```
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      UUUUUUUUUUUUUUUUU     GGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB      UUUUUUUUUUUUUUUUU     GGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB      UUUUUUUUUUUUUUUUU     GGGGGGGGG
```

```
DDDDDDDD    BBBBBBBB      GGGGGGGG  TTTTTTTTTT    AAAAAA      SSSSSSSS  KK      KK
DDDDDDDD    BBBBBBBB      GGGGGGGG  TTTTTTTTTT    AAAAAA      SSSSSSSS  KK      KK
DD      DD  BB      BB  GG            TT       AA      AA  SS          KK      KK
DD      DD  BB      BB  GG            TT       AA      AA  SS          KK      KK
DD      DD  BB      BB  GG            TT       AA      AA  SS          KK    KK
DD      DD  BB      BB  GG            TT       AA      AA  SS          KK    KK
DD      DD  BBBBBBBB    GG            TT       AA      AA    SSSSSS    KKKKKK
DD      DD  BBBBBBBB    GG            TT       AA      AA    SSSSSS    KKKKKK
DD      DD  BB      BB  GG  GGGGGG    TT       AAAAAAAAAA          SS  KK    KK
DD      DD  BB      BB  GG  GGGGGG    TT       AAAAAAAAAA          SS  KK    KK
DD      DD  BB      BB  GG      GG    TT       AA      AA          SS  KK      KK      ....
DD      DD  BB      BB  GG      GG    TT       AA      AA          SS  KK      KK      ....
DDDDDDDD    BBBBBBBB      GGGGGG      TT       AA      AA  SSSSSSSS  KK      KK      ....
DDDDDDDD    BBBBBBBB      GGGGGG      TT       AA      AA  SSSSSSSS  KK      KK      ....


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 DBGTASK (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 !       Edward Freedman         December, 1983
;    30          0030  1 !
;    31          0031  1 ! MODULE FUNCTION
;    32          0032  1 !       This module contains all routines that parse and execute all commands
;    33          0033  1 !       related to DEBUG's multi-tasking support for ADA.
;    34          0034  1 !
;    35          0035  1
```

```
;    37          0036  1 REQUIRE 'SRC$:DBGPROLOG.REQ';
;    38          0170  1 REQUIRE 'SRC$:DBGEXT.REQ';                ! %((REQUIRE OR LIB IN DBGPROLOG? -tbs))%
;    39          1242  1
;    40          1243  1 LIBRARY 'LIB$:DBGGEN.L32';                ! %((NEEDED FOR FAULT_EXC AND TRAP_EXC -tbs))%
;    41          1244  1
;    42          1245  1
;    43          1246  1 FORWARD ROUTINE
;    44          1247  1     DBG$CONV_TASK_NUM_VALUE : NOVALUE,    ! Converts an ADA task number to the corresponding task value.
;    45          1248  1     DBG$CONV_TASK_VALUE_NUM : NOVALUE,    ! Converts an ADA task value to the corresponding task number.
;    46          1249  1     DBG$NEXECUTE_SET_TASK : NOVALUE,      ! Execute the SET TASK command
;    47          1250  1     DBG$NEXECUTE_SHOW_TASK : NOVALUE,     ! Execute the SHOW TASK command
;    48          1251  1     DBG$NPARSE_SET_TASK : NOVALUE,        ! Parse the SET TASK command
;    49          1252  1     DBG$NPARSE_SHOW_TASK : NOVALUE,       ! Parse the SHOW TASK command
;    50          1253  1     DBGEXT$PRINT_ROUTINE : NOVALUE,       ! %((-tbs))%
;    51          1254  1     LOCAL_ROUT_NAME;                      !<<-----------------------------------
;    52          1255  1
;    53          1256  1
;    54          1257  1 EXTERNAL ROUTINE
;    55          1258  1     DBG$GET_TEMPMEM,                      ! Allocates and lists dynamic storage
;    56          1259  1     DBG$NMATCH,                           ! Counted string matching routine
;    57          1260  1     DBG$NPARSE_EXPRESSION,                ! Interface to Address Expression Interpreter
;    58          1261  1     DBG$NSAVE_DECIMAL_INTEGER,            ! Converts ASCII input to integer
;    59          1262  1     DBG$SYNTAX_ERROR : NOVALUE,           ! Signal a syntax error in command
;    60          1263  1     DBG$TRACEBACK : NOVALUE;              ! Shows current runframe nesting
;    61          1264  1
;    62          1265  1
;    63          1266  1 EXTERNAL ROUTINE ADA$DBGEXT : WEAK ADDRESSING_MODE (GENERAL);   %((WHERE WILL THESE BE DECLARED? -tbs))%
;    64        L 1267  1 %IF NOT %DECLARED (ADA$_FACILITY)         ! To be declared in STARLET.REQ
;    65          1268  1 %THEN
;    66          1269  1 LITERAL ADA$_FACILITY = 49 ;              ! %((-tbs))%
;    67          1270  1 %FI
;    68          1271  1
;    69          1272  1
;    70          1273  1 EXTERNAL
;    71          1274  1     DBG$GB_LANGUAGE : BYTE,               ! Code for language setting
;    72          1275  1     DBG$GB_RADIX : VECTOR[3, BYTE],       ! Radix settings
;    73          1276  1     DBG$RUNFRAME: BLOCK [,BYTE];          ! User runframe
;    74          1277  1
;    75          1278  1
;    76          1279  1 LITERAL
;    77          1280  1     !+
;    78          1281  1     !   These literals are used both to identify the ADVERB node type and to
;    79          1282  1     !   index into a bitvector to indicate the presence of particular ADVERB
;    80          1283  1     !   or NOUN nodes.
;    81          1284  1     !-
;    82          1285  1     TASK_TASK_LIST                = 0,    ! NOUN literal
;    83          1286  1     TASK_ACTIVE                   = 1,    ! ADVERB (qualifier) literals
;    84          1287  1     TASK_ALL                      = 2,    !
;    85          1288  1     TASK_CALLS                    = 3,    !
;    86          1289  1     TASK_DEADLOCK                 = 4,    !
;    87          1290  1     TASK_FULL                     = 5,    !
;    88          1291  1     TASK_HOLD                     = 6,    !
;    89          1292  1     TASK_NOHOLD                   = 8,    ! (synonym for "RELEASE")
;    90          1293  1     TASK_PRIORITY                 = 7,    !
;    91          1294  1     TASK_RELEASE                  = 8,    !
;    92          1295  1     TASK_RESTORE                  = 9,    !
;    93          1296  1     TASK_STATE                    = 10,   !
```

```
;    94          1297  1      TASK_STATISTICS             = 11,   !
;    95          1298  1      TASK_TERMINATE              = 12,   !
;    96          1299  1      TASK_VISIBLE                = 13,   !
;    97          1300  1      TASK_MAX_QUAL               = 13;   ! Max value.
;    98          1301  1
;    99          1302  1
;   100          1303  1  MACRO
;   101          1304  1      !+
;   102          1305  1      !   These two macros are used to test for conflicting qualifiers and
;   103          1306  1      !   parameters in a given command.  The test is on bits in a flag word
;   104          1307  1      !   which are set as the syntax tree is built.  The macros depend on
;   105          1308  1      !   the bit position being given by literals of the form TASK_xxx.
;   106          1309  1      !-
;   107        M 1310  1      CONFLICT (flags) [] =
;   108          1311  1          (0 + _conflict( flags, %REMOVE(%REMAINING) ) GTR 1) %,
;   109          1312  1
;   110        M 1313  1      _conflict (flags) [list] =
;   111          1314  1          ( .flags < %name('TASK_',list), 1, 0> ) %;
;   112          1315  1
;   113          1316  1
;   114          1317  1  BIND
;   115          1318  1      DBG$CS_ACTIVE               = UPLIT BYTE (%ASCIC 'ACTIVE'),         ! Qualifier names
;   116          1319  1      DBG$CS_ALL                  = UPLIT BYTE (%ASCIC 'ALL'),            !
;   117          1320  1      DBG$CS_CALLS                = UPLIT BYTE (%ASCIC 'CALLS'),          !
;   118          1321  1      DBG$CS_DEADLOCK             = UPLIT BYTE (%ASCIC 'DEADLOCK'),       !
;   119          1322  1      DBG$CS_FULL                 = UPLIT BYTE (%ASCIC 'FULL'),           !
;   120          1323  1      DBG$CS_HOLD                 = UPLIT BYTE (%ASCIC 'HOLD'),           !
;   121          1324  1      DBG$CS_NOHOLD               = UPLIT BYTE (%ASCIC 'NOHOLD'),         !
;   122          1325  1      DBG$CS_PRIORITY             = UPLIT BYTE (%ASCIC 'PRIORITY'),       !
;   123          1326  1      DBG$CS_RELEASE              = UPLIT BYTE (%ASCIC 'RELEASE'),        !
;   124          1327  1      DBG$CS_RESTORE              = UPLIT BYTE (%ASCIC 'RESTORE'),        !
;   125          1328  1      DBG$CS_STATE                = UPLIT BYTE (%ASCIC 'STATE'),          !
;   126          1329  1      DBG$CS_STATISTICS           = UPLIT BYTE (%ASCIC 'STATISTICS'),     !
;   127          1330  1      DBG$CS_TERMINATE            = UPLIT BYTE (%ASCIC 'TERMINATE'),      !
;   128          1331  1      DBG$CS_VISIBLE              = UPLIT BYTE (%ASCIC 'VISIBLE'),        !
;   129          1332  1
;   130          1333  1      DBG$CS_READY                = UPLIT BYTE (%ASCIC 'READY'),          ! STATE names
;   131          1334  1      DBG$CS_RUNNING              = UPLIT BYTE (%ASCIC 'RUNNING'),        !
;   132          1335  1      DBG$CS_SUSPENDED            = UPLIT BYTE (%ASCIC 'SUSPENDED'),      !
;   133          1336  1      DBG$CS_TERMINATED           = UPLIT BYTE (%ASCIC 'TERMINATED'),     !
;   134          1337  1
;   135          1338  1      dbg$cs_left_paren           = UPLIT BYTE (1, dbg$k_left_parenthesis),       ! Punctuation
;   136          1339  1      dbg$cs_right_paren          = UPLIT BYTE (1, dbg$k_right_parenthesis),      !
;   137          1340  1      DBG$CS_COLON                = UPLIT BYTE (%ASCIC ':'),              !
;   138          1341  1      dbg$cs_comma                = UPLIT BYTE (1, dbg$k_comma),          !
;   139          1342  1      dbg$cs_cr                   = UPLIT BYTE (1, dbg$k_car_return),     !
;   140          1343  1      dbg$cs_equal                = UPLIT BYTE (1, dbg$k_equal),          !
;   141          1344  1      dbg$cs_slash                = UPLIT BYTE (1, dbg$k_slash);          !
;   142          1345  1
;   143          1346  1
```

```
;  145          1347  1 %SBTTL 'DBG$CONV_TASK_NUM_VALUE'
;  146          1348  1 GLOBAL ROUTINE DBG$CONV_TASK_NUM_VALUE ( TASK_NUMBER, TASK_VALUE ) : NOVALUE =
;  147          1349  1 !
;  148          1350  1 ! FUNCTION
;  149          1351  1 !       This routine converts an ADA task number to the corresponding task
;  150          1352  1 !       value.  It calls the ADA run time system to perform the actual
;  151          1353  1 !       conversion.
;  152          1354  1 !
;  153          1355  1 ! INPUT
;  154          1356  1 !       TASK_NUMBER - Address of a longword containing the task number to be
;  155          1357  1 !             converted.
;  156          1358  1 !
;  157          1359  1 ! OUTPUT
;  158          1360  1 !       TASK_VALUE - Address of a longword to contain the resulting task value.
;  159          1361  1 !
;  160          1362  1 !
;  161          1363  1
;  162          1364  2     BEGIN
;  163          1365  2
;  164          1366  2
;  165          1367  2 .TASK_VALUE = %x'0DEC0ADA';     %((TO BE REPLACED WITH SOME REAL CODE -tbs))%
;  166          1368  2
;  167          1369  2
;  168          1370  2     RETURN 0;
;  169          1371  2
;  170          1372  1     END;                        ! end of DBG$CONV_TASK_NUM_VALUE

                                                        .TITLE  DBGTASK
                                                        .IDENT  \V04-000\

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

                                45  56  49  54  43  41  06  00000 P.AAA:   .ASCII  <6>\ACTIVE\                    ;
                                        4C  4C  41  03  00007 P.AAB:   .ASCII  <3>\ALL\                       ;
                                    53  4C  4C  41  43  05  0000B P.AAC:   .ASCII  <5>\CALLS\                     ;
                            48  43  4F  4C  44  41  45  44  08  00011 P.AAD:   .ASCII  <8>\DEADLOCK\              ;
                                            4C  4C  55  46  04  0001A P.AAE:   .ASCII  <4>\FULL\                  ;
                                            44  4C  4F  48  04  0001F P.AAF:   .ASCII  <4>\HOLD\                  ;
                                44  4C  4F  48  4F  4E  06  00024 P.AAG:   .ASCII  <6>\NOHOLD\                 ;
                            59  54  49  52  4F  49  52  50  08  0002B P.AAH:   .ASCII  <8>\PRIORITY\             ;
                                45  53  41  45  4C  45  52  07  00034 P.AAI:   .ASCII  <7>\RELEASE\             ;
                                45  52  4F  54  53  45  52  07  0003C P.AAJ:   .ASCII  <7>\RESTORE\             ;
                                        45  54  41  54  53  05  00044 P.AAK:   .ASCII  <5>\STATE\               ;
                    53  43  49  54  53  49  54  41  54  53  0A  0004A P.AAL:   .ASCII  <10>\STATISTICS\         ;
                        45  54  41  4E  49  4D  52  45  54  09  00055 P.AAM:   .ASCII  <9>\TERMINATE\           ;
                                45  4C  42  49  53  49  56  07  0005F P.AAN:   .ASCII  <7>\VISIBLE\             ;
                                        59  44  41  45  52  05  00067 P.AAO:   .ASCII  <5>\READY\               ;
                                47  4E  49  4E  4E  55  52  07  0006D P.AAP:   .ASCII  <7>\RUNNING\             ;
                            44  45  44  4E  45  50  53  55  53  09  00075 P.AAQ:   .ASCII  <9>\SUSPENDED\         ;
                        44  45  54  41  4E  49  4D  52  45  54  0A  0007F P.AAR:   .ASCII  <10>\TERMINATED\       ;
                                                            28  01  0008A P.AAS:   .BYTE   1, 40                  ;
                                                            29  01  0008C P.AAT:   .BYTE   1, 41                  ;
                                                            3A  01  0008E P.AAU:   .ASCII  <1>\:\                ;
                                                            2C  01  00090 P.AAV:   .BYTE   1, 44                  ;
                                                            0D  01  00092 P.AAW:   .BYTE   1, 13                  ;
                                                            3D  01  00094 P.AAX:   .BYTE   1, 61                  ;
```

```
                     2F  01  00096 P.AAY:  .BYTE   1, 47                                    ;

                                    DBG$CS_ACTIVE=      P.AAA
                                    DBG$CS_ALL=         P.AAB
                                    DBG$CS_CALLS=       P.AAC
                                    DBG$CS_DEADLOCK=    P.AAD
                                    DBG$CS_FULL=        P.AAE
                                    DBG$CS_HOLD=        P.AAF
                                    DBG$CS_NOHOLD=      P.AAG
                                    DBG$CS_PRIORITY=    P.AAH
                                    DBG$CS_RELEASE=     P.AAI
                                    DBG$CS_RESTORE=     P.AAJ
                                    DBG$CS_STATE=       P.AAK
                                    DBG$CS_STATISTICS=  P.AAL
                                    DBG$CS_TERMINATE=   P.AAM
                                    DBG$CS_VISIBLE=     P.AAN
                                    DBG$CS_READY=       P.AAO
                                    DBG$CS_RUNNING=     P.AAP
                                    DBG$CS_SUSPENDED=   P.AAQ
                                    DBG$CS_TERMINATED=  P.AAR
                                    DBG$CS_LEFT_PAREN=  P.AAS
                                    DBG$CS_RIGHT_PAREN= P.AAT
                                    DBG$CS_COLON=       P.AAU
                                    DBG$CS_COMMA=       P.AAV
                                    DBG$CS_CR=          P.AAW
                                    DBG$CS_EQUAL=       P.AAX
                                    DBG$CS_SLASH=       P.AAY
                                            .EXTRN  DBG$GET_TEMPMEM
                                            .EXTRN  DBG$NMATCH, DBG$NPARSE_EXPRESSION
                                            .EXTRN  DBG$NSAVE_DECIMAL_INTEGER
                                            .EXTRN  DBG$SYNTAX_ERROR
                                            .EXTRN  DBG$TRACEBACK, DBG$GB_LANGUAGE
                                            .EXTRN  DBG$GB_RADIX, DBG$RUNFRAME
                                            .WEAK   ADA$DBGEXT

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

                                 0000 00000         .ENTRY  DBG$CONV_TASK_NUM_VALUE, Save nothing       ; 1348
                     08   BC ODECOADA   8F   D0 00002  MOVL   #233573082, @TASK_VALUE                   ; 1367
                                           04 0000A  RET                                                ; 1372

; Routine Size:  11 bytes,    Routine Base:  DBG$CODE + 0000
```

```
;  172          1373  1 %SBTTL 'DBG$CONV_TASK_VALUE_NUM'
;  173          1374  1 GLOBAL ROUTINE DBG$CONV_TASK_VALUE_NUM ( TASK_VALUE, TASK_NUMBER ) : NOVALUE =
;  174          1375  1 !
;  175          1376  1 ! FUNCTION
;  176          1377  1 !       This routine converts an ADA task value to the corresponding task
;  177          1378  1 !       number.  It calls the ADA run time system to perform the actual
;  178          1379  1 !       conversion.
;  179          1380  1 !
;  180          1381  1 ! INPUT
;  181          1382  1 !       TASK_VALUE - Address of a longword containing the task value to be
;  182          1383  1 !               converted.
;  183          1384  1 !
;  184          1385  1 ! OUTPUT
;  185          1386  1 !       TASK_NUMBER - Address of a longword to contain the resulting task
;  186          1387  1 !               number.
;  187          1388  1 !
;  188          1389  1 !
;  189          1390  1
;  190          1391  2     BEGIN
;  191          1392  2
;  192          1393  2
;  193          1394  2 .TASK_NUMBER = 42;      %((TO BE REPLACED WITH SOME REAL CODE -tbs))%
;  194          1395  2
;  195          1396  2     RETURN 0;
;  196          1397  2
;  197          1398  1     END;                          ! end of DBG$CONV_TASK_VALUE_NUM


                                        0000 00000          .ENTRY  DBG$CONV_TASK_VALUE_NUM, Save nothing       ; 1374
                              08    BC          2A  D0 00002  MOVL    #42, @TASK_NUMBER                           ; 1394
                                                    04 00006  RET                                               ; 1398

; Routine Size:  7 bytes,    Routine Base:  DBG$CODE + 000B
```

```
;  199          1399  1 %SBTTL 'DBG$NEXECUTE_SET_TASK'
;  200          1400  1 GLOBAL ROUTINE DBG$NEXECUTE_SET_TASK ( VERB_NODE : REF DBG$VERB_NODE ) :
;  201          1401  1     NOVALUE =
;  202          1402  1 !
;  203          1403  1 ! FUNCTION
;  204          1404  1 !       This routine executes the SET TASK command.  It accepts the address
;  205          1405  1 !       of a Verb Node as input and executes the corresponding command.
;  206          1406  1 !
;  207          1407  1 ! INPUTS
;  208          1408  1 !       VERB_NODE - A pointer to the Verb Node for the SET TASK command
;  209          1409  1 !                   to be executed.  The Verb Node and its attached Adverb
;  210          1410  1 !                   and Noun Nodes contain all information picked up during
;  211          1411  1 !                   the parsing of the command.
;  212          1412  1 !
;  213          1413  1 ! OUTPUTS
;  214          1414  1 !       NONE
;  215          1415  1 !
;  216          1416  1
;  217          1417  2     BEGIN
;  218          1418  2
;  219          1419  2     LOCAL
;  220          1420  2         XXXXXXX;                        !<<--------- Local declarations --------
;  221          1421  2
;  222          1422  2
;  223          1423  2     !+
;  224          1424  2     !   Check for conflicting qualifiers and parameters.  %((REQUIRED? -tbs))%
;  225          1425  2     !-
;  226          1426  2 !    IF CONFLICT (QUALIFIERS, (ALL, TASK_LIST) )
;  227          1427  2 !       OR CONFLICT (QUALIFIERS, (ALL, ACTIVE) )
;  228          1428  2 !       OR CONFLICT (QUALIFIERS, (ALL, VISIBLE) )       %((NEED OTHER CONFLICTS? -tbs))%
;  229          1429  2 !    THEN
;  230          1430  2 !       SIGNAL (DBG$_CONFLICT);
;  231          1431  2
;  232          1432  2
;  233          1433  2     RETURN 0;
;  234          1434  2
;  235          1435  1     END;                            ! end of DBG$NEXECUTE_SET_TASK



                                        0000 00000          .ENTRY  DBG$NEXECUTE_SET_TASK, Save nothing         ; 1400
                                          04 00002          RET                                                 ; 1435

; Routine Size:  3 bytes,    Routine Base:  DBG$CODE + 0012
```

```
;  237     1436  1 %SBTTL 'DBG$NEXECUTE_SHOW_TASK'
;  238     1437  1 GLOBAL ROUTINE DBG$NEXECUTE_SHOW_TASK ( VERB_NODE : REF DBG$VERB_NODE ) :
;  239     1438  1     NOVALUE =
;  240     1439  1 !
;  241     1440  1 ! FUNCTION
;  242     1441  1 !       This routine executes the SHOW TASK command.  It accepts the address
;  243     1442  1 !       of a Verb Node as input and executes the corresponding command.
;  244     1443  1 !
;  245     1444  1 ! INPUTS
;  246     1445  1 !       VERB_NODE - A pointer to the Verb Node for the SHOW TASK command
;  247     1446  1 !                   to be executed.  The Verb Node and its attached Adverb
;  248     1447  1 !                   and Noun Nodes contain all information picked up during
;  249     1448  1 !                   the parsing of the command.
;  250     1449  1 !
;  251     1450  1 ! OUTPUTS
;  252     1451  1 !       NONE
;  253     1452  1 !
;  254     1453  1
;  255     1454  1 !+
;  256     1455  1 !   Semantics of the various qualifiers and parameters for a simple SHOW TASK
;  257     1456  1 !   command or a SHOW TASK /CALLS (i.e. not /DEADLOCK or /STATISTICS).  In this
;  258     1457  1 !   chart, 1 and 0 indicate presence or absence of the qualifiers and parameters
;  259     1458  1 !   in the command:
;  260     1459  1 !       SHOW TASK [ /CALL ] [ /PRI ] [ /STATE ] [ /HOLD ] [ /ALL ]  [ TASK_LIST,,, ]
;  261     1460  1 !   TASK_SET is the set of tasks the command is applied to where
;  262     1461  1 !       %VISIBLE =      visible task
;  263     1462  1 !       T_LIST =        tasks in the task_list
;  264     1463  1 !       ALL =           all existing tasks %((terminated as well? -tbs))%
;  265     1464  1 !       PSH =           all existing tasks matching ( /PRI and /STATE and /HOLD )
;  266     1465  1 !       T_LIST PSH =    tasks in the task_list matching ( /PRI and /STATE and /HOLD )
;  267     1466  1 !   ALGORITHM indicates the logic to implement the command, where
;  268     1467  1 !       S =     SHOW_TASK [ GET_REGISTER, DBG$TRACEBACK ]
;  269     1468  1 !       NS =    NEXT_TASK SHOW_TASK [ GET_REGISTER, DBG$TRACEBACK ]
;  270     1469  1 !       GS =    GET_PRIORITY GET_STATE SHOW_TASK [ GET_REGISTER, DBG$TRACEBACK ]
;  271     1470  1 !       ... =   repetition of the sequence
;  272     1471  1 !   The [ GET_REGISTER, DBG$TRACEBACK ] is done when /CALLS is specified.
;  273     1472  1 !
;  274     1473  1 !   /PRI or
;  275     1474  1 !   /STATE or
;  276     1475  1 !   /HOLD       /ALL    TASK_LIST       TASK SET        ALGORITHM       FAILURES
;  277     1476  1 !
;  278     1477  1 !       0       0       0               %VISIBLE        S               %((-tbs))%
;  279     1478  1 !       0       0       1               T_LIST          S...
;  280     1479  1 !       0       1       0               ALL             NS...
;  281     1480  1 !       0       1       1               T_LIST          S...
;  282     1481  1 !       1       0       0               PSH             NS...
;  283     1482  1 !       1       0       1               T_LIST PSH      GS...
;  284     1483  1 !       1       1       0               PSH             NS...
;  285     1484  1 !       1       1       1               T_LIST PSH      GS...
;  286     1485  1 !
;  287     1486  1 !   This results in four different sequences as follows:
;  288     1487  1 !   P := /PRI or /STATE or /HOLD    A := /ALL    T := TASK_LIST
;  289     1488  1 !
;  290     1489  1 !   (P + ~PA)~T         ==> NS...
;  291     1490  1 !   PT                  ==> GS...
;  292     1491  1 !   ~PT                 ==> S..
;  293     1492  1 !   ~P~A~T = ~(P+A+T)   ==> S
```

```
;  294          1493  1 !
;  295          1494  1 !-
;  296          1495  1
;  297          1496  2     BEGIN
;  298          1497  2
;  299          1498  2 MACRO
;  300          1499  2       !+
;  301          1500  2       !   $DBG_VALFLD_INI  --  Dynamically initializes a block field with a value.
;  302          1501  2       !-
;  303        M 1502  2       $DBG_VALFLD_INI (block_name, field_name, value) [] =
;  304          1503  2       block_name [field_name] = value %;
;  305          1504  2
;  306          1505  2
;  307          1506  2 KEYWORDMACRO
;  308          1507  2
;  309          1508  2       !+
;  310          1509  2       !   DBGEXT_INIT  --  Initializes the DBGEXT CONTROL BLOCK.  It BINDs the
;  311          1510  2       !   name of the block for later use by the other DBGEXT function macros,
;  312          1511  2       !   zero fills the block, sets the facility and print routine fields with
;  313          1512  2       !   predetermined values, and optionally sets other fields with the values
;  314          1513  2       !   given by the keyword parameters.
;  315          1514  2       !-
;  316        M 1515  2       DBGEXT_INIT (dbgext, function, value, number, priority, state, hold) =
;  317        M 1516  2
;  318        M 1517  2           %IF %NULL (dbgext) %THEN %WARN ('DBGEXT must be specified') %FI
;  319        M 1518  2
;  320        M 1519  2           BIND DBGEXT$$CONTROL_BLOCK = dbgext : DBGEXT$CONTROL_BLOCK;
;  321        M 1520  2
;  322        M 1521  2           CH$FILL (0, DBGEXT$K_ADA_SIZE1 * %UPVAL, CH$PTR (dbgext) );
;  323        M 1522  2
;  324        M 1523  2           DBGEXT$$CONTROL_BLOCK [DBGEXT$V_FACILITY_ID] = ADA$_FACILITY;
;  325        M 1524  2           DBGEXT$$CONTROL_BLOCK [DBGEXT$L_PRINT_ROUTINE] = DBGEXT$PRINT_ROUTINE;
;  326        M 1525  2
;  327        M 1526  2           $DBG_VALFLD_INI (DBGEXT$$CONTROL_BLOCK, DBGEXT$W_FUNCTION_CODE, function);
;  328        M 1527  2           $DBG_VALFLD_INI (DBGEXT$$CONTROL_BLOCK, DBGEXT$L_TASK_VALUE, value);
;  329        M 1528  2           $DBG_VALFLD_INI (DBGEXT$$CONTROL_BLOCK, DBGEXT$L_TASK_NUMBER, number);
;  330        M 1529  2
;  331        M 1530  2           ! MAY INITIALIZE SOME FLAG BITS %((-tbs))%
;  332        M 1531  2           !       DBGEXT$V_ALL            =
;  333        M 1532  2           !       DBGEXT$V_FULL           =
;  334        M 1533  2           !       DBGEXT$V_NO_HEADER      = NOT %NULL (no_header)
;  335        M 1534  2
;  336        M 1535  2           DBGEXT$$CONTROL_BLOCK [DBGEXT$V_PRIORITY_SPECIFIED] = NOT %NULL (priority);
;  337        M 1536  2           $DBG_VALFLD_INI (DBGEXT$$CONTROL_BLOCK, DBGEXT$L_PRIORITY, priority);
;  338        M 1537  2
;  339        M 1538  2           DBGEXT$$CONTROL_BLOCK [DBGEXT$V_STATE_SPECIFIED] = NOT %NULL (state);
;  340        M 1539  2           $DBG_VALFLD_INI (DBGEXT$$CONTROL_BLOCK, DBGEXT$V_STATE, state);
;  341        M 1540  2
;  342        M 1541  2           DBGEXT$$CONTROL_BLOCK [DBGEXT$V_HOLD_SPECIFIED] = NOT %NULL (hold);
;  343        M 1542  2           $DBG_VALFLD_INI (DBGEXT$$CONTROL_BLOCK, DBGEXT$V_HOLD, hold);
;  344          1543  2           % ;
;  345          1544  2
;  346          1545  2       !+
;  347          1546  2       !   CALL_ADA --  Calls the ADA run time system via the DEBUG External
;  348          1547  2       !   Interface.  It assumes that a DBGEXT_INIT has been performed to bind
;  349          1548  2       !   name DBGEXT$$CONTROL_BLOCK to a real control block.
;  350          1549  2       !   It optionally sets other fields with the values
```

```
;  351          1550  2      !   given by the keyword parameters.
;  352          1551  2      !-
;  353        M 1552  2      CALL_ADA (function, value, number, priority, state, hold) =
;  354        M 1553  2          BEGIN
;  355        M 1554  2          $DBG_VALFLD_INI (DBGEXT$$CONTROL_BLOCK, DBGEXT$W_FUNCTION_CODE, function);
;  356        M 1555  2          $DBG_VALFLD_INI (DBGEXT$$CONTROL_BLOCK, DBGEXT$L_TASK_VALUE, value);
;  357        M 1556  2          $DBG_VALFLD_INI (DBGEXT$$CONTROL_BLOCK, DBGEXT$L_TASK_NUMBER, number);
;  358        M 1557  2
;  359        M 1558  2          ! MAY INITIALIZE SOME FLAG BITS %((-tbs))%
;  360        M 1559  2          !       DBGEXT$V_ALL            =
;  361        M 1560  2          !       DBGEXT$V_FULL           =
;  362        M 1561  2          !       DBGEXT$V_NO_HEADER      = NOT %NULL (no_header)
;  363        M 1562  2
;  364        M 1563  2          DBGEXT$$CONTROL_BLOCK [DBGEXT$V_PRIORITY_SPECIFIED] = NOT %NULL (priority);
;  365        M 1564  2          $DBG_VALFLD_INI (DBGEXT$$CONTROL_BLOCK, DBGEXT$L_PRIORITY, priority);
;  366        M 1565  2
;  367        M 1566  2          DBGEXT$$CONTROL_BLOCK [DBGEXT$V_STATE_SPECIFIED] = NOT %NULL (state);
;  368        M 1567  2          $DBG_VALFLD_INI (DBGEXT$$CONTROL_BLOCK, DBGEXT$V_STATE, state);
;  369        M 1568  2
;  370        M 1569  2          DBGEXT$$CONTROL_BLOCK [DBGEXT$V_HOLD_SPECIFIED] = NOT %NULL (hold);
;  371        M 1570  2          $DBG_VALFLD_INI (DBGEXT$$CONTROL_BLOCK, DBGEXT$V_HOLD, hold);
;  372        M 1571  2
;  373        M 1572  2          IF NOT ADA$DBGEXT (DBGEXT$$CONTROL_BLOCK)                          ! Call ADA
;  374        M 1573  2          THEN
;  375        M 1574  2              SIGNAL (%((INTERNAL ERROR -tbs))%);
;  376        M 1575  2          IF NOT .DBGEXT$$CONTROL_BLOCK [DBGEXT$L_STATUS]                    ! and check status.
;  377        M 1576  2          THEN
;  378        M 1577  2              SIGNAL (%((SOME ERROR -tbs))%);
;  379        M 1578  2          END;
;  380          1579  2          % ;
;  381          1580  2
;  382          1581  2  MACRO
;  383          1582  2
;  384          1583  2      !+
;  385          1584  2      !   DO_NEXT_TASK  --  Calls the NEXT_TASK function without changing any
;  386          1585  2      !   fields of the control block other than FUNCTION_CODE, STATUS, and
;  387          1586  2      !   optionally TASK_VALUE.  It assumes that a DBGEXT_INIT has
;  388          1587  2      !   been performed to bind the name DBGEXT$$CONTROL_BLOCK to a real control
;  389          1588  2      !   block.  It returns the new TASK_VALUE.
;  390          1589  2      !-
;  391        M 1590  2      DO_NEXT_TASK (task) =
;  392        M 1591  2          BEGIN
;  393        M 1592  2          DBGEXT$$CONTROL_BLOCK [DBGEXT$W_FUNCTION_CODE] = DBGEXT$K_NEXT_TASK;     ! set function
;  394        M 1593  2          DBGEXT$$CONTROL_BLOCK [DBGEXT$L_STATUS] = 0;                              ! and clear status
;  395        M 1594  2          %IF NOT %NULL (TASK)                                                      ! optionally use a
;  396        M 1595  2              %THEN DBGEXT$$CONTROL_BLOCK [DBGEXT$L_TASK_VALUE] = TASK;             !  task value
;  397        M 1596  2          %FI
;  398        M 1597  2          IF NOT ADA$DBGEXT (DBGEXT$$CONTROL_BLOCK)                                 ! call ada
;  399        M 1598  2          THEN
;  400        M 1599  2              SIGNAL (%((INTERNAL ERROR -tbs))%);
;  401        M 1600  2          IF NOT .DBGEXT$$CONTROL_BLOCK [DBGEXT$L_STATUS]                           ! and check status
;  402        M 1601  2          THEN
;  403        M 1602  2              SIGNAL (%((SOME ERROR -tbs))%);
;  404        M 1603  2          .DBGEXT$$CONTROL_BLOCK [DBGEXT$L_TASK_VALUE]                              ! return the new task value
;  405          1604  2          END % ,
;  406          1605  2
;  407          1606  2      !+
```

```
;  408          1607  2      !   DO_SHOW_TASK  --  Calls the SHOW_TASK function without changing any
;  409          1608  2      !   fields of the control block other than FUNCTION_CODE, STATUS, and
;  410          1609  2      !   optionally TASK_VALUE.  It assumes that a DBGEXT_INIT has
;  411          1610  2      !   been performed to bind the name DBGEXT$$CONTROL_BLOCK to a real control
;  412          1611  2      !   block.
;  413          1612  2      !-
;  414        M 1613  2      DO_SHOW_TASK (task) =
;  415        M 1614  2          BEGIN
;  416        M 1615  2          DBGEXT$$CONTROL_BLOCK [DBGEXT$W_FUNCTION_CODE] = DBGEXT$K_SHOW_TASK;     ! set function
;  417        M 1616  2          DBGEXT$$CONTROL_BLOCK [DBGEXT$L_STATUS] = 0;                             ! and clear status
;  418        M 1617  2          %IF NOT %NULL (TASK)                                                     ! optionally use a
;  419        M 1618  2              %THEN DBGEXT$$CONTROL_BLOCK [DBGEXT$L_TASK_VALUE] = TASK;            !  task value
;  420        M 1619  2          %FI
;  421        M 1620  2          IF NOT ADA$DBGEXT (DBGEXT$$CONTROL_BLOCK)                                ! call ada
;  422        M 1621  2          THEN
;  423        M 1622  2              SIGNAL (%((INTERNAL ERROR -tbs))%);
;  424        M 1623  2          IF NOT .DBGEXT$$CONTROL_BLOCK [DBGEXT$L_STATUS]                          ! and check status
;  425        M 1624  2          THEN
;  426        M 1625  2              SIGNAL (%((SOME ERROR -tbs))%);
;  427          1626  2          END %  ,
;  428          1627  2
;  429          1628  2      !+
;  430          1629  2      !   DO_SHOW_CALLS  --  Implements part of SHOW TASK /CALLS by calling the
;  431          1630  2      !   GET_REGISTER function and passing the PC and FP to the DEBUG traceback
;  432          1631  2      !   facility, without changing any fields of the control block other than
;  433          1632  2      !   FUNCTION_CODE, STATUS, and optionally TASK_VALUE.  It assumes that a
;  434          1633  2      !   DBGEXT_INIT has been performed
;  435          1634  2      !   to bind the name DBGEXT$$CONTROL_BLOCK to a real control block.
;  436          1635  2      !-
;  437        M 1636  2      DO_SHOW_CALLS (call_level) =                  ! (task) = %((DONT THINK TASK IS NEEDED HERE -tbs))%
;  438        M 1637  2          BEGIN
;  439        M 1638  2          DBGEXT$$CONTROL_BLOCK [DBGEXT$W_FUNCTION_CODE] = DBGEXT$K_GET_REGISTERS;! set function
;  440        M 1639  2          DBGEXT$$CONTROL_BLOCK [DBGEXT$L_STATUS] = 0;                             ! and clear status
;  441        M 1640  2          %IF NOT %NULL (TASK)                                                     ! optionally use a
;  442        M 1641  2              %THEN DBGEXT$$CONTROL_BLOCK [DBGEXT$L_TASK_VALUE] = TASK;            !  task value
;  443        M 1642  2          %FI
;  444        M 1643  2          IF NOT ADA$DBGEXT (DBGEXT$$CONTROL_BLOCK)                                ! call ada
;  445        M 1644  2          THEN
;  446        M 1645  2              SIGNAL (%((INTERNAL ERROR -tbs))%);
;  447        M 1646  2
;  448        M 1647  2          IF NOT ( .DBGEXT$$CONTROL_BLOCK [DBGEXT$L_STATUS]                        ! and check status
;  449        M 1648  2              OR .DBGEXT$$CONTROL_BLOCK [DBGEXT$L_STATUS] EQL DBGEXT$K_TASK_IS_ACTIVE )
;  450        M 1649  2          THEN
;  451        M 1650  2              SIGNAL (%((SOME ERROR -tbs))%);
;  452        M 1651  2
;  453        M 1652  2          ! Check for active task and pass registers to traceback.
;  454        M 1653  2          !
;  455        M 1654  2          IF .DBGEXT$$CONTROL_BLOCK [DBGEXT$L_STATUS] EQL DBGEXT$K_TASK_IS_ACTIVE
;  456        M 1655  2          THEN                                                  ! DEBUG has the register set
;  457        M 1656  2              BEGIN
;  458        M 1657  2              LOCAL
;  459        M 1658  2                  EXC_TYPE;                 ! Exception type (trap=1, fault=2)
;  460        M 1659  2
;  461        M 1660  2              ! exception type is based on whether the last exception
;  462        M 1661  2              ! was a fault, break or step-end
;  463        M 1662  2              !
;  464        M 1663  2              IF  .dbg$runframe [dbg$v_at_fault] OR
```

```
;  465        M 1664  2                    .dbg$runframe [dbg$v_at_break] OR
;  466        M 1665  2                    .dbg$runframe [dbg$v_at_step_end]
;  467        M 1666  2                THEN        exc_type = fault_exc                    ! %(( NEED TO LIB DBGGEN -tbs))%
;  468        M 1667  2                ELSE        exc_type = trap_exc;
;  469        M 1668  2
;  470        M 1669  2                dbg$traceback (.dbg$runframe [dbg$l_user_pc],
;  471        M 1670  2                               .dbg$runframe [dbg$l_user_fp],
;  472        M 1671  2                               .EXC_TYPE, call_level );
;  473        M 1672  2                END
;  474        M 1673  2
;  475        M 1674  2            ELSE                                                    ! ADA has the register set
;  476        M 1675  2                DBG$TRACEBACK (.DBGEXT$$CONTROL_BLOCK [DBGEXT$L_PC],
;  477        M 1676  2                               .DBGEXT$$CONTROL_BLOCK [DBGEXT$L_FP],
;  478        M 1677  2                               trap_EXC, call_level );             ! %((FAULT or TRAP? -tbs))%
;  479          1678  2            END % ;
;  480          1679  2
;  481          1680  2
;  482          1681  2        LOCAL
;  483          1682  2            ADA_CONTROL : REF DBGEXT$CONTROL_BLOCK,
;  484          1683  2            ADVERB_NODE : REF DBG$ADVERB_NODE,
;  485          1684  2            NOUN_NODE : REF DBG$NOUN_NODE,
;  486          1685  2            LINK,                                               ! Link field to next adverb or noun node.
;  487          1686  2            CALLS_VALUE : INITIAL (0),
;  488          1687  2            PRIORITY_VALUE : INITIAL (0),
;  489          1688  2            STATE_VALUE : INITIAL (0),
;  490          1689  2            QUALIFIERS : BITVECTOR [TASK_MAX_QUAL + 1]          ! Qualifier state vector.
;  491          1690  2                         INITIAL (BYTE (REP TASK_MAX_QUAL / %BPUNIT + 1 OF (0)));
;  492          1691  2
;  493          1692  2        !+
;  494          1693  2        !   Walk the tree and set bits in the qualifier state vector.  Also pick up the values of the adverb
;  495          1694  2        !   nodes representing the parameters supplied to the /CALLS, /PRIORITY, and /STATE qualifiers.  This
;  496          1695  2        !   algorithm will cause the last value to superceed earlier values, when multiple values are given.
;  497          1696  2        !-
;  498          1697  2        IF .VERB_NODE [DBG$L_VERB_OBJECT_PTR] NEQ 0             ! Check for an explicit task list.
;  499          1698  2        THEN
;  500          1699  2            QUALIFIERS [TASK_TASK_LIST] = TRUE;
;  501          1700  2        LINK = VERB_NODE [DBG$L_VERB_ADVERB_PTR];               ! Get link to the adverb nodes.
;  502          1701  2        WHILE ..LINK NEQ 0  DO                                  ! Chain down the adverb nodes.
;  503          1702  3            BEGIN
;  504          1703  3            ADVERB_NODE = ..LINK;
;  505          1704  3            QUALIFIERS [ .ADVERB_NODE [DBG$B_ADVERB_LITERAL] ] = TRUE;
;  506          1705  3            SELECTONE .ADVERB_NODE [DBG$B_ADVERB_LITERAL] OF
;  507          1706  3                SET
;  508          1707  3                [ TASK_CALLS ] :
;  509          1708  3                    CALLS_VALUE = .ADVERB_NODE [DBG$L_ADVERB_VALUE];
;  510          1709  3                [ TASK_PRIORITY ] :
;  511          1710  3                    PRIORITY_VALUE = .ADVERB_NODE [DBG$L_ADVERB_VALUE];
;  512          1711  3                [ TASK_STATE ] :
;  513          1712  3                    STATE_VALUE = .ADVERB_NODE [DBG$L_ADVERB_VALUE];
;  514          1713  3                TES;
;  515          1714  3            LINK = ADVERB_NODE [DBG$L_ADVERB_LINK];             ! Link to next node.
;  516          1715  2            END;
;  517          1716  2
;  518          1717  2        !+
;  519          1718  2        !   Check for conflicting qualifiers and parameters.  %((what about /FULL ? -tbs))%
;  520          1719  2        !-
;  521          1720  3        IF CONFLICT (QUALIFIERS, (CALLS, DEADLOCK, STATISTICS) )    ! Only one action allowed.
```

```
;  522          1721  2      THEN
;  523          1722  2          SIGNAL (DBG$_CONFLICT);
;  524          1723  2
;  525          1724  2      !+
;  526          1725  2      !   Get a control block.
;  527          1726  2      !-
;  528          1727  2      IF .QUALIFIERS [TASK_CALLS]
;  529          1728  2      THEN
;  530          1729  2          ADA_CONTROL = DBG$GET_TEMPMEM (DBGEXT$K_ADA_SIZE2)      ! Need long block.
;  531          1730  2      ELSE
;  532          1731  2          ADA_CONTROL = DBG$GET_TEMPMEM (DBGEXT$K_ADA_SIZE1);     ! Need short block.
;  533          1732  2
;  534          1733  2      !+
;  535          1734  2      !   Fill out the control block and perform the required action.
;  536          1735  2      !-
;  537          1736  2      SELECTONE TRUE OF
;  538          1737  2          SET
;  539          1738  2
;  540          1739  2          !+
;  541          1740  2          !   SHOW TASK /DEADLOCK
;  542          1741  2          !-
;  543          1742  2          [ .QUALIFIERS [TASK_DEADLOCK] ] :
;  544          1743  3              BEGIN
;  545        P 1744  3              DBGEXT_INIT (DBGEXT = .ADA_CONTROL,                 ! Initialize block
;  546          1745  3                  FUNCTION = DBGEXT$K_SHOW_DEADLOCK);             !  and set function.
;  547          1746  3              IF NOT ADA$DBGEXT (.ADA_CONTROL)                    ! Call ADA
;  548          1747  3              THEN
;  549          1748  3                  SIGNAL (%((INTERNAL ERROR -tbs))%);
;  550          1749  3              IF NOT .DBGEXT$$CONTROL_BLOCK [DBGEXT$L_STATUS]     ! and check status.
;  551          1750  3              THEN
;  552          1751  3                  SIGNAL (%((SOME ERROR -tbs))%);
;  553          1752  2              END;
;  554          1753  2
;  555          1754  2          !+
;  556          1755  2          !   SHOW TASK /STATISTICS
;  557          1756  2          !-
;  558          1757  2          [ .QUALIFIERS [TASK_STATISTICS] ] :
;  559          1758  3              BEGIN
;  560        P 1759  3              DBGEXT_INIT (DBGEXT = .ADA_CONTROL,                 ! Initialize block
;  561        P 1760  3  !%((-tbs))%                 FUNCTION = DBGEXT$K_SHOW_STATISTICS);          !  and set function.
;  562          1761  3                  FUNCTION = DBGEXT$K_SHOW_STAT);         !  and set function.
;  563          1762  3              IF NOT ADA$DBGEXT (.ADA_CONTROL)                    ! Call ADA
;  564          1763  3              THEN
;  565          1764  3                  SIGNAL (%((INTERNAL ERROR -tbs))%);
;  566          1765  3              IF NOT .DBGEXT$$CONTROL_BLOCK [DBGEXT$L_STATUS]     ! and check status.
;  567          1766  3              THEN
;  568          1767  3                  SIGNAL (%((SOME ERROR -tbs))%);
;  569          1768  2              END;
;  570          1769  2
;  571          1770  2          !+
;  572          1771  2          !   SHOW TASK or SHOW TASK /CALLS
;  573          1772  2          !-
;  574          1773  2          [ OTHERWISE ] :
;  575          1774  3              BEGIN
;  576          1775  3              BIND
;  577          1776  3                  ALL = .QUALIFIERS [TASK_ALL],
;  578          1777  3                  LIST = .QUALIFIERS [TASK_TASK_LIST],
```

```
;  579          1778  3                PSH = .QUALIFIERS [TASK_PRIORITY] OR .QUALIFIERS [TASK_STATE] OR .QUALIFIERS [TASK_HOLD];
;  580          1779  3
;  581          1780  3            SELECTONE TRUE OF
;  582          1781  3                SET
;  583          1782  3
;  584          1783  3                !   (P + ^PA)^T          ==> NS...
;  585          1784  3                [ (PSH OR (NOT PSH AND ALL)) AND NOT LIST ] :
;  586          1785  3
;  587          1786  4                    BEGIN
;  588          1787  4                    LOCAL
;  589          1788  4                        FIRST_TASK;
;  590        P 1789  4                    DBGEXT_INIT (DBGEXT = .ADA_CONTROL,
;  591        P 1790  4                                 PRIORITY = .PRIORITY_VALUE,
;  592        P 1791  4                                 STATE = .STATE_VALUE,
;  593          1792  4                                 HOLD = .QUALIFIERS [TASK_HOLD] );
;  594          1793  4                    FIRST_TASK = DO_NEXT_TASK (0);
;  595          1794  4                    IF FIRST_TASK EQLU 0                ! null task ==> EXIT
;  596          1795  4                    THEN
;  597          1796  4                        SIGNAL (%((NO TASKS MATCH RESTRICTION -tbs))%);
;  598          1797  4                    DO
;  599          1798  5                        BEGIN
;  600          1799  5                        DO_SHOW_TASK ();                %((HEADER CONTROL NEEDED -tbs))%        !
;  601          1800  5                        IF .QUALIFIERS [TASK_CALLS]
;  602          1801  5                        THEN
;  603          1802  5                            DO_SHOW_CALLS (.CALLS_VALUE);
;  604          1803  5                        END
;  605          1804  4                    UNTIL .FIRST_TASK EQLU DO_NEXT_TASK ();     ! cycled through all tasks
;  606          1805  3                    END;
;  607          1806  3
;  608          1807  3                !   PT                   ==> GS...
;  609          1808  3                [ PSH AND LIST ] :
;  610          1809  4                    BEGIN
;  611          1810  4                    DBGEXT_INIT (DBGEXT = .ADA_CONTROL);
;  612          1811  4                    !+
;  613          1812  4                    !   Walk down the chain of noun nodes.  Pick up the pointer to %((THE PRIMARY DESC -tbs)
;  614          1813  4                    !   and the value of the task.  Then do the SHOW_TASK.
;  615          1814  4                    !-
;  616          1815  4                    LINK = VERB_NODE [DBG$L_VERB_OBJECT_PTR];           ! Get link to the noun nodes.
;  617          1816  4                    WHILE ..LINK NEQ 0  DO                              ! Chain down the noun nodes.
;  618          1817  5                        BEGIN
;  619          1818  5                        LABEL
;  620          1819  5                            CHECK_PSH;
;  621          1820  5                        NOUN_NODE = ..LINK;
;  622          1821  5 !                      <task_value> =  (.NOUN_NODE [DBG$L_NOUN_VALUE]) [<task_value_field>] ;  %((need stru
;  623          1822  5
;  624          1823  5                        !+
;  625          1824  5                        !   Check PRIORITY, STATE, and HOLD
;  626          1825  5                        !-
;  627          1826  5                        CHECK_PSH:
;  628          1827  6                            BEGIN
;  629          1828  6                            SELECT TRUE OF
;  630          1829  6                                SET
;  631          1830  6
;  632          1831  6                                [ .QUALIFIERS [TASK_PRIORITY] ] :
;  633          1832  7                                    BEGIN
;  634          1833  7                                    CALL_ADA (FUNCTION = DBGEXT$K_GET_PRIORITY);
;  635          1834  7                                    IF .ADA_CONTROL [DBGEXT$L_PRIORITY] AND .PRIORITY_VALUE EQL 0
```

```
;  636     1835  7                                    THEN
;  637     1836  7                                        LEAVE CHECK_PSH;
;  638     1837  6                                    END;
;  639     1838  6
;  640     1839  6                                [ .QUALIFIERS [TASK_STATE] ] :
;  641     1840  7                                    BEGIN
;  642     1841  7                                    CALL_ADA (FUNCTION = DBGEXT$K_GET_STATE);
;  643     1842  7                                    IF .ADA_CONTROL [DBGEXT$V_STATE] AND .STATE_VALUE EQL 0
;  644     1843  7                                    THEN
;  645     1844  7                                        LEAVE CHECK_PSH;
;  646     1845  6                                    END;
;  647     1846  6
;  648     1847  6                                [ .QUALIFIERS [TASK_HOLD] ] :
;  649     1848  7                                    BEGIN
;  650     1849  7                                    CALL_ADA (FUNCTION = DBGEXT$K_GET_STATE);
;  651     1850  7                                    IF NOT .ADA_CONTROL [DBGEXT$V_HOLD]
;  652     1851  7                                    THEN
;  653     1852  7                                        LEAVE CHECK_PSH;
;  654     1853  6                                    END;
;  655     1854  6
;  656     1855  6                                    TES;
;  657     1856  6
;  658     1857  6                                DO_SHOW_TASK ();            ! <task_value> %((-tbs))%
;  659     1858  6                                IF .QUALIFIERS [TASK_CALLS]
;  660     1859  6                                THEN
;  661     1860  6                                    DO_SHOW_CALLS (.CALLS_VALUE);
;  662     1861  5                                END;
;  663     1862  5
;  664     1863  5                            LINK = NOUN_NODE [DBG$L_NOUN_LINK];          ! Link to next node.
;  665     1864  4                            END;
;  666     1865  3                        END;
;  667     1866  3
;  668     1867  3                    !   ~PT                 ==> S..
;  669     1868  3                    [ NOT PSH AND LIST ] :
;  670     1869  4                        BEGIN
;  671     1870  4                        DBGEXT_INIT (DBGEXT = .ADA_CONTROL);
;  672     1871  4                        !+
;  673     1872  4                        !   Walk down the chain of noun nodes.  Pick up the pointer to %((THE PRIMARY DESC -tbs)
;  674     1873  4                        !   and the value of the task.  Then do the SHOW_TASK.
;  675     1874  4                        !-
;  676     1875  4                        LINK = VERB_NODE [DBG$L_VERB_OBJECT_PTR];        ! Get link to the noun nodes.
;  677     1876  4                        WHILE ..LINK NEQ 0  DO                           ! Chain down the noun nodes.
;  678     1877  5                            BEGIN
;  679     1878  5                            NOUN_NODE = ..LINK;
;  680     1879  5 !                          <task_value> =  (.NOUN_NODE [DBG$L_NOUN_VALUE]) [<task_value_field>] ;  %((need stru
;  681     1880  5
;  682     1881  5                            DO_SHOW_TASK ();            ! <task_value> %((-tbs))%
;  683     1882  5                            IF .QUALIFIERS [TASK_CALLS]
;  684     1883  5                            THEN
;  685     1884  5                                DO_SHOW_CALLS (.CALLS_VALUE);
;  686     1885  5                            LINK = NOUN_NODE [DBG$L_NOUN_LINK];          ! Link to next node.
;  687     1886  4                            END;
;  688     1887  3                        END;
;  689     1888  3
;  690     1889  3                    !   ~P~A~T = ~(P+A+T)   ==> S
;  691     1890  3                    [ NOT (PSH AND ALL AND LIST) ] :
;  692     1891  4                        BEGIN
```

```
4                   DBGEXT_INIT (DBGEXT = .ADA_CONTROL);
4                   DO_SHOW_TASK ();                  ! %((NEED GLOBAL SYMBOL FOR %VISIBLE -tbs))%
4                   IF .QUALIFIERS [TASK_CALLS]
4                   THEN
4                       DO_SHOW_CALLS (.CALLS_VALUE);
3                   END;
3
3               TES
3
2           END;
2
2       TES;
2
2
2   RETURN 0;                       ! %((0?-tbs))%
2
1   END;                            ! end of DBG$NEXECUTE_SHOW_TASK



                              0FFC 00000        .ENTRY  DBG$NEXECUTE_SHOW_TASK, Save R2,R3,R4,R5,-  ; 1437
                                                        R6,R7,R8,R9,R10,R11                        ;
                   5E      04  C2 00002        SUBL2   #4, SP                                      ;
                           7E  D4 00005        CLRL    PRIORITY_VALUE                              ; 1496
                           5A  7C 00007        CLRQ    CALLS_VALUE                                 ;
                           59  B4 00009        CLRW    QUALIFIERS                                  ;
                   58  04  AC  D0 0000B        MOVL    VERB_NODE, R8                               ; 1697
                       08  A8  D5 0000F        TSTL    8(R8)                                       ;
                           03  13 00012        BEQL    1$                                          ;
                   59      01  88 00014        BISB2   #1, QUALIFIERS                              ; 1699
                   56  04  A8  9E 00017 1$:    MOVAB   4(R8), LINK                                 ; 1700
                           66  D5 0001B 2$:    TSTL    (LINK)                                      ; 1701
                           2F  13 0001D        BEQL    7$                                          ;
                   50      66  D0 0001F        MOVL    (LINK), ADVERB_NODE                         ; 1703
                   51      60  9A 00022        MOVZBL  (ADVERB_NODE), R1                           ; 1704
             00    59      51  E2 00025        BBSS    R1, QUALIFIERS, 3$                          ;
                   03      60  91 00029 3$:    CMPB    (ADVERB_NODE), #3                           ; 1707
                           06  12 0002C        BNEQ    4$                                          ;
                   5A  04  A0  D0 0002E        MOVL    4(ADVERB_NODE), CALLS_VALUE                 ; 1708
                           14  11 00032        BRB     6$                                          ;
                   07      60  91 00034 4$:    CMPB    (ADVERB_NODE), #7                           ; 1709
                           06  12 00037        BNEQ    5$                                          ;
                   6E  04  A0  D0 00039        MOVL    4(ADVERB_NODE), PRIORITY_VALUE              ; 1710
                           09  11 0003D        BRB     6$                                          ;
                   0A      60  91 0003F 5$:    CMPB    (ADVERB_NODE), #10                          ; 1711
                           04  12 00042        BNEQ    6$                                          ;
                   5B  04  A0  D0 00044        MOVL    4(ADVERB_NODE), STATE_VALUE                 ; 1712
                   56  08  A0  9E 00048 6$:    MOVAB   8(R0), LINK                                 ; 1714
                           CD  11 0004C        BRB     2$                                          ; 1701
   50        59    01      03  EF 0004E 7$:    EXTZV   #3, #1, QUALIFIERS, R0                      ; 1720
   51        59    01      04  EF 00053        EXTZV   #4, #1, QUALIFIERS, R1                      ;
                   50      51  C0 00058        ADDL2   R1, R0                                      ;
   52        59    01      0B  EF 0005B        EXTZV   #11, #1  QUALIFIERS, R2                     ;
                   50      52  C0 00060        ADDL2   R2, R0                                      ;
                   01      50  D1 00063        CMPL    R0, #1                                      ;
```

```
                                        0D  15 00066          BLEQ    8$
                              00028158  8F  DD 00068          PUSHL   #164184                                   ; 1722
                 00000000G  00          01  FB 0006E          CALLS   #1, LIB$SIGNAL                            ;
             04             59          03  E1 00075 8$:      BBC     #3, QUALIFIERS, 9$                        ; 1727
                                        1B  DD 00079          PUSHL   #27                                       ; 1729
                                        02  11 0007B          BRB     10$                                       ;
                                        0A  DD 0007D 9$:      PUSHL   #10                                       ; 1731
                 00000000G  00          01  FB 0007F 10$:     CALLS   #1, DBG$GET_TEMPMEM                       ;
                            57          50  D0 00086          MOVL    R0, ADA_CONTROL                           ;
             27             59          04  E1 00089          BBC     #4, QUALIFIERS, 11$                       ; 1742
      28     00             6E          00  2C 0008D          MOVC5   #0, (SP), #0, #40, (ADA_CONTROL)          ; 1745
                                        67     00092                                                            ;
02    A7     0C             00          31  F0 00093          INSV    #49, #0, #12, 2(ADA_CONTROL)              ;
                     20     A7    0000V CF  9E 00099          MOVAB   DBGEXT$PRINT_ROUTINE, 32(ADA_CONTROL)     ;
                            67          06  B0 0009F          MOVW    #6, (ADA_CONTROL)                         ;
                     18     A7          07  8A 000A2          BICB2   #7, 24(ADA_CONTROL)                       ;
                                        57  DD 000A6          PUSHL   ADA_CONTROL                               ; 1746
                 00000000G  00          01  FB 000A8          CALLS   #1, ADA$DBGEXT                            ;
                            2B          50  E9 000AF          BLBC    R0, 12$                                   ;
                                        30  11 000B2          BRB     13$                                       ; 1749
             39             59          0B  E1 000B4 11$:     BBC     #11, QUALIFIERS, 15$                      ; 1757
      28     00             6E          00  2C 000B8          MOVC5   #0, (SP), #0, #40, (ADA_CONTROL)          ; 1761
                                        67     000BD                                                            ;
02    A7     0C             00          31  F0 000BE          INSV    #49, #0, #12, 2(ADA_CONTROL)              ;
                     20     A7    0000V CF  9E 000C4          MOVAB   DBGEXT$PRINT_ROUTINE, 32(ADA_CONTROL)     ;
                            67          05  B0 000CA          MOVW    #5, (ADA_CONTROL)                         ;
                     18     A7          07  8A 000CD          BICB2   #7, 24(ADA_CONTROL)                       ;
                                        57  DD 000D1          PUSHL   ADA_CONTROL                               ; 1762
                 00000000G  00          01  FB 000D3          CALLS   #1, ADA$DBGEXT                            ;
                            07          50  E8 000DA          BLBS    R0, 13$                                   ;
                 00000000G  00          00  FB 000DD 12$:     CALLS   #0, LIB$SIGNAL                            ; 1764
                            01       04 A7  E9 000E4 13$:     BLBC    4(ADA_CONTROL), 14$                       ; 1765
                                        04     000E8          RET                                               ;
                 00000000G  00          00  FB 000E9 14$:     CALLS   #0, LIB$SIGNAL                            ; 1767
                                        04     000F0          RET                                               ; 1736
      50     59             01          07  EF 000F1 15$:     EXTZV   #7, #1, QUALIFIERS, R0                    ; 1778
      51     59             01          0A  EF 000F6          EXTZV   #10, #1, QUALIFIERS, R1                   ;
                            50          51  C8 000FB          BISL2   R1, R0                                    ;
      52     59             01          06  EF 000FE          EXTZV   #6, #1, QUALIFIERS, R2                    ;
                            50          52  C8 00103          BISL2   R2, R0                                    ;
      51     59             01          02  EF 00106          EXTZV   #2, #1, QUALIFIERS, R1                    ; 1784
                            51          50  CA 0010B          BICL2   R0, R1                                    ;
                            51          50  C8 0010E          BISL2   R0, R1                                    ;
      52     59             01          00  EF 00111          EXTZV   #0, #1, QUALIFIERS, R2                    ;
                            51          52  CA 00116          BICL2   R2, R1                                    ;
                            01          51  D1 00119          CMPL    R1, #1                                    ;
                                        03  13 0011C          BEQL    16$                                       ;
                                      012F  31 0011E          BRW     33$                                       ;
      28     00             6E          00  2C 00121 16$:     MOVC5   #0, (SP), #0, #40, (ADA_CONTROL)          ; 1792
                                        67     00126                                                            ;
02    A7     0C             00          31  F0 00127          INSV    #49, #0, #12, 2(ADA_CONTROL)              ;
                     20     A7    0000V CF  9E 0012D          MOVAB   DBGEXT$PRINT_ROUTINE, 32(ADA_CONTROL)     ;
                            50       18 A7  9E 00133          MOVAB   24(ADA_CONTROL), R0                       ;
                            60          04  88 00137          BISB2   #4, (R0)                                  ;
                     1C     A7          6E  D0 0013A          MOVL    PRIORITY_VALUE, 28(ADA_CONTROL)           ;
                            60          02  88 0013E          BISB2   #2, (R0)                                  ;
02    A0     04             00          5B  F0 00141          INSV    STATE_VALUE, #0, #4, 2(R0)                ;
```

```
                                  60           01 88 00147        BISB2   #1, (R0)                              ;
        51          59            01           06 EF 0014A        EXTZV   #6, #1, QUALIFIERS, R1                ;
        60          01            14           51 F0 0014F        INSV    R1, #20, #1, (R0)                     ;
                                  67           03 B0 00154        MOVW    #3, (ADA_CONTROL)                     ; 1793
                                  52      04   A7 9E 00157        MOVAB   4(ADA_CONTROL), R2                    ;
                                               62 D4 0015B        CLRL    (R2)                                  ;
                                          10   A7 D4 0015D        CLRL    16(ADA_CONTROL)                       ;
                                               57 DD 00160        PUSHL   ADA_CONTROL                           ;
                      00000000G   00           01 FB 00162        CALLS   #1, ADA$DBGEXT                        ;
                                  07           50 E8 00169        BLBS    R0, 17$                               ;
                      00000000G   00           00 FB 0016C        CALLS   #0, LIB$SIGNAL                        ;
                                  07           62 E8 00173 17$:   BLBS    (R2), 18$                             ;
                      00000000G   00           00 FB 00176        CALLS   #0, LIB$SIGNAL                        ;
                             04   AE      10   A7 D0 0017D 18$:   MOVL    16(ADA_CONTROL), FIRST_TASK           ;
                                  50      04   AE 9E 00182        MOVAB   FIRST_TASK, R0                        ; 1794
                                               07 12 00186        BNEQ    19$                                   ;
                      00000000G   00           00 FB 00188        CALLS   #0, LIB$SIGNAL                        ; 1796
                                  67           04 B0 0018F 19$:   MOVW    #4, (ADA_CONTROL)                     ; 1799
                                               62 D4 00192        CLRL    (R2)                                  ;
                                               57 DD 00194        PUSHL   ADA_CONTROL                           ;
                      00000000G   00           01 FB 00196        CALLS   #1, ADA$DBGEXT                        ;
                                  07           50 E8 0019D        BLBS    R0, 20$                               ;
                      00000000G   00           00 FB 001A0        CALLS   #0, LIB$SIGNAL                        ;
                                  07           62 E8 001A7 20$:   BLBS    (R2), 21$                             ;
                      00000000G   00           00 FB 001AA        CALLS   #0, LIB$SIGNAL                        ;
                    6E            59           03 E1 001B1 21$:   BBC     #3, QUALIFIERS, 29$                   ; 1800
                                  67           0F B0 001B5        MOVW    #15, (ADA_CONTROL)                    ; 1802
                                               62 D4 001B8        CLRL    (R2)                                  ;
                                               57 DD 001BA        PUSHL   ADA_CONTROL                           ;
                      00000000G   00           01 FB 001BC        CALLS   #1, ADA$DBGEXT                        ;
                                  07           50 E8 001C3        BLBS    R0, 22$                               ;
                      00000000G   00           00 FB 001C6        CALLS   #0, LIB$SIGNAL                        ;
                                  0C           62 E8 001CD 22$:   BLBS    (R2), 23$                             ;
                                  02           62 D1 001D0        CMPL    (R2), #2                              ;
                                               07 13 001D3        BEQL    23$                                   ;
                      00000000G   00           00 FB 001D5        CALLS   #0, LIB$SIGNAL                        ;
                                  02           62 D1 001DC 23$:   CMPL    (R2), #2                              ;
                                               31 12 001DF        BNEQ    27$                                   ;
                    0F 00000000G  00           05 E0 001E1        BBS     #5, DBG$RUNFRAME+73, 24$              ;
                                  08 00000000G 00 E8 001E9        BLBS    DBG$RUNFRAME+72, 24$                  ;
                    05 00000000G  00           04 E1 001F0        BBC     #4, DBG$RUNFRAME+73, 25$              ;
                                  50           02 D0 001F8 24$:   MOVL    #2, EXC_TYPE                          ;
                                               03 11 001FB        BRB     26$                                   ;
                                  50           01 D0 001FD 25$:   MOVL    #1, EXC_TYPE                          ;
                                        0401   8F BB 00200 26$:   PUSHR   #^M<R0,R10>                           ;
                                   00000000G   00 DD 00204        PUSHL   DBG$RUNFRAME+56                       ;
                                   00000000G   00 DD 0020A        PUSHL   DBG$RUNFRAME+64                       ;
                                               0A 11 00210        BRB     28$                                   ;
                                               5A DD 00212 27$:   PUSHL   CALLS_VALUE                           ;
                                               01 DD 00214        PUSHL   #1                                    ;
                                          5C   A7 DD 00216        PUSHL   92(ADA_CONTROL)                       ;
                                          64   A7 DD 00219        PUSHL   100(ADA_CONTROL)                      ;
                      00000000G   00           04 FB 0021C 28$:   CALLS   #4, DBG$TRACEBACK                     ;
                                  67           03 B0 00223 29$:   MOVW    #3, (ADA_CONTROL)                     ; 1804
                                               62 D4 00226        CLRL    (R2)                                  ;
                                               57 DD 00228        PUSHL   ADA_CONTROL                           ;
                      00000000G   00           01 FB 0022A        CALLS   #1, ADA$DBGEXT                        ;
```

```
                            07         50  E8 00231        BLBS    R0, 30$                                       ;
                 00000000G  00         00  FB 00234        CALLS   #0, LIB$SIGNAL                                ;
                            07         62  E8 0023B 30$:   BLBS    (R2), 31$                                     ;
                 00000000G  00         00  FB 0023E        CALLS   #0, LIB$SIGNAL                                ;
                        10  A7    04   AE  D1 00245 31$:   CMPL    FIRST_TASK, 16(ADA_CONTROL)                   ;
                                       03  13 0024A        BEQL    32$                                           ;
                                     FF40  31 0024C        BRW     19$                                           ;
                                       04  04 0024F 32$:   RET                                                   ; 1780
     51            59       01         00  EF 00250 33$:   EXTZV   #0, #1, QUALIFIERS, R1                        ; 1808
                            51         51  D2 00255        MCOML   R1, R1                                        ;
                   51       50         51  CB 00258        BICL3   R1, R0, R1                                    ;
                            01         51  D1 0025C        CMPL    R1, #1                                        ;
                                       03  13 0025F        BEQL    34$                                           ;
                                     0156  31 00261        BRW     57$                                           ;
     28            00       6E         00  2C 00264 34$:   MOVC5   #0, (SP), #0, #40, (ADA_CONTROL)              ; 1810
                                       67     00269                                                              ;
02   A7            0C       00         31  F0 0026A        INSV    #49, #0, #12, 2(ADA_CONTROL)                  ;
                        20  A7  0000V  CF  9E 00270        MOVAB   DBGEXT$PRINT_ROUTINE, 32(ADA_CONTROL)         ;
                            53    18   A7  9E 00276        MOVAB   24(ADA_CONTROL), R3                           ;
                            63         07  8A 0027A        BICB2   #7, (R3)                                      ;
                            56    08   A8  9E 0027D        MOVAB   8(R8), LINK                                   ; 1815
                                       66  D5 00281 35$:   TSTL    (LINK)                                        ; 1816
                                       01  12 00283        BNEQ    36$                                           ;
                                       04  04 00285        RET                                                   ;
                            52         66  D0 00286 36$:   MOVL    (LINK), NOUN_NODE                             ; 1820
                                       59  95 00289        TSTB    QUALIFIERS                                    ; 1831
                                       2C  18 0028B        BGEQ    39$                                           ;
                            67         0C  B0 0028D        MOVW    #12, (ADA_CONTROL)                            ; 1833
                            63         07  8A 00290        BICB2   #7, (R3)                                      ;
                                       57  DD 00293        PUSHL   ADA_CONTROL                                   ;
                 00000000G  00         01  FB 00295        CALLS   #1, ADA$DBGEXT                                ;
                            07         50  E8 0029C        BLBS    R0, 37$                                       ;
                 00000000G  00         00  FB 0029F        CALLS   #0, LIB$SIGNAL                                ;
                            07    04   A7  E8 002A6 37$:   BLBS    4(ADA_CONTROL), 38$                           ;
                 00000000G  00         00  FB 002AA        CALLS   #0, LIB$SIGNAL                                ;
                            04    1C   A7  E9 002B1 38$:   BLBC    28(ADA_CONTROL), 39$                          ; 1834
                                       6E  D5 002B5        TSTL    PRIORITY_VALUE                                ;
                                       5D  13 002B7        BEQL    45$                                           ;
                   2C       59         0A  E1 002B9 39$:   BBC     #10, QUALIFIERS, 42$                          ; 1839
                            67         07  B0 002BD        MOVW    #7, (ADA_CONTROL)                             ; 1841
                            63         07  8A 002C0        BICB2   #7, (R3)                                      ;
                                       57  DD 002C3        PUSHL   ADA_CONTROL                                   ;
                 00000000G  00         01  FB 002C5        CALLS   #1, ADA$DBGEXT                                ;
                            07         50  E8 002CC        BLBS    R0, 40$                                       ;
                 00000000G  00         00  FB 002CF        CALLS   #0, LIB$SIGNAL                                ;
                            07    04   A7  E8 002D6 40$:   BLBS    4(ADA_CONTROL), 41$                           ;
                 00000000G  00         00  FB 002DA        CALLS   #0, LIB$SIGNAL                                ;
                            04    1A   A7  E9 002E1 41$:   BLBC    26(ADA_CONTROL), 42$                          ; 1842
                                       5B  D5 002E5        TSTL    STATE_VALUE                                   ;
                                       2D  13 002E7        BEQL    45$                                           ;
                   2C       59         06  E1 002E9 42$:   BBC     #6, QUALIFIERS, 46$                           ; 1847
                            67         07  B0 002ED        MOVW    #7, (ADA_CONTROL)                             ; 1849
                            63         07  8A 002F0        BICB2   #7, (R3)                                      ;
                                       57  DD 002F3        PUSHL   ADA_CONTROL                                   ;
                 00000000G  00         01  FB 002F5        CALLS   #1, ADA$DBGEXT                                ;
                            07         50  E8 002FC        BLBS    R0, 43$                                       ;
                 00000000G  00         00  FB 002FF        CALLS   #0, LIB$SIGNAL                                ;
```

```
                    07        04   A7  E8 00306 43$:    BLBS    4(ADA_CONTROL), 44$                    ;
                00000000G     00        FB 0030A         CALLS   #0, LIB$SIGNAL                        ;
                    03        1A   A7       04  E0 00311 44$:    BBS     #4, 26(ADA_CONTROL), 46$       ; 1850
                                       009A  31 00316 45$:    BRW     56$                            ;
                    67                 04  B0 00319 46$:    MOVW    #4, (ADA_CONTROL)              ; 1857
                              04   A7       D4 0031C         CLRL    4(ADA_CONTROL)                 ;
                                            57 DD 0031F         PUSHL   ADA_CONTROL                    ;
                00000000G     00             01 FB 00321         CALLS   #1, ADA$DBGEXT                 ;
                    07                       50 E8 00328         BLBS    R0, 47$                        ;
                00000000G     00             00 FB 0032B         CALLS   #0, LIB$SIGNAL                 ;
                    07        04   A7       E8 00332 47$:    BLBS    4(ADA_CONTROL), 48$            ;
                00000000G     00             00 FB 00336         CALLS   #0, LIB$SIGNAL                 ;
                    72                       03 E1 0033D 48$:    BBC     #3, QUALIFIERS, 56$            ; 1858
                    67                       0F B0 00341         MOVW    #15, (ADA_CONTROL)             ; 1860
                              04   A7       D4 00344         CLRL    4(ADA_CONTROL)                 ;
                                            57 DD 00347         PUSHL   ADA_CONTROL                    ;
                00000000G     00             01 FB 00349         CALLS   #1, ADA$DBGEXT                 ;
                    07                       50 E8 00350         BLBS    R0, 49$                        ;
                00000000G     00             00 FB 00353         CALLS   #0, LIB$SIGNAL                 ;
                    0D        04   A7       E8 0035A 49$:    BLBS    4(ADA_CONTROL), 50$            ;
                    02        04   A7       D1 0035E         CMPL    4(ADA_CONTROL), #2             ;
                                            07 13 00362         BEQL    50$                            ;
                00000000G     00             00 FB 00364         CALLS   #0, LIB$SIGNAL                 ;
                    02        04   A7       D1 0036B 50$:    CMPL    4(ADA_CONTROL), #2             ;
                                            31 12 0036F         BNEQ    54$                            ;
                0F 00000000G  00             05 E0 00371         BBS     #5, DBG$RUNFRAME+73, 51$       ;
                              08 00000000G   00 E8 00379         BLBS    DBG$RUNFRAME+72, 51$           ;
                05 00000000G  00             04 E1 00380         BBC     #4, DBG$RUNFRAME+73, 52$       ;
                    50                       02 D0 00388 51$:    MOVL    #2, EXC_TYPE                   ;
                                            03 11 0038B         BRB     53$                            ;
                    50                       01 D0 0038D 52$:    MOVL    #1, EXC_TYPE                   ;
                              0401           8F BB 00390 53$:    PUSHR   #^M<R0,R10>                    ;
                          00000000G          00 DD 00394         PUSHL   DBG$RUNFRAME+56                ;
                          00000C0G           00 DD 0039A         PUSHL   DBG$RUNFRAME+64                ;
                                            0A 11 003A0         BRB     55$                            ;
                                            5A DD 003A2 54$:    PUSHL   CALLS_VALUE                    ;
                                            01 DD 003A4         PUSHL   #1                             ;
                              5C   A7       DD 003A6         PUSHL   92(ADA_CONTROL)                ;
                              64   A7       DD 003A9         PUSHL   100(ADA_CONTROL)               ;
                    00000000G     00       04 FB 003AC 55$:    CALLS   #4, DBG$TRACEBACK              ;
                                   56   08  A2 9E 003B3 56$:    MOVAB   8(R2), LINK                    ; 1863
                                       FEC7  31 003B7         BRW     35$                            ; 1816
                    51                 59    01    00 EF 003BA 57$:    EXTZV   #0, #1, QUALIFIERS, R1         ; 1868
                                            51    50 CA 003BF         BICL2   R0, R1                         ;
                                            01    51 D1 003C2         CMPL    R1, #1                         ;
                                            03    13 003C5         BEQL    58$                            ;
                                          00C3    31 003C7         BRW     71$                            ;
                    28                 00    6E    00 2C 003CA 58$:    MOVC5   #0, (SP), #0, #40, (ADA_CONTROL)       ; 1870
                                            67       003CF                                                  ;
                02  A7                 0C          00 F0 003D0         INSV    #49, #0, #12, 2(ADA_CONTROL)   ;
                                   20   A7    0000V  CF 9E 003D6         MOVAB   DBGEXT$PRINT_ROUTINE, 32(ADA_CONTROL)  ;
                                   18   A7          07 8A 003DC         BICB2   #7, 24(ADA_CONTROL)            ;
                                   56   08    A8    9E 003E0         MOVAB   8(R8), LINK                    ; 1875
                                                66 D5 003E4 59$:    TSTL    (LINK)                         ; 1876
                                                01 12 003E6         BNEQ    60$                            ;
                                                04 003E8         RET                                    ;
                                   52           66 D0 003E9 60$:    MOVL    (LINK), NOUN_NODE              ; 1878
```

```
                                67              04  B0 003EC          MOVW    #4, (ADA_CONTROL)                       ; 1881
                                            04  A7  D4 003EF          CLRL    4(ADA_CONTROL)                          ;
                                57              DD 003F2          PUSHL   ADA_CONTROL                             ;
                    00000000G   00              01  FB 003F4          CALLS   #1, ADA$DBGEXT                          ;
                                07              50  E8 003FB          BLBS    R0, 61$                                 ;
                    00000000G   00              00  FB 003FE          CALLS   #0, LIB$SIGNAL                          ;
                                07          04  A7  E8 00405 61$:     BLBS    4(ADA_CONTROL), 62$                     ;
                    00000000G   00              00  FB 00409          CALLS   #0, LIB$SIGNAL                          ;
                72              59              03  E1 00410 62$:     BBC     #3, QUALIFIERS, 70$                     ; 1882
                                67              0F  B0 00414          MOVW    #15, (ADA_CONTROL)                      ; 1884
                                            04  A7  D4 00417          CLRL    4(ADA_CONTROL)                          ;
                                57              DD 0041A          PUSHL   ADA_CONTROL                             ;
                    00000000G   00              01  FB 0041C          CALLS   #1, ADA$DBGEXT                          ;
                                07              50  E8 00423          BLBS    R0, 63$                                 ;
                    00000000G   00              00  FB 00426          CALLS   #0, LIB$SIGNAL                          ;
                                0D          04  A7  E8 0042D 63$:     BLBS    4(ADA_CONTROL), 64$                     ;
                                02          04  A7  D1 00431          CMPL    4(ADA_CONTROL), #2                      ;
                                                07  13 00435          BEQL    64$                                     ;
                    00000000G   00              00  FB 00437          CALLS   #0, LIB$SIGNAL                          ;
                                02          04  A7  D1 0043E 64$:     CMPL    4(ADA_CONTROL), #2                      ;
                                                31  12 00442          BNEQ    68$                                     ;
             0F 00000000G   00                  05  E0 00444          BBS     #5, DBG$RUNFRAME+73, 65$                ;
                            08 00000000G        00  E8 0044C          BLBS    DBG$RUNFRAME+72, 65$                    ;
             05 00000000G   00                  04  E1 00453          BBC     #4, DBG$RUNFRAME+73, 66$                ;
                            50                  02  D0 0045B 65$:     MOVL    #2, EXC_TYPE                            ;
                                                03  11 0045E          BRB     67$                                     ;
                            50                  01  D0 00460 66$:     MOVL    #1, EXC_TYPE                            ;
                                  0401          8F  BB 00463 67$:     PUSHR   #^M<R0,R10>                             ;
                                00000000G       00  DD 00467          PUSHL   DBG$RUNFRAME+56                         ;
                                00000000G       00  DD 0046D          PUSHL   DBG$RUNFRAME+64                         ;
                                                0A  11 00473          BRB     69$                                     ;
                                                5A  DD 00475 68$:     PUSHL   CALLS_VALUE                             ;
                                                01  DD 00477          PUSHL   #1                                      ;
                                            5C  A7  DD 00479          PUSHL   92(ADA_CONTROL)                         ;
                                            64  A7  DD 0047C          PUSHL   100(ADA_CONTROL)                        ;
                        00000000G   00              04  FB 0047F 69$:     CALLS   #4, DBG$TRACEBACK                   ;
                                56          08  A2  9E 00486 70$:     MOVAB   8(R2), LINK                             ; 1885
                                                FF57  31 0048A          BRW     59$                                     ; 1876
          51                59                01      02  EF 0048D 71$:     EXTZV   #2, #1, QUALIFIERS, R1          ; 1890
                                                51      51  D2 00492          MCOML   R1, R1                          ;
                                                50      51  CA 00495          BICL2   R1, R0                          ;
          52                59                01      00  EF 00498          EXTZV   #0, #1, QUALIFIERS, R2          ;
                                                52      52  D2 0049D          MCOML   R2, R2                          ;
                                                50      52  CA 004A0          BICL2   R2, R0                          ;
                                                50      50  D2 004A3          MCOML   R0, R0                          ;
                                                01      50  D1 004A6          CMPL    R0, #1                          ;
                                                        01  13 004A9          BEQL    72$                             ;
                                                            04 004AB          RET                                     ;
          28                00                6E      00  2C 004AC 72$:     MOVC5   #0, (SP), #0, #40, (ADA_CONTROL)    ; 1892
                                                        67      004B1         ;
    02    A7                0C                00      31  F0 004B2          INSV    #49, #0, #12, 2(ADA_CONTROL)    ;
                                      20    A7      0000V   CF  9E 004B8          MOVAB   DBGEXT$PRINT_ROUTINE, 32(ADA_CONTROL) ;
                                      18    A7              07  8A 004BE          BICB2   #7, 24(ADA_CONTROL)             ;
                                                67          04  B0 004C2          MOVW    #4, (ADA_CONTROL)               ; 1893
                                                        04  A7  D4 004C5          CLRL    4(ADA_CONTROL)                  ;
                                                57              DD 004C8          PUSHL   ADA_CONTROL                     ;
                                00000000G       00              01  FB 004CA          CALLS   #1, ADA$DBGEXT          ;
```

```
                     07              50  E8 004D1          BLBS    R0, 73$                                 ;
                00000000G  00        00  FB 004D4          CALLS   #0, LIB$SIGNAL                          ;
                     07         04   A7  E8 004DB 73$:     BLBS    4(ADA_CONTROL), 74$                     ;
                00000000G  00        00  FB 004DF          CALLS   #0, LIB$SIGNAL                          ;
             72         59           03  E1 004E6 74$:     BBC     #3, QUALIFIERS, 82$                     ; 1894
                     67              0F  B0 004EA          MOVW    #15, (ADA_CONTROL)                      ; 1896
                              04     A7  D4 004ED          CLRL    4(ADA_CONTROL)                          ;
                     57              DD 004F0              PUSHL   ADA_CONTROL                             ;
                00000000G  00        01  FB 004F2          CALLS   #1, ADA$DBGEXT                          ;
                     07              50  E8 004F9          BLBS    R0, 75$                                 ;
                00000000G  00        00  FB 004FC          CALLS   #0, LIB$SIGNAL                          ;
                     0D         04   A7  E8 00503 75$:     BLBS    4(ADA_CONTROL), 76$                     ;
                     02         04   A7  D1 00507          CMPL    4(ADA_CONTROL), #2                      ;
                                     07  13 0050B          BEQL    76$                                     ;
                00000000G  00        00  FB 0050D          CALLS   #0, LIB$SIGNAL                          ;
                     02         04   A7  D1 00514 76$:     CMPL    4(ADA_CONTROL), #2                      ;
                                     31  12 00518          BNEQ    80$                                     ;
             0F 00000000G  00        05  E0 0051A          BBS     #5, DBG$RUNFRAME+73, 77$                ;
                     08 00000000G    00  E8 00522          BLBS    DBG$RUNFRAME+72, 77$                    ;
             05 00000000G  00        04  E1 00529          BBC     #4, DBG$RUNFRAME+73, 78$                ;
                     50              02  D0 00531 77$:     MOVL    #2, EXC_TYPE                            ;
                                     03  11 00534          BRB     79$                                     ;
                     50              01  D0 00536 78$:     MOVL    #1, EXC_TYPE                            ;
                           0401      8F  BB 00539 79$:     PUSHR   #^M<R0,R10>                             ;
                        00000000G    00  DD 00539          PUSHL   DBG$RUNFRAME+56                         ;
                        00000000G    00  DD 00543          PUSHL   DBG$RUNFRAME+64                         ;
                                     0A  11 00549          BRB     81$                                     ;
                                     5A  DD 0054B 80$:     PUSHL   CALLS_VALUE                             ;
                                     01  DD 0054D          PUSHL   #1                                      ;
                              5C     A7  DD 0054F          PUSHL   92(ADA_CONTROL)                         ;
                              64     A7  DD 00552          PUSHL   100(ADA_CONTROL)                        ;
                00000000G  00        04  FB 00555 81$:     CALLS   #4, DBG$TRACEBACK                       ;
                                         04 0055C 82$:     RET                                             ; 1908

; Routine Size:  1373 bytes,    Routine Base:  DBG$CODE + 0015

;  710          1909  1
```

```
;  712     1910  1 %SBTTL 'DBG$NPARSE_SET_TASK'
;  713     1911  1 GLOBAL ROUTINE DBG$NPARSE_SET_TASK ( INPUT_DESC : REF BLOCK [ , BYTE ],
;  714     1912  1     VERB_NODE : REF DBG$VERB_NODE ) : NOVALUE =
;  715     1913  1 !
;  716     1914  1 ! FUNCTION
;  717     1915  1 !       This routine parses the SET TASK command.  It accepts a command line
;  718     1916  1 !       string descriptor as input and produces a Verb Node for the parsed
;  719     1917  1 !       string as output.  The Verb Node and its attached Adverb Nodes and
;  720     1918  1 !       Noun Nodes, as built by this routine, later serve as input to the
;  721     1919  1 !       DBG$NEXECUTE_SET_TASK routine which actually executes the command.
;  722     1920  1 !
;  723     1921  1 ! INPUTS
;  724     1922  1 !       INPUT_DESC - A string descriptor pointing to the input line being
;  725     1923  1 !                    parsed.  The descriptor is assumed to be pointing to the
;  726     1924  1 !                    first character after the SET TASK keywords.
;  727     1925  1 !
;  728     1926  1 !       VERB_NODE - A pointer to the Verb Node to be built up for the command
;  729     1927  1 !                   being parsed.
;  730     1928  1 !
;  731     1929  1 ! OUTPUTS
;  732     1930  1 !       INPUT_DESC - The input string descriptor is updated to point to the
;  733     1931  1 !                    first character after the end of the command.  This normally
;  734     1932  1 !                    means that the input string is exhausted.
;  735     1933  1 !
;  736     1934  1 !       VERB_NODE - The passed-in Verb Node is filled in so that it and its
;  737     1935  1 !                   attached Adverb and Noun Nodes contain all information picked
;  738     1936  1 !                   up during the parse of the SET TASK command.
;  739     1937  1 !
;  740     1938  1
;  741     1939  2     BEGIN
;  742     1940  2
;  743     1941  2     LOCAL
;  744     1942  2         ADVERB_NODE : REF DBG$ADVERB_NODE,
;  745     1943  2         NOUN_NODE : REF DBG$NOUN_NODE,
;  746     1944  2         LINK,                                   ! Link field to next adverb or noun node.
;  747     1945  2         PRIORITY;                               ! Temporary for storing priority.
;  748     1946  2
;  749     1947  2
;  750     1948  2     ! The field VERB_NODE [DBG$B_VERB_COMPOSITE] has already been set = SET_TASK
;  751     1949  2     ! to indicate that the command was SET TASK in routine DBG$NPARSE_SET.
;  752     1950  2
;  753     1951  2     link = verb_node [dbg$l_verb_adverb_ptr];
;  754     1952  2
;  755     1953  2     !
;  756     1954  2     ! Scan for command qualifiers.  If found, construct adverb nodes.
;  757     1955  2     !
;  758     1956  2     WHILE dbg$nmatch (.input_desc, dbg$cs_slash, 1) DO
;  759     1957  3         BEGIN
;  760     1958  3         !
;  761     1959  3         ! Case on the qualifier.
;  762     1960  3         !
;  763     1961  3         SELECTONE TRUE OF
;  764     1962  3             SET
;  765     1963  3             !
;  766     1964  3             ! SET TASK /ACTIVE.  Construct an Adverb Node and link it in.
;  767     1965  3             !
;  768     1966  3             [ DBG$NMATCH( .INPUT_DESC, DBG$CS_ACTIVE, 2 ) ]:
```

```
;  769          1967  4                  BEGIN
;  770          1968  4                  ADVERB_NODE = DBG$GET_TEMPMEM (DBG$K_ADVERB_NODE_SIZE);
;  771          1969  4                  .LINK = .ADVERB_NODE;
;  772          1970  4                  LINK = ADVERB_NODE [DBG$L_ADVERB_LINK];
;  773          1971  4                  ADVERB_NODE [DBG$B_ADVERB_LITERA[] = TASK_ACTIVE;
;  774          1972  3                  END;
;  775          1973  3
;  776          1974  3              !
;  777          1975  3              ! SET TASK /ALL.  Construct an Adverb Node and link it in.
;  778          1976  3              !
;  779          1977  3              [ DBG$NMATCH( .INPUT_DESC, DBG$CS_ALL, 2 ) ]:
;  780          1978  4                  BEGIN
;  781          1979  4                  ADVERB_NODE = DBG$GET_TEMPMEM (DBG$K_ADVERB_NODE_SIZE);
;  782          1980  4                  .LINK = .ADVERB_NODE;
;  783          1981  4                  LINK = ADVERB_NODE [DBG$L_ADVERB_LINK];
;  784          1982  4                  ADVERB_NODE [DBG$B_ADVERB_LITERA[] = TASK_ALL;
;  785          1983  3                  END;
;  786          1984  3
;  787          1985  3              !
;  788          1986  3              ! SET TASK /VISIBLE.  Construct an Adverb Node and link it in.
;  789          1987  3              !
;  790          1988  3              [ DBG$NMATCH( .INPUT_DESC, DBG$CS_VISIBLE, 1 ) ]:
;  791          1989  4                  BEGIN
;  792          1990  4                  ADVERB_NODE = DBG$GET_TEMPMEM (DBG$K_ADVERB_NODE_SIZE);
;  793          1991  4                  .LINK = .ADVERB_NODE;
;  794          1992  4                  LINK = ADVERB_NODE [DBG$L_ADVERB_LINK];
;  795          1993  4                  ADVERB_NODE [DBG$B_ADVERB_LITERA[] = TASK_VISIBLE;
;  796          1994  3                  END;
;  797          1995  3
;  798          1996  3              !
;  799          1997  3              ! SET TASK /PRIORITY=(n).  Construct an Adverb Node and link it in.
;  800          1998  3              !
;  801          1999  3              [ DBG$NMATCH( .INPUT_DESC, DBG$CS_PRIORITY, 1 ) ]:
;  802          2000  4                  BEGIN
;  803          2001  4                  ADVERB_NODE = DBG$GET_TEMPMEM (DBG$K_ADVERB_NODE_SIZE);
;  804          2002  4                  .LINK = .ADVERB_NODE;
;  805          2003  4                  LINK = ADVERB_NODE [DBG$L_ADVERB_LINK];
;  806          2004  4                  ADVERB_NODE [DBG$B_ADVERB_LITERA[] = TASK_PRIORITY;
;  807          2005  4
;  808          2006  4                  IF DBG$NMATCH( .INPUT_DESC, DBG$CS_COLON, 1 )
;  809          2007  4                      OR DBG$NMATCH( .INPUT_DESC, DBG$CS_EQUAL, 1 )
;  810          2008  4                  THEN
;  811          2009  4                      IF DBG$NMATCH( .INPUT_DESC, dbg$cs_left_paren, 1 )
;  812          2010  4                      THEN
;  813          2011  5                          BEGIN                                   %((FIX THIS CAN'T HAVE MULTIPLE PRIO
;  814          2012  5                          DO
;  815          2013  6                              BEGIN
;  816          2014  6                              DBG$NSAVE_DECIMAL_INTEGER(
;  817          2015  6                                  .INPUT_DESC,                    ! read input value
;  818          2016  6                                  PRIORITY);
;  819          2017  6                              IF .PRIORITY GTRU 31                ! %((need priority limit -tbs))%
;  820          2018  6                              THEN
;  821          2019  6                                  SIGNAL (DBG$_BITRANGE );        ! %((NEED A BETTER MESSAGE -tbs))%
;  822          2020  6                              (ADVERB_NODE [DBG$L_ADVERB_VALUE]) <.PRIORITY, 1, 0> = 1;   ! set corresponding
;  823          2021  6                              END
;  824          2022  5                          WHILE DBG$NMATCH( .INPUT_DESC, dbg$cs_comma, 1 );
;  825          2023  5                          IF NOT DBG$NMATCH( .INPUT_DESC, dbg$cs_right_paren, 1 )
```

```
;  826     2024  5                    THEN
;  827     2025  5                        SIGNAL (dbg$_UNMTCHPARN);   ! Unmatched left parenthesis found.
;  828     2026  5                    END
;  829     2027  4                ELSE
;  830     2028  5                    BEGIN
;  831     2029  5                    DBG$NSAVE_DECIMAL_INTEGER(
;  832     2030  5                            .INPUT_DESC,                        ! read input value
;  833     2031  5                            PRIORITY);
;  834     2032  5                    IF .PRIORITY GTRU 31                    ! %((need a limit -tbs))%
;  835     2033  5                    THEN
;  836     2034  5                        SIGNAL (DBG$_BITRANGE );            ! %((NEED A BETTER MESSAGE -tbs))%
;  837     2035  5                    (ADVERB_NODE [DBG$L_ADVERB_VALUE]) <.PRIORITY, 1, 0> = 1;       ! set corresponding
;  838     2036  5                    END
;  839     2037  4            ELSE
;  840     2038  4                SIGNAL (DBG$_NEEDMORE);
;  841     2039  4
;  842     2040  3                END;
;  843     2041  3
;  844     2042  3            !
;  845     2043  3            ! SET TASK /RESTORE.  Construct an Adverb Node and link it in.
;  846     2044  3            !
;  847     2045  3            [ DBG$NMATCH( .INPUT_DESC, DBG$CS_RESTORE, 3 ) ]:
;  848     2046  4                BEGIN
;  849     2047  4                ADVERB_NODE = DBG$GET_TEMPMEM (DBG$K_ADVERB_NODE_SIZE);
;  850     2048  4                .LINK = .ADVERB_NODE;
;  851     2049  4                LINK = ADVERB_NODE [DBG$L_ADVERB_LINK];
;  852     2050  4                ADVERB_NODE [DBG$B_ADVERB_LITERA[] = TASK_RESTORE;
;  853     2051  3                END;
;  854     2052  3
;  855     2053  3            !
;  856     2054  3            ! SET TASK /RELEASE  or  SET TASK /NOHOLD.  Construct an Adverb Node and link it in.
;  857     2055  3            !
;  858     2056  3            [ DBG$NMATCH( .INPUT_DESC, DBG$CS_RELEASE, 3 ) ,
;  859     2057  3              DBG$NMATCH( .INPUT_DESC, DBG$CS_NOHOLD, 3 ) ]:
;  860     2058  4                BEGIN
;  861     2059  4                ADVERB_NODE = DBG$GET_TEMPMEM (DBG$K_ADVERB_NODE_SIZE);
;  862     2060  4                .LINK = .ADVERB_NODE;
;  863     2061  4                LINK = ADVERB_NODE [DBG$L_ADVERB_LINK];
;  864     2062  4                ADVERB_NODE [DBG$B_ADVERB_LITERA[] = TASK_RELEASE;
;  865     2063  3                END;
;  866     2064  3
;  867     2065  3            !
;  868     2066  3            ! SET TASK /HOLD.  Construct an Adverb Node and link it in.
;  869     2067  3            !
;  870     2068  3            [ DBG$NMATCH( .INPUT_DESC, DBG$CS_HOLD, 1 ) ]:
;  871     2069  4                BEGIN
;  872     2070  4                ADVERB_NODE = DBG$GET_TEMPMEM (DBG$K_ADVERB_NODE_SIZE);
;  873     2071  4                .LINK = .ADVERB_NODE;
;  874     2072  4                LINK = ADVERB_NODE [DBG$L_ADVERB_LINK];
;  875     2073  4                ADVERB_NODE [DBG$B_ADVERB_LITERA[] = TASK_HOLD;
;  876     2074  3                END;
;  877     2075  3
;  878     2076  3            !
;  879     2077  3            ! SET TASK /TERMINATE.  Construct an Adverb Node and link it in.
;  880     2078  3            !
;  881     2079  3            [ DBG$NMATCH( .INPUT_DESC, DBG$CS_TERMINATE, 1 ) ]:
;  882     2080  4                BEGIN
```

```
;  883          2081  4                  ADVERB_NODE = DBG$GET_TEMPMEM (DBG$K_ADVERB_NODE_SIZE);
;  884          2082  4                  .LINK = .ADVERB_NODE;
;  885          2083  4                  LINK = ADVERB_NODE [DBG$L_ADVERB_LINK];
;  886          2084  4                  ADVERB_NODE [DBG$B_ADVERB_LITERA[] = TASK_TERMINATE;
;  887          2085  3                  END;
;  888          2086  3
;  889          2087  3              !
;  890          2088  3              ! Any other condition is an error.
;  891          2089  3              !
;  892          2090  3              [ OTHERWISE ]:
;  893          2091  3                  DBG$SYNTAX_ERROR(.INPUT_DESC);
;  894          2092  3
;  895          2093  3              TES;
;  896          2094  3
;  897          2095  2          END;                ! of WHILE /qualifier
;  898          2096  2
;  899          2097  2      .LINK = 0;              ! End of adverb node chain.
;  900          2098  2
;  901          2099  2      IF NOT DBG$NMATCH( .INPUT_DESC, DBG$CS_CR, 1 )      ! If more input exists then
;  902          2100  2      THEN                                                ! try to parse a task list.
;  903          2101  3          BEGIN
;  904          2102  3          LINK = VERB_NODE [DBG$L_VERB_OBJECT_PTR];
;  905          2103  3          DO
;  906          2104  3              !+
;  907          2105  3              ! Parse tasks in a task list and build noun nodes and value
;  908          2106  3              ! descriptors until end of list or an error is encountered.
;  909          2107  3              !-
;  910          2108  4              BEGIN
;  911          2109  4              NOUN_NODE = DBG$GET_TEMPMEM (DBG$K_NOUN_NODE_SIZE); ! %((NEED LONG NOUN ?-tbs))%
;  912          2110  4              .LINK = .NOUN_NODE;
;  913          2111  4              LINK = NOUN_NODE [DBG$L_NOUN_LINK];
;  914          2112  4
;  915          2113  4              DBG$NPARSE_EXPRESSION (
;  916          2114  4                  .INPUT_DESC,                            ! rest of command
;  917          2115  4                  .DBG$GB_RADIX[DBG$B_RADIX_INPUT],       ! default input radix
;  918          2116  4                  NOUN_NODE [DBG$L_NOUN_VALUE],           ! where to store ptr to value desc
;  919          2117  4                  TOKEN$K_TERM_COMMA );                   ! task terminator token
;  920          2118  4
;  921          2119  4              !   .MESSAGE_VECT);                        %((REMOVE MESSAGE VEC FROM ROUTINE -tbs))%
;  922          2120  4
;  923          2121  4              END
;  924          2122  3          WHILE DBG$NMATCH( .INPUT_DESC, DBG$CS_COMMA, 1 );
;  925          2123  3
;  926          2124  3          IF NOT DBG$NMATCH( .INPUT_DESC, DBG$CS_CR, 1 )  ! If more input exists then
;  927          2125  3          THEN                                            ! we have an error.
;  928          2126  3              DBG$SYNTAX_ERROR(.INPUT_DESC);              ! Signal the error.
;  929          2127  2          END;
;  930          2128  2
;  931          2129  2
;  932          2130  2      RETURN 0;                   ! %((0?-tbs))%
;  933          2131  2
;  934          2132  1      END;                        ! end of DBG$NPARSE_SET_TASK
```

```
                          OFFC 00000          .ENTRY  DBG$NPARSE_SET_TASK, Save R2,R3,R4,R5,R6,-      ; 1911
                                                      R7,R8,R9,R10,R11
                5B 00000000G   00  9E 00002          MOVAB   DBG$SYNTAX_ERROR, R11                   ;
                5A 00000000G   00  9E 00009          MOVAB   DBG$NSAVE_DECIMAL_INTEGER, R10          ;
                59 00000000G   00  9E 00010          MOVAB   LIB$SIGNAL, R9                          ;
                58 00000000G   00  9E 00017          MOVAB   DBG$GET_TEMPMEM, R8                     ;
                57 00000000'   EF  9E 0001E          MOVAB   DBG$CS_COMMA, R7                        ;
                56 00000000G   00  9E 00025          MOVAB   DBG$NMATCH, R6                          ;
                5E                 04  C2 0002C      SUBL2   #4, SP                                  ;
          54    08             AC  04  C1 0002F      ADDL3   #4, VERB_NODE, LINK                     ; 1951
                53        04   AC  00  D0 00034      MOVL    INPUT_DESC, R3                          ; 1956
                               01  DD 00038 1$:      PUSHL   #1                                      ;
                          06   A7  9F 0003A          PUSHAB  DBG$CS_SLASH                            ;
                               53  DD 0003D          PUSHL   R3                                      ;
                66             03  FB 0003F          CALLS   #3, DBG$NMATCH                          ;
                03             50  E8 00042          BLBS    R0, 2$                                  ;
                             01BE  31 00045          BRW     26$                                     ;
                               02  DD 00048 2$:      PUSHL   #2                                      ; 1966
                        FF70   C7  9F 0004A          PUSHAB  DBG$CS_ACTIVE                           ;
                               53  DD 0004E          PUSHL   R3                                      ;
                66             03  FB 00050          CALLS   #3, DBG$NMATCH                          ;
                01             50  D1 00053          CMPL    R0, #1                                  ;
                               14  12 00056          BNEQ    3$                                      ;
                               03  DD 00058          PUSHL   #3                                      ; 1968
                68             01  FB 0005A          CALLS   #1, DBG$GET_TEMPMEM                     ;
                52             50  D0 0005D          MOVL    R0, ADVERB_NODE                         ;
                64             52  D0 00060          MOVL    ADVERB_NODE, (LINK)                     ; 1969
                54        08   A2  9E 00063          MOVAB   8(R2), LINK                             ; 1970
                62             01  90 00067          MOVB    #1, (ADVERB_NODE)                       ; 1971
                               CC  11 0006A          BRB     1$                                      ; 1961
                               02  DD 0006C 3$:      PUSHL   #2                                      ; 1977
                        FF77   C7  9F 0006E          PUSHAB  DBG$CS_ALL                              ;
                               53  DD 00072          PUSHL   R3                                      ;
                66             03  FB 00074          CALLS   #3, DBG$NMATCH                          ;
                01             50  D1 00077          CMPL    R0, #1                                  ;
                               14  12 0007A          BNEQ    4$                                      ;
                               03  DD 0007C          PUSHL   #3                                      ; 1979
                68             01  FB 0007E          CALLS   #1, DBG$GET_TEMPMEM                     ;
                52             50  D0 00081          MOVL    R0, ADVERB_NODE                         ;
                64             52  D0 00084          MOVL    ADVERB_NODE, (LINK)                     ; 1980
                54        08   A2  9E 00087          MOVAB   8(R2), LINK                             ; 1981
                62             02  90 0008B          MOVB    #2, (ADVERB_NODE)                       ; 1982
                               A8  11 0008E          BRB     1$                                      ; 1961
                               01  DD 00090 4$:      PUSHL   #1                                      ; 1988
                        CF     A7  9F 00092          PUSHAB  DBG$CS_VISIBLE                          ;
                               53  DD 00095          PUSHL   R3                                      ;
                66             03  FB 00097          CALLS   #3, DBG$NMATCH                          ;
                01             50  D1 0009A          CMPL    R0, #1                                  ;
                               14  12 0009D          BNEQ    6$                                      ;
                               03  DD 0009F          PUSHL   #3                                      ; 1990
                68             01  FB 000A1          CALLS   #1, DBG$GET_TEMPMEM                     ;
                52             50  D0 000A4          MOVL    R0, ADVERB_NODE                         ;
                64             52  D0 000A7          MOVL    ADVERB_NODE, (LINK)                     ; 1991
                54        08   A2  9E 000AA          MOVAB   8(R2), LINK                             ; 1992
                62             0D  90 000AE          MOVB    #13, (ADVERB_NODE)                      ; 1993
                               85  11 000B1 5$:      BRB     1$                                      ; 1961
                               01  DD 000B3 6$:      PUSHL   #1                                      ; 1999
```

```
                    9B   A7  9F 000B5          PUSHAB  DBG$CS_PRIORITY                          ;
                         53  DD 000B8          PUSHL   R3                                      ;
                66       03  FB 000BA          CALLS   #3, DBG$NMATCH                          ;
                01       50  D1 000BD          CMPL    R0, #1                                  ;
                         03  13 000C0          BEQL    7$                                      ;
                      009B  31 000C2          BRW     16$                                     ;
                         03  DD 000C5 7$:      PUSHL   #3                                      ; 2001
                68       01  FB 000C7          CALLS   #1, DBG$GET_TEMPMEM                     ;
                52       50  D0 000CA          MOVL    R0, ADVERB_NODE                         ;
                64       52  D0 000CD          MOVL    ADVERB_NODE, (LINK)                     ; 2002
                54    08 A2  9E 000D0          MOVAB   8(R2), LINK                             ; 2003
                62       07  90 000D4          MOVB    #7, (ADVERB_NODE)                       ; 2004
                         01  DD 000D7          PUSHL   #1                                      ; 2006
                    FE   A7  9F 000D9          PUSHAB  DBG$CS_COLON                            ;
                         53  DD 000DC          PUSHL   R3                                      ;
                66       03  FB 000DE          CALLS   #3, DBG$NMATCH                          ;
                0D       50  E8 000E1          BLBS    R0, 8$                                  ;
                         01  DD 000E4          PUSHL   #1                                      ; 2007
                    04   A7  9F 000E6          PUSHAB  DBG$CS_EQUAL                            ;
                         53  DD 000E9          PUSHL   R3                                      ;
                66       03  FB 000EB          CALLS   #3, DBG$NMATCH                          ;
                64       50  E9 000EE          BLBC    R0, 14$                                 ;
                         01  DD 000F1 8$:      PUSHL   #1                                      ; 2009
                    FA   A7  9F 000F3          PUSHAB  DBG$CS_LEFT_PAREN                       ;
                         53  DD 000F6          PUSHL   R3                                      ;
                66       03  FB 000F8          CALLS   #3, DBG$NMATCH                          ;
                3B       50  E9 000FB          BLBC    R0, 12$                                 ;
                  4008   8F  BB 000FE 9$:      PUSHR   #^M<R3,SP>                              ; 2015
                6A       02  FB 00102          CALLS   #2, DBG$NSAVE_DECIMAL_INTEGER           ;
                1F       6E  D1 00105          CMPL    PRIORITY, #31                           ; 2017
                         09  1B 00108          BLEQU   10$                                     ;
              00028248   8F  DD 0010A          PUSHL   #164424                                 ; 2019
                69       01  FB 00110          CALLS   #1, LIB$SIGNAL                          ;
          00    04   A2      6E  E2 00113 10$:  BBSS    PRIORITY, 4(ADVERB_NODE), 11$           ; 2020
                         01  DD 00118 11$:      PUSHL   #1                                      ; 2022
                  0088   8F  BB 0011A          PUSHR   #^M<R3,R7>                              ;
                66       03  FB 0011E          CALLS   #3, DBG$NMATCH                          ;
                DA       50  E8 00121          BLBS    R0, 9$                                  ;
                         01  DD 00124          PUSHL   #1                                      ; 2023
                    FC   A7  9F 00126          PUSHAB  DBG$CS_RIGHT_PAREN                      ;
                         53  DD 00129          PUSHL   R3                                      ;
                66       03  FB 0012B          CALLS   #3, DBG$NMATCH                          ;
                80       50  E8 0012E          BLBS    R0, 5$                                  ;
              000287D0   8F  DD 00131          PUSHL   #165840                                 ; 2025
                         22  11 00137          BRB     15$                                     ;
                  4008   8F  BB 00139 12$:      PUSHR   #^M<R3,SP>                              ; 2030
                6A       02  FB 0013D          CALLS   #2, DBG$NSAVE_DECIMAL_INTEGER           ;
                1F       6E  D1 00140          CMPL    PRIORITY, #31                           ; 2032
                         09  1B 00143          BLEQU   13$                                     ;
              00028248   8F  DD 00145          PUSHL   #164424                                 ; 2034
                69       01  FB 0014B          CALLS   #1, LIB$SIGNAL                          ;
          63    04   A2      6E  E2 0014E 13$:  BBSS    PRIORITY, 4(ADVERB_NODE), 19$           ; 2035
                         61  11 00153          BRB     19$                                     ; 2009
              000280D0   8F  DD 00155 14$:      PUSHL   #164048                                 ; 2038
                69       01  FB 0015B 15$:      CALLS   #1, LIB$SIGNAL                          ;
                         79  11 0015E          BRB     21$                                     ; 1961
                         03  DD 00160 16$:      PUSHL   #3                                      ; 2045
```

```
                    AC  A7  9F 00162          PUSHAB  DBG$CS_RESTORE                  ;
                        53  DD 00165          PUSHL   R3                              ;
                66      03  FB 00167          CALLS   #3, DBG$NMATCH                  ;
                01      50  D1 0016A          CMPL    R0, #1                          ;
                        14  12 0016D          BNEQ    17$                             ;
                        03  DD 0016F          PUSHL   #3                              ; 2047
                68      01  FB 00171          CALLS   #1, DBG$GET_TEMPMEM             ;
                52      50  D0 00174          MOVL    R0, ADVERB_NODE                 ;
                64      52  D0 00177          MOVL    ADVERB_NODE, (LINK)             ; 2048
                54  08  A2  9E 0017A          MOVAB   8(R2), LINK                     ; 2049
                62      09  90 0017E          MOVB    #9, (ADVERB_NODE)               ; 2050
                        79  11 00181          BRB     25$                             ; 1961
                        03  DD 00183  17$:     PUSHL   #3                              ; 2056
                    A4  A7  9F 00185          PUSHAB  DBG$CS_RELEASE                  ;
                        53  DD 00188          PUSHL   R3                              ;
                66      03  FB 0018A          CALLS   #3, DBG$NMATCH                  ;
                55      50  D0 0018D          MOVL    R0, R5                          ;
                01      55  D1 00190          CMPL    R5, #1                          ;
                        0F  13 00193          BEQL    18$                             ;
                        03  DD 00195          PUSHL   #3                              ; 2057
                    94  A7  9F 00197          PUSHAB  DBG$CS_NOHOLD                   ;
                        53  DD 0019A          PUSHL   R3                              ;
                66      03  FB 0019C          CALLS   #3, DBG$NMATCH                  ;
                01      50  D1 0019F          CMPL    R0, #1                          ;
                        14  12 001A2          BNEQ    20$                             ;
                        03  DD 001A4  18$:     PUSHL   #3                              ; 2059
                68      01  FB 001A6          CALLS   #1, DBG$GET_TEMPMEM             ;
                52      50  D0 001A9          MOVL    R0, ADVERB_NODE                 ;
                64      52  D0 001AC          MOVL    ADVERB_NODE, (LINK)             ; 2060
                54  08  A2  9E 001AF          MOVAB   8(R2), LINK                     ; 2061
                62      08  90 001B3          MOVB    #8, (ADVERB_NODE)               ; 2062
                        4B  11 001B6  19$:     BRB     25$                             ; 1961
                        01  DD 001B8  20$:     PUSHL   #1                              ; 2068
                    8F  A7  9F 001BA          PUSHAB  DBG$CS_HOLD                     ;
                        53  DD 001BD          PUSHL   R3                              ;
                66      03  FB 001BF          CALLS   #3, DBG$NMATCH                  ;
                01      50  D1 001C2          CMPL    R0, #1                          ;
                        14  12 001C5          BNEQ    22$                             ;
                        03  DD 001C7          PUSHL   #3                              ; 2070
                68      01  FB 001C9          CALLS   #1, DBG$GET_TEMPMEM             ;
                52      50  D0 001CC          MOVL    R0, ADVERB_NODE                 ;
                64      52  D0 001CF          MOVL    ADVERB_NODE, (LINK)             ; 2071
                54  08  A2  9E 001D2          MOVAB   8(R2), LINK                     ; 2072
                62      06  90 001D6          MOVB    #6, (ADVERB_NODE)               ; 2073
                        28  11 001D9  21$:     BRB     25$                             ; 1961
                        01  DD 001DB  22$:     PUSHL   #1                              ; 2079
                    C5  A7  9F 001DD          PUSHAB  DBG$CS_TERMINATE                ;
                        53  DD 001E0          PUSHL   R3                              ;
                66      03  FB 001E2          CALLS   #3, DBG$NMATCH                  ;
                01      50  D1 001E5          CMPL    R0, #1                          ;
                        14  12 001E8          BNEQ    24$                             ;
                        03  DD 001EA          PUSHL   #3                              ; 2081
                68      01  FB 001EC          CALLS   #1, DBG$GET_TEMPMEM             ;
                52      50  D0 001EF          MOVL    R0, ADVERB_NODE                 ;
                64      52  D0 001F2          MOVL    ADVERB_NODE, (LINK)             ; 2082
                54  08  A2  9E 001F5          MOVAB   8(R2), LINK                     ; 2083
                62      0C  90 001F9          MOVB    #12, (ADVERB_NODE)              ; 2084
```

```
                                                05  11 001FC 23$:     BRB     25$                                     ; 1961
                                                53  DD 001FE 24$:     PUSHL   R3                                      ; 2091
                                            6B      01  FB 00200       CALLS   #1, DBG$SYNTAX_ERROR                    ;
                                              FE32  31 00203 25$:     BRW     1$                                      ; 1956
                                                64  D4 00206 26$:     CLRL    (LINK)                                  ; 2097
                                                01  DD 00208         PUSHL   #1                                      ; 2099
                                          02    A7  9F 0020A         PUSHAB  DBG$CS_CR                               ;
                                                53  DD 0020D         PUSHL   R3                                      ;
                                            66      03  FB 0020F       CALLS   #3, DBG$NMATCH                          ;
                                            46      50  E8 00212       BLBS    R0, 28$                                 ;
                                 54     08  AC      08  C1 00215       ADDL3   #8, VERB_NODE, LINK                     ; 2102
                                                04  DD 0021A 27$:     PUSHL   #4                                      ; 2109
                                            68      01  FB 0021C       CALLS   #1, DBG$GET_TEMPMEM                     ;
                                            52      50  D0 0021F       MOVL    R0, NOUN_NODE                           ;
                                            64      52  D0 00222       MOVL    NOUN_NODE, (LINK)                       ; 2110
                                            54  08  A2  9E 00225       MOVAB   8(R2), LINK                             ; 2111
                                                01  DD 00229         PUSHL   #1                                      ; 2116
                                                52  DD 0022B         PUSHL   NOUN_NODE                               ;
                                            7E  00000000G   00  9A 0022D       MOVZBL  DBG$GB_RADIX, -(SP)             ;
                                                53  DD 00234         PUSHL   R3                                      ;
                                00000000G       00      04  FB 00236       CALLS   #4, DBG$NPARSE_EXPRESSION               ;
                                                01  DD 0023D         PUSHL   #1                                      ; 2122
                                          0088  8F  BB 0023F         PUSHR   #^M<R3,R7>                              ;
                                            66      03  FB 00243       CALLS   #3, DBG$NMATCH                          ;
                                            D1      50  E8 00246       BLBS    R0, 27$                                 ;
                                                01  DD 00249         PUSHL   #1                                      ; 2124
                                          02    A7  9F 0024B         PUSHAB  DBG$CS_CR                               ;
                                                53  DD 0024E         PUSHL   R3                                      ;
                                            66      03  FB 00250       CALLS   #3, DBG$NMATCH                          ;
                                            05      50  E8 00253       BLBS    R0, 28$                                 ;
                                                53  DD 00256         PUSHL   R3                                      ; 2126
                                            68      01  FB 00258       CALLS   #1, DBG$SYNTAX_ERROR                    ;
                                                    04  0025B 28$:     RET                                             ; 2132

; Routine Size:  604 bytes,    Routine Base:  DBG$CODE + 0572

;  935          2133  1
```

```
;  937          2134  1 %SBTTL 'DBG$NPARSE_SHOW_TASK'
;  938          2135  1 GLOBAL ROUTINE DBG$NPARSE_SHOW_TASK ( INPUT_DESC : REF BLOCK [ , BYTE ],
;  939          2136  1     VERB_NODE : REF DBG$VERB_NODE ) : NOVALUE =
;  940          2137  1 !
;  941          2138  1 ! FUNCTION
;  942          2139  1 !       This routine parses the SHOW TASK command.  It accepts a command line
;  943          2140  1 !       string descriptor as input and produces a Verb Node for the parsed
;  944          2141  1 !       string as output.  The Verb Node and its attached Adverb Nodes and
;  945          2142  1 !       Noun Nodes, as built by this routine, later serve as input to the
;  946          2143  1 !       DBG$NEXECUTE_SHOW_TASK routine which actually executes the command.
;  947          2144  1 !
;  948          2145  1 ! INPUTS
;  949          2146  1 !       INPUT_DESC - A string descriptor pointing to the input line being
;  950          2147  1 !                    parsed.  The descriptor is assumed to be pointing to the
;  951          2148  1 !                    first character after the SHOW TASK keywords.
;  952          2149  1 !
;  953          2150  1 !       VERB_NODE - A pointer to the Verb Node to be built up for the command
;  954          2151  1 !                   being parsed.
;  955          2152  1 !
;  956          2153  1 ! OUTPUTS
;  957          2154  1 !       INPUT_DESC - The input string descriptor is updated to poirt to the
;  958          2155  1 !                    first character after the end of the command.  This normally
;  959          2156  1 !                    means that the input string is exhausted.
;  960          2157  1 !
;  961          2158  1 !       VERB_NODE - The passed-in Verb Node is filled in so that it and its
;  962          2159  1 !                   attached Adverb and Noun Nodes contain all information picked
;  963          2160  1 !                   up during the parse of the SHOW TASK command.
;  964          2161  1 !
;  965          2162  1
;  966          2163  2     BEGIN
;  967          2164  2
;  968          2165  2     LOCAL
;  969          2166  2         ADVERB_NODE : REF DBG$ADVERB_NODE,
;  970          2167  2         NOUN_NODE : REF DBG$NOUN_NODE,
;  971          2168  2         LINK,                                   ! Link field to next adverb or noun node.
;  972          2169  2         PRIORITY;                               ! Temporary for storing priority.
;  973          2170  2
;  974          2171  2
;  975          2172  2     ! The field VERB_NODE [DBG$B_VERB_COMPOSITE] has already been set = SHOW_TASK
;  976          2173  2     ! to indicate that the command was SHOW TASK in routine DBG$NPARSE_SHOW.
;  977          2174  2
;  978          2175  2     link = verb_node [dbg$l_verb_adverb_ptr];
;  979          2176  2
;  980          2177  2     !
;  981          2178  2     ! Scan for command qualifiers.  If found, construct adverb nodes.
;  982          2179  2     !
;  983          2180  2     WHILE dbg$nmatch (.input_desc, dbg$cs_slash, 1) DO
;  984          2181  3         BEGIN
;  985          2182  3         !
;  986          2183  3         ! Case on the qualifier.
;  987          2184  3         !
;  988          2185  3         SELECTONE TRUE OF
;  989          2186  3             SET
;  990          2187  3             !
;  991          2188  3             ! SHOW TASK /ALL.  Construct an Adverb Node and link it in.
;  992          2189  3             !
;  993          2190  3             [ DBG$NMATCH( .INPUT_DESC, DBG$CS_ALL, 1 ) ]:
```

```
;  994          2191  4                BEGIN
;  995          2192  4                ADVERB_NODE = DBG$GET_TEMPMEM (DBG$K_ADVERB_NODE_SIZE);
;  996          2193  4                .LINK = .ADVERB_NODE;
;  997          2194  4                LINK = ADVERB_NODE [DBG$L_ADVERB_LINK];
;  998          2195  4                ADVERB_NODE [DBG$B_ADVERB_LITERA[] = TASK_ALL;
;  999          2196  3                END;
; 1000          2197  3
; 1001          2198  3            !
; 1002          2199  3            ! SHOW TASK /CALLS [ = n ].  Construct an Adverb Node and link it in.  Pickup the
; 1003          2200  3            ! call depth to display.  Assume -1 (very large number) if not given explicitly.
; 1004          2201  3            !
; 1005          2202  3            [ DBG$NMATCH( .INPUT_DESC, DBG$CS_CALLS, 1 ) ]:
; 1006          2203  4                BEGIN
; 1007          2204  4                ADVERB_NODE = DBG$GET_TEMPMEM (DBG$K_ADVERB_NODE_SIZE);
; 1008          2205  4                .LINK = .ADVERB_NODE;
; 1009          2206  4                LINK = ADVERB_NODE [DBG$L_ADVERB_LINK];
; 1010          2207  4                ADVERB_NODE [DBG$B_ADVERB_LITERA[] = TASK_CALLS;
; 1011          2208  4
; 1012          2209  4                IF DBG$NMATCH( .INPUT_DESC, DBG$CS_COLON, 1 )    ! did user give a call depth?
; 1013          2210  4                    OR DBG$NMATCH( .INPUT_DESC, DBG$CS_EQUAL, 1 )
; 1014          2211  4                THEN
; 1015          2212  4                    DBG$NSAVE_DECIMAL_INTEGER(                   ! this routine checks for errors
; 1016          2213  4                        .INPUT_DESC,                             ! read input value
; 1017          2214  4                        ADVERB_NODE [DBG$L_ADVERB_VALUE])        ! store in adverb node
; 1018          2215  4                ELSE
; 1019          2216  4                    ADVERB_NODE [DBG$L_ADVERB_VALUE] = -1;       ! use default value
; 1020          2217  3                END;
; 1021          2218  3
; 1022          2219  3            !
; 1023          2220  3            ! SHOW TASK /DEADLOCK.  Construct an Adverb Node and link it in.
; 1024          2221  3            !
; 1025          2222  3            [ DBG$NMATCH( .INPUT_DESC, DBG$CS_DEADLOCK, 1 ) ]:
; 1026          2223  4                BEGIN
; 1027          2224  4                ADVERB_NODE = DBG$GET_TEMPMEM (DBG$K_ADVERB_NODE_SIZE);
; 1028          2225  4                .LINK = .ADVERB_NODE;
; 1029          2226  4                LINK = ADVERB_NODE [DBG$L_ADVERB_LINK];
; 1030          2227  4                ADVERB_NODE [DBG$B_ADVERB_LITERA[] = TASK_DEADLOCK;
; 1031          2228  3                END;
; 1032          2229  3
; 1033          2230  3            !
; 1034          2231  3            ! SHOW TASK /FULL.  Construct an Adverb Node and link it in.
; 1035          2232  3            !
; 1036          2233  3            [ DBG$NMATCH( .INPUT_DESC, DBG$CS_FULL, 1 ) ]:
; 1037          2234  4                BEGIN
; 1038          2235  4                ADVERB_NODE = DBG$GET_TEMPMEM (DBG$K_ADVERB_NODE_SIZE);
; 1039          2236  4                .LINK = .ADVERB_NODE;
; 1040          2237  4                LINK = ADVERB_NODE [DBG$L_ADVERB_LINK];
; 1041          2238  4                ADVERB_NODE [DBG$B_ADVERB_LITERA[] = TASK_FULL;
; 1042          2239  3                END;
; 1043          2240  3
; 1044          2241  3            !
; 1045          2242  3            ! SHOW TASK /HOLD.  Construct an Adverb Node and link it in.
; 1046          2243  3            !
; 1047          2244  3            [ DBG$NMATCH( .INPUT_DESC, DBG$CS_HOLD, 1 ) ]:
; 1048          2245  4                BEGIN
; 1049          2246  4                ADVERB_NODE = DBG$GET_TEMPMEM (DBG$K_ADVERB_NODE_SIZE);
; 1050          2247  4                .LINK = .ADVERB_NODE;
```

```
;  1051     2248  4                LINK = ADVERB_NODE [DBG$L_ADVERB_LINK];
;  1052     2249  4                ADVERB_NODE [DBG$B_ADVERB_LITERA[] = TASK_HOLD;
;  1053     2250  3                END;
;  1054     2251  3
;  1055     2252  3
;  1056     2253  3            ! SHOW TASK /PRIORITY=(n).  Construct an Adverb Node and link it in.
;  1057     2254  3            !
;  1058     2255  3            [ DBG$NMATCH( .INPUT_DESC, DBG$CS_PRIORITY, 1 ) ]:
;  1059     2256  4                BEGIN
;  1060     2257  4                ADVERB_NODE = DBG$GET_TEMPMEM (DBG$K_ADVERB_NODE_SIZE);
;  1061     2258  4                .LINK = .ADVERB_NODE;
;  1062     2259  4                LINK = ADVERB_NODE [DBG$L_ADVERB_LINK];
;  1063     2260  4                ADVERB_NODE [DBG$B_ADVERB_LITERA[] = TASK_PRIORITY;
;  1064     2261  4
;  1065     2262  4                IF DBG$NMATCH( .INPUT_DESC, DBG$CS_COLON, 1 )
;  1066     2263  4                    OR DBG$NMATCH( .INPUT_DESC, DBG$CS_EQUAL, 1 )
;  1067     2264  4                THEN
;  1068     2265  4                    IF DBG$NMATCH( .INPUT_DESC, dbg$cs_left_paren, 1 )
;  1069     2266  4                    THEN
;  1070     2267  5                        BEGIN
;  1071     2268  5                        DO
;  1072     2269  6                            BEGIN
;  1073     2270  6                            DBG$NSAVE_DECIMAL_INTEGER(
;  1074     2271  6                                .INPUT_DESC,                        ! read input value
;  1075     2272  6                                PRIORITY);
;  1076     2273  6                            IF .PRIORITY GTRU 31                    ! %((need priority limit -tbs))%
;  1077     2274  6                            THEN
;  1078     2275  6                                SIGNAL (DBG$_BITRANGE );            ! %((NEED A BETTER MESSAGE -tbs))%
;  1079     2276  6                            (ADVERB_NODE [DBG$L_ADVERB_VALUE]) <.PRIORITY, 1, 0> = 1;   ! set corresponding
;  1080     2277  6                            END
;  1081     2278  5                        WHILE DBG$NMATCH( .INPUT_DESC, dbg$cs_comma, 1 );
;  1082     2279  5                        IF NOT DBG$NMATCH( .INPUT_DESC, dbg$cs_right_paren, 1 )
;  1083     2280  5                        THEN
;  1084     2281  5                            SIGNAL (dbg$_UNMTCHPARN);   ! Unmatched left parenthesis found.
;  1085     2282  5                        END
;  1086     2283  4                    ELSE
;  1087     2284  5                        BEGIN
;  1088     2285  5                        DBG$NSAVE_DECIMAL_INTEGER(
;  1089     2286  5                                .INPUT_DESC,                        ! read input value
;  1090     2287  5                                PRIORITY);
;  1091     2288  5                        IF .PRIORITY GTRU 31                    ! %((need a limit -tbs))%
;  1092     2289  5                        THEN
;  1093     2290  5                            SIGNAL (DBG$_BITRANGE );            ! %((NEED A BETTER MESSAGE -tbs))%
;  1094     2291  5                        (ADVERB_NODE [DBG$L_ADVERB_VALUE]) <.PRIORITY, 1, 0> = 1;       ! set corresponding
;  1095     2292  5                        END
;  1096     2293  4                ELSE
;  1097     2294  4                    SIGNAL (DBG$_NEEDMORE);
;  1098     2295  4
;  1099     2296  3                END;
;  1100     2297  3
;  1101     2298  3
;  1102     2299  3            ! SHOW TASK /STATE=(x).  Construct an Adverb Node and link it in.
;  1103     2300  3            !
;  1104     2301  3            [ DBG$NMATCH( .INPUT_DESC, DBG$CS_STATE, 5 ) ]:
;  1105     2302  4                BEGIN
;  1106     2303  4                ADVERB_NODE = DBG$GET_TEMPMEM (DBG$K_ADVERB_NODE_SIZE);
;  1107     2304  4                .LINK = .ADVERB_NODE;
```

```
;  1108          2305  4                  LINK = ADVERB_NODE [DBG$L_ADVERB_LINK];
;  1109          2306  4                  ADVERB_NODE [DBG$B_ADVERB_LITERAL] = TASK_STATE;
;  1110          2307  4
;  1111          2308  4                  IF DBG$NMATCH( .INPUT_DESC, DBG$CS_COLON, 1 )
;  1112          2309  4                      OR DBG$NMATCH( .INPUT_DESC, DBG$CS_EQUAL, 1 )
;  1113          2310  4                  THEN
;  1114          2311  4                      IF DBG$NMATCH( .INPUT_DESC, dbg$cs_left_paren, 1 )
;  1115          2312  4                      THEN
;  1116          2313  5                          BEGIN
;  1117          2314  5                          DO
;  1118          2315  6                              BEGIN
;  1119          2316  6                              SELECTONE TRUE OF
;  1120          2317  6                                  SET
;  1121          2318  6 %((THIS WILL OVERWRITE ADVERB VALUE WITH THE MOST RECENT STATE OF THE LIST -- MUST BE FIXED -tbs))%
;  1122          2319  6                                  [ DBG$NMATCH( .INPUT_DESC, DBG$CS_RUNNING, 1 ) ]:
;  1123          2320  6                                      ADVERB_NODE [DBG$L_ADVERB_VALUE] = DBGEXT$K_STATE_RUNNING;
;  1124          2321  6
;  1125          2322  6                                  [ DBG$NMATCH( .INPUT_DESC, DBG$CS_READY, 1 ) ]:
;  1126          2323  6                                      ADVERB_NODE [DBG$L_ADVERB_VALUE] = DBGEXT$K_STATE_READY;
;  1127          2324  6
;  1128          2325  6                                  [ DBG$NMATCH( .INPUT_DESC, DBG$CS_SUSPENDED, 1 ) ]:
;  1129          2326  6                                      ADVERB_NODE [DBG$L_ADVERB_VALUE] = DBGEXT$K_STATE_SUSPENDED;
;  1130          2327  6
;  1131          2328  6                                  [ DBG$NMATCH( .INPUT_DESC, DBG$CS_TERMINATED, 1 ) ]:
;  1132          2329  6                                      ADVERB_NODE [DBG$L_ADVERB_VALUE] = DBGEXT$K_STATE_TERMINATED;
;  1133          2330  6                                  !
;  1134          2331  6                                  ! Any other condition is an error.
;  1135          2332  6                                  !
;  1136          2333  6                                  [ OTHERWISE ]:
;  1137          2334  6                                      DBG$SYNTAX_ERROR(.INPUT_DESC);
;  1138          2335  6
;  1139          2336  6                                  TES
;  1140          2337  6                              END
;  1141          2338  5                          WHILE DBG$NMATCH( .INPUT_DESC, dbg$cs_comma, 1 );
;  1142          2339  5                          IF NOT DBG$NMATCH( .INPUT_DESC, dbg$cs_right_paren, 1 )
;  1143          2340  5                          THEN
;  1144          2341  5                              SIGNAL (dbg$_UNMTCHPARN);   ! Unmatched left parenthesis found.
;  1145          2342  5                          END
;  1146          2343  4                  ELSE
;  1147          2344  4                      SELECTONE TRUE OF
;  1148          2345  4                          SET
;  1149          2346  4
;  1150          2347  4                          [ DBG$NMATCH( .INPUT_DESC, DBG$CS_RUNNING, 1 ) ]:
;  1151          2348  4                              ADVERB_NODE [DBG$L_ADVERB_VALUE] = DBGEXT$K_STATE_RUNNING;
;  1152          2349  4
;  1153          2350  4                          [ DBG$NMATCH( .INPUT_DESC, DBG$CS_READY, 1 ) ]:
;  1154          2351  4                              ADVERB_NODE [DBG$L_ADVERB_VALUE] = DBGEXT$K_STATE_READY;
;  1155          2352  4
;  1156          2353  4                          [ DBG$NMATCH( .INPUT_DESC, DBG$CS_SUSPENDED, 1 ) ]:
;  1157          2354  4                              ADVERB_NODE [DBG$L_ADVERB_VALUE] = DBGEXT$K_STATE_SUSPENDED;
;  1158          2355  4
;  1159          2356  4                          [ DBG$NMATCH( .INPUT_DESC, DBG$CS_TERMINATED, 1 ) ]:
;  1160          2357  4                              ADVERB_NODE [DBG$L_ADVERB_VALUE] = DBGEXT$K_STATE_TERMINATED;
;  1161          2358  4                          !
;  1162          2359  4                          ! Any other condition is an error.
;  1163          2360  4                          !
;  1164          2361  4                          [ OTHERWISE ]:
```

```
;  1165     2362  4                                DBG$SYNTAX_ERROR(.INPUT_DESC);
;  1166     2363  4
;  1167     2364  4                            TES
;  1168     2365  4                    ELSE
;  1169     2366  4                        SIGNAL (DBG$_NEEDMORE);
;  1170     2367  4
;  1171     2368  3                    END;
;  1172     2369  3
;  1173     2370  3                !
;  1174     2371  3                ! SHOW TASK /STATISTICS.  Construct an Adverb Node and link it in.
;  1175     2372  3                !
;  1176     2373  3                [ DBG$NMATCH( .INPUT_DESC, DBG$CS_STATISTICS, 5 ) ]:
;  1177     2374  4                    BEGIN
;  1178     2375  4                    ADVERB_NODE = DBG$GET_TEMPMEM (DBG$K_ADVERB_NODE_SIZE);
;  1179     2376  4                    .LINK = .ADVERB_NODE;
;  1180     2377  4                    LINK = ADVERB_NODE [DBG$L_ADVERB_LINK];
;  1181     2378  4                    ADVERB_NODE [DBG$B_ADVERB_LITERA[] = TASK_STATISTICS;
;  1182     2379  3                    END;
;  1183     2380  3
;  1184     2381  3                !
;  1185     2382  3                ! Any other condition is an error.
;  1186     2383  3                !
;  1187     2384  3                [ OTHERWISE ]:
;  1188     2385  3                    DBG$SYNTAX_ERROR(.INPUT_DESC);
;  1189     2386  3
;  1190     2387  3                TES;
;  1191     2388  3
;  1192     2389  2            END;                ! of WHILE /qualifier
;  1193     2390  2
;  1194     2391  2        .LINK = 0;              ! End of adverb node chain.
;  1195     2392  2
;  1196     2393  2
;  1197     2394  2        IF NOT DBG$NMATCH( .INPUT_DESC, DBG$CS_CR, 1 )      ! If more input exists then
;  1198     2395  2        THEN                                                ! try to parse a task list.
;  1199     2396  3            BEGIN
;  1200     2397  3            LINK = VERB_NODE [DBG$L_VERB_OBJECT_PTR];
;  1201     2398  3            DO
;  1202     2399  3                !+
;  1203     2400  3                !   Parse tasks in a task list and build noun nodes and value
;  1204     2401  3                !   descriptors until end of list or an error is encountered.
;  1205     2402  3                !-
;  1206     2403  4                BEGIN
;  1207     2404  4                NOUN_NODE = DBG$GET_TEMPMEM (DBG$K_NOUN_NODE_SIZE); ! %((NEED LONG NOUN ?-tbs))%
;  1208     2405  4                .LINK = .NOUN_NODE;
;  1209     2406  4                LINK = NOUN_NODE [DBG$L_NOUN_LINK];
;  1210     2407  4
;  1211     2408  4                DBG$NPARSE_EXPRESSION (
;  1212     2409  4                    .INPUT_DESC,                            ! rest of command
;  1213     2410  4                    .DBG$GB_RADIX[DBG$B_RADIX_INPUT],       ! default input radix
;  1214     2411  4                    NOUN_NODE [DBG$L_NOUN_VALUE],           ! where to store ptr to value desc
;  1215     2412  4                    TOKEN$K_TERM_COMMA );                   ! task terminator token
;  1216     2413  4
;  1217     2414  4                !   .MESSAGE_VECT);                        %((REMOVE MESSAGE VEC FROM ROUTINE -tbs))%
;  1218     2415  4
;  1219     2416  4                END
;  1220     2417  3            WHILE DBG$NMATCH( .INPUT_DESC, DBG$CS_COMMA, 1 );
;  1221     2418  3
```

```
;  1222     2419  3          IF NOT DBG$NMATCH( .INPUT_DESC, DBG$CS_CR, 1 )  ! If more input exists then
;  1223     2420  3          THEN                                            ! we have an error.
;  1224     2421  3              DBG$SYNTAX_ERROR(.INPUT_DESC);              ! Signal the error.
;  1225     2422  2          END;
;  1226     2423  2
;  1227     2424  2
;  1228     2425  2      RETURN 0;                       ! %((0?-tbs))%
;  1229     2426  2
;  1230     2427  1      END;                            ! end of DBG$NPARSE_SHOW_TASK
```

```
                                07FC 00000          .ENTRY  DBG$NPARSE_SHOW_TASK, Save R2,R3,R4,R5,R6,- ; 2135
                                                            R7,R8,R9,R10
                    5A 00000000G   00  9E 00002     MOVAB   DBG$SYNTAX_ERROR, R10                       ;
                    59 00000000G   00  9E 00009     MOVAB   LIB$SIGNAL, R9                              ;
                    58 00000000G   00  9E 00010     MOVAB   DBG$NSAVE_DECIMAL_INTEGER, R8               ;
                    57 00000000G   00  9E 00017     MOVAB   DBG$GET_TEMPMEM, R7                         ;
                    56 00000000'   EF  9E 0001E     MOVAB   DBG$CS_COLON, R6                            ;
                    55 00000000G   00  9E 00025     MOVAB   DBG$NMATCH, R5                              ;
                    5E             04  C2 0002C     SUBL2   #4, SP                                      ;
              54    08             AC  04  C1 0002F     ADDL3   #4, VERB_NODE, LINK                     ; 2175
                    53       04    AC  D0 00034     MOVL    INPUT_DESC, R3                              ; 2180
                                   01  DD 00038 1$:  PUSHL   #1                                         ;
                             08    A6  9F 0003A     PUSHAB  DBG$CS_SLASH                                ;
                                   53  DD 0003D     PUSHL   R3                                          ;
                    65             03  FB 0003F     CALLS   #3, DBG$NMATCH                              ;
                    03             50  E8 00042     BLBS    R0, 2$                                      ;
                              02BF 31  00045     BRW     39$                                         ;
                                   01  DD 00048 2$:  PUSHL   #1                                         ; 2190
                          FF79     C6  9F 0004A     PUSHAB  DBG$CS_ALL                                  ;
                                   53  DD 0004E     PUSHL   R3                                          ;
                    65             03  FB 00050     CALLS   #3, DBG$NMATCH                              ;
                    01             50  D1 00053     CMPL    R0, #1                                      ;
                                   14  12 00056     BNEQ    4$                                          ;
                                   03  DD 00058     PUSHL   #3                                          ; 2192
                    67             01  FB 0005A     CALLS   #1, DBG$GET_TEMPMEM                         ;
                    52             50  D0 0005D     MOVL    R0, ADVERB_NODE                             ;
                    64             52  D0 00060     MOVL    ADVERB_NODE, (LINK)                         ; 2193
                    54       08    A2  9E 00063     MOVAB   8(R2), LINK                                 ; 2194
                    62             02  90 00067     MOVB    #2, (ADVERB_NODE)                           ; 2195
                                   CC  11 0006A 3$:  BRB     1$                                          ; 2185
                                   01  DD 0006C 4$:  PUSHL   #1                                         ; 2202
                          FF7D     C6  9F 0006E     PUSHAB  DBG$CS_CALLS                                ;
                                   53  DD 00072     PUSHL   R3                                          ;
                    65             03  FB 00074     CALLS   #3, DBG$NMATCH                              ;
                    01             50  D1 00077     CMPL    R0, #1                                      ;
                                   3B  12 0007A     BNEQ    8$                                          ;
                                   03  DD 0007C     PUSHL   #3                                          ; 2204
                    67             01  FB 0007E     CALLS   #1, DBG$GET_TEMPMEM                         ;
                    52             50  D0 00081     MOVL    R0, ADVERB_NODE                             ;
                    64             52  D0 00084     MOVL    ADVERB_NODE, (LINK)                         ; 2205
                    54       08    A2  9E 00087     MOVAB   8(R2), LINK                                 ; 2206
                    62             03  90 0008B     MOVB    #3, (ADVERB_NODE)                           ; 2207
                                   01  DD 0008E     PUSHL   #1                                          ; 2209
```

```
                 0048  8F  BB 00090          PUSHR   #^M<R3,R6>                                  ;
           65          03  FB 00094          CALLS   #3, DBG$NMATCH                             ;
           0D          50  E8 00097          BLBS    R0, 5$                                     ;
                       01  DD 0009A          PUSHL   #1                                         ; 2210
                 06    A6  9F 0009C          PUSHAB  DBG$CS_EQUAL                               ;
                       53  DD 0009F          PUSHL   R3                                         ;
           65          03  FB 000A1          CALLS   #3, DBG$NMATCH                             ;
           0A          50  E9 000A4          BLBC    R0, 7$                                     ;
                 04    A2  9F 000A7  5$:     PUSHAB  4(ADVERB_NODE)                             ; 2214
                       53  DD 000AA          PUSHL   R3                                         ;
           68          02  FB 000AC          CALLS   #2, DBG$NSAVE_DECIMAL_INTEGER              ;
                       87  11 000AF  6$:     BRB     1$                                         ;
        04 A2          01  CE 000B1  7$:     MNEGL   #1, 4(ADVERB_NODE)                         ; 2216
                       81  11 000B5          BRB     1$                                         ; 2185
                       01  DD 000B7  8$:     PUSHL   #1                                         ; 2222
                 83    A6  9F 000B9          PUSHAB  DBG$CS_DEADLOCK                            ;
                       53  DD 000BC          PUSHL   R3                                         ;
           65          03  FB 000BE          CALLS   #3, DBG$NMATCH                             ;
           01          50  D1 000C1          CMPL    R0, #1                                     ;
                       14  12 000C4          BNEQ    9$                                         ;
                       03  DD 000C6          PUSHL   #3                                         ; 2224
           67          01  FB 000C8          CALLS   #1, DBG$GET_TEMPMEM                        ;
           52          50  D0 000CB          MOVL    R0, ADVERB_NODE                            ;
           64          52  D0 000CE          MOVL    ADVERB_NODE, (LINK)                        ; 2225
           54    08    A2  9E 000D1          MOVAB   8(R2), LINK                                ; 2226
           62          04  90 000D5          MOVB    #4, (ADVERB_NODE)                          ; 2227
                       90  11 000D8          BRB     3$                                         ; 2185
                       01  DD 000DA  9$:     PUSHL   #1                                         ; 2233
                 8C    A6  9F 000DC          PUSHAB  DBG$CS_FULL                                ;
                       53  DD 000DF          PUSHL   R3                                         ;
           65          03  FB 000E1          CALLS   #3, DBG$NMATCH                             ;
           01          50  D1 000E4          CMPL    R0, #1                                     ;
                       14  12 000E7          BNEQ    10$                                        ;
                       03  DD 000E9          PUSHL   #3                                         ; 2235
           67          01  FB 000EB          CALLS   #1, DBG$GET_TEMPMEM                        ;
           52          50  D0 000EE          MOVL    R0, ADVERB_NODE                            ;
           64          52  D0 000F1          MOVL    ADVERB_NODE, (LINK)                        ; 2236
           54    08    A2  9E 000F4          MOVAB   8(R2), LINK                                ; 2237
           62          05  90 000F8          MOVB    #5, (ADVERB_NODE)                          ; 2238
                       B2  11 000FB          BRB     6$                                         ; 2185
                       01  DD 000FD  10$:     PUSHL   #1                                         ; 2244
                 91    A6  9F 000FF          PUSHAB  DBG$CS_HOLD                                ;
                       53  DD 00102          PUSHL   R3                                         ;
           65          03  FB 00104          CALLS   #3, DBG$NMATCH                             ;
           01          50  D1 00107          CMPL    R0, #1                                     ;
                       14  12 0010A          BNEQ    11$                                        ;
                       03  DD 0010C          PUSHL   #3                                         ; 2246
           67          01  FB 0010E          CALLS   #1, DBG$GET_TEMPMEM                        ;
           52          50  D0 00111          MOVL    R0, ADVERB_NODE                            ;
           64          52  D0 00114          MOVL    ADVERB_NODE, (LINK)                        ; 2247
           54    08    A2  9E 00117          MOVAB   8(R2), LINK                                ; 2248
           62          06  90 0011B          MOVB    #6, (ADVERB_NODE)                          ; 2249
                       8F  11 0011E          BRB     6$                                         ; 2185
                       01  DD 00120  11$:     PUSHL   #1                                         ; 2255
                 9D    A6  9F 00122          PUSHAB  DBG$CS_PRIORITY                            ;
                       53  DD 00125          PUSHL   R3                                         ;
           65          03  FB 00127          CALLS   #3, DBG$NMATCH                             ;
```

```
01                50  D1 0012A          CMPL    R0, #1                                  ;
                  03  13 0012D          BEQL    12$                                     ;
                0095  31 0012F          BRW     21$                                     ;
                  03  DD 00132 12$:     PUSHL   #3                                      ; 2257
67                01  FB 00134          CALLS   #1, DBG$GET_TEMPMEM                     ;
52                50  D0 00137          MOVL    R0, ADVERB_NODE                         ;
64                52  D0 0013A          MOVL    ADVERB_NODE, (LINK)                     ; 2258
54          08    A2  9E 0013D          MOVAB   8(R2), LINK                             ; 2259
62                07  90 00141          MOVB    #7, (ADVERB_NODE)                       ; 2260
                  01  DD 00144          PUSHL   #1                                      ; 2262
              0048  8F  BB 00146          PUSHR   #^M<R3,R6>                              ;
65                03  FB 0014A          CALLS   #3, DBG$NMATCH                          ;
10                50  E8 0014D          BLBS    R0, 13$                                 ;
                  01  DD 00150          PUSHL   #1                                      ; 2263
              06  A6  9F 00152          PUSHAB  DBG$CS_EQUAL                            ;
                  53  DD 00155          PUSHL   R3                                      ;
65                03  FB 00157          CALLS   #3, DBG$NMATCH                          ;
03                50  E8 0015A          BLBS    R0, 13$                                 ;
              0171  31 0015D          BRW     34$                                     ;
                  01  DD 00160 13$:     PUSHL   #1                                      ; 2265
              FC  A6  9F 00162          PUSHAB  DBG$CS_LEFT_PAREN                       ;
                  53  DD 00165          PUSHL   R3                                      ;
65                03  FB 00167          CALLS   #3, DBG$NMATCH                          ;
3D                50  E9 0016A          BLBC    R0, 18$                                 ;
            4008  8F  BB 0016D 14$:     PUSHR   #^M<R3,SP>                              ; 2271
68                02  FB 00171          CALLS   #2, DBG$NSAVE_DECIMAL_INTEGER           ;
1F                6E  D1 00174          CMPL    PRIORITY, #31                           ; 2273
                  09  1B 00177          BLEQU   15$                                     ;
        00028248  8F  DD 00179          PUSHL   #164424                                 ; 2275
69                01  FB 0017F          CALLS   #1, LIB$SIGNAL                          ;
00      04        A2  6E  E2 00182 15$:     BBSS    PRIORITY, 4(ADVERB_NODE), 16$       ; 2276
                  01  DD 00187 16$:     PUSHL   #1                                      ; 2278
              02  A6  9F 00189          PUSHAB  DBG$CS_COMMA                            ;
                  53  DD 0018C          PUSHL   R3                                      ;
65                03  FB 0018E          CALLS   #3, DBG$NMATCH                          ;
D9                50  E8 00191          BLBS    R0, 14$                                 ;
                  01  DD 00194 17$:     PUSHL   #1                                      ; 2279
              FE  A6  9F 00196          PUSHAB  DBG$CS_RIGHT_PAREN                      ;
                  53  DD 00199          PUSHL   R3                                      ;
65                03  FB 0019B          CALLS   #3, DBG$NMATCH                          ;
23                50  E8 0019E          BLBS    R0, 20$                                 ;
        000287D0  8F  DD 001A1          PUSHL   #165840                                 ; 2281
              012D  31 001A7          BRW     35$                                     ;
            4008  8F  BB 001AA 18$:     PUSHR   #^M<R3,SP>                              ; 2286
68                02  FB 001AE          CALLS   #2, DBG$NSAVE_DECIMAL_INTEGER           ;
1F                6E  D1 001B1          CMPL    PRIORITY, #31                           ; 2288
                  09  1B 001B4          BLEQU   19$                                     ;
        00028248  8F  DD 001B6          PUSHL   #164424                                 ; 2290
69                01  FB 001BC          CALLS   #1, LIB$SIGNAL                          ;
00      04        A2  6E  E2 001BF 19$:     BBSS    PRIORITY, 4(ADVERB_NODE), 20$       ; 2291
              FE71  31 001C4 20$:     BRW     1$                                      ; 2265
                  05  DD 001C7 21$:     PUSHL   #5                                      ; 2301
              B6  A6  9F 001C9          PUSHAB  DBG$CS_STATE                            ;
                  53  DD 001CC          PUSHL   R3                                      ;
65                03  FB 001CE          CALLS   #3, DBG$NMATCH                          ;
01                50  D1 001D1          CMPL    R0, #1                                  ;
                  03  13 001D4          BEQL    22$                                     ;
```

```
                0103  31 001D6          BRW     36$
                  03  DD 001D9 22$:     PUSHL   #3                                      ;2303
              67    01  FB 001DB          CALLS   #1, DBG$GET_TEMPMEM
              52    50  D0 001DE          MOVL    R0, ADVERB_NODE
              64    52  D0 001E1          MOVL    ADVERB_NODE, (LINK)                     ;2304
              54  08    A2 9E 001E4       MOVAB   8(R2), LINK                             ;2305
              62    0A  90 001E8          MOVB    #10, (ADVERB_NODE)                      ;2306
                  01  DD 001EB          PUSHL   #1                                      ;2308
            0048  8F  BB 001ED          PUSHR   #^M<R3,R6>
              65  03  FB 001F1          CALLS   #3, DBG$NMATCH
              10  50  E8 001F4          BLBS    R0, 23$
                  01  DD 001F7          PUSHL   #1                                      ;2309
                06  A6 9F 001F9          PUSHAB  DBG$CS_EQUAL
                  53  DD 001FC          PUSHL   R3
              65  03  FB 001FE          CALLS   #3, DBG$NMATCH
              03  50  E8 00201          BLBS    R0, 23$
                00CA  31 00204          BRW     34$
                  01  DD 00207 23$:     PUSHL   #1                                      ;2311
                FC  A6 9F 00209          PUSHAB  DBG$CS_LEFT_PAREN
                  53  DD 0020C          PUSHL   R3
              65  03  FB 0020E          CALLS   #3, DBG$NMATCH
              69  50  E9 00211          BLBC    R0, 30$
                  01  DD 00214 24$:     PUSHL   #1                                      ;2319
                DF  A6 9F 00216          PUSHAB  DBG$CS_RUNNING
                  53  DD 00219          PUSHL   R3
              65  03  FB 0021B          CALLS   #3, DBG$NMATCH
              01  50  D1 0021E          CMPL    R0, #1
                  06  12 00221          BNEQ    25$
          04  A2      01  D0 00223          MOVL    #1, 4(ADVERB_NODE)                  ;2320
                  44  11 00227          BRB     29$
                  01  DD 00229 25$:     PUSHL   #1                                      ;2322
                D9  A6 9F 0022B          PUSHAB  DBG$CS_READY
                  53  DD 0022E          PUSHL   R3
              65  03  FB 00230          CALLS   #3, DBG$NMATCH
              01  50  D1 00233          CMPL    R0, #1
                  06  12 00236          BNEQ    26$
          04  A2      02  D0 00238          MOVL    #2, 4(ADVERB_NODE)                  ;2323
                  2F  11 0023C          BRB     29$
                  01  DD 0023E 26$:     PUSHL   #1                                      ;2325
                E7  A6 9F 00240          PUSHAB  DBG$CS_SUSPENDED
                  53  DD 00243          PUSHL   R3
              65  03  FB 00245          CALLS   #3, DBG$NMATCH
              01  50  D1 00248          CMPL    R0, #1
                  06  12 0024B          BNEQ    27$
          04  A2      04  D0 0024D          MOVL    #4, 4(ADVERB_NODE)                  ;2326
                  1A  11 00251          BRB     29$
                  01  DD 00253 27$:     PUSHL   #1                                      ;2328
                F1  A6 9F 00255          PUSHAB  DBG$CS_TERMINATED
                  53  DD 00258          PUSHL   R3
              65  03  FB 0025A          CALLS   #3, DBG$NMATCH
              01  50  D1 0025D          CMPL    R0, #1
                  06  12 00260          BNEQ    28$
          04  A2      08  D0 00262          MOVL    #8, 4(ADVERB_NODE)                  ;2329
                  05  11 00266          BRB     29$
                  53  DD 00268 28$:     PUSHL   R3                                      ;2334
              6A  01  FB 0026A          CALLS   #1, DBG$SYNTAX_ERROR
                  01  DD 0026D 29$:     PUSHL   #1                                      ;2338
```

```
                                02    A6  9F 0026F          PUSHAB  DBG$CS_COMMA                    ;
                                      53  DD 00272          PUSHL   R3                              ;
                                65    03  FB 00274          CALLS   #3, DBG$NMATCH                  ;
                                9A    50  E8 00277          BLBS    R0, 24$                         ;
                                   FF17 31  00027A          BRW     17$                             ; 2339
                                      01  DD 0027D 30$:      PUSHL   #1                              ; 2347
                                DF    A6  9F 0027F          PUSHAB  DBG$CS_RUNNING                  ;
                                      53  DD 00282          PUSHL   R3                              ;
                                65    03  FB 00284          CALLS   #3, DBG$NMATCH                  ;
                                01    50  D1 00287          CMPL    R0, #1                          ;
                                      06  12 0028A          BNEQ    31$                             ;
                             04 A2    01  D0 0028C          MOVL    #1, 4(ADVERB_NODE)              ; 2348
                                      72  11 00290          BRB     38$                             ;
                                      01  DD 00292 31$:      PUSHL   #1                              ; 2350
                                D9    A6  9F 00294          PUSHAB  DBG$CS_READY                    ;
                                      53  DD 00297          PUSHL   R3                              ;
                                65    03  FB 00299          CALLS   #3, DBG$NMATCH                  ;
                                01    50  D1 0029C          CMPL    R0, #1                          ;
                                      06  12 0029F          BNEQ    32$                             ;
                             04 A2    02  D0 002A1          MOVL    #2, 4(ADVERB_NODE)              ; 2351
                                      5D  11 002A5          BRB     38$                             ;
                                      01  DD 002A7 32$:      PUSHL   #1                              ; 2353
                                E7    A6  9F 002A9          PUSHAB  DBG$CS_SUSPENDED                ;
                                      53  DD 002AC          PUSHL   R3                              ;
                                65    03  FB 002AE          CALLS   #3, DBG$NMATCH                  ;
                                01    50  D1 002B1          CMPL    R0, #1                          ;
                                      06  12 002B4          BNEQ    33$                             ;
                             04 A2    04  D0 002B6          MOVL    #4, 4(ADVERB_NODE)              ; 2354
                                      48  11 002BA          BRB     38$                             ;
                                      01  DD 002BC 33$:      PUSHL   #1                              ; 2356
                                F1    A6  9F 002BE          PUSHAB  DBG$CS_TERMINATED               ;
                                      53  DD 002C1          PUSHL   R3                              ;
                                65    03  FB 002C3          CALLS   #3, DBG$NMATCH                  ;
                                01    50  D1 002C6          CMPL    R0, #1                          ;
                                      34  12 002C9          BNEQ    37$                             ;
                             04 A2    08  D0 002CB          MOVL    #8, 4(ADVERB_NODE)              ; 2357
                                      33  11 002CF          BRB     38$                             ;
                          000280D0    8F  DD 002D1 34$:      PUSHL   #164048                         ; 2366
                                69    01  FB 002D7 35$:      CALLS   #1, LIB$SIGNAL                  ;
                                      28  11 002DA          BRB     38$                             ; 2185
                                      05  DD 002DC 36$:      PUSHL   #5                              ; 2373
                                   BC A6  9F 002DE          PUSHAB  DBG$CS_STATISTICS               ;
                                      53  DD 002E1          PUSHL   R3                              ;
                                65    03  FB 002E3          CALLS   #3, DBG$NMATCH                  ;
                                01    50  D1 002E6          CMPL    R0, #1                          ;
                                      14  12 002E9          BNEQ    37$                             ;
                                      03  DD 002EB          PUSHL   #3                              ; 2375
                                67    01  FB 002ED          CALLS   #1, DBG$GET_TEMPMEM             ;
                                52    50  D0 002F0          MOVL    R0, ADVERB_NODE                 ;
                                64    52  D0 002F3          MOVL    ADVERB_NODE, (LINK)             ; 2376
                                54 08 A2    9E 002F6          MOVAB   8(R2), LINK                     ; 2377
                                62    0B  90 002FA          MOVB    #11, (ADVERB_NODE)              ; 2378
                                      05  11 002FD          BRB     38$                             ; 2185
                                      53  DD 002FF 37$:      PUSHL   R3                              ; 2385
                                6A    01  FB 00301          CALLS   #1, DBG$SYNTAX_ERROR            ;
                                   FD31 31  00304 38$:      BRW     1$                              ; 2180
                                      64  D4 00307 39$:      CLRL    (LINK)                          ; 2391
```

```
                                    01  DD 00309          PUSHL   #1                                          ; 2394
                                04  A6  9F 0030B          PUSHAB  DBG$CS_CR
                                    53  DD 0030E          PUSHL   R3
                            65      03  FB 00310          CALLS   #3, DBG$NMATCH
                            47      50  E8 00313          BLBS    R0, 41$
                    54      08  AC      08  C1 00316      ADDL3   #8, VERB_NODE, LINK                        ; 2397
                                    04  DD 0031B 40$:     PUSHL   #4                                          ; 2404
                            67      01  FB 0031D          CALLS   #1, DBG$GET_TEMPMEM
                            52      50  D0 00320          MOVL    R0, NOUN_NODE
                            64      52  D0 00323          MOVL    NOUN_NODE, (LINK)                           ; 2405
                            54      08  A2  9E 00326      MOVAB   8(R2), LINK                                 ; 2406
                                    01  DD 0032A          PUSHL   #1                                          ; 2411
                                    52  DD 0032C          PUSHL   NOUN_NODE
                        7E  00000000G   00  9A 0032E      MOVZBL  DBG$GB_RADIX, -(SP)
                                    53  DD 00335          PUSHL   R3
                00000000G   00          04  FB 00337      CALLS   #4, DBG$NPARSE_EXPRESSION
                                    01  DD 00[illegible]  PUSHL   #1                                          ; 2417
                                02  A6  9F [illegible]    PUSHAB  DBG$CS_COMMA
                                    53  DD [illegible]    PUSHL   R3
                            65      03  [illegible] 00345 CALLS   #3, DBG$NMATCH
                            D0      50  E8 00348          BLBS    R0, 40$
                                    01  DD 0034B          PUSHL   #1                                          ; 2419
                                04  A6  9F 0034D          PUSHAB  DBG$CS_CR
                                    53  DD 00350          PUSHL   R3
                            65      03  FB 00352          CALLS   #3, DBG$NMATCH
                            05      50  E8 00355          BLBS    R0, 41$
                                    53  DD 00358          PUSHL   R3                                          ; 2421
                            6A      01  FB 0035A          CALLS   #1, DBG$SYNTAX_ERROR
                                        04 0035D 41$:     RET                                                 ; 2427

; Routine Size:  862 bytes,    Routine Base:  DBG$CODE + 07CE

; 1231          2428  1
```

```
;  1233          2429  1 ROUTINE DBGEXT$PRINT_ROUTINE (FLAG, FUNCTION, STRING, FAO_ARG) : NOVALUE =      ! %((-tbs))%
;  1234          2430  1 !
;  1235          2431  1 ! FUNCTION
;  1236          2432  1 !       Function of this routine goes here.
;  1237          2433  1 !
;  1238          2434  1 ! INPUTS
;  1239          2435  1 !       List of inputs goes here, both explicit and implicit,
;  1240          2436  1 !       complete with descriptions.
;  1241          2437  1 !
;  1242          2438  1 ! OUTPUTS
;  1243          2439  1 !       List of outputs goes here, together with known side effects.
;  1244          2440  1 !
;  1245          2441  1
;  1246          2442  2     BEGIN
;  1247          2443  2
;  1248          2444  2     LOCAL
;  1249          2445  2         XXXXXXX;                        !<<--------- Local declarations --------
;  1250          2446  2
;  1251          2447  2
;  1252          2448  2
;  1253          2449  2     ! The text of the routine starts here.
;  1254          2450  2     !
;  1255          2451  2     !<<-------------------------- FIRST LINE OF CODE -------------------------
;  1256          2452  2     RETURN 0;
;  1257          2453  2
;  1258          2454  1     END;

                                0000 00000 DBGEXT$PRINT_ROUTINE:
                                                .WORD   Save nothing                    ;  2429
                                  04 00002          RET                                 ;  2454

; Routine Size:  3 bytes,    Routine Base:  DBG$CODE + 0B2C
```

```
;  1260          2455  1 ROUTINE LOCAL_ROUT_NAME =
;  1261          2456  1 !
;  1262          2457  1 ! FUNCTION
;  1263          2458  1 !       Function of this routine goes here.
;  1264          2459  1 !
;  1265          2460  1 ! INPUTS
;  1266          2461  1 !       List of inputs goes here, both explicit and implicit,
;  1267          2462  1 !       complete with descriptions.
;  1268          2463  1 !
;  1269          2464  1 ! OUTPUTS
;  1270          2465  1 !       List of outputs goes here, together with known side effects.
;  1271          2466  1 !
;  1272          2467  1
;  1273          2468  2     BEGIN
;  1274          2469  2
;  1275          2470  2     LOCAL
;  1276          2471  2         XXXXXXX;                        !<<--------- Local declarations --------
;  1277          2472  2
;  1278          2473  2
;  1279          2474  2
;  1280          2475  2     ! The text of the routine starts here.
;  1281          2476  2     !
;  1282          2477  2     !<<-------------------------- FIRST LINE OF CODE --------------------------
;  1283          2478  2     RETURN 0;
;  1284          2479  2
;  1285          2480  1     END;


                                        0000 00000 LOCAL_ROUT_NAME:
                                                           .WORD   Save nothing                        ; 2455
                                      50  D4 00002           CLRL    R0                                  ; 2478
                                          04 00004           RET                                         ; 2480

; Routine Size:  5 bytes,    Routine Base:  DBG$CODE + 0B2F

;  1286          2481  1
;  1287          2482  0 END ELUDOM



                                                           .EXTRN  LIB$SIGNAL

;                                  PSECT SUMMARY
;
;       Name                            Bytes                           Attributes
;
;  DBG$PLIT                              152  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)
;  DBG$CODE                             2868  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         0         0        1000       00:01.9
;  _$255$DUA28:[DEBUG.OBJ]STRUCDEF.L32;1     32         0         0           7       00:00.2
;  _$255$DUA28:[DEBUG.OBJ]DBGLIB.L32;1     1545        31         2          97       00:02.0
;  _$255$DUA28:[DEBUG.OBJ]DSTRECRDS.L32;1
;                                           418         0         0          31       00:00.3
;  _$255$DUA28:[DEBUG.OBJ]DBGMSG.L32;1      386         9         2          22       00:00.3
;  _$255$DUA28:[DEBUG.OBJ]DBGGEN.L32;1      150         2         1          12       00:00.3
```

```
;                                   COMMAND QUALIFIERS

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

; Size:         2868 code + 152 data bytes
; Run Time:       00:52.7
; Elapsed Time:   00:58.7
; Lines/CPU Min:   2828
; Lexemes/CPU-Min: 16781
; Memory Used:  451 pages
; Compilation Complete
```

DBGTASK
LIS

DBGVALUES
LIS

DBGTBK
LIS