          0683  1 ! Storage definitions
;  688          0684  1 !
;  689          0685  1
;  690          0686  1 GLOBAL
;  691          0687  1     symbol_header:      VECTOR [64]     ! List of listheads for symbol tables
;  692          0688  1                         INITIAL(REP 64 OF (0)); ! Set all listheads empty
;  693          0689  1
;  694          0690  1 !
;  695          0691  1 ! External routines
;  696          0692  1 !
;  697          0693  1
;  698          0694  1 EXTERNAL ROUTINE
;  699          0695  1     lib$get_vm: ADDRESSING_MODE(GENERAL),       ! Allocate dynamic storage
;  700          0696  1     lib$free_vm: ADDRESSING_MODE(GENERAL);      ! Deallocate dynamic storage
```

```
;  702          0697  1 GLOBAL ROUTINE add_record (listhead, address, length) =
;  703          0698  1
;  704          0699  1 !---
;  705          0700  1 !
;  706          0701  1 !       This routine adds a given data record to the
;  707          0702  1 !       end of a given linked list.
;  708          0703  1 !
;  709          0704  1 ! Inputs:
;  710          0705  1 !
;  711          0706  1 !       listhead = Address of listhead for list
;  712          0707  1 !       address = Address of data record
;  713          0708  1 !       length = Length of data record
;  714          0709  1 !
;  715          0710  1 ! Outputs:
;  716          0711  1 !
;  717          0712  1 !       routine = status (already signaled)
;  718          0713  1 !---
;  719          0714  1
;  720          0715  2 BEGIN
;  721          0716  2
;  722          0717  2 LOCAL
;  723          0718  2     new_entry:  REF VECTOR,             ! Address of newly allocated entry
;  724          0719  2     entry:      REF VECTOR;             ! Current entry address
;  725          0720  2
;  726          0721  2
;  727        P 0722  2 RETURN_IF_ERROR                       ! Allocate space; signal any error
;  728          0723  2         (allocate(.length+8,new_entry));
;  729          0724  2
;  730          0725  2 new_entry [1] = .length;                ! Set length into entry
;  731          0726  2 CH$MOVE(.length, .address, new_entry [2]);      ! Copy data into entry
;  732          0727  2
;  733          0728  2 entry = .listhead;                      ! Start at listhead itself
;  734          0729  2
;  735          0730  2 WHILE .entry [0] NEQ 0                  ! While not end of list
;  736          0731  2 DO
;  737          0732  2     entry = .entry [0];                 ! link to next in chain
;  738          0733  2
;  739          0734  2 entry [0] = .new_entry;                 ! set link of last entry to new one
;  740          0735  2 new_entry [0] = 0;                      ! and set new "end of list"
;  741          0736  2
;  742          0737  2 RETURN true;
;  743          0738  2
;  744          0739  1 END;


                                                .TITLE  SYMBOLS
                                                .IDENT  \V04-000\

                                                .PSECT  $GLOBAL$,NOEXE,2

                                00000000# 00000 SYMBOL_HEADER::
                                                .LONG   0[64]                                   ;

                                                .EXTRN  LIB$GET_VM, LIB$FREE_VM

                                                .PSECT  $CODE$,NOWRT,2
```

```
                                                 007C 00000          .ENTRY  ADD_RECORD, Save R2,R3,R4,R5,R6                 ; 0697
                                              5E                 04   C2 00002          SUBL2   #4, SP                          ;
                                                                 5E   DD 00005          PUSHL   SP                              ; 0723
                                        7E        0C        AC     08   C1 00007          ADDL3   #8, LENGTH, -(SP)               ;
                                              0000V     CF         02   FB 0000C          CALLS   #2, ALLOCATE                    ;
                                                        24         50   E9 00011          BLBC    STATUS, 3$                      ;
                                                        56         6E   D0 00014          MOVL    NEW_ENTRY, R6               ; 0725
                                                  04    A6    0C   AC   D0 00017          MOVL    LENGTH, 4(R6)                   ;
                                        08    A6        08    BC   0C   AC   28 0001C          MOVC3   LENGTH, @ADDRESS, 8(R6)     ; 0726
                                                        50         04   AC   D0 00023          MOVL    LISTHEAD, ENTRY             ; 0728
                                                                        60   D5 00027 1$:      TSTL    (ENTRY)                     ; 0730
                                                                        05   13 00029          BEQL    2$                          ;
                                                        50              60   D0 0002B          MOVL    (ENTRY), ENTRY              ; 0732
                                                                        F7   11 0002E          BRB     1$                          ;
                                                        60              56   D0 00030 2$:      MOVL    R6, (ENTRY)                 ; 0734
                                                                        66   D4 00033          CLRL    (R6)                        ; 0735
                                                        50              01   D0 00035          MOVL    #1, R0                      ; 0737
                                                                             04 00038 3$:      RET                                 ; 0739

; Routine Size:  57 bytes,    Routine Base:  $CODE$ + 0000
```

```
;  746     0740  1 GLOBAL ROUTINE find_record (listhead, number, retadr) =
;  747     0741  1
;  748     0742  1 !---
;  749     0743  1 !
;  750     0744  1 !       This routine locates a given record of data by
;  751     0745  1 !       record number in any given list.  The address
;  752     0746  1 !       returned is the address of the data itself.
;  753     0747  1 !
;  754     0748  1 ! Inputs:
;  755     0749  1 !
;  756     0750  1 !       listhead = Address of listhead of list
;  757     0751  1 !       number = Record number to find
;  758     0752  1 !       retadr = Address of longword to receive data address
;  759     0753  1 !
;  760     0754  1 ! Outputs:
;  761     0755  1 !
;  762     0756  1 !       routine = true if found, else false
;  763     0757  1 !---
;  764     0758  1
;  765     0759  2 BEGIN
;  766     0760  2
;  767     0761  2 LOCAL
;  768     0762  2     entry:      REF VECTOR;             ! Address of current entry
;  769     0763  2
;  770     0764  2 entry = ..listhead;                     ! Start at first entry
;  771     0765  2
;  772     0766  2 INCR i FROM 1 TO .number-1              ! Skip first number-1 entries
;  773     0767  2 DO
;  774     0768  3     BEGIN
;  775     0769  3     IF .entry EQL 0                     ! If premature end of list,
;  776     0770  3     THEN
;  777     0771  3         RETURN false;                   ! return entry not found
;  778     0772  3     entry = .entry [0];                 ! Skip to next entry in list
;  779     0773  2     END;
;  780     0774  2
;  781     0775  2 IF .entry EQL 0                         ! End of list
;  782     0776  2 THEN
;  783     0777  2     RETURN false;
;  784     0778  2
;  785     0779  2 .retadr = entry [2];                    ! Return address of data itself
;  786     0780  2 RETURN true;                            ! Return successful
;  787     0781  2
;  788     0782  1 END;


                                  0000 00000          .ENTRY  FIND_RECORD, Save nothing                       ; 0740
                        51    04  BC   D0 00002         MOVL    @LISTHEAD, ENTRY                                ; 0764
                                  50   D4 00006         CLRL    I                                               ; 0766
                                  07   11 00008         BRB     2$                                              ;
                                  51   D5 0000A 1$:     TSTL    ENTRY                                           ; 0769
                                  15   13 0000C         BEQL    3$                                              ;
                        51        61   D0 0000E         MOVL    (ENTRY), ENTRY                                  ; 0772
                     F4 50    08  AC   F2 00011 2$:     AOBLSS  NUMBER, I, 1$                                   ; 0766
                                  51   D5 00016         TSTL    ENTRY                                           ; 0775
```

```
                                        09  13 00018          BEQL    3$                                              ;
                                0C  BC      08  A1  9E 0001A          MOVAB   8(R1), @RETADR                          ; 0779
                                    50          01  D0  0001F          MOVL    #1, R0                                  ; 0780
                                                04  00022          RET                                             ;
                                                                                                                    ;
                                        50  D4  00023  3$:          CLRL    R0                                      ; 0782
                                            04  00025          RET                                             ;

; Routine Size:  38 bytes,    Routine Base:  $CODE$ + 0039
```

```
;  790          0783  1 GLOBAL ROUTINE delete_list (listhead) =
;  791          0784  1
;  792          0785  1 !---
;  793          0786  1 !
;  794          0787  1 !       This routine deallocates all storage associated
;  795          0788  1 !       with a given record list.
;  796          0789  1 !
;  797          0790  1 ! Inputs:
;  798          0791  1 !
;  799          0792  1 !       listhead = Address of listhead for list
;  800          0793  1 !
;  801          0794  1 ! Outputs:
;  802          0795  1 !
;  803          0796  1 !       None
;  804          0797  1 !---
;  805          0798  1
;  806          0799  2 BEGIN
;  807          0800  2
;  808          0801  2 LOCAL
;  809          0802  2     entry:      REF VECTOR;             ! Address of current entry
;  810          0803  2
;  811          0804  2 entry = ..listhead;                     ! Start at first entry
;  812          0805  2
;  813          0806  2 WHILE .entry NEQ 0                      ! For each entry in list,
;  814          0807  2 DO
;  815          0808  3     BEGIN
;  816          0809  3     LOCAL next_entry;
;  817          0810  3     next_entry = .entry [0];            ! Save pointer to next entry
;  818          0811  3     deallocate(.entry [1]+8, .entry);   ! Deallocate memory for entry
;  819          0812  3     entry = .next_entry;                ! Skip to next entry in list
;  820          0813  2     END;
;  821          0814  2
;  822          0815  2 .listhead = 0;                          ! Zero listhead
;  823          0816  2
;  824          0817  2 RETURN true;                            ! Success
;  825          0818  2
;  826          0819  1 END;


                                                000C 00000          .ENTRY  DELETE_LIST, Save R2,R3                 ; 0783
                                52          04  BC   D0 00002      MOVL    @LISTHEAD, ENTRY                        ; 0804
                                                14   13 00006 1$:   BEQL    2$                                      ; 0806
                                53              62   D0 00008      MOVL    (ENTRY), NEXT_ENTRY                     ; 0810
                                                52   DD 0000B      PUSHL   ENTRY                                   ; 0811
                        7E        04        A2    08   C1 0000D      ADDL3   #8, 4(ENTRY), -(SP)                     ;
                              0000V     CF          02   FB 00012      CALLS   #2, DEALLOCATE                      ;
                                52              53   D0 00017      MOVL    NEXT_ENTRY, ENTRY                       ; 0812
                                                EA   11 0001A      BRB     1$                                      ; 0806
                                            04  BC   D4 0001C 2$:   CLRL    @LISTHEAD                               ; 0815
                                50              01   D0 0001F      MOVL    #1, R0                                  ; 0817
                                                     04 00022      RET                                             ; 0819

; Routine Size:  35 bytes,    Routine Base:  $CODE$ + 005F
```

```
;  828          0820  1 GLOBAL ROUTINE add_symbol (table, name_desc, value) =
;  829          0821  1
;  830          0822  1 !---
;  831          0823  1 !
;  832          0824  1 !       This routine adds a given symbol name and value to
;  833          0825  1 !       the symbol table.  The symbol table list is sorted
;  834          0826  1 !       by symbol name.
;  835          0827  1 !
;  836          0828  1 ! Inputs:
;  837          0829  1 !
;  838          0830  1 !       table = Symbol table index
;  839          0831  1 !       name_desc = Address of descriptor of symbol name
;  840          0832  1 !       value = Value to be assigned to the symbol
;  841          0833  1 !
;  842          0834  1 ! Outputs:
;  843          0835  1 !
;  844          0836  1 !       r0 = status (already signaled)
;  845          0837  1 !---
;  846          0838  1
;  847          0839  2 BEGIN
;  848          0840  2
;  849          0841  2 MAP
;  850          0842  2     name_desc:  REF BBLOCK [DSC$K_S_BLN];! Address of name descriptor
;  851          0843  2
;  852          0844  2 LOCAL
;  853          0845  2     entry:      REF BBLOCK,             ! Address of symbol table entry
;  854          0846  2     location:   REF BBLOCK;             ! Address of closest symbol name
;  855          0847  2
;  856          0848  2 IF lookup_symbol (.table, .name_desc, location) ! If already in symbl table,
;  857          0849  2 THEN
;  858          0850  3     BEGIN
;  859          0851  3 !    SIGNAL(emsg(dupsym), 1, .name_desc);! signal user with bad symbol
;  860          0852  3     RETURN 1;                           ! return success
;  861          0853  2     END;
;  862          0854  2
;  863          0855  2
;  864        P 0856  2 RETURN_IF_ERROR                               ! Allocate a symbol entry
;  865          0857  2         (allocate(sym$c_fixedlen+.name_desc [dsc$w_length],entry));
;  866          0858  2
;  867          0859  2 entry [sym$l_value] = .value;                 ! Set value of symbol
;  868          0860  2 entry [sym$b_symlen]                          ! Set length of symbol
;  869          0861  2         = .name_desc [dsc$w_length];
;  870          0862  2
;  871          0863  2 CH$MOVE (.name_desc [dsc$w_length],           ! Copy symbol
;  872          0864  2         .name_desc [dsc$a_pointer],
;  873          0865  2         entry [sym$t_symbol]);
;  874          0866  2
;  875          0867  2 entry [sym$l_link]                           ! Link into symbol table
;  876          0868  2         = .location [sym$l_link];           ! in sorted order
;  877          0869  2 location [sym$l_link] = .entry;
;  878          0870  2
;  879          0871  2 RETURN true;
;  880          0872  2
;  881          0873  1 END;
```

```
                                    007C 00000          .ENTRY  ADD_SYMBOL, Save R2,R3,R4,R5,R6                 ; 0820
                              5E        08  C2 00002     SUBL2   #8, SP                                          ;
                                        5E  DD 00005     PUSHL   SP                                              ; 0848
                              52  08    AC  D0 00007     MOVL    NAME_DESC, R2                                   ;
                                        52  DD 0000B     PUSHL   R2                                              ;
                                  04    AC  DD 0000D     PUSHL   TABLE                                           ;
                        0000V CF        03  FB 00010     CALLS   #3, LOOKUP_SYMBOL                               ;
                              2C        50  E8 00015     BLBS    R0, 1$                                          ;
                                  04    AE  9F 00018     PUSHAB  ENTRY                                           ; 0857
                              7E        62  3C 0001B     MOVZWL  (R2), -(SP)                                     ;
                              6E        09  C0 0001E     ADDL2   #9, (SP)                                        ;
                        0000V CF        02  FB 00021     CALLS   #2, ALLOCATE                                    ;
                              1E        50  E9 00026     BLBC    STATUS, 2$                                      ;
                              56  04    AE  D0 00029     MOVL    ENTRY, R6                                       ; 0859
                            04  A6  0C    AC  D0 0002D     MOVL    VALUE, 4(R6)                                    ;
                            08  A6        62  90 00032     MOVB    (R2), 8(R6)                                     ; 0861
                09  A6      04  B2        62  28 000??     MOVC3   (R2), @4(R2), 9(R6)                             ; 0865
                              66  00    BE  D0 00?           MOVL    @LOCATION, (R6)                                 ; 0868
                            00  BE        56  D0 0C        MOVL    R6, @LOCATION                                   ; 0869
                              50        01  D0 00044 1$:     MOVL    #1, R0                                          ; 0871
                                        04  00047 2$:     RET                                                     ; 0873

; Routine Size:  72 bytes,    Routine Base:  $CODE$ + 0082
```

```
;  883          0874  1 GLOBAL ROUTINE lookup_symbol (table, name_desc, value) =
;  884          0875  1
;  885          0876  1 !---
;  886          0877  1 !
;  887          0878  1 !       This routine looks up a given symbol in the symbol
;  888          0879  1 !       table and returns the value associated with it.
;  889          0880  1 !       If the symbol is not found, then the address of the
;  890          0881  1 !       last entry preceeding the symbol in collation
;  891          0882  1 !       sequence is returned instead.
;  892          0883  1 !
;  893          0884  1 ! Inputs:
;  894          0885  1 !
;  895          0886  1 !       table = Symbol table index (1-n)
;  896          0887  1 !       name_desc = Descriptor of desired symbol name
;  897          0888  1 !       value = Address of longword to receive value if found
;  898          0889  1 !
;  899          0890  1 ! Outputs:
;  900          0891  1 !
;  901          0892  1 !       value = Value of symbol if found
;  902          0893  1 !       value = Address of previous entry if not found
;  903          0894  1 !       r0 = status
;  904          0895  1 !---
;  905          0896  1
;  906          0897  2 BEGIN
;  907          0898  2
;  908          0899  2 MAP
;  909          0900  2     name_desc:  REF BBLOCK [DSC$K_S_BLN];! Address of descriptor
;  910          0901  2
;  911          0902  2 LOCAL
;  912          0903  2     ptr:        REF BBLOCK;             ! Pointer into list
;  913          0904  2
;  914          0905  2 ptr = symbol_header [.table] - $BYTEOFFSET(sym$l_link); ! Start at listhead
;  915          0906  2 .value = .ptr;
;  916          0907  2
;  917          0908  2 WHILE (ptr = .ptr [sym$l_link]) NEQ 0   ! Until end of list
;  918          0909  2 DO
;  919          0910  2      CASE CH$COMPARE(.ptr [sym$b_symlen], ptr [sym$t_symbol],
;  920          0911  2                      .name_desc [dsc$w_length], .name_desc [dsc$a_pointer])
;  921          0912  2      FROM -1 TO 1 OF SET
;  922          0913  2      [-1]:  .value = .ptr;               ! Table symbol < user symbol
;  923          0914  3      [0]:  BEGIN                         ! Table symbol = user symbol
;  924          0915  3            .value = .ptr [sym$l_value]; !   Return value of symbol
;  925          0916  3            RETURN true;                 !   and exit successful
;  926          0917  2            END;
;  927          0918  2      [1]: RETURN false;                  ! Table symbol > user symbol
;  928          0919  2      TES;
;  929          0920  2
;  930          0921  2 RETURN false;                            ! return symbol not found
;  931          0922  2
;  932          0923  1 END;


                                005C 00000          .ENTRY  LOOKUP_SYMBOL, Save R2,R3,R4,R6          ; 0874
                     50    04   AC  D0 00002          MOVL    TABLE, R0                                ; 0905
```

```
                        54    0000'CF40  DE 00006          MOVAL   SYMBOL_HEADER[R0], PTR                          ;
                  0C    BC          54  D0 0000C          MOVL    PTR, @VALUE                                     ; 0906
                        56    08    AC  D0 00010          MOVL    NAME_DESC, R6                                   ; 0911
                        54          64  D0 00014 1$:      MOVL    (PTR), PTR                                      ; 0908
                                    20  13 00017          BEQL    3$                                              ;
                        50    08    A4  9A 00019          MOVZBL  8(PTR), R0                                      ; 0910
      08    BC          00    09    A4      50  2D 0001D  CMPC5   R0, 9(PTR), #0, @NAME_DESC, @4(R6)              ;
                                    04  B6 00024                                                                  ;
                                    11  1A 00026          BGTRU   3$                                              ;
                                    06  1E 00028          BGEQU   2$                                              ;
                  0C    BC          54  D0 0002A          MOVL    PTR, @VALUE                                     ; 0913
                                    E4  11 0002E          BRB     1$                                              ;
                  0C    BC    04    A4  D0 00030 2$:      MOVL    4(PTR), @VALUE                                  ; 0915
                        50          01  D0 00035          MOVL    #1, R0                                          ; 0916
                                        04 00038          RET                                                     ;
                                    50  D4 00039 3$:      CLRL    R0                                              ; 0923
                                        04 0003B          RET                                                     ;

; Routine Size:  60 bytes,    Routine Base:  $CODE$ + 00CA
```

```
;  934     0924  1 GLOBAL ROUTINE lookup_value (table, value, retdesc) =
;  935     0925  1
;  936     0926  1 !---
;  937     0927  1 !
;  938     0928  1 !       This routine locates the first occurrence of a symbol
;  939     0929  1 !       containing the specified value and returns a descriptor
;  940     0930  1 !       of the symbol associated with the value.
;  941     0931  1 !
;  942     0932  1 ! Inputs:
;  943     0933  1 !
;  944     0934  1 !       table = Symbol table index (1-n)
;  945     0935  1 !       value = Value to be looked up
;  946     0936  1 !       retdesc = Address of quadword to receive descriptor
;  947     0937  1 !
;  948     0938  1 ! Outputs:
;  949     0939  1 !
;  950     0940  1 !       routine = status
;  951     0941  1 !---
;  952     0942  1
;  953     0943  2 BEGIN
;  954     0944  2
;  955     0945  2 MAP
;  956     0946  2     retdesc:    REF VECTOR;             ! Address of descriptor
;  957     0947  2
;  958     0948  2 LOCAL
;  959     0949  2     ptr:        REF BBLOCK;             ! Pointer into list
;  960     0950  2
;  961     0951  2 ptr = .symbol_header [.table];          ! Start at first entry
;  962     0952  2
;  963     0953  2 WHILE .ptr NEQ 0                        ! Until end of list
;  964     0954  2 DO
;  965     0955  3     BEGIN
;  966     0956  3     IF .ptr [sym$l_value] EQL .value    ! If value matches,
;  967     0957  3     THEN
;  968     0958  4         BEGIN                           ! Return descriptor of name
;  969     0959  4         retdesc [0] = .ptr [sym$b_symlen];
;  970     0960  4         retdesc [1] = ptr [sym$t_symbol];
;  971     0961  4         RETURN true;                    ! and exit successful
;  972     0962  3         END;
;  973     0963  3     ptr = .ptr [sym$l_link];            ! If no match, go to next entry
;  974     0964  2     END;
;  975     0965  2
;  976     0966  2 RETURN false;                           ! return symbol not found
;  977     0967  2
;  978     0968  1 END;


                                    0000 00000          .ENTRY  LOOKUP_VALUE, Save nothing                      ; 0924
                          50    04    AC  D0 00002          MOVL    TABLE, R0                                       ; 0951
                          51  0000'CF40  D0 00006          MOVL    SYMBOL_HEADER[R0], PTR                          ;
                                    1D  13 0000C 1$:        BEQL    3$                                              ; 0953
                       08 AC      04    A1  D1 0000E          CMPL    4(PTR), VALUE                                   ; 0956
                                    11  12 00013          BNEQ    2$                                              ;
                          50    0C    AC  D0 00015          MOVL    RETDESC, R0                                     ; 0959
```

```
                        60      08    A1  9A 00019           MOVZBL  8(PTR), (R0)                    ;
                    04  A0      09    A1  9E 0001D           MOVAB   9(R1), 4(R0)                   ; 0960
                        50            01  D0 00022           MOVL    #1, R0                         ; 0961
                                      04  00025           RET                                    ;
                        51            61  D0 00026 2$:     MOVL    (PTR), PTR                     ; 0963
                                      E1  11 00029           BRB     1$                             ; 0953
                                      50  D4 0002B 3$:     CLRL    R0                             ; 0966
                                      04  0002D           RET                                    ; 0968

; Routine Size:  46 bytes,    Routine Base:  $CODE$ + 0106
```

```
;  980          0969  1 GLOBAL ROUTINE scan_symbols (table, action_routine) =
;  981          0970  1
;  982          0971  1 !---
;  983          0972  1 !
;  984          0973  1 !       This routine calls a specified action routine for
;  985          0974  1 !       each symbol in the specified symbol table.
;  986          0975  1 !
;  987          0976  1 ! Inputs:
;  988          0977  1 !
;  989          0978  1 !       table = Symbol table index (1-n)
;  990          0979  1 !       action_routine = Address of action routine to call
;  991          0980  1 !                        with the following argument list:
;  992          0981  1 !                                1) Address of descriptor of symbol name
;  993          0982  1 !                                2) Value associated with symbol
;  994          0983  1 !
;  995          0984  1 ! Outputs:
;  996          0985  1 !
;  997          0986  1 !       The status of the last action routine executed is returned.
;  998          0987  1 !---
;  999          0988  1
; 1000          0989  2 BEGIN
; 1001          0990  2
; 1002          0991  2 LOCAL
; 1003          0992  2     status,                         ! Catch-all status return bucket
; 1004          0993  2     desc:       VECTOR [2],         ! Descriptor of symbol name
; 1005          0994  2     ptr:        REF BBLOCK;         ! Address of current symbol entry
; 1006          0995  2
; 1007          0996  2 ptr = .symbol_header [.table];      ! Start at first entry
; 1008          0997  2
; 1009          0998  2 WHILE .ptr NEQ 0                    ! Until end of list,
; 1010          0999  2 DO
; 1011          1000  3     BEGIN                           ! Setup descriptor of name
; 1012          1001  3     desc [0] = .ptr [sym$b_symlen];
; 1013          1002  3     desc [1] = ptr [sym$t_symbol];
; 1014          1003  3     status = (.action_routine)(desc, .ptr [sym$l_value]); ! Call action routine
; 1015          1004  3     IF NOT .status THEN EXITLOOP;   ! If failed, exit unsuccessful
; 1016          1005  3     ptr = .ptr [sym$l_link];        ! Skip to next in chain
; 1017          1006  2     END;
; 1018          1007  2
; 1019          1008  2 RETURN .status;                     ! return successful
; 1020          1009  2
; 1021          1010  1 END;


                                0004 00000          .ENTRY  SCAN_SYMBOLS, Save R2                   ; 0969
                             5E       08  C2 00002          SUBL2   #8, SP                          ;
                             50   04  AC  D0 00005          MOVL    TABLE, R0                       ; 0996
                             52 0000'CF40  D0 00009          MOVL    SYMBOL_HEADER[R0], PTR          ;
                                      1B  13 0000F 1$:       BEQL    2$                              ; 0998
                             6E       08  A2  9A 00011          MOVZBL  8(PTR), DESC                ; 1001
                          04 AE       09  A2  9E 00015          MOVAB   9(R2), DESC+4               ; 1002
                                      04  A2  DD 0001A          PUSHL   4(PTR)                      ; 1003
                                      04  AE  9F 0001D          PUSHAB  DESC                        ;
                          08 BC           02  FB 00020          CALLS   #2, @ACTION_ROUTINE         ;
```

```
                        05              50  E9 00024            BLBC    STATUS, 2$                              ; 1004
                        52              62  D0 00027            MOVL    (PTR), PTR                              ; 1005
                                        E3  11 0002A            BRB     1$                                      ; 0998
                                            04 0002C 2$:        RET                                             ; 1010

; Routine Size:  45 bytes,    Routine Base:  $CODE$ + 0134
```

```
;  1023          1011  1 GLOBAL ROUTINE delete_symbol (table, name_desc) =
;  1024          1012  1
;  1025          1013  1 !---
;  1026          1014  1 !
;  1027          1015  1 !       This routine deletes a given symbol from the symbol
;  1028          1016  1 !       table.
;  1029          1017  1 !
;  1030          1018  1 ! Inputs:
;  1031          1019  1 !
;  1032          1020  1 !       table = Symbol table index (1-n)
;  1033          1021  1 !       name_desc = Descriptor of symbol name to be deleted
;  1034          1022  1 !
;  1035          1023  1 ! Outputs:
;  1036          1024  1 !
;  1037          1025  1 !       r0 = true if deleted, false if not found
;  1038          1026  1 !---
;  1039          1027  1
;  1040          1028  2 BEGIN
;  1041          1029  2
;  1042          1030  2 MAP
;  1043          1031  2     name_desc:  REF BBLOCK [DSC$K_S_BLN];! Address of descriptor
;  1044          1032  2
;  1045          1033  2 LOCAL
;  1046          1034  2     prev:       REF BBLOCK,             ! Pointer to previous symbol
;  1047          1035  2     ptr:        REF BBLOCK;             ! Pointer into list
;  1048          1036  2
;  1049          1037  2 ptr = symbol_header [.table] - $BYTEOFFSET(sym$l_link); ! Start at listhead
;  1050          1038  2 prev = .ptr;                            !  Ditto
;  1051          1039  2
;  1052          1040  2 WHILE (ptr = .ptr [sym$l_link]) NEQ 0   ! Until end of list
;  1053          1041  2 DO
;  1054          1042  2     CASE CH$COMPARE(.ptr [sym$b_symlen], ptr [sym$t_symbol],
;  1055          1043  2                     .name_desc [dsc$w_length], .name_desc [dsc$a_pointer])
;  1056          1044  2     FROM -1 TO 1 OF SET
;  1057          1045  2     [-1]:  prev = .ptr;                 ! Table symbol < user symbol
;  1058          1046  3     [0]:  BEGIN                         ! Table symbol = user symbol
;  1059          1047  3           prev [sym$l_link] = .ptr [sym$l_link];       ! Delink it
;  1060          1048  3           RETURN deallocate (sym$c_fixedlen+.ptr[sym$b_symlen], .ptr); ! free VM
;  1061          1049  2           END;
;  1062          1050  2     [1]: RETURN false;                  ! Table symbol > user symbol
;  1063          1051  2     TES;
;  1064          1052  2
;  1065          1053  2
;  1066          1054  2 RETURN false;                          ! return symbol not found
;  1067          1055  2
;  1068          1056  1 END;


                                00DC 00000          .ENTRY  DELETE_SYMBOL, Save R2,R3,R4,R6,R7          ; 1011
                     50    04   AC  D0 00002          MOVL    TABLE, R0                                   ; 1037
                     54 0000'CF40  DE 00006          MOVAL   SYMBOL_HEADER[R0], PTR                      ;
                     57       54  D0 0000C          MOVL    PTR, PREV                                   ; 1038
                     56    08   AC  D0 0000F          MOVL    NAME_DESC, R6                               ; 1043
                     54       64  D0 00013 1$:       MOVL    (PTR), PTR                                  ; 1040
```

```
                                      28 13 00016        BEQL    3$                                    ;
                          50     08   A4 9A 00018        MOVZBL  8(PTR), R0                            ; 1042
    08  BC          00    09  A4      50 2D 0001C        CMPC5   R0, 9(PTR), #0, @NAME_DESC, @4(R6)    ;
                                 04   B6    00023                                                      ;
                                      19 1A 00025        BGTRU   3$                                    ;
                                      05 1E 00027        BGEQU   2$                                    ;
                          57          54 D0 00029        MOVL    PTR, PREV                             ; 1045
                                      E5 11 0002C        BRB     1$                                    ;
                          67          64 D0 0002E 2$:    MOVL    (PTR), (PREV)                         ; 1047
                                      54 DD 00031        PUSHL   PTR                                   ; 1048
                          7E     08   A4 9A 00033        MOVZBL  8(PTR), -(SP)                         ;
                          6E          09 C0 00037        ADDL2   #9, (SP)                              ;
                    0000V CF          02 FB 0003A        CALLS   #2, DEALLOCATE                        ;
                                         04 0003F        RET                                           ;
                                      50 D4 00040 3$:    CLRL    R0                                    ; 1056
                                         04 00042        RET                                           ;

; Routine Size:  67 bytes,    Routine Base:  $CODE$ + 0161
```

```
;  1070          1057  1 GLOBAL ROUTINE delete_symbols (table) =
;  1071          1058  1
;  1072          1059  1 !---
;  1073          1060  1 !
;  1074          1061  1 !       This routine deallocates all symbols in a specified
;  1075          1062  1 !       symbol table.
;  1076          1063  1 !
;  1077          1064  1 ! Inputs:
;  1078          1065  1 !
;  1079          1066  1 !       table = Symbol table index (1-n)
;  1080          1067  1 !
;  1081          1068  1 ! Outputs:
;  1082          1069  1 !
;  1083          1070  1 !       None
;  1084          1071  1 !---
;  1085          1072  1
;  1086          1073  2 BEGIN
;  1087          1074  2
;  1088          1075  2 LOCAL
;  1089          1076  2     ptr:        REF BBLOCK;             ! Address of current entry
;  1090          1077  2
;  1091          1078  2 ptr = .symbol_header [.table];         ! Start at first entry
;  1092          1079  2
;  1093          1080  2 WHILE .ptr NEQ 0                       ! Until end of list,
;  1094          1081  2 DO
;  1095          1082  3     BEGIN
;  1096          1083  3     LOCAL next_entry;
;  1097          1084  3     next_entry = .ptr [sym$l_link];     ! Save pointer to next entry
;  1098          1085  3     deallocate(sym$c_fixedlen+.ptr [sym$b_symlen], .ptr);       ! Deallocate entry
;  1099          1086  3     ptr = .next_entry;                  ! Point to next entry in list
;  1100          1087  2     END;
;  1101          1088  2
;  1102          1089  2 symbol_header [.table] = 0;            ! Zero listhead
;  1103          1090  2
;  1104          1091  2 RETURN true;
;  1105          1092  2
;  1106          1093  1 END;


                                                001C 00000          .ENTRY  DELETE_SYMBOLS, Save R2,R3,R4                   ; 1057
                                          52    04   AC  D0 00002    MOVL    TABLE, R2                                       ; 1078
                                          53 0000'CF42  D0 00006    MOVL    SYMBOL_HEADER[R2], PTR                          ;
                                                   16  13 0000C 1$:  BEQL    2$                                              ; 1080
                                          54       63  D0 0000E    MOVL    (PTR), NEXT_ENTRY                               ; 1084
                                                   53  DD 00011    PUSHL   PTR                                             ; 1085
                                          7E    08   A3  9A 00013    MOVZBL  8(PTR), -(SP)                                   ;
                                          6E       09  C0 00017    ADDL2   #9, (SP)                                        ;
                                    0000V CF       02  FB 0001A    CALLS   #2, DEALLOCATE                                  ;
                                          53       54  D0 0001F    MOVL    NEXT_ENTRY, PTR                                 ; 1086
                                                   E8  11 00022    BRB     1$                                              ; 1080
                                             0000'CF42  D4 00024 2$:  CLRL    SYMBOL_HEADER[R2]                               ; 1089
                                          50       01  D0 00029    MOVL    #1, R0                                          ; 1091
                                                   04  0002C      RET                                                     ; 1093
```

; Routine Size: 45 bytes,   Routine Base: $CODE$ + 01A4

```
;  1108          1094  1 GLOBAL ROUTINE allocate (bytes, address) =
;  1109          1095  1
;  1110          1096  1 !---
;  1111          1097  1 !
;  1112          1098  1 !       Allocate dynamic storage and return the address.
;  1113          1099  1 !       If an error occurs, the error is signaled.
;  1114          1100  1 !
;  1115          1101  1 ! Inputs:
;  1116          1102  1 !
;  1117          1103  1 !       bytes = Number of bytes to allocate
;  1118          1104  1 !       address = Longword to receive address of storage
;  1119          1105  1 !
;  1120          1106  1 ! Outputs:
;  1121          1107  1 !
;  1122          1108  1 !       address = Address of storage
;  1123          1109  1 !---
;  1124          1110  1
;  1125          1111  2 BEGIN
;  1126          1112  2
;  1127          1113  2 LOCAL
;  1128          1114  2     status;
;  1129          1115  2
;  1130          1116  2 status = lib$get_vm(bytes,.address);
;  1131          1117  2
;  1132          1118  2 IF NOT .status                    ! if unsuccessful,
;  1133          1119  2 THEN
;  1134          1120  2     SIGNAL(.status);             ! then signal the error
;  1135          1121  2
;  1136          1122  2 RETURN .status;                  ! return with status;
;  1137          1123  2
;  1138          1124  1 END;


                                    0004 00000          .ENTRY  ALLOCATE, Save R2                       ; 1094
                                08  AC  DD 00002          PUSHL   ADDRESS                                 ; 1116
                                04  AC  9F 00005          PUSHAB  BYTES                                   ;
                    00000000G   00      02  FB 00008          CALLS   #2, LIB$GET_VM                      ;
                                52      50  D0 0000F          MOVL    R0, STATUS                          ;
                                09      52  E8 00012          BLBS    STATUS, 1$                          ; 1118
                                        52  DD 00015          PUSHL   STATUS                              ; 1120
                    00000000G   00      01  FB 00017          CALLS   #1, LIB$SIGNAL                      ;
                                50      52  D0 0001E 1$:       MOVL    STATUS, R0                          ; 1122
                                        04  00021              RET                                         ; 1124

; Routine Size:  34 bytes,    Routine Base:  $CODE$ + 01D1
```

```
;  1140          1125  1 GLOBAL ROUTINE deallocate (bytes, address) =
;  1141          1126  1
;  1142          1127  1 !---
;  1143          1128  1 !
;  1144          1129  1 !       Deallocate dynamic storage.
;  1145          1130  1 !       If an error occurs, the error is signaled.
;  1146          1131  1 !
;  1147          1132  1 ! Inputs:
;  1148          1133  1 !
;  1149          1134  1 !       bytes = Number of bytes to deallocate
;  1150          1135  1 !       address = Address of storage to deallocate
;  1151          1136  1 !
;  1152          1137  1 ! Outputs:
;  1153          1138  1 !
;  1154          1139  1 !       None
;  1155          1140  1 !---
;  1156          1141  1
;  1157          1142  2 BEGIN
;  1158          1143  2
;  1159          1144  2 LOCAL
;  1160          1145  2     status;
;  1161          1146  2
;  1162          1147  2 status = lib$free_vm(bytes,address);
;  1163          1148  2
;  1164          1149  2 IF NOT .status                    ! if unsuccessful,
;  1165          1150  2 THEN
;  1166          1151  2     SIGNAL(.status);             ! then signal the error
;  1167          1152  2
;  1168          1153  2 RETURN .status;                  ! return with status;
;  1169          1154  2
;  1170          1155  1 END;


                                      0004 00000          .ENTRY  DEALLOCATE, Save R2                     ;  1125
                                08    AC  9F 00002        PUSHAB  ADDRESS                                 ;  1147
                                04    AC  9F 00005        PUSHAB  BYTES                                   ;
                    00000000G   00        02  FB 00008    CALLS   #2, LIB$FREE_VM                         ;
                                52        50  D0 0000F    MOVL    R0, STATUS                              ;
                                09        52  E8 00012    BLBS    STATUS, 1$                              ;  1149
                                          52  DD 00015    PUSHL   STATUS                                  ;  1151
                    00000000G   00        01  FB 00017    CALLS   #1, LIB$SIGNAL                          ;
                                50        52  D0 0001E 1$: MOVL    STATUS, R0                              ;  1153
                                          04  00021        RET                                             ;  1155

; Routine Size:  34 bytes,    Routine Base:  $CODE$ + 01[illegible]
```

```
; 1172          1156  1 END
; 1173          1157  0 ELUDOM


                                                        .EXTRN  LIB$SIGNAL

;                               PSECT SUMMARY
;
;       Name                    Bytes                   Attributes
;
;  $GLOBAL$                        256  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $CODE$                          533  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        22         0       581          00:01.0


;                                   COMMAND QUALIFIERS

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

; Size:         533 code + 256 data bytes
; Run Time:       00:19.7
; Elapsed Time:   00:49.6
; Lines/CPU Min:   3529
; Lexemes/CPU-Min: 22581
; Memory Used:  90 pages
; Compilation Complete
```

AEDCLENUP
LIS

SYMBOLS
LIS

UTILITY
LIS

ACLEDTDEF
REQ

ACLEDT

SETSHOACL
MAP

ACLEDT
MAP

AEDLCLDEE
SDL

SUMMARY
LIS

SYMBOLMSG
LIS

AEDDECODE
LIS