```
DDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBB      UUU         UUU     GGGGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUU         UUU     GGGGGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUU         UUU     GGGGGGGGGGGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEEEEEEEEEE      BBBBBBBBBBBB     UUU         UUU GGG
DDD         DDD EEEEEEEEEEE      BBBBBBBBBBBB     UUU         UUU GGG
DDD         DDD EEEEEEEEEEE      BBBBBBBBBBBB     UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG    GGGGGGGGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG    GGGGGGGGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG    GGGGGGGGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG          GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG          GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG          GGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUUUUUUUUUUUUUUU    GGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUUUUUUUUUUUUUUU    GGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUUUUUUUUUUUUUUU    GGGGGGGGG
```

FILEID**DBGEXC

```
DDDDDDDD    BBBBBBBB      GGGGGGGG  EEEEEEEEEE  XX      XX    CCCCCCCC
DDDDDDDD    BBBBBBBB      GGGGGGGG  EEEEEEEEEE  XX      XX    CCCCCCCC
DD      DD  BB      BB  GG          EE          XX      XX  CC
DD      DD  BB      BB  GG          EE          XX      XX  CC
DD      DD  BB      BB  GG          EE            XX  XX    CC
DD      DD  BB      BB  GG          EE            XX  XX    CC
DD      DD  BBBBBBBB    GG          EEEEEEEE        XX      CC
DD      DD  BBBBBBBB    GG          EEEEEEEE        XX      CC
DD      DD  BB      BB  GG  GGGGGG  EE            XX  XX    CC
DD      DD  BB      BB  GG  GGGGGG  EE            XX  XX    CC
DD      DD  BB      BB  GG      GG  EE          XX      XX  CC              ....
DD      DD  BB      BB  GG      GG  EE          XX      XX  CC              ....
DDDDDDDD    BBBBBBBB      GGGGGG    EEEEEEEEEE  XX      XX    CCCCCCCC    ....
DDDDDDDD    BBBBBBBB      GGGGGG    EEEEEEEEEE  XX      XX    CCCCCCCC    ....

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 DBGEXC (IDENT = 'V04-000') =
;     2        0002  0
;     3        0003  1 BEGIN
;     4        0004  1
;     5        0005  1 !**************************************************************************
;     6        0006  1 !*                                                                        *
;     7        0007  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                              *
;     8        0008  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.               *
;     9        0009  1 !*  ALL RIGHTS RESERVED.                                                 *
;    10        0010  1 !*                                                                        *
;    11        0011  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
;    12        0012  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE *
;    13        0013  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER *
;    14        0014  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
;    15        0015  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY *
;    16        0016  1 !*  TRANSFERRED.                                                           *
;    17        0017  1 !*                                                                        *
;    18        0018  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE *
;    19        0019  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT *
;    20        0020  1 !*  CORPORATION.                                                           *
;    21        0021  1 !*                                                                        *
;    22        0022  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS *
;    23        0023  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.               *
;    24        0024  1 !*                                                                        *
;    25        0025  1 !*                                                                        *
;    26        0026  1 !**************************************************************************
;    27        0027  1 !
;    28        0028  1 ! WRITTEN BY
;    29        0029  1 !       Carol Peters, 05 Oct 1976: Version 01
;    30        0030  1 !
;    31        0031  1 ! MODULE FUNCTION
;    32        0032  1 !       This module contains DEBUG's Primary Exception Handler and associated
;    33        0033  1 !       routines.  DEBUG's Primary Handler is actually located in module
;    34        0034  1 !       DBGSTART, but that code calls DBG$EXC_HANDLER in this module to do
;    35        0035  1 !       most of the work of handling primary exceptions.
;    36        0036  1 !
;    37        0037  1 ! Revision History
;    38        0038  1 !       R. Title        May 1983        Most of the original code was
;    39        0039  1 !                                       removed from this module.
;    40        0040  1 !       P. Sager        Aug 1983        Added a read error count to force
;    41        0041  1 !                                       DEBUG to take exit.
;    42        0042  1 !       P. Sager        Aug 1983        Added the Facility code and Bit 15
;    43        0043  1 !                                       in Message number test in
;    44        0044  1 !                                       DBG$EXCEPTION_IS_FAULT and
;    45        0045  1 !                                       DBG$PUTMSG.  This is also a bug
;    46        0046  1 !                                       reported by our user through SPR.
;    47        0047  1 REQUIRE 'SRC$:DBGPROLOG.REQ';
;    48        0181  1
;    49        0182  1 LIBRARY 'LIB$:DBGGEN.L32';
;    50        0183  1
;    51        0184  1 FORWARD ROUTINE
;    52        0185  1     DBG$CREATE_VIRTUAL_KEYBOARD,        ! Initialize keypad input
;    53        0186  1     DBG$COMMAND_PROC : NOVALUE,         ! Accepts a command from the user
;    54        0187  1     DBG$EXC_HANDLER,                    ! Handles DEBUG set exception conditions
;    55        0188  1     DBG$EXCEPTION_IS_FAULT,             ! If exception is fault, true
;    56        0189  1     DBG$PUTMSG : NOVALUE;               ! Checks exception type before calling
;    57        0190  1                                         !       SYS$PUTMSG
```

```
;    59          0191  1 EXTERNAL ROUTINE
;    60          0192  1     DBG$INIT_STEP : NOVALUE,                ! Reset step level to what is
;    61          0193  1     DBG$KEY_INITIALIZE: NOVALUE,            ! Keypad initialization
;    62          0194  1     DBG$SET_STP_LVL : NOVALUE,              ! Needed at BEGIN_OF_COMMAND
;    63          0195  1     DBG$REL_MEMORY : NOVALUE,               ! Releases free storage
;    64          0196  1     DBG$GET_MEMORY,                         ! Allocates free storage
;    65          0197  1     DBG$OUT_MESSAGE : NOVALUE,              ! Writes string descriptor message
;    66          0198  1                                             !       to DBG$OUTPUT
;    67          0199  1     DBG$NCONTROL : NOVALUE,                 ! New debugger control routine
;    68          0200  1     DBG$CIS_ADD,                            ! Add a link to the cis
;    69          0201  1     DBG$CIS_REMOVE,                         ! Remove a link from the cis
;    70          0202  1     DBG$FINAL_HANDL,                        ! Call frame exception handler
;    71          0203  1     DBG$SCR_GENERATE_SCREEN: NOVALUE,       ! Generate all automatic screen displays
;    72          0204  1     DBG$SCR_OUTPUT_SCREEN: NOVALUE,         ! Output Screen Displays to terminal
;    73          0205  1     DBG$EXCEPTION_HANDLER,                  ! New Event exception handler
;    74          0206  1     DBG$ACTIVATE_EVENTS : NOVALUE,          ! Activate events
;    75          0207  1     SMG$CREATE_KEY_TABLE,                   ! Keypad initialization routine
;    76          0208  1     SMG$CREATE_VIRTUAL_KEYBOARD,            ! Keypad initialization routine
;    77          0209  1     SMG$READ_COMPOSED_LINE,                 ! Keypad input
;    78          0210  1     SYS$PUTMSG: ADDRESSING_MODE(GENERAL);   ! System output message routine
;    79          0211  1
;    80          0212  1 EXTERNAL
;    81          0213  1     DBG$GB_KEYPAD_INPUT: BYTE,              ! TRUE if keypad input is enabled
;    82          0214  1     DBG$GL_KEYBOARD_ID,
;    83          0215  1     DBG$GB_LANGUAGE: BYTE,                  ! Language setting
;    84          0216  1     DBG$GL_KEY_TABLE_ID,
;    85          0217  1     DBG$GB_DEF_OUT: VECTOR[,BYTE],          ! Current OUTPUT configuration
;    86          0218  1     DBG$GL_EXIT_STATUS,                     ! Last known user error status
;    87          0219  1     DBG$GL_INPRAB: BLOCK[,BYTE],            ! RAB for 'INPUT'
;    88          0220  1     DBG$GL_OUTPRAB: BLOCK[,BYTE],           ! RAB for 'OUTPUT'
;    89          0221  1     DBG$GL_LOGRAB: BLOCK[,BYTE],            ! RAB for LOG file
;    90          0222  1     DBG$GL_LOG_BUF,                         ! Ptr to log filespec
;    91          0223  1     DBG$GL_READERR_CNT,                     ! Read error count
;    92          0224  1     DBG$GL_SCREEN_MODE,                     ! Set to TRUE if screen mode is active
;    93          0225  1     DBG$GL_CISHEAD: REF CIS$LINK,           ! Head of command input stream
;    94          0226  1     DBG$GV_CONTROL: DBG$CONTROL_FLAGS;      ! DEBUG control bits
;    95          0227  1
;    96          0228  1 ! Declare a global which is used to store the address of the exit handler
;    97          0229  1 ! routine declared by SMG$CREATE_VIRTUAL_KEYBOARD.
;    98          0230  1 !
;    99          0231  1 GLOBAL
;   100          0232  1     DBG$GL_SMG_EXIT_HANDLER : INITIAL(0);
;   101          0233  1
;   102          0234  1 ! Declare an own variable which says whether keypad initialization
;   103          0235  1 ! has been done yet.
;   104          0236  1 !
;   105          0237  1 OWN
;   106          0238  1     KEYPAD_INITIALIZATION_DONE: INITIAL(0);
;   107          0239  1
;   108          0240  1 EXTERNAL LITERAL
;   109          0241  1     SMG$_EOF;                               ! End-of-file code
;   110          0242  1
;   111          0243  1 MACRO
;   112          0244  1     ! INP_READ_ERROR signals any RMS error encountered when reading input
;   113          0245  1     !
;   114        M 0246  1     INP_READ_ERROR =
;   115        M 0247  1                 BEGIN
```

```
;  116          M 0248  1                   LOCAL
;  117          M 0249  1                           FAB_PTR  : REF $FAB_DECL,
;  118          M 0250  1                           MSG_DESC : BLOCK [8,BYTE];
;  119          M 0251  1
;  120          M 0252  1                   FAB_PTR = .INPRAB [RAB$L_FAB];
;  121          M 0253  1                   MSG_DESC [DSC$W_LENGTH]  = .FAB_PTR [FAB$B_FNS];
;  122          M 0254  1                   MSG_DESC [DSC$A_POINTER] = .FAB_PTR [FAB$L_FNA];
;  123          M 0255  1
;  124          M 0256  1                   SIGNAL ((SHR$_READERR + DBG_FAC_CODE) OR FATAL_BIT, 1, MSG_DESC,
;  125          M 0257  1                           .INPRAB [RAB$L_STS], .INPRAB [RAB$L_STV]);
;  126            0258  1                   END %,
;  127            0259  1
;  128            0260  1
;  129            0261  1           ! LOG_WRITE_ERROR signals any RMS error encountered in writing to the LOG
;  130            0262  1           ! file.
;  131            0263  1           !
;  132          M 0264  1           LOG_WRITE_ERROR =
;  133          M 0265  1                   BEGIN
;  134          M 0266  1
;  135          M 0267  1                   LOCAL
;  136          M 0268  1                           FAB_PTR : REF $FAB_DECL,
;  137          M 0269  1                           MSG_DESC : BLOCK [8,BYTE];
;  138          M 0270  1
;  139          M 0271  1                   FAB_PTR = .DBG$GL_LOGRAB [RAB$L_FAB];
;  140          M 0272  1                   IF .DBG$GL_LOG_BUF NEQ 0
;  141          M 0273  1                   THEN
;  142          M 0274  1                           BEGIN
;  143          M 0275  1                           MSG_DESC [DSC$W_LENGTH]  = .FAB_PTR [FAB$B_FNS];
;  144          M 0276  1                           MSG_DESC [DSC$A_POINTER] = .FAB_PTR [FAB$L_FNA];
;  145          M 0277  1                           END
;  146          M 0278  1                   ELSE
;  147          M 0279  1                           BEGIN
;  148          M 0280  1                           MSG_DESC [DSC$W_LENGTH]  = .FAB_PTR [FAB$B_DNS];
;  149          M 0281  1                           MSG_DESC [DSC$A_POINTER] = .FAB_PTR [FAB$L_DNA];
;  150          M 0282  1                           END;
;  151          M 0283  1
;  152          M 0284  1                   SIGNAL (SHR$_WRITEERR + DBG_FAC_CODE, 1, MSG_DESC,
;  153          M 0285  1                           .DBG$GL_LOGRAB[RAB$L_STS], .DBG$GL_LOGRAB[RAB$L_STV]);
;  154            0286  1                   END %;
```

```
;   156          0287  1 GLOBAL ROUTINE DBG$CREATE_VIRTUAL_KEYBOARD =
;   157          0288  1 !
;   158          0289  1 ! FUNCTION
;   159          0290  1 !
;   160          0291  1 !       This routine initializes the keypad input data structures.
;   161          0292  1 !       The routine is just a cover routine for the RTL routine
;   162          0293  1 !       SMG$CREATE_VIRTUAL_KEYBOARD. This initialization routine
;   163          0294  1 !       is called once from DBG$COMMAND_PROC, the first time we
;   164          0295  1 !       get input after mode has been set to "KEYPAD". The routine
;   165          0296  1 !       is called again from the exit handler in DBGSTART. The
;   166          0297  1 !       reason for this is that the keypad routines declare an exit
;   167          0298  1 !       handler that disables keypad input, and we need to re-enable
;   168          0299  1 !       it from our exit handler so that keypad input continues to
;   169          0300  1 !       work after running to the end of the program.
;   170          0301  1 !
;   171          0302  1 ! INPUTS
;   172          0303  1 !       none
;   173          0304  1 !
;   174          0305  1 ! OUTPUTS
;   175          0306  1 !       The global variable DBG$GL_KEYBOARD_ID is set.
;   176          0307  1 !       A status is returned (STS$K_SUCCESS if all goes well).
;   177          0308  1 !
;   178          0309  2     BEGIN
;   179          0310  2     OWN
;   180          0311  2         desblk: VECTOR[4],
;   181          0312  2         dummy1,
;   182          0313  2         dummy2;
;   183          0314  2     LOCAL
;   184          0315  2         filespec: dbg$stg_desc,
;   185          0316  2         forward_link: REF VECTOR[4],
;   186          0317  2         save_link,
;   187          0318  2         status;
;   188          0319  2
;   189          0320  2     ! Initialize the block that is passed to the "declare exit handler"
;   190          0321  2     ! and "cancel exit handler" system services.
;   191          0322  2     !
;   192          0323  2     desblk[0] = 0;
;   193          0324  2     desblk[1] = dummy1;
;   194          0325  2     desblk[2] = 1;
;   195          0326  2     desblk[3] = dummy2;
;   196          0327  2
;   197          0328  2     ! Initialize the file spec that is passed in to the
;   198          0329  2     ! SMG$CREATE_VIRTUAL_KEYBOARD routine.
;   199          0330  2     ! We supply the file name to open for input - either DBG$INPUT,
;   200          0331  2     ! or if that fails, then SYS$INPUT.
;   201          0332  2     !
;   202          0333  2     filespec[dsc$b_class] = dsc$k_class_s;
;   203          0334  2     filespec[dsc$b_dtype] = dsc$k_dtype_t;
;   204          0335  2     filespec[dsc$w_length] = 9;
;   205          0336  2     filespec[dsc$a_pointer] = UPLIT BYTE(%ASCII 'DBG$INPUT');
;   206          0337  2
;   207          0338  2     ! Declare a temporary exit handler for the purpose of finding out
;   208          0339  2     ! the most recently declared exit handler. We will need to know this
;   209          0340  2     ! to discover if SMG$CREATE_VIRTUAL_KEYBOARD set up a new exit handler.
;   210          0341  2     !
;   211          0342  2     $dclexh (desblk = desblk);
;   212          0343  2     save_link = .desblk[0];
```

```
;  213          0344  2     $canexh (desblk = desblk);
;  214          0345  2
;  215          0346  2     ! Call the routine that initializes the keypad input data
;  216          0347  2     ! structures. If this fails with DBG$INPUT as the input device,
;  217          0348  2     ! then call it with SYS$INPUT.
;  218          0349  2     !
;  219          0350  2     status = smg$create_virtual_keyboard(dbg$gl_keyboard_id, filespec);
;  220          0351  2     IF NOT .status
;  221          0352  2     THEN
;  222          0353  3         BEGIN
;  223          0354  3         filespec[dsc$a_pointer] = UPLIT BYTE(%ASCII 'SYS$INPUT');
;  224          0355  3         status = smg$create_virtual_keyboard(dbg$gl_keyboard_id, filespec);
;  225          0356  2         END;
;  226          0357  2
;  227          0358  2     ! We want to get rid of the exit handler that was declared by
;  228          0359  2     ! SMG$CREATE_VIRTUAL_KEYBOARD, if indeed it declared one.
;  229          0360  2     ! We first declare a temporary exit handler,
;  230          0361  2     ! so we can pick up the address of the most recent exit handler
;  231          0362  2     ! from the forward link. If this address is different from the
;  232          0363  2     ! one in SAVE_LINK which we determined before the call to
;  233          0364  2     ! the SMG routine, then the SMG routine set up a new handler.
;  234          0365  2     ! In that case, we get rid of the handler here. We save the
;  235          0366  2     ! handler routine so we can call it ourselves when DEBUG exits.
;  236          0367  2     !
;  237          0368  2     $dclexh (desblk = desblk);
;  238          0369  2     forward_link = .desblk[0];
;  239          0370  2     $canexh (desblk = desblk);
;  240          0371  2     IF .forward_link NEQ .save_link
;  241          0372  2     THEN
;  242          0373  3         BEGIN
;  243          0374  3         dbg$gl_smg_exit_handler = .forward_link[1];
;  244          0375  3         $canexh (desblk = .forward_link);
;  245          0376  2         END;
;  246          0377  2
;  247          0378  2     RETURN .status;
;  248          0379  1     END;


                                                        .TITLE  DBGEXC
                                                        .IDENT  \V04-000\

                                                        .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

                    54  55  50  4E  49  24  47  42  44  00000 P.AAA:  .ASCII  \DBG$INPUT\                          ;
                    54  55  50  4E  49  24  53  59  53  00009 P.AAB:  .ASCII  \SYS$INPUT\                          ;

                                                        .PSECT  DBG$OWN,NOEXE,  PIC,2

                                            00000000    00000 KEYPAD_INITIALIZATION_DONE:
                                                        .LONG   0                                                 ;
                                                    00004 DESBLK: .BLKB   16
                                                    00014 DUMMY1: .BLKB   4
                                                    00018 DUMMY2: .BLKB   4

                                                        .PSECT  DBG$GLOBAL,NOEXE,  PIC,2

                                            00000000    00000 DBG$GL_SMG_EXIT_HANDLER::
```

```
                        .LONG   0                                                       ;

                        .EXTRN  DBG$INIT_STEP, DBG$KEY_INITIALIZE
                        .EXTRN  DBG$SET_STP_LVL
                        .EXTRN  DBG$REL_MEMORY, DBG$GET_MEMORY
                        .EXTRN  DBG$OUT_MESSAGE
                        .EXTRN  DBG$NCONTROL, DBG$CIS_ADD
                        .EXTRN  DBG$CIS_REMOVE, DBG$FINAL_HANDL
                        .EXTRN  DBG$SCR_GENERATE_SCREEN
                        .EXTRN  DBG$SCR_OUTPUT_SCREEN
                        .EXTRN  DBG$EXCEPTION_HANDLER
                        .EXTRN  DBG$ACTIVATE_EVENTS
                        .EXTRN  SMG$CREATE_KEY_TABLE
                        .EXTRN  SMG$CREATE_VIRTUAL_KEYBOARD
                        .EXTRN  SMG$READ_COMPOSED_LINE
                        .EXTRN  SYS$PUTMSG, DBG$GB_KEYPAD_INPUT
                        .EXTRN  DBG$GL_KEYBOARD_ID
                        .EXTRN  DBG$GB_LANGUAGE
                        .EXTRN  DBG$GL_KEY_TABLE_ID
                        .EXTRN  DBG$GB_DEF_OUT, DBG$GL_EXIT_STATUS
                        .EXTRN  DBG$GL_INPRAB, DBG$GL_OUTPRAB
                        .EXTRN  DBG$GL_LOGRAB, DBG$GL_LOG_BUF
                        .EXTRN  DBG$GL_READERR_CNT
                        .EXTRN  DBG$GL_SCREEN_MODE
                        .EXTRN  DBG$GL_CISHEAD, DBG$GV_CONTROL
                        .EXTRN  SMG$_EOF, SYS$DCLEXH
                        .EXTRN  SYS$CANEXH

                        .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                03FC 000C0      .ENTRY  DBG$CREATE_VIRTUAL_KEYBOARD, Save R2,R3,R4,-    ; 0287
                                        R5,R6,R7,R8,R9                                  ;
     59 00000000G   00  9E 00002        MOVAB   SMG$CREATE_VIRTUAL_KEYBOARD, R9         ;
     58 00000000G   00  9E 00009        MOVAB   DBG$GL_KEYBOARD_ID, R8                  ;
     57 00000000G   00  9E 00010        MOVAB   SYS$DCLEXH, R7                          ;
     56 00000000G   00  9E 00017        MOVAB   SYS$CANEXH, R6                          ;
     55 00000000'   EF  9E 0001E        MOVAB   DESBLK, R5                              ;
     5E             08  C2 00025        SUBL2   #8, SP                                  ;
                    65  D4 00028        CLRL    DESBLK                                  ; 0323
04   A5       10    A5  9E 0002A        MOVAB   DUMMY1, DESBLK+4                        ; 0324
08   A5             01  D0 0002F        MOVL    #1, DESBLK+8                            ; 0325
0C   A5       14    A5  9E 00033        MOVAB   DUMMY2, DESBLK+12                       ; 0326
         010E0009   8F  DD 00038        PUSHL   #17694729                               ; 0335
04   AE 00000000'   EF  9E 0003E        MOVAB   P.AAA, FILESPEC+4                       ; 0336
                    55  DD 00046        PUSHL   R5                                      ; 0342
     67             01  FB 00048        CALLS   #1, SYS$DCLEXH                          ;
     54             65  D0 0004B        MOVL    DESBLK, SAVE_LINK                       ; 0343
                    55  DD 0004E        PUSHL   R5                                      ; 0344
     66             01  FB 00050        CALLS   #1, SYS$CANEXH                          ;
           4100     8F  BB 00053        PUSHR   #^M<R8,SP>                              ; 0350
     69             02  FB 00057        CALLS   #2, SMG$CREATE_VIRTUAL_KEYBOARD         ;
     53             50  D0 0005A        MOVL    R0, STATUS                              ;
     12             53  E8 0005D        BLBS    STATUS, 1$                              ; 0351
04   AE 00000000'   EF  9E 00060        MOVAB   P.AAB, FILESPEC+4                       ; 0354
             4100   8F  BB 00068        PUSHR   #^M<R8,SP>                              ; 0355
     69             02  FB 0006C        CALLS   #2, SMG$CREATE_VIRTUAL_KEYBOARD         ;
     53             50  D0 0006F        MOVL    R0, STATUS                              ;
```

```
                                 55  DD 00072 1$:    PUSHL   R5                                                      ; 0368
                              67 01  FB 00074        CALLS   #1, SYS$DCLEXH                                          ;
                              52 65  D0 00077        MOVL    DESBLK, FORWARD_LINK                                    ; 0369
                                 55  DD 0007A        PUSHL   R5                                                      ; 0370
                              66 01  FB 0007C        CALLS   #1, SYS$CANEXH                                          ;
                              54 52  D1 0007F        CMPL    FORWARD_LINK, SAVE_LINK                                 ; 0371
                                 0D  13 00082        BEQL    2$                                                      ;
                00000000'  EF 04  A2  D0 00084        MOVL    4(FORWARD_LINK), DBG$GL_SMG_EXIT_HANDLER                ; 0374
                                 52  DD 0008C        PUSHL   FORWARD_LINK                                            ; 0375
                              66 01  FB 0008E        CALLS   #1, SYS$CANEXH                                          ;
                              50 53  D0 00091 2$:     MOVL    STATUS, R0                                              ; 0378
                                 04  00094          RET                                                             ; 0379

; Routine Size:  149 bytes,    Routine Base:  DBG$CODE + 0000
```

```
;  250          0380  1 GLOBAL ROUTINE DBG$COMMAND_PROC : NOVALUE =
;  251          0381  1 !
;  252          0382  1 ! FUNCTIONAL DESCRIPTION:
;  253          0383  1 !       Accepts a single (possibly multiple through the use of
;  254          0384  1 !       continuation lines) command sequence from the user at DEBUG
;  255          0385  1 !       command level. If the argument buffer_desc holds a value other
;  256          0386  1 !       than a zero, then the command comes from this buffer.
;  257          0387  1 !       Otherwise a command is read from the input device.
;  258          0388  1 !
;  259          0389  1 !       This routine declares an exception vector. Exceptions
;  260          0390  1 !       encountered from this point generally cause an unwind, and then
;  261          0391  1 !       this routine is called again by user_proc. They are generally
;  262          0392  1 !       caused by user typing errors.
;  263          0393  1 !
;  264          0394  1 !       If the command read from the device INPUT is interpreted by
;  265          0395  1 !       RMS as EOF, or any other nonsuccessful return from RMS is seen,
;  266          0396  1 !       then set the exit flag, cancel the command taking flag, and return.
;  267          0397  1 !
;  268          0398  1 ! FORMAL PARAMETERS:
;  269          0399  1 !       None
;  270          0400  1 !
;  271          0401  1 ! IMPLICIT INPUTS:
;  272          0402  1 !       The name of the DEBUG command level exception handler that is
;  273          0403  1 !       declared within the context of this routine.
;  274          0404  1 !
;  275          0405  1 !       The fact that if the DBG$_L_BPT_PC field in the runframe
;  276          0406  1 !       has a non-zero value then it must be the address
;  277          0407  1 !       of a "temporary" breakpoint which DEBUG set to implement
;  278          0408  1 !       step /OVER.
;  279          0409  1 !
;  280          0410  1 ! IMPLICIT OUTPUTS:
;  281          0411  1 !       none
;  282          0412  1 !
;  283          0413  1 ! ROUTINE VALUE:
;  284          0414  1 !       novalue
;  285          0415  1 !
;  286          0416  1 ! SIDE EFFECTS:
;  287          0417  1 !       The parser is called with the contents of the input buffer.
;  288          0418  1 !
;  289          0419  1
;  290          0420  2     BEGIN
;  291          0421  2
;  292          0422  2     BUILTIN
;  293          0423  2         FP;
;  294          0424  2
;  295          0425  2     LITERAL
;  296          0426  2         NULL_BYTE_LOC = 1;
;  297          0427  2
;  298          0428  2     BIND
;  299          0429  2         PMT_STRING_1 = UPLIT BYTE
;  300          0430  2                 (%ASCII %STRING(%CHAR(CARRIAGE_RET), %CHAR(LINEFEED), 'DBG>')),
;  301    L     0431  2         PMT_SIZE_1 = %CHARCOUNT
;  302          0432  2                 (%ASCII %STRING(%CHAR(CARRIAGE_RET), %CHAR(LINEFEED), 'DBG>')),
;  303          0433  2         PMT_STRING_SUP = UPLIT BYTE
;  304          0434  2                 (%ASCII %STRING(%CHAR(CARRIAGE_RET), %CHAR(LINEFEED), 'SDBG>')),
;  305    L     0435  2         PMT_SIZE_SUP = %CHARCOUNT
;  306          0436  2                 (%ASCII %STRING(%CHAR(CARRIAGE_RET), %CHAR(LINEFEED), 'SDBG>')),
```

```
;  307          0437  2          PMT_STRING_2 = UPLIT BYTE
;  308          0438  2                  (%ASCII %STRING(%CHAR(CARRIAGE_RET), %CHAR(LINEFEED), '_')),
;  309        L 0439  2          PMT_SIZE_2 = %CHARCOUNT
;  310          0440  2                  (%ASCII %STRING(%CHAR(CARRIAGE_RET), %CHAR(LINEFEED), '_'));
;  311          0441  2
;  312          0442  2      LOCAL
;  313          0443  2          ALPHAPTR,
;  314          0444  2          ALPHAVECTOR: VECTOR[150,BYTE],
;  315          0445  2          filespec: DBG$STG_DESC,
;  316          0446  2          HAVE_A_LINE,                    ! Flag set when we have a command line
;  317          0447  2          INPRAB : REF $RAB_DECL,         ! Record Access Block (RAB) for input
;  318          0448  2          INPUT_BUFFER: VECTOR[           ! Command input buffer
;  319          0449  2                              NO_OF_INP_CHARS + %UPVAL, BYTE], !
;  320          0450  2          INP_LENGTH,                     ! Input line length
;  321          0451  2          LENGTH,
;  322          0452  2          MUST_UPDATE_SCREEN,             ! Flag set to TRUE if screen displays
;  323          0453  2                                          !     must be updated before read
;  324          0454  2          NBUF: VECTOR[TTY_OUT_WIDTH,BYTE],!
;  325          0455  2          NEW_POINTER: REF VECTOR[,BYTE], ! Pointer to current buffer
;  326          0456  2          OLD_POINTER,                    ! Pointer to previous buffer
;  327          0457  2          PREV_COUNT,                     ! Current character count
;  328          0458  2          PROMPT_STG_DESC: DBG$STG_DESC,  ! String descriptor for prompt.
;  329          0459  2          STATUS,                         ! Status returned by $GET operation
;  330          0460  2          STATUS1,
;  331          0461  2          STG_DESC: DBG$STG_DESC,         ! String descriptor for keypad input
;  332          0462  2          STOP_FLAG;                      ! Flag set if Control-Y DEBUG was done
;  333          0463  2
;  334          0464  2
;  335          0465  2
;  336          0466  2      ! Enable a condition handler as described above.
;  337          0467  2      !
;  338          0468  2      .FP = DBG$FINAL_HANDL;
;  339          0469  2
;  340          0470  2
;  341          0471  2      ! Reset the level in the STP type structure so that we forget about
;  342          0472  2      ! any kind of "override" type stepping we may have been doing.
;  343          0473  2      !
;  344          0474  2      DBG$INIT_STEP (OVERRIDE_STEP, USER_DEF_STEP);
;  345          0475  2      DBG$SET_STP_LVL (USER_DEF_STEP);
;  346          0476  2
;  347          0477  2
;  348          0478  2      ! Also set the Update-Screen flag to indicate whether we should update the
;  349          0479  2      ! contents of the terminal screen.  Screen updating is done only if we are
;  350          0480  2      ! in screen mode.
;  351          0481  2      !
;  352          0482  2      MUST_UPDATE_SCREEN = .DBG$GL_SCREEN_MODE;
;  353          0483  2
;  354          0484  2
;  355          0485  2      ! See whether we need to initialize the keypad.
;  356          0486  2      !
;  357          0487  2      IF .DBG$GB_KEYPAD_INPUT AND
;  358          0488  2        (NOT .KEYPAD_INITIALIZATION_DONE)
;  359          0489  2      THEN
;  360          0490  3          BEGIN
;  361          0491  3
;  362          0492  3          ! Check that we are on a V4 system (else we cannot use keypad input).
;  363          0493  3          !
```

```
;   364     0494  3          IF .dbg$gv_control[dbg$v_control_version_4]
;   365     0495  3          THEN
;   366     0496  4              BEGIN
;   367     0497  4              status = dbg$create_virtual_keyboard();
;   368     0498  4              IF .status
;   369     0499  4              THEN
;   370     0500  4                  ! Intialize the key table used in DEFINE/KEY.
;   371     0501  4                  !
;   372     0502  4                  status1 = smg$create_key_table(dbg$gl_key_table_id);
;   373     0503  5              IF (NOT .status) OR (NOT .status1)
;   374     0504  4              THEN
;   375     0505  5                  BEGIN
;   376     0506  5                  dbg$gb_keypad_input = FALSE;
;   377     0507  5
;   378     0508  5                  ! This is an information message (so we do not get signalled
;   379     0509  5                  ! out of this routine).
;   380     0510  5                  !
;   381     0511  5                  SIGNAL(dbg$_nokeypad, 1,
;   382     0512  5                      (IF .status THEN .status1 ELSE .status));
;   383     0513  4                  END;
;   384     0514  4
;   385     0515  4              DBG$KEY_INITIALIZE();
;   386     0516  4              KEYPAD_INITIALIZATION_DONE = TRUE;
;   387     0517  4              END
;   388     0518  4
;   389     0519  3          ELSE
;   390     0520  3
;   391     0521  3              ! Not a version 4 system - set keypad mode back to false
;   392     0522  3              ! and signal an informational informing the user what is
;   393     0523  3              ! happening.
;   394     0524  3              !
;   395     0525  4              BEGIN
;   396     0526  4              dbg$gb_keypad_input = FALSE;
;   397     0527  4              SIGNAL(dbg$_keypadv4);
;   398     0528  3              END;
;   399     0529  2          END;
;   400     0530  2
;   401     0531  2      ! If we have re-entered DEBUG by means of a ^Y, DEBUG sequence then
;   402     0532  2      ! the flag DBG$V_CONTROL_STOP will be set. If this is the case, all
;   403     0533  2      ! command buffers, etc., are to be deleted, and we are to return to
;   404     0534  2      ! taking commands from the default input device.
;   405     0535  2      !
;   406     0536  2      STOP_FLAG = FALSE;
;   407     0537  2      IF .DBG$GV_CONTROL[DBG$V_CONTROL_STOP]
;   408     0538  2      THEN
;   409     0539  3          BEGIN
;   410     0540  3          DBG$GV_CONTROL[DBG$V_CONTROL_STOP] = FALSE;
;   411     0541  3          WHILE .DBG$GL_CISHEAD[CIS$B_INPUT_TYPE] NEQ CIS_DBG$INPUT DO
;   412     0542  3              DBG$CIS_REMOVE(FALSE);
;   413     0543  3
;   414     0544  3          STOP_FLAG = TRUE;
;   415     0545  2          END;
;   416     0546  2
;   417     0547  2
;   418     0548  2      ! Set up the string descriptor that describes the prompt.  The prompt is
;   419     0549  2      ! either the SUPERDEBUG prompt "SDBG>" or the normal DEBUG prompt "DBG>".
;   420     0550  2      !
```

```
;  421     0551  2     PROMPT_STG_DESC[DSC$B_CLASS] = DSC$K_CLASS_S;
;  422     0552  2     PROMPT_STG_DESC[DSC$B_DTYPE] = DSC$K_DTYPE_T;
;  423     0553  2     IF .DBG$GV_CONTROL[DBG$V_CONTROL_SDBG]
;  424     0554  2     THEN
;  425     0555  3         BEGIN
;  426     0556  3         PROMPT_STG_DESC[DSC$W_LENGTH] = 7;
;  427     0557  3         PROMPT_STG_DESC[DSC$A_POINTER] = UPLIT BYTE
;  428     0558  3             (%ASCII %STRING(%CHAR(CARRIAGE_RET), %CHAR(LINEFEED), 'SDBG>'));
;  429     0559  3         END
;  430     0560  2
;  431     0561  2     ELSE
;  432     0562  3         BEGIN
;  433     0563  3         PROMPT_STG_DESC[DSC$W_LENGTH] = 6;
;  434     0564  3         PROMPT_STG_DESC[DSC$A_POINTER] = UPLIT BYTE
;  435     0565  3             (%ASCII %STRING(%CHAR(CARRIAGE_RET), %CHAR(LINEFEED), 'DBG>'));
;  436     0566  3         END;
;  437     0567  2
;  438     0568  2
;  439     0569  2     ! Enter the read loop.  Here we loop, reading input from the user's
;  440     0570  2     ! terminal (or DBG$INPUT) until we get a complete command line.  We
;  441     0571  2     ! stay in the loop to collect all continuation lines until no more
;  442     0572  2     ! continuation lines are present.
;  443     0573  2     !
;  444     0574  2     HAVE_A_LINE = FALSE;
;  445     0575  2     WHILE NOT .HAVE_A_LINE DO
;  446     0576  3         BEGIN
;  447     0577  3
;  448     0578  3
;  449     0579  3         ! If screen mode is set and the user program has gained control since
;  450     0580  3         ! the last time we updated all automatically updated screen displays,
;  451     0581  3         ! then we update all automatic screen displays at this point.  (This is
;  452     0582  3         ! suppressed if the STOP_FLAG is set due to a Control-Y DEBUG.)  The
;  453     0583  3         ! effect of doing so is to add CIS_SCREEN entries to the Command Input
;  454     0584  3         ! Stream.  These entries then cause the necessary commands to be exe-
;  455     0585  3         ! cuted below to fill in the contents of these screen displays.
;  456     0586  3         !
;  457     0587  3         IF .DBG$GL_SCREEN_MODE                          AND
;  458     0588  3            .DBG$GV_CONTRO[[DBG$V_CONTROL_SCREEN] AND
;  459     0589  3            (NOT .STOP_FLAG)                             AND
;  460     0590  4            (.DBG$GL_CISHEAD[CIS$B_INPUT_TYPE] EQL CIS_DBG$INPUT)
;  461     0591  3         THEN
;  462     0592  4             BEGIN
;  463     0593  4             DBG$GV_CONTROL[DBG$V_CONTROL_SCREEN] = FALSE;
;  464     0594  4             DBG$SCR_GENERATE_SCREEN(0);
;  465     0595  4             END;
;  466     0596  3
;  467     0597  3
;  468     0598  3         ! If the head of the command argument list is of type buffer, process
;  469     0599  3         ! it.
;  470     0600  3         !
;  471     0601  3         IF (.DBG$GL_CISHEAD[CIS$B_INPUT_TYPE] EQL CIS_INPBUF) OR
;  472     0602  3            (.DBG$GL_CISHEAD[CIS$B_INPUT_TYPE] EQL CIS_ACBUF)  OR
;  473     0603  3            (.DBG$GL_CISHEAD[CIS$B_INPUT_TYPE] EQL CIS_IF)     OR
;  474     0604  3            (.DBG$GL_CISHEAD[CIS$B_INPUT_TYPE] EQL CIS_REPEAT) OR
;  475     0605  3            (.DBG$GL_CISHEAD[CIS$B_INPUT_TYPE] EQL CIS_WHILE)  OR
;  476     0606  3            (.DBG$GL_CISHEAD[CIS$B_INPUT_TYPE] EQL CIS_FOR)    OR
;  477     0607  4            (.DBG$GL_CISHEAD[CIS$B_INPUT_TYPE] EQL CIS_SCREEN)
```

```
;  478     0608  3          THEN
;  479     0609  4              BEGIN
;  480     0610  4              DBG$NCONTROL (.DBG$GL_CISHEAD);
;  481     0611  4              RETURN;
;  482     0612  4              END;
;  483     0613  3
;  484     0614  3
;  485     0615  3          ! If we are reading from the user's terminal (DBG$INPUT in general)
;  486     0616  3          ! or from an indirect command file, set up and do such a read.
;  487     0617  3          !
;  488     0618  3          IF (.DBG$GL_CISHEAD[CIS$B_INPUT_TYPE] EQL CIS_RAB) OR
;  489     0619  4             (.DBG$GL_CISHEAD[CIS$B_INPUT_TYPE] EQL CIS_DBG$INPUT)
;  490     0620  3          THEN
;  491     0621  4              BEGIN
;  492     0622  4
;  493     0623  4
;  494     0624  4              ! If link is flagged for removal due to RMS problems, do it now.
;  495     0625  4              !
;  496     0626  4              IF .DBG$GL_CISHEAD [CIS$V_REM_FLAG]
;  497     0627  4              THEN
;  498     0628  4                  DBG$CIS_REMOVE (FALSE)
;  499     0629  4
;  500     0630  4
;  501     0631  4              ! Otherwise we must collect an entire command line before calling
;  502     0632  4              ! the parser.  Enter a loop that collects multiple lines of input,
;  503     0633  4              ! ceasing only when a line ends with other than a hyphen ("-"),
;  504     0634  4              ! which is the line continuation character. Buffer the possibly
;  505     0635  4              ! multiple lines into free storage.
;  506     0636  4              !
;  507     0637  4              ELSE
;  508     0638  5                  BEGIN
;  509     0639  5                  INPRAB = .DBG$GL_CISHEAD [CIS$A_INPUT_PTR];
;  510     0640  5                  PREV_COUNT = 0;
;  511     0641  5                  OLD_POINTER = 0;
;  512     0642  5                  IF .DBG$GL_CISHEAD [CIS$B_INPUT_TYPE] EQL CIS_DBG$INPUT
;  513     0643  5                  THEN
;  514     0644  6                      BEGIN
;  515     0645  6                      IF .DBG$GV_CONTROL[DBG$V_CONTROL_SDBG]
;  516     0646  6                      THEN
;  517     0647  7                          BEGIN
;  518     0648  7                          INPRAB [RAB$L_PBF] = PMT_STRING_SUP;
;  519     0649  7                          INPRAB [RAB$B_PSZ] = PMT_SIZE_SUP;
;  520     0650  7                          END
;  521     0651  7
;  522     0652  6                      ELSE
;  523     0653  7                          BEGIN
;  524     0654  7                          INPRAB [RAB$L_PBF] = PMT_STRING_1;
;  525     0655  7                          INPRAB [RAB$B_PSZ] = PMT_SIZE_1;
;  526     0656  6                          END;
;  527     0657  6
;  528     0658  5                      END;
;  529     0659  5
;  530     0660  5
;  531     0661  5                  ! If screen mode is active and we have not yet updated the
;  532     0662  5                  ! displays in this call on DBG$COMMAND_PROC, we do so now.
;  533     0663  5                  ! This means that the user sees all his screen displays
;  534     0664  5                  ! updated just before he is prompted for more input.
```

```
;  535     0665  5                    !
;  536     0666  5                    IF .MUST_UPDATE_SCREEN AND (NOT .STOP_FLAG) AND
;  537     0667  6                       (.DBG$GL_CISREAD[CIS$B_INPUT_TYPE] EQL CIS_DBG$INPUT)
;  538     0668  5                    THEN
;  539     0669  6                        BEGIN
;  540     0670  6                        MUST_UPDATE_SCREEN = FALSE;
;  541     0671  6                        DBG$SCR_OUTPUT_SCREEN();
;  542     0672  5                        END;
;  543     0673  5
;  544     0674  5
;  545     0675  5                    ! If keypad input is enabled, then we read a line of input
;  546     0676  5                    ! using the RTL routine SMG$READ_COMPOSED_LINE, which
;  547     0677  5                    ! handles keypad input.
;  548     0678  5                    !
;  549     0679  5                    INPRAB[RAB$W_USZ] = NO_OF_INP_CHARS;
;  550     0680  5                    INPRAB[RAB$L_UBF] = INPUT_BUFFER;
;  551     0681  5                    IF .DBG$GB_KEYPAD_INPUT AND
;  552     0682  6                       (.DBG$GL_CISHEAD[CIS$B_INPUT_TYPE] EQL CIS_DBG$INPUT)
;  553     0683  5                    THEN
;  554     0684  6                        BEGIN
;  555     0685  6
;  556     0686  6
;  557     0687  6                        ! Set up a string descriptor for the input line.
;  558     0688  6                        !
;  559     0689  6                        STG_DESC[DSC$B_CLASS] = DSC$K_CLASS_S;
;  560     0690  6                        STG_DESC[DSC$B_DTYPE] = DSC$K_DTYPE_T;
;  561     0691  6                        STG_DESC[DSC$W_LENGTH] = NO_OF_INP_CHARS;
;  562     0692  6                        STG_DESC[DSC$A_POINTER] = INPUT_BUFFER;
;  563     0693  6
;  564     0694  6
;  565     0695  6                        ! Call the keypad input routine.
;  566     0696  6                        ! Zero the INP_LENGTH variable first because
;  567     0697  6                        ! SMG$READ_COMPOSED_LINE writes only into the low word,
;  568     0698  6                        ! so we first clear out junk in the high word.
;  569     0699  6                        !
;  570     0700  6                        INP_LENGTH = 0;
;  571     0701  6                        STATUS = SMG$READ_COMPOSED_LINE (
;  572     0702  6                                        DBG$GL_KEYBOARD_ID,
;  573     0703  6                                        DBG$GL_KEY_TABLE_ID,
;  574     0704  6                                        STG_DESC,
;  575     0705  6                                        PROMPT_STG_DESC,
;  576     0706  6
;  577     0707  6 !                      *** Note - the fifth parameter (DEFAULT_STATE) is going away from
;  578     0708  6 !                      *** SMG$READ_COMPOSED_LINE in this build (according to Steve Lionel,
;  579     0709  6 !                      *** so this "0" is commented out. If Steve's change does not get in,
;  580     0710  6 !                      *** the "0" fifth parameter must be put back.
;  581     0711  6 !
;  582     0712  6 !                                      0,
;  583     0713  6
;  584     0714  6
;  585     0715  6                                        INP_LENGTH);
;  586     0716  6
;  587     0717  6                        ! If we got back a bad status and it was not EOF,
;  588     0718  6                        ! then we try reverting to ordinary RMS input.
;  589     0719  6                        !
;  590     0720  7                        IF (NOT .STATUS)
;  591     0721  7                        AND (.STATUS NEQ RMS$_EOF)
```

```
;  592     0722  7                AND (.STATUS NEQ SMG$_EOF)
;  593     0723  6                THEN
;  594     0724  7                    BEGIN
;  595     0725  7                    DBG$GB_KEYPAD_INPUT = FALSE;
;  596     0726  7                    SIGNAL(dbg$_nokeypad, 1, .status);
;  597     0727  7                    STATUS = $GET(RAB = .INPRAB);
;  598     0728  7                    INP_LENGTH = .INPRAB[RAB$W_RSZ];
;  599     0729  6                    END;
;  600     0730  6                END
;  601     0731  6
;  602     0732  6
;  603     0733  6            ! Keypad input is not enabled or we are reading from an indirect
;  604     0734  6            ! command file.  Hence we do read by calling $GET to read a line
;  605     0735  6            ! of input.
;  606     0736  6            !
;  607     0737  5            ELSE
;  608     0738  6                BEGIN
;  609     0739  6                STATUS = $GET(RAB = .INPRAB);
;  610     0740  6                INP_LENGTH = .INPRAB[RAB$W_RSZ];
;  611     0741  5                END;
;  612     0742  5
;  613     0743  5
;  614     0744  5            ! If $GET returned a bad status, try to determine why.  If we
;  615     0745  5            ! got an End-of-File, resume taking input from the next link
;  616     0746  5            ! in the CIS.  Any other error is simply signalled.
;  617     0747  5            !
;  618     0748  5            IF NOT .STATUS
;  619     0749  5            THEN
;  620     0750  6                BEGIN
;  621     0751  6
;  622     0752  6
;  623     0753  6                ! Check for an End-of-File--in this case, resume taking
;  624     0754  6                ! input from the next link in the CIS and if none exists,
;  625     0755  6                ! simply exit from DEBUG.
;  626     0756  6                !
;  627     0757  6                IF (.STATUS EQL RMS$_EOF) OR
;  628     0758  7                   (.STATUS EQL SMG$_EOF)
;  629     0759  6                THEN
;  630     0760  7                    BEGIN
;  631     0761  7                    IF .DBG$GL_CISHEAD [CIS$B_INPUT_TYPE] EQL CIS_DBG$INPUT
;  632     0762  7                    THEN
;  633     0763  8                        BEGIN
;  634     0764  8                        DBG$GV_CONTROL[DBG$V_CONTROL_EXIT] = TRUE;
;  635     0765  8                        DBG$GV_CONTROL[DBG$V_CONTROL_USER] = TRUE;
;  636     0766  8
;  637     0767  8                        ! Call the SMG exit handler - this resets the
;  638     0768  8                        ! terminal to what it was when we entered.
;  639     0769  8                        !
;  640     0770  8                        IF .DBG$GL_SMG_EXIT_HANDLER NEQ 0
;  641     0771  8                        THEN
;  642     0772  8                            (.DBG$GL_SMG_EXIT_HANDLER)();
;  643     0773  8
;  644     0774  8                        $EXIT(CODE = .DBG$GL_EXIT_STATUS OR STS$M_INHIB_MSG);
;  645     0775  8                        END
;  646     0776  8
;  647     0777  7                    ELSE
;  648     0778  7                        DBG$CIS_REMOVE (FALSE);
```

```
;  649     0779  7                        END
;  650     0780  7
;  651     0781  7
;  652     0782  7                    ! On any other read problem, simply signal the error.
;  653     0783  7                    !
;  654     0784  7                    ELSE
;  655     0785  6                        BEGIN
;  656     0786  7                        DBG$GL_READERR_CNT = .DBG$GL_READERR_CNT + 1;
;  657     0787  7                        IF .DBG$GB_KEYPAD_INPUT
;  658     0788  7                        THEN
;  659     0789  7                            SIGNAL(DBG$_INPREADERR, 0, .STATUS)
;  660     0790  7
;  661     0791  7                        ELSE
;  662     0792  7                            INP_READ_ERROR;
;  663     0793  7                        END;
;  664     0794  6
;  665     0795  6                    END
;  666     0796  6
;  667     0797  6
;  668     0798  6                ! There was no read problem--we successfully got the line.
;  669     0799  6                !
;  670     0800  6                ELSE
;  671     0801  5                    BEGIN
;  672     0802  6                    HAVE_A_LINE = TRUE;
;  673     0803  6                    DBG$GL_READERR_CNT = 0;
;  674     0804  6                    END;
;  675     0805  5
;  676     C806  5                END;
;  677     0807  4
;  678     0808  4            END;
;  679     0809  3
;  680     0810  3        END;                            ! End of read loop for complete command
;  681     0811  2            END;
;  682     0812  2
;  683     0813  2        ! We have now read a complete command line, including all continuation
;  684     0814  2        ! lines.
;  685     0815  2        !
;  686     0816  2        INPRAB [RAB$V_PTA] = FALSE;
;  687     0817  2        IF .DBG$GL_CISHEAD[CIS$B_INPUT_TYPE] EQL CIS_DBG$INPUT
;  688     0818  2        THEN
;  689     0819  2            BEGIN
;  690     0820  3            INPRAB [RAB$L_PBF] = PMT_STRING_2;
;  691     0821  3            INPRAB [RAB$B_PSZ] = PMT_SIZE_2;
;  692     0822  3            PROMPT_STG_DESC[DSC$W_LENGTH] = 3;
;  693     0823  3            PROMPT_STG_DESC[DSC$A_POINTER] = UPLIT BYTE
;  694     0824  3                (%ASCII %STRING(%CHAR(CARRIAGE_RET), %CHAR(LINEFEED), '_'));
;  695     0825  3
;  696     0826  3
;  697     0827  3            ! If logging is enabled, copy the newly read input line to the LOG file
;  698     0828  3            ! Note this is only done if we are reading commands from DBG$INPUT,
;  699     0829  3            ! otherwise DBG$VERIFY_OUT takes care of things.
;  700     0830  3            !
;  701     0831  3            IF .DBG$GB_DEF_OUT [OUT_LOG]
;  702     0832  3            THEN
;  703     0833  3                BEGIN
;  704     0834  4
;  705     0835  4
```

```
;  706     0836  4          LOCAL
;  707     0837  4              CMT_BUF : VECTOR [NO_OF_INP_CHARS + %UPVAL + 1, BYTE];
;  708     0838  4
;  709     0839  4
;  710     0840  4          ! If this is a comment line, insert a leading "!" if there
;  711     0841  4          ! are less than two already.
;  712     0842  4          !
;  713     0843  5          IF (.INPUT_BUFFER[0] EQL %C'!') AND (.INPUT_BUFFER[1] NEQ %C'!')
;  714     0844  4          THEN
;  715     0845  5              BEGIN
;  716     0846  5              CMT_BUF[0] = %C'!';
;  717     0847  5              LENGTH = MIN(.INP_LENGTH, NO_OF_INP_CHARS - 1);
;  718     0848  5              INCR K FROM 0 TO .LENGTH - 1 DO
;  719     0849  5                  CMT_BUF [.K + 1] = .INPUT_BUFFER [.K];
;  720     0850  5
;  721     0851  5              DBG$GL_LOGRAB [RAB$L_RBF] = CMT_BUF;
;  722     0852  5              DBG$GL_LOGRAB [RAB$W_RSZ] = .LENGTH + 1;
;  723     0853  5              END
;  724     0854  5
;  725     0855  4          ELSE
;  726     0856  5              BEGIN
;  727     0857  5              DBG$GL_LOGRAB [RAB$L_RBF] = INPUT_BUFFER;
;  728     0858  5              DBG$GL_LOGRAB [RAB$W_RSZ] = .INP_LENGTH;
;  729     0859  4              END;
;  730     0860  4
;  731     0861  4
;  732     0862  4          ! We were reading from DBG$INPUT and logging is enabled, so we
;  733     0863  4          ! write the read line to the Log file.  If we get a Record Stream
;  734     0864  4          ! Active error, we wait and retry the write operation.  Any other
;  735     0865  4          ! error we simply signal.
;  736     0866  4          !
;  737     0867  4          STATUS = $PUT(RAB = DBG$GL_LOGRAB);
;  738     0868  4          IF .STATUS EQL RMS$_RSA
;  739     0869  4          THEN
;  740     0870  5              BEGIN
;  741     0871  5              $WAIT(RAB = DBG$GL_LOGRAB);
;  742     0872  5              STATUS = $PUT (RAB = DBG$GL_LOGRAB);
;  743     0873  4              END;
;  744     0874  4
;  745     0875  4          IF NOT .STATUS THEN LOG_WRITE_ERROR;
;  746     0876  3          END;
;  747     0877  3
;  748     0878  2      END;
;  749     0879  2
;  750     0880  2  WHILE TRUE DO
;  751     0881  3      BEGIN
;  752     0882  3
;  753     0883  3      LOCAL
;  754     0884  3          CONT_LINE;                  ! Boolean test for end of line character
;  755     0885  3
;  756     0886  3
;  757     0887  3      ! Check for continuation character '-' only if the
;  758     0888  3      ! length of the input line was greater than zero.
;  759     0889  3      !
;  760     0890  3      CONT_LINE = FALSE;
;  761     0891  3      IF .INP_LENGTH GTR 0
;  762     0892  3      THEN
```

```
;  763     0893  4          BEGIN
;  764     0894  4          IF .INPUT_BUFFER[.INP_LENGTH - 1] EQL '-'
;  765     0895  4          THEN
;  766     0896  4
;  767     0897  4              ! Assume '--' at end of line in C is post-decrement operator.
;  768     0898  4              !
;  769     0899  4              IF .DBG$GB_LANGUAGE NEQ DBG$K_C
;  770     0900  5              OR (IF .INP_LENGTH GEQ 2
;  771     0901  5                  THEN .INPUT_BUFFER[.INP_LENGTH - 2] NEQ '-'
;  772     0902  5                  ELSE TRUE)
;  773     0903  4              THEN
;  774     0904  5                  BEGIN
;  775     0905  5                  INP_LENGTH = .INP_LENGTH - 1;
;  776     0906  5                  CONT_LINE = TRUE;
;  777     0907  4                  END;
;  778     0908  4
;  779     0909  3          END;
;  780     0910  3
;  781     0911  3
;  782     0912  3          ! Allocate space for this buffer plus all previous buffers.
;  783     0913  3          ! If the space can be found, write the old and new buffers
;  784     0914  3          ! into the new space.
;  785     0915  3          !
;  786     0916  4          NEW_POINTER = DBG$GET_MEMORY((.PREV_COUNT + NULL_BYTE_LOC +
;  787     0917  3                                                .INP_LENGTH + 3)/4);
;  788     0918  3          IF .OLD_POINTER NEQ 0
;  789     0919  3          THEN
;  790     0920  4              BEGIN
;  791     0921  4              CH$MOVE(.PREV_COUNT, .OLD_POINTER, .NEW_POINTER);
;  792     0922  4              DBG$REL_MEMORY(.OLD_POINTER);
;  793     0923  3              END;
;  794     0924  3
;  795     0925  3          CH$MOVE(.INP_LENGTH , INPUT_BUFFER,
;  796     0926  3                                  CH$PLUS(.NEW_POINTER, .PREV_COUNT));
;  797     0927  3          PREV_COUNT = .PREV_COUNT + .INP_LENGTH;
;  798     0928  3          NEW_POINTER[.PREV_COUNT] = 0;
;  799     0929  3          OLD_POINTER = .NEW_POINTER;
;  800     0930  3
;  801     0931  3
;  802     0932  3          ! See whether this line ends with a continuation character.  If so, get
;  803     0933  3          ! another line, either from $GET or the active input Screen Display (if
;  804     0934  3          ! there is one).  If the $GET or screen read fails, set the status so
;  805     0935  3          ! that DEBUG returns to the CLI.
;  806     0936  3          !
;  807     0937  3          IF NOT .CONT_LINE THEN EXITLOOP;
;  808     0938  3          IF .DBG$GB_KEYPAD_INPUT AND
;  809     0939  4            (.DBG$GL_CISHEAD[CIS$B_INPUT_TYPE] EQL CIS_DBG$INPUT)
;  810     0940  3          THEN
;  811     0941  4              BEGIN
;  812     0942  4
;  813     0943  4              ! Set up a string descriptor for the input line.
;  814     0944  4              !
;  815     0945  4              STG_DESC[DSC$B_CLASS] = DSC$K_CLASS_S;
;  816     0946  4              STG_DESC[DSC$B_DTYPE] = DSC$K_DTYPE_T;
;  817     0947  4              STG_DESC[DSC$W_LENGTH] = NO_OF_INP_CHARS;
;  818     0948  4              STG_DESC[DSC$A_POINTER] = INPUT_BUFFER;
;  819     0949  4
```

```
;   820     0950  4            ! Call the keypad input routine.
;   821     0951  4            !
;   822     0952  4            INP_LENGTH = 0;
;   823     0953  4            STATUS = SMG$READ_COMPOSED_LINE (
;   824     0954  4                                             DBG$GL_KEYBOARD_ID,
;   825     0955  4                                             DBG$GL_KEY_TABLE_ID,
;   826     0956  4                                             STG_DESC,
;   827     0957  4                                             PROMPT_STG_DESC,
;   828     0958  4
;   829     0959  4 ! *** Note - the fifth parameter (DEFAULT_STATE) is going away from
;   830     0960  4 ! *** SMG$READ_COMPOSED_LINE in this build (according to Steve Lionel,
;   831     0961  4 ! *** so this "0" is commented out. If Steve's change does not get in,
;   832     0962  4 ! *** the "0" fifth parameter must be put back.
;   833     0963  4 !
;   834     0964  4 !                                           0,
;   835     0965  4
;   836     0966  4                                             INP_LENGTH);
;   837     0967  4            ! If we got back a bad status and it was not EOF,
;   838     0968  4            ! then we try reverting to ordinary RMS input.
;   839     0969  4            !
;   840     0970  5            IF (NOT .STATUS)
;   841     0971  5            AND (.STATUS NEQ RMS$_EOF)
;   842     0972  5            AND (.STATUS NEQ SMG$_EOF)
;   843     0973  4            THEN
;   844     0974  5                BEGIN
;   845     0975  5                DBG$GB_KEYPAD_INPUT = FALSE;
;   846     0976  5                SIGNAL(dbg$_nokeypad, 1, .status);
;   847     0977  5                STATUS = $GET(RAB = .INPRAB);
;   848     0978  5                INP_LENGTH = .INPRAB[RAB$W_RSZ];
;   849     0979  4                END;
;   850     0980  4            END
;   851     0981  4
;   852     0982  3        ELSE
;   853     0983  4            BEGIN
;   854     0984  4            STATUS = $GET(RAB = .INPRAB);
;   855     0985  4            INP_LENGTH = .INPRAB[RAB$W_RSZ];
;   856     0986  3            END;
;   857     0987  3        IF NOT .STATUS
;   858     0988  3        THEN
;   859     0989  4            BEGIN
;   860     0990  4            DBG$GL_READERR_CNT = .DBG$GL_READERR_CNT + 1;
;   861     0991  4            IF .DBG$GB_KEYPAD_INPUT
;   862     0992  4            THEN
;   863     0993  4                SIGNAL(DBG$_INPREADERR, 0, .STATUS)
;   864     0994  4            ELSE
;   865     0995  4                INP_READ_ERROR;
;   866     0996  4            END
;   867     0997  4
;   868     0998  4        ELSE
;   869     0999  4            DBG$GL_READERR_CNT = 0;
;   870     1000
;   871     1001
;   872     1002  4        ! Another write to LOG file, but only if we are taking commands
;   873     1003  4        ! from DBG$INPUT.
;   874     1004  4        !
;   875     1005  4        IF .DBG$GB_DEF_OUT[OUT_LOG] AND
;   876     1006  4           (.DBG$GL_CISHEAD[CIS$B_INPUT_TYPE] EQL CIS_DBG$INPUT)
```

```
;  877          1007  3          THEN
;  878          1008  4              BEGIN
;  879          1009  4              DBG$GL_LOGRAB[RAB$L_RBF] = INPUT_BUFFER;
;  880          1010  4              DBG$GL_LOGRAB[RAB$W_RSZ] = .INP_LENGTH;
;  881          1011  4              STATUS = $PUT (RAB = DBG$GL_LOGRAB);
;  882          1012  4              IF .STATUS EQL RMS$_RSA                         ! Record stream active error
;  883          1013  4              THEN
;  884          1014  5                  BEGIN
;  885          1015  5                  $WAIT(RAB = DBG$GL_LOGRAB);      ! Wait and retry
;  886          1016  5                  STATUS = $PUT (RAB = DBG$GL_LOGRAB);
;  887          1017  4                  END;
;  888          1018  4
;  889          1019  4              IF NOT .STATUS THEN LOG_WRITE_ERROR;
;  890          1020  3              END;
;  891          1021  3
;  892          1022  2          END;
;  893          1023  2
;  894          1024  2
;  895          1025  2      ! A complete line has been collected. Put the just read in
;  896          1026  2      ! buffer at the top of the command input stream. Call the parser with
;  897          1027  2      ! the address of a string descriptor that describes the
;  898          1028  2      ! concatenated input string.
;  899          1029  2      !
;  900          1030  2      DBG$CIS_ADD (.NEW_POINTER, .PREV_COUNT, CIS_INPBUF, 0, 0);
;  901          1031  2      DBG$NCONTROL (.DBG$GL_CISHEAD);
;  902          1032  2      RETURN;
;  903          1033  1      END;


                                                .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

                    3E  47  42  44  0A  0D  00012 P.AAC:  .ASCII  <13><10>\DBG>\                          ;
                3E  47  42  44  53  0A  0D  00018 P.AAD:  .ASCII  <13><10>\SDBG>\                         ;
                                5F  0A  0D  0001F P.AAE:  .ASCII  <13><10>\_\                            ;
                3E  47  42  44  53  0A  0D  00022 P.AAF:  .ASCII  <13><10>\SDBG>\                         ;
                    3E  47  42  44  0A  0D  00029 P.AAG:  .ASCII  <13><10>\DBG>\                          ;
                                5F  0A  0D  0002F P.AAH:  .ASCII  <13><10>\_\                            ;

                                                PMT_STRING_1=       P.AAC
                                                PMT_SIZE_1=         6
                                                PMT_STRING_SUP=     P.AAD
                                                PMT_SIZE_SUP=       7
                                                PMT_STRING_2=       P.AAE
                                                PMT_SIZE_2=         3
                                                .EXTRN  SYS$GET, SYS$EXIT
                                                .EXTRN  SYS$PUT, SYS$WAIT

                                                .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                    0FFC 00000  .ENTRY  DBG$COMMAND_PROC, Save R2,R3,R4,R5,R6,R7,-  ; 0380
                                                R8,R9,R10,R11
                        5E      FDA0    CE  9E 00002    MOVAB   -608(SP), SP                        ;
                        6D 00000000G    00  9E 00007    MOVAB   DBG$FINAL_HANDL, (FP)               ; 0468
                                        01  DD 0000E    PUSHL   #1                                  ; 0474
                                        02  DD 00010    PUSHL   #2                                  ;
00000000G               00              02  FB 00012    CALLS   #2, DBG$INIT_STEP                   ;
```

```
                                 01 DD 00019          PUSHL   #1                                                      ; 0475
          00000000G   00         01 FB 0001B          CALLS   #1, DBG$SET_STP_LVL
                      55 00000000G   00 D0 00022      MOVL    DBG$GL_SCREEN_MODE, MUST_UPDATE_SCREEN                  ; 0482
                      6E 00000000G   00 E9 00029      BLBC    DBG$GB_KEYPAD_INPUT, 6$                                 ; 0487
                      67 00000000'   EF E8 00030      BLBS    KEYPAD_INITIALIZATION_DONE, 6$                          ; 0488
4C 00000000G          00             04 E1 00037      BBC     #4, DBG$GV_CONTROL+1, 5$                                ; 0494
        FF27          CF             00 FB 0003F      CALLS   #0, DBG$CREATE_VIRTUAL_KEYBOARD                         ; 0497
                      59             50 D0 00044      MOVL    R0, STATUS
                      13             59 E9 00047      BLBC    STATUS, 1$                                              ; 0498
                           00000000G 00 9F 0004A      PUSHAB  DBG$GL_KEY_TABLE_ID                                     ; 0502
          00000000G   00             01 FB 00050      CALLS   #1, SMG$CREATE_KEY_TABLE
                      03             59 E9 00057      BLBC    STATUS, 1$                                              ; 0503
                      1E             50 E8 0005A      BLBS    STATUS1, 4$
                           00000000G 00 94 0005D 1$:  CLRB    DBG$GB_KEYPAD_INPUT                                     ; 0506
                      04             59 E9 00063      BLBC    STATUS, 2$                                              ; 0512
                                     50 DD 00066      PUSHL   STATUS1
                                     02 11 00068      BRB     3$
                                     59 DD 0006A 2$:  PUSHL   STATUS
                                     01 DD 0006C 3$:  PUSHL   #1                                                      ; 0511
                           00028763  8F DD 0006E      PUSHL   #165731
          00000000G   00             03 FB 00074      CALLS   #3, LIB$SIGNAL
          00000000G   00             00 FB 0007B 4$:  CALLS   #0, DBG$KEY_INITIALIZE                                  ; 0515
          00000000'   EF             01 D0 00082      MOVL    #1, KEYPAD_INITIALIZATION_DONE                          ; 0516
                                     13 11 00089      BRB     6$                                                      ; 0494
                           00000000G 00 94 0008B 5$:  CLRB    DBG$GB_KEYPAD_INPUT                                     ; 0526
                           0002877B  8F DD 00091      PUSHL   #165755                                                 ; 0527
          00000000G   00             01 FB 00097      CALLS   #1, LIB$SIGNAL
                                     53 D4 0009E 6$:  CLRL    STOP_FLAG                                               ; 0536
21 00000000G          00             01 E1 000A0      BBC     #1, DBG$GV_CONTROL+1, 9$                                ; 0537
          00000000G   00             02 8A 000A8      BICB2   #2, DBG$GV_CONTROL+1                                    ; 0540
                      50 00000000G   00 D0 000AF 7$:  MOVL    DBG$GL_CISHEAD, R0                                      ; 0541
                                02   A0 95 000B6      TSTB    2(R0)
                                     0B 13 000B9      BEQL    8$
                                     7E D4 000BB      CLRL    -(SP)                                                   ; 0542
          00000000G   00             01 FB 000BD      CALLS   #1, DBG$CIS_REMOVE
                                     E9 11 000C4      BRB     7$
                      53             01 D0 000C6 8$:  MOVL    #1, STOP_FLAG                                           ; 0544
              00A6    CE       010E  8F B0 000C9 9$:  MOVW    #270, PROMPT_STG_DESC+2                                 ; 0552
10 00000000G          00             01 E1 000D0      BBC     #1, DBG$GV_CONTROL, 10$                                 ; 0553
              00A4    CE             07 B0 000D8      MOVW    #7, PROMPT_STG_DESC                                     ; 0556
              00A8    CE 00000000'   EF 9E 000DD      MOVAB   P.AAF, PROMPT_STG_DESC+4                                ; 0557
                                     0E 11 000E6      BRB     11$                                                     ; 0553
              00A4    CE             06 B0 000E8 10$: MOVW    #6, PROMPT_STG_DESC                                     ; 0563
              00A8    CE 00000000'   EF 9E 000ED      MOVAB   P.AAG, PROMPT_STG_DESC+4                                ; 0564
                                     54 D4 000F6 11$: CLRL    HAVE_A_LINE                                             ; 0574
                      03             54 E9 000F8 12$: BLBC    HAVE_A_LINE, 13$                                        ; 0575
                                 01F9 31 000FB      BRW     32$
                      27 00000000G   00 E9 000FE 13$: BLBC    DBG$GL_SCREEN_MODE, 14$                                 ; 0587
1F 00000000G          00             03 E1 00105      BBC     #3, DBG$GV_CONTROL+1, 14$                               ; 0588
                      1C             53 E8 0010D      BLBS    STOP_FLAG, 14$                                          ; 0589
                      50 00000000G   00 D0 00110      MOVL    DBG$GL_CISHEAD, R0                                      ; 0590
                                02   A0 95 00117      TSTB    2(R0)
                                     10 12 0011A      BNEQ    14$
          00000000G   00             08 8A 0011C      BICB2   #8, DBG$GV_CONTROL+1                                    ; 0593
                                     7E D4 00123      CLRL    -(SP)                                                   ; 0594
          00000000G   00             01 FB 00125      CALLS   #1, DBG$SCR_GENERATE_SCREEN
                      51 00000000G   00 D0 0012C 14$: MOVL    DBG$GL_CISHEAD, R1                                      ; 0601
```

```
                50   02    A1   9A 00133           MOVZBL  2(R1), R0                                                    ;
                02           50   91 00137           CMPB    R0, #2                                                     ;
                             1E   13 0013A           BEQL    15$                                                        ;
                03           50   91 0013C           CMPB    R0, #3                                                     ; 0602
                             19   13 0013F           BEQL    15$                                                        ;
                06           50   91 00141           CMPB    R0, #6                                                     ; 0603
                             14   13 00144           BEQL    15$                                                        ;
                04           50   91 00146           CMPB    R0, #4                                                     ; 0604
                             0F   13 00149           BEQL    15$                                                        ;
                05           50   91 0014B           CMPB    R0, #5                                                     ; 0605
                             0A   13 0014E           BEQL    15$                                                        ;
                07           50   91 00150           CMPB    R0, #7                                                     ; 0606
                             05   13 00153           BEQL    15$                                                        ;
                08           50   91 00155           CMPB    R0, #8                                                     ; 0607
                             05   12 00158           BNEQ    16$                                                        ;
                             51   DD 0015A 15$:       PUSHL   R1                                                         ; 0610
                         047F     31 0015C           BRW     60$                                                        ;
                52 00000000G 00   D0 0015F 16$:       MOVL    DBG$GL_CISHEAD, R2                                         ; 0618
                01           02   A2 91 00166        CMPB    2(R2), #1                                                  ;
                             05   13 0016A           BEQL    17$                                                        ;
                             02   A2 95 0016C        TSTB    2(R2)                                                      ; 0619
                             87   12 0016F           BNEQ    12$                                                        ;
                03           12   A2 E9 00171 17$:    BLBC    18(R2), 18$                                                ; 0626
                         011F     31 00175           BRW     27$                                                        ;
                57           04   A2 D0 00178 18$:    MOVL    4(R2), INPRAB                                              ; 0639
                                  56 D4 0017C        CLRL    PREV_COUNT                                                 ; 0640
                                  5B D4 0017E        CLRL    OLD_POINTER                                                ; 0641
                                  50 D4 00180        CLRL    R0                                                         ; 0642
                             02   A2 95 00182        TSTB    2(R2)                                                      ;
                                  24 12 00185        BNEQ    20$                                                        ;
                                  50 D6 00187        INCL    R0                                                         ;
                0E 00000000G      00 01 E1 00189     BBC     #1, DBG$GV_CONTROL, 19$                                    ; 0645
                          30      A7 00000000' EF 9E 00191     MOVAB   PMT_STRING_SUP, 48(INPRAB)                         ; 0648
                          34      A7 07 90 00199     MOVB    #7, 52(INPRAB)                                             ; 0649
                                     0C 11 0019D     BRB     20$                                                        ; 0645
                          30      A7 00000000' EF 9E 0019F 19$:    MOVAB   PMT_STRING_1, 48(INPRAB)                           ; 0654
                          34      A7 06 90 001A7     MOVB    #6, 52(INPRAB)                                             ; 0655
                                  0F 55 E9 001AB 20$:    BLBC    MUST_UPDATE_SCREEN, 21$                                    ; 0666
                                  0C 53 E8 001AE     BLBS    STOP_FLAG, 21$                                             ;
                                  09 50 E9 001B1     BLBC    R0, 21$                                                    ; 0667
                                     55 D4 001B4     CLRL    MUST_UPDATE_SCREEN                                         ; 0670
                00000000G         00 00 FB 001B6     CALLS   #0, DBG$SCR_OUTPUT_SCREEN                                  ; 0671
                          20      A7      84 8F 9B 001BD 21$:    MOVZBW  #132, 32(INPRAB)                                   ; 0679
                          24      A7    FED4 CD 9E 001C2     MOVAB   INPUT_BUFFER, 36(INPRAB)                           ; 0680
                                  6A 00000000G 00 E9 001C8     BLBC    DBG$GB_KEYPAD_INPUT, 22$                           ; 0681
                                  50 00000000G 00 D0 001CF     MOVL    DBG$GL_CISHEAD, R0                                 ; 0682
                                        02 A0 95 001D6     TSTB    2(R0)                                                      ;
                                     5E 12 001D9        BNEQ    22$                                                        ;
                             0098    CE 010E0084 8F D0 001DB     MOVL    #17694852, STG_DESC                                ; 0691
                             009C    CE    FED4 CD 9E 001E4     MOVAB   INPUT_BUFFER, STG_DESC+4                           ; 0692
                                     6E D4 001EB        CLRL    INP_LENGTH                                                 ; 0700
                                     5E DD 001ED        PUSHL   SP                                                         ; 0701
                                        00A8 CE 9F 001EF     PUSHAB  PROMPT_STG_DESC                                    ;
                                        00A0 CE 9F 001F3     PUSHAB  STG_DESC                                           ;
                                  00000000G 00 9F 001F7     PUSHAB  DBG$GL_KEY_TABLE_ID                                ;
                                  00000000G 00 9F 001FD     PUSHAB  DBG$GL_KEYBOARD_ID                                 ;
                00000000G         00 05 FB 00203     CALLS   #5, SMG$READ_COMPOSED_LINE                                 ;
```

```
                        59              50  D0 0020A            MOVL    R0, STATUS                                      ;
                        39              59  E8 0020D            BLBS    STATUS, 23$                                     ; 0720
                0001827A  8F            59  D1 00210            CMPL    STATUS, #98938                                  ; 0721
                                        30  13 00217            BEQL    23$                                             ;
                00000000G  8F           59  D1 00219            CMPL    STATUS, #SMG$_EOF                               ; 0722
                                        27  13 00220            BEQL    23$                                             ;
                            00000000G   00  94 00222            CLRB    DBG$GB_KEYPAD_INPUT                             ; 0725
                                        59  DD 00228            PUSHL   STATUS                                          ; 0726
                                        01  DD 0022A            PUSHL   #1                                              ;
                            00028763    8F  DD 0022C            PUSHL   #165731                                         ;
                00000000G   00          03  FB 00232            CALLS   #3, LIB$SIGNAL                                  ;
                                        57  DD 00239 22$:       PUSHL   INPRAB                                          ; 0739
                00000000G   00          01  FB 0023B            CALLS   #1, SYS$GET                                     ;
                                        59  50  D0 00242        MOVL    R0, STATUS                                      ;
                                    22  A7  3C 00245            MOVZWL  34(INPRAB), INP_LENGTH                          ; 0740
                                        03  59  E9 00249 23$:    BLBC    STATUS, 24$                                     ; 0748
                                            009C 31 0024C       BRW     30$                                             ;
                0001827A  8F            59  D1 0024F 24$:      CMPL    STATUS, #98938                                  ; 0757
                                        09  13 00256            BEQL    25$                                             ;
                00000000G  8F           59  D1 00258            CMPL    STATUS, #SMG$_EOF                               ; 0758
                                        41  12 0025F            BNEQ    28$                                             ;
                            50 00000000G    00  D0 00261 25$:   MOVL    DBG$GL_CISHEAD, R0                              ; 0761
                                        02  A0 95 00268        TSTB    2(R0)                                           ;
                                        2A  12 0026B            BNEQ    27$                                             ;
                00000000G   00      0110    8F  A8 0026D        BISW2   #272, DBG$GV_CONTROL                            ; 0765
                            50 00000000'    EF  D0 00276        MOVL    DBG$GL_SMG_EXIT_HANDLER, R0                     ; 0770
                                        03  13 0027D            BEQL    26$                                             ;
                            60              00  FB 0027F        CALLS   #0, (R0)                                        ; 0772
            7E  00000000G   00  10000000    8F  C9 00282 26$:   BISL3   #268435456, DBG$GL_EXIT_STATUS, -(SP)           ; 0774
                00000000G   00              01  FB 0028E        CALLS   #1, SYS$EXIT                                    ;
                                        5D  11 00295            BRB     31$                                             ; 0761
                                        7E  D4 00297 27$:       CLRL    -(SP)                                           ; 0778
                00000000G   00              01  FB 00299        CALLS   #1, DBG$CIS_REMOVE                              ;
                                        52  11 002A0            BRB     31$                                             ; 0757
                            00000000G   00  D6 002A2 28$:       INCL    DBG$GL_READERR_CNT                              ; 0787
                            13 00000000G    00  E9 002A8        BLBC    DBG$GB_KEYPAD_INPUT, 29$                        ; 0788
                                        59  DD 002AF            PUSHL   STATUS                                          ; 0790
                                        7E  D4 002B1            CLRL    -(SP)                                           ;
                            00028138    8F  DD 002B3            PUSHL   #164152                                         ;
                00000000G   00              03  FB 002B9        CALLS   #3, LIB$SIGNAL                                  ;
                                        32  11 002C0            BRB     31$                                             ;
                            50      3C  A7  D0 002C2 29$:       MOVL    60(INPRAB), FAB_PTR                             ; 0792
                    0090    CE      34  A0 9B 002C6            MOVZBW  52(FAB_PTR), MSG_DESC                           ;
                    0094    CE      2C  A0 D0 002CC            MOVL    44(FAB_PTR), MSG_DESC+4                         ;
                            7E      08  A7 7D 002D2            MOVQ    8(INPRAB), -(SP)                                ;
                                0098    CE  9F 002D6            PUSHAB  MSG_DESC                                        ;
                                        01  DD 002DA            PUSHL   #1                                              ;
                            000210B4    8F  DD 002DC            PUSHL   #135348                                         ;
                00000000G   00              05  FB 002E2        CALLS   #5, LIB$SIGNAL                                  ;
                                        09  11 002E9            BRB     31$                                             ;
                            54              01  D0 002EB 30$:   MOVL    #1, HAVE_A_LINE                                 ; 0748
                                            00  D4 002EE        CLRL    DBG$GL_READERR_CNT                              ; 0803
                                        FE01    31  002F4 31$:  BRW     12$                                             ; 0804
                    07  A7                  20  8A 002F7 32$:  BICB2   #32, 7(INPRAB)                                 ; 0618
                            50 00000000G    00  D0 002FB        MOVL    DBG$GL_CISHEAD, R0                              ; 0817
                                        02  A0 95 00302        TSTB    2(R0)                                           ; 0818
                                        03  13 00305            BEQL    34$                                             ;
```

```
                                 00F0  31 00307 33$:    BRW     44$                                          ;
           30  A7 00000000'        EF  9E 0030A 34$:    MOVAB   PMT_STRING_2, 48(INPRAB)                     ; 0821
           34  A7                  03  90 00312         MOVB    #3, 52(INPRAB)                               ; 0822
         00A4  CE                  03  B0 00316         MOVW    #3, PROMPT_STG_DESC                          ; 0823
         00A8  CE 00000000'        EF  9E 0031B         MOVAB   P.AAH, PROMPT_STG_DESC+4                     ; 0824
               DC 00000000G        00  E9 00324         BLBC    DBG$GB_DEF_OUT, 33$                          ; 0832
               21       FED4       CD  91 0032B         CMPB    INPUT_BUFFER, #33                            ; 0843
                                   41  12 00330         BNEQ    38$                                          ;
               21       FED5       CD  91 00332         CMPB    INPUT_BUFFER+1, #33                          ;
                                   3A  13 00337         BEQL    38$                                          ;
           0C  AE                  21  90 00339         MOVB    #33, CMT_BUF                                 ; 0846
               50                  6E  D0 0033D         MOVL    INP_LENGTH, R0                               ; 0847
     00000083  8F                  50  D1 00340         CMPL    R0, #131                                     ;
                                   04  15 00347         BLEQ    35$                                          ;
               50         83       8F  9A 00349         MOVZBL  #131, R0                                     ;
               51                  50  D0 0034D 35$:    MOVL    R0, LENGTH                                   ;
               50                  01  CE 00350         MNEGL   #1, K                                        ; 0848
                                   08  11 00353         BRB     37$                                          ;
        0D AE40         FED4 CD40      90 00355 36$:    MOVB    INPUT_BUFFER[K], CMT_BUF+1[K]                ; 0849
      F4       50                  51  F2 0035D 37$:    AOBLSS  LENGTH, K, 36$                               ;
    00000000G  C0         0C       AE  9E 00361         MOVAB   CMT_BUF, DBG$GL_LOGRAB+40                    ; 0851
00000000G  00  51                  01  A1 00369         ADDW3   #1, LENGTH, DBG$GL_LOGRAB+34                 ; 0852
                                   10  11 00371         BRB     39$                                          ; 0843
    00000000G  00       FED4       CD  9E 00373 38$:    MOVAB   INPUT_BUFFER, DBG$GL_LOGRAB+40               ; 0857
    00000000G  00                  6E  B0 0037C         MOVW    INP_LENGTH, DBG$GL_LOGRAB+34                 ; 0858
                  0J000000G        00  9F 00383 39$:    PUSHAB  DBG$GL_LOGRAB                                ; 0867
    00000000G  00                  01  FB 00389         CALLS   #1, SYS$PUT                                  ;
               59                  50  D0 00390         MOVL    R0, STATUS                                   ;
     000182DA  8F                  59  D1 00393         CMPL    STATUS, #99034                               ; 0868
                                   1D  12 0039A         BNEQ    40$                                          ;
                  00000000G        00  9F 0039C         PUSHAB  DBG$GL_LOGRAB                                ; 0871
    00000000G  00                  01  FB 003A2         CALLS   #1, SYS$WAIT                                 ;
                  00000000G        00  9F 003A9         PUSHAB  DBG$GL_LOGRAB                                ; 0872
    00000000G  00                  01  FB 003AF         CALLS   #1, SYS$PUT                                  ;
               59                  50  D0 003B6         MOVL    R0, STATUS                                   ;
               3E                  59  E8 003B9 40$:    BLBS    STATUS, 44$                                  ; 0875
               50 00000000G        00  D0 003BC         MOVL    DBG$GL_LOGRAB+60, FAB_PTR                    ;
                  00000000G        00  D5 003C3         TSTL    DBG$GL_LOG_BUF                               ;
                                   0C  13 003C9         BEQL    41$                                          ;
           04  AE         34       A0  9B 003CB         MOVZBW  52(FAB_PTR), MSG_DESC                        ;
           08  AE         2C       A0  D0 003D0         MOVL    44(FAB_PTR), MSG_DESC+4                      ;
                                   0A  11 003D5         BRB     42$                                          ;
           04  AE         35       A0  9B 003D7 41$:    MOVZBW  53(FAB_PTR), MSG_DESC                        ;
           08  AE         30       A0  D0 003DC         MOVL    48(FAB_PTR), MSG_DESC+4                      ;
               7E 00000000G        00  7D 003E1 42$:    MOVQ    DBG$GL_LOGRAB+8, -(SP)                       ;
                          0C       AE  9F 003E8         PUSHAB  MSG_DESC                                     ;
                                   01  DD 003EB 43$:    PUSHL   #1                                           ;
                  000210D0         8F  DD 003ED         PUSHL   #135376                                      ;
    00000000G  00                  05  FB 003F3         CALLS   #5, LIB$SIGNAL                               ;
                                   5A  D4 003FA 44$:    CLRL    CONT_LINE                                    ; 0890
               50                  6E  D0 003FC         MOVL    INP_LENGTH, R0                               ; 0891
                                   23  15 003FF         BLEQ    46$                                          ;
               2D       FED3 CD40      91 00401         CMPB    INPUT_BUFFER-1[R0], #45                      ; 0894
                                   1B  12 00407         BNEQ    46$                                          ;
               07 00000000G        00  91 00409         CMPB    DBG$GB_LANGUAGE, #7                          ; 0899
                                   0D  12 00410         BNEQ    45$                                          ;
               02                  50  D1 00412         CMPL    R0, #2                                       ; 0900
```

```
                        08  19 00415          BLSS    45$                                         ;
          2D     FED2 CD40  91 00417          CMPB    INPUT_BUFFER-2[R0], #45                     ; 0901
                        05  13 0041D          BEQL    46$                                         ;
                        6E  D7 0041F 45$:     DECL    INP_LENGTH                                  ; 0905
          5A            01  D0 00421          MOVL    #1, CONT_LINE                               ; 0906
   50     56            6E  C1 00424 46$:     ADDL3   INP_LENGTH, PREV_COUNT, R0                  ; 0917
          50            04  C0 00428          ADDL2   #4, R0                                      ; 0916
   7E     50            04  C7 0042B          DIVL3   #4, R0, -(SP)                               ; 0917
 00000000G  00          01  FB 0042F          CALLS   #1, DBG$GET_MEMORY                          ;
          58            50  D0 00436          MOVL    R0, NEW_POINTER                             ;
                        5B  D5 00439          TSTL    OLD_POINTER                                 ; 0918
                        0D  13 0043B          BEQL    47$                                         ;
   68     6B            56  28 0043D          MOVC3   PREV_COUNT, (OLD_POINTER), (NEW_POINTER)    ; 0921
                        5B  DD 00441          PUSHL   OLD_POINTER                                 ; 0922
 00000000G  00          01  FB 00443          CALLS   #1, DBG$REL_MEMORY                          ;
 6648    FED4  CD       6E  28 0044A 47$:     MOVC3   INP_LENGTH, INPUT_BUFFER, (PREV_COUNT)-     ; 0926
                                                      [NEW_POINTER]
          56            6E  C0 00451          ADDL2   INP_LENGTH, PREV_COUNT                      ; 0927
                      6648  94 00454          CLRB    (PREV_COUNT)[NEW_POINTER]                   ; 0928
          5B            58  D0 00457          MOVL    NEW_POINTER, OLD_POINTER                    ; 0929
          03            5A  E8 0045A          BLBS    CONT_LINE, 48$                              ; 0937
                      0169  31 0045D          BRW     59$                                         ;
          6A 00000000G  00  E9 00460 48$:     BLBC    DBG$GB_KEYPAD_INPUT, 49$                    ; 0938
          50 00000000G  00  D0 00467          MOVL    DBG$GL_CISHEAD, R0                          ; 0939
                   02   A0  95 0046E          TSTB    2(R0)                                       ;
                        5E  12 00471          BNEQ    49$                                         ;
      0098    CE 010E0084   8F  D0 00473          MOVL    #17694852, STG_DESC                     ; 0947
      009C    CE     FED4   CD  9E 0047C          MOVAB   INPUT_BUFFER, STG_DESC+4                ; 0948
                        6E  D4 00483          CLRL    INP_LENGTH                                  ; 0952
                        5E  DD 00485          PUSHL   SP                                          ; 0953
                 00A8   CE  9F 00487          PUSHAB  PROMPT_STG_DESC                             ;
                 00A0   CE  9F 0048B          PUSHAB  STG_DESC                                    ;
            00000000G   00  9F 0048F          PUSHAB  DBG$GL_KEY_TABLE_ID                         ;
            00000000G   00  9F 00495          PUSHAB  DBG$GL_KEYBOARD_ID                          ;
 00000000G  00          05  FB 0049B          CALLS   #5, SMG$READ_COMPOSED_LINE                  ;
          59            50  D0 004A2          MOVL    R0, STATUS                                  ;
          39            59  E8 004A5          BLBS    STATUS, 50$                                 ; 0970
 0001827A   8F          59  D1 004A8          CMPL    STATUS, #98938                              ; 0971
                        30  13 004AF          BEQL    50$                                         ;
 00000000G  8F          59  D1 004B1          CMPL    STATUS, #SMG$_EOF                           ; 0972
                        27  13 004B8          BEQL    50$                                         ;
            00000000G   00  94 004BA          CLRB    DBG$GB_KEYPAD_INPUT                         ; 0975
                        59  DD 004C0          PUSHL   STATUS                                      ; 0976
                        01  DD 004C2          PUSHL   #1                                          ;
            00028763    8F  DD 004C4          PUSHL   #165731                                     ;
 00000000G  00          03  FB 004CA          CALLS   #3, LIB$SIGNAL                              ;
                        57  DD 004D1 49$:     PUSHL   INPRAB                                      ; 0984
 00000000G  00          01  FB 004D3          CALLS   #1, SYS$GET                                 ;
          59            50  D0 004DA          MOVL    R0, STATUS                                  ;
          6E        22  A7  3C 004DD          MOVZWL  34(INPRAB), INP_LENGTH                      ; 0985
          49            59  E8 004E1 50$:     BLBS    STATUS, 52$                                 ; 0987
            00000000G   00  D6 004E4          INCL    DBG$GL_READERR_CNT                          ; 0990
          13 00000000G  00  E9 004EA          BLBC    DBG$GB_KEYPAD_INPUT, 51$                    ; 0991
                        59  DD 004F1          PUSHL   STATUS                                      ; 0993
                        7E  D4 004F3          CLRL    -(SP)                                       ;
            00028138    8F  DD 004F5          PUSHL   #164152                                     ;
 00000000G  00          03  FB 004FB          CALLS   #3, LIB$SIGNAL                              ;
```

```
                                2F   11 00502          BRB      53$                                      ;
               50         3C    A7   D0 00504 51$:     MOVL     60(INPRAB), FAB_PTR                      ; 0994
        0090   CE         34    A0   9B 00508          MOVZBW   52(FAB_PTR), MSG_DESC                    ;
        0094   CE         2C    A0   D0 0050E          MOVL     44(FAB_PTR), MSG_DESC+4                  ;
               7E         08    A7   7D 00514          MOVQ     8(INPRAB), -(SP)                         ;
                        0098    CE   9F 00518          PUSHAB   MSG_DESC                                 ;
                                01   DD 0051C          PUSHL    #1                                       ;
                    000210B4    8F   DD 0051E          PUSHL    #135348                                  ;
   00000000G   00               05   FB 00524          CALLS    #5, LIB$SIGNAL                           ;
                                06   11 0052B          BRB      53$                                      ; 0987
                   00000000G    00   D4 0052D 52$:     CLRL     DBG$GL_READERR_CNT                       ; 0999
               03  00000000G    00   E8 00533 53$:     BLBS     DBG$GB_DEF_OUT, 55$                      ; 1005
                              FEBD   31 0053A 54$:     BRW      44$                                      ;
               50  00000000G    00   D0 0053D 55$:     MOVL     DBG$GL_CISHEAD, R0                       ; 1006
                          02    A0   95 00544          TSTB     2(R0)                                    ;
                                F1   12 00547          BNEQ     54$                                      ;
   00000000G   00       FED4    CD   9E 00549          MOVAB    INPUT_BUFFER, DBG$GL_LOGRAB+40           ; 1009
   00000000G   00               6E   B0 00552          MOVW     INP_LENGTH, DBG$GL_LOGRAB+34             ; 1010
                   00000000G    00   9F 00559          PUSHAB   DBG$GL_LOGRAB                            ; 1011
   00000000G   00               01   FB 0055F          CALLS    #1, SYS$PUT                              ;
               59               50   D0 00566          MOVL     R0, STATUS                               ;
    000182DA   8F               59   D1 00569          CMPL     STATUS, #99034                           ; 1012
                                1D   12 00570          BNEQ     56$                                      ;
                   00000000G    00   9F 00572          PUSHAB   DBG$GL_LOGRAB                            ; 1015
   00000000G   00               01   FB 00578          CALLS    #1, SYS$WAIT                             ;
                   00000000G    00   9F 0057F          PUSHAB   DBG$GL_LOGRAB                            ; 1016
   00000000G   00               01   FB 00585          CALLS    #1, SYS$PUT                              ;
               59               50   D0 0058C          MOVL     R0, STATUS                               ;
               A8               59   E8 0058F 56$:     BLBS     STATUS, 54$                              ; 1019
               50  00000000G    00   D0 00592          MOVL     DBG$GL_LOGRAB+60, FAB_PTR                ;
                   00000000G    00   D5 00599          TSTL     DBG$GL_LOG_BUF                           ;
                                0E   13 0059F          BEQL     57$                                      ;
        0090   CE         34    A0   9B 005A1          MOVZBW   52(FAB_PTR), MSG_DESC                    ;
        0094   CE         2C    A0   D0 005A7          MOVL     44(FAB_PTR), MSG_DESC+4                  ;
                                0C   11 005AD          BRB      58$                                      ;
        0090   CE         35    A0   9B 005AF 57$:     MOVZBW   53(FAB_PTR), MSG_DESC                    ;
        0094   CE         30    A0   D0 005B5          MOVL     48(FAB_PTR), MSG_DESC+4                  ;
               7E  00000000G    00   7D 005BB 58$:     MOVQ     DBG$GL_LOGRAB+8, -(SP)                   ;
                        0098    CE   9F 005C2          PUSHAB   MSG_DESC                                 ;
                              FE22   31 005C6          BRW      43$                                      ;
                                7E   7C 005C9 59$:     CLRQ     -(SP)                                    ; 1030
                                02   DD 005CB          PUSHL    #2                                       ;
                                56   DD 005CD          PUSHL    PREV_COUNT                               ;
                                58   DD 005CF          PUSHL    NEW_POINTER                              ;
   00000000G   00               05   FB 005D1          CALLS    #5, DBG$CIS_ADD                          ;
                   00000000G    00   DD 005D8          PUSHL    DBG$GL_CISHEAD                           ; 1031
   00000000G   00               01   FB 005DE 60$:     CALLS    #1, DBG$NCONTROL                         ;
                                     04 005E5          RET                                               ; 1033

; Routine Size:  1510 bytes,    Routine Base:  DBG$CODE + 0095
```

```
;  905          1034  1 GLOBAL ROUTINE dbg$exc_handler (signal_arg_ptr, mechan_arg_ptr) =
;  906          1035  1 !++
;  907          1036  1 ! FUNCTIONAL DESCRIPTION:
;  908          1037  1 !       Exception analyzer called by the primary vector exception handler,
;  909          1038  1 !       which is a MARS routine found in DBGSTART.MAR. The MARS routine
;  910          1039  1 !       immediately resignals if the user program was not running. Otherwise
;  911          1040  1 !       it saves the registers of the user program and disables ASTs for
;  912          1041  1 !       the time that DEBUG is running.
;  913          1042  1 !
;  914          1043  1 !       Then it calls this routine, where the exception is analyzed for
;  915          1044  1 !       the type of exception. Breakpoints and trace traps are given
;  916          1045  1 !       special handling, which usually ends with control being passed
;  917          1046  1 !       to the user. If the breakpoint or trace trap was illegal, then
;  918          1047  1 !       the exception is resignaled unless the user has asked for control
;  919          1048  1 !       on every exception.
;  920          1049  1 !
;  921          1050  1 !       Some trace traps cause an interim halt that requires some action,
;  922          1051  1 !       but doesn't pass control back to the user. After checking the
;  923          1052  1 !       validity of these trace traps, the value ss$_continue is returned.
;  924          1053  1 !
;  925          1054  1 !       After the exception is analyzed and it is determined that immediate
;  926          1055  1 !       resignaling or continuing is not desired, the user_proc routine is
;  927          1056  1 !       called. This routine accepts user commands either from prespecified
;  928          1057  1 !       action commands from breakpoints, or interactively from the terminal.
;  929          1058  1 !       Eventually, a command is given that either causes the user program
;  930          1059  1 !       to continue or DEBUG to exit. If the user program is to continue,
;  931          1060  1 !       the value returned from user_proc is ss$_continue, and that value
;  932          1061  1 !       is passed back to the MARS handler.
;  933          1062  1 !
;  934          1063  1 !       If an exception occurs during DEBUG processing, the exception
;  935          1064  1 !       handler is final_handl, not this routine.
;  936          1065  1 !
;  937          1066  1 ! FORMAL PARAMETERS:
;  938          1067  1 !       signal_arg_ptr  - address of block that contains at least four longwords.
;  939          1068  1 !                         THE PERTINENT WORDS ARE THE EXCEPTION NAME, THE
;  940          1069  1 !                         PC AT THE TIME OF THE EXCEPTION, AND THE PSL AT
;  941          1070  1 !                         THE TIME OF THE EXCEPTION. THE NAME IS ALWAYS
;  942          1071  1 !                         THE SECOND LONGWORD, THE PC AND THE PSL THE NEXT
;  943          1072  1 !                         TO LAST AND LAST RESPECTIVELY.
;  944          1073  1 !       mechan_arg_ptr  - address of block that contains five longwords.
;  945          1074  1 !                         THE PERTINENT WORDS ARE THE SAVED R0 AND R1.
;  946          1075  1 !                         THEY ARE IN THE FOURTH AND FIFTH LONGWORDS RESPECTIVELY.
;  947          1076  1 !                         NEITHER IS USED AT THIS TIME.
;  948          1077  1 !
;  949          1078  1 ! IMPLICIT INPUTS:
;  950          1079  1 !       SOME VARIABLE NUMBER OF ADDITIONAL ARGUMENTS MAY EXIST BETWEEN THE EXCEPTION
;  951          1080  1 !       NAME AND THE PC. FLAGS INDICATING THE VALIDITY OF TBITS AND BREAKPOINTS
;  952          1081  1 !       ARE REFERENCED. THE FLAG DBG$GB_RESIGNAL CAUSES ILLEGAL EXCEPTIONS TO
;  953          1082  1 !       BE RESIGNALED IF THE FLAG IS SET TO TRUE.
;  954          1083  1 !
;  955          1084  1 ! IMPLICIT OUTPUTS:
;  956          1085  1 !       The TBIT in the RUNFRAME PSL may be changed.
;  957          1086  1 !
;  958          1087  1 ! ROUTINE VALUE:
;  959          1088  1 !       ss$_resignal OR ss$_continue FOR RESIGNALING AND CONTINUING
;  960          1089  1 !       RESPECTIVELY.
;  961          1090  1 !
```

```
;   962          1091  1 ! SIDE EFFECTS:
;   963          1092  1 !       ANY NUMBER OF THINGS.
;   964          1093  1 !--
;   965          1094  1
;   966          1095  2         BEGIN
;   967          1096  2
;   968          1097  2         MAP
;   969          1098  2                 signal_arg_ptr : REF VECTOR;
;   970          1099  2
;   971          1100  2         LOCAL
;   972          1101  2                 dummy,
;   973          1102  2                 string_desc : BLOCK [8,BYTE],
;   974          1103  2                 sig_arg_count;
;   975          1104  2
;   976          1105  2
;   977          1106  2         ! If the EVENT developer bit is on, call DBG$EXCEPTION_HANDLER
;   978          1107  2         ! instead of anything else here....
;   979          1108  2         ! With the conversion to the new eventpoint code, just call
;   980          1109  2         ! the new exception handler here...
;   981          1110  2         !
;   982          1111  2         RETURN DBG$EXCEPTION_HANDLER (.SIGNAL_ARG_PTR, .MECHAN_ARG_PTR);
;   983          1112  1         END;


                                    0000 00000         .ENTRY  DBG$EXC_HANDLER, Save nothing                   ; 1034
                                08  C2 00002         SUBL2   #8, SP                                          ;
                             5E                                                                                 ;
                             7E     04  AC  7D 00005   MOVQ    SIGNAL_ARG_PTR, -(SP)                           ; 1111
                   00000000G 00         02  FB 00009   CALLS   #2, DBG$EXCEPTION_HANDLER                       ;
                                        04 00010       RET                                                     ; 1112

; Routine Size:  17 bytes,    Routine Base:  DBG$CODE + 067B
```

```
;  985          1113  1 GLOBAL  ROUTINE dbg$exception_is_fault  (exception) =
;  986          1114  1 !++
;  987          1115  1 !
;  988          1116  1 ! Functional Description:
;  989          1117  1 !
;  990          1118  1 !       Given an exception name - the longword which encodes the
;  991          1119  1 !       type, etc, of an exception - deduce if this exception is
;  992          1120  1 !       the so-called FAULT_EXC type.  This is for the PC_TO_LINE
;  993          1121  1 !       translation - we have to know if the PC is on the instruction
;  994          1122  1 !       which caused the exception, or if it is on the next instruction.
;  995          1123  1 !
;  996          1124  1 !       The answer to the question is simply whether
;  997          1125  1 !       the given EXC_NAME is in our table of exceptions.  The only
;  998          1126  1 !       trickery is that this routine makes sure only to look at
;  999          1127  1 !       the part of the longword which encodes the error code - and
; 1000          1128  1 !       not at the rest of it since that may change.
; 1001          1129  1 !
; 1002          1130  1 ! Formal Parameters:
; 1003          1131  1 !
; 1004          1132  1 !       EXCEPTION - the longword system-defined exception name.
; 1005          1133  1 !
; 1006          1134  1 ! Routine Value:
; 1007          1135  1 !
; 1008          1136  1 !       TRUE or FALSE.  See above.
; 1009          1137  1 !
; 1010          1138  1 ! Side Effects:
; 1011          1139  1 !       None.
; 1012          1140  1 !--
; 1013          1141  1
; 1014          1142  2 BEGIN   MAP     exception :     BLOCK   [%UPVAL, BYTE];
; 1015          1143  2
; 1016          1144  2         BIND    ! The 0-ended list of exception codes.
; 1017          1145  2
; 1018          1146  2                 exception_list =        UPLIT WORD
; 1019          1147  2                                         (       SS$_ACCVIO,
; 1020          1148  2                                                 SS$_NOTRAN,
; 1021          1149  2                                                 SS$_RADRMOD,
; 1022          1150  2                                                 SS$_ROPRAND,
; 1023          1151  2                                                 SS$_OPCDEC,
; 1024          1152  2                                                 SS$_OPCCUS,
; 1025          1153  2                                                 SS$_BREAK,
; 1026          1154  2                                                 SS$_FLTOVF_F,
; 1027          1155  2                                                 SS$_FLTUND_F,
; 1028          1156  2                                                 SS$_FLTDIV_F,
; 1029          1157  2                                                 SS$_TBIT,
; 1030          1158  2                                                 SS$_COMPAT,
; 1031          1159  2                                                 0
; 1032          1160  2                                         ) :     VECTOR  [, WORD];
; 1033          1161  2
; 1034          1162  2         ! Simply loop thru the list checking each one,
; 1035          1163  2         ! ending when the 0 one is encountered.
; 1036          1164  2         !
; 1037          1165  2         INCR    i
; 1038          1166  2         FROM    0
; 1039          1167  3         DO      BEGIN   LOCAL   list_entry :    BLOCK   [%UPVAL, BYTE];
; 1040          1168  3
; 1041          1169  4                         IF      ((list_entry = .exception_list [.i]) EQL 0)
```

```
;  1042          1170  3                        THEN    EXITLOOP;
;  1043          1171  3
;  1044          1172  3                        IF      (.exception [STS$V_FAC_NO] EQL 0) AND
;  1045          1173  4                                (.exception [STS$V_MSG_NO] EQL .list_entry [STS$V_MSG_NO])
;  1046          1174  3                        THEN    RETURN  (TRUE);
;  1047          1175  2                    END;
;  1048          1176  2
;  1049          1177  2                ! Entry not found in the exception list.
;  1050          1178  2
;  1051          1179  2                RETURN  (FALSE);
;  1052          1180  1 END;


                                                        .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

04BC  04C4  04B4  0414  0434  043C  0454  044C  0629  000C  00032 P.AAI:  .WORD   12, 1577, 1100, 1108, 1084, 1076, 1044, -   ;
                                  0000  042C  0464  00046               1204, 1220, 1212, 1124, 1068, 0                 ;

                                                    EXCEPTION_LIST=     P.AAI


                                                        .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                                0004 00000              .ENTRY  DBG$EXCEPTION_IS_FAULT, Save R2             ; 1113
                                              50  D4 00002              CLRL    I                                           ; 1172
                                51 00000000'EF40  3C 00004 1$:          MOVZWL  EXCEPTION_LIST[I], LIST_ENTRY               ; 1169
                                              20  13 0000C              BEQL    3$                                          ;
                                    0FFF   8F     06  AC  B3 0000E      BITW    EXCEPTION+2, #4095                          ; 1172
                                              10  12 00014              BNEQ    2$                                          ;
                                52            51     04  AC  AD 00016      XORW3   EXCEPTION, LIST_ENTRY, R2               ; 1173
                                    FFF8   8F                 52  B3 0001B      BITW    R2, #65528                          ;
                                                        04  12 00020      BNEQ    2$                                          ;
                                              50                01  D0 00022      MOVL    #1, R0                              ; 1174
                                                                    04 00025      RET                                         ;
                                D6            50 7FFFFFFF   8F    F3 00026 2$:  AOBLEQ  #2147483647, I, 1$                  ; 1165
                                                            50  D4 0002E 3$:  CLRL    R0                                  ; 1179
                                                                    04 00030      RET                                         ; 1180

; Routine Size:  49 bytes,    Routine Base:  DBG$CODE + 068C
```

```
;  1054          1181  1 GLOBAL ROUTINE DBG$PUTMSG (SIG_ARG_LIST) : NOVALUE =
;  1055          1182  1 !
;  1056          1183  1 ! FUNCTION
;  1057          1184  1 !       Reports a message by calling SYS$PUTMSG with an action routine
;  1058          1185  1 !       address of a routine to write the formatted string to DBG$OUTPUT.
;  1059          1186  1 !       This routine checks the exception name to see if the exception is not
;  1060          1187  1 !       a hardware exception.  If it is not a hardware exception 2 is sub-
;  1061          1188  1 !       tracted from the signal argument list count before calling SYS$PUTMSG.
;  1062          1189  1 !       After SYS$PUTMSG returns the original count is restored.
;  1063          1190  1 !
;  1064          1191  1 ! INPUTS
;  1065          1192  1 !       SIG_ARG_LIST    - The address of the signal argument list.
;  1066          1193  1 !
;  1067          1194  1 ! OUTPUTS
;  1068          1195  1 !       NONE
;  1069          1196  1 !
;  1070          1197  1
;  1071          1198  2     BEGIN
;  1072          1199  2
;  1073          1200  2     LOCAL
;  1074          1201  2         ORIG_ARG_COUNT,                     !
;  1075          1202  2         INDEX,                              !
;  1076          1203  2         EXCEP_NAME: BLOCK[%UPVAL,BYTE],     !
;  1077          1204  2         TABLE_VALUE: BLOCK[%UPVAL,BYTE];    !
;  1078          1205  2
;  1079          1206  2     MAP
;  1080          1207  2         SIG_ARG_LIST: REF VECTOR;       ! The input signal argument list
;  1081          1208  2
;  1082          1209  2     BIND
;  1083          1210  2         HARDWARE_EXCEP = UPLIT WORD(SS$_ACCVIO, SS$_ARTRES, SS$_INTOVF,
;  1084          1211  2                                     SS$_INTDIV, SS$_FLTOVF, SS$_FLTDIV, SS$_FLTUND,
;  1085          1212  2                                     SS$_DECOVF, SS$_SUBRNG, SS$_ASTFLT, SS$_BREAK,
;  1086          1213  2                                     SS$_CMODSUPR, SS$_CMODUSER, SS$_COMPAT,
;  1087          1214  2                                     SS$_DEBUG, SS$_OPCCUS, SS$_OPCDEC, SS$_PAGRDERR,
;  1088          1215  2                                     SS$_RADRMOD, SS$_ROPRAND, SS$_SSFAIL, SS$_TBIT,
;  1089          1216  2                                     0): VECTOR[,WORD];
;  1090          1217  2
;  1091          1218  2
;  1092          1219  2
;  1093          1220  2     ! Get the original argument count and the exception name.
;  1094          1221  2     !
;  1095          1222  2     ORIG_ARG_COUNT = .SIG_ARG_LIST[0];
;  1096          1223  2     EXCEP_NAME = .SIG_ARG_LIST[1];
;  1097          1224  3     IF (.EXCEP_NAME [STS$V_FAC_NO] NEQ 0)       ! Not SYSTEM facility
;  1098          1225  2     THEN
;  1099          1226  2         SIG_ARG_LIST[0] = .SIG_ARG_LIST[0] - 2
;  1100          1227  2
;  1101          1228  2     ELSE
;  1102          1229  3         BEGIN
;  1103          1230  3         INDEX = 0;
;  1104          1231  3
;  1105          1232  3
;  1106          1233  3         ! This loop will exit with -1 if the exception name is not found.
;  1107          1234  3         ! In that case we must subtract 2 from the signal argument list
;  1108          1235  3         ! argument count before calling SYS$PUTMSG.
;  1109          1236  3         !
;  1110          1237  4         IF (WHILE (.HARDWARE_EXCEP[.INDEX] NEQ 0) DO
```

```
;  1111          1238  5                   BEGIN
;  1112          1239  5                   TABLE_VALUE = .HARDWARE_EXCEP [.INDEX]; ! pick up next value
;  1113          1240  5
;  1114          1241  6                   IF (.EXCEP_NAME [STS$V_MSG_NO] EQL .TABLE_VALUE [STS$V_MSG_NO])
;  1115          1242  5                   THEN
;  1116          1243  5                       EXITLOOP 0;
;  1117          1244  5
;  1118          1245  5                   INDEX = .INDEX + 1;
;  1119          1246  5                   END
;  1120          1247  4               )
;  1121          1248  3           THEN
;  1122          1249  3               SIG_ARG_LIST [0] = .SIG_ARG_LIST [0] - 2;
;  1123          1250  3
;  1124          1251  2           END;
;  1125          1252  2
;  1126          1253  2       SYS$PUTMSG (.SIG_ARG_LIST, DBG$OUT_MESSAGE, 0);
;  1127          1254  2       SIG_ARG_LIST [0] = .ORIG_ARG_COUNT;
;  1128          1255  1       END;


                                                                .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

040C  04AC  04A4  049C  0494  048C  0484  047C  0474  000C  0004C P.AAJ:  .WORD   12, 1140, 1148, 1156, 1164, 1172, 1180, -  ;
0454  044C  0444  043C  0434  046C  042C  0424  041C  0414  00060                1188, 1196, 1036, 1044, 1052, 1060, 1068, -  ;
                                  0000  0464  045C  00074                      1132, 1076, 1084, 1092, 1100, 1108, 1116, -  ;
                                                                               1124, 0                                      ;

                                                                HARDWARE_EXCEP=     P.AAJ


                                                                .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                        003C  00000             .ENTRY  DBG$PUTMSG, Save R2,R3,R4,R5                    ;  1181
                          52        04  AC  D0  00002         MOVL    SIG_ARG_LIST, R2                                ;  1222
                          55            62  D0  00006         MOVL    (R2), ORIG_ARG_COUNT                            ;
                          53        04  A2  D0  00009         MOVL    4(R2), EXCEP_NAME                               ;  1223
      00                  0C            10  ED  0000D         CMPZV   #16, #12, EXCEP_NAME, #0                        ;  1224
                                        1E  12  00012         BNEQ    2$                                              ;
                                        50  D4  00014         CLRL    INDEX                                           ;  1230
                          51 00000000'EF40  3C  00016 1$:     MOVZWL  HARDWARE_EXCEP[INDEX], R1                       ;  1237
                                        12  13  0001E         BEQL    2$                                              ;
                          54            51  D0  00020         MOVL    R1, TABLE_VALUE                                 ;  1239
                51        54            53  AD  00023         XORW3   EXCEP_NAME, TABLE_VALUE, R1                     ;  1241
                    FFF8  8F            51  B3  00027         BITW    R1, #65528                                      ;
                                        07  13  0002C         BEQL    3$                                              ;
                                        50  D6  0002E         INCL    INDEX                                           ;  1245
                                        E4  11  00030         BRB     1$                                              ;  1237
                          62            02  C2  00032 2$:     SUBL2   #2, (R2)                                        ;  1249
                                        7E  D4  00035 3$:     CLRL    -(SP)                                           ;  1253
                             00000000G  00  9F  00037         PUSHAB  DBG$OUT_MESSAGE                                 ;
                                        52  DD  0003D         PUSHL   R2                                              ;
                    00000000G           03  FB  0003F         CALLS   #3, SYS$PUTMSG                                  ;
                          62            55  D0  00046         MOVL    ORIG_ARG_COUNT, (R2)                            ;  1254
                                            04  00049         RET                                                     ;  1255

; Routine Size:  74 bytes,    Routine Base:  DBG$CODE + 06BD
```

```
; 1129          1256  0 END ELUDOM


                                                        .EXTRN  LIB$SIGNAL

;                               PSECT SUMMARY
;
;       Name                    Bytes                   Attributes
;
;  DBG$GLOBAL                      4  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  DBG$OWN                        28  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  DBG$PLIT                      122  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)
;  DBG$CODE                     1799  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)


;                           Library Statistics
;                                    --------  Symbols  --------     Pages       Processing
;       File                         Total     Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1      18619         66         0        1000       00:01.8
;  _$255$DUA28:[DEBUG.OBJ]STRUCDEF.L32;1
;                                        32          0         0           7       00:00.1
;  _$255$DUA28:[DEBUG.OBJ]DBGLIB.L32;1
;                                      1545         42         2          97       00:01.9
;  _$255$DUA28:[DEBUG.OBJ]DSTRECRDS.L32;1
;                                       418          1         0          31       00:00.3
;  _$255$DUA28:[DEBUG.OBJ]DBGMSG.L32;1
;                                       386          3         0          22       00:00.3
;  _$255$DUA28:[DEBUG.OBJ]DBGGEN.L32;1
;                                       150          9         6          12       00:00.3


;                               COMMAND QUALIFIERS

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

; Size:          1799 code + 154 data bytes
; Run Time:        00:34.9
; Elapsed Time:    01:52.4
; Lines/CPU Min:    2160
; Lexemes/CPU-Min: 16592
; Memory Used:  384 pages
; Compilation Complete
```

DBGEXT
LIS

DBGEXADEP
LIS

DBGEXC
LIS