```
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
```

FILEID**RSTTYPES

```
RRRRRRRR      SSSSSSSS  TTTTTTTTTT  TTTTTTTTTT  YY      YY  PPPPPPPP    EEEEEEEEEE    SSSSSSSS
RRRRRRRR      SSSSSSSS  TTTTTTTTTT  TTTTTTTTTT  YY      YY  PPPPPPPP    EEEEEEEEEE    SSSSSSSS
RR      RR  SS              TT          TT      YY      YY  PP      PP  EE            SS
RR      RR  SS              TT          TT      YY      YY  PP      PP  EE            SS
RR      RR  SS              TT          TT        YY  YY    PP      PP  EE            SS
RR      RR  SS              TT          TT        YY  YY    PP      PP  EE            SS
RRRRRRRR      SSSSSS        TT          TT          YY      PPPPPPPP    EEEEEEEE      SSSSSS
RRRRRRRR      SSSSSS        TT          TT          YY      PPPPPPPP    EEEEEEEE      SSSSSS
RR  RR              SS      TT          TT          YY      PP          EE                  SS
RR  RR              SS      TT          TT          YY      PP          EE                  SS
RR    RR            SS      TT          TT          YY      PP          EE                  SS
RR    RR            SS      TT          TT          YY      PP          EE                  SS
RR      RR  SSSSSSSS        TT          TT          YY      PP          EEEEEEEEEE  SSSSSSSS
RR      RR  SSSSSSSS        TT          TT          YY      PP          EEEEEEEEEE  SSSSSSSS


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

```
:     1          0001  0 MODULE RSTTYPES (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 !       Bert Beander    August, 1980
:    30          0030  1 !       Ken Nappa       August, 1980
:    31          0031  1 !
:    32          0032  1 ! MODULE FUNCTION
:    33          0033  1 !       This module contains symbol table access routines for types.  These
:    34          0034  1 !       routines accept a Type ID and return information about the correspond-
:    35          0035  1 !       ing data type.
:    36          0036  1 !
:    37          0037  1 ! MODIFIED BY
:    38          0038  1 !       Ping Sager      May 1982        Correct the way to get BITSIZE value
:    39          0039  1 !                                       in DBG$STA_TYP_VARIANT.
:    40          0040  1 !       Ping Sager      Oct 1982        Added DBG$TYPEID_FOR_SET routine
:    41          0041  1 !       Rich Title      Nov 1982        Added DBG$TYPEID_FOR_TPTR routine
:    42          0042  1 !       Ping Sager      Jan 1983        Handled more cases for DST has SEPTYP
:    43          0043  1 !       Rich Title      Jan 1982        Added DBG$TYPEID_FOR_ARRAY routine
:    44          0044  1 ! 006   Walter Carrell  Oct 1983        Added support for continuation DST records
:    45          0045  1 !
:    46          0046  1
:    47          0047  1 REQUIRE 'SRC$:DBGPROLOG.REQ';
:    48          0181  1
:    49          0182  1 FORWARD ROUTINE
:    50          0183  1     DBG$FIND_SEPTYP,                   ! Get Separate Type Specification
:    51          0184  1                                        !      from a given DST record
:    52          0185  1     DBG$GET_BITSIZE_FROM_DESC,         ! Get the size from the descriptor
:    53          0186  1     DBG$GET_BITSIZE_FROM_TYPESPEC,     ! Get the size from Type Spec.
:    54          0187  1     DBG$STA_SYMTYPE:    NOVALUE,       ! Return a symbol's TYPEID and FCODE
:    55          0188  1     DBG$STA_SYMSIZE:    NOVALUE,       ! Return the bit size of a data type
:    56          0189  1     DBG$STA_TYPEFCODE,                 ! Return the type format code
:    57          0190  1     DBG$STA_TYP_AREA: NOVALUE,         ! Return the info for an area type
```

```
;    58          0191  1     DBG$STA_TYP_ARRAY:  NOVALUE,          ! Return info for an array type
;    59          0192  1     DBG$STA_TYP_ATOMIC: NOVALUE,          ! Return type code for an atomic type
;    60          0193  1     DBG$STA_TYP_DESCR:  NOVALUE,          ! Return descriptor for descriptor type
;    61          0194  1     DBG$STA_TYP_ENUM:   NOVALUE,          ! Return info for an enumeration type
;    62          0195  1     DBG$STA_TYP_FILE:   NOVALUE,          ! Return info for a file type
;    63          0196  1     DBG$STA_TYP_OFFSET: NOVALUE,          ! Return info for a offset type
;    64          0197  1     DBG$STA_TYP_PICT:   NOVALUE,          ! Return info for a picture type
;    65          0198  1     DBG$STA_TYP_TYPEDPTR: NOVALUE,        ! Return info for a typed pointer type
;    66          0199  1     DBG$STA_TYP_RECORD: NOVALUE,          ! Return info for a record type
;    67          0200  1     DBG$STA_TYP_SET:    NOVALUE,          ! Return info for a set type
;    68          0201  1     DBG$STA_TYP_SUBRNG: NOVALUE,          ! Return info for a subrange type
;    69          0202  1     DBG$STA_TYP_VARIANT:NOVALUE,          ! Return info for a variant type
;    70          0203  1     DBG$STA_TYP_VARIANT_COMP: NOVALUE,    ! XXX
;    71          0204  1     DBG$TRANS_TYPE_CODE,                  ! Translates DST fcodes to RST fcodes
;    72          0205  1     DBG$TYPEID_FOR_ARRAY,                 ! Return Type ID for array
;    73          0206  1     DBG$TYPEID_FOR_ATOMIC,                ! Return Type ID for atomic type
;    74          0207  1     DBG$TYPEID_FOR_SET,                   ! Return Type ID for set type
;    75          0208  1     DBG$TYPEID_FOR_TPTR,                  ! Return Type ID for TPTR type
;    76          0209  1     TYPEID_FOR_COB_HACK,                  ! Return Type ID for COBOL Hack record
;    77          0210  1     TYPEID_FOR_DESCR,                     ! Return Type ID for descriptor type
;    78          0211  1     TYPEID_FROM_DST_TYPESPEC,             ! Return Type ID from a Type Speci-
;    79          0212  1                                           !      fication embedded in the DST
;    80          0213  1     TYPEID_FROM_DST_RECORD,
;    81          0214  1     DESC_OR_ARRAY_OR_ATOM,                ! Determines the nature of the type.
;    82          0215  1     FIND_RST_TYPE_ENTRY,
;    83          0216  1     FIND_TYPE_RECORD,                     ! Returns symid of type info
;    84          0217  1     FIND_TYPESPEC,                        ! Get Type Spec from Sep Type
;    85          0218  1     FIND_TYPREC_FROM_TSPEC,
;    86          0219  1     GET_BITSIZE_FROM_DTYPE,               ! Get the size from Dtype
;    87          0220  1     GET_BITSIZE_FROM_BLIDST;              ! Get the size from BLI DST
```

```
;    89          0221  1 EXTERNAL ROUTINE
;    90          0222  1     DBG$DATA_LENGTH,                ! Obtain bitsize from VMS descriptor
;    91          0223  1     DBG$GET_MEMORY,                 ! Get a permanent memory block
;    92          0224  1     DBG$GET_TEMPMEM,                ! Get a "temporary" memory block
;    93          0225  1     DBG$READ_ACCESS: NOVALUE,       !
;    94          0226  1     DBG$RST_DST_PTR,                ! Return the real DST pointer
;    95          0227  1     DBG$RST_NEXT_DST,               ! Return the next DST given a DST
;    96          0228  1     DBG$STA_SYMKIND,                ! Get a specified symbol's kind
;    97          0229  1     DBG$STA_SYMNAME: NOVALUE,       ! Obtain symbol name.
;    98          0230  1     DBG$STA_SYMVALUE: NOVALUE,      ! Evaluate a specified symbol's value
;    99          0231  1     DBG$STA_VALSPEC: NOVALUE;       ! Evaluate a DST Value Spec
;   100          0232  1
;   101          0233  1 EXTERNAL
;   102          0234  1     DST$BEGIN_ADDR,                 ! The start address of the DST in memory
;   103          0235  1     RST$START_ADDR: REF RST$ENTRY,  ! Pointer to RST anonymous module
;   104          0236  1     RST$TEMP_LIST;                  ! Pointer to Temporary RST Entry List
;   105          0237  1
;   106          0238  1 OWN
;   107          0239  1     NOVEL_LENGTH_FLAG;              ! This flag is used to indicate
;   108          0240  1                                     !   we have encounted a novel length
;   109          0241  1                                     !   dst, we need to take the size
;   110          0242  1                                     !   specificed in this dst to override
;   111          0243  1                                     !   the original length.  Especially,
;   112          0244  1                                     !   when a data has novel length dst,
;   113          0245  1                                     !   and this data item has a typeid,
;   114          0246  1                                     !   (typid has its own size), in
;   115          0247  1                                     !   this case we need to modify the
;   116          0248  1                                     !   typeid to reflect the novel
;   117          0249  1                                     !   length by making a copy of
;   118          0250  1                                     !   the typeid with new novel length
;   119          0251  1                                     !   for the data.
```

```
;  121     0252  1 GLOBAL ROUTINE DBG$FIND_SEPTYP(DST_PTR, TYPE_PTR, TYPE_SPEC) =
;  122     0253  1 !
;  123     0254  1 ! FUNCTION
;  124     0255  1 !       Given Separate Type Specification DST Record, Figure out its Type
;  125     0256  1 !       Specification and the size of the described object is returned.
;  126     0257  1 !
;  127     0258  1 ! INPUTS
;  128     0259  1 !
;  129     0260  1 !       DST_PTR         - Pointer to DST Record.
;  130     0261  1 !
;  131     0262  1 !       TYPE_PTR        - Pointer to DST Type Spec fields.
;  132     0263  1 !
;  133     0264  1 !       TYPE_SPEC       - Pointer to DST Type Spec.
;  134     0265  1 !
;  135     0266  1 ! OUTPUTS
;  136     0267  1 !       Size of the object in Type Spec is returned.
;  137     0268  1 !
;  138     0269  1
;  139     0270  2     BEGIN
;  140     0271  2
;  141     0272  2     MAP
;  142     0273  2         DST_PTR: REF DST$RECORD,          ! Pointer to DST Type Spec Record
;  143     0274  2         TYPE_PTR: REF VECTOR[,LONG],      ! Pointer to DST Type Spec fields
;  144     0275  2         TYPE_SPEC: REF VECTOR[,LONG];     ! Pointer to DST Type Spec
;  145     0276  2
;  146     0277  2     LOCAL
;  147     0278  2         ENUMBEG: REF DST$RECORD,          ! Pointer to Enum. begin record                    41
;  148     0279  2         RECBEG: REF DST$RECBEG_TRLR,      ! Pointer to Rec. begin trailer record             41
;  149     0280  2         NAMEPTR: REF VECTOR[,BYTE],       ! Pointer to name of the record
;  150     0281  2         TYPEPTR: REF DST$RECORD,          ! Pointer to DST Type Spec fields                  41
;  151     0282  2         TYPESPEC: REF DST$TYPE_SPEC,      ! Pointer to DST Type Spec
;  152     0283  2         SIZE;                             ! Size of the Object
;  153     0284  2
;  154     0285  2     SIZE = 0;
;  155     0286  2 !     TYPEPTR = DST_PTR[DST$A_NEXT] + .DST_PTR[DST$B_LENGTH];          ! D006
;  156     0287  2     TYPEPTR = DBG$RST_NEXT_DST( .DST_PTR );                          ! A006
;  157     0288  2     TYPESPEC =
;  158     0289  2         TYPEPTR[DST$A_TYPSPEC_TS_ADDR] + .TYPEPTR[DST$B_TYPSPEC_NAME];
;  159     0290  2
;  160     0291  2     WHILE TRUE DO
;  161     0292  3         BEGIN
;  162     0293  3         SELECTONE .TYPEPTR[DST$B_TYPE] OF
;  163     0294  3             SET
;  164     0295  3             [DST$K_SEPTYP]:
;  165     0296  4                 BEGIN
;  166     0297  4                 TYPEPTR = DBG$RST_NEXT_DST( .TYPEPTR );              ! M006
;  167     0298  4                 TYPESPEC = .TYPEPTR[DST$B_TYPSPEC_NAME] +
;  168     0299  4                         TYPEPTR[DST$A_TYPSPEC_TS_ADDR];
;  169     0300  3                 END;
;  170     0301  3
;  171     0302  3             [DST$K_GLOBNXT]:
;  172     0303  4                 BEGIN
;  173     0304  4 !               TYPEPTR = .TYPEPTR + 2;                              ! A006
;  174     0305  4                 TYPEPTR = DBG$RST_NEXT_DST( .TYPEPTR );              ! A006
;  175     0306  4                 TYPESPEC = .TYPEPTR[DST$B_TYPSPEC_NAME] +
;  176     0307  4                         TYPEPTR[DST$A_TYPSPEC_TS_ADDR];
;  177     0308  3                 END;
```

```
;  178          0309  3
;  179          0310  3          [DST$K_TYPSPEC]:
;  180          0311  4              BEGIN
;  181          0312  4              TYPESPEC = FIND_TYPESPEC(.TYPESPEC, SIZE, TYPEPTR);
;  182          0313  4              EXITLOOP;
;  183          0314  3              END;
;  184          0315  3
;  185          0316  3          [DST$K_ENUMBEG]:
;  186          0317  4              BEGIN
;  187          0318  4              ENUMBEG = .TYPEPTR;
;  188          0319  4              SIZE = .TYPEPTR[DST$B_ENUMBEG_LENG];
;  189          0320  4              EXITLOOP;
;  190          0321  3              END;
;  191          0322  3
;  192          0323  3          [DST$K_RECBEG]:
;  193          0324  4              BEGIN
;  194          0325  4
;  195          0326  4
;  196          0327  4              ! Advance the pointer to the name field.
;  197          0328  4              ! (1 byte length, 1 byte type, 1 byte vflags, 4 bytes value).
;  198          0329  4              !
;  199          0330  4              RECBEG = .TYPEPTR + 1 + 1 + 1 + 4;
;  200          0331  4              NAMEPTR = .RECBEG;
;  201          0332  4
;  202          0333  4              ! Advance the pointer to the trailer field.
;  203          0334  4              !
;  204          0335  4              RECBEG = .RECBEG + .NAMEPTR[0] + 1;
;  205          0336  4              SIZE = .RECBEG[DST$L_RECBEG_SIZE];
;  206          0337  4              EXITLOOP;
;  207          0338  3              END;
;  208          0339  3
;  209          0340  3          [OTHERWISE]:
;  210          0341  3              EXITLOOP;
;  211          0342  3
;  212          0343  3              TES;
;  213          0344  3
;  214          0345  2          END;                        ! End of WHILE Loop.
;  215          0346  2
;  216          0347  2      IF .SIZE EQL 0 THEN SIZE = DBG$GET_BITSIZE_FROM_TYPESPEC(.TYPESPEC);
;  217          0348  2      TYPE_PTR[0] = .TYPEPTR;
;  218          0349  2      TYPE_SPEC[0] = .TYPESPEC;
;  219          0350  2      RETURN .SIZE;
;  220          0351  1      END;


                                                        .TITLE  RSTTYPES
                                                        .IDENT  \V04-000\

                                                        .PSECT  DBG$OWN,NOEXE,  PIC,2

                                                00000 NOVEL_LENGTH_FLAG:
                                                        .BLKB   4

                                                        .EXTRN  DBG$DATA_LENGTH
                                                        .EXTRN  DBG$GET_MEMORY, DBG$GET_TEMPMEM
                                                        .EXTRN  DBG$READ_ACCESS
                                                        .EXTRN  DBG$RST_DST_PTR
```

```
                                        .EXTRN  DBG$RST_NEXT_DST
                                        .EXTRN  DBG$STA_SYMKIND
                                        .EXTRN  DBG$STA_SYMNAME
                                        .EXTRN  DBG$STA_SYMVALUE
                                        .EXTRN  DBG$STA_VALSPEC
                                        .EXTRN  DST$BEGIN_ADDR, RST$START_ADDR
                                        .EXTRN  RST$TEMP_LIST

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

                     01FC 00000         .ENTRY  DBG$FIND_SEPTYP, Save R2,R3,R4,R5,R6,R7,R8  ; 0252
    58 00000000G      00  9E 00002      MOVAB   DBG$RST_NEXT_DST, R8                        ;
    5E                08  C2 00009      SUBL2   #8, SP                                      ;
                04    AE  D4 0000C      CLRL    SIZE                                        ; 0285
                04    AC  DD 0000F      PUSHL   DST_PTR                                     ; 0287
    68                01  FB 00012      CALLS   #1, DBG$RST_NEXT_DST                        ;
    6E                50  D0 00015      MOVL    R0, TYPEPTR                                 ;
    51                6E  D0 00018      MOVL    TYPEPTR, R1                                 ; 0289
    50          02    A1  9A 0001B      MOVZBL  2(R1), R0                                   ;
    55          03 A140  9E 0001F      MOVAB   3(R1)[R0], TYPESPEC                         ;
    53                6E  D0 00024 1$:  MOVL    TYPEPTR, R3                                 ; 0293
    54          01    A3  9A 00027      MOVZBL  1(R3), R4                                   ;
A3  8F                54  91 0002B      CMPB    R4, #^163                                   ; 0295
                      06  13 0002F      BEQL    2$                                          ;
A0  8F                54  91 00031      CMPB    R4, #160                                    ; 0302
                      15  12 00035      BNEQ    3$                                          ;
                      53  DD 00037 2$:  PUSHL   R3                                          ; 0305
    68                01  FB 00039      CALLS   #1, DBG$RST_NEXT_DST                        ;
    6E                50  D0 0003C      MOVL    R0, TYPEPTR                                 ;
    51          02    A0  9A 0003F      MOVZBL  2(R0), R1                                   ; 0307
    50                51  C0 00043      ADDL2   R1, R0                                      ;
    55          03    A0  9E 00046      MOVAB   3(R0), TYPESPEC                             ; 0306
                      D8  11 0004A      BRB     1$                                          ; 0293
AF  8F                54  91 0004C 3$:  CMPB    R4, #175                                    ; 0310
                      11  12 00050      BNEQ    4$                                          ;
                      5E  DD 00052      PUSHL   SP                                          ; 0312
                08    AE  9F 00054      PUSHAB  SIZE                                        ;
                      55  DD 00057      PUSHL   TYPESPEC                                    ;
0000V CF              03  FB 00059      CALLS   #3, FIND_TYPESPEC                           ;
    55                50  D0 0005E      MOVL    R0, TYPESPEC                                ;
                      29  11 00061      BRB     6$                                          ; 0311
A5  8F                54  91 00063 4$:  CMPB    R4, #165                                    ; 0316
                      0A  12 00067      BNEQ    5$                                          ;
    57                53  D0 00069      MOVL    R3, ENUMBEG                                 ; 0318
04  AE          02    A3  9A 0006C      MOVZBL  2(R3), SIZE                                 ; 0319
                      19  11 00071      BRB     6$                                          ; 0317
AB  8F                54  91 00073 5$:  CMPB    R4, #171                                    ; 0323
                      13  12 00077      BNEQ    6$                                          ;
    52          07    A3  9E 00079      MOVAB   7(R3), RECBEG                               ; 0330
    56                52  D0 0007D      MOVL    RECBEG, NAMEPTR                             ; 0331
    50                66  9A 00080      MOVZBL  (NAMEPTR), R0                               ; 0335
    52          01 A042  9E 00083      MOVAB   1(R0)[RECBEG], RECBEG                       ;
04  AE                62  D0 00088      MOVL    (RECBEG), SIZE                              ; 0336
                04    AE  D5 0008C 6$:  TSTL    SIZE                                        ; 0347
                      0B  12 0008F      BNEQ    7$                                          ;
                      55  DD 00091      PUSHL   TYPESPEC                                    ;
0000V CF              01  FB 00093      CALLS   #1, DBG$GET_BITSIZE_FROM_TYPESPEC           ;
```

```
                04   AE          50  D0 00098          MOVL    R0, SIZE                                ;
                08   BC          6E  D0 0009C 7$:      MOVL    TYPEPTR, @TYPE_PTR                      ; 0348
                0C   BC          55  D0 000A0          MOVL    TYPESPEC, @TYPE_SPEC                    ; 0349
                     50      04  AE  D0 000A4          MOVL    SIZE, R0                                ; 0350
                                      04 000A8          RET                                             ; 0351

; Routine Size:  169 bytes,    Routine Base:  DBG$CODE + 0000

;  221          0352  1
```

```
;  223          0353  1 GLOBAL ROUTINE DBG$STA_SYMTYPE(SYMID, FCODE, TYPEID): NOVALUE =
;  224          0354  1 !
;  225          0355  1 ! FUNCTION
;  226          0356  1 !       This routine takes a SYMID and returns a TYPEID and FCODE for the
;  227          0357  1 !       type associated with the input SYMID.  In the case where the SYMID
;  228          0358  1 !       defines a type item, the returned TYPEID is identical to the SYMID.
;  229          0359  1 !
;  230          0360  1 ! INPUTS
;  231          0361  1 !       SYMID   - The SYMID for the symbol or type whose TYPEID and  Format Code
;  232          0362  1 !                 are to be returned.
;  233          0363  1 !
;  234          0364  1 !       FCODE   - The address of a longword location to receive the FCODE
;  235          0365  1 !                 associated with the given SYMID.
;  236          0366  1 !
;  237          0367  1 !       TYPEID  - The address of a longword location to receive the
;  238          0368  1 !                 TYPEID describing the type associated with SYMID.
;  239          0369  1 !
;  240          0370  1 ! OUTPUTS
;  241          0371  1 !       FCODE   - The format code associated with SYMID is returned to FCODE.
;  242          0372  1 !
;  243          0373  1 !       TYPEID  - A TYPEID defining the type associated with the input
;  244          0374  1 !                 SYMID is returned to TYPEID.
;  245          0375  1 !
;  246          0376  1 !
;  247          0377  1
;  248          0378  2     BEGIN
;  249          0379  2
;  250          0380  2     MAP
;  251          0381  2         SYMID: REF RST$ENTRY,           ! Pointer to input symbol's RST entry
;  252          0382  2         FCODE: REF VECTOR[1],           ! Address where FCODE is returned
;  253          0383  2         TYPEID: REF VECTOR[1            ! Address where TYPEID is returned
;  254          0384  2
;  255          0385  2     LOCAL
;  256          0386  2         SIZE;                           ! Bit size of a data item of the type
;  257          0387  2
;  258          0388  2
;  259          0389  2
;  260          0390  2     ! Find the Type RST Entry associated with the input SYMID.  Return its
;  261          0391  2     ! address to TYPEID[0] and its Format Code to FCODE[0].  Signal an internal
;  262          0392  2     ! error if the TYPEID turns out to be zero--that should never happen.
;  263          0393  2     !
;  264          0394  2     TYPEID[0] = FIND_TYPE_RECORD(.SYMID, FCODE[0], SIZE);
;  265          0395  2     IF .TYPEID[0] EQL 0
;  266          0396  2     THEN
;  267          0397  2         $DBG_ERROR('RSTTYPES\SYMTYPE');
;  268          0398  2
;  269          0399  2     RETURN;
;  270          0400  2
;  271          0401  1     END;


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

59  54  4D  59  53  5C  53  45  50  59  54  54  53  52  10  00000 P.AAA:  .ASCII  <16>\RSTTYPES\<92>\SYMTYPE\
                                                    45  50  0000F
```

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

                                                0000 00000      .ENTRY  DBG$STA_SYMTYPE, Save nothing                   ; 0353
                                     5E         04   C2 00002      SUBL2   #4, SP                                      ;
                                                5E   DD 00005      PUSHL   SP                                          ; 0394
                                     7E    04   AC   7D 00007      MOVQ    SYMID, -(SP)                                ;
                                0000V CF         03   FB 0000B      CALLS   #3, FIND_TYPE_RECORD                        ;
                                  0C  BC         50   D0 00010      MOVL    R0, @TYPEID                                 ;
                                                15   12 00014      BNEQ    1$                                          ; 0395
                                      00000000' EF   9F 00016      PUSHAB  P.AAA                                       ; 0397
                                                01   DD 0001C      PUSHL   #1                                          ;
                                      00028362  8F   DD 0001E      PUSHL   #164706                                     ;
                            00000000G   00      03   FB 00024      CALLS   #3, LIB$SIGNAL                              ;
                                                04 0002B 1$:    RET                                                     ; 0401

; Routine Size:  44 bytes,    Routine Base:  DBG$CODE + 00A9
```

```
;  273     0402  1 GLOBAL ROUTINE DBG$STA_SYMSIZE(SYMID, SIZE): NOVALUE =
;  274     0403  1 !
;  275     0404  1 ! FUNCTION
;  276     0405  1 !       This routine takes a SYMID and returns the size in bits associated
;  277     0406  1 !       with that input SYMID.
;  278     0407  1 !
;  279     0408  1 ! INPUTS
;  280     0409  1 !       SYMID   - The SYMID for the symbol or type whose bitsize is to be
;  281     0410  1 !                 returned.
;  282     0411  1 !
;  283     0412  1 !       SIZE    - The address of a longword location to receive the size in bits
;  284     0413  1 !                 of the item described by the given SYMID.
;  285     0414  1 !
;  286     0415  1 ! OUTPUTS
;  287     0416  1 !       SIZE    - The size in bits of the input item is returned to SIZE.
;  288     0417  1 !
;  289     0418  1 !       No value is returned.
;  290     0419  1 !
;  291     0420  1
;  292     0421  2     BEGIN
;  293     0422  2
;  294     0423  2     MAP
;  295     0424  2         SYMID: REF RST$ENTRY,           ! Pointer to input symbol's RST entry
;  296     0425  2         SIZE: REF VECTOR[1];            ! Address where size is to be returned
;  297     0426  2
;  298     0427  2     LOCAL
;  299     0428  2         TYPEID: REF RST$ENTRY,          ! The symbol's Type ID
;  300     0429  2         FCODE;                          ! The symbol's Type Format Code
;  301     0430  2
;  302     0431  2
;  303     0432  2
;  304     0433  2     ! Get to the Type DST record and pick up the size information.  If no type
;  305     0434  2     ! info was found, set the bit size to zero (wich means size is unknown).
;  306     0435  2     !
;  307     0436  2     TYPEID = FIND_TYPE_RECORD(.SYMID, FCODE, SIZE[0]);
;  308     0437  2     IF .TYPEID EQL 0 THEN SIZE[0] = 0
;  309     0438  2     ELSE IF (.SIZE[0] EQL 0) AND
;  310     0439  3       ((.FCODE EQL RST$K_TYPE_PTR) OR (.FCODE EQL RST$K_TYPE_TPTR))
;  311     0440  2         THEN SIZE[0] = 32;
;  312     0441  2     RETURN;
;  313     0442  1     END;

                                0000 00000          .ENTRY  DBG$STA_SYMSIZE, Save nothing      ; 0402
                         5E     04   C2 00002          SUBL2   #4, SP
                      08 AC     DD 00005          PUSHL   SIZE                               ; 0436
                      04 AE     9F 00008          PUSHAB  FCODE
                      04 AC     DD 0000B          PUSHL   SYMID
                0000V CF        03 FB 0000E          CALLS   #3, FIND_TYPE_RECORD
                                50 D5 00013          TSTL    TYPEID                             ; 0437
                                04 12 00015          BNEQ    1$
                      08 BC     D4 00017          CLRL    @SIZE
                                04 0001A          RET
                      08 BC     D5 0001B 1$:      TSTL    @SIZE                              ; 0438
```

```
                     0E  12 0001E             BNEQ    3$                                      ;
                10   6E  D1 00020             CMPL    FCODE, #16                              ; 0439
                     05  13 00023             BEQL    2$                                      ;
                06   6E  D1 00025             CMPL    FCODE, #6                               ;
                     04  12 00028             BNEQ    3$                                      ;
             08  BC  20  D0 0002A 2$:          MOVL    #32, @SIZE                              ; 0440
                     04  0002E 3$:             RET                                             ; 0442

; Routine Size:  47 bytes,    Routine Base:  DBG$CODE + 00D5
```

```
;  315          0443  1 GLOBAL ROUTINE DBG$STA_TYPEFCODE(SYMID) =
;  316          0444  1 !
;  317          0445  1 ! FUNCTION
;  318          0446  1 !       This routine returns the Format Code (FCODE) associated with a specified
;  319          0447  1 !       type (given by a SYMID or TYPEID).  The Format Code is simply picked up
;  320          0448  1 !       from the type's RST entry.
;  321          0449  1 !
;  322          0450  1 ! INPUTS
;  323          0451  1 !       SYMID   - The SYMID for the symbol or type whose type Format Code
;  324          0452  1 !                 is to be returned.
;  325          0453  1 !
;  326          0454  1 ! OUTPUTS
;  327          0455  1 !       The type's Format Code is returned as the routine value.  The possible
;  328          0456  1 !                 Format Codes are listed in DBGLIB.REQ.
;  329          0457  1 !
;  330          0458  1 !
;  331          0459  1
;  332          0460  2     BEGIN
;  333          0461  2
;  334          0462  2     LOCAL
;  335          0463  2         TYPEID,                         ! The symbol's Type ID
;  336          0464  2         FCODE;                          ! The symbol's type Format Code
;  337          0465  2
;  338          0466  2
;  339          0467  2
;  340          0468  2     ! Get the symbol's type and Format Code.  Then return the type Format Code.
;  341          0469  2     !
;  342          0470  2     DBG$STA_SYMTYPE(.SYMID, FCODE, TYPEID);
;  343          0471  2     RETURN .FCODE;
;  344          0472  2
;  345          0473  1     END;


                                        0000 00000      .ENTRY  DBG$STA_TYPEFCODE, Save nothing         ; 0443
                                    5E  08   C2 00002   SUBL2   #8, SP                                  ;
                                        5E   DD 00005   PUSHL   SP                                      ; 0470
                                08      AE   9F 00007   PUSHAB  FCODE                                   ;
                                04      AC   DD 0000A   PUSHL   SYMID                                   ;
                            94  AF          03   FB 0000D   CALLS   #3, DBG$STA_SYMTYPE                 ;
                                50      04  AE   D0 00011   MOVL    FCODE, R0                           ; 0471
                                            04   00015   RET                                           ; 0473

; Routine Size:  22 bytes,    Routine Base:  DBG$CODE + 0104
```

```
;  347     0474  1 GLOBAL ROUTINE DBG$STA_TYP_ARRAY(TYPEID, DSCADDR, CELLTYPE,
;  348     0475  1                                   NDIMS, DIMVECPTR, BITSIZE): NOVALUE =
;  349     0476  1 !
;  350     0477  1 ! FUNCTION
;  351     0478  1 !       This routine returns the type information associated with an Array data
;  352     0479  1 !       type.  The type of interest is identified by a Type ID, and the returned
;  353     0480  1 !       information includes the array descriptor, the cell type, the number of
;  354     0481  1 !       dimensions, the size in bits of the array, and information about each
;  355     0482  1 !       specific dimension (the subscript type, the lower bound, and the upper
;  356     0483  1 !       bound).
;  357     0484  1 !
;  358     0485  1 ! INPUTS
;  359     0486  1 !       TYPEID  - The Type ID of the Array data type about which information is
;  360     0487  1 !                 desired.  Its Format Code must be RST$K_TYPE_ARRAY.
;  361     0488  1 !
;  362     0489  1 !       DSCADDR - The address of a longword location to receive a pointer to the
;  363     0490  1 !                 array's descriptor.
;  364     0491  1 !
;  365     0492  1 !       CELLTYPE - The address of a longword location to receive the Type ID of
;  366     0493  1 !                  the data type of the individual array elements.
;  367     0494  1 !
;  368     0495  1 !       NDIMS   - The address of a longword location to receive the number of
;  369     0496  1 !                 dimensions in the array.
;  370     0497  1 !
;  371     0498  1 !       DIMVECPTR - The address of a longword location to receive a pointer to a
;  372     0499  1 !                   vector of subscript information.
;  373     0500  1 !
;  374     0501  1 !       BITSIZE - The address of a longword location to receive the size in bits
;  375     0502  1 !                 of an array of this type.
;  376     0503  1 !
;  377     0504  1 ! OUTPUTS
;  378     0505  1 !       DSCADDR - The address of the array descriptor is returned to DSCADDR.
;  379     0506  1 !                 This may be a contiguous or noncontiguous array descriptor.
;  380     0507  1 !                 The descriptor may disappear at the end of the current Debug
;  381     0508  1 !                 command.
;  382     0509  1 !
;  383     0510  1 !       CELLTYPE - The Type ID of the individual array elements' data type is
;  384     0511  1 !                  returned to CELLTYPE.
;  385     0512  1 !
;  386     0513  1 !       NDIMS   - The number of array dimensions is returned to NDIMS.
;  387     0514  1 !
;  388     0515  1 !       DIMVECPTR - A pointer to a subscript vector is returned to
;  389     0516  1 !                   DIMVECPTR.  This vector contains one longword
;  390     0517  1 !                   per dimension, giving the subscript Type ID.
;  391     0518  1 !                   The vector is allocated in temporary storage--it disappears
;  392     0519  1 !                   at the end of the current Debug command.
;  393     0520  1 !
;  394     0521  1 !       BITSIZE - The size in bits is returned to BITSIZE.
;  395     0522  1 !
;  396     0523  1 !       No value is returned.
;  397     0524  1 !
;  398     0525  1 ! WARNING:
;  399     0526  1 !       The five output parameters must be distinct locations in the
;  400     0527  1 !       callers address space. For example, you cannot do:
;  401     0528  1 !       DBG$STA_TYP_ARRAY (.TYPEID, JUNK, CELLTYPE, JUNK, JUNK, JUNK)
;  402     0529  1 !       to just find out the celltype. The problem is that the routine
;  403     0530  1 !       is using the addresses passed in (.JUNK in this case) as local
```

```
;  404     0531  1 !       storage, which it really should not be doing, and stores into
;  405     0532  1 !       one .JUNK will trip up reads from another. So you must pass
;  406     0533  1 !       in distinct parameters,
;  407     0534  1 !       DBG$STA_TYP_ARRAY (.TYPEID, JUNK1, CELLTYPE, JUNK2, JUNK3, JUNK4)
;  408     0535  1 !       I got tripped up on this so I thought I'd warn others.
;  409     0536  1 !       R. Title Nov 1982.
;  410     0537  1 !
;  411     0538  1
;  412     0539  2     BEGIN
;  413     0540  2
;  414     0541  2     MAP
;  415     0542  2         TYPEID : REF RST$ENTRY;          ! The input TYPEID for the array type
;  416     0543  2
;  417     0544  2     LOCAL
;  418     0545  2         I,
;  419     0546  2         BITS_USED,
;  420     0547  2         DATA_TYPE,
;  421     0548  2         SIZE,
;  422     0549  2         FCODE,
;  423     0550  2         VALKIND,
;  424     0551  2         TYPE_SPEC : REF DST$TYPE_SPEC,
;  425     0552  2         TYPE_SPEC1 : REF DST$TYPE_SPEC,
;  426     0553  2         DST_PTR : REF DST$RECORD,
;  427     0554  2         TYPEID_PTR : REF RST$ENTRY,
;  428     0555  2         DESC_PTR : REF BLOCK[,BYTE],
;  429     0556  2         SUBSCR_VECT : REF VECTOR,
;  430     0557  2         VAL_VECT : VECTOR[3],
;  431     0558  2         VAL_PTR : REF DST$VAL_SPEC;
;  432     0559  2
;  433     0560  2
;  434     0561  2
;  435     0562  2     TYPEID_PTR = FIND_TYPE_RECORD(.TYPEID, FCODE, SIZE);
;  436     0563  2
;  437     0564  2
;  438     0565  2     ! TYPEID_PTR -> RST Entry for this array type.  Verify the format code.
;  439     0566  2     !
;  440     0567  2     IF .FCODE NEQ RST$K_TYPE_ARRAY THEN $DBG_ERROR('RSTTYPES\TYP_ARRAY 10');
;  441     0568  2     DST_PTR = .TYPEID_PTR[RST$L_DSTPTR];
;  442     0569  2     TYPE_SPEC = DST_PTR[DST$A_TYPSPEC_TS_ADDR] + .DST_PTR[DST$B_TYPSPEC_NAME];
;  443     0570  2     TYPE_SPEC = FIND_TYPESPEC(.TYPE_SPEC, SIZE, DST_PTR);
;  444     0571  2
;  445     0572  2
;  446     0573  2     ! For array, Typeid does not carray the size information.  However
;  447     0574  2     ! FIND_TYPESPEC routine does go further into the DST to dig out
;  448     0575  2     ! the size of the array for contiguous array.
;  449     0576  2     !
;  450     0577  2     .BITSIZE = .SIZE;
;  451     0578  2
;  452     0579  2
;  453     0580  2     ! TYPE_SPEC -> Type Specification field for this array type.
;  454     0581  2     ! If old format array spec, get info from descriptor
;  455     0582  2     !
;  456     0583  2     IF .TYPE_SPEC[DST$B_TS_KIND] EQL DST$K_TS_DSC
;  457     0584  2     THEN
;  458     0585  3         BEGIN
;  459     0586  3
;  460     0587  3         ! Obtain a descriptor by evaluating the value specification.
```

```
;  461     0588  3          !
;  462     0589  3          VAL_PTR = TYPE_SPEC[DST$A_TS_DSC_VSPEC_ADDR];
;  463     0590  3          DBG$STA_VALSPEC(.VAL_PTR, VAL_VECT, VALKIND);
;  464     0591  3          IF .VALKIND NEQ DBG$K_VAL_DESCR
;  465     0592  3          THEN
;  466     0593  3              $DBG_ERROR('RSTTYPES\TYP_ARRAY 20');
;  467     0594  3          DESC_PTR = .VAL_VECT[0];
;  468     0595  3          DBG$READ_ACCESST.DESC_PTR, 12);
;  469     0596  3          .DSCADDR = .DESC_PTR;
;  470     0597  3
;  471     0598  3          ! Obtain the dtype of an element and the bitsize of an element.
;  472     0599  3          ! Note that DBG$DATA_LENGTH returns the bitsize of an element
;  473     0600  3          ! because it just looks at the dtype field and the length field,
;  474     0601  3          ! not the class field.
;  475     0602  3          !
;  476     0603  3          DATA_TYPE = .DESC_PTR[DSC$B_DTYPE];
;  477     0604  3          BITS_USED = DBG$DATA_LENGTH(.DESC_PTR);
;  478     0605  3
;  479     0606  3          ! Build a typeid for the array element.
;  480     0607  3          !
;  481     0608  3          IF (.BITS_USED NEQ 0)              AND
;  482     0609  3             (.DESC_PTR[DSC$B_SCALE] EQL 0) AND
;  483     0610  4             (.DESC_PTR[DSC$B_DIGITS] EQL 0)
;  484     0611  3          THEN
;  485     0612  3              .CELLTYPE = DBG$TYPEID_FOR_ATOMIC(.DATA_TYPE,.BITS_USED,TRUE)
;  486     0613  3
;  487     0614  3          ELSE
;  488     0615  3              .CELLTYPE = TYPEID_FOR_DESCR(.DESC_PTR,RST$K_TYPE_DESCR,TRUE);
;  489     0616  3
;  490     0617  3          ! Use the descriptor to fill in the remaining output parameters.
;  491     0618  3          !
;  492     0619  3          .NDIMS = .DESC_PTR[DSC$B_DIMCT];
;  493     0620  3          SUBSCR_VECT = DBG$GET_TEMPMEM(..NDIMS);
;  494     0621  3          .DIMVECPTR = .SUBSCR_VECT;
;  495     0622  3          INCR I FROM 0 TO ..NDIMS - 1 DO
;  496     0623  3              SUBSCR_VECT[.I] = 0;
;  497     0624  3          END
;  498     0625  3
;  499     0626  3
;  500     0627  3      ! New format type specification for this array.
;  501     0628  3      !
;  502     0629  2      ELSE
;  503     0630  3          BEGIN
;  504     0631  3          IF .TYPE_SPEC[DST$B_TS_KIND] EQL DST$K_TS_ARRAY
;  505     0632  3          THEN
;  506     0633  4              BEGIN
;  507     0634  4              LOCAL
;  508     0635  4                  FLAGS_PTR : REF BITVECTOR;
;  509     0636  4
;  510     0637  4              .NDIMS = .TYPE_SPEC[DST$B_TS_ARRAY_DIM];
;  511     0638  4              FLAGS_PTR = TYPE_SPEC[DST$A_TS_ARRAY_FLAGS_ADDR];
;  512     0639  4
;  513     0640  4
;  514     0641  4              ! FLAGS_PTR -> Bitvector of optional Type-Spec indicators.
;  515     0642  4              !
;  516     0643  4              VAL_PTR = .FLAGS_PTR + (..NDIMS/8) + 1;
;  517     0644  4              DBG$STA_VALSPEC(.VAL_PTR, VAL_VECT, VALKIND);
```

```
;  518     0645  4          IF .VALKIND NEQ DBG$K_VAL_DESCR
;  519     0646  4          THEN
;  520     0647  4              $DBG_ERROR('RSTTYPES\TYP_ARRAY 30');
;  521     0648  4
;  522     0649  4          DESC_PTR = .VAL_VECT[0];
;  523     0650  4          .DSCADDR = .DESC_PTR;
;  524     0651  4          IF .VAL_PTR[DST$B_VS_VFLAGS] EQL DST$K_VS_FOLLOWS
;  525     0652  4          THEN
;  526     0653  4              TYPE_SPEC1 = .VAL_PTR + .VAL_PTR[DST$W_VS_LENGTH] + 3
;  527     0654  4
;  528     0655  4          ELSE
;  529     0656  4              TYPE_SPEC1 = .VAL_PTR + 5;
;  530     0657  4
;  531     0658  4
;  532     0659  4          ! TYPE_SPEC1 -> First optional Type Spec within this record.
;  533     0660  4          ! TYPE_SPEC1 will move along through the optional Type Specs.
;  534     0661  4          !
;  535     0662  4          IF .FLAGS_PTR[0]
;  536     0663  4          THEN
;  537     0664  5              BEGIN
;  538     0665  5              ! Optional cell type is present.
;  539     0666  5              .CELLTYPE = FIND_TYPREC_FROM_TSPEC(.TYPE_SPEC1);
;  540     0667  5              TYPE_SPEC1 = .TYPE_SPEC1 + 2 + .TYPE_SPEC1[DST$W_TS_LENGTH];
;  541     0668  5              END                                                              ;
;  542     0669  5
;  543     0670  4          ELSE
;  544     0671  5              BEGIN
;  545     0672  5              ! no optional cell type; use cell type from descriptor.
;  546     0673  5              DBG$READ_ACCESS(.DESC_PTR, 12);
;  547     0674  5
;  548     0675  5              ! Obtain the dtype of an element and the bitsize of an element.
;  549     0676  5              ! Note that DBG$DATA_LENGTH returns the bitsize of an element
;  550     0677  5              ! because it just looks at the dtype field and the length field,
;  551     0678  5              ! not the class field.
;  552     0679  5              !
;  553     0680  5              DATA_TYPE = .DESC_PTR[DSC$B_DTYPE];
;  554     0681  5              BITS_USED = DBG$DATA_LENGTH(.DESC_PTR);
;  555     0682  5
;  556     0683  5              IF (.BITS_USED NEQ 0)              AND
;  557     0684  5                 (.DESC_PTR[DSC$B_SCALE] EQL 0) AND
;  558     0685  6                 (.DESC_PTR[DSC$B_DIGITS] EQL 0)
;  559     0686  5              THEN
;  560     0687  5                  .CELLTYPE = DBG$TYPEID_FOR_ATOMIC(.DATA_TYPE,.BITS_USED,TRUE)
;  561     0688  5
;  562     0689  5              ELSE
;  563     0690  5                  .CELLTYPE = TYPEID_FOR_DESCR(.DESC_PTR,RST$K_TYPE_DESCR,TRUE);
;  564     0691  5
;  565     0692  4              END;
;  566     0693  4
;  567     0694  4
;  568     0695  4          ! Now build DIMVECPTR.  Each element receives either a TYPEID or
;  569     0696  4          ! zero, depending on the appropriate bit in the flags vector.
;  570     0697  4          !
;  571     0698  4          SUBSCR_VECT = DBG$GET_TEMPMEM(..NDIMS);
;  572     0699  4          .DIMVECPTR = .SUBSCR_VECT;
;  573     0700  4          INCR I FROM 1 TO ..NDIMS DO
;  574     0701  5              BEGIN
```

```
;  575          0702  5                IF .FLAGS_PTR[.I]
;  576          0703  5                THEN
;  577          0704  6                    BEGIN
;  578          0705  6                    ! Optional subscript type exists; type_spec1 points to it.
;  579          0706  6                    SUBSCR_VECT[.I - 1] = FIND_TYPREC_FROM_TSPEC(.TYPE_SPEC1);
;  580          0707  6                    TYPE_SPEC1 = .TYPE_SPEC1 + 2 + .TYPE_SPEC1[DST$W_TS_LENGTH];
;  581          0708  6                    END
;  582          0709  6
;  583          0710  5                ELSE
;  584          0711  5                    SUBSCR_VECT[.I-1] = 0;
;  585          0712  5
;  586          0713  4                END;
;  587          0714  4
;  588          0715  3            END;
;  589          0716  3
;  590          0717  2        END;
;  591          0718  2
;  592          0719  2    RETURNS;
;  593          0720  2
;  594          0721  1    END;


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

41  5F  50  59  54  5C  53  45  50  59  54  54  53  52  15  00011 P.AAB:  .ASCII  <21>\RSTTYPES\<92>\TYP_ARRAY 10\          ;
                             30  31  20  59  41  52  52  00020
41  5F  50  59  54  5C  53  45  50  59  54  54  53  52  15  00027 P.AAC:  .ASCII  <21>\RSTTYPES\<92>\TYP_ARRAY 20\          ;
                             30  32  20  59  41  52  52  00036
41  5F  50  59  54  5C  53  45  50  59  54  54  53  52  15  0003D P.AAD:  .ASCII  <21>\RSTTYPES\<92>\TYP_ARRAY 30\          ;
                             30  33  20  59  41  52  52  0004C



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

                                    0FFC 00000          .ENTRY  DBG$STA_TYP_ARRAY, Save R2,R3,R4,R5,R6,R7,-  ; 0474
                                                                R8,R9,RT0,RT1
                5B 00000000G    00  9E 00002          MOVAB   DBG$READ_ACCESS, R11                          ;
                5A 00000000G    00  9E 00009          MOVAB   DBG$STA_VALSPEC, R10                          ;
                59 00000000G    00  9E 00010          MOVAB   LIB$SIGNAL, R9                                ;
                58 00000000'    EF  9E 00017          MOVAB   P.AAB, R8                                     ;
                5E              1C  C2 0001E          SUBL2   #28, SP                                       ;
                         08     AE  9F 00021          PUSHAB  SIZE                                          ; 0562
                         04     AE  9F 00024          PUSHAB  FCODE                                         ;
                         04     AC  DD 00027          PUSHL   TYPEID                                        ;
          0000V CF              03  FB 0002A          CALLS   #3, FIND_TYPE_RECORD                          ;
                52              50  D0 0002F          MOVL    R0, TYPEID_PTR                                ;
                01              6E  D1 00032          CMPL    FCODE, #1                                     ; 0567
                                0D  13 00035          BEQL    1$                                            ;
                                58  DD 00037          PUSHL   R8                                            ;
                                01  DD 00039          PUSHL   #1                                            ;
                   00028362     8F  DD 0003B          PUSHL   #164706                                       ;
                69              03  FB 00041          CALLS   #3, LIB$SIGNAL                                ;
          04    AE        0C    A2  D0 00044 1$:      MOVL    12(TYPEID_PTR), DST_PTR                       ; 0568
                51        04    AE  D0 00049          MOVL    DST_PTR, R1                                   ; 0569
                50        02    A1  9A 0004D          MOVZBL  2(RT), R0                                     ;
```

```
                52        03 A140  9E 00051          MOVAB   3(R1)[R0], TYPE_SPEC                    ;
                04          AE     9F 00056          PUSHAB  DST_PTR                                 ; 0570
                0C          AE     9F 00059          PUSHAB  SIZE                                    ;
                            52     DD 0005C          PUSHL   TYPE_SPEC                               ;
          0000V CF                 03 FB 0005E          CALLS   #3, FIND_TYPESPEC                    ;
                52                 50 D0 00063          MOVL    R0, TYPE_SPEC                       ;
           18   BC          08     AE D0 00066          MOVL    SIZE, @BITSIZE                      ; 0577
                02          02     A2 91 0006B          CMPB    2(TYPE_SPEC), #2                    ; 0583
                                   03 13 0006F          BEQL    2$                                  ;
                                008D 31 00071          BRW     8$                                  ;
                53          03     A2 9E 00074 2$:      MOVAB   3(R2), VAL_PTR                      ; 0589
                0C          AE     9F 00078            PUSHAB  VALKIND                             ; 0590
                14          AE     9F 0007B            PUSHAB  VAL_VECT                            ;
                            53     DD 0007E            PUSHL   VAL_PTR                             ;
                6A                 03 FB 00080          CALLS   #3, DBG$STA_VALSPEC                 ;
                03          0C     AE D1 00083          CMPL    VALKIND, #3                         ; 0591
                                   0E 13 00087          BEQL    3$                                  ;
                16          A8     9F 00089          PUSHAB  P.AAC                               ; 0593
                            01     DD 0008C          PUSHL   #1                                  ;
                00028362    8F     DD 0008E          PUSHL   #164706                             ;
                69                 03 FB 00094          CALLS   #3, LIB$SIGNAL                      ;
                54          10     AE D0 00097 3$:      MOVL    VAL_VECT, DESC_PTR                  ; 0594
                            0C     DD 0009B          PUSHL   #12                                 ; 0595
                            54     DD 0009D          PUSHL   DESC_PTR                            ;
                6B                 02 FB 0009F          CALLS   #2, DBG$READ_ACCESS                 ;
           08   BC                 54 D0 000A2          MOVL    DESC_PTR, @DSCADDR                  ; 0596
                56          02     A4 9A 000A6          MOVZBL  2(DESC_PTR), DATA_TYPE              ; 0603
                            54     DD 000AA          PUSHL   DESC_PTR                            ; 0604
  00000000G     00                 01 FB 000AC          CALLS   #1, DBG$DATA_LENGTH                 ;
                57                 50 D0 000B3          MOVL    R0, BITS_USED                       ;
                                   16 13 000B6          BEQL    4$                                  ; 0608
                            08     A4 95 000B8          TSTB    8(DESC_PTR)                         ; 0609
                                   11 12 000BB          BNEQ    4$                                  ;
                            09     A4 95 000BD          TSTB    9(DESC_PTR)                         ; 0610
                                   0C 12 000C0          BNEQ    4$                                  ;
                                   01 DD 000C2          PUSHL   #1                                  ; 0612
                7E                 56 7D 000C4          MOVQ    DATA_TYPE, -(SP)                    ;
          0000V CF                 03 FB 000C7          CALLS   #3, DBG$TYPEID_FOR_ATOMIC           ;
                                   0B 11 000CC          BRB     5$                                  ;
                                   01 DD 000CE 4$:      PUSHL   #1                                  ; 0615
                                   03 DD 000D0          PUSHL   #3                                  ;
                                   54 DD 000D2          PUSHL   DESC_PTR                            ;
          0000V CF                 03 FB 000D4          CALLS   #3, TYPEID_FOR_DESCR                ;
            0C  BC                 50 D0 000D9 5$:      MOVL    R0, @CELLTYPE                       ;
            10  BC          0B     A4 9A 000DD          MOVZBL  11(DESC_PTR), @NDIMS                ; 0619
                           10  BC  DD 000E2          PUSHL   @NDIMS                              ; 0620
    00000000G   00                 01 FB 000E5          CALLS   #1, DBG$GET_TEMPMEM                 ;
                53                 50 D0 000EC          MOVL    R0, SUBSCR_VECT                     ;
            14  BC                 53 D0 000EF          MOVL    SUBSCR_VECT, @DIMVECPTR             ; 0621
                50                 01 CE 000F3          MNEGL   #1, I                               ; 0622
                                   03 11 000F6          BRB     7$                                  ;
                                 6340 D4 000F8 6$:      CLRL    (SUBSCR_VECT)[I]                    ; 0623
          F8    50          10     BC F2 000FB 7$:      AOBLSS  @NDIMS, I, 6$                       ;
                                   04 00100             RET                                         ; 0583
                07          02     A2 91 00101 8$:      CMPB    2(TYPE_SPEC), #7                    ; 0631
                                   01 13 00105          BEQL    9$                                  ;
                                      04 00107          RET                                         ;
```

```
                10 BC      03   A2  9A 00108 9$:     MOVZBL  3(TYPE_SPEC), @NDIMS                    ; 0637
                55         04   A2  9E 0010D         MOVAB   4(R2), FLAGS_PTR                        ; 0638
          50    10 BC           08  C7 00111         DIVL3   #8, @NDIMS, R0                          ; 0643
                53         01 A045  9E 00116         MOVAB   1(R0)[FLAGS_PTR], VAL_PTR
                           0C   AE  9F 0011B         PUSHAB  VALKIND                                 ; 0644
                           14   AE  9F 0011E         PUSHAB  VAL_VECT
                           53   DD  00121           PUSHL   VAL_PTR
                6A              03  FB 00123         CALLS   #3, DBG$STA_VALSPEC
                03         0C   AE  D1 00126         CMPL    VALKIND, #3                             ; 0645
                           0E   13  00129           BEQL    10$
                           2C   A8  9F 0012C         PUSHAB  P.AAD                                   ; 0647
                           01   DD  0012F           PUSHL   #1
                00028362   8F   DD  00131           PUSHL   #164706
                69              03  FB 00137         CALLS   #3, LIB$SIGNAL
                54         10   AE  D0 0013A 10$:     MOVL    VAL_VECT, DESC_PTR                      ; 0649
          08    BC              54  D0 0013E         MOVL    DESC_PTR, @DSC$ADDR                     ; 0650
          FD    8F              63  91 00142         CMPB    (VAL_PTR), #253                         ; 0651
                                0B  12 00146         BNEQ    11$
                50         01   A3  3C 00148         MOVZWL  1(VAL_PTR), R0                          ; 0653
                52         03 A043  9E 0014C         MOVAB   3(R0)[VAL_PTR], TYPE_SPEC1
                                04  11 00151         BRB     12$
                52         05   A3  9E 00153 11$:     MOVAB   5(R3), TYPE_SPEC1                       ; 0656
                15              65  E9 00157 12$:     BLBC    (FLAGS_PTR), 13$                        ; 0662
                                52  DD 0015A         PUSHL   TYPE_SPEC1                              ; 0666
          0000V CF              01  FB 0015C         CALLS   #1, FIND_TYPREC_FROM_TSPEC
            0C  BC              50  D0 00161         MOVL    R0, @CEL[TYPE
                50              62  3C 00165         MOVZWL  (TYPE_SPEC1), R0                        ; 0667
                52         02 A042  9E 00168         MOVAB   2(R0)[TYPE_SPEC1], TYPE_SPEC1
                                3E  11 0016D         BRB     16$                                     ; 0662
                                0C  DD 0016F 13$:     PUSHL   #12                                     ; 0673
                                54  DD 00171         PUSHL   DESC_PTR
                6B              02  FB 00173         CALLS   #2, DBG$READ_ACCESS
                56         02   A4  9A 00176         MOVZBL  2(DESC_PTR), DATA_TYPE                  ; 0680
                                54  DD 0017A         PUSHL   DESC_PTR                                ; 0681
 00000000G      00              01  FB 0017C         CALLS   #1, DBG$DATA_LENGTH
                57              50  D0 00183         MOVL    R0, BITS_USED
                                16  13 00186         BEQL    14$                                     ; 0683
                           08   A4  95 00188         TSTB    8(DESC_PTR)                             ; 0684
                                11  12 0018B         BNEQ    14$
                           09   A4  95 0018D         TSTB    9(DESC_PTR)                             ; 0685
                                0C  12 00190         BNEQ    14$
                                01  DD 00192         PUSHL   #1                                      ; 0687
                7E              56  7D 00194         MOVQ    DATA_TYPE, -(SP)
          0000V CF              03  FB 00197         CALLS   #3, DBG$TYPEID_FOR_ATOMIC
                                0B  11 0019C         BRB     15$
                                01  DD 0019E 14$:     PUSHL   #1                                      ; 0690
                                03  DD 001A0         PUSHL   #3
                                54  DD 001A2         PUSHL   DESC_PTR
          0000V CF              03  FB 001A4         CALLS   #3, TYPEID_FOR_DESCR
            0C  BC              50  D0 001A9 15$:     MOVL    R0, @CELLTYPE
                10 BC           DD  001AD 16$:     PUSHL   @NDIMS                                  ; 0698
 00000000G      00              01  FB 001B0         CALLS   #1, DBG$GET_TEMPMEM
                53              50  D0 001B7         MOVL    R0, SUBSCR_VECT
            14  BC              53  D0 001BA         MOVL    SUBSCR_VECT, @DIMVECPTR                 ; 0699
                                54  D4 001BE         CLRL    I                                       ; 0700
                                1E  11 001C0         BRB     19$
          16    65              54  E1 001C2 17$:     BBC     I, (FLAGS_PTR), 18$                     ; 0702
```

```
                                    52  DD 001C6          PUSHL   TYPE_SPEC1                                  ; 0706
                          0000V  CF     01  FB 001C8          CALLS   #1, FIND_TYPREC_FROM_TSPEC                  ;
                            FC A344     50  D0 001CD          MOVL    R0, -4(SUBSCR_VECT)[I]                      ;
                               50       62  3C 001D2          MOVZWL  (TYPE_SPEC1), R0                            ; 0707
                               52  02 A042  9E 001D5          MOVAB   2(R0)[TYPE_SPEC1], TYPE_SPEC1               ;
                                       04  11 001DA          BRB     19$                                         ; 0702
                            FC A344     D4 001DC 18$:       CLRL    -4(SUBSCR_VECT)[I]                          ; 0711
                         DD         54  10  BC  F3 001E0 19$:       AOBLEQ  @NDIMS, I, 17$                      ; 0700
                                        04 001E5          RET                                                 ; 0721

; Routine Size:  486 bytes,    Routine Base:  DBG$CODE + 011A
```

```
;  596     0722  1 GLOBAL ROUTINE DBG$STA_TYP_ATOMIC(TYPEID, TYPECODE, BITSIZE): NOVALUE =
;  597     0723  1 !
;  598     0724  1 ! FUNCTION
;  599     0725  1 !       This routine returns the VAX standard type code ("one-byte type code")
;  600     0726  1 !       and the bit length for a specified atomic data type.
;  601     0727  1 !
;  602     0728  1 ! INPUTS
;  603     0729  1 !       TYPEID  - The Type ID of an Atomic data type whose VAX standard type
;  604     0730  1 !                 code is to be returned.  This data type must have the Format
;  605     0731  1 !                 Code RST$K_TYPE_ATOMIC.
;  606     0732  1 !
;  607     0733  1 !       TYPECODE - The address of a longword location to receive the type code.
;  608     0734  1 !
;  609     0735  1 !       BITSIZE  - The address of a longword location to receive the bit length
;  610     0736  1 !                  of an item of this type.
;  611     0737  1 !
;  612     0738  1 ! OUTPUTS
;  613     0739  1 !       TYPECODE - The VAX standard type code for the specified data type is
;  614     0740  1 !                  returned to TYPECODE.
;  615     0741  1 !
;  616     0742  1 !       BITSIZE  - The bit length is returned to BITSIZE.
;  617     0743  1 !
;  618     0744  1 !       No value is returned.
;  619     0745  1 !
;  620     0746  1
;  621     0747  2     BEGIN
;  622     0748  2
;  623     0749  2     MAP
;  624     0750  2         TYPEID: REF RST$ENTRY;          ! The input TYPEID for the atomic type
;  625     0751  2
;  626     0752  2     LOCAL
;  627     0753  2         FCODE,
;  628     0754  2         SIZE,
;  629     0755  2         DST_PTR : REF DST$RECORD,
;  630     0756  2         TYPE_SPEC : REF DST$TYPE_SPEC,
;  631     0757  2         TYPEID_PTR : REF RST$ENTRY;
;  632     0758  2
;  633     0759  2
;  634     0760  2
;  635     0761  2     TYPEID_PTR = FIND_TYPE_RECORD(.TYPEID, FCODE, SIZE);
;  636     0762  2     IF .FCODE NEQ RST$K_TYPE_ATOMIC
;  637     0763  2     THEN
;  638     0764  2         $DBG_ERROR('RSTTYPES\TYP_ATOMIC');
;  639     0765  2
;  640     0766  2     DST_PTR = .TYPEID_PTR[RST$L_DSTPTR];
;  641     0767  2     TYPE_SPEC = DST_PTR[DST$A_TYPSPEC_TS_ADDR] + .DST_PTR[DST$B_TYPSPEC_NAME];
;  642     0768  2     .TYPECODE = .TYPE_SPEC[DST$B_TS_ATOM_TYP];
;  643     0769  2     .BITSIZE = .SIZE;
;  644     0770  2     IF (..TYPECODE EQL DSC$K_DTYPE_T) AND (.SIZE EQL 0) THEN .BITSIZE = %BPUNIT;
;  645     0771  2     RETURN;
;  646     0772  2
;  647     0773  1     END;


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

```
41 5F 50 59 54 5C 53 45 50 59 54 54 53 52 13 00053 P.AAE:  .ASCII  <19>\RSTTYPES\<92>\TYP_ATOMIC\                    ;
                                  43 49 4D 4F 54 00062                                                            ;


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

                                                0004 00000          .ENTRY  DBG$STA_TYP_ATOMIC, Save R2              ; 0722
                                    5E              08  C2 00002     SUBL2   #8, SP                                  ;
                                                    5E  DD 00005     PUSHL   SP                                      ; 0761
                                            08      AE  9F 00007     PUSHAB  FCODE                                   ;
                                            04      AC  DD 0000A     PUSHL   TYPEID                                  ;
                                0000V       CF          03  FB 0000D     CALLS   #3, FIND_TYPE_RECORD                ;
                                    52              50  D0 00012     MOVL    R0, TYPEID_PTR                          ;
                                    02          04  AE  D1 00015     CMPL    FCODE, #2                               ; 0762
                                                    15  13 00019     BEQL    1$                                      ;
                                       00000000'    EF  9F 0001B     PUSHAB  P.AAE                                   ; 0764
                                                    01  DD 00021     PUSHL   #1                                      ;
                                       00028362     8F  DD 00023     PUSHL   #164706                                 ;
                            00000000G   00          03  FB 00029     CALLS   #3, LIB$SIGNAL                          ;
                                    50          0C  A2  D0 00030 1$:  MOVL    12(TYPEID_PTR), DST_PTR                 ; 0766
                                    51          02  A0  9A 00034     MOVZBL  2(DST_PTR), R1                          ; 0767
                                    50          03 A140 9E 00038     MOVAB   3(R1)[DST_PTR], TYPE_SPEC               ;
                                08  BC          03  A0  9A 0003D     MOVZBL  3(TYPE_SPEC), @TYPECODE                 ; 0768
                                0C  BC              6E  D0 00042     MOVL    SIZE, @BITSIZE                          ; 0769
                                    0E      08  BC  D1 00046     CMPL    @TYPECODE, #14                          ; 0770
                                                    08  12 0004A     BNEQ    2$                                      ;
                                                    6E  D5 0004C     TSTL    SIZE                                    ;
                                                    04  12 0004E     BNEQ    2$                                      ;
                                0C  BC              08  D0 00050     MOVL    #8, @BITSIZE                            ;
                                                        04 00054 2$:  RET                                             ; 0773

; Routine Size:  85 bytes,    Routine Base:  DBG$CODE + 0300
```

```
;  649          0774  1 GLOBAL ROUTINE DBG$STA_TYP_DESCR(TYPEID, DSCADDR): NOVALUE =
;  650          0775  1 !
;  651          0776  1 ! FUNCTION
;  652          0777  1 !       This routine returns the address of a VAX standard descriptor for a
;  653          0778  1 !       specified descriptor-defined data type.
;  654          0779  1 !
;  655          0780  1 ! INPUTS
;  656          0781  1 !       TYPEID  - The Type ID of the data type whose descriptor is to be
;  657          0782  1 !                 returned.  Its Format Code must be RST$K_TYPE_DESCR.
;  658          0783  1 !
;  659          0784  1 !       DSCADDR - The address of a longword location to receive the descriptor
;  660          0785  1 !                 address.
;  661          0786  1 !
;  662          0787  1 ! OUTPUTS
;  663          0788  1 !       DSCADDR - A pointer to the data type's VAX standard descriptor is
;  664          0789  1 !                 returned to DSCADDR.  This descriptor may not survive past
;  665          0790  1 !                 the end of the current command.
;  666          0791  1 !
;  667          0792  1 !       No value is returned.
;  668          0793  1 !
;  669          0794  1
;  670          0795  2     BEGIN
;  671          0796  2
;  672          0797  2     MAP
;  673          0798  2         TYPEID : REF RST$ENTRY;          ! The input TYPEID for the descriptor
;  674          0799  2                                          !      data type
;  675          0800  2
;  676          0801  2     LOCAL
;  677          0802  2         SIZE,
;  678          0803  2         FCODE,
;  679          0804  2         DSC_PTR,
;  680          0805  2         TYPE_SPEC : REF DST$TYPE_SPEC,
;  681          0806  2         DST_PTR : REF DST$RECORD,
;  682          0807  2         TYPEID_PTR : REF RST$ENTRY,
;  683          0808  2         VALKIND,
;  684          0809  2         VAL_VECT : VECTOR[3],
;  685          0810  2         VAL_PTR : REF DST$VAL_SPEC;
;  686          0811  2
;  687          0812  2
;  688          0813  2
;  689          0814  2     TYPEID_PTR = FIND_TYPE_RECORD(.TYPEID, FCODE, SIZE);
;  690          0815  2     IF .FCODE NEQ RST$K_TYPE_DESCR THEN $DBG_ERROR('RSTTYPES\TYP_DESCR 10');
;  691          0816  2     DST_PTR = .TYPEID_PTR[RST$L_DSTPTR];
;  692          0817  2     TYPE_SPEC = DST_PTR[DST$A_TYPSPEC_TS_ADDR] + .DST_PTR[DST$B_TYPSPEC_NAME];
;  693          0818  2     TYPE_SPEC = FIND_TYPESPEC(.TYPE_SPEC, SIZE, DST_PTR);
;  694          0819  2     VAL_PTR = TYPE_SPEC[DST$A_TS_DSC_VSPEC_ADDR];
;  695          0820  2     DBG$STA_VALSPEC(.VAL_PTR, VAL_VECT, VALKIND);
;  696          0821  2     IF .VALKIND NEQ DBG$K_VAL_DESCR THEN $DBG_ERROR('RSTTYPES\TYP_DESCR 20');
;  697          0822  2     .DSCADDR = .VAL_VECT[0];
;  698          0823  2     RETURN;
;  699          0824  2
;  700          0825  1     END;


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

```
44  5F  50  59  54  5C  53  45  50  59  54  54  53  52  15  00067 P.AAF:  .ASCII  <21>\RSTTYPES\<92>\TYP_DESCR 10\
                                30  31  20  52  43  53  45  00076
44  5F  50  59  54  5C  53  45  50  59  54  54  53  52  15  0007D P.AAG:  .ASCII  <21>\RSTTYPES\<92>\TYP_DESCR 20\
                                30  32  20  52  43  53  45  0008C


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

                                                    000C 00000      .ENTRY  DBG$STA_TYP_DESCR, Save R2,R3                ; 0774
                        53 00000000G    00    9E 00002      MOVAB   LIB$SIGNAL, R3
                        5E              1C    C2 00009      SUBL2   #28, SP
                                        08    AE 9F 0000C   PUSHAB  SIZE                                         ; 0814
                                        04    AE 9F 0000F   PUSHAB  FCODE
                                        04    AC DD 00012   PUSHL   TYPEID
                    0000V   CF                  03 FB 00015   CALLS   #3, FIND_TYPE_RECORD
                            52                  50 D0 0001A   MOVL    R0, TYPEID_PTR
                            03                  6E D1 0001D   CMPL    FCODE, #3                                    ; 0815
                                                11 13 00020   BEQL    1$
                                00000000'       EF 9F 00022   PUSHAB  P.AAF
                                                01 DD 00028   PUSHL   #1
                                00028362        8F DD 0002A   PUSHL   #164706
                            63                  03 FB 00030   CALLS   #3, LIB$SIGNAL
                      04    AE            0C    A2 D0 00033 1$:   MOVL    12(TYPEID_PTR), DST_PTR                  ; 0816
                            51            04    AE D0 00038   MOVL    DST_PTR, R1                                  ; 0817
                            50            02    A1 9A 0003C   MOVZBL  2(R1), R0
                            50            03 A140 9E 00040    MOVAB   3(R1)[R0], TYPE_SPEC
                                          04    AE 9F 00045   PUSHAB  DST_PTR                                      ; 0818
                                          0C    AE 9F 00048   PUSHAB  SIZE
                                                50 DD 0004B   PUSHL   TYPE_SPEC
                    0000V   CF                  03 FB 0004D   CALLS   #3, FIND_TYPESPEC
                            50                  03 C0 00052   ADDL2   #3, VAL_PTR                                  ; 0819
                                          0C    AE 9F 00055   PUSHAB  VALKIND                                      ; 0820
                                          14    AE 9F 00058   PUSHAB  VAL_VECT
                                                50 DD 0005B   PUSHL   VAL_PTR
                00000000G   00                  03 FB 0005D   CALLS   #3, DBG$STA_VALSPEC
                            03            0C    AE D1 00064   CMPL    VALKIND, #3                                  ; 0821
                                                11 13 00068   BEQL    2$
                                00000000'       EF 9F 0006A   PUSHAB  P.AAG
                                                01 DD 00070   PUSHL   #1
                                00028362        8F DD 00072   PUSHL   #164706
                            63                  03 FB 00078   CALLS   #3, LIB$SIGNAL
                      08    BC            10    AE D0 0007B 2$:   MOVL    VAL_VECT, @DSCADDR                       ; 0822
                                                04 00080      RET                                                  ; 0825

; Routine Size:  129 bytes,    Routine Base:  DBG$CODE + 0355
```

```
;  702     0826  1 GLOBAL ROUTINE DBG$STA_TYP_ENUM(TYPEID, NELTS, ELTVECPTR, BITSIZE): NOVALUE =
;  703     0827  1 !
;  704     0828  1 ! FUNCTION
;  705     0829  1 !       This routine returns information about a specified enumeration type.
;  706     0830  1 !       The type is identified by a Type ID, and the returned information
;  707     0831  1 !       includes the number of enumeration elements, a vector of SYMIDs
;  708     0832  1 !       for all the elements, and the bit length of an element of this type.
;  709     0833  1 !       (Each "element" is a constant of the enumeration type in this terminology.)
;  710     0834  1 !
;  711     0835  1 ! INPUTS
;  712     0836  1 !       TYPEID  - The Type ID of the Enumeration type whose information is to
;  713     0837  1 !                 be returned.  Its Format Code must be RST$K_TYPE_ENUM.
;  714     0838  1 !
;  715     0839  1 !       NELTS   - The address of a longword location to receive the number of
;  716     0840  1 !                 constants there are of the enumeration type.
;  717     0841  1 !
;  718     0842  1 !       ELTVECPTR - The address of a longword location to receive a pointer to
;  719     0843  1 !                 a vector of SYMIDs for the enumeration type elements.
;  720     0844  1 !
;  721     0845  1 !       BITSIZE - The address of a longword location to receive the bit length of
;  722     0846  1 !                 an item of this type.
;  723     0847  1 !
;  724     0848  1 ! OUTPUTS
;  725     0849  1 !       NELTS   - The number of enumeration type "elements" (i.e., the number of
;  726     0850  1 !                 distinct values of the type) is returned to NELTS.
;  727     0851  1 !
;  728     0852  1 !       ELTVECPTR - A pointer to a vector of SYMIDs for the enumeration type
;  729     0853  1 !                 elements is returned to ELTVECPTR.  The elements are stored
;  730     0854  1 !                 in order of their values.  Each element's SYMID can then be
;  731     0855  1 !                 used to extract the element's name and value.  This vector
;  732     0856  1 !                 disappears at the end of the current Debug command.
;  733     0857  1 !
;  734     0858  1 !       BITSIZE - The bit length is returned to BITSIZE.
;  735     0859  1 !
;  736     0860  1 !       No value is returned.
;  737     0861  1 !
;  738     0862  1
;  739     0863  2     BEGIN
;  740     0864  2
;  741     0865  2     MAP
;  742     0866  2         TYPEID : REF RST$ENTRY;          ! The input TYPEID for the enumeration
;  743     0867  2                                          !     type
;  744     0868  2
;  745     0869  2     LOCAL
;  746     0870  2         TYPEID_PTR : REF RST$ENTRY,
;  747     0871  2         FCODE,
;  748     0872  2         SIZE;
;  749     0873  2
;  750     0874  2
;  751     0875  2
;  752     0876  2     TYPEID_PTR = FIND_TYPE_RECORD(.TYPEID, FCODE, SIZE);
;  753     0877  2     IF .FCODE NEQ RST$K_TYPE_ENUM THEN $DBG_ERROR('RSTTYPES\TYP_ENUM');
;  754     0878  2     .NELTS = .TYPEID_PTR[RST$L_TYPCOMPCNT];
;  755     0879  2     .ELTVECPTR = TYPEID_PTR[RST$A_TYPCOMPLST];
;  756     0880  2     .BITSIZE = .SIZE;
;  757     0881  2     RETURN;
;  758     0882  2
```

```
;  759          0883  1     END;


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

45 5F 50 59 54 5C 53 45 50 59 54 54 53 52 11  00093 P.AAH:    .ASCII  <17>\RSTTYPES\<92>\TYP_ENUM\                    ;
                                          4D 55 4E  000A2                                                       ;


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

                                             0004 00000         .ENTRY  DBG$STA_TYP_ENUM, Save R2                ; 0826
                                  5E          08 C2 00002        SUBL2   #8, SP                                   ;
                                              5E DD 00005        PUSHL   SP                                       ; 0876
                                        08    AE 9F 00007        PUSHAB  FCODE                                    ;
                                        04    AC DD 0000A        PUSHL   TYPEID                                   ;
                              0000V     CF    03 FB 0000D        CALLS   #3, FIND_TYPE_RECORD                     ;
                                        52    50 D0 00012        MOVL    R0, TYPEID_PTR                           ;
                                  04    04    AE D1 00015        CMPL    FCODE, #4                                ; 0877
                                              15 13 00019        BEQL    1$                                       ;
                                 00000000'    EF 9F 0001B        PUSHAB  P.AAH                                    ;
                                              01 DD 00021        PUSHL   #1                                       ;
                                 00028362     8F DD 00023        PUSHL   #164706                                  ;
                          00000000G   00      03 FB 00029        CALLS   #3, LIB$SIGNAL                           ;
                                08    BC    28  A2 D0 00030 1$:  MOVL    40(TYPEID_PTR), @NELTS                   ; 0878
                                0C    BC    2C  A2 9E 00035    MOVAB   44(R2), @ELTVECPTR                       ; 0879
                                10    BC        6E D0 0003A    MOVL    SIZE, @BITSIZE                           ; 0880
                                                04 0003E       RET                                              ; 0883

; Routine Size:  63 bytes,    Routine Base:  DBG$CODE + 03D6
```

```
;  761     0884  1 GLOBAL ROUTINE DBG$STA_TYP_PICT(TYPEID, LANGCODE, PICTPTR, PICTVAL, PSCALE): NOVALUE =
;  762     0885  1 !
;  763     0886  1 ! FUNCTION
;  764     0887  1 !       This routine returns information about a Picture data type (as used in
;  765     0888  1 !       Cobol and PL/I).  The input data type is represented by a Type ID, and
;  766     0889  1 !       the picture's language code and picture string are returned as output.
;  767     0890  1 !
;  768     0891  1 ! INPUTS
;  769     0892  1 !       TYPEID  - The Type ID of the Picture data type whose information is to
;  770     0893  1 !                 be returned.  Its format Code must be RST$K_TYPE_PICT.
;  771     0894  1 !
;  772     0895  1 !       LANGCODE - The address of a longword location to receive the language
;  773     0896  1 !                  code.
;  774     0897  1 !
;  775     0898  1 !       PICTPTR - The address of a longword location to receive a pointer to the
;  776     0899  1 !                 picture itself.
;  777     0900  1 !
;  778     0901  1 !       PICTVAL - The address of a longword location to receive a pointer to the
;  779     0902  1 !                 language-specific picture encoding.
;  780     0903  1 !
;  781     0904  1 !       PSCALE  - (Optional) The address of a longword to receive the
;  782     0905  1 !                  scale factor and digit-count for this picture.
;  783     0906  1 ! OUTPUTS
;  784     0907  1 !       LANGCODE - The picture language code is returned to LANGCODE.
;  785     0908  1 !
;  786     0909  1 !       PICTPTR - A pointer to the picture itself, represented as a Counted
;  787     0910  1 !                 ASCII string, is returned to PICTPTR.
;  788     0911  1 !
;  789     0912  1 !       PICTVAL - A pointer to the language-specific picture encoding is returned
;  790     0913  1 !                 to PICTVAL; if no such encoding exists, zero is returned.
;  791     0914  1 !
;  792     0915  1 !       No value is returned.
;  793     0916  1 !
;  794     0917  1
;  795     0918  2     BEGIN
;  796     0919  2
;  797     0920  2     BUILTIN
;  798     0921  2         ACTUALCOUNT;
;  799     0922  2
;  800     0923  2     MAP
;  801     0924  2         TYPEID : REF RST$ENTRY;          ! The input TYPEID for the picture type
;  802     0925  2
;  803     0926  2     LOCAL
;  804     0927  2         TYPEID_PTR : REF RST$ENTRY,
;  805     0928  2         FCODE,
;  806     0929  2         DST_PTR : REF DST$RECORD,
;  807     0930  2         SIZE,
;  808     0931  2         VALKIND,
;  809     0932  2         VAL_SPEC,
;  810     0933  2         VAL_VECT : VECTOR[3],
;  811     0934  2         TYPE_SPEC : REF DST$TYPE_SPEC;
;  812     0935  2
;  813     0936  2
;  814     0937  2
;  815     0938  2     TYPEID_PTR = FIND_TYPE_RECORD(.TYPEID, FCODE, SIZE);
;  816     0939  2     IF .FCODE NEQ RST$K_TYPE_PICT THEN $DBG_ERROR('RSTTYPES\TYP_PICT');
;  817     0940  2     DST_PTR = .TYPEID_PTR[RST$L_DSTPTR];
```

```
;  818          0941  2     TYPE_SPEC = DST_PTR[DST$A_TYPSPEC_TS_ADDR] + .DST_PTR[DST$B_TYPSPEC_NAME];
;  819          0942  2     TYPE_SPEC = FIND_TYPESPEC(.TYPE_SPEC, SIZE, DST_PTR);
;  820          0943  2     .LANGCODE = .TYPE_SPEC[DST$B_TS_PIC_LANG];
;  821          0944  2     .PICTPTR = TYPE_SPEC[DST$A_TS_PIC_ADDR];
;  822          0945  2     IF .TYPE_SPEC[DST$W_TS_LENGTH] GTR .TYPE_SPEC[DST$B_TS_PIC_PLENG] + 3
;  823          0946  2     THEN
;  824          0947  3         BEGIN
;  825          0948  3         VAL_SPEC = ..PICTPTR + .TYPE_SPEC[DST$B_TS_PIC_PLENG];
;  826          0949  3         DBG$STA_VALSPEC(.VAL_SPEC, VAL_VECT, VAL_KIND);
;  827          0950  3         .PICTVAL = .VAL_VECT[0];
;  828          0951  3         IF (..LANGCODE EQL DBG$K_COBOL) AND ACTUALCOUNT() GTR 4
;  829          0952  3         THEN
;  830          0953  3             .PSCALE=.(.TYPE_SPEC+.TYPE_SPEC[DST$W_TS_LENGTH])<0,16,0>;
;  831          0954  3
;  832          0955  3         END
;  833          0956  3
;  834          0957  2     ELSE
;  835          0958  2         .PICTVAL = 0;
;  836          0959  2
;  837          0960  2     RETURN;
;  838          0961  2
;  839          0962  1     END;


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

50  5F  50  59  54  5C  53  45  50  59  54  54  53  52  11  000A5 P.AAI:  .ASCII  <17>\RSTTYPES\<92>\TYP_PICT\                    ;
                                                    54  43  49  000B4                                                          ;


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

                                                000C 00000          .ENTRY  DBG$STA_TYP_PICT, Save R2,R3                    ; 0884
                5E                          1C  C2 00002          SUBL2   #28, SP                                         ;
                            08              AE  9F 00005          PUSHAB  SIZE                                            ; 0938
                            04              AE  9F 00008          PUSHAB  FCODE                                           ;
                            04              AC  DD 0000B          PUSHL   TYPEID                                          ;
          0000V             CF              03  FB 0000E          CALLS   #3, FIND_TYPE_RECORD                            ;
                            52              50  DO 00013          MOVL    R0, TYPEID_PTR                                  ;
                            05              6E  D1 00016          CMPL    FCODE, #5                                       ; 0939
                                            15  13 00019          BEQL    1$                                              ;
                        00000000'           EF  9F 0001B          PUSHAB  P.AAI                                           ;
                                            01  DD 00021          PUSHL   #1                                              ;
                        00028362            8F  DD 00023          PUSHL   #164706                                         ;
        00000000G       00                  03  FB 00029          CALLS   #3, LIB$SIGNAL                                  ;
              04        AE          0C      A2  DO 00030 1$:      MOVL    12(TYPEID_PTR), DST_PTR                         ; 0940
                        51          04      AE  DO 00035          MOVL    DST_PTR, R1                                     ; 0941
                        50          02      A1  9A 00039          MOVZBL  2(R1), R0                                       ;
                        52          03  A140  9E 0003D          MOVAB   3(R1)[R0], TYPE_SPEC                            ;
                                    04      AE  9F 00042          PUSHAB  DST_PTR                                         ; 0942
                                    0C      AE  9F 00045          PUSHAB  SIZE                                            ;
                                            52  DD 00048          PUSHL   TYPE_SPEC                                       ;
          0000V             CF              03  FB 0004A          CALLS   #3, FIND_TYPESPEC                               ;
                            52              50  DO 0004F          MOVL    R0, TYPE_SPEC                                   ;
              08            BC          04  A2  9A 00052          MOVZBL  4(TYPE_SPEC), @LANGCODE                         ; 0943
```

```
                 0C   BC        06    A2  9E 00057          MOVAB   6(R2), @PICTPTR                 ; 0944
                      53              62  3C 0005C          MOVZWL  (TYPE_SPEC), R3                 ; 0945
                      50        05    A2  9A 0005F          MOVZBL  5(TYPE_SPEC), R0
                      50              03  C0 00063          ADDL2   #3, R0
                      50              53  D1 00066          CMPL    R3, R0
                                      2F  15 00069          BLEQ    2$
                      50        05    A2  9A 0006B          MOVZBL  5(TYPE_SPEC), VAL_SPEC          ; 0948
                      50        0C    BC  C0 0006F          ADDL2   @PICTPTR, VAL_SPEC
                                0C    AE  9F 00073          PUSHAB  VALKIND                         ; 0949
                                14    AE  9F 00076          PUSHAB  VAL_VECT
                                      50  DD 00079          PUSHL   VAL_SPEC
           00000000G            00        03  FB 0007B      CALLS   #3, DBG$STA_VALSPEC
                 10   BC        10    AE  D0 00082          MOVL    VAL_VECT, @PICTVAL              ; 0950
                      03        08    BC  D1 00087          CMPL    @LANGCODE, #3                   ; 0951
                                      10  12 0008B          BNEQ    3$
                      04              6C  91 0008D          CMPB    (AP), #4
                                      0B  1B 00090          BLEQU   3$
                                    6342  9F 00092          PUSHAB  (R3)[TYPE_SPEC]                 ; 0953
                 14   BC              9E  3C 00095          MOVZWL  @(SP)+, @PSCALE
                                          04 00099          RET                                     ; 0945
                                10    BC  D4 0009A 2$:      CLRL    @PICTVAL                        ; 0958
                                          04 0009D 3$:      RET                                     ; 0962

; Routine Size:  158 bytes,    Routine Base:  DBG$CODE + 0415
```

```
;  841          0963  1 GLOBAL ROUTINE DBG$STA_TYP_TYPEDPTR(TYPEID, REFTYPEID): NOVALUE =
;  842          0964  1 !
;  843          0965  1 ! FUNCTION
;  844          0966  1 !       This routine accepts as input the Type ID of a Typed Pointer data type
;  845          0967  1 !       and returns the Type ID of the data type it points to.
;  846          0968  1 !
;  847          0969  1 ! INPUTS
;  848          0970  1 !       TYPEID  - The Type ID of the Typed Pointer type whose referenced type
;  849          0971  1 !                 is to be returned.  Its Format Code must be RST$K_TYPE_TPTR.
;  850          0972  1 !
;  851          0973  1 !       REFTYPEID - The address of a longword location to receive the pointed-to
;  852          0974  1 !                 data type's Type ID.
;  853          0975  1 !
;  854          0976  1 ! OUTPUTS
;  855          0977  1 !       REFTYPEID - The Type ID of the data type which is pointed to by the
;  856          0978  1 !                 TYPEID Typed Pointer type is returned to REFTYPEID.
;  857          0979  1 !
;  858          0980  1 !       No value is returned.
;  859          0981  1 !
;  860          0982  1
;  861          0983  2     BEGIN
;  862          0984  2
;  863          0985  2     MAP
;  864          0986  2         TYPEID: REF RST$ENTRY;          ! The input TYPEID of the Typed Pointer
;  865          0987  2                                         !      data type
;  866          0988  2
;  867          0989  2     LOCAL
;  868          0990  2         FCODE,
;  869          0991  2         TYPEID_PTR: REF RST$ENTRY,
;  870          0992  2         SIZE,
;  871          0993  2         DST_PTR: REF DST$RECORD,
;  872          0994  2         TYPE_SPEC: REF DST$TYPE_SPEC;
;  873          0995  2
;  874          0996  2
;  875          0997  2
;  876          0998  2     TYPEID_PTR = FIND_TYPE_RECORD(.TYPEID, FCODE, SIZE);
;  877          0999  2     IF .FCODE NEQ RST$K_TYPE_TPTR THEN $DBG_ERROR('RSTTYPES\TYP_TYPEDPTR');                    ;
;  878          1000  2     DST_PTR = .TYPEID_PTR[RST$L_DSTPTR];
;  879          1001  2     TYPE_SPEC = DST_PTR[DST$A_TYPSPEC_TS_ADDR] + .DST_PTR[DST$B_TYPSPEC_NAME];
;  880          1002  2     TYPE_SPEC = FIND_TYPESPEC(.TYPE_SPEC, SIZE, DST_PTR);
;  881          1003  2     TYPE_SPEC = TYPE_SPEC[DST$A_TS_TPTR_TSPEC_ADDR];
;  882          1004  2     .REFTYPEID = FIND_TYPREC_FROM_TSPEC(.TYPE_SPEC);
;  883          1005  2     RETURN;
;  884          1006  2
;  885          1007  1     END;


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

54  5F  50  59  54  5C  53  45  50  59  54  54  53  52  15  000B7 P.AAJ:  .ASCII  <21>\RSTTYPES\<92>\TYP_TYPEDPTR\          ;
                             52  54  50  44  45  50  59  000C6                                                              ;


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

```
                                 0004 00000          .ENTRY  DBG$STA_TYP_TYPEDPTR, Save R2                   ; 0963
                          5E       0C   C2 00002     SUBL2   #12, SP                                         ;
                             08    AE   9F 00005     PUSHAB  SIZE                                            ; 0998
                             04    AE   9F 00008     PUSHAB  FCODE                                           ;
                             04    AC   DD 0000B     PUSHL   TYPEID                                          ;
                    0000V    CF         03 FB 0000E  CALLS   #3, FIND_TYPE_RECORD                            ;
                          52           50 DO 00013  MOVL    R0, TYPEID_PTR                                  ;
                          06           6E D1 00016  CMPL    FCODE, #6                                       ; 0999
                                       15 13 00019  BEQL    1$                                              ;
                       00000000'    EF   9F 0001B  PUSHAB  P.AAJ                                           ;
                                    01   DD 00021  PUSHL   #1                                              ;
                       00028362     8F   DD 00023  PUSHL   #164706                                         ;
                00000000G   00           03 FB 00029  CALLS   #3, LIB$SIGNAL                               ;
                      04    AE       0C   A2 DO 00030 1$:  MOVL    12(TYPEID_PTR), DST_PTR                   ; 1000
                          51   04    AE   DO 00035  MOVL    DST_PTR, R1                                     ; 1001
                          50   02    A1   9A 00039  MOVZBL  2(R1), R0
                          50   03 A140   9E 0003D  MOVAB   3(R1)[R0], TYPE_SPEC
                               04    AE   9F 00042  PUSHAB  DST_PTR                                         ; 1002
                               0C    AE   9F 00045  PUSHAB  SIZE
                                          50 DD 00048  PUSHL   TYPE_SPEC
                    0000V    CF         03 FB 0004A  CALLS   #3, FIND_TYPESPEC
                          50           03 CO 0004F  ADDL2   #3, TYPE_SPEC                                  ; 1003
                                       50 DD 00052  PUSHL   TYPE_SPEC                                      ; 1004
                    0000V    CF         01 FB 00054  CALLS   #1, FIND_TYPREC_FROM_TSPEC
                      08     BC         50 DO 00059  MOVL    R0, @REFTYPEID
                                          04 0005D  RET                                                    ; 1007

; Routine Size:  94 bytes,    Routine Base:  DBG$CODE + 04B3
```

```
;  887          1008  1 GLOBAL ROUTINE DBG$STA_TYP_FILE(TYPEID, LANGUAGE, RECTYPEID): NOVALUE =
;  888          1009  1 !
;  889          1010  1 ! FUNCTION
;  890          1011  1 !       This routine accepts as input the Type ID of a File data type
;  891          1012  1 !       and returns the associated language code and record-type typeid.
;  892          1013  1 !
;  893          1014  1 ! INPUTS
;  894          1015  1 !       TYPEID  - The Type ID of the File type.
;  895          1016  1 !                 Its Format Code must be RST$K_TYPE_FILE.
;  896          1017  1 !
;  897          1018  1 !       LANGUAGE - The address of a longword location to receive the File
;  898          1019  1 !                  type's LANGUAGE code.
;  899          1020  1 !
;  900          1021  1 !       RECTYPEID - The address of a longword location to receive
;  901          1022  1 !                   the typeid for the File's record type.
;  902          1023  1 !
;  903          1024  1 ! OUTPUTS
;  904          1025  1 !       LANGUAGE - The File type's language code.
;  905          1026  1 !
;  906          1027  1 !       RECTYPEID - The File's record-type typeid is returned to RECTYPEID.
;  907          1028  1 !
;  908          1029  1 !       No value is returned.
;  909          1030  1 !
;  910          1031  1
;  911          1032  2     BEGIN
;  912          1033  2
;  913          1034  2     MAP
;  914          1035  2         TYPEID: REF RST$ENTRY;          ! The input TYPEID of the file type
;  915          1036  2
;  916          1037  2     LOCAL
;  917          1038  2         DST_PTR: REF DST$RECORD,
;  918          1039  2         TYPE_SPEC: REF DST$TYPE_SPEC,
;  919          1040  2         TYPEID_PTR: REF RST$ENTRY,
;  920          1041  2         SIZE,
;  921          1042  2         FCODE;
;  922          1043  2
;  923          1044  2
;  924          1045  2
;  925          1046  2     TYPEID_PTR = FIND_TYPE_RECORD(.TYPEID, FCODE, SIZE);
;  926          1047  2     IF .FCODE NEQ RST$K_TYPE_FILE THEN $DBG_ERROR('RSTTYPES\TYP_FILE');
;  927          1048  2     DST_PTR = .TYPEID_PTR[RST$L_DSTPTR];
;  928          1049  2     TYPE_SPEC = DST_PTR[DST$A_TYPSPEC_TS_ADDR] + .DST_PTR[DST$B_TYPSPEC_NAME];
;  929          1050  2     TYPE_SPEC = FIND_TYPESPEC(.TYPE_SPEC, SIZE, DST_PTR);
;  930          1051  2     .LANGUAGE = .TYPE_SPEC[DST$B_TS_FILE_LANG];
;  931          1052  2     IF .TYPE_SPEC[DST$W_TS_LENGTH] GTR 2
;  932          1053  2     THEN
;  933          1054  3         BEGIN
;  934          1055  3         TYPE_SPEC = TYPE_SPEC[DST$A_TS_FILE_RCRD_TYP];
;  935          1056  3         .RECTYPEID = FIND_TYPREC_FROM_TSPEC(.TYPE_SPEC);
;  936          1057  3         END
;  937          1058  3
;  938          1059  2     ELSE
;  939          1060  2         .RECTYPEID = 0;
;  940          1061  2
;  941          1062  2     RETURN;
;  942          1063  2
;  943          1064  1     END;
```

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

46 5F 50 59 54 5C 53 45 50 59 54 54 53 52 11  000CD P.AAK:  .ASCII  <17>\RSTTYPES\<92>\TYP_FILE\
                                        45 4C 49  000DC

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

                                           0004 00000           .ENTRY  DBG$STA_TYP_FILE, Save R2                ; 1008
                                     5E     0C   C2 00002           SUBL2   #12, SP
                                            08   AF 9F 00005           PUSHAB  SIZE                             ; 1046
                                            04   AE 9F 00008           PUSHAB  FCODE
                                            04   AC DD 0000B           PUSHL   TYPEID
                                0000V       CF      03 FB 0000E           CALLS   #3, FIND_TYPE_RECORD
                                     52             50 D0 00013           MOVL    R0, TYPEID_PTR
                                     0F             6E D1 00016           CMPL    FCODE, #15                       ; 1047
                                                    15 13 00019           BEQL    1$
                                        00000000'   EF 9F 0001B           PUSHAB  P.AAK
                                                    01 DD 00021           PUSHL   #1
                                        00028362    8F DD 00023           PUSHL   #164706
                            00000000G   00          03 FB 00029           CALLS   #3, LIB$SIGNAL
                                  04    AE      0C   A2 D0 00030 1$:      MOVL    12(TYPEID_PTR), DST_PTR          ; 1048
                                     51           04   AE D0 00035           MOVL    DST_PTR, R1                      ; 1049
                                     50           02   A1 9A 00039           MOVZBL  2(R1), R0
                                     50           03 A140 9E 0003D           MOVAB   3(R1)[R0], TYPE_SPEC
                                                  04   AE 9F 00042           PUSHAB  DST_PTR                          ; 1050
                                                  0C   AE 9F 00045           PUSHAB  SIZE
                                                       50 DD 00048           PUSHL   TYPE_SPEC
                                0000V       CF      03 FB 0004A           CALLS   #3, FIND_TYPESPEC
                                  08        BC   03   A0 9A 0004F           MOVZBL  3(TYPE_SPEC), @LANGUAGE          ; 1051
                                     02             60 B1 00054           CMPW    (TYPE_SPEC), #2                  ; 1052
                                                    0F 1B 00057           BLEQU   2$
                                     50             04 C0 00059           ADDL2   #4, TYPE_SPEC                    ; 1055
                                                    50 DD 0005C           PUSHL   TYPE_SPEC                        ; 1056
                                0000V       CF      01 FB 0005E           CALLS   #1, FIND_TYPREC_FROM_TSPEC
                                  0C        BC      50 D0 00063           MOVL    R0, @RECTYPEID
                                                       04 00067           RET                                      ; 1052
                                                  0C   BC D4 00068 2$:      CLRL    @RECTYPEID                       ; 1060
                                                       04 0006B           RET                                      ; 1064

; Routine Size:  108 bytes,    Routine Base:  DBG$CODE + 0511
```

```
;  945          1065  1 GLOBAL ROUTINE DBG$STA_TYP_RECORD(TYPEID, NCOMPS, COMPVECPTR, BITSIZE): NOVALUE =
;  946          1066  1 !
;  947          1067  1 ! FUNCTION
;  948          1068  1 !       This routine accepts as input a Type ID for a record (or "structure")
;  949          1069  1 !       data type, and returns the number of record components, a vector of
;  950          1070  1 !       record component SYMIDs, and the bit length of the record as output.
;  951          1071  1 !       The record component SYMIDs can then be used as input to other symbol
;  952          1072  1 !       table access routines to extract the component names, types, and values.
;  953          1073  1 !
;  954          1074  1 ! INPUTS
;  955          1075  1 !       TYPEID  - The Type ID of the Record data type whose component informa-
;  956          1076  1 !                 tion is to be returned.  The Format Code of this data type
;  957          1077  1 !                 must be RST$K_TYPE_RECORD.
;  958          1078  1 !
;  959          1079  1 !       NCOMPS  - The address of a longword location to receive the number of
;  960          1080  1 !                 record components associated with the specified data type.
;  961          1081  1 !
;  962          1082  1 !       COMPVECPTR - The address of a longword location to receive a pointer to
;  963          1083  1 !                 a vect   of component information.
;  964          1084  1 !
;  965          1085  1 !       BITSIZE - The address of a longword location to receive the bit length
;  966          1086  1 !                 of the record.
;  967          1087  1 !
;  968          1088  1 ! OUTPUTS
;  969          1089  1 !       NCOMPS  - The number of record components is returned to NCOMPS.
;  970          1090  1 !
;  971          1091  1 !       COMPVECPTR - A pointer to a vector of record component SYMIDs is re-
;  972          1092  1 !                 turned to COMPVECPTR.  There is one SYMID for each record
;  973          1093  1 !                 component in the vector; this SYMID can then be used to
;  974          1094  1 !                 extract the component's data type, name, or value (offset
;  975          1095  1 !                 from the start of the record).  This vector disappears at
;  976          1096  1 !                 the end of the current Debug command.
;  977          1097  1 !
;  978          1098  1 !       BITSIZE - The record's bit length is returned to BITSIZE.
;  979          1099  1 !
;  980          1100  1 !       No value is returned.
;  981          1101  1 !
;  982          1102  1
;  983          1103  2     BEGIN
;  984          1104  2
;  985          1105  2     MAP
;  986          1106  2         TYPEID: REF RST$ENTRY;          ! The input TYPEID of the record type
;  987          1107  2
;  988          1108  2     LOCAL
;  989          1109  2         TYPEID_PTR: REF RST$ENTRY,
;  990          1110  2         FCODE,
;  991          1111  2         SIZE;
;  992          1112  2
;  993          1113  2
;  994          1114  2
;  995          1115  2     TYPEID_PTR = FIND_TYPE_RECORD(.TYPEID, FCODE, SIZE);
;  996          1116  2     IF .FCODE NEQ RST$K_TYPE_RECORD THEN $DBG_ERROR('RSTTYPES\TYP_RECORD');
;  997          1117  2     .NCOMPS = .TYPEID_PTR[RST$L_TYPCOMPCNT];
;  998          1118  2     .COMPVECPTR = TYPEID_PTR[RST$A_TYPCOMPLST];
;  999          1119  2     .BITSIZE = .SIZE;
; 1000          1120  2     RETURN;
; 1001          1121  2
```

```
: 1002          1122  1     END;


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

52 5F 50 59 54 5C 53 45 50 59 54 54 53 52 13  000DF P.AAL:   .ASCII  <19>\RSTTYPES\<92>\TYP_RECORD\                ;
                                  44 52 4F 43 45  000EE                                                         ;


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

                                                  0004 00000    .ENTRY  DBG$STA_TYP_RECORD, Save R2                ; 1065
                                5E                08  C2 00002   SUBL2  #8, SP                                  ;
                                                  5E  DD 00005   PUSHL  SP                                      ; 1115
                                        08        AE  9F 00007   PUSHAB  FCODE                                  ;
                                        04        AC  DD 0000A   PUSHL  TYPEID                                  ;
                            0000V       CF              03  FB 0000D   CALLS  #3, FIND_TYPE_RECORD              ;
                                52                      50  D0 00012   MOVL  R0, TYPEID_PTR                    ;
                                07              04      AE  D1 00015   CMPL  FCODE, #7                          ; 1116
                                                        15  13 00019   BEQL  1$                                 ;
                                    00000000'           EF  9F 0001B   PUSHAB  P.AAL                            ;
                                                        01  DD 00021   PUSHL  #1                               ;
                                    00028362            8F  DD 00023   PUSHL  #164706                         ;
                        00000000G   00                  03  FB 00029   CALLS  #3, LIB$SIGNAL                  ;
                              08    BC          28      A2  D0 00030 1$:  MOVL  40(TYPEID_PTR), @NCOMPS         ; 1117
                              0C    BC          2C      A2  9E 00035   MOVAB  44(R2), @COMPVECPTR              ; 1118
                              10    BC                  6E  D0 0003A   MOVL  SIZE, @BITSIZE                   ; 1119
                                                        04 0003E    RET                                         ; 1122

; Routine Size:  63 bytes,    Routine Base:  DBG$CODE + 057D
```

```
;  1004          1123  1 GLOBAL ROUTINE DBG$STA_TYP_SET(TYPEID, PARENT_TYPE, BITSIZE, OFFSET): NOVALUE =
;  1005          1124  1 !
;  1006          1125  1 ! FUNCTION
;  1007          1126  1 !       This routine accepts as input the Type ID of a Set data type (as in
;  1008          1127  1 !       Pascal, for example), and returns the set's length and parent data
;  1009          1128  1 !       type as output.  A set is always assumed to consist of a bit string
;  1010          1129  1 !       where each bit represents one member of the parent type.  The parent
;  1011          1130  1 !       data type, i.e. the data type of the set elements, can be any ordinal
;  1012          1131  1 !       type (such as integer, enumeration type, character, etc.).
;  1013          1132  1 !
;  1014          1133  1 ! INPUTS
;  1015          1134  1 !       TYPEID  - The Type ID of the Set type whose attributes are to be
;  1016          1135  1 !                 returned.  Its Format Code must be RST$K_TYPE_SET.
;  1017          1136  1 !
;  1018          1137  1 !       PARENT_TYPE - The address of a longword location to receive the Type ID
;  1019          1138  1 !                 of the set's parent type.
;  1020          1139  1 !
;  1021          1140  1 !       BITSIZE - The address of a longword location to receive the length of
;  1022          1141  1 !                 the set.
;  1023          1142  1 !
;  1024          1143  1 !       OFFSET  - (optional parameter) The address of a longword location to
;  1025          1144  1 !                 receive the virtual bit index of the first allocated bit
;  1026          1145  1 !
;  1027          1146  1 ! OUTPUTS
;  1028          1147  1 !       PARENT_TYPE - The Type ID of the set's parent type (i.e., the data type
;  1029          1148  1 !                 of the set elements) is returned to PARENT_TYPE.
;  1030          1149  1 !
;  1031          1150  1 !       BITSIZE - The length of the set type (in bits used to represent the
;  1032          1151  1 !                 set) is returned to BITSIZE.
;  1033          1152  1 !
;  1034          1153  1 !       OFFSET  - If this optional parameter was specified, the bit number of
;  1035          1154  1 !                 the first allocated bit is returned to OFFSET.  Currently
;  1036          1155  1 !                 all SETs are allocated starting at bit zero - this parameter
;  1037          1156  1 !                 is provided for future expansion.
;  1038          1157  1 !
;  1039          1158  1 !       No value is returned.
;  1040          1159  1 !
;  1041          1160  1
;  1042          1161  2     BEGIN
;  1043          1162  2
;  1044          1163  2     BUILTIN
;  1045          1164  2         ACTUALCOUNT;
;  1046          1165  2
;  1047          1166  2     MAP
;  1048          1167  2         TYPEID : REF RST$ENTRY;          ! The input TYPEID of the set type
;  1049          1168  2
;  1050          1169  2     LOCAL
;  1051          1170  2         FCODE,
;  1052          1171  2         TYPEID_PTR: REF RST$ENTRY,
;  1053          1172  2         DST_PTR: REF DST$RECORD,
;  1054          1173  2         SIZE,
;  1055          1174  2         SIZEfMP,
;  1056          1175  2         TYPE_SPEC: REF DST$TYPE_SPEC;
;  1057          1176  2
;  1058          1177  2
;  1059          1178  2
;  1060          1179  2     TYPEID_PTR = FIND_TYPE_RECORD(.TYPEID, FCODE, SIZE);
```

```
;  1061          1180  2     IF .FCODE NEQ RST$K_TYPE_SET THEN $DBG_ERROR('RSTTYPES\TYP_SET');
;  1062          1181  2     DST_PTR = .TYPEID_PTR[RST$L_DSTPTR];
;  1063          1182  2     TYPE_SPEC = DST_PTR[DST$A_TYPSPEC_TS_ADDR] + .DST_PTR[DST$B_TYPSPEC_NAME];
;  1064          1183  2     TYPE_SPEC = FIND_TYPESPEC(.TYPE_SPEC, SIZETMP, DST_PTR);
;  1065          1184  2     TYPE_SPEC = TYPE_SPEC[DST$A_TS_SET_PAR_TSPEC_ADDR];
;  1066          1185  2     .PARENT_TYPE = FIND_TYPREC_FROM_TSPEC(.TYPE_SPEC);
;  1067          1186  2     .BITSIZE = .SIZE;
;  1068          1187  2     IF ACTUALCOUNT() GEQ 4 THEN .OFFSET = 0;
;  1069          1188  2     RETURN;
;  1070          1189  2
;  1071          1190  1     END;


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

53  5F  50  59  54  5C  53  45  50  59  54  54  53  52  10  000F3 P.AAM:  .ASCII  <16>\RSTTYPES\<92>\TYP_SET\                    ;
                                                        54  45  00102                                                           ;


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

                                                  0004 00000            .ENTRY  DBG$STA_TYP_SET, Save R2                        ; 1123
                                        5E          10  C2 00002         SUBL2   #16, SP                                         ;
                                                    5E  DD 00005         PUSHL   SP                                              ; 1179
                                              08    AE  9F 00007         PUSHAB  FCODE                                           ;
                                              04    AC  DD 0000A         PUSHL   TYPEID                                          ;
                                    0000V     CF        03  FB 0000D     CALLS   #3, FIND_TYPE_RECORD                            ;
                                              52        50  D0 00012     MOVL    R0, TYPEID_PTR                                  ;
                                              08    04  AE  D1 00015     CMPL    FCODE, #8                                       ; 1180
                                                    15  13 00019         BEQL    1$                                              ;
                                 00000000'          EF  9F 0001B         PUSHAB  P.AAM                                           ;
                                                    01  DD 00021         PUSHL   #1                                              ;
                                 00028362           8F  DD 00023         PUSHL   #164706                                         ;
                          00000000G 00              03  FB 00029         CALLS   #3, LIB$SIGNAL                                  ;
                              08  AE        0C     A2  D0 00030 1$:      MOVL    12(TYPEID_PTR), DST_PTR                         ; 1181
                                  51        08  AE  D0 00035         MOVL    DST_PTR, R1                                     ; 1182
                                  50        02  A1  9A 00039         MOVZBL  2(R1), R0                                       ;
                                  50        03 A140 9E 0003D         MOVAB   3(R1)[R0], TYPE_SPEC                            ;
                                            08  AE  9F 00042         PUSHAB  DST_PTR                                         ; 1183
                                            10  AE  9F 00045         PUSHAB  SIZETMP                                         ;
                                                50  DD 00048         PUSHL   TYPE_SPEC                                       ;
                                    0000V   CF      03  FB 0004A     CALLS   #3, FIND_TYPESPEC                               ;
                                            50      07  C0 0004F     ADDL2   #7, TYPE_SPEC                                   ; 1184
                                                50  DD 00052         PUSHL   TYPE_SPEC                                       ; 1185
                                    0000V   CF      01  FB 00054     CALLS   #1, FIND_TYPREC_FROM_TSPEC                      ;
                                      08    BC      50  D0 00059     MOVL    R0, @PARENT_TYPE                                ;
                                      0C    BC      6E  D0 0005D     MOVL    SIZE, @BITSIZE                                  ; 1186
                                            04      6C  91 00061     CMPB    (AP), #4                                        ; 1187
                                                    03  1F 00064     BLSSU   2$                                              ;
                                              10    BC  D4 00066     CLRL    @OFFSET                                         ;
                                                        04 00069 2$:  RET                                                     ; 1190

; Routine Size:  106 bytes,    Routine Base:  DBG$CODE + 05BC
```

```
;  1073          1191  1 GLOBAL ROUTINE DBG$S1A_TYP_SUBRNG(TYPEID, PARENT_TYPE, LOWPTR, HIGHPTR,
;  1074          1192  1                                   BITSIZE): NOVALUE =
;  1075          1193  1 !
;  1076          1194  1 ! FUNCTION
;  1077          1195  1 !       This routine accepts the Type ID of a Subrange data type (as in Pascal,
;  1078          1196  1 !       for instance) as input, and returns that type's parent type, subrange
;  1079          1197  1 !       interval, and bit length.  The parent type can be any data type with
;  1080          1198  1 !       values of reasonable size (e.g., anything that can be represented as an
;  1081          1199  1 !       integer or real value), and the subrange interval is returned as
;  1082          1200  1 !       pointers to the lower bound value and the upper bound value.  Both
;  1083          1201  1 !       values are assumed to be of the parent type.
;  1084          1202  1 !
;  1085          1203  1 ! INPUTS
;  1086          1204  1 !       TYPEID  - The Type ID of the Subrange data type whose attributes are to
;  1087          1205  1 !                 be returned.  Its Format Code must be RST$K_TYPE_SUBRNG.
;  1088          1206  1 !
;  1089          1207  1 !       PARENT_TYPE - The address of a longword location to receive the returned
;  1090          1208  1 !                 Type ID of subrange's parent type.
;  1091          1209  1 !
;  1092          1210  1 !       LOWPTR  - The address of a longword location to receive a pointer to the
;  1093          1211  1 !                 lower bound value of the subrange.
;  1094          1212  1 !
;  1095          1213  1 !       HIGHPTR - The address of a longword location to receive a pointer to the
;  1096          1214  1 !                 upper bound value of the subrange.
;  1097          1215  1 !
;  1098          1216  1 !       BITSIZE - The address of a longword location to receive the length in bits
;  1099          1217  1 !                 of an item of this type.
;  1100          1218  1 !
;  1101          1219  1 ! OUTPUTS
;  1102          1220  1 !       PARENT_TYPE - The Type ID of the subrange's parent type (i.e., the data
;  1103          1221  1 !                 type of which this is a subrange) is returned to PARENT_TYPE.
;  1104          1222  1 !
;  1105          1223  1 !       LOWPTR  - A pointer to the lower bound value of the subrange is returned
;  1106          1224  1 !                 to LOWPTR.  This value must be interpreted as being of the
;  1107          1225  1 !                 parent type.  The value disappears at the end of the current
;  1108          1226  1 !                 Debug command.
;  1109          1227  1 !
;  1110          1228  1 !       HIGHPTR - A pointer to the upper bound value of the subrange is returned
;  1111          1229  1 !                 to HIGHPTR.  This value must be intepreted as being of the
;  1112          1230  1 !                 parent type.  The value disappears at the end of the current
;  1113          1231  1 !                 Debug command.
;  1114          1232  1 !
;  1115          1233  1 !       BITSIZE - The length in bits is returned to BITSIZE.
;  1116          1234  1 !
;  1117          1235  1 !       No value is returned.
;  1118          1236  1 !
;  1119          1237  1
;  1120          1238  2     BEGIN
;  1121          1239  2
;  1122          1240  2     MAP
;  1123          1241  2         TYPEID: REF RST$ENTRY;          ! The input TYPEID of the subrange type
;  1124          1242  2
;  1125          1243  2     LOCAL
;  1126          1244  2         SIZE,
;  1127          1245  2         SIZETMP,
;  1128          1246  2         TYPEID_PTR: REF RST$ENTRY,
;  1129          1247  2         TYPE_SPEC: REF DST$TYPE_SPEC,
```

```
;  1130          1248  2          VAL_PTR: REF DST$VAL_SPEC,
;  1131          1249  2          DST_PTR: REF DST$RECORD,
;  1132          1250  2          VAL_VECT: VECTOR[3],
;  1133          1251  2          VALKIND,
;  1134          1252  2          FCODE;
;  1135          1253  2
;  1136          1254  2
;  1137          1255  2
;  1138          1256  2      TYPEID_PTR = FIND_TYPE_RECORD(.TYPEID, FCODE, SIZE);
;  1139          1257  2      IF .FCODE NEQ RST$K_TYPE_SUBRNG THEN $DBG_ERROR('RSTTYPES\TYP_SUBRNG');
;  1140          1258  2      DST_PTR = .TYPEID_PTR[RST$L_DSTPTR];
;  1141          1259  2      TYPE_SPEC = DST_PTR[DST$A_TYPSPEC_TS_ADDR] + .DST_PTR[DST$B_TYPSPEC_NAME];
;  1142          1260  2      TYPE_SPEC = FIND_TYPESPEC(.TYPE_SPEC, SIZETMP, DST_PTR);
;  1143          1261  2      TYPE_SPEC = TYPE_SPEC[DST$A_TS_SUBR_PAR_TSPEC_ADDR];
;  1144          1262  2      .PARENT_TYPE = FIND_TYPREC_FROM_TSPEC(.TYPE_SPEC);
;  1145          1263  2      VAL_PTR = .TYPE_SPEC + .TYPE_SPEC[DST$W_TS_LENGTH] + 2;
;  1146          1264  2      DBG$STA_VALSPEC(.VAL_PTR, VAL_VECT, VALKIND);
;  1147          1265  2      .LOWPTR = .VAL_VECT[0];
;  1148          1266  2      IF .VAL_PTR[DST$B_VS_VFLAGS] EQL DST$K_VS_FOLLOWS
;  1149          1267  2      THEN
;  1150          1268  2          VAL_PTR = .VAL_PTR + .VAL_PTR[DST$W_VS_LENGTH] + 3
;  1151          1269  2
;  1152          1270  2      ELSE
;  1153          1271  2          VAL_PTR = .VAL_PTR + 5;
;  1154          1272  2
;  1155          1273  2      DBG$STA_VALSPEC(.VAL_PTR, VAL_VECT, VALKIND);
;  1156          1274  2      .HIGHPTR = .VAL_VECT[0];
;  1157          1275  2      .BITSIZE = .SIZE;
;  1158          1276  2
;  1159          1277  2      RETURN;
;  1160          1278  2
;  1161          1279  1      END;


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

53 5F 50 59 54 5C 53 45 50 59 54 54 53 52 13  00104 P.AAN:    .ASCII  <19>\RSTTYPES\<92>\TYP_SUBRNG\                  ;
                              47 4E 52 42 55  00113                                                             ;


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

                                                      000C 00000        .ENTRY  DBG$STA_TYP_SUBRNG, Save R2,R3          ; 1191
                                                53 00000000G  00  9E 00002  MOVAB   DBG$STA_VALSPEC, R3                 ;
                                                5E            20  C2 00009  SUBL2   #32, SP                          ;
                                                              5E  DD 0000C  PUSHL   SP                                ; 1256
                                                        08    AE  9F 0000E  PUSHAB  FCODE                             ;
                                                        04    AC  DD 00011  PUSHL   TYPEID                            ;
                                             0000V      CF          03  FB 00014  CALLS   #3, FIND_TYPE_RECORD         ;
                                                52            50  D0 00019  MOVL    R0, TYPEID_PTR                    ;
                                                09        04  AE  D1 0001C  CMPL    FCODE, #9                         ; 1257
                                                              15  13 00020  BEQL    1$                                ;
                                                00000000'     EF  9F 00022  PUSHAB  P.AAN                             ;
                                                              01  DD 00028  PUSHL   #1                               ;
                                                00028362      8F  DD 0002A  PUSHL   #164706                           ;
```

```
                00000000G  00          03  FB 00030          CALLS   #3, LIB$SIGNAL                          ;
                      08   AE    0C    A2  D0 00037 1$:      MOVL    12(TYPEID_PTR), DST_PTR                ; 1258
                           51    08    AE  D0 0003C          MOVL    DST_PTR, R1                            ; 1259
                           50    02    A1  9A 00040          MOVZBL  2(R1), R0                              ;
                           52    03 A140   9E 00044          MOVAB   3(R1)[R0], TYPE_SPEC                   ;
                                 08    AE  9F 00049          PUSHAB  DST_PTR                                ; 1260
                                 10    AE  9F 0004C          PUSHAB  SIZETMP                                ;
                                       52  DD 0004F          PUSHL   TYPE_SPEC                              ;
                          0000V    CF        03  FB 00051    CALLS   #3, FIND_TYPESPEC                      ;
                           52              50  D0 00056    MOVL    R0, TYPE_SPEC                          ;
                           52              07  C0 00059    ADDL2   #7, TYPE_SPEC                          ; 1261
                                           52  DD 0005C    PUSHL   TYPE_SPEC                              ; 1262
                          0000V    CF        01  FB 0005E    CALLS   #1, FIND_TYPREC_FROM_TSPEC             ;
                             08    BC        50  D0 00063    MOVL    R0, @PARENT_TYPE                       ;
                           50              62  3C 00067    MOVZWL  (TYPE_SPEC), R0                        ; 1263
                           52    02 A042   9E 0006A          MOVAB   2(R0)[TYPE_SPEC], VAL_PTR              ;
                                 10    AE  9F 0006F          PUSHAB  VALKIND                                ; 1264
                                 18    AE  9F 00072          PUSHAB  VAL_VECT                               ;
                                       52  DD 00075          PUSHL   VAL_PTR                                ;
                              63           03  FB 00077    CALLS   #3, DBG$STA_VALSPEC                    ;
                             0C    BC    14    AE  D0 0007A    MOVL    VAL_VECT, @LOWPTR                      ; 1265
                             FD    8F          62  91 0007F    CMPB    (VAL_PTR), #253                        ; 1266
                                               0B  12 00083    BNEQ    2$                                     ;
                           50    01    A2        3C 00085    MOVZWL  1(VAL_PTR), R0                         ; 1268
                           52    03 A042         9E 00089    MOVAB   3(R0)[VAL_PTR], VAL_PTR                ;
                                               03  11 0008E    BRB     3$                                     ;
                           52                  05  C0 00090 2$: ADDL2   #5, VAL_PTR                        ; 1271
                                 10    AE  9F 00093 3$:      PUSHAB  VALKIND                                ; 1273
                                 18    AE  9F 00096          PUSHAB  VAL_VECT                               ;
                                       52  DD 00099          PUSHL   VAL_PTR                                ;
                              63           03  FB 0009B    CALLS   #3, DBG$STA_VALSPEC                    ;
                             10    BC    14    AE  D0 0009E    MOVL    VAL_VECT, @HIGHPTR                     ; 1274
                             14    BC          6E  D0 000A3    MOVL    SIZE, @BITSIZE                         ; 1275
                                               04  000A7    RET                                            ; 1279

; Routine Size:  168 bytes,    Routine Base:  DBG$CODE + 0626
```

```
;  1163          1280  1 GLOBAL ROUTINE DBG$STA_TYP_OFFSET(TYPEID, AREA_ADDR, AREA_LEN): NOVALUE =
;  1164          1281  1 !
;  1165          1282  1 ! FUNCTION
;  1166          1283  1 !       This routine accepts as input the Type ID of an Offset data type and
;  1167          1284  1 !       it returns the address and length of the associated Area.  Areas and
;  1168          1285  1 !       Offsets are PL/I data types.
;  1169          1286  1 !
;  1170          1287  1 ! INPUTS
;  1171          1288  1 !       TYPEID  - The Type ID of the Offset data type whose attributes are to be
;  1172          1289  1 !                 returned.  Its Format Code must be RST$K_TYPE_OFFSET.
;  1173          1290  1 !
;  1174          1291  1 !       AREA_ADDR - The address of a longword location to receive the byte
;  1175          1292  1 !                 address of the Area associated with this Offset.
;  1176          1293  1 !
;  1177          1294  1 !       AREA_LEN - The address of a longword location to receive the byte
;  1178          1295  1 !                 length of the Area associated with this Offset.
;  1179          1296  1 !
;  1180          1297  1 ! OUTPUTS
;  1181          1298  1 !       AREA_ADDR - The byte address of the Area associated with this Offset
;  1182          1299  1 !                 is returned to AREA_ADDR.
;  1183          1300  1 !
;  1184          1301  1 !       AREA_LEN - The byte length of the Area associated with this Offset
;  1185          1302  1 !                 is returned to AREA_LEN.
;  1186          1303  1 !
;  1187          1304  1 !       No routine value is returned.
;  1188          1305  1 !
;  1189          1306  1 !
;  1190          1307  2     BEGIN
;  1191          1308  2
;  1192          1309  2     MAP
;  1193          1310  2         TYPEID: REF RST$ENTRY,          ! Pointer to the input Type RST Entry
;  1194          1311  2         AREA_ADDR: REF VECTOR[1],       ! Address where we return Area address
;  1195          1312  2         AREA_LEN: REF VECTOR[1];        ! Address where we return Area length
;  1196          1313  2
;  1197          1314  2     LOCAL
;  1198          1315  2         DSTPTR: REF DST$RECORD,         ! Pointer to Offset Type Spec DST record
;  1199          1316  2         FCODE,                          ! The Format Code for this data type
;  1200          1317  2         SIZE,                           ! The bit size of this data object
;  1201          1318  2         TSPTR: REF DST$TYPE_SPEC,       ! Pointer to DST Type Spec for Offset
;  1202          1319  2         TYPEPTR: REF RST$ENTRY,         ! Pointer to the offset Type RST Entry
;  1203          1320  2         VALKIND,                        ! The kind of value returned by the
;  1204          1321  2                                         !     DBG$STA_VALSPEC routine
;  1205          1322  2         VALPTR: REF VECTOR[,LONG],      ! Pointer to the Area length value
;  1206          1323  2         VALVECTOR: VECTOR[3,LONG],      ! Three-longword value vector returned
;  1207          1324  2                                         !     by the DBG$STA_VALSPEC routine
;  1208          1325  2         VSPTR: REF DST$VAL_SPEC;        ! Pointer to Value Specs in the DST Type
;  1209          1326  2                                         !     Spec for Offset
;  1210          1327  2
;  1211          1328  2
;  1212          1329  2
;  1213          1330  2     ! In case the caller passed in a SYMID for the Offset symbol instead of its
;  1214          1331  2     ! Type ID, we get the symbol's actual Type ID here.  We also make sure we
;  1215          1332  2     ! really have an Offset data type.
;  1216          1333  2     !
;  1217          1334  2     TYPEPTR = FIND_TYPE_RECORD(.TYPEID, FCODE, SIZE);
;  1218          1335  2     IF .FCODE NEQ RST$K_TYPE_OFFSET
;  1219          1336  2     THEN
```

```
;  1220          1337  2          $DBG_ERROR('RSTTYPES\TYP_OFFSET');
;  1221          1338  2
;  1222          1339  2
;  1223          1340  2      ! Get a pointer to the Type Spec DST Record and get a pointer to the actual
;  1224          1341  2      ! Type Spec within it (possibly after Type Spec indirection).
;  1225          1342  2      !
;  1226          1343  2      DSTPTR = .TYPEPTR[RST$L_DSTPTR];
;  1227          1344  2      TSPTR = DSTPTR[DST$A_TYPSPEC_TS_ADDR] + .DSTPTR[DST$B_TYPSPEC_NAME];
;  1228          1345  2      TSPTR = FIND_TYPESPEC(.TSPTR, SIZE, DSTPTR);
;  1229          1346  2
;  1230          1347  2
;  1231          1348  2      ! Evaluate the first Value Spec in the Offset Type Spec.  Return the result
;  1232          1349  2      ! (i.e., the address of the Area) to the AREA_ADDR parameter.
;  1233          1350  2      !
;  1234          1351  2      VSPTR = TSPTR[DST$A_TS_OFFSET_VALSPEC];
;  1235          1352  2      DBG$STA_VALSPEC(.VSPTR, VALVECTOR, VALKIND);
;  1236          1353  2      AREA_ADDR[0] = .VALVECTOR[0];
;  1237          1354  2
;  1238          1355  2
;  1239          1356  2      ! Get to the second Value Spec in the Offset Type Spec and evaluate it.
;  1240          1357  2      ! Return its result (i.e., the length of the Area) to AREA_LEN.
;  1241          1358  2      !
;  1242          1359  2      IF .VSPTR[DST$B_VS_VFLAGS] EQL DST$K_VS_FOLLOWS
;  1243          1360  2      THEN
;  1244          1361  2          VSPTR = VSPTR[DST$B_VS_ALLOC] + .VSPTR[DST$W_VS_LENGTH]
;  1245          1362  2
;  1246          1363  2      ELSE
;  1247          1364  2          VSPTR = .VSPTR + 5;
;  1248          1365  2
;  1249          1366  2      DBG$STA_VALSPEC(.VSPTR, VALVECTOR, VALKIND);
;  1250          1367  2      VALPTR = .VALVECTOR[0];
;  1251          1368  2      AREA_LEN[0] = .VALPTR[0];
;  1252          1369  2
;  1253          1370  2
;  1254          1371  2      ! We are all done--return to the caller.
;  1255          1372  2      !
;  1256          1373  2      RETURN;
;  1257          1374  2
;  1258          1375  1      END;


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

4F  5F  50  59  54  5C  53  45  50  59  54  54  53  52  13  00118 P.AAO:  .ASCII  <19>\RSTTYPES\<92>\TYP_OFFSET\                  ;
                                        54  45  53  46  46  00127                                                               ;


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

                                                    000C 00000          .ENTRY  DBG$STA_TYP_OFFSET, Save R2,R3                  ; 1280
                                    53 00000000G    00   9E 00002      MOVAB   DBG$STA_VALSPEC, R3                             ;
                                    5E              1C   C2 00009      SUBL2   #28, SP                                         ;
                                              08    AE   9F 0000C      PUSHAB  SIZE                                            ; 1334
                                              04    AE   9F 0000F      PUSHAB  FCODE                                           ;
                                              04    AC   DD 00012      PUSHL   TYPEID                                          ;
```

```
                0000V  CF            03  FB 00015          CALLS   #3, FIND_TYPE_RECORD
                       52            50  D0 0001A          MOVL    R0, TYPEPTR
                       12            6E  D1 0001D          CMPL    FCODE, #18                      ; 1335
                                     15  13 00020          BEQL    1$
                            00000000'  EF  9F 00022          PUSHAB  P.AAO                           ; 1337
                                     01  DD 00028          PUSHL   #1
                            00028362   8F  DD 0002A          PUSHL   #164706
                00000000G   00           03  FB 00030          CALLS   #3, LIB$SIGNAL
                      04    AE       0C    A2  D0 00037 1$:     MOVL    12(TYPEPTR), DSTPTR             ; 1343
                            51       04    AE  D0 0003C         MOVL    DSTPTR, R1                      ; 1344
                            50       02    A1  9A 00040         MOVZBL  2(R1), R0
                            50       03  A140  9E 00044         MOVAB   3(R1)[R0], TSPTR
                                     04    AE  9F 00049         PUSHAB  DSTPTR                          ; 1345
                                     0C    AE  9F 0004C         PUSHAB  SIZE
                                     50        DD 0004F         PUSHL   TSPTR
                0000V  CF                03  FB 00051         CALLS   #3, FIND_TYPESPEC
                       52            03    A0  9E 00056         MOVAB   3(R0), VSPTR                    ; 1351
                                     0C    AE  9F 0005A         PUSHAB  VALKIND                         ; 1352
                                     14    AE  9F 0005D         PUSHAB  VALVECTOR
                                     52        DD 00060         PUSHL   VSPTR
                            63               03  FB 00062         CALLS   #3, DBG$STA_VALSPEC
                      08    BC       10    AE  D0 00065         MOVL    VALVECTOR, @AREA_ADDR           ; 1353
                      FD    8F               62  91 0006A         CMPB    (VSPTR), #253                   ; 1359
                                             0B  12 0006E         BNEQ    2$
                            50       01    A2  3C 00070         MOVZWL  1(VSPTR), R0                    ; 1361
                            52       03  A042  9E 00074         MOVAB   3(R0)[VSPTR], VSPTR
                                             03  11 00079         BRB     3$
                            52               05  C0 0007B 2$:     ADDL2   #5, VSPTR                       ; 1364
                                     0C    AE  9F 0007E 3$:     PUSHAB  VALKIND                         ; 1366
                                     14    AE  9F 00081         PUSHAB  VALVECTOR
                                     52        DD 00084         PUSHL   VSPTR
                            63               03  FB 00086         CALLS   #3, DBG$STA_VALSPEC
                            50       10    AE  D0 00089         MOVL    VALVECTOR, VALPTR               ; 1367
                      0C    BC               60  D0 0008D         MOVL    (VALPTR), @AREA_LEN             ; 1368
                                             04  00091         RET                                     ; 1375

; Routine Size:  146 bytes,    Routine Base:  DBG$CODE + 06CE
```

```
;1260     1376  1 GLOBAL ROUTINE DBG$STA_TYP_AREA(TYPEID, BYTE_LEN): NOVALUE =
;1261     1377  1 !
;1262     1378  1 ! FUNCTION
;1263     1379  1 !       This routine accepts as input the Type ID of an AREA data type
;1264     1380  1 !       and returns the byte length of the AREA.
;1265     1381  1 !
;1266     1382  1 ! INPUTS
;1267     1383  1 !       TYPEID  - The Type ID of the AREA data type whose attributes are to be
;1268     1384  1 !                 returned.  Its Format Code must be RST$K_TYPE_AREA.
;1269     1385  1 !
;1270     1386  1 !       BYTE_LEN - The address of a longword location to receive a pointer
;1271     1387  1 !                  to the byte length of the AREA.
;1272     1388  1 !
;1273     1389  1 ! OUTPUTS
;1274     1390  1 !       BYTE_LEN - A pointer to the byte length of the AREA is returned to
;1275     1391  1 !                   BYTE_LEN
;1276     1392  1 !
;1277     1393  1 !
;1278     1394  1 !       No value is returned.
;1279     1395  1 !
;1280     1396  1 !
;1281     1397  2     BEGIN
;1282     1398  2
;1283     1399  2     MAP
;1284     1400  2         TYPEID: REF RST$ENTRY;          ! The input TYPEID of the PL/I AREA type
;1285     1401  2
;1286     1402  2     LOCAL
;1287     1403  2         TYPEID_PTR: REF RST$ENTRY,
;1288     1404  2         DST_PTR: REF DST$RECORD,
;1289     1405  2         TYPE_SPEC: REF DST$TYPE_SPEC,
;1290     1406  2         VAL_SPEC: REF DST$VAL_SPEC,
;1291     1407  2         VALKIND,
;1292     1408  2         FCODE,
;1293     1409  2         VAL_VECT: VECTOR[3],
;1294     1410  2         SIZE;
;1295     1411  2
;1296     1412  2
;1297     1413  2
;1298     1414  2     TYPEID_PTR = FIND_TYPE_RECORD(.TYPEID, FCODE, SIZE);
;1299     1415  2     IF .FCODE NEQ RST$K_TYPE_AREA THEN $DBG_ERROR('RSTTYPES\TYP_AREA');
;1300     1416  2     DST_PTR = .TYPEID_PTR[RST$L_DSTPTR];
;1301     1417  2     TYPE_SPEC = DST_PTR[DST$A_TYPSPEC_TS_ADDR] + .DST_PTR[DST$B_TYPSPEC_NAME];
;1302     1418  2     TYPE_SPEC = FIND_TYPESPEC(.TYPE_SPEC, SIZE, DST_PTR);
;1303     1419  2     IF .SIZE EQL 0
;1304     1420  2     THEN
;1305     1421  3         BEGIN
;1306     1422  3         VAL_SPEC = TYPE_SPEC[DST$A_TS_AREA_BYTE_LEN];
;1307     1423  3         DBG$STA_VALSPEC(.VAL_SPEC, VAL_VECT, VALKIND);
;1308     1424  3         .BYTE_LEN = .VAL_VECT[0];
;1309     1425  3         END
;1310     1426  3
;1311     1427  2     ELSE
;1312     1428  2         .BYTE_LEN = .SIZE;
;1313     1429  2
;1314     1430  2     RETURN;
;1315     1431  2
;1316     1432  1     END;
```

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

41  5F  50  59  54  5C  53  45  50  59  54  54  53  52  11  0012C P.AAP:  .ASCII  <17>\RSTTYPES\<92>\TYP_AREA\                    ;
                                                    41  45  52  00138

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

                                                        0004 00000      .ENTRY  DBG$STA_TYP_AREA, Save R2                       ; 1376
                                                5E          1C  C2 00002        SUBL2   #28, SP                             ;
                                                        08  AE  9F 00005        PUSHAB  SIZE                                ; 1414
                                                        04  AE  9F 00008        PUSHAB  FCODE                               ;
                                                        04  AC  DD 0000B        PUSHL   TYPEID                              ;
                                        0000V   CF          03  FB 0000E        CALLS   #3, FIND_TYPE_RECORD                ;
                                                52          50  D0 00013        MOVL    R0, TYPEID_PTR                      ;
                                                11          6E  D1 00016        CMPL    FCODE, #17                          ; 1415
                                                            15  13 00019        BEQL    1$                                  ;
                                            00000000'   EF  9F 0001B        PUSHAB  P.AAP                               ;
                                                        01  DD 00021        PUSHL   #1                                  ;
                                            00028362    8F  DD 00023        PUSHL   #164706                             ;
                                    00000000G   00          03  FB 00029        CALLS   #3, LIB$SIGNAL                      ;
                                            04  AE      0C  A2  D0 00030 1$:    MOVL    12(TYPEID_PTR), DST_PTR             ; 1416
                                                51      04  AE  D0 00035        MOVL    DST_PTR, R1                         ; 1417
                                                50      02  A1  9A 00039        MOVZBL  2(R1), R0                           ;
                                                50      03 A140 9E 0003D        MOVAB   3(R1)[R0], TYPE_SPEC                ;
                                                        04  AE  9F 00042        PUSHAB  DST_PTR                             ; 1418
                                                        0C  AE  9F 00045        PUSHAB  SIZE                                ;
                                                            50  DD 00048        PUSHL   TYPE_SPEC                           ;
                                        0000V   CF          03  FB 0004A        CALLS   #3, FIND_TYPESPEC                   ;
                                                        08  AE  D5 0004F        TSTL    SIZE                                ; 1419
                                                            18  12 00052        BNEQ    2$                                  ;
                                                50          03  C0 00054        ADDL2   #3, VAL_SPEC                        ; 1422
                                                        0C  AE  9F 00057        PUSHAB  VALKIND                             ; 1423
                                                        14  AE  9F 0005A        PUSHAB  VAL_VECT                            ;
                                                            50  DD 0005D        PUSHL   VAL_SPEC                            ;
                                    00000000G   00          03  FB 0005F        CALLS   #3, DBG$STA_VALSPEC                 ;
                                            08  BC      10  AE  D0 00066        MOVL    VAL_VECT, @BYTE_LEN                 ; 1424
                                                            04 0006B        RET                                         ; 1419
                                            08  BC      08  AE  D0 0006C 2$:    MOVL    SIZE, @BYTE_LEN                     ; 1428
                                                            04 00071        RET                                         ; 1432

; Routine Size:  114 bytes,    Routine Base:  DBG$CODE + 0760
```

```
;  1318          1433  1 GLOBAL ROUTINE DBG$STA_TYP_VARIANT(TYPEID, NCOMPS, COMPVECPTR, TAG,
;  1319          1434  1                                                  BITSIZE): NOVALUE =
;  1320          1435  1 !
;  1321          1436  1 ! FUNCTION
;  1322          1437  1 !       This routine accepts as input the Type ID of a Variant data type or the
;  1323          1438  1 !       SYMID of a symbol of variant type, and it returns the number of variant
;  1324          1439  1 !       components, the SYMID of each such component, and the SYMID of the vari-
;  1325          1440  1 !       ant's tag field.
;  1326          1441  1 !
;  1327          1442  1 ! INPUTS
;  1328          1443  1 !       TYPEID  - The Type ID of the Variant data type whose attributes and var-
;  1329          1444  1 !                 iants are to be returned.  The Kind of this data type
;  1330          1445  1 !                 must be RST$K_VARIANT.
;  1331          1446  1 !
;  1332          1447  1 !       NCOMPS  - The address of a longword location to receive the number of
;  1333          1448  1 !                 variant components of the TYPEID data type.
;  1334          1449  1 !
;  1335          1450  1 !       COMPVECPTR - The address of a longword location to receive a pointer to
;  1336          1451  1 !                 a vector of variant component SYMIDs and tag values.
;  1337          1452  1 !
;  1338          1453  1 !       TAG     - The address of a longword location to receive the SYMID of the
;  1339          1454  1 !                 Variant data type's tag field.
;  1340          1455  1 !
;  1341          1456  1 !       BITSIZE - The address of a longword location to receive the size in bits
;  1342          1457  1 !                 of an item of this type.
;  1343          1458  1 !
;  1344          1459  1 ! OUTPUTS
;  1345          1460  1 !       NCOMPS  - The number of components of the Variant data type (the number
;  1346          1461  1 !                 of different "variants") is returned to NCOMPS.
;  1347          1462  1 !
;  1348          1463  1 !       COMPVECPTR - A pointer to a vector of variant component information is
;  1349          1464  1 !                 returned to COMPVECPTR.  In this vector, there is a two-long-
;  1350          1465  1 !                 word entry for each variant component; each such entry gives
;  1351          1466  1 !                 the SYMID and a pointer to the tag value, respectively, of
;  1352          1467  1 !                 the corresponding component.  The SYMID can then be used to
;  1353          1468  1 !                 extract the component's name, data type, and value.  The tag
;  1354          1469  1 !                 value must be intepreted as a value of tag field's data type.
;  1355          1470  1 !                 This vector disappears at the end of the current command.
;  1356          1471  1 !
;  1357          1472  1 !       TAG     - The SYMID of the variant tag field is returned to TAG.  This
;  1358          1473  1 !                 SYMID can be used to extract the tag field name, data type,
;  1359          1474  1 !                 and value.
;  1360          1475  1 !
;  1361          1476  1 !       BITSIZE - The size in bits of an item of this type is returned to
;  1362          1477  1 !                 BITSIZE.
;  1363          1478  1 !
;  1364          1479  1 !       No value is returned.
;  1365          1480  1 !
;  1366          1481  1
;  1367          1482  2     BEGIN
;  1368          1483  2
;  1369          1484  2     MAP
;  1370          1485  2         TYPEID : REF RST$ENTRY;          ! Pointer to Variant RST Entry
;  1371          1486  2
;  1372          1487  2     LOCAL
;  1373          1488  2         DSTPTR: REF DST$RECORD,
;  1374          1489  2         TYPEID_PTR: REF RST$ENTRY,
```

```
;  1375          1490  2          FCODE,
;  1376          1491  2          SIZE;
;  1377          1492  2
;  1378          1493  2
;  1379          1494  2
;  1380          1495  2      TYPEID_PTR = FIND_TYPE_RECORD(.TYPEID, FCODE, SIZE);
;  1381          1496  2      IF .FCODE NEQ RST$K_VARIANT THEN $DBG_ERROR('RSTTYPES\TYP_VARIANT');
;  1382          1497  2      .NCOMPS = .TYPEID_PTR[RST$L_VARSET_CNT];
;  1383          1498  2      .COMPVECPTR = TYPEID_PTR[RST$A_VARSETTBL];
;  1384          1499  2      .TAG = .TYPEID_PTR[RST$L_VARTAGPTR];
;  1385          1500  2      .BITSIZE = .SIZE;
;  1386          1501  2      RETURN;
;  1387          1502  2
;  1388          1503  1      END;


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

56 5F 50 59 54 5C 53 45 50 59 54 54 53 52 14  0013E P.AAQ:   .ASCII  <20>\RSTTYPES\<92>\TYP_VARIANT\                ;
                        54 4E 41 49 52 41     0014D                                                               ;


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

                                    0004 00000          .ENTRY  DBG$STA_TYP_VARIANT, Save R2                    ; 1433
                              5E      08   C2 00002     SUBL2   #8, SP                                          ;
                                      5E   DD 00005     PUSHL   SP                                              ; 1495
                                  08  AE   9F 00007     PUSHAB  FCODE                                           ;
                                  04  AC   DD 0000A     PUSHL   TYPEID                                          ;
                          0000V   CF        03 FB 0000D     CALLS   #3, FIND_TYPE_RECORD                        ;
                                  52        50 D0 00012     MOVL    R0, TYPEID_PTR                              ;
                                  0B    04  AE D1 00015     CMPL    FCODE, #11                                  ; 1496
                                            15 13 00019     BEQL    1$                                          ;
                                00000000'   EF 9F 0001B     PUSHAB  P.AAQ                                       ;
                                            01 DD 00021     PUSHL   #1                                          ;
                                00028362    8F DD 00023     PUSHL   #164706                                     ;
                          00000000G   00        03 FB 00029     CALLS   #3, LIB$SIGNAL                          ;
                                08  BC    08  A2 D0 00030 1$:     MOVL    8(TYPEID_PTR), @NCOMPS              ; 1497
                                0C  BC    18  A2 9E 00035     MOVAB   24(R2), @COMPVECPTR                     ; 1498
                                10  BC    10  A2 D0 0003A     MOVL    16(TYPEID_PTR), @TAG                    ; 1499
                                14  BC        6E D0 0003F     MOVL    SIZE, @BITSIZE                          ; 1500
                                            04 00043        RET                                             ; 1503

; Routine Size:  68 bytes,    Routine Base:  DBG$CODE + 07D2
```

```
;  1390          1504  1 GLOBAL ROUTINE DBG$STA_TYP_VARIANT_COMP(LIST_ID, NCOMPS, COMPVECPTR,
;  1391          1505  1                                         NTAGS, TAGVECPTR, SIZE): NOVALUE =
;  1392          1506  1 !
;  1393          1507  1 ! FUNCTION
;  1394          1508  1 !       This routine accepts as input a pointer to a set of Variant fields,
;  1395          1509  1 !       and it returns the number of fields, the SYMID for each field, the number
;  1396          1510  1 !       of tag blocks, a value range for each tag block, and the bit length of
;  1397          1511  1 !       the variant.
;  1398          1512  1 !
;  1399          1513  1 ! INPUTS
;  1400          1514  1 !       LIST_ID - A pointer to a list of variant fields; this list was produced
;  1401          1515  1 !                 by DBG$STA_TYP_VARIANT and was pointed to by one element of
;  1402          1516  1 !                 the COMPVECPTR it returned.
;  1403          1517  1 !
;  1404          1518  1 !       NCOMPS  - The address of a longword location to receive the number of
;  1405          1519  1 !                 fields in this variant.
;  1406          1520  1 !
;  1407          1521  1 !       COMPVECPTR - The address of a longword location to receive a pointer to
;  1408          1522  1 !                 a vector of field SYMIDs.
;  1409          1523  1 !
;  1410          1524  1 !       NTAGS   - The address of a longword location to receive the number of
;  1411          1525  1 !                 tag blocks.
;  1412          1526  1 !
;  1413          1527  1 !       TAGVECPTR - The address of a longword location to receive a pointer to a
;  1414          1528  1 !                   vector of tag value ranges. An element of this vector is the
;  1415          1529  1 !                   address of a three longword block; each block has the following
;  1416          1530  1 !                   format:
;  1417          1531  1 !                       longword 1: dst$k_varval_single = single tag value
;  1418          1532  1 !                                                 or
;  1419          1533  1 !                                   dst$k_varval_range = range of tag values
;  1420          1534  1 !                       longword 2: pointer to low bound value of tag range
;  1421          1535  1 !                       longword 3: pointer to high bound value of tag range
;  1422          1536  1 !                                   (or zero if single value)
;  1423          1537  1 !
;  1424          1538  1 !       SIZE    - The address of a longword location to receive the size in bits
;  1425          1539  1 !                 of an item of this type.
;  1426          1540  1 !
;  1427          1541  1 ! OUTPUTS
;  1428          1542  1 !       NCOMPS  - The number of fields is returned to NCOMPS.
;  1429          1543  1 !
;  1430          1544  1 !       COMPVECPTR - A pointer to a vector of field SYMIDs is returned to
;  1431          1545  1 !                 COMPVECPTR.
;  1432          1546  1 !                 This vector disappears at the end of the current command.
;  1433          1547  1 !
;  1434          1548  1 !       NTAGS   - The number of tag blocks is returned to NTAGS.
;  1435          1549  1 !
;  1436          1550  1 !       SIZE    - The size in bits of this variant is returned to SIZE.
;  1437          1551  1 !
;  1438          1552  1 !       No value is returned.
;  1439          1553  1 !
;  1440          1554  1
;  1441          1555  2     BEGIN
;  1442          1556  2
;  1443          1557  2     MAP
;  1444          1558  2         LIST_ID: REF RST$VAR_ENTRY;     ! Pointer to Variant Component List Entry
;  1445          1559  2
;  1446          1560  2     LOCAL
```

```
;  1447          1561  2          TAG_LIST: REF RST$TAG_LIST,
;  1448          1562  2          TAG_PTR: REF VECTOR[,BYTE],
;  1449          1563  2          DST_PTR: REF DST$RECORD,
;  1450          1564  2          VALUE_SPEC: REF DST$VAL_SPEC,
;  1451          1565  2          VAL_LNGTH,
;  1452          1566  2          VAL_VECT: VECTOR[3],
;  1453          1567  2          VALKIND;
;  1454          1568  2
;  1455          1569  2
;  1456          1570  2
;  1457          1571  2      DST_PTR = .LIST_ID[RST$L_VAR_DSTPTR];
;  1458          1572  2      IF .DST_PTR[DST$B_TYPE] NEQ DST$K_VARVAL
;  1459          1573  2      THEN
;  1460          1574  2          $DBG_ERROR('RSTTYPES\VARIANT_COMP');
;  1461          1575  2
;  1462          1576  2      .NCOMPS = .LIST_ID[RST$L_VAR_COMPCNT];
;  1463          1577  2      .NTAGS = .DST_PTR[DST$W_VARVAL_COUNT];
;  1464          1578  2      .SIZE = .DST_PTR[DST$L_VARVAL_SIZE];
;  1465          1579  2      IF ..NCOMPS NEQ 0
;  1466          1580  2      THEN
;  1467          1581  2          .COMPVECPTR = LIST_ID[RST$A_VAR_COMPLST]
;  1468          1582  2      ELSE
;  1469          1583  2          .COMPVECPTR = 0;
;  1470          1584  2
;  1471          1585  2      TAG_LIST = DBG$GET_TEMPMEM(RST$K_TAG_BLOCK_SIZE * ..NTAGS);
;  1472          1586  2      TAG_PTR = DST_PTR[DST$A_VARVAL_RNGSPEC];
;  1473          1587  2      INCR I FROM 0 TO ..NTAGS - 1 DO
;  1474          1588  3          BEGIN
;  1475          1589  3          VALUE_SPEC = TAG_PTR[1];
;  1476          1590  3          TAG_LIST[.I,RST$L_TAG_NUMVALS] = .TAG_PTR[0];
;  1477          1591  3          DBG$STA_VALSPEC(.VALUE_SPEC, VAL_VECT, VALKIND);
;  1478          1592  3          TAG_LIST[.I,RST$L_TAG_LOWBOUND] = .VAL_VECT[0];
;  1479          1593  3          VAL_LNGTH = 5;
;  1480          1594  3          IF .VALUE_SPEC[DST$B_VS_VFLAGS] EQL DST$K_VS_FOLLOWS
;  1481          1595  3          THEN
;  1482          1596  3              VAL_LNGTH = .VALUE_SPEC[DST$W_VS_LENGTH] + 3;
;  1483          1597  3
;  1484          1598  3          IF .TAG_PTR[0] EQL DST$K_VARVAL_RANGE
;  1485          1599  3          THEN
;  1486          1600  4              BEGIN
;  1487          1601  4              VALUE_SPEC = .TAG_PTR + .VAL_LNGTH + 1;
;  1488          1602  4              DBG$STA_VALSPEC(.VALUE_SPEC, VAL_VECT, VALKIND);
;  1489          1603  4              TAG_LIST[.I,RST$L_TAG_HIGHBOUND] = .VAL_VECT[0];
;  1490          1604  4              VAL_LNGTH = 5;
;  1491          1605  4              IF .VALUE_SPEC[DST$B_VS_VFLAGS] EQL DST$K_VS_FOLLOWS
;  1492          1606  4              THEN
;  1493          1607  4                  VAL_LNGTH = .VALUE_SPEC[DST$W_VS_LENGTH] + 3;
;  1494          1608  4
;  1495          1609  4              TAG_PTR = .TAG_PTR + .VAL_LNGTH;
;  1496          1610  4              END
;  1497          1611  4
;  1498          1612  3          ELSE
;  1499          1613  3              TAG_PTR = .TAG_PTR + .VAL_LNGTH + 1;
;  1500          1614  3
;  1501          1615  2          END;
;  1502          1616  2
;  1503          1617  2      .TAGVECPTR = .TAG_LIST;
```

```
;  1504          1618  2     RETURN;
;  1505          1619  2
;  1506          1620  1     END;


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

41 49 52 41 56 5C 53 45 50 59 54 54 53 52 15  00153 P.AAR:  .ASCII  <21>\RSTTYPES\<92>\VARIANT_COMP\                ;
                        50 4D 4F 43 5F 54 4E  00162                                                              ;


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

                                          01FC 00000        .ENTRY  DBG$STA_TYP_VARIANT_COMP, Save R2,R3,R4,R5,-  ; 1504
                                                                    R6,R7,R8                                    ;
                                58 00000000G   00  9E 00002        MOVAB   DBG$STA_VALSPEC, R8                 ;
                                5E             10  C2 00009        SUBL2   #16, SP                             ;
                                53       04    AC  D0 0000C        MOVL    LIST_ID, R3                         ; 1571
                                52             63  D0 00010        MOVL    (R3), DST_PTR                       ;
                             9D 8F       01    A2  91 00013        CMPB    1(DST_PTR), #157                    ; 1572
                                            15  13 00018        BEQL    1$                                  ;
                                00000000'   EF  9F 0001A        PUSHAB  P.AAR                               ; 1574
                                            01  DD 00020        PUSHL   #1                                  ;
                                00028362    8F  DD 00022        PUSHL   #164706                             ;
                          00000000G  00     03  FB 00028        CALLS   #3, LIB$SIGNAL                      ;
                                08   BC     04  A3  D0 0002F 1$:  MOVL    4(R3), @NCOMPS                      ; 1576
                                10   BC     06  A2  3C 00034     MOVZWL  6(DST_PTR), @NTAGS                  ; 1577
                                18   BC     02  A2  D0 00039     MOVL    2(DST_PTR), @SIZE                   ; 1578
                                            08  BC  D5 0003E        TSTL    @NCOMPS                             ; 1579
                                            07  13 00041        BEQL    2$                                  ;
                                   0C   BC     08  A3  9E 00043     MOVAB   8(R3), @COMPVECPTR                  ; 1581
                                            03  11 00048        BRB     3$                                  ;
                                            0C  BC  D4 0004A 2$:  CLRL    @COMPVECPTR                         ; 1583
                             7E     10   BC     03  C5 0004D 3$:  MULL3   #3, @NTAGS, -(SP)                   ; 1585
                          00000000G  00     01  FB 00052        CALLS   #1, DBG$GET_TEMPMEM                 ;
                                55             50  D0 00059        MOVL    R0, TAG_LIST                        ;
                                53       08    A2  9E 0005C        MOVAB   8(R2), TAG_PTR                      ; 1586
                                54             01  CE 00060        MNEGL   #1, I                               ; 1590
                                            67  11 00063        BRB     8$                                  ;
                                52       01    A3  9E 00065 4$:  MOVAB   1(R3), VALUE_SPEC                   ; 1589
                             56 54             0C  C5 00069        MULL3   #12, I, R6                          ; 1590
                                          6645  9F 0006D        PUSHAB  (R6)[TAG_LIST]                      ;
                                9E             63  9A 00070        MOVZBL  (TAG_PTR), @(SP)+                   ;
                                               5E  DD 00073        PUSHL   SP                                  ; 1591
                                            08  AE  9F 00075        PUSHAB  VAL_VECT                            ;
                                               52  DD 00078        PUSHL   VALUE_SPEC                          ;
                                68             03  FB 0007A        CALLS   #3, DBG$STA_VALSPEC                 ;
                                         04 A645  9F 0007D        PUSHAB  4(R6)[TAG_LIST]                     ; 1592
                                9E       08    AE  D0 00081        MOVL    VAL_VECT, @(SP)+                    ;
                                57             05  D0 00085        MOVL    #5, VAL_LNGTH                       ; 1593
                             FD 8F             62  91 00088        CMPB    (VALUE_SPEC), #253                  ; 1594
                                               07  12 0008C        BNEQ    5$                                  ;
                                57       01    A2  3C 0008E        MOVZWL  1(VALUE_SPEC), VAL_LNGTH            ; 1596
                                57             03  C0 00092        ADDL2   #3, VAL_LNGTH                       ;
                                50       01 A743  9E 00095 5$:     MOVAB   1(VAL_LNGTH)[TAG_PTR], R0           ; 1601
```

```
                    02              63  91 0009A          CMPB    (TAG_PTR), #2                   ; 1598
                                    2A  12 0009D          BNEQ    7$                              ;
                    52              50  D0 0009F          MOVL    R0, VALUE_SPEC                  ; 1601
                                    5E  DD 000A2          PUSHL   SP                              ; 1602
                            08      AE  9F 000A4          PUSHAB  VAL_VECT                        ;
                                    52  DD 000A7          PUSHL   VALUE_SPEC                      ;
                    68              03  FB 000A9          CALLS   #3, DBG$STA_VALSPEC             ;
                            08 A645 9F  9F 000AC          PUSHAB  8(R6)[TAG_LIST]                 ; 1603
                    9E      08      AE  D0 000B0          MOVL    VAL_VECT, @(SP)+                ;
                    57              05  D0 000B4          MOVL    #5, VAL_LNGTH                   ; 1604
                FD  8F              62  91 000B7          CMPB    (VALUE_SPEC), #253              ; 1605
                                    07  12 000BB          BNEQ    6$                              ;
                    57      01      A2  3C 000BD          MOVZWL  1(VALUE_SPEC), VAL_LNGTH        ; 1607
                    57              03  C0 000C1          ADDL2   #3, VAL_LNGTH                   ;
                    53              57  C0 000C4 6$:      ADDL2   VAL_LNGTH, TAG_PTR              ; 1609
                                    03  11 000C7          BRB     8$                              ; 1598
                    53              50  D0 000C9 7$:      MOVL    R0, TAG_PTR                     ; 1613
                94                  54  10 BC F2 000CC 8$: AOBLSS  @NTAGS, I, 4$                  ; 1587
                14  BC              55  D0 000D1          MOVL    TAG_LIST, @TAGVECPTR            ; 1617
                                        04 000D5          RET                                     ; 1620

; Routine Size:  214 bytes,    Routine Base:  DBG$CODE + 0816
```

```
;  1508          1621  1 GLOBAL ROUTINE DBG$TRANS_TYPE_CODE(CODE) =
;  1509          1622  1 !
;  1510          1623  1 ! FUNCTION
;  1511          1624  1 !       This routine takes a DST type code and translates it to the
;  1512          1625  1 !       corresponding RST type code.
;  1513          1626  1 !
;  1514          1627  1 ! INPUTS
;  1515          1628  1 !       CODE    - The DST type code to be translated.
;  1516          1629  1 !
;  1517          1630  1 ! OUTPUTS
;  1518          1631  1 !       The corresponding RST type code is returned as the routine's value.
;  1519          1632  1 !
;  1520          1633  1
;  1521          1634  2     BEGIN
;  1522          1635  2
;  1523          1636  2     CASE .CODE FROM DST$K_TS_DTYPE_LOWEST TO DST$K_TS_DTYPE_HIGHEST OF
;  1524          1637  2         SET
;  1525          1638  2
;  1526          1639  2         [DST$K_TS_ATOM]:
;  1527          1640  2             RETURN RST$K_TYPE_ATOMIC;
;  1528          1641  2
;  1529          1642  2         [DST$K_TS_DSC]:
;  1530          1643  2             RETURN RST$K_TYPE_DESCR;
;  1531          1644  2
;  1532          1645  2         [DST$K_TS_TPTR]:
;  1533          1646  2             RETURN RST$K_TYPE_TPTR;
;  1534          1647  2
;  1535          1648  2         [DST$K_TS_PTR]:
;  1536          1649  2             RETURN RST$K_TYPE_PTR;
;  1537          1650  2
;  1538          1651  2         [DST$K_TS_PIC]:
;  1539          1652  2             RETURN RST$K_TYPE_PICT;
;  1540          1653  2
;  1541          1654  2         [DST$K_TS_ARRAY]:
;  1542          1655  2             RETURN RST$K_TYPE_ARRAY;
;  1543          1656  2
;  1544          1657  2         [DST$K_TS_SET]:
;  1545          1658  2             RETURN RST$K_TYPE_SET;
;  1546          1659  2
;  1547          1660  2         [DST$K_TS_SUBRANGE]:
;  1548          1661  2             RETURN RST$K_TYPE_SUBRNG;
;  1549          1662  2
;  1550          1663  2         [DST$K_TS_FILE]:
;  1551          1664  2             RETURN RST$K_TYPE_FILE;
;  1552          1665  2
;  1553          1666  2         [DST$K_TS_SELF_REL_LABEL]:
;  1554          1667  2             RETURN RST$K_TYPE_SELF_REL_LAB;
;  1555          1668  2
;  1556          1669  2         [DST$K_TS_RFA]:
;  1557          1670  2             RETURN RST$K_TYPE_RFA;
;  1558          1671  2
;  1559          1672  2         [DST$K_TS_AREA]:
;  1560          1673  2             RETURN RST$K_TYPE_AREA;
;  1561          1674  2
;  1562          1675  2         [DST$K_TS_OFFSET]:
;  1563          1676  2             RETURN RST$K_TYPE_OFFSET;
;  1564          1677  2
```

```
;  1565          1678  2          [DST$K_TS_TASK]:
;  1566          1679  2              RETURN RST$K_TYPE_TASK;
;  1567          1680  2
;  1568          1681  2          [INRANGE,OUTRANGE]:
;  1569          1682  2              RETURN 0;
;  1570          1683  2
;  1571          1684  2          TES;
;  1572          1685  2
;  1573          1686  1      END;


                                                0000 00000          .ENTRY  DBG$TRANS_TYPE_CODE, Save nothing                ; 1621
                      11            01      04   AC   CF 00002     CASEL   CODE, #1, #17                                    ; 1679
            002F      0024          002B          0027      00007 1$: .WORD   3$-1$,-
            003F      003B          0037          0033      0000F             4$-1$,-
            0053      0047          0024          0043      00017             2$-1$,-
            J04B      0024          0024          0057      0001F             5$-1$,-
                                    005B          004F      00027             6$-1$,-
                                                                              7$-1$,-
                                                                              8$-1$,-
                                                                              9$-1$,-
                                                                              10$-1$,-
                                                                              2$-1$,-
                                                                              11$-1$,-
                                                                              14$-1$,-
                                                                              15$-1$,-
                                                                              2$-1$,-
                                                                              2$-1$,-
                                                                              12$-1$,-
                                                                              13$-1$,-
                                                                              16$-1$
                            50  D4 0002B 2$:       CLRL    R0                                                       ; 1682
                                04 0002D          RET
                      50    02  D0 0002E 3$:      MOVL    #2, R0                                                   ; 1679
                                04 00031          RET
                      50    03  D0 00032 4$:      MOVL    #3, R0
                                04 00035          RET
                      50    06  D0 00036 5$:      MOVL    #6, R0
                                04 00039          RET
                      50    10  D0 0003A 6$:      MOVL    #16, R0
                                04 0003D          RET
                      50    05  D0 0003E 7$:      MOVL    #5, R0
                                04 00041          RET
                      50    01  D0 00042 8$:      MOVL    #1, R0
                                04 00045          RET
                      50    08  D0 00046 9$:      MOVL    #8, R0
                                04 00049          RET
                      50    09  D0 0004A 10$:     MOVL    #9, R0
                                04 0004D          RET
                      50    0F  D0 0004E 11$:     MOVL    #15, R0
                                04 00051          RET
                      50    15  D0 00052 12$:     MOVL    #21, R0
                                04 00055          RET
                      50    14  D0 00056 13$:     MOVL    #20, R0
```

```
                                04 00059                RET
                        50  11  D0 0005A 14$:           MOVL    #17, R0                 ;
                                04 0005D                RET                             ;
                        50  12  D0 0005E 15$:           MOVL    #18, R0                 ;
                                04 00061                RET                             ;
                        50  16  D0 00062 16$:           MOVL    #22, R0                 ;
                                04 00065                RET                             ; 1686

; Routine Size:  102 bytes,    Routine Base:  DBG$CODE + 08EC
```

```
;1575     1687  1 GLOBAL ROUTINE DBG$TYPEID_FOR_ARRAY(TYPEID, ADDRESS) =
;1576     1688  1 !
;1577     1689  1 ! FUNCTION
;1578     1690  1 !       This routine is called from DBGPARSER in the process of parsing
;1579     1691  1 !       a C expression of the form PTR[n], where PTR was declared as
;1580     1692  1 !       a pointer. Since arrays and pointers are treated the same in C,
;1581     1693  1 !       we want to convert the pointer typeid to an equivalent array
;1582     1694  1 !       typeid.
;1583     1695  1 !
;1584     1696  1 ! INPUT
;1585     1697  1 !       TYPEID  - The RST entry describing a pointer data type
;1586     1698  1 !       ADDRESS - The value of the pointer (which will be the base
;1587     1699  1 !                 address of the array).
;1588     1700  1 !
;1589     1701  1 ! OUTPUT
;1590     1702  1 !       A new typeid is constructed out of temporary memory, and
;1591     1703  1 !       a pointer to this typeid is returned.
;1592     1704  1 !
;1593     1705  1 ! Example:
;1594     1706  1 !       REAL *RPTR;
;1595     1707  1 !       If RPTR is declared as a pointer to a floating number,
;1596     1708  1 !       then if we pass in the TYPEID for RPTR, this routine returns
;1597     1709  1 !       a typeid for an array of floating numbers (e.g., as if we
;1598     1710  1 !       had declared an object
;1599     1711  1 !       REAL RARRAY[n];
;1600     1712  1 !
;1601     1713  2     BEGIN
;1602     1714  2     MAP
;1603     1715  2         TYPEID: REF RST$ENTRY;          ! The input TYPEID for the Type Pointer
;1604     1716  2                                         !      data type
;1605     1717  2
;1606     1718  2     LOCAL
;1607     1719  2         ADDRPTR,                        ! Pointer into array descriptor
;1608     1720  2         ARG_DSTPTR: REF DST$RECORD,     ! Pointer to DST record for argument
;1609     1721  2         ARG_TYPEID: REF RST$ENTRY,      ! Pointer to RST record for pointed-to
;1610     1722  2                                         !   object.
;1611     1723  2         BITSIZE,                        ! Bitsize of pointed-to object
;1612     1724  2         BOUNDVEC: REF VECTOR[],         ! Pointer to array bounds
;1613     1725  2         DSCADDR: REF BLOCK[,BYTE],      ! Pointer to array descriptor
;1614     1726  2         DSTPTR: REF DST$RECORD,         ! Pointer to dummy DST record
;1615     1727  2         FLAGS_PTR: REF BITVECTOR[],     ! "flags" field withing array type spec
;1616     1728  2         RSTPTR: REF RST$ENTRY,          ! Pointer to new RST record
;1617     1729  2         TSPTR: REF DST$TYPE_SPEC,       ! Pointer to type spec within DST record
;1618     1730  2         TSPTR1: REF DST$TYPE_SPEC,      ! Pointer to type spec within DST record
;1619     1731  2         TSPTR2: REF DST$TYPE_SPEC,      ! Pointer to type spec within DST record
;1620     1732  2         VSPTR: REF DST$VAL_SPEC;        ! Pointer to val spec within DST record
;1621     1733  2
;1622     1734  2
;1623     1735  2     ! Obtain a TYPEID and a DSTPTR for the pointed-to object.
;1624     1736  2     !
;1625     1737  2     DBG$STA_TYP_TYPEDPTR (.TYPEID, ARG_TYPEID);
;1626     1738  2     DBG$STA_SYMSIZE (.ARG_TYPEID, BITSIZE);
;1627     1739  2     ARG_DSTPTR = .ARG_TYPEID[RST$L_DSTPTR];
;1628     1740  2
;1629     1741  2     ! Allocate a memory block large enough to hold the newly created
;1630     1742  2     ! RST record, the dummy DST record that it will point to, and the
;1631     1743  2     ! array descriptor for this dummy array we are creating.
```

```
;  1632          1744  2     !
;  1633          1745  2     RSTPTR = DBG$GET_MEMORY (RST$K_TYPENTSIZ + 14);
;  1634          1746  2     DSTPTR = .RSTPTR + 4*RST$K_TYPENTSIZ;
;  1635          1747  2
;  1636          1748  2     ! Fill in some of the fields of the RST record.
;  1637          1749  2     !
;  1638          1750  2     RSTPTR[RST$L_DSTPTR] = .DSTPTR;
;  1639          1751  2     RSTPTR[RST$L_UPSCOPEPTR] = .RST$START_ADDR;
;  1640          1752  2     RSTPTR[RST$B_KIND] = RST$K_TYPE;
;  1641          1753  2     RSTPTR[RST$B_FCODE] = RST$R_TYPE_ARRAY;
;  1642          1754  2
;  1643          1755  2     ! Put the RST record on the temporary RST list. This will ensure that
;  1644          1756  2     ! the space gets released when there are no more references to it.
;  1645          1757  2     !
;  1646          1758  2     RSTPTR[RST$L_HASH_FLINK] = .RST$TEMP_LIST;
;  1647          1759  2     RST$TEMP_LIST = .RSTPTR;
;  1648          1760  2
;  1649          1761  2     ! Fill in the header of the dummy DST record.
;  1650          1762  2     !
;  1651          1763  2     DSTPTR[DST$B_LENGTH] = 23;
;  1652          1764  2     DSTPTR[DST$B_TYPE] = DST$K_TYPSPEC;
;  1653          1765  2     DSTPTR[DST$B_TYPSPEC_NAME] = 0;
;  1654          1766  2
;  1655          1767  2     ! Fill in the Type Spec record within the dummy DST record.
;  1656          1768  2     ! This describes the array as a whole.
;  1657          1769  2     !
;  1658          1770  2     TSPTR = .DSTPTR + 3;
;  1659          1771  2     TSPTR[DST$W_TS_LENGTH] = 19;
;  1660          1772  2     TSPTR[DST$B_TS_KIND] = DST$K_TS_ARRAY;
;  1661          1773  2     TSPTR[DST$B_TS_ARRAY_DIM] = 1;
;  1662          1774  2     FLAGS_PTR = TSPTR[DST$A_TS_ARRAY_FLAGS_ADDR];
;  1663          1775  2     FLAGS_PTR[0] = TRUE;
;  1664          1776  2
;  1665          1777  2     ! Fill in the Val Spec record within the dummy DST record.
;  1666          1778  2     ! This points to the array descriptor that we will construct
;  1667          1779  2     ! below this dummy DST record.
;  1668          1780  2     !
;  1669          1781  2     VSPTR =  FLAGS_PTR + 1;
;  1670          1782  2     VSPTR[DST$B_VS_VFLAGS] = DST$K_VFLAGS_DSC;
;  1671          1783  2     VSPTR[DST$L_VS_DSC_OFFS] = 11;
;  1672          1784  2
;  1673          1785  2     ! Fill in the celltype type spec record within the dummy DST record.
;  1674          1786  2     ! This describes the type of each element of the array. This type
;  1675          1787  2     ! should be the same as the type of the pointed-to object, so
;  1676          1788  2     ! we use an indirect type specification, and point to the DST
;  1677          1789  2     ! we obtained from the argument.
;  1678          1790  2     !
;  1679          1791  2     TSPTR1 = .VSPTR + 5;
;  1680          1792  2     TSPTR1[DST$W_TS_LENGTH] = 5;
;  1681          1793  2     TSPTR1[DST$B_TS_KIND] = DST$K_TS_IND;
;  1682          1794  2     TSPTR1[DST$L_TS_IND_PTR] = DBG$RST_DST_PTR( .ARG_DSTPTR - .DST$BEGIN_ADDR );
;  1683          1795  2
;  1684          1796  2
;  1685          1797  2     ! Fill in the type specification for the type of the subscript.
;  1686          1798  2     ! This type will just be longword integer.
;  1687          1799  2     !
;  1688          1800  2     TSPTR2 = .TSPTR1 + .TSPTR1[DST$W_TS_LENGTH] + 2;
```

```
;  1689          1801  2      TSPTR2[DST$W_TS_LENGTH] = 2;
;  1690          1802  2      TSPTR2[DST$B_TS_KIND] = DST$K_TS_ATOM;
;  1691          1803  2      TSPTR2[DST$B_TS_ATOM_TYP] = DSC$K_DTYPE_L;
;  1692          1804  2
;  1693          1805  2      ! Fill in the array descriptor.
;  1694          1806  2      !
;  1695          1807  2      DSCADDR = .TSPTR2 + .TSPTR2[DST$W_TS_LENGTH] + 2;
;  1696          1808  2      DSCADDR[DSC$B_CLASS] = DSC$K_CLASS_A;
;  1697          1809  2      DSCADDR[DSC$W_LENGTH] = .BITSIZE/8;
;  1698          1810  2      DSCADDR[DSC$A_POINTER] = .ADDRESS;
;  1699          1811  2      DSCADDR[DSC$B_DIMCT] = 1;
;  1700          1812  2      DSCADDR[DSC$V_FL_COEFF] = TRUE;
;  1701          1813  2      DSCADDR[DSC$V_FL_BOUNDS] = TRUE;
;  1702          1814  2      ADDRPTR = .DSCADDR + 16;
;  1703          1815  2      .ADDRPTR = .ADDRESS;
;  1704          1816  2      BOUNDVEC = .DSCADDR + 24;
;  1705          1817  2      BOUNDVEC[0] = 0;
;  1706          1818  2      BOUNDVEC[1] = 2000000000;
;  1707          1819  2
;  1708          1820  2      ! Return the pointer to the RST record.
;  1709          1821  2      !
;  1710          1822  2      RETURN .RSTPTR;
;  1711          1823  1      END;


                                003C 00000     .ENTRY  DBG$TYPEID_FOR_ARRAY, Save R2,R3,R4,R5      ; 1687
                     55 00000000G   00  9E 00002     MOVAB   RST$TEMP_LIST, R5                           ;
                     5E             08  C2 00009     SUBL2   #8, SP                                      ;
                                    5E  DD 0000C     PUSHL   SP                                          ; 1737
                           04       AC  DD 0000E     PUSHL   TYPEID                                      ;
                FB4B       CF       02  FB 00011     CALLS   #2, DBG$STA_TYP_TYPEDPTR                    ;
                           04       AE  9F 00016     PUSHAB  BITSIZE                                     ; 1738
                     52    04       AE  D0 00019     MOVL    ARG_TYPEID, R2                              ;
                                    52  DD 0001D     PUSHL   R2                                          ;
                F75F       CF       02  FB 0001F     CALLS   #2, DBG$STA_SYMSIZE                         ;
                     54    0C       A2  D0 00024     MOVL    12(R2), ARG_DSTPTR                          ; 1739
                                    19  DD 00028     PUSHL   #25                                         ; 1745
          00000000G   00             01  FB 0002A     CALLS   #1, DBG$GET_MEMORY                          ;
                     53             50  D0 00031     MOVL    R0, RSTPTR                                  ;
                     50    2C       A3  9E 00034     MOVAB   44(R3), DSTPTR                              ; 1746
                0C   A3             50  D0 00038     MOVL    DSTPTR, 12(RSTPTR)                          ; 1750
                10   A3 00000000G   00  D0 0003C     MOVL    RST$START_ADDR, 16(RSTPTR)                  ; 1751
                14   A3             07  90 00044     MOVB    #7, 20(RSTPTR)                              ; 1752
                18   A3             01  90 00048     MOVB    #1, 24(RSTPTR)                              ; 1753
                     63             65  D0 0004C     MOVL    RST$TEMP_LIST, (RSTPTR)                     ; 17[illegible]
                     65             53  D0 0004F     MOVL    RSTPTR, RST$TEMP_LIST                       ; 175[illegible]
                     80       AF17  8F  B0 00052     MOVW    #44823, (DSTPTR)+                           ; 1763
                                    80  94 00057     CLRB    (DSTPTR)+                                   ; 1765
                     80 01070013    8F  D0 00059     MOVL    #17235987, (TSPTR)+                         ; 1771
                     80             01  88 00060     BISB2   #1, (FLAGS_PTR)+                            ; 1775
                     60             06  8E 00063     MNEGB   #6, (VSPTR)                                 ; 1782
                01   A0             0B  D0 00066     MOVL    #11, 1(VSPTR)                               ; 1783
                     52       05    A0  9E 0006A     MOVAB   5(R0), TSPTR1                               ; 1791
                     62             05  B0 0006E     MOVW    #5, (TSPTR1)                                ; 1792
```

```
                02   A2                 03   90 00071        MOVB    #3, 2(TSPTR1)                                   ; 1793
             7E      54 00000000G       00   C3 00075        SUBL3   DST$BEGIN_ADDR, ARG_DSTPTR, -(SP)            ; 1794
    00000000G        00                 01   FB 0007D        CALLS   #1, DBG$RST_DST_PTR                         ;
                03   A2                 50   D0 00084        MOVL    R0, 3(TSPTR1)                               ;
                50                      62   3C 00088        MOVZWL  (TSPTR1), R0                                ; 1800
                50              02 A042  9E   8B             MOVAB   2(R0)[TSPTR1], TSPTR2                       ;
                60 08010002      8F   D0 00090               MOVL    #134283266, (TSPTR2)                        ; 1801
                51                      60   3C 00097        MOVZWL  (TSPTR2), R1                                ; 1807
                50              02 A140  9E   0009A          MOVAB   2(R1)[TSPTR2], DSCADDR                      ;
                03   A0                 04   90 0009F        MOVB    #4, 3(DSCADDR)                              ; 1808
             51 04   AE                 08   C7 000A3        DIVL3   #8, BITSIZE, R1                             ; 1809
                60                      51   B0 000A8        MOVW    R1, (DSCADDR)                               ;
                04   A0            08   AC   D0 000AB        MOVL    ADDRESS, 4(DSCADDR)                         ; 1810
                0B   A0                 01   90 000B0        MOVB    #1, 11(DSCADDR)                             ; 1811
                0A   A0            C0   8F   88 000B4        BISB2   #192, 10(DSCADDR)                           ; 1813
                51              10   A0   9E 000B9           MOVAB   16(R0), ADDRPTR                             ; 1814
                61              08   AC   D0 000BD           MOVL    ADDRESS, (ADDRPTR)                          ; 1815
                50                      18   C0 000C1        ADDL2   #24, BOUNDVEC                               ; 1816
                                        60   D4 000C4        CLRL    (BOUNDVEC)                                  ; 1817
                04   A0 77359400      8F   D0 000C6          MOVL    #2000000000, 4(BOUNDVEC)                    ; 1818
                50                      53   D0 000CE        MOVL    RSTPTR, R0                                  ; 1822
                                        04   000D1          RET                                                 ; 1823

; Routine Size:  210 bytes,    Routine Base:  DBG$CODE + 0952
```

```
;1713     1824  1 GLOBAL ROUTINE DBG$TYPEID_FOR_ATOMIC(TYPECODE, BITSIZE, DSC_FLAG) =
;1714     1825  1 !
;1715     1826  1 ! FUNCTION
;1716     1827  1 !       This routine returns the Type ID associated with a specified VAX stand-
;1717     1828  1 !       ard type code (a "one-byte type code").  The Type ID is a pointer to an
;1718     1829  1 !       RST entry for the data type.  This RST entry is added to the Temporary
;1719     1830  1 !       RST Entry List so that it disappears as soon as there are no longer any
;1720     1831  1 !       references to it.
;1721     1832  1 !
;1722     1833  1 !       The routine builds a Data Type RST Entry and a Type Specification DST
;1723     1834  1 !       record, both in the same memory block.  The RST entry's FCODE is set to
;1724     1835  1 !       be atomic and its DST pointer points to the DST record.  The DST record
;1725     1836  1 !       is built to contain a DST Type Specification for the specified atomic
;1726     1837  1 !       data type.
;1727     1838  1 !
;1728     1839  1 ! INPUTS
;1729     1840  1 !       TYPECODE - The VAX standard type code ("one-byte" type code) whose Type
;1730     1841  1 !                  ID is to be returned.
;1731     1842  1 !
;1732     1843  1 !       BITSIZE  - The size in bits of an item of this type.
;1733     1844  1 !
;1734     1845  1 !       DSC_FLAG - A flag which, when set, indicates that we are processing a
;1735     1846  1 !                  descriptor item; thus, we will signal INVALID ARRAY DESC
;1736     1847  1 !                  in certain error situations.
;1737     1848  1 !
;1738     1849  1 ! OUTPUTS
;1739     1850  1 !       The corresponding Type ID (Type RST Entry pointer) is returned as the
;1740     1851  1 !               routine value.
;1741     1852  1 !
;1742     1853  1
;1743     1854  2     BEGIN
;1744     1855  2
;1745     1856  2     LOCAL
;1746     1857  2         DSTPTR: REF DST$RECORD,         ! Pointer to the generated DST record
;1747     1858  2         RSTPTR: REF RST$ENTRY,          ! Pointer to the Type RST Entry
;1748     1859  2         TSPTR: REF DST$TYPE_SPEC;       ! Pointer to Type Spec in DST record
;1749     1860  2
;1750     1861  2
;1751     1862  2
;1752     1863  2     ! Make sure the VAX standard type code is in the valid range.
;1753     1864  2     !
;1754     1865  3     IF ((.TYPECODE LSS DSC$K_DTYPE_LOWEST) OR
;1755     1866  2         (.TYPECODE GTR DSC$K_DTYPE_HIGHEST)) AND
;1756     1867  2        (.TYPECODE NEQ DST$K_BOOL) AND
;1757     1868  3        (.TYPECODE NEQ DSC$K_DTYPE_TF)
;1758     1869  2     THEN
;1759     1870  3         BEGIN
;1760     1871  3         IF .DSC_FLAG
;1761     1872  3         THEN
;1762     1873  3             SIGNAL(DBG$_INVARRDSC)
;1763     1874  3
;1764     1875  3         ELSE
;1765     1876  3             $DBG_ERROR('RSTTYPES\TYPE!D_FOR_ATOMIC');
;1766     1877  3
;1767     1878  2         END;
;1768     1879  2
;1769     1880  2
```

```
;  1770          1881  2      ! Get a memory block large enough to contain both the RST entry and the
;  1771          1882  2      ! dummy DST record.
;  1772          1883  2      !
;  1773          1884  2      RSTPTR = DBG$GET_MEMORY(RST$K_TYPENTSIZ + 2);
;  1774          1885  2      DSTPTR = .RSTPTR + 4*RST$K_TYPENTSIZ;
;  1775          1886  2
;  1776          1887  2
;  1777          1888  2      ! Build the Data Type RST Entry for the atomic type.
;  1778          1889  2      !
;  1779          1890  2      RSTPTR[RST$L_DSTPTR] = .DSTPTR;
;  1780          1891  2      RSTPTR[RST$L_UPSCOPEPTR] = .RST$START_ADDR;
;  1781          1892  2      RSTPTR[RST$B_KIND] = RST$K_TYPE;
;  1782          1893  2      RSTPTR[RST$B_FCODE] = RST$K_TYPE_ATOMIC;
;  1783          1894  2      RSTPTR[RST$L_BITSIZE] = .BITSIZE;
;  1784          1895  2
;  1785          1896  2
;  1786          1897  2      ! Put this Data Type RST Entry on the Temporary RST Entry List.  This will
;  1787          1898  2      ! cause the entry to be released at the end of a DEBUG command when there
;  1788          1899  2      ! are no references to it (when it is not locked).
;  1789          1900  2      !
;  1790          1901  2      RSTPTR[RST$L_HASH_FLINK] = .RST$TEMP_LIST;
;  1791          1902  2      RST$TEMP_LIST = .RSTPTR;
;  1792          1903  2
;  1793          1904  2
;  1794          1905  2      ! Build the dummy Type Specification DST entry for the atomic type.
;  1795          1906  2      !
;  1796          1907  2      DSTPTR[DST$B_LENGTH] = 6;
;  1797          1908  2      DSTPTR[DST$B_TYPE] = DST$K_TYPSPEC;
;  1798          1909  2      DSTPTR[DST$B_TYPSPEC_NAME] = 0;
;  1799          1910  2      TSPTR = .DSTPTR + 3;
;  1800          1911  2      TSPTR[DST$W_TS_LENGTH] = 2;
;  1801          1912  2      TSPTR[DST$B_TS_KIND] = DST$K_TS_ATOM;
;  1802          1913  2      TSPTR[DST$B_TS_ATOM_TYP] = .TYPECODE;
;  1803          1914  2
;  1804          1915  2
;  1805          1916  2      ! Return the type's Type ID (Data Type RST Entry address) to the caller.
;  1806          1917  2      !
;  1807          1918  2      RETURN .RSTPTR;
;  1808          1919  2
;  1809          1920  1      END;


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

49  45  50  59  54  5C  53  45  50  59  54  54  53  52  1A  00169 P.AAS:  .ASCII  <26>\RSTTYPES\<92>\TYPEID_FOR_ATOMIC\        ;
        43  49  4D  4F  54  41  5F  52  4F  46  5F  44  00178                                                                   ;


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

                                                  001C 00000    .ENTRY  DBG$TYPEID_FOR_ATOMIC, Save R2,R3,R4                    ; 1824
                                54 00000000G   00   9E 00002    MOVAB   LIB$SIGNAL, R4                                          ;
                                53 00000000G   00   9E 00009    MOVAB   RST$TEMP_LIST, R3                                       ;
                                52         04   AC   D0 00010    MOVL    TYPECODE, R2                                            ; 1865
                                                  05   15 00014    BLEQ    1$                                                      ;
```

```
                    25              52  D1 00016          CMPL    R2, #37                         ; 1866
                                    2E  15 00019          BLEQ    3$
            0000009E    8F          52  D1 0001B 1$:      CMPL    R2, #158                        ; 1867
                                    25  13 00022          BEQL    3$
                    28              52  D1 00024          CMPL    R2, #40                         ; 1868
                                    20  13 00027          BEQL    3$
                    0B      0C      AC  E9 00029          BLBC    DSC_FLAG, 2$                    ; 1871
                        00028198    8F  DD 0002D          PUSHL   #164248                         ; 1873
                    64              01  FB 00033          CALLS   #1, LIB$SIGNAL
                                    11  11 00036          BRB     3$
                        00000000'   EF  9F 00038 2$:      PUSHAB  P.AAS                           ; 1876
                                    01  DD 0003E          PUSHL   #1
                        00028362    8F  DD 00040          PUSHL   #164706
                    64              03  FB 00046          CALLS   #3, LIB$SIGNAL
                                    0D  DD 00049 3$:      PUSHL   #13                             ; 1884
            00000000G   00          01  FB 0004B          CALLS   #1, DBG$GET_MEMORY
                    51      2C      A0  9E 00052          MOVAB   44(R0), DSTPTR                  ; 1885
                0C  A0              51  D0 00056          MOVL    DSTPTR, 12(RSTPTR)              ; 1890
                10  A0  00000000G   00  D0 0005A          MOVL    RST$START_ADDR, 16(RSTPTR)      ; 1891
                14  A0              07  90 00062          MOVB    #7, 20(RSTPTR)                  ; 1892
                18  A0              02  90 00066          MOVB    #2, 24(RSTPTR)                  ; 1893
                20  A0      08      AC  D0 0006A          MOVL    BITSIZE, 32(RSTPTR)             ; 1894
                    60              63  D0 0006F          MOVL    RST$TEMP_LIST, (RSTPTR)         ; 1901
                    63              50  D0 00072          MOVL    RSTPTR, RST$TEMP_LIST           ; 1902
                    81      AF06    8F  B0 00075          MOVW    #44806, (DSTPTR)+               ; 1907
                                    81  94 0007A          CLRB    (DSTPTR)+                       ; 1909
                    61              02  B0 0007C          MOVW    #2, (TSPTR)                     ; 1911
                02  A1              01  90 0007F          MOVB    #1, 2(TSPTR)                    ; 1912
                03  A1              52  90 00083          MOVB    R2, 3(TSPTR)                    ; 1913
                                        04 00087          RET                                     ; 1920

; Routine Size:  136 bytes,    Routine Base:  DBG$CODE + 0A24
```

```
;  1811          1921  1 GLOBAL ROUTINE DBG$TYPEID_FOR_SET(ARG1, FCODE, BITSIZE) =
;  1812          1922  1 !
;  1813          1923  1 !  FUNCTION
;  1814          1924  1 !        This routine builds a Data Type RST Entry for a set data type.
;  1815          1925  1 !        This routine is normally called when a Type ID (i.e., an RST
;  1816          1926  1 !        pointer to a Type RST Entry) must be produced for a set constant
;  1817          1927  1 !        data type.  The RST entry is added to the Temporary RST Entry List so
;  1818          1928  1 !        that it is released back to the memory pool once there are no longer
;  1819          1929  1 !        any references to it.
;  1820          1930  1 !
;  1821          1931  1 !        The routine builds a Data Type RST Entry and a dummy DST record in the
;  1822          1932  1 !        same memory block.  The DST record is a Type Specification DST record
;  1823          1933  1 !        with the null name and an Set Type Specification.  Parent Type
;  1824          1934  1 !        Specification in the Set Type Specification can be either an atomic
;  1825          1935  1 !        type or an Indirect Type Specification pointing to the DST Type
;  1826          1936  1 !        Specification given as the input parameter.
;  1827          1937  1 !
;  1828          1938  1 !  INPUTS
;  1829          1939  1 !        ARG1    - DTYPE or DSTADDR
;  1830          1940  1 !                  Dtype is vax standard data type.
;  1831          1941  1 !                  Dstaddr is the address of the DST Parent Type Specification
;  1832          1942  1 !                  for which a Set Type ID is to be generated.
;  1833          1943  1 !
;  1834          1944  1 !        FCODF   - The type Format Code associated with the Type Specification.
;  1835          1945  1 !
;  1836          1946  1 !        BITSIZE - The size in bits of an item of this type.
;  1837          1947  1 !
;  1838          1948  1 !        Optional 4th argument - If presents, that means the 1st argument is
;  1839          1949  1 !                                passed in as Dtype.  An Atomic parent type
;  1840          1950  1 !                                specification will be generated.
;  1841          1951  1 !
;  1842          1952  1 !  OUTPUTS
;  1843          1953  1 !        The data type's Type ID (i.e., a pointer to the Type RST Entry) is
;  1844          1954  1 !                returned as the routine's value.
;  1845          1955  1 !
;  1846          1956  1
;  1847          1957  2     BEGIN
;  1848          1958  2
;  1849          1959  2     BUILTIN
;  1850          1960  2         ACTUALCOUNT;
;  1851          1961  2
;  1852          1962  2     LOCAL
;  1853          1963  2         DSTADDR,                            ! Parent type DST pointer
;  1854          1964  2         DTYPE: BYTE,                        ! Data type
;  1855          1965  2         DSTPTR: REF DST$RECORD,             ! Pointer to generated DST record
;  1856          1966  2         PARENT_TSPTR: REF DST$TYPE_SPEC,    ! Pointer to Parent Type Spec in Set
;  1857          1967  2                                             !       DST record
;  1858          1968  2         RSTPTR: REF RST$ENTRY,              ! Pointer to generated Type RST Entry
;  1859          1969  2         TSPTR: REF DST$TYPE_SPEC;           ! Pointer to Type Spec in DST record
;  1860          1970  2
;  1861          1971  2
;  1862          1972  2
;  1863          1973  2     ! Depending on whether we have a fourth argument to this routine or not,
;  1864          1974  2     ! the first argument is either a DTYPE or a DSTADDR.  Pick up the Data
;  1865          1975  2     ! Type or the Parent type DST pointer accordingly.
;  1866          1976  2     !
;  1867          1977  2     IF ACTUALCOUNT() GTR 3 THEN DTYPE = .ARG1 ELSE DSTADDR = .ARG1;
```

```
;  1868          1978  2
;  1869          1979  2
;  1870          1980  2      ! Get a memory block large enough to accommodate both the RST entry and
;  1871          1981  2      ! the dummy DST record.
;  1872          1982  2      !
;  1873          1983  2      RSTPTR = DBG$GET_MEMORY(RST$K_TYPENTSIZ + 5);
;  1874          1984  2      DSTPTR = .RSTPTR + 4*RST$K_TYPENTSIZ;
;  1875          1985  2
;  1876          1986  2
;  1877          1987  2      ! Build the Data Type RST Entry for the input type.
;  1878          1988  2      !
;  1879          1989  2      RSTPTR[RST$L_DSTPTR] = .DSTPTR;
;  1880          1990  2      RSTPTR[RST$L_UPSCOPEPTR] = .RST$START_ADDR;
;  1881          1991  2      RSTPTR[RST$B_KIND] = RST$K_TYPE;
;  1882          1992  2      RSTPTR[RST$B_FCODE] = .FCODE;
;  1883          1993  2      RSTPTR[RST$L_BITSIZE] = .BITSIZE;
;  1884          1994  2
;  1885          1995  2
;  1886          1996  2      ! Put this Data Type RST Entry on the Temporary RST Entry List.  This will
;  1887          1997  2      ! cause the entry to be released at the end of a DEBUG command when there
;  1888          1998  2      ! are no references to it (when it is not locked).
;  1889          1999  2      !
;  1890          2000  2      RSTPTR[RST$L_HASH_FLINK] = .RST$TEMP_LIST;
;  1891          2001  2      RST$TEMP_LIST = .RSTPTR;
;  1892          2002  2
;  1893          2003  2
;  1894          2004  2      ! Build the dummy DST record for the type.
;  1895          2005  2      !
;  1896          2006  2      DSTPTR[DST$B_LENGTH] = 4 ;
;  1897          2007  2      DSTPTR[DST$B_TYPE] = DST$K_TYPSPEC;
;  1898          2008  2      DSTPTR[DST$B_TYPSPEC_NAME] = 0;
;  1899          2009  2      TSPTR = .DSTPTR + 3;
;  1900          2010  2      TSPTR[DST$W_TS_LENGTH] = 5;
;  1901          2011  2      TSPTR[DST$B_TS_KIND] = DST$K_TS_SET;
;  1902          2012  2      TSPTR[DST$L_TS_SET_LENG] = .BITSIZE;
;  1903          2013  2      PARENT_TSPTR = .TSPTR + 7;
;  1904          2014  2      IF ACTUALCOUNT() GTR 3
;  1905          2015  2      THEN
;  1906          2016  3          BEGIN
;  1907          2017  3          TSPTR[DST$W_TS_LENGTH] = .TSPTR[DST$W_TS_LENGTH] + DST$K_TS_ATOM_LENG;
;  1908          2018  3          PARENT_TSPTR[DST$W_TS_LENGTH] = 2;
;  1909          2019  3          PARENT_TSPTR[DST$B_TS_KIND] = DST$K_TS_ATOM;
;  1910          2020  3          PARENT_TSPTR[DST$B_TS_ATOM_TYP] = .DTYPE;
;  1911          2021  3          END
;  1912          2022  3
;  1913          2023  2      ELSE
;  1914          2024  3          BEGIN
;  1915          2025  3          TSPTR[DST$W_TS_LENGTH] = .TSPTR[DST$W_TS_LENGTH] + DST$K_TS_IND_LENG;
;  1916          2026  3          PARENT_TSPTR[DST$W_TS_LENGTH] = 5;
;  1917          2027  3          PARENT_TSPTR[DST$B_TS_KIND] = DST$K_TS_IND;
;  1918          2028  3          PARENT_TSPTR[DST$L_TS_IND_PTR] = DBG$RST_DST_PTR( .DSTADDR - .DST$BEGIN_ADDR );
;  1919          2029  2          END;
;  1920          2030  2
;  1921          2031  2      DSTPTR[DST$B_LENGTH] = .DSTPTR[DST$B_LENGTH] + .TSPTR[DST$W_TS_LENGTH];
;  1922          2032  2
;  1923          2033  2
;  1924          2034  2      ! Return the RST pointer to the caller.
```

```
;  1925          2035  2     !
;  1926          2036  2     RETURN .RSTPTR;
;  1927          2037  2
;  1928          2038  1     END;


                                        01FC 00000          .ENTRY  DBG$TYPEID_FOR_SET, Save R2,R3,R4,R5,R6,R7,-; 1921
                                                                    R8
                    58 00000000G    00  9E 00002          MOVAB   RST$TEMP_LIST, R8
                    03              6C  91 00009          CMPB    (AP), #3                                          ; 1977
                                    06  1B 0000C          BLEQU   1$
                    56        04    AC  90 0000E          MOVB    ARG1, DTYPE
                                    04  11 00012          BRB     2$
                    57        04    AC  D0 00014 1$:      MOVL    ARG1, DSTADDR
                                    10  DD 00018 2$:      PUSHL   #16                                               ; 1983
              00000000G     00      01  FB 0001A          CALLS   #1, DBG$GET_MEMORY
                    54              50  D0 00021          MOVL    R0, RSTPTR
                    55        2C    A4  9E 00024          MOVAB   44(R4), DSTPTR                                    ; 1984
                 0C A4              55  D0 00028          MOVL    DSTPTR, 12(RSTPTR)                                ; 1989
                 10 A4 00000000G    00  D0 0002C          MOVL    RST$START_ADDR, 16(RSTPTR)                        ; 1990
                 14 A4              07  90 00034          MOVB    #7, 20(RSTPTR)                                    ; 1991
                 18 A4        08    AC  90 00038          MOVB    FCODE, 24(RSTPTR)                                 ; 1992
                 20 A4        0C    AC  D0 0003D          MOVL    BITSIZE, 32(RSTPTR)                               ; 1993
                    64              68  D0 00042          MOVL    RST$TEMP_LIST, (RSTPTR)                           ; 2000
                    68              54  D0 00045          MOVL    RSTPTR, RST$TEMP_LIST                             ; 2001
                    65      AF04    8F  B0 00048          MOVW    #44804, (DSTPTR)                                  ; 2006
                              02    A5  94 0004D          CLRB    2(DSTPTR)                                         ; 2008
                    53        03    A5  9E 00050          MOVAB   3(R5), TSPTR                                      ; 2009
                    63              05  B0 00054          MOVW    #5, (TSPTR)                                       ; 2010
                 02 A3              08  90 00057          MOVB    #8, 2(TSPTR)                                      ; 2011
                 03 A3        0C    AC  D0 0005B          MOVL    BITSIZE, 3(TSPTR)                                 ; 2012
                    52        07    A3  9E 00060          MOVAB   7(R3), PARENT_TSPTR                               ; 2013
                    03              6C  91 00064          CMPB    (AP), #3                                          ; 2014
                                    10  1B 00067          BLEQU   3$
                    63              04  A0 00069          ADDW2   #4, (TSPTR)                                       ; 2017
                    62              02  B0 0006C          MOVW    #2, (PARENT_TSPTR)                                ; 2018
                 02 A2              01  90 0006F          MOVB    #1, 2(PARENT_TSPTR)                               ; 2019
                 03 A2              56  90 00073          MOVB    DTYPE, 3(PARENT_TSPTR)                            ; 2020
                                    1D  11 00077          BRB     4$                                                ; 2014
                    63              07  A0 00079 3$:      ADDW2   #7, (TSPTR)                                       ; 2025
                    62              05  B0 0007C          MOVW    #5, (PARENT_TSPTR)                                ; 2026
                 02 A2              03  90 0007F          MOVB    #3, 2(PARENT_TSPTR)                               ; 2027
              7E                    57 00000000G  00  C3 00083  SUBL3   DST$BEGIN_ADDR, DSTADDR, -(SP)             ; 2028
           00000000G     00              01  FB 0008B          CALLS   #1, DBG$RST_DST_PTR
                 03 A2              50  D0 00092          MOVL    R0, 3(PARENT_TSPTR)
                    65              63  80 00096 4$:      ADDB2   (TSPTR), (DSTPTR)                                 ; 2031
                    50              54  D0 00099          MOVL    RSTPTR, R0                                        ; 2036
                                        04 0009C          RET                                                       ; 2038

; Routine Size:  157 bytes,    Routine Base:  DBG$CODE + 0AAC
```

```
;  1930          2039  1 GLOBAL ROUTINE DBG$TYPEID_FOR_TPTR(TYPEID) =
;  1931          2040  1 !
;  1932          2041  1 ! FUNCTION
;  1933          2042  1 !       Given a TYPEID describing an arbitrary data type, this routine
;  1934          2043  1 !       constructs a TYPEID for a Typed Pointer data type, where the
;  1935          2044  1 !       typed pointer points to the given data type.
;  1936          2045  1 !
;  1937          2046  1 !       This is needed for the & operator in C.  For example, if X is data
;  1938          2047  1 !       of type T, then &X returns the address of X, and the type of &X is
;  1939          2048  1 !       "pointer to data of type T".  This routine is then used to transform
;  1940          2049  1 !       the TYPEID for X into the TYPEID for &X.
;  1941          2050  1 !
;  1942          2051  1 ! INPUTS
;  1943          2052  1 !       TYPEID  - A TYPEID for the object to which the & operator is being
;  1944          2053  1 !                 applied.
;  1945          2054  1 !
;  1946          2055  1 ! OUTPUTS
;  1947          2056  1 !       Return Value: A TYPEID is returned, as described above.
;  1948          2057  1 !
;  1949          2058  1
;  1950          2059  2     BEGIN
;  1951          2060  2
;  1952          2061  2     MAP
;  1953          2062  2         TYPEID: REF RST$ENTRY;          ! The input TYPEID for the Type Pointer
;  1954          2063  2                                         !      data type
;  1955          2064  2
;  1956          2065  2     LOCAL
;  1957          2066  2         ARG_DSTPTR: REF DST$RECORD,     ! Pointer to DST record for argument
;  1958          2067  2         ARG_TSPTR: REF DST$TYPE_SPEC,   ! Pointer to Type Spec for argument
;  1959          2068  2         DSTPTR: REF DST$RECORD,         ! Pointer to dummy DST record
;  1960          2069  2         RSTPTR: REF RST$ENTRY,          ! Pointer to new RST record
;  1961          2070  2         TSPTR: REF DST$TYPE_SPEC;       ! Pointer to type spec within DST record
;  1962          2071  2
;  1963          2072  2
;  1964          2073  2
;  1965          2074  2     ! Obtain the DST pointer and Type Spec pointer for the argument.
;  1966          2075  2     !
;  1967          2076  2     ARG_DSTPTR = .TYPEID[RST$L_DSTPTR];
;  1968          2077  2     ARG_TSPTR =  ARG_DSTPTR[DST$A_TYPSPEC_TS_ADDR] +
;  1969          2078  2                 .ARG_DSTPTR[DST$B_TYPSPEC_NAME];
;  1970          2079  2
;  1971          2080  2
;  1972          2081  2     ! Allocate a memory block large enough to hold both the newly created
;  1973          2082  2     ! RST record and the dummy DST record that it will point to.
;  1974          2083  2     !
;  1975          2084  2     RSTPTR = DBG$GET_MEMORY (RST$K_TYPENTSIZ + 4);
;  1976          2085  2     DSTPTR = .RSTPTR + 4*RST$K_TYPENTSIZ;
;  1977          2086  2
;  1978          2087  2
;  1979          2088  2     ! Fill in some of the fields of the RST record.
;  1980          2089  2     !
;  1981          2090  2     RSTPTR[RST$L_DSTPTR] = .DSTPTR;
;  1982          2091  2     RSTPTR[RST$L_UPSCOPEPTR] = .RST$START_ADDR;
;  1983          2092  2     RSTPTR[RST$B_KIND] = RST$K_TYPE;
;  1984          2093  2     RSTPTR[RST$B_FCODE] = RST$K_TYPE_TPTR;
;  1985          2094  2     RSTPTR[RST$L_BITSIZE] = 32;
;  1986          2095  2
```

```
;  1987          2096  2
;  1988          2097  2      ! Put the RST record on the temporary RST list. This will ensure that
;  1989          2098  2      ! the space gets released when there are no more references to it.
;  1990          2099  2      !
;  1991          2100  2      RSTPTR[RST$L_HASH_FLINK] = .RST$TEMP_LIST;
;  1992          2101  2      RST$TEMP_LIST = .RSTPTR;
;  1993          2102  2
;  1994          2103  2
;  1995          2104  2      ! Fill in the dummy DST record.
;  1996          2105  2      !
;  1997          2106  2      DSTPTR[DST$B_LENGTH] = 12;
;  1998          2107  2      DSTPTR[DST$B_TYPE] = DST$K_TYPSPEC;
;  1999          2108  2      DSTPTR[DST$B_TYPSPEC_NAME] = 0;
;  2000          2109  2
;  2001          2110  2
;  2002          2111  2      ! Fill in the Type Spec record within the dummy DST record.
;  2003          2112  2      !
;  2004          2113  2      TSPTR = .DSTPTR + 3;
;  2005          2114  2      TSPTR[DST$W_TS_LENGTH] = 8;
;  2006          2115  2      TSPTR[DST$B_TS_KIND] = DST$K_TS_TPTR;
;  2007          2116  2      TSPTR = TSPTR[DST$A_TS_TPTR_TSPEC_ADDR];
;  2008          2117  2
;  2009          2118  2
;  2010          2119  2      ! Fill in the Type Spec for the pointed-to object.
;  2011          2120  2      ! Atomic types are filled in in-line, other types use the
;  2012          2121  2      ! "indirect" Type Spec to point back to to argument DST.
;  2013          2122  2      !
;  2014          2123  2      IF .ARG_TSPTR[DST$B_TS_KIND] EQL DST$K_TS_ATOM
;  2015          2124  2      THEN
;  2016          2125  3          BEGIN
;  2017          2126  3          TSPTR[DST$W_TS_LENGTH] = 2;
;  2018          2127  3          TSPTR[DST$B_TS_KIND] = DST$K_TS_ATOM;
;  2019          2128  3          TSPTR[DST$B_TS_ATOM_TYP] = .ARG_TSPTR[DST$B_TS_ATOM_TYP];
;  2020          2129  3          END
;  2021          2130  3
;  2022          2131  2      ELSE
;  2023          2132  3          BEGIN
;  2024          2133  3          TSPTR[DST$W_TS_LENGTH] = 5;
;  2025          2134  3          TSPTR[DST$B_TS_KIND] = DST$K_TS_IND;
;  2026          2135  3          TSPTR[DST$L_TS_IND_PTR] = DBG$RST_DST_PTR( .ARG_DSTPTR - .DST$BEGIN_ADDR );
;  2027          2136  2          END;
;  2028          2137  2
;  2029          2138  2
;  2030          2139  2      ! Return the pointer to the RST record.
;  2031          2140  2      !
;  2032          2141  2      RETURN .RSTPTR;
;  2033          2142  2
;  2034          2143  1      END;


                                  007C 00000          .ENTRY  DBG$TYPEID_FOR_TPTR, Save R2,R3,R4,R5,R6    ; 2039
                56 00000000G   00  9E 00002          MOVAB   RST$TEMP_LIST, R6                           ;
                50         04  AC  D0 00009          MOVL    TYPEID, R0                                  ; 2076
                54         0C  A0  D0 0000D          MOVL    12(R0), ARG_DSTPTR                          ;
```

```
                        50        02    A4  9A 00011          MOVZBL  2(ARG_DSTPTR), R0                        ; 2078
                        55        03  A044  9E 00015          MOVAB   3(R0)[ARG_DSTPTR], ARG_TSPTR             ; 2077
                                      0F    DD 0001A          PUSHL   #15                                      ; 2084
                00000000G       00          01    FB 0001C    CALLS   #1, DBG$GET_MEMORY                       ;
                        53                  50    D0 00023    MOVL    R0, RSTPTR                               ;
                        50        2C        A3    9E 00026    MOVAB   44(R3), DSTPTR                           ; 2085
                    0C  A3                  50    D0 0002A    MOVL    DSTPTR, 12(RSTPTR)                       ; 2090
                    10  A3  00000000G       00    D0 0002E    MOVL    RST$START_ADDR, 16(RSTPTR)               ; 2091
                    14  A3                  07    90 00036    MOVB    #7, 20(RSTPTR)                           ; 2092
                    18  A3                  06    90 0003A    MOVB    #6, 24(RSTPTR)                           ; 2093
                    20  A3                  20    D0 0003E    MOVL    #32, 32(RSTPTR)                          ; 2094
                        63                  66    D0 00042    MOVL    RST$TEMP_LIST, (RSTPTR)                  ; 2100
                        66                  53    D0 00045    MOVL    RSTPTR, RST$TEMP_LIST                    ; 2101
                        60        AF0C      8F    B0 00048    MOVW    #44812, (DSTPTR)                         ; 2106
                                    02      A0    94 0004D    CLRB    2(DSTPTR)                                ; 2108
                        52        03        A0    9E 00050    MOVAB   3(R0), TSPTR                             ; 2113
                        82                  08    B0 00054    MOVW    #8, (TSPTR)+                             ; 2114
                        82                  04    90 00057    MOVB    #4, (TSPTR)+                             ; 2115
                        01        02        A5    91 0005A    CMPB    2(ARG_TSPTR), #1                         ; 2123
                                            0E    12 0005E    BNEQ    1$                                       ;
                        62                  02    B0 00060    MOVW    #2, (TSPTR)                              ; 2126
                    02  A2                  01    90 00063    MOVB    #1, 2(TSPTR)                             ; 2127
                    03  A2        03        A5    90 00067    MOVB    3(ARG_TSPTR), 3(TSPTR)                   ; 2128
                                            1A    11 0006C    BRB     2$                                       ; 2123
                        62                  05    B0 0006E 1$:    MOVW    #5, (TSPTR)                          ; 2133
                    02  A2                  03    90 00071    MOVB    #3, 2(TSPTR)                             ; 2134
                7E              54  00000000G       00    C3 00075    SUBL3   DST$BEGIN_ADDR, ARG_DSTPTR, -(SP)    ; 2135
                00000000G       00          01    FB 0007D    CALLS   #1, DBG$RST_DST_PTR                      ;
                    03  A2                  50    D0 00084    MOVL    R0, 3(TSPTR)                             ;
                        50                  53    D0 00088 2$:    MOVL    RSTPTR, R0                           ; 2141
                                            04    0008B       RET                                              ; 2143

; Routine Size:  140 bytes,    Routine Base:  DBG$CODE + 0B49
```

```
;2036     2144  1 ROUTINE TYPEID_FOR_COB_HACK(CH_DSTPTR) =
;2037     2145  1 !
;2038     2146  1 ! FUNCTION
;2039     2147  1 !       This routine builds an appropriate Data Type RST Entry for the data
;2040     2148  1 !       type of an object described by a Cobol Hack DST Record.  It accepts
;2041     2149  1 !       as input a pointer to the DST record and it builds and returns the
;2042     2150  1 !       address of a Data Type RST Entry for the corresponding type.  The
;2043     2151  1 !       memory block which contains the Type RST Entry also contains a dummy
;2044     2152  1 !       Type Spec DST Record for the data type in question.
;2045     2153  1 !
;2046     2154  1 !       Cobol Hack records are obsolete, but are still found in programs com-
;2047     2155  1 !       piled with the Version 1 COBOL compiler.  This routine therefore con-
;2048     2156  1 !       verts such DST records into more "regular" DST structures so that on-
;2049     2157  1 !       ly this routi .e needs to understand the obsolete Cobol Hack Record.
;2050     2158  1 !
;2051     2159  1 ! INPUTS
;2052     2160  1 !       CH_DSTPTR - A pointer to the Cobol Hack DST Record whose data type is
;2053     2161  1 !                   to be extracted.
;2054     2162  1 !
;2055     2163  1 ! OUTPUTS
;2056     2164  1 !       This routine returns as its value a pointer to a Data Type RST Entry
;2057     2165  1 !                   for the data type specified in the Cobol Hack DST Record.
;2058     2166  1 !
;2059     2167  1
;2060     2168  2     BEGIN
;2061     2169  2
;2062     2170  2     MAP
;2063     2171  2         CH_DSTPTR: REF DST$RECORD;      ! Pointer to Cobol Hack DST Record
;2064     2172  2
;2065     2173  2     LOCAL
;2066     2174  2         BITSIZ,                         ! The bitsize of an atomic data type
;2067     2175  2         CH_TRLR_PTR: REF DST$CH_TRLR,   ! Pointer to Cobol Hack record trailer
;2068     2176  2         DESCADDR: REF BLOCK[,BYTE],     ! Pointer to VAX standard descriptor
;2069     2177  2         DSTPTR: REF DST$RECORD,         ! Pointer to dummy Type Spec DST Record
;2070     2178  2                                         !     built by this routine
;2071     2179  2         RSTPTR: REF RST$ENTRY,          ! Pointer to Type RST Entry built by
;2072     2180  2                                         !     by this routine for data item
;2073     2181  2         TSPTR: REF DST$TYPE_SPEC,       ! Pointer to Type Spec in Type Spec DST
;2074     2182  2                                         !     record built by this routine
;2075     2183  2         VSPTR: REF DST$VAL_SPEC;        ! Pointer to Value Spec in Type Spec
;2076     2184  2                                         !     DST record built by this routine
;2077     2185  2
;2078     2186  2
;2079     2187  2
;2080     2188  2     ! If the descriptor address in the DST record is zero, this is an atomic
;2081     2189  2     ! data type defined by the DST$CH_TYPE field.  Build a Type RST Entry for
;2082     2190  2     ! it and return that entry's address as a TYPEID.
;2083     2191  2     !
;2084     2192  2     IF .CH_DSTPTR[DST$L_VALUE] EQL 0
;2085     2193  2     THEN
;2086     2194  3         BEGIN
;2087     2195  3         CH_TRLR_PTR = CH_DSTPTR[DST$A_COBHACK_TRLR] + .CH_DSTPTR[DST$B_NAME];
;2088     2196  3         BITSIZ = GET_BITSIZE_FROM_DTYPE(.CH_TRLR_PTR[DST$B_CH_TYPE]);
;2089     2197  3         RSTPTR = DBG$TYPEID_FOR_ATOMIC(.CH_TRLR_PTR[DST$B_CH_TYPE], .BITSIZ, FALSE);
;2090     2198  3         RETURN .RSTPTR;
;2091     2199  2         END;
;2092     2200  2
```

```
;  2093          2201  2
;  2094          2202  2      ! We have a descriptor-specified data type.  Compute the address of the
;  2095          2203  2      ! descriptor.
;  2096          2204  2      !
;  2097          2205  2      IF .CH_DSTPTR[DST$B_VFLAGS] EQL DST$K_VFLAGS_DSC
;  2098          2206  2      THEN
;  2099          2207  2          DESCADDR = CH_DSTPTR[DST$A_DSC_BASE] + .CH_DSTPTR[DST$L_DSC_OFFS]
;  2100          2208  2
;  2101          2209  2      ELSE
;  2102          2210  2          SIGNAL(DBG$_INVDSTREC);
;  2103          2211  2
;  2104          2212  2
;  2105          2213  2      ! If the descriptor is an array descriptor, we build an Array Type RST Entry
;  2106          2214  2      ! and the corresponding dummy Type Spec DST Record.  We also return the
;  2107          2215  2      ! address of the Type RST Entry as a TYPEID.
;  2108          2216  2      !
;  2109          2217  2      IF (.DESCADDR[DSC$B_CLASS] EQL DSC$K_CLASS_A)   OR
;  2110          2218  2         (.DESCADDR[DSC$B_CLASS] EQL DSC$K_CLASS_NCA) OR
;  2111          2219  2         (.DESCADDR[DSC$B_CLASS] EQL DSC$K_CLASS_UBA) OR
;  2112          2220  3         (.DESCADDR[DSC$B_CLASS] EQL DSC$K_CLASS_VSA)
;  2113          2221  2      THEN
;  2114          2222  3          BEGIN
;  2115          2223  3
;  2116          2224  3
;  2117          2225  3          ! Get a memory block to hold both the Type RST Entry and the Type Spec
;  2118          2226  3          ! DST Record for the array.
;  2119          2227  3          !
;  2120          2228  3          RSTPTR = DBG$GET_MEMORY(RST$K_TYPENTSIZ + 4);
;  2121          2229  3          DSTPTR = .RSTPTR + 4*RST$K_TYPENTSIZ;
;  2122          2230  3
;  2123          2231  3
;  2124          2232  3          ! Build the Data Type RST Entry.
;  2125          2233  3          !
;  2126          2234  3          RSTPTR[RST$L_DSTPTR] = .DSTPTR;
;  2127          2235  3          RSTPTR[RST$L_UPSCOPEPTR] = .RST$START_ADDR;
;  2128          2236  3          RSTPTR[RST$B_KIND] = RST$K_TYPE;
;  2129          2237  3          RSTPTR[RST$B_FCODE] = RST$K_TYPE_ARRAY;
;  2130          2238  3
;  2131          2239  3
;  2132          2240  3          ! Put this Data Type RST Entry on the Temporary RST Entry List.  This
;  2133          2241  3          ! will cause the entry to be released at the end of a DEBUG command
;  2134          2242  3          ! when there are no references to it.
;  2135          2243  3          !
;  2136          2244  3          RSTPTR[RST$L_HASH_FLINK] = .RST$TEMP_LIST;
;  2137          2245  3          RST$TEMP_LIST = .RSTPTR;
;  2138          2246  3
;  2139          2247  3
;  2140          2248  3          ! Then build the Type Specification DST Record.  This contains a Type
;  2141          2249  3          ! Spec containing a Value Spec which contains the descriptor address.
;  2142          2250  3          !
;  2143          2251  3          DSTPTR[DST$B_LENGTH] = 12;
;  2144          2252  3          DSTPTR[DST$B_TYPE] = DST$K_TYPSPEC;
;  2145          2253  3          TSPTR = DSTPTR[DST$A_TYPSPEC_TS_ADDR];
;  2146          2254  3          TSPTR[DST$W_TS_LENGTH] = 8;
;  2147          2255  3          TSPTR[DST$B_TS_KIND] = DST$K_TS_ARRAY;
;  2148          2256  3          TSPTR[DST$B_TS_ARRAY_DIM] = .DESCADDR[DSC$B_DIMCT];
;  2149          2257  3          VSPTR = TSPTR[DST$A_TS_ARRAY_FLAGS_ADDR] + 1;
```

```
; 2150          2258  3          VSPTR[DST$B_VS_VFLAGS] = DST$K_VALKIND_DESC;
; 2151          2259  3          VSPTR[DST$L_VS_VALUE] = .DESCADDR;
; 2152          2260  3
; 2153          2261  3
; 2154          2262  3          ! Return the address of the Type RST Entry we built (the TYPEID).
; 2155          2263  3          !
; 2156          2264  3          RETURN .RSTPTR;
; 2157          2265  2          END;
; 2158          2266  2
; 2159          2267  2
; 2160          2268  2      ! It is not an array descriptor.  We therefore generate a Descriptor Type
; 2161          2269  2      ! RST Entry and the corresponding dummy Type Spec DST Record.
; 2162          2270  2      !
; 2163          2271  2      RSTPTR = DBG$GET_MEMORY(RST$K_TYPENTSIZ + 3);
; 2164          2272  2      DSTPTR = .RSTPTR + 4*RST$K_TYPENTSIZ;
; 2165          2273  2
; 2166          2274  2
; 2167          2275  2      ! Build the Data Type RST Entry.
; 2168          2276  2      !
; 2169          2277  2      RSTPTR[RST$L_DSTPTR] = .DSTPTR;
; 2170          2278  2      RSTPTR[RST$L_UPSCOPEPTR] = .RST$START_ADDR;
; 2171          2279  2      RSTPTR[RST$B_KIND] = RST$K_TYPE;
; 2172          2280  2      RSTPTR[RST$B_FCODE] = RST$R_TYPE_DESCR;
; 2173          2281  2
; 2174          2282  2
; 2175          2283  2      ! Put the Data Type RST Entry on the Temporary RST Entry List.  This will
; 2176          2284  2      ! cause the entry to be released when no longer referenced.
; 2177          2285  2      !
; 2178          2286  2      RSTPTR[RST$L_HASH_FLINK] = .RST$TEMP_LIST;
; 2179          2287  2      RST$TEMP_LIST = .RSTPTR;
; 2180          2288  2
; 2181          2289  2
; 2182          2290  2      ! Then build the Type Specification DST Record.  This will contain a Type
; 2183          2291  2      ! Spec containing a Value Spec containing the descriptor address.
; 2184          2292  2      !
; 2185          2293  2      DSTPTR[DST$B_LENGTH] = 10;
; 2186          2294  2      DSTPTR[DST$B_TYPE] = DST$K_TYPSPEC;
; 2187          2295  2      TSPTR = DSTPTR[DST$A_TYPSPEC_TS_ADDR];
; 2188          2296  2      TSPTR[DST$W_TS_LENGTH] = 6;
; 2189          2297  2      TSPTR[DST$B_TS_KIND] = DST$K_TS_DSC;
; 2190          2298  2      VSPTR = TSPTR[DST$A_TS_DSC_VSPEC_ADDR];
; 2191          2299  2      VSPTR[DST$B_VS_VFLAGS] = DST$K_VALKIND_DESC;
; 2192          2300  2      VSPTR[DST$L_VS_VALUE] = .DESCADDR;
; 2193          2301  2
; 2194          2302  2
; 2195          2303  2      ! Return the address of the Type RST Entry we build, i.e. the TYPEID.
; 2196          2304  2      !
; 2197          2305  2      RETURN .RSTPTR;
; 2198          2306  2
; 2199          2307  1      END;


                                        03FC 00000 TYPEID_FOR_COB_HACK:
                                                   .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9          ; 2144
```

```
                59 00000000G   00   9E 00002          MOVAB   RST$START_ADDR, R9                                ;
                58 00000000G   00   9E 00009          MOVAB   DBG$GET_MEMORY, R8                               ;
                57 00000000G   00   9E 00010          MOVAB   RST$TEMP_LIST, R7                                ;
                52         04   AC   D0 00017          MOVL    CH_DSTPTR, R2                                    ; 2192
                           03   A2   D5 0001B          TSTL    3(R2)                                            ;
                                23   12 0001E          BNEQ    1$                                               ;
                50         07   A2   9A 00020          MOVZBL  7(R2), R0                                        ; 2195
                53         08 A042   9E 00024          MOVAB   8(R0)[R2], CH_TRLR_PTR                           ;
                7E               63   9A 00029          MOVZBL  (CH_TRLR_PTR), -(SP)                             ; 2196
          0000V CF               01   FB 0002C          CALLS   #1, GET_BITSIZE_FROM_DTYPE                       ;
                                 7E   D4 00031          CLRL    -(SP)                                            ; 2197
                                 50   DD 00033          PUSHL   BITSIZ                                           ;
                7E               63   9A 00035          MOVZBL  (CH_TRLR_PTR), -(SP)                             ;
            FE12 CF              03   FB 00038          CALLS   #3, DBG$TYPEID_FOR_ATOMIC                        ;
                53               50   D0 0003D          MOVL    R0, RSTPTR                                       ;
                              00AF   31 00040          BRW     7$                                               ; 2198
              FA 8F          02   A2   91 00043 1$:     CMPB    2(R2), #250                                      ; 2205
                                09   12 00048          BNEQ    2$                                               ;
                52           03   A2   C0 0004A          ADDL2   3(R2), R2                                        ; 2207
                52                07   C0 0004E          ADDL2   #7, DESCADDR                                     ;
                                  0D   11 00051          BRB     3$                                               ;
                        0002832A  8F   DD 00053 2$:     PUSHL   #164650                                          ; 2210
          00000000G 00            01   FB 00059          CALLS   #1, LIB$SIGNAL                                   ;
                04           03   A2   91 00060 3$:     CMPB    3(DESCADDR), #4                                  ; 2217
                                  12   13 00064          BEQL    4$                                               ;
                0A           03   A2   91 00066          CMPB    3(DESCADDR), #10                                 ; 2218
                                  0C   13 0006A          BEQL    4$                                               ;
                0E           03   A2   91 0006C          CMPB    3(DESCADDR), #14                                 ; 2219
                                  06   13 00070          BEQL    4$                                               ;
                0C           03   A2   91 00072          CMPB    3(DESCADDR), #12                                 ; 2220
                                  3D   12 00076          BNEQ    5$                                               ;
                                  0F   DD 00078 4$:     PUSHL   #15                                              ; 2228
                68                01   FB 0007A          CALLS   #1, DBG$GET_MEMORY                               ;
                53                50   D0 0007D          MOVL    R0, RSTPTR                                       ;
                56           2C   A3   9E 00080          MOVAB   44(R3), DSTPTR                                   ; 2229
          0C    A3                56   D0 00084          MOVL    DSTPTR, 12(RSTPTR)                               ; 2234
          10    A3                69   D0 00088          MOVL    RST$START_ADDR, 16(RSTPTR)                       ; 2235
          14    A3                07   90 0008C          MOVB    #7, 20(RSTPTR)                                   ; 2236
          18    A3                01   90 00090          MOVB    #1, 24(RSTPTR)                                   ; 2237
                63                67   D0 00094          MOVL    RST$TEMP_LIST, (RSTPTR)                          ; 2244
                67                53   D0 00097          MOVL    RSTPTR, RST$TEMP_LIST                            ; 2245
                66         AF0C   8F   B0 0009A          MOVW    #44812, (DSTPTR)                                 ; 2251
                55           03   A6   9E 0009F          MOVAB   3(R6), TSPTR                                     ; 2253
                65                08   B0 000A3          MOVW    #8, (TSPTR)                                      ; 2254
          02    A5                07   90 000A6          MOVB    #7, 2(TSPTR)                                     ; 2255
          03    A5           0B   A2   90 000AA          MOVB    11(DESCADDR), 3(TSPTR)                           ; 2256
                54           05   A5   9E 000AF          MOVAB   5(R5), VSPTR                                     ; 2257
                                  36   11 000B3          BRB     6$                                               ; 2258
                                  0E   DD 000B5 5$:     PUSHL   #14                                              ; 2271
                68                01   FB 000B7          CALLS   #1, DBG$GET_MEMORY                               ;
                53                50   D0 000BA          MOVL    R0, RSTPTR                                       ;
                56           2C   A3   9E 000BD          MOVAB   44(R3), DSTPTR                                   ; 2272
          0C    A3                56   D0 000C1          MOVL    DSTPTR, 12(RSTPTR)                               ; 2277
          10    A3                69   D0 000C5          MOVL    RST$START_ADDR, 16(RSTPTR)                       ; 2278
          14    A3                07   90 000C9          MOVB    #7, 20(RSTPTR)                                   ; 2279
          18    A3                03   90 000CD          MOVB    #3, 24(RSTPTR)                                   ; 2280
                63                67   D0 000D1          MOVL    RST$TEMP_LIST, (RSTPTR)                          ; 2286
```

```
                67              53  D0 000D4          MOVL    RSTPTR, RST$TEMP_LIST                   ; 2287
                66      AF0A    8F  B0 000D7          MOVW    #44810, (DSTPTR)                        ; 2293
                55        03    A6  9E 000DC          MOVAB   3(R6), TSPTR                            ; 2295
                65              06  B0 000E0          MOVW    #6, (TSPTR)                             ; 2296
            02  A5              02  90 000E3          MOVB    #2, 2(TSPTR)                            ; 2297
                54        03    A5  9E 000E7          MOVAB   3(R5), VSPTR                            ; 2298
                64              02  90 000EB 6$:      MOVB    #2, (VSPTR)                             ; 2299
            01  A4              52  D0 000EE          MOVL    DESCADDR, 1(VSPTR)                      ; 2300
                50              53  D0 000F2 7$:      MOVL    RSTPTR, R0                              ; 2305
                                    04 000F5          RET                                             ; 2307

; Routine Size:  246 bytes,    Routine Base:  DBG$CODE + 0BD5
```

```
;  2201          2308  1 ROUTINE TYPEID_FOR_DESCR(DESCADDR, FCODE) =
;  2202          2309  1 !
;  2203          2310  1 ! FUNCTION
;  2204          2311  1 !       This routine returns the Type ID associated with a VAX standard descrip-
;  2205          2312  1 !       tor type.  The routine builds a Data Type RST Entry for the data type
;  2206          2313  1 !       and returns its address as the Type ID.  This RST entry points to a dum-
;  2207          2314  1 !       my Type Specification DST Record, also built by this routine, which con-
;  2208          2315  1 !       tains a descriptor type specification which in turn contains a value
;  2209          2316  1 !       specification which points to the specified descriptor.  The RST entry
;  2210          2317  1 !       and the DST record are both built in the same memory block.  This block
;  2211          2318  1 !       is put on the Temporary RST Entry List so that it disappears as soon as
;  2212          2319  1 !       there no longer are any references to it.
;  2213          2320  1 !
;  2214          2321  1 ! INPUTS
;  2215          2322  1 !       DESCADDR - The address of the descriptor for which a Type ID is to be
;  2216          2323  1 !                  returned.
;  2217          2324  1 !
;  2218          2325  1 !       FCODE    - The format code of the resultant Type ID.
;  2219          2326  1 !
;  2220          2327  1 !       ARRFLAG - (Optional). If this argument is TRUE, DESCADDR is assumed
;  2221          2328  1 !                 to be an ARRAY descriptor,  and a VAX standard descriptor
;  2222          2329  1 !                 is built to describe an element of the array.  The descrip-
;  2223          2330  1 !                 tor is also placed in the same memory block.
;  2224          2331  1 !
;  2225          2332  1 ! OUTPUTS
;  2226          2333  1 !       The corresponding Type ID is returned as the routine value.
;  2227          2334  1 !
;  2228          2335  1
;  2229          2336  2     BEGIN
;  2230          2337  2
;  2231          2338  2     MAP
;  2232          2339  2         DESCADDR: REF BLOCK[,BYTE];
;  2233          2340  2
;  2234          2341  2     BUILTIN
;  2235          2342  2         ACTUALCOUNT,
;  2236          2343  2         ACTUALPARAMETER;
;  2237          2344  2
;  2238          2345  2     LOCAL
;  2239          2346  2         ARRFLAG,                        ! Flag set to TRUE if the input descrip-
;  2240          2347  2                                         !      tor is an array descriptor
;  2241          2348  2         DSTPTR: REF DST$RECORD,         ! Pointer to the type's DST record
;  2242          2349  2         ELEMENT: REF DBG$STG_DESC,      ! Pointer to array element descriptor
;  2243          2350  2                                         !      in constructed DST entry
;  2244          2351  2         RSTPTR: REF RST$FNTRY,          ! Pointer to the type's RST entry
;  2245          2352  2         SIZE,
;  2246          2353  2         TSPTR: REF DST$TYPE_SPEC,       ! Pointer to the DST Type Specification
;  2247          2354  2         VSPTR: REF DST$VAL_SPEC;        ! Pointer to the DST Value Specification
;  2248          2355  2
;  2249          2356  2
;  2250          2357  2     ! *** How am I supposed to get this definition properly?
;  2251          2358  2     !
;  2252          2359  2     MACRO
;  2253          2360  2         DSC$B_DIMCNT = 8, 24, 8, 0%;
;  2254          2361  2
;  2255          2362  2     ! If the input descriptor is an array descriptor, as declared by a third
;  2256          2363  2     ! argument (the ARRFLAG argument), then we set the ARRFLAG flag to TRUE.
;  2257          2364  2     ! Otherwise, it is not an array descriptor and ARRFLAG is set to FALSE.
```

```
;2258      2365  2      !
;2259      2366  2      ARRFLAG = (IF ACTUALCOUNT() GTR 2 THEN ACTUALPARAMETER(3) ELSE FALSE);
;2260      2367  2
;2261      2368  2      ! Figure out how big the descriptor is.
;2262      2369  2      ! Assume 12 to start with.
;2263      2370  2      ! Then check for array descriptors, which are bigger.
;2264      2371  2      ! In some cases, there may be some slop (SIZE may be bigger
;2265      2372  2      ! than the descriptor really is), but that is OK.
;2266      2373  2      !
;2267      2374  2      SIZE = 12;
;2268      2375  2      IF .ARRFLAG
;2269      2376  2      THEN
;2270      2377  2          SIZE = 10
;2271      2378  2      ELSE
;2272      2379  2          IF (.DESCADDR[DSC$B_CLASS] EQL DSC$K_CLASS_A) OR
;2273      2380  2             (.DESCADDR[DSC$B_CLASS] EQL DSC$K_CLASS_NCA) OR
;2274      2381  2             (.DESCADDR[DSC$B_CLASS] EQL DSC$K_CLASS_UBA) OR
;2275      2382  3             (.DESCADDR[DSC$B_CLASS] EQL DSC$K_CLASS_VSA)
;2276      2383  2          THEN
;2277      2384  2              SIZE = 20 + (12 * .DESCADDR[DSC$B_DIMCNT]);
;2278      2385  2
;2279      2386  2      ! Get a memory block to hold both the RST entry and the DST record,
;2280      2387  2      ! and the descriptor. Copy the descriptor into the end of this
;2281      2388  2      ! memory block, and reset DESCADDR to point to this copy. (The
;2282      2389  2      ! reason for this is so that the descriptor resides in permanent
;2283      2390  2      ! memory, and in fact in the same permanent memory block as the
;2284      2391  2      ! typeid. This makes sure that the descriptor does not outlive
;2285      2392  2      ! the typeid or vice-versa.
;2286      2393  2      !
;2287      2394  2      RSTPTR = DBG$GET_MEMORY(RST$K_TYPENTSIZ + 3 + (.SIZE+3)/4);
;2288      2395  2      DSTPTR = .RSTPTR + 4*RST$K_TYPENTSIZ;
;2289      2396  2      ELEMENT = .DSTPTR+12;
;2290      2397  2      CH$MOVE(.SIZE,.DESCADDR,.ELEMENT);
;2291      2398  2      DESCADDR = .ELEMENT;
;2292      2399  2
;2293      2400  2      ! For array descriptors we change the class field so it describes the
;2294      2401  2      ! element and not the entire array.
;2295      2402  2      !
;2296      2403  2      IF .ARRFLAG
;2297      2404  2      THEN
;2298      2405  3          BEGIN
;2299      2406  4          ELEMENT[DSC$B_CLASS] = (IF .ELEMENT[DSC$B_DTYPE] EQL DSC$K_DTYPE_T
;2300      2407  3                                  THEN DSC$K_CLASS_S ELSE DSC$K_CLASS_SD);
;2301      2408  3          ELEMENT[DSC$A_POINTER] = 0;
;2302      2409  2          END;
;2303      2410  2
;2304      2411  2      ! Build the Data Type RST Entry for the descriptor type.
;2305      2412  2      !
;2306      2413  2      RSTPTR[RST$L_DSTPTR] = .DSTPTR;
;2307      2414  2      RSTPTR[RST$L_UPSCOPEPTR] = .RST$START_ADDR;
;2308      2415  2      RSTPTR[RST$B_KIND] = RST$K_TYPE;
;2309      2416  2      RSTPTR[RST$B_FCODE] = .FCODE;
;2310      2417  2
;2311      2418  2
;2312      2419  2      ! Put this Data Type RST Entry on the Temporary RST Entry List.  This will
;2313      2420  2      ! cause the entry to be released at the end of a DEBUG command when there
;2314      2421  2      ! are no references to it (when it is not locked).
```

```
;  2315          2422  2     !
;  2316          2423  2     RSTPTR[RST$L_HASH_FLINK] = .RST$TEMP_LIST;
;  2317          2424  2     RST$TEMP_LIST = .RSTPTR;
;  2318          2425  2
;  2319          2426  2
;  2320          2427  2     ! Then build the corresponding Type Specification DST Record.  This contains
;  2321          2428  2     ! a type spec which contains a value spec containing the descriptor address.
;  2322          2429  2     !
;  2323          2430  2     DSTPTR[DST$B_LENGTH] = 10;
;  2324          2431  2     DSTPTR[DST$B_TYPE] = DST$K_TYPSPEC;
;  2325          2432  2     DSTPTR[DST$B_TYPSPEC_NAME] = 0;
;  2326          2433  2     TSPTR = DSTPTR[DST$A_TYPSPEC_TS_ADDR];
;  2327          2434  2     TSPTR[DST$W_TS_LENGTH] = 6;
;  2328          2435  2     TSPTR[DST$B_TS_KIND] = DST$K_TS_DSC;
;  2329          2436  2     VSPTR = TSPTR[DST$A_TS_DSC_VSPEC_ADDR];
;  2330          2437  2     VSPTR[DST$B_VS_VFLAGS] = 0;
;  2331          2438  2     VSPTR[DST$V_VS_VALKIND] = DST$K_VALKIND_DESC;
;  2332          2439  2     VSPTR[DST$L_VS_VALUE] = .DESCADDR;
;  2333          2440  2
;  2334          2441  2
;  2335          2442  2     ! Return the address of the Data Type RST Entry as the Type ID.
;  2336          2443  2     !
;  2337          2444  2     RETURN .RSTPTR;
;  2338          2445  1     END;


                                07FC 00000 TYPEID_FOR_DESCR:
                                           .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10                ; 2308
                    5A 00000000G   00  9E 00002    MOVAB   RST$TEMP_LIST, R10                      ;
                    02             6C  91 00009    CMPB    (AP), #2                                ; 2366
                                   06  1B 0000C    BLEQU   1$                                      ;
                    59        0C   AC  D0 0000E    MOVL    12(AP), ARRFLAG                         ;
                                   02  11 00012    BRB     2$                                      ;
                                   59  D4 00014 1$: CLRL    ARRFLAG                                 ;
                    52             0C  D0 00016 2$: MOVL    #12, SIZE                               ; 2374
                    05             59  E9 00019    BLBC    ARRFLAG, 3$                             ; 2375
                    52             0A  D0 0001C    MOVL    #10, SIZE                               ; 2377
                                   27  11 0001F    BRB     5$                                      ;
                    50        04   AC  D0 00021 3$: MOVL    DESCADDR, R0                            ; 2379
                    04        03   A0  91 00025    CMPB    3(R0), #4                               ;
                                   12  13 00029    BEQL    4$                                      ;
                    0A        03   A0  91 0002B    CMPB    3(R0), #10                              ; 2380
                                   0C  13 0002F    BEQL    4$                                      ;
                    0E        03   A0  91 00031    CMPB    3(R0), #14                              ; 2381
                                   06  13 00035    BEQL    4$                                      ;
                    0C        03   A0  91 00037    CMPB    3(R0), #12                              ; 2382
                                   0B  12 0003B    BNEQ    5$                                      ;
                    50        0B   A0  9A 0003D 4$: MOVZBL  11(R0), R0                              ; 2384
                    50             0C  C4 00041    MULL2   #12, R0                                 ;
                    52        14   A0  9E 00044    MOVAB   20(R0), SIZE                            ;
                    50        03   A2  9E 00048 5$: MOVAB   3(R2), R0                               ; 2394
                    50             04  C6 0004C    DIVL2   #4, R0                                  ;
                              0E   A0  9F 0004F    PUSHAB  14(R0)                                  ;
          00000000G 00             01  FB 00052    CALLS   #1, DBG$GET_MEMORY                      ;
```

```
                        58          50  D0 00059          MOVL    R0, RSTPTR                              ;
                        57      2C  A8  9E 0005C          MOVAB   44(R8), DSTPTR                          ; 2395
                        56      0C  A7  9E 00060          MOVAB   12(R7), ELEMENT                         ; 2396
                66      04  BC      52  28 00064          MOVC3   SIZE, @DESCADDR, (ELEMENT)              ; 2397
                        04  AC      56  D0 00069          MOVL    ELEMENT, DESCADDR                       ; 2398
                            15      59  E9 0006D          BLBC    ARRFLAG, 8$                             ; 2403
                            0E  02  A6  91 00070          CMPB    2(ELEMENT), #14                         ; 2406
                                    05  12 00074          BNEQ    6$                                      ;
                            50      01  D0 00076          MOVL    #1, R0                                  ;
                                    03  11 00079          BRB     7$                                      ;
                            50      09  D0 0007B  6$:      MOVL    #9, R0                                  ;
                        03  A6      50  90 0007E  7$:      MOVB    R0, 3(ELEMENT)                          ;
                            04  A6  D4 00082              CLRL    4(ELEMENT)                              ; 2408
                        0C  A8      57  D0 00085  8$:      MOVL    DSTPTR, 12(RSTPTR)                      ; 2413
                        10  A8 00000000G    00  D0 00089  MOVL    RST$START_ADDR, 16(RSTPTR)              ; 2414
                        14  A8      07  90 00091          MOVB    #7, 20(RSTPTR)                          ; 2415
                        18  A8  08  AC  90 00095          MOVB    FCODE, 24(RSTPTR)                       ; 2416
                            68      6A  D0 0009A          MOVL    RST$TEMP_LIST, (RSTPTR)                 ; 2423
                            6A      58  D0 0009D          MOVL    RSTPTR, RST$TEMP_LIST                   ; 2424
                            67  AF0A    8F  B0 000A0      MOVW    #44810, (DSTPTR)                        ; 2430
                                02  A7  94 000A5          CLRB    2(DSTPTR)                               ; 2432
                            50  03  A7  9E 000A8          MOVAB   3(R7), TSPTR                            ; 2433
                            80      06  B0 000AC          MOVW    #6, (TSPTR)+                            ; 2434
                            80      02  90 000AF          MOVB    #2, (TSPTR)+                            ; 2435
                                    60  94 000B2          CLRB    (VSPTR)                                 ; 2437
            60      02              00  02  F0 000B4      INSV    #2, #0, #2, (VSPTR)                     ; 2438
                        01  A0  04  AC  D0 000B9          MOVL    DESCADDR, 1(VSPTR)                      ; 2439
                            50      58  D0 000BE          MOVL    RSTPTR, R0                              ; 2444
                                        04 000C1          RET                                             ; 2445

; Routine Size:  194 bytes,    Routine Base:  DBG$CODE + 0CCB
```

```
;  2340          2446  1 ROUTINE TYPEID_FROM_DST_TYPESPEC(DSTADDR, FCODE, BITSIZE) =
;  2341          2447  1 !
;  2342          2448  1 ! FUNCTION
;  2343          2449  1 !       This routine builds a Data Type RST Entry for a data type specified by a
;  2344          2450  1 !       DST Type Specification embedded in some other symbol's DST record.  For
;  2345          2451  1 !       example, the cell type of an array or the referenced type of a typed
;  2346          2452  1 !       pointer is specified by a Type Specification in the DST record for the
;  2347          2453  1 !       array or the typed pointer, but does not have a separate DST record of
;  2348          2454  1 !       its own.  This routine is normally called when a Type ID (i.e., an RST
;  2349          2455  1 !       pointer to a Type RST Entry) must be produced for such a data type.
;  2350          2456  1 !       The RST entry is added to the Temporary RST Entry List so that it is
;  2351          2457  1 !       released back to the memory pool once there are no longer any references
;  2352          2458  1 !       to it.
;  2353          2459  1 !
;  2354          2460  1 !       The routine builds a Data Type RST Entry and a dummy DST record in the
;  2355          2461  1 !       same memory block.  The DST record is a Type Specification DST record
;  2356          2462  1 !       with the null name and an Indirect Type Specification pointing to the
;  2357          2463  1 !       DST Type Specification given as the input parameter.  The RST entry is
;  2358          2464  1 !       then set to point to this dummy DST record.
;  2359          2465  1 !
;  2360          2466  1 ! INPUTS
;  2361          2467  1 !       DSTADDR - The address of the DST Type Specification for which a Type ID
;  2362          2468  1 !                 is to be generated.
;  2363          2469  1 !
;  2364          2470  1 !       FCODE   - The type Format Code associated with the Type Specification.
;  2365          2471  1 !
;  2366          2472  1 !       BITSIZE - The size in bits of an item of this type.
;  2367          2473  1 !
;  2368          2474  1 ! OUTPUTS
;  2369          2475  1 !       The data type's Type ID (i.e., a pointer to the Type RST Entry) is
;  2370          2476  1 !                 returned as the routine's value.
;  2371          2477  1 !
;  2372          2478  1
;  2373          2479  2     BEGIN
;  2374          2480  2
;  2375          2481  2     LOCAL
;  2376          2482  2         DSTPTR: REF DST$RECORD,          ! Pointer to generated DST record
;  2377          2483  2         RSTPTR: REF RST$ENTRY,           ! Pointer to generated Type RST Entry
;  2378          2484  2         TSPTR: REF DST$TYPE_SPEC;        ! Pointer to Type Spec in DST record
;  2379          2485  2
;  2380          2486  2
;  2381          2487  2
;  2382          2488  2     ! Get a memory block large enough to accommodate both the RST entry and
;  2383          2489  2     ! the dummy DST record.
;  2384          2490  2     !
;  2385          2491  2     RSTPTR = DBG$GET_MEMORY(RST$K_TYPENTSIZ + 3);
;  2386          2492  2     DSTPTR = .RSTPTR + 4*RST$K_TYPENTSIZ;
;  2387          2493  2
;  2388          2494  2
;  2389          2495  2     ! Build the Data Type RST Entry for the input type.
;  2390          2496  2     !
;  2391          2497  2     RSTPTR[RST$L_DSTPTR] = .DSTPTR;
;  2392          2498  2     RSTPTR[RST$L_UPSCOPEPTR] = .RST$START_ADDR;
;  2393          2499  2     RSTPTR[RST$B_KIND] = RST$K_TYPE;
;  2394          2500  2     RSTPTR[RST$B_FCODE] = .FCODE;
;  2395          2501  2     RSTPTR[RST$L_BITSIZE] = .BITSIZE;
;  2396          2502  2
```

```
;  2397          2503  2
;  2398          2504  2      ! Put this Data Type RST Entry on the Temporary RST Entry List.  This will
;  2399          2505  2      ! cause the entry to be released at the end of a DEBUG command when there
;  2400          2506  2      ! are no references to it (when it is not locked).
;  2401          2507  2      !
;  2402          2508  2      RSTPTR[RST$L_HASH_FLINK] = .RST$TEMP_LIST;
;  2403          2509  2      RST$TEMP_LIST = .RSTPTR;
;  2404          2510  2
;  2405          2511  2
;  2406          2512  2      ! Build the dummy DST record for the type.
;  2407          2513  2      !
;  2408          2514  2      DSTPTR[DST$B_LENGTH] = 9;
;  2409          2515  2      DSTPTR[DST$B_TYPE] = DST$K_TYPSPEC;
;  2410          2516  2      DSTPTR[DST$B_TYPSPEC_NAME] = 0;
;  2411          2517  2      TSPTR = .DSTPTR + 3;
;  2412          2518  2      TSPTR[DST$W_TS_LENGTH] = 5;
;  2413          2519  2      TSPTR[DST$B_TS_KIND] = DST$K_TS_IND_TSPEC;
;  2414          2520  2      TSPTR[DST$L_TS_IND_PTR] = DBG$RST_DST_PTR( .DSTADDR - .DST$BEGIN_ADDR );
;  2415          2521  2
;  2416          2522  2
;  2417          2523  2      ! Return the RST pointer to the caller.
;  2418          2524  2      !
;  2419          2525  2      RETURN .RSTPTR;
;  2420          2526  2
;  2421          2527  1      END;


                                001C 00000 TYPEID_FROM_DST_TYPESPEC:
                                           .WORD   Save R2,R3,R4                                   ;  2446
                    54 00000000G   00   9E 00002           MOVAB   RST$TEMP_LIST, R4               ;
                                   0E   DD 00009           PUSHL   #14                             ;  2491
          00000000G   00           01   FB 0000B           CALLS   #1, DBG$GET_MEMORY              ;
                      53           50   D0 00012           MOVL    R0, RSTPTR                      ;
                      50     2C    A3   9E 00015           MOVAB   44(R3), DSTPTR                  ;  2492
                0C    A3           50   D0 00019           MOVL    DSTPTR, 12(RSTPTR)              ;  2497
                10    A3 00000000G   00   D0 0001D           MOVL    RST$START_ADDR, 16(RSTPTR)      ;  2498
                14    A3           07   90 00025           MOVB    #7, 20(RSTPTR)                  ;  2499
                18    A3     08    AC   90 00029           MOVB    FCODE, 24(RSTPTR)               ;  2500
                20    A3     0C    AC   D0 0002E           MOVL    BITSIZE, 32(RSTPTR)             ;  2501
                      63           64   D0 00033           MOVL    RST$TEMP_LIST, (RSTPTR)         ;  2508
                      64           53   D0 00036           MOVL    RSTPTR, RST$TEMP_LIST           ;  2509
                      60     AF09  8F   B0 00039           MOVW    #44809, (DSTPTR)                ;  2514
                             02    A0   94 0003E           CLRB    2(DSTPTR)                       ;  2516
                      52     03    A0   9E 00041           MOVAB   3(R0), TSPTR                    ;  2517
                      62           05   B0 00045           MOVW    #5, (TSPTR)                     ;  2518
                02    A2           0F   90 00048           MOVB    #15, 2(TSPTR)                   ;  2519
             7E 04    AC 00000000G   00   C3 0004C           SUBL3   DST$BEGIN_ADDR, DSTADDR, -(SP)  ;  2520
          00000000G   00           01   FB 00055           CALLS   #1, DBG$RST_DST_PTR             ;
                03    A2           50   D0 0005C           MOVL    R0, 3(TSPTR)                    ;
                      50           53   D0 00060           MOVL    RSTPTR, R0                      ;  2525
                                        04 00063           RET                                     ;  2527

; Routine Size:  100 bytes,    Routine Base:  DBG$CODE + 0D8D
```

```
;  2423          2528  1 ROUTINE TYPEID_FROM_DST_RECORD(DSTADDR, FCODE, BITSIZE) =
;  2424          2529  1 !
;  2425          2530  1 ! FUNCTION
;  2426          2531  1 !       This routine returns a Type ID for the item described by the DST record
;  2427          2532  1 !       pointed to by DSTADDR.
;  2428          2533  1 !
;  2429          2534  1 ! INPUTS
;  2430          2535  1 !       DSTADDR - The DST address of the record defining this item.
;  2431          2536  1 !
;  2432          2537  1 !       FCODE   - The Format Code which corresponds to the type of this item.
;  2433          2538  1 !
;  2434          2539  1 !       BITSIZE - The size in bits of this item.
;  2435          2540  1 !
;  2436          2541  1 ! OUTPUTS
;  2437          2542  1 !       A Type ID for the type of the item is returned as the routine's value.
;  2438          2543  1 !
;  2439          2544  1
;  2440          2545  1
;  2441          2546  2     BEGIN
;  2442          2547  2
;  2443          2548  2     MAP
;  2444          2549  2         DSTADDR: REF DST$RECORD;         ! Pointer to DST record from which we
;  2445          2550  2                                          !      will construct a Type ID
;  2446          2551  2
;  2447          2552  2     LOCAL
;  2448          2553  2         RSTPTR: REF RST$ENTRY;           ! Pointer to generated Type RST Entry
;  2449          2554  2
;  2450          2555  2
;  2451          2556  2
;  2452          2557  2     ! If the data item is atomic, we call DBG$TYPEID_FOR_ATOMIC to build the
;  2453          2558  2     ! desired Data Type RST Entry.  We return its output directly.
;  2454          2559  2     !
;  2455          2560  2     IF .FCODE EQL RST$K_TYPE_ATOMIC
;  2456          2561  2     THEN
;  2457          2562  2         RETURN DBG$TYPEID_FOR_ATOMIC(.DSTADDR[DST$B_TYPE], .BITSIZE, FALSE);
;  2458          2563  2
;  2459          2564  2
;  2460          2565  2     ! Get a memory block to hold the Data Type RST Entry.
;  2461          2566  2     !
;  2462          2567  2     RSTPTR = DBG$GET_MEMORY(RST$K_TYPENTSIZ);
;  2463          2568  2
;  2464          2569  2
;  2465          2570  2     ! Build the Data Type RST Entry for the input type.
;  2466          2571  2     !
;  2467          2572  2     RSTPTR[RST$L_DSTPTR] = .DSTADDR;
;  2468          2573  2     RSTPTR[RST$L_UPSCOPEPTR] = .RST$START_ADDR;
;  2469          2574  2     RSTPTR[RST$B_KIND] = RST$K_TYPE;
;  2470          2575  2     RSTPTR[RST$B_FCODE] = .FCODE;
;  2471          2576  2     RSTPTR[RST$L_BITSIZE] = .BITSIZE;
;  2472          2577  2
;  2473          2578  2
;  2474          2579  2     ! Put this Data Type RST Entry on the Temporary RST Entry List.  This will
;  2475          2580  2     ! cause the entry to be released at the end of a DEBUG command when there
;  2476          2581  2     ! are no references to it (when it is not locked).
;  2477          2582  2     !
;  2478          2583  2     RSTPTR[RST$L_HASH_FLINK] = .RST$TEMP_LIST;
;  2479          2584  2     RST$TEMP_LIST = .RSTPTR;
```

```
;  2480          2585  2
;  2481          2586  2
;  2482          2587  2      ! Return the RST pointer to the caller.
;  2483          2588  2      !
;  2484          2589  2      RETURN .RSTPTR;
;  2485          2590  2
;  2486          2591  1      END;


                                        0004 00000 TYPEID_FROM_DST_RECORD:
                                                   .WORD   Save R2                                  ; 2528
                          52 00000000G   00   9E 00002         MOVAB   RST$TEMP_LIST, R2
                          02        08   AC   D1 00009         CMPL    FCODE, #2                    ; 2560
                                         13   12 0000D         BNEQ    1$
                                         7E   D4 0000F         CLRL    -(SP)                        ; 2562
                                   0C    AC   DD 00011         PUSHL   BITSIZE
                          50        04   AC   D0 00014         MOVL    DSTADDR, R0
                          7E        01   A0   9A 00018         MOVZBL  1(R0), -(SP)
                    FC12  CF              03   FB 0001C         CALLS   #3, DBG$TYPEID_FOR_ATOMIC
                                         04   00021         RET
                                         0B   DD 00022 1$:     PUSHL   #11                          ; 2567
                00000000G   00           01   FB 00024         CALLS   #1, DBG$GET_MEMORY
                      0C    A0        04   AC   D0 0002B         MOVL    DSTADDR, 12(RSTPTR)          ; 2572
                      10    A0 00000000G   00   D0 00030         MOVL    RST$START_ADDR, 16(RSTPTR)   ; 2573
                      14    A0             07   90 00038         MOVB    #7, 20(RSTPTR)               ; 2574
                      18    A0        08   AC   90 0003C         MOVB    FCODE, 24(RSTPTR)            ; 2575
                      20    A0        0C   AC   D0 00041         MOVL    BITSIZE, 32(RSTPTR)          ; 2576
                                   60        62   D0 00046         MOVL    RST$TEMP_LIST, (RSTPTR)      ; 2583
                                   62        50   D0 00049         MOVL    RSTPTR, RST$TEMP_LIST        ; 2584
                                         04   0004C         RET                                  ; 2591

; Routine Size:  77 bytes,    Routine Base:  DBG$CODE + 0DF1
```

```
;  2488          2592  1 ROUTINE FIND_TYPE_RECORD(SYMID, FCODE, BITSIZE) =
;  2489          2593  1 !
;  2490          2594  1 ! FUNCTION
;  2491          2595  1 !       This routine takes a symid, and tracks down the type spec which
;  2492          2596  1 !       describes it. In the process, it determines the Format Code and
;  2493          2597  1 !       bit size of the item.
;  2494          2598  1 !
;  2495          2599  1 ! INPUTS
;  2496          2600  1 !       SYMID   - The SYMID of the item whose type we want.  (Note that the
;  2497          2601  1 !                 SYMID may itself represent a named type.)
;  2498          2602  1 !
;  2499          2603  1 !       FCODE   - The address of a longword to receive the format code of
;  2500          2604  1 !                 the specified item.
;  2501          2605  1 !
;  2502          2606  1 !       BITSIZE - The address of a longword to receive the size in bits of
;  2503          2607  1 !                 the specified item.
;  2504          2608  1 !
;  2505          2609  1 ! OUTPUTS
;  2506          2610  1 !       FCODE   - The format code of the item is returned to FCODE.
;  2507          2611  1 !
;  2508          2612  1 !       BITSIZE - The size in bits of the item is returned to BITSIZE.
;  2509          2613  1 !
;  2510          2614  1 !       The data type's Type ID (i.e., a pointer to the Type RST Entry) is
;  2511          2615  1 !                 returned as the routine's value; zero is returned if no
;  2512          2616  1 !                 type exists.
;  2513          2617  1 !
;  2514          2618  1
;  2515          2619  2     BEGIN
;  2516          2620  2
;  2517          2621  2     MAP
;  2518          2622  2         SYMID: REF RST$ENTRY;
;  2519          2623  2
;  2520          2624  2     LOCAL
;  2521          2625  2         CLASS,
;  2522          2626  2         DESC_PTR: REF BLOCK[,BYTE],
;  2523          2627  2         DST_PTR: REF DST$RECORD,
;  2524          2628  2         HAVE_TYPE,
;  2525          2629  2         KIND,
;  2526          2630  2         LENGTH,
;  2527          2631  2         SAVEFCODE,
;  2528          2632  2         SAVEID: REF RST$ENTRY,
;  2529          2633  2         SIZE,
;  2530          2634  2         TYPEID: REF RST$ENTRY,
;  2531          2635  2         TYPE_PTR: REF DST$RECORD,
;  2532          2636  2         TYPE_SPEC: REF DST$TYPE_SPEC,
;  2533          2637  2         VALKIND,
;  2534          2638  2         VAL_PTR,
;  2535          2639  2         VAL_VECT: VECTOR[3];
;  2536          2640  2
;  2537          2641  2
;  2538          2642  2     NOVEL_LENGTH_FLAG = FALSE;
;  2539          2643  2     DBG$STA_SYMKIND(.SYMID, KIND);
;  2540          2644  2     SELECTONE .KIND OF
;  2541          2645  2         SET
;  2542          2646  2         [RST$K_TYPE]:
;  2543          2647  3             BEGIN
;  2544          2648  3             .FCODE = .SYMID[RST$B_FCODE];
```

```
;  2545          2649  3              .BITSIZE = .SYMID[RST$L_BITSIZE];
;  2546          2650  3              RETURN .SYMID;
;  2547          2651  2              END;
;  2548          2652  2
;  2549          2653  2          [RST$K_TYPCOMP, RST$K_DATA]:
;  2550          2654  3              BEGIN
;  2551          2655  3              TYPEID = .SYMID[RST$L_TYPEPTR];
;  2552          2656  3              IF .TYPEID NEQ 0
;  2553          2657  3              THEN
;  2554          2658  4                  BEGIN
;  2555          2659  4                  HAVE_TYPE = TRUE;
;  2556          2660  4                  SAVEFCODE = .TYPEID[RST$B_FCODE];
;  2557          2661  4                  SAVEID = .TYPEID;
;  2558          2662  4                  END
;  2559          2663  4
;  2560          2664  3              ELSE
;  2561          2665  3                  HAVE_TYPE = FALSE;
;  2562          2666  3
;  2563          2667  3              DST_PTR = .SYMID[RST$L_DSTPTR];
;  2564          2668  3              SELECTONE .DST_PTR[DST$B_TYPE] OF
;  2565          2669  3                  SET
;  2566          2670  3                  [DST$K_SEPTYP]:
;  2567          2671  4                      BEGIN
;  2568          2672  4
;  2569          2673  4
;  2570          2674  4                      ! Normally, TYPE_SPEC returned from DBG$FIND_SEPTYP does
;  2571          2675  4                      ! not mean anything if TYPE_PTR[DST$B_TYPE] EQL
;  2572          2676  4                      ! DST$K_ENUMBEG or DST$K_RECBEG.  In here this case
;  2573          2677  4                      ! is caught early by HAVE_TYPE set to TRUE.  In other
;  2574          2678  4                      ! words, in this routine, DST$K_ENUMBEG or DST$K_RECBEG
;  2575          2679  4                      ! always comes in to have TYPEID. (We hope).
;  2576          2680  4                      !
;  2577          2681  4                      SIZE = DBG$FIND_SEPTYP(.DST_PTR, TYPE_PTR, TYPE_SPEC);
;  2578          2682  4                      .BITSIZE = .SIZE;
;  2579          2683  4                      IF .HAVE_TYPE
;  2580          2684  4                      THEN
;  2581          2685  5                          BEGIN
;  2582          2686  5                          .FCODE = .SAVEFCODE;
;  2583          2687  5
;  2584          2688  5
;  2585          2689  5                          ! The novel length flags gets set earlier, in
;  2586          2690  5                          ! FIND_TYPSPEC, if we encounterd a novel length
;  2587          2691  5                          ! type spec on the way to our "real" type_spec.
;  2588          2692  5                          ! If this flag is set, and if the "novel" size
;  2589          2693  5                          ! is different from the size we have in the
;  2590          2694  5                          ! typeid, then we build a new typeid with
;  2591          2695  5                          ! the novel length in it.
;  2592          2696  5                          !
;  2593          2697  5                          IF .NOVEL_LENGTH_FLAG AND
;  2594          2698  5                            (.SAVEFCODE EQL RST$K_TYPE_ENUM) AND
;  2595          2699  6                            (.SIZE NEQ .TYPEID[RST$L_BITSIZE])
;  2596          2700  5                          THEN
;  2597          2701  6                              BEGIN
;  2598          2702  6                              OWN
;  2599          2703  6                                  DUMMY: VECTOR[2];
;  2600          2704  6                              LOCAL
;  2601          2705  6                                  LAST_SYMID: REF RST$ENTRY,
```

```
;  2602          2706  6                    TEMP_SYMID: REF RST$ENTRY;
;  2603          2707  6
;  2604          2708  6                LENGTH = RST$K_TYPENTSIZ + .TYPEID[RST$L_TYPCOMPCNT];
;  2605          2709  6                SAVEID = DBG$GET_MEMORY(.LENGTH + 3);
;  2606          2710  6
;  2607          2711  6
;  2608          2712  6                ! Make the copy.
;  2609          2713  6                !
;  2610          2714  6                CH$MOVE(.LENGTH*4, .TYPEID, .SAVEID);
;  2611          2715  6
;  2612          2716  6
;  2613          2717  6                ! Modify the typeid. We create a "dummy" hash
;  2614          2718  6                ! chain for it.
;  2615          2719  6                !
;  2616          2720  6                SAVEID[RST$L_UPSCOPEPTR] = .RST$START_ADDR;
;  2617          2721  6                SAVEID[RST$L_BITSIZE] = .SIZE;
;  2618          2722  6                DUMMY[0] = .SAVEID;
;  2619          2723  6                DUMMY[1] = .SAVEID;
;  2620          2724  6                SAVEID[RST$L_HASH_FLINK] = .DUMMY;
;  2621          2725  6                SAVEID[RST$L_HASH_BLINK] = .DUMMY;
;  2622          2726  6                SAVEID[RST$L_SYMCHNPTR] = 0;
;  2623          2727  6
;  2624          2728  6                ! We put the new SAVEID typeid on the symchain
;  2625          2729  6                ! for this module so it gets freed up when we
;  2626          2730  6                ! cancel the module.
;  2627          2731  6                !
;  2628          2732  6                TEMP_SYMID = .SYMID;
;  2629          2733  6                WHILE .TEMP_SYMID NEQ 0 DO
;  2630          2734  7                    BEGIN
;  2631          2735  7                    LAST_SYMID = .TEMP_SYMID;
;  2632          2736  7                    TEMP_SYMID = .TEMP_SYMID[RST$L_SYMCHNPTR];
;  2633          2737  6                    END;
;  2634          2738  6                LAST_SYMID[RST$L_SYMCHNPTR] = .SAVEID;
;  2635          2739  6                SYMID[RST$L_TYPEPTR] = .SAVEID;
;  2636          2740  5                END;
;  2637          2741  5
;  2638          2742  5            RETURN .SAVEID;
;  2639          2743  4            END;
;  2640          2744  4
;  2641          2745  4        ! If we have a descriptor type, then we need to check
;  2642          2746  4        ! whether this is an array descriptor.
;  2643          2747  4        !
;  2644          2748  5        IF  (.TYPE_SPEC[DST$B_TS_KIND] EQL DST$K_TS_DSC)
;  2645          2749  4        THEN
;  2646          2750  5            BEGIN
;  2647          2751  5            VAL_PTR = TYPE_SPEC[DST$A_TS_DSC_VSPEC_ADDR];
;  2648          2752  5            DBG$STA_VALSPEC(.VAL_PTR, VAL_VECT, VAL[KIND);
;  2649          2753  5            IF .VAL[KIND NEQ DBG$K_VAL_DESCR
;  2650          2754  5            THEN
;  2651          2755  5                $DBG_ERROR('RSTTYPES\FIND_TYPE_RECORD 20');
;  2652          2756  5
;  2653          2757  5            DESC_PTR = .VAL_VECT[0];
;  2654          2758  5            DBG$READ_ACCESS(.DESC_PTR, 12);
;  2655          2759  5            CLASS = .DESC_PTR[DSC$B_CLASS];
;  2656          2760  5            IF (.CLASS EQL DSC$K_CLASS_A)   OR
;  2657          2761  5               (.CLASS EQL DSC$K_CLASS_NCA) OR
;  2658          2762  5               (.CLASS EQL DSC$K_CLASS_UBA) OR
```

```
;  2659          2763  6                    (.CLASS EQL DSC$K_CLASS_VSA)
;  2660          2764  5                THEN
;  2661          2765  6                    BEGIN
;  2662          2766  6                    .FCODE = RST$K_TYPE_ARRAY;
;  2663          2767  6                    RETURN TYPEID_FOR_DESCR(.DESC_PTR, ..FCODE);
;  2664          2768  6                    END
;  2665          2769  5                ELSE
;  2666          2770  6                    BEGIN
;  2667          2771  6                    .FCODE = RST$K_TYPE_DESCR;
;  2668          2772  6                    RETURN TYPEID_FOR_DESCR(.DESC_PTR, ..FCODE);
;  2669          2773  5                    END;
;  2670          2774  4                END;
;  2671          2775  4
;  2672          2776  4            .FCODE = DBG$TRANS_TYPE_CODE(.TYPE_SPEC[DST$B_TS_KIND]);
;  2673          2777  4            IF ..FCODE EQL RST$K_TYPE_ATOMIC
;  2674          2778  4            THEN
;  2675          2779  4                RETURN DBG$TYPEID_FOR_ATOMIC(
;  2676          2780  4                        .TYPE_SPEC[DST$B_TS_ATOM_TYP], .SIZE, FALSE)
;  2677          2781  4
;  2678          2782  4            ELSE
;  2679          2783  4                RETURN TYPEID_FROM_DST_RECORD(.TYPE_PTR,..FCODE,..SIZE);
;  2680          2784  4
;  2681          2785  3            END;
;  2682          2786  3
;  2683          2787  3        [DST$K_BLI]:
;  2684          2788  4            BEGIN
;  2685          2789  4            .FCODE = RST$K_TYPE_BLIDATA;
;  2686          2790  4            .BITSIZE = GET_BITSIZE_FROM_BLIDST(.SYMID[RST$L_DSTPTR]);
;  2687          2791  4            RETURN DBG$TYPEID_FOR_ATOMIC(RST$K_TYPE_BLIDATA,
;  2688          2792  4                    ..BITSIZE, FALSE);
;  2689          2793  3            END;
;  2690          2794  3
;  2691          2795  3        [DST$K_ENUMELT, DST$K_RECBEG]:
;  2692          2796  4            BEGIN
;  2693          2797  4            .FCODE = .SAVEID[RST$B_FCODE];
;  2694          2798  4            .BITSIZE = .SAVEID[RST$L_BITSIZE];
;  2695          2799  4            RETURN .SAVEID;
;  2696          2800  3            END;
;  2697          2801  3
;  2698          2802  3
;  2699          2803  3        ! Handle the COBOL Hack DST Record.  Here we call a routine
;  2700          2804  3        ! which converts that DST record into a more regular Type RST
;  2701          2805  3        ! Entry.  We then get the FCODE from that RST entry and re-
;  2702          2806  3        ! turn the entry's address as a Type ID.
;  2703          2807  3        !
;  2704          2808  3        [DST$K_COB_HACK]:
;  2705          2809  4            BEGIN
;  2706          2810  4            TYPEID = TYPEID_FOR_COB_HACK(.DST_PTR);
;  2707          2811  4            .FCODE = .TYPEID[RST$B_FCODE];
;  2708          2812  4            .BITSIZE = 0;
;  2709          2813  4            RETURN .TYPEID;
;  2710          2814  3            END;
;  2711          2815  3
;  2712          2816  3        [DST$K_BLIFLD]:
;  2713          2817  4            BEGIN
;  2714          2818  4            .FCODE = RST$K_TYPE_BLIFLD;
;  2715          2819  4            .BITSIZE = 0;
```

```
;  2716          2820  4                 RETURN DBG$TYPEID_FOR_ATOMIC(RST$K_TYPE_BLIFLD,0,FALSE);
;  2717          2821  3                 END;
;  2718          2822  3
;  2719          2823  3             [DST$K_LBLORLIT]:
;  2720          2824  4                 BEGIN
;  2721          2825  4                 .FCODE = RST$K_TYPE_ATOMIC;
;  2722          2826  4                 .BITSIZE = 32;
;  2723          2827  4                 RETURN DBG$TYPEID_FOR_ATOMIC(DSC$K_DTYPE_LU,32,FALSE);
;  2724          2828  3                 END;
;  2725          2829  3
;  2726          2830  3             [OTHERWISE]:
;  2727          2831  4                 BEGIN
;  2728          2832  5                 IF (.DST_PTR[DST$B_TYPE] LSS DSC$K_DTYPE_LOWEST OR
;  2729          2833  4                     .DST_PTR[DST$B_TYPE] GTR DSC$K_DTYPE_HIGHEST) AND
;  2730          2834  4                    (.DST_PTR[DST$B_TYPE] NEQ DST$K_BOOL) AND
;  2731          2835  5                    (.DST_PTR[DST$B_TYPE] NEQ DSC$K_DTYPE_TF)
;  2732          2836  4                 THEN
;  2733          2837  4                     $DBG_ERROR('RSTTYPES\FIND_TYPE_RECORD');
;  2734          2838  4
;  2735          2839  4                 .FCODE = DESC_OR_ARRAY_OR_ATOM(.SYMID, DESC_PTR);
;  2736          2840  4                 IF ..FCODE EQL RST$K_TYPE_ATOMIC
;  2737          2841  4                 THEN
;  2738          2842  5                     BEGIN
;  2739          2843  5                     .BITSIZE = GET_BITSIZE_FROM_DTYPE(.DST_PTR[DST$B_TYPE]);
;  2740          2844  5                     RETURN DBG$TYPEID_FOR_ATOMIC(.DST_PTR[DST$B_TYPE],
;  2741          2845  5                                 ..BITSIZE,FALSE);
;  2742          2846  5                     END
;  2743          2847  5
;  2744          2848  4                 ELSE
;  2745          2849  5                     BEGIN
;  2746          2850  5                     .BITSIZE = 0;
;  2747          2851  5                     RETURN TYPEID_FOR_DESCR(.DESC_PTR, ..FCODE);
;  2748          2852  4                     END;
;  2749          2853  4
;  2750          2854  3                 END;
;  2751          2855  3
;  2752          2856  3             TES;
;  2753          2857  3
;  2754          2858  2         END;
;  2755          2859  2
;  2756          2860  2     [RST$K_VARIANT]:
;  2757          2861  3         BEGIN
;  2758          2862  3         LOCAL
;  2759          2863  3             VAR_PTR: REF DST$VARBEG_TRAILER;
;  2760          2864  3
;  2761          2865  3         .FCODE = RST$K_VARIANT;
;  2762          2866  3         DST_PTR = .SYMID[RST$L_DSTPTR];
;  2763          2867  3         VAR_PTR = DST_PTR[DST$B_NAME] + 1 + .DST_PTR[DST$B_NAME];
;  2764          2868  3         .BITSIZE = .VAR_PTR[DST$L_VARBEG_SIZE];
;  2765          2869  3         RETURN .SYMID;
;  2766          2870  2         END;
;  2767          2871  2
;  2768          2872  2     [OTHERWISE]:
;  2769          2873  2         RETURN 0;
;  2770          2874  2
;  2771          2875  2     TES;
;  2772          2876  2
```

```
; 2773          2877  1     END;


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

5F 44 4E 49 46 5C 53 45 50 59 54 54 53 52 1C  00184 P.AAT:    .ASCII  <28>\RSTTYPES\<92>\FIND_TYPE_RECORD 20\      ;
   30 32 20 44 52 4F 43 45 52 5F 45 50 59 54  00193
5F 44 4E 49 46 5C 53 45 50 59 54 54 53 52 19  001A1 P.AAU:    .ASCII  <25>\RSTTYPES\<92>\FIND_TYPE_RECORD\         ;
            44 52 4F 43 45 52 5F 45 50 59 54  001B0

                                                                .PSECT  DBG$OWN,NOEXE,  PIC,2

                                                        00004 DUMMY:    .BLKB   8


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

                                                    0FFC 00000 FIND_TYPE_RECORD:
                                                               .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11        ; 2592
                                  5B 00000000G   00    9E 00002          MOVAB   LIB$SIGNAL, R11                    ;
                                  5A 00000000'   EF    9E 00009          MOVAB   DUMMY, R10                         ;
                                  5E             20    C2 00010          SUBL2   #32, SP                            ;
                                                 FC    AA D4 00013       CLRL    NOVEL_LENGTH_FLAG                  ; 2642
                                                       5E DD 00016       PUSHL   SP                                 ; 2643
                                  56             04    AC D0 00018       MOVL    SYMID, R6                          ;
                                                       56 DD 0001C       PUSHL   R6                                 ;
                          00000000G   00               02 FB 0001E       CALLS   #2, DBG$STA_SYMKIND                ;
                                  50                   6E D0 00025       MOVL    KIND, R0                           ; 2644
                                  07                   50 D1 00028       CMPL    R0, #7                             ; 2646
                                                       0D 12 0002B       BNEQ    1$                                 ;
                              08  BC             18    A6 9A 0002D       MOVZBL  24(R6), @FCODE                     ; 2648
                              0C  BC             20    A6 D0 00032       MOVL    32(R6), @BITSIZE                   ; 2649
                                                 023D  31 00037          BRW     32$                                ; 2650
                                  06                   50 D1 0003A 1$:   CMPL    R0, #6                             ; 2653
                                                       08 13 0003D       BEQL    2$                                 ;
                                  0A                   50 D1 0003F       CMPL    R0, #10                            ;
                                                       03 13 00042       BEQL    2$                                 ;
                                                 0216  31 00044          BRW     31$                                ;
                                  53             18    A6 D0 00047 2$:   MOVL    24(R6), TYPEID                     ; 2655
                                                       0C 13 0004B       BEQL    3$                                 ; 2656
                                  58                   01 D0 0004D       MOVL    #1, HAVE_TYPE                      ; 2659
                                  55             18    A3 9A 00050       MOVZBL  24(TYPEID), SAVEFCODE              ; 2660
                                  57                   53 D0 00054       MOVL    TYPEID, SAVEID                     ; 2661
                                                       02 11 00057       BRB     4$                                 ; 2656
                                                       58 D4 00059 3$:   CLRL    HAVE_TYPE                          ; 2665
                                  52             0C    A6 D0 0005B 4$:   MOVL    12(R6), DST_PTR                    ; 2667
                                  54             01    A2 9A 0005F       MOVZBL  1(DST_PTR), R4                     ; 2668
                              A3  8F                   54 91 00063       CMPB    R4, #T63                           ; 2670
                                                       03 13 00067       BEQL    5$                                 ;
                                                 0116  31 00069          BRW     16$                                ;
                                                 04    AE 9F 0006C 5$:   PUSHAB  TYPE_SPEC                         ; 2681
                                                 0C    AE 9F 0006F       PUSHAB  TYPE_PTR                           ;
                                                       52 DD 00072       PUSHL   DST_PTR                            ;
                          F149  CF                     03 FB 00074       CALLS   #3, DBG$FIND_SEPTYP                ;
                                  59                   50 D0 00079       MOVL    R0, SIZE                           ;
```

```
                0C   BC            59  D0 0007C          MOVL    SIZE, @BITSIZE                          ; 2682
                62                 58  E9 00080          BLBC    HAVE_TYPE, 9$                           ; 2683
                08   BC            55  D0 00083          MOVL    SAVEFCODE, @FCODE                       ; 2686
                     57       FC   AA  E9 00087          BLBC    NOVEL_LENGTH_FLAG, 8$                   ; 2697
                     04            55  D1 0008B          CMPL    SAVEFCODE, #4                           ; 2698
                                   52  12 0008E          BNEQ    8$                                      ;
                     20   A3       59  D1 00090          CMPL    SIZE, 32(TYPEID)                        ; 2699
                                   4C  13 00094          BEQL    8$                                      ;
52                   28   A3       0B  C1 00096          ADDL3   #11, 40(TYPEID), LENGTH                 ; 2708
                          03   A2  9F  00099            PUSHAB  3(LENGTH)                               ; 2709
    00000000G     00             01  FB 0009E          CALLS   #1, DBG$GET_MEMORY                      ;
                     57            50  D0 000A5          MOVL    R0, SAVEID                              ;
                     52            04  C4 000A8          MULL2   #4, R2                                  ; 2714
67                   63            52  28 000AB          MOVC3   R2, (TYPEID), (SAVEID)                  ;
                10   A7 00000000G  00  D0 000AF          MOVL    RST$START_ADDR, 16(SAVEID)              ; 2720
                20   A7            59  D0 000B7          MOVL    SIZE, 32(SAVEID)                        ; 2721
                     6A            57  D0 000BB          MOVL    SAVEID, DUMMY                           ; 2722
                04   AA            57  D0 000BE          MOVL    SAVEID, DUMMY+4                         ; 2723
                     67            6A  D0 000C2          MOVL    DUMMY, (SAVEID)                         ; 2724
                04   A7            6A  D0 000C5          MOVL    DUMMY, 4(SAVEID)                        ; 2725
                          08   A7  D4  000C9            CLRL    8(SAVEID)                               ; 2726
                     50            56  D0 000CC          MOVL    R6, TEMP_SYMID                          ; 2732
                                   09  13 000CF  6$:      BEQL    7$                                      ; 2733
                     51            50  D0 000D1          MOVL    TEMP_SYMID, LAST_SYMID                  ; 2735
                     50       08   A0  D0 000D4          MOVL    8(TEMP_SYMID), TEMP_SYMID               ; 2736
                                   F5  11 000D8          BRB     6$                                      ; 2733
                08   A1            57  D0 000DA  7$:      MOVL    SAVEID, 8(LAST_SYMID)                   ; 2738
                18   A6            57  D0 000DE          MOVL    SAVEID, 24(R6)                          ; 2739
                                 00D0  31 000E2  8$:      BRW     19$                                     ; 2742
                     52       04   AE  D0 000E5  9$:      MOVL    TYPE_SPEC, R2                           ; 2748
                     02       02   A2  91 000E9          CMPB    2(R2), #2                               ;
                                   68  12 000ED          BNEQ    14$                                     ;
                     50       03   A2  9E 000EF          MOVAB   3(R2), VAL_PTR                          ; 2751
                              0C   AE  9F 000F3          PUSHAB  VALKIND                                 ; 2752
                              18   AE  9F 000F6          PUSHAB  VAL_VECT                                ;
                                   50  DD 000F9          PUSHL   VAL_PTR                                 ;
    00000000G     00             03  FB 000FB          CALLS   #3, DBG$STA_VALSPEC                     ;
                     03       0C   AE  D1 00102          CMPL    VALKIND, #3                             ; 2753
                                   11  13 00106          BEQL    10$                                     ;
                          00000000'  EF  9F 00108          PUSHAB  P.AAT                                   ; 2755
                                   01  DD 0010E          PUSHL   #1                                      ;
                          00028362   8F  DD 00110          PUSHL   #164706                                 ;
                     6B            03  FB 00116          CALLS   #3, LIB$SIGNAL                          ;
                10   AE       14   AE  D0 00119  10$:      MOVL    VAL_VECT, DESC_PTR                      ; 2757
                              0C   DD 0011E            PUSHL   #12                                     ; 2758
                     53       14   AE  D0 00120          MOVL    DESC_PTR, R3                            ;
                                   53  DD 00124          PUSHL   R3                                      ;
    00000000G     00             02  FB 00126          CALLS   #2, DBG$READ_ACCESS                     ;
                     50       03   A3  9A 0012D          MOVZBL  3(R3), CLASS                            ; 2759
                     04            50  D1 00131          CMPL    CLASS, #4                               ; 2760
                                   0F  13 00134          BEQL    11$                                     ;
                     0A            50  D1 00136          CMPL    CLASS, #10                              ; 2761
                                   0A  13 00139          BEQL    11$                                     ;
                     0E            50  D1 0013B          CMPL    CLASS, #14                              ; 2762
                                   05  13 0013E          BEQL    11$                                     ;
                     0C            50  D1 00140          CMPL    CLASS, #12                              ; 2763
                                   06  12 00143          BNEQ    12$                                     ;
```

```
                08   BC              01   D0 00145 11$:   MOVL    #1, @FCODE                      ; 2766
                                     04   11 00149         BRB     13$                             ; 2767
                08   BC              03   D0 0014B 12$:   MOVL    #3, @FCODE                      ; 2771
                                08   BC   DD 0014F 13$:   PUSHL   @FCODE                          ; 2772
                                     53   DD 00152         PUSHL   R3                              ;
                                0100 31   00154         BRW     30$                             ;
                     7E         02   A2   9A 00157 14$:   MOVZBL  2(R2), -(SP)                   ; 2776
              F94E   CF              01   FB 0015B         CALLS   #1, DBG$TRANS_TYPE_CODE         ;
                08   BC              50   D0 00160         MOVL    R0, @FCODE                      ;
                     02         08   BC   D1 00164         CMPL    @FCODE, #2                      ; 2777
                                     0A   12 00168         BNEQ    15$                             ;
                                     7E   D4 0016A         CLRL    -(SP)                           ; 2779
                                     59   DD 0016C         PUSHL   SIZE                            ; 2780
                     7E         03   A2   9A 0016E         MOVZBL  3(R2), -(SP)                    ;
                                     72   11 00172         BRB     22$                             ;
                                     59   DD 00174 15$:   PUSHL   SIZE                            ; 2783
                                08   BC   DD 00176         PUSHL   @FCODE                          ;
                                     10   AE   DD 00179         PUSHL   TYPE_PTR                    ;
              FE32   CF              03   FB 0017C         CALLS   #3, TYPEID_FROM_DST_RECORD      ;
                                     04   00181         RET                                     ;
                                     54   D5 00182 16$:   TSTL    R4                              ; 2787
                                     19   12 00184         BNEQ    17$                             ;
                08   BC              0D   D0 00186         MOVL    #13, @FCODE                     ; 2789
                                0C   A6   DD 0018A         PUSHL   12(R6)                          ; 2790
              0000V  CF              01   FB 0018D         CALLS   #1, GET_BITSIZE_FROM_BLIDST     ;
                0C   BC              50   D0 00192         MOVL    R0, @BITSIZE                    ;
                                     7E   D4 00196         CLRL    -(SP)                           ; 2791
                                0C   BC   DD 00198         PUSHL   @BITSIZE                        ; 2792
                                     0D   DD 0019B         PUSHL   #13                             ; 2791
                                     5C   11 0019D         BRB     24$                             ;
                A4   8F              54   91 0019F 17$:   CMPB    R4, #164                        ; 2795
                                     06   13 001A3         BEQL    18$                             ;
                AB   8F              54   91 001A5         CMPB    R4, #171                        ;
                                     0E   12 001A9         BNEQ    20$                             ;
                08   BC         18   A7   9A 001AB 18$:   MOVZBL  24(SAVEID), @FCODE              ; 2797
                0C   BC         20   A7   D0 001B0         MOVL    32(SAVEID), @BITSIZE            ; 2798
                     50              57   D0 001B5 19$:   MOVL    SAVEID, R0                      ; 2799
                                     04   001B8         RET                                     ;
                B2   8F              54   91 001B9 20$:   CMPB    R4, #178                        ; 2808
                                     16   12 001BD         BNEQ    21$                             ;
                                     52   DD 001BF         PUSHL   DST_PTR                         ; 2810
              FBD1   CF              01   FB 001C1         CALLS   #1, TYPEID_FOR_COB_HACK         ;
                     53              50   D0 001C6         MOVL    R0, TYPEID                      ;
                08   BC         18   A3   9A 001C9         MOVZBL  24(TYPEID), @FCODE              ; 2811
                                0C   BC   D4 001CE         CLRL    @BITSIZE                        ; 2812
                     50              53   D0 001D1         MOVL    TYPEID, R0                      ; 2813
                                     04   001D4         RET                                     ;
                B7   8F              54   91 001D5 21$:   CMPB    R4, #183                        ; 2816
                                     0D   12 001D9         BNEQ    23$                             ;
                08   BC              0E   D0 001DB         MOVL    #14, @FCODE                     ; 2818
                                0C   BC   D4 001DF         CLRL    @BITSIZE                        ; 2819
                                     7E   7C 001E2         CLRQ    -(SP)                           ; 2820
                                     0E   DD 001E4         PUSHL   #14                             ;
                                     60   11 001E6 22$:   BRB     28$                             ;
                BA   8F              54   91 001E8 23$:   CMPB    R4, #186                        ; 2823
                                     0F   12 001EC         BNEQ    25$                             ;
                08   BC              02   D0 001EE         MOVL    #2, @FCODE                      ; 2825
```

```
                0C   BC             20  D0 001F2          MOVL    #32, @BITSIZE                       ; 2826
                     7E             20  7D 001F6          MOVQ    #32, -(SP)                          ; 2827
                                    04  DD 001F9          PUSHL   #4                                  ;
                                    4B  11 001FB 24$:      BRB     28$                                 ;
                                    54  D5 001FD 25$:      TSTL    R4                                  ; 2832
                                    05  15 001FF          BLEQ    26$                                 ;
                     25             54  91 00201          CMPB    R4, #37                             ; 2833
                                    1C  1B 00204          BLEQU   27$                                 ;
                9E   8F             54  91 00206 26$:      CMPB    R4, #158                            ; 2834
                                    16  13 0020A          BEQL    27$                                 ;
                     28             54  91 0020C          CMPB    R4, #40                             ; 2835
                                    11  13 0020F          BEQL    27$                                 ;
                         00000000'  EF  9F 00211          PUSHAB  P.AAU                               ; 2837
                                    01  DD 00217          PUSHL   #1                                  ;
                         00028362   8F  DD 00219          PUSHL   #164706                             ;
                     6B              03  FB 0021F          CALLS   #3, LIB$SIGNAL                      ;
                              10     AE  9F 00222 27$:      PUSHAB  DESC_PTR                            ; 2839
                                    56  DD 00225          PUSHL   R6                                  ;
              0000V  CF             02  FB 00227          CALLS   #2, DESC_OR_ARRAY_OR_ATOM           ;
                08   BC             50  D0 0022C          MOVL    R0, @FCODE                          ;
                     02        08   BC  D1 00230          CMPL    @FCODE, #2                          ; 2840
                                    18  12 00234          BNEQ    29$                                 ;
                                    54  DD 00236          PUSHL   R4                                  ; 2843
              0000V  CF             01  FB 00238          CALLS   #1, GET_BITSIZE_FROM_DTYPE          ;
                0C   BC             50  D0 0023D          MOVL    R0, @BITSIZE                        ;
                                    7E  D4 00241          CLRL    -(SP)                               ; 2844
                              0C    BC  DD 00243          PUSHL   @BITSIZE                            ; 2845
                                    54  DD 00246          PUSHL   R4                                  ; 2844
              F999   CF             03  FB 00248 28$:      CALLS   #3, DBG$TYPEID_FOR_ATOMIC           ;
                                    04  0024D            RET                                         ; 2849
                           0C   BC  D4  0024E 29$:        CLRL    @BITSIZE                            ; 2850
                           08   BC  DD  00251             PUSHL   @FCODE                              ; 2851
                           14   AE  DD  00254             PUSHL   DESC_PTR                            ;
              FC31   CF             02  FB 00257 30$:      CALLS   #2, TYPEID_FOR_DESCR                ;
                                    04  0025C            RET                                         ; 2849
                     0B             50  D1 0025D 31$:      CMPL    R0, #11                             ; 2860
                                    19  12 00260          BNEQ    33$                                 ;
                08   BC             0B  D0 00262          MOVL    #11, @FCODE                         ; 2865
                     52        0C   A6  D0 00266          MOVL    12(R6), DST_PTR                     ; 2866
                     50        07   A2  9A 0026A          MOVZBL  7(DST_PTR), R0                      ; 2867
                     50        08 A042  9E 0026E          MOVAB   8(R0)[DST_PTR], VAR_PTR             ;
                0C   BC             60  D0 00273          MOVL    (VAR_PTR), @BITSIZE                 ; 2868
                     50             56  D0 00277 32$:      MOVL    R6, R0                              ; 2869
                                    04  0027A            RET                                         ;
                                    50  D4 0027B 33$:      CLRL    R0                                  ; 2873
                                    04  0027D            RET                                         ; 2877

; Routine Size:  638 bytes,    Routine Base:  DBG$CODE + 0E3E
```

```
;  2775          2878  1 ROUTINE DESC_OR_ARRAY_OR_ATOM(SYMID, DESC_PTR) =
;  2776          2879  1 !
;  2777          2880  1 ! FUNCTION
;  2778          2881  1 !       This routine takes a SYMID and determines whether it is of type atomic,
;  2779          2882  1 !       array, or descriptor.
;  2780          2883  1 !
;  2781          2884  1 ! INPUTS
;  2782          2885  1 !       SYMID   - The SYMID in question.
;  2783          2886  1 !
;  2784          2887  1 !       DESC_PTR - The address of a longword location to receive the address
;  2785          2888  1 !                  of the item's descriptor (if present).
;  2786          2889  1 !
;  2787          2890  1 ! OUTPUTS
;  2788          2891  1 !       DESC_PTR - For return values of RST$K_TYPE_DESCR and RST$K_TYPE_ARRAY,
;  2789          2892  1 !                  DESC_PTR receives the address of the item's descriptor.
;  2790          2893  1 !
;  2791          2894  1 !       The routine's value will be one of RST$K_TYPE_ATOMIC, RST$K_TYPE_DESCR,
;  2792          2895  1 !                  or RST$K_TYPE_ARRAY.
;  2793          2896  1 !
;  2794          2897  1
;  2795          2898  2     BEGIN
;  2796          2899  2     BUILTIN
;  2797          2900  2         PROBER;
;  2798          2901  2     MAP
;  2799          2902  2         DESC_PTR: REF BLOCK[,BYTE],
;  2800          2903  2         SYMID: REF RST$ENTRY;
;  2801          2904  2
;  2802          2905  2     LOCAL
;  2803          2906  2         CLASS,
;  2804          2907  2         DESC_ADDR: REF BLOCK[,BYTE],
;  2805          2908  2         DST_PTR: REF DST$RECORD,
;  2806          2909  2         VALPTR: VECTOR[3],
;  2807          2910  2         VKIND;
;  2808          2911  2
;  2809          2912  2
;  2810          2913  2
;  2811          2914  2     DST_PTR = .SYMID[RST$L_DSTPTR];
;  2812          2915  2     IF .DST_PTR[DST$V_VALKIND] NEQ DST$K_VALKIND_DESC
;  2813          2916  2     THEN
;  2814          2917  2         RETURN RST$K_TYPE_ATOMIC
;  2815          2918  2
;  2816          2919  2     ELSE
;  2817          2920  3         BEGIN
;  2818          2921  3         IF .DST_PTR[DST$B_VFLAGS] EQL DST$K_VFLAGS_DSC
;  2819          2922  3         THEN
;  2820          2923  3             DESC_ADDR = .DST_PTR[DST$L_DSC_OFFS] + DST_PTR[DST$A_DSC_BASE]
;  2821          2924  3
;  2822          2925  3         ELSE
;  2823          2926  4             BEGIN
;  2824          2927  4             DBG$STA_SYMVALUE(.SYMID,VALPTR,VKIND);
;  2825          2928  4             IF .VKIND NEQ DBG$K_VAL_DESCR
;  2826          2929  4             THEN
;  2827          2930  4                 $DBG_ERROR('RSTTYPES\DESC_OR_ARRAY_OR_ATOM');
;  2828          2931  4
;  2829          2932  4             DESC_ADDR = .VALPTR[0];
;  2830          2933  3             END;
;  2831          2934  3
```

```
;  2832          2935  3          IF NOT PROBER( %REF(0), %REF(12), .DESC_ADDR)
;  2833          2936  3          THEN
;  2834          2937  4              BEGIN
;  2835          2938  4              LOCAL
;  2836          2939  4                  NAME;
;  2837          2940  4              DBG$STA_SYMNAME(.SYMID, NAME);
;  2838          2941  4              SIGNAL(DBG$_BADDESCR, 1, .NAME);
;  2839          2942  3              END;
;  2840          2943  3          CLASS = .DESC_ADDR[DSC$B_CLASS];
;  2841          2944  3          .DESC_PTR = .DESC_ADDR;
;  2842          2945  3          IF (.CLASS EQL DSC$K_CLASS_A)   OR
;  2843          2946  3             (.CLASS EQL DSC$K_CLASS_NCA) OR
;  2844          2947  3             (.CLASS EQL DSC$K_CLASS_UBA) OR
;  2845          2948  4             (.CLASS EQL DSC$K_CLASS_VSA)
;  2846          2949  3          THEN
;  2847          2950  4              RETURN (RST$K_TYPE_ARRAY)
;  2848          2951  4
;  2849          2952  3          ELSE
;  2850          2953  3              RETURN (RST$K_TYPE_DESCR);
;  2851          2954  3
;  2852          2955  2          END;
;  2853          2956  2
;  2854          2957  1      END;


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

5F 43 53 45 44 5C 53 45 50 59 54 54 53 52 1E  001BB P.AAV:   .ASCII  <30>\RSTTYPES\<92>\DESC_OR_ARRAY_OR_AT\      ;
   54 41 5F 52 4F 5F 59 41 52 52 41 5F 52 4F  001CA                                                             ;
                                              001D8          .ASCII  \OM\                                        ;


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

                                              001C 00000 DESC_OR_ARRAY_OR_ATOM:
                                                                .WORD   Save R2,R3,R4                           ; 2878
                                 54 00000000G   00  9E 00002    MOVAB   LIB$SIGNAL, R4                         ;
                                 5E             14  C2 00009    SUBL2   #20, SP                                 ;
                                 53       04    AC  D0 0000C    MOVL    SYMID, R3                               ; 2914
                                 50       0C    A3  D0 00010    MOVL    12(R3), DST_PTR                         ;
          02       02   A0       02            00  ED 00014    CMPZV   #0, #2, 2(DST_PTR), #2                  ; 2915
                                                 04  13 0001A    BEQL    1$                                      ;
                                 50             02  D0 0001C    MOVL    #2, R0                                  ; 2920
                                                04  0001F       RET                                             ;
                           FA    8F       02    A0  91 00020 1$:  CMPB    2(DST_PTR), #250                        ; 2921
                                                 0A  12 00025    BNEQ    2$                                      ;
                                 50       03    A0  C0 00027    ADDL2   3(DST_PTR), R0                          ; 2923
                                 52       07    A0  9E 0002B    MOVAB   7(R0), DESC_ADDR                        ;
                                                 28  11 0002F    BRB     4$                                      ;
                                                 5E  DD 00031 2$:  PUSHL   SP                                    ; 2927
                                          0C     AE  9F 00033    PUSHAB  VALPTR                                  ;
                                                 53  DD 00036    PUSHL   R3                                      ;
                    00000000G    00              03  FB 00038    CALLS   #3, DBG$STA_SYMVALUE                    ;
                                 03              6E  D1 0003F    CMPL    VKIND, #3                               ; 2928
                                                 11  13 00042    BEQL    3$                                      ;
```

```
                          00000000'  EF  9F 00044          PUSHAB  P.AAV                           ; 2930
                                     01  DD 0004A          PUSHL   #1                              ;
                          00028362   8F  DD 0004C          PUSHL   #164706                         ;
                                64   03  FB 00052          CALLS   #3, LIB$SIGNAL                  ;
                                52   08  AE  D0 00055 3$:  MOVL    VALPTR, DESC_ADDR               ; 2932
                          62    0C       00  0C 00C59 4$:  PROBER  #0, #12, (DESC_ADDR)            ; 2935
                                         1A  12 0005D       BNEQ    5$                              ;
                                   04    AE  9F 0005F       PUSHAB  NAME                            ; 2940
                                         53  DD 00062       PUSHL   R3                              ;
                             00000000G   00  02  FB 00064   CALLS   #2, DBG$STA_SYMNAME             ;
                                   04    AE  DD 0006B       PUSHL   NAME                            ; 2941
                                         01  DD 0006E       PUSHL   #1                              ;
                          00028F80   8F  DD 00070          PUSHL   #167808                         ;
                                64   03  FB 00076          CALLS   #3, LIB$SIGNAL                  ;
                                50   03  A2  9A 00079 5$:  MOVZBL  3(DESC_ADDR), CLASS            ; 2943
                          08    BC       52  D0 0007D       MOVL    DESC_ADDR, @DESC_PTR            ; 2944
                                04       50  D1 00081       CMPL    CLASS, #4                       ; 2945
                                         0F  13 00084       BEQL    6$                              ;
                                0A       50  D1 00086       CMPL    CLASS, #10                      ; 2946
                                         0A  13 00089       BEQL    6$                              ;
                                0E       50  D1 0008B       CMPL    CLASS, #14                      ; 2947
                                         05  13 0008E       BEQL    6$                              ;
                                0C       50  D1 00090       CMPL    CLASS, #12                      ; 2948
                                         04  12 00093       BNEQ    7$                              ;
                                50       01  D0 00095 6$:  MOVL    #1, R0                          ; 2953
                                             04 00098       RET                                     ;
                                50       03  D0 00099 7$:  MOVL    #3, R0                          ;
                                             04 0009C       RET                                     ; 2957

; Routine Size:  157 bytes,    Routine Base:  DBG$CODE + 10BC

; 2855          2958  1
```

```
;2857     2959  1 ROUTINE FIND_TYPESPEC(TYPE_SPEC, BITSIZE, DST_PTR) =
;2858     2960  1 !
;2859     2961  1 !  FUNCTION
;2860     2962  1 !        This routine takes a starting TYPE_SPEC and traces through the DST
;2861     2963  1 !        to find the destination type specification; in this process, the
;2862     2964  1 !        routine is sensitive to DST$K_TS_IND, DST$K_SEPTYP, DST$K_GLOBNXT,
;2863     2965  1 !        and DST$K_TS_NOV_LENG. (All of these types indicate that the actual
;2864     2966  1 !        type specification is found elsewhere.) When a DST$K_TS_NOV_LENG is
;2865     2967  1 !        detected, BITSIZE is updated to reflect the novel length.
;2866     2968  1 !
;2867     2969  1 !        The routine loops until a true Type Spec is located.
;2868     2970  1 !
;2869     2971  1 !  INPUTS
;2870     2972  1 !        TYPE_SPEC - The DST address of the initial type-specification.
;2871     2973  1 !
;2872     2974  1 !        BITSIZE   - The address of a longword location to receive the size in
;2873     2975  1 !                    bits of the specified type.
;2874     2976  1 !
;2875     2977  1 !        DST_PTR   - The address of a longword location to receive the DST
;2876     2978  1 !                    address of the DST record which contains the desired
;2877     2979  1 !                    type specification.
;2878     2980  1 !
;2879     2981  1 !  OUTPUTS
;2880     2982  1 !        The routine's value is the DST address of the desired type-specification.
;2881     2983  1 !
;2882     2984  1 !        BITSIZE - The size in bits of an item of this type is returned to
;2883     2985  1 !                  BITSIZE. If a novel length record was detected, BITSIZE
;2884     2986  1 !                  reflects the novel length; otherwise, BITSIZE receives the
;2885     2987  1 !                  default size for the indicated data type.
;2886     2988  1 !
;2887     2989  1 !        DST_PTR - The address of the DST record which contains the desired
;2888     2990  1 !                  type-spec is returned to DST_PTR. The routine's value is
;2889     2991  1 !                  always of type DST$TYPE_SPEC; DST_PTR enables the calling
;2890     2992  1 !                  routine to know the start address of the containing DST record.
;2891     2993  1 !
;2892     2994  1
;2893     2995  2     BEGIN
;2894     2996  2
;2895     2997  2     LOCAL
;2896     2998  2         RECBEG_TRLR: REF DST$RECBEG_TRLR,
;2897     2999  2         SIZE,
;2898     3000  2         TEMP_TS: REF DST$TYPE_SPEC,
;2899     3001  2         TYPE_PTR: REF DST$RECORD;
;2900     3002  2
;2901     3003  2
;2902     3004  2     TEMP_TS = .TYPE_SPEC;
;2903     3005  2     SIZE = 0;
;2904     3006  2     TYPE_PTR = 0;
;2905     3007  2     WHILE TRUE DO
;2906     3008  3         BEGIN
;2907     3009  3         SELECTONE .TEMP_TS[DST$B_TS_KIND] OF
;2908     3010  3             SET
;2909     3011  3             [DST$K_TS_IND]:
;2910     3012  4                 BEGIN
;2911     3013  4                 TYPE_PTR = DBG$RST_DST_PTR( .DST$BEGIN_ADDR + .TEMP_TS[DST$L_TS_IND_PTR] );
;2912     3014  4                 IF .TYPE_PTR[DST$B_TYPE] EQL DST$K_SEPTYP
;2913     3015  4                 THEN
```

```
;  2914          3016  5                 BEGIN
;  2915          3017  5                 TYPE_PTR = DBG$RST_NEXT_DST( .TYPE_PTR );
;  2916          3018  5                 IF .TYPE_PTR[DST$B_TYPE] EQL DST$K_GLOBNXT
;  2917          3019  5                 THEN
;  2918          3020  5                     TYPE_PTR = DBG$RST_NEXT_DST( .TYPE_PTR );
;  2919          3021  5
;  2920          3022  4                 END;
;  2921          3023  4
;  2922          3024  4             IF (.TYPE_PTR[DST$B_TYPE] EQL DST$K_ENUMBEG) OR
;  2923          3025  5                (.TYPE_PTR[DST$B_TYPE] EQL DST$K_RECBEG)
;  2924          3026  4             THEN
;  2925          3027  4                 EXITLOOP;
;  2926          3028  4
;  2927          3029  4             TEMP_TS = TYPE_PTR[DST$A_TYPSPEC_TS_ADDR] +
;  2928          3030  4                       .TYPE_PTR[DST$B_TYPSPEC_NAME];
;  2929          3031  3             END;
;  2930          3032  3
;  2931          3033  3
;  2932          3034  3         ! Handle the kind of indirect pointer which points directly to
;  2933          3035  3         ! a DST Type Spec (as opposed to a Type Spec DST record).
;  2934          3036  3         !
;  2935          3037  3         [DST$K_TS_IND_TSPEC]:
;  2936          3038  3             TEMP_TS = DBG$RST_DST_PTR( .DST$BEGIN_ADDR + .TEMP_TS[DST$L_TS_IND_PTR] );
;  2937          3039  3
;  2938          3040  3
;  2939          3041  3         ! Handle the Novel Length Type Spec.  Here we pick up the length
;  2940          3042  3         ! of data objects of this type and indirect to the Type Spec.
;  2941          3043  3         !
;  2942          3044  3         [DST$K_TS_NOV_LENG]:
;  2943          3045  4             BEGIN
;  2944          3046  4
;  2945          3047  4
;  2946          3048  4         ! Make a note for the fact, we have encounted a novel
;  2947          3049  4         ! length dst.
;  2948          3050  4         !
;  2949          3051  4         NOVEL_LENGTH_FLAG = TRUE;
;  2950          3052  4         SIZE = .TEMP_TS[DST$L_TS_NOV_LENG];
;  2951          3053  4         TYPE_PTR = DBG$RST_DST_PTR( .DST$BEGIN_ADDR + .TEMP_TS[DST$L_TS_NOV_LENG_PAR_TSPEC] );
;  2952          3054  4         IF .TYPE_PTR[DST$B_TYPE] EQL DST$K_SEPTYP
;  2953          3055  4         THEN
;  2954          3056  5             BEGIN
;  2955          3057  5             TYPE_PTR = DBG$RST_NEXT_DST( .TYPE_PTR );
;  2956          3058  5             IF .TYPE_PTR[DST$B_TYPE] EQL DST$K_GLOBNXT
;  2957          3059  5             THEN
;  2958          3060  5                 TYPE_PTR = DBG$RST_NEXT_DST( .TYPE_PTR );
;  2959          3061  5
;  2960          3062  4             END;
;  2961          3063  4
;  2962          3064  4         IF (.TYPE_PTR[DST$B_TYPE] EQL DST$K_ENUMBEG) OR
;  2963          3065  5            (.TYPE_PTR[DST$B_TYPE] EQL DST$K_RECBEG)
;  2964          3066  4         THEN
;  2965          3067  4             EXITLOOP;
;  2966          3068  4
;  2967          3069  4         TEMP_TS = TYPE_PTR[DST$A_TYPSPEC_TS_ADDR] +
;  2968          3070  4                   .TYPE_PTR[DST$B_TYPSPEC_NAME];
;  2969          3071  3         END;
;  2970          3072  3
```

```
;  2971          3073  3
;  2972          3074  3              ! In all other cases, just exit the loop.
;  2973          3075  3              !
;  2974          3076  3              [OTHERWISE]:
;  2975          3077  3                  EXITLOOP;
;  2976          3078  3              TES;
;  2977          3079  3
;  2978          3080  2          END;
;  2979          3081  2
;  2980          3082  2      IF .TYPE_PTR NEQ 0
;  2981          3083  2      THEN
;  2982          3084  3          BEGIN
;  2983          3085  3          .DST_PTR = .TYPE_PTR;
;  2984          3086  3
;  2985          3087  3
;  2986          3088  3          ! Note TEMP_TS does not mean anything for TYPE_PTR[DST$B_TYPE] EQL
;  2987          3089  3          ! DST$K_ENUMBEG or DST$K_RECBEG.
;  2988          3090  3          !
;  2989          3091  3          IF .TYPE_PTR[DST$B_TYPE] EQL DST$K_RECBEG
;  2990          3092  3          THEN
;  2991          3093  4              BEGIN
;  2992          3094  4              RECBEG_TRLR = TYPE_PTR[DST$B_NAME] + 1 + .TYPE_PTR[DST$B_NAME];
;  2993          3095  4
;  2994          3096  4
;  2995          3097  4              ! Get size only if there is no novel length overrides.
;  2996          3098  4              !
;  2997          3099  4              IF .SIZE EQL 0 THEN SIZE = .RECBEG_TRLR[DST$L_RECBEG_SIZE];
;  2998          3100  4              .BITSIZE = .SIZE;
;  2999          3101  4              RETURN .TEMP_TS;
;  3000          3102  4              END
;  3001          3103  4
;  3002          3104  3          ELSE
;  3003          3105  4              BEGIN
;  3004          3106  4              IF .TYPE_PTR[DST$B_TYPE] EQL DST$K_ENUMBEG
;  3005          3107  4              THEN
;  3006          3108  5                  BEGIN
;  3007          3109  5
;  3008          3110  5
;  3009          3111  5                  ! Get size only if there is no novel length overrides.
;  3010          3112  5                  !
;  3011          3113  5                  IF .SIZE EQL 0 THEN SIZE = .TYPE_PTR[DST$B_ENUMBEG_LENG];
;  3012          3114  5                  .BITSIZE = .SIZE;
;  3013          3115  5                  RETURN .TEMP_TS;
;  3014          3116  4                  END;
;  3015          3117  4
;  3016          3118  3              END;
;  3017          3119  3
;  3018          3120  2          END;
;  3019          3121  2
;  3020          3122  2      IF .SIZE EQL 0 THEN SIZE = DBG$GET_BITSIZE_FROM_TYPESPEC(.TEMP_TS);
;  3021          3123  2      .BITSIZE = .SIZE;
;  3022          3124  2      RETURN .TEMP_TS;
;  3023          3125  2
;  3024          3126  1      END;
```

```
                                00FC 00000 FIND_TYPESPEC:
                                           .WORD   Save R2,R3,R4,R5,R6,R7                          ; 2959
                          57 00000000G  00  9E 00002        MOVAB   DST$BEGIN_ADDR, R7
                          56 00000000G  00  9E 00009        MOVAB   DBG$RST_NEXT_DST, R6
                          55 00000000G  00  9E 00010        MOVAB   DBG$RST_DST_PTR, R5
                          53       04   AC  D0 00017        MOVL    TYPE_SPEC, TEMP_TS             ; 3004
                                        54  D4 0001B        CLRL    SIZE                           ; 3005
                                        52  D4 0001D        CLRL    TYPE_PTR                       ; 3006
                          50       02   A3  9A 0001F 1$:    MOVZBL  2(TEMP_TS), R0                 ; 3009
                          03            50  91 00023        CMPB    R0, #3                         ; 3011
                                        23  12 00026        BNEQ    2$
                 7E       67       03   A3  C1 00028        ADDL3   3(TEMP_TS), DST$BEGIN_ADDR, -(SP) ; 3013
                          65            01  FB 0002D        CALLS   #1, DBG$RST_DST_PTR
                          52            50  D0 00030        MOVL    R0, TYPE_PTR
                    A3    8F       01   A2  91 00033        CMPB    1(TYPE_PTR), #163              ; 3014
                                        5C  12 00038        BNEQ    6$
                                        52  DD 0003A        PUSHL   TYPE_PTR                       ; 3017
                          66            01  FB 0003C        CALLS   #1, DBG$RST_NEXT_DST
                          52            50  D0 0003F        MOVL    R0, TYPE_PTR
                    A0    8F       01   A2  91 00042        CMPB    1(TYPE_PTR), #160              ; 3018
                                        45  13 00047        BEQL    5$
                                        48  11 00049        BRB     6$                             ; 3024
                          0F            50  91 0004B 2$:    CMPB    R0, #15                        ; 303'
                                        0D  12 0004E        BNEQ    4$
                 7E       67       03   A3  C1 00050        ADDL3   3(TEMP_TS), DST$BEGIN_ADDR, -(SP) ; 3038
                          65            01  FB 00055        CALLS   #1, DBG$RST_DST_PTR
                          53            50  D0 00058        MOVL    R0, TEMP_TS
                                        C2  11 0005B 3$:    BRB     1$
                          0E            50  91 0005D 4$:    CMPB    R0, #14                        ; 3044
                                        4D  12 00060        BNEQ    7$
              00000000'   EF            01  D0 00062        MOVL    #1, NOVEL_LENGTH_FLAG          ; 3051
                          54       03   A3  D0 00069        MOVL    3(TEMP_TS), SIZE               ; 3052
                 7E       67       07   A3  C1 0006D        ADDL3   7(TEMP_TS), DST$BEGIN_ADDR, -(SP) ; 3053
                          65            01  FB 00072        CALLS   #1, DBG$RST_DST_PTR
                          52            50  D0 00075        MOVL    R0, TYPE_PTR
                    A3    8F       01   A2  91 00078        CMPB    1(TYPE_PTR), #163              ; 3054
                                        17  12 0007D        BNEQ    6$
                                        52  DD 0007F        PUSHL   TYPE_PTR                       ; 3057
                          66            01  FB 00081        CALLS   #1, DBG$RST_NEXT_DST
                          52            50  D0 00084        MOVL    R0, TYPE_PTR
                    A0    8F       01   A2  91 00087        CMPB    1(TYPE_PTR), #160              ; 3058
                                        08  12 0008C        BNEQ    6$
                                        52  DD 0008E 5$:    PUSHL   TYPE_PTR                       ; 3060
                          66            01  FB 00090        CALLS   #1, DBG$RST_NEXT_DST
                          52            50  D0 00093        MOVL    R0, TYPE_PTR
                    A5    8F       01   A2  91 00096 6$:    CMPB    1(TYPE_PTR), #165              ; 3064
                                        12  13 0009B        BEQL    7$
                    AB    8F       01   A2  91 0009D        CMPB    1(TYPE_PTR), #171              ; 3065
                                        0B  13 000A2        BEQL    7$
                          50       02   A2  9A 000A4        MOVZBL  2(TYPE_PTR), R0                ; 3070
                          53       03 A042  9E 000A8        MOVAB   3(R0)[TYPE_PTR], TEMP_TS       ; 3069
                                        AC  11 000AD        BRB     3$                             ; 3009
                                        52  D5 000AF 7$:    TSTL    TYPE_PTR                       ; 3082
                                        2E  13 000B1        BEQL    9$
                    0C  BC              52  D0 000B3        MOVL    TYPE_PTR, @DST_PTR             ; 3085
```

```
                AB   8F        01    A2  91 000B7          CMPB    1(TYPE_PTR), #171                       ; 3091
                                    12  12 000BC          BNEQ    8$                                      ;
                     50        07    A2  9A 000BE          MOVZBL  7(TYPE_PTR), R0                         ; 3094
                     50        08 A042  9E 000C2          MOVAB   8(R0)[TYPE_PTR], RECBEG_TRLR            ;
                                    54  D5 000C7          TSTL    SIZE                                    ; 3099
                                    24  12 000C9          BNEQ    10$                                     ;
                     54             60  D0 000CB          MOVL    (RECBEG_TRLR), SIZE                     ;
                                    1F  11 000CE          BRB     10$                                     ; 3100
                A5   8F        01    A2  91 000D0 8$:      CMPB    1(TYPE_PTR), #165                       ; 3106
                                    0A  12 000D5          BNEQ    9$                                      ;
                                    54  D5 000D7          TSTL    SIZE                                    ; 3113
                                    14  12 000D9          BNEQ    10$                                     ;
                     54        02    A2  9A 000DB          MOVZBL  2(TYPE_PTR), SIZE                       ;
                                    0E  11 000DF          BRB     10$                                     ; 3114
                                    54  D5 000E1 9$:      TSTL    SIZE                                    ; 3122
                                    0A  12 000E3          BNEQ    10$                                     ;
                                    53  DD 000E5          PUSHL   TEMP_TS                                 ;
              0000V CF              01  FB 000E7          CALLS   #1, DBG$GET_BITSIZE_FROM_TYPESPEC       ;
                     54             50  D0 000EC          MOVL    R0, SIZE                                ;
                08   BC             54  D0 000EF 10$:     MOVL    SIZE, @BITSIZE                          ; 3123
                     50             53  D0 000F3          MOVL    TEMP_TS, R0                             ; 3124
                                        04 000F6          RET                                             ; 3126

; Routine Size:  247 bytes,    Routine Base:  DBG$CODE + 1159
```

```
;  3026          3127  1 ROUTINE FIND_TYPREC_FROM_TSPEC( TYPE_SPEC ) =
;  3027          3128  1 !
;  3028          3129  1 ! FUNCTION
;  3029          3130  1 !       This routine takes a TYPE SPECIFICATION and returns the (possibly)
;  3030          3131  1 !       constructed TYPEID which describes the type.
;  3031          3132  1 !
;  3032          3133  1 ! INPUTS
;  3033          3134  1 !       TYPE_SPEC - The type specification for which we need a TYPEID.
;  3034          3135  1 !
;  3035          3136  1 ! OUTPUTS
;  3036          3137  1 !       The address of a TYPEID describing type_spec.
;  3037          3138  1 !
;  3038          3139  1
;  3039          3140  2 BEGIN
;  3040          3141  2
;  3041          3142  2     LOCAL
;  3042          3143  2         FCODE,
;  3043          3144  2         SIZE,
;  3044          3145  2         TEMP_TS: REF DST$TYPE_SPEC,
;  3045          3146  2         TYPEID: REF RST$ENTRY,
;  3046          3147  2         TYPE_PTR: REF DST$RECORD;
;  3047          3148  2
;  3048          3149  2
;  3049          3150  2     SIZE = 0;
;  3050          3151  2     TYPE_PTR = 0;
;  3051          3152  2     TEMP_TS = FIND_TYPESPEC(.TYPE_SPEC, SIZE, TYPE_PTR);
;  3052          3153  2     IF .TYPE_PTR NEQ 0
;  3053          3154  2     THEN
;  3054          3155  3         BEGIN
;  3055          3156  3         IF (.TYPE_PTR[DST$B_TYPE] EQL DST$K_ENUMBEG) OR
;  3056          3157  4            (.TYPE_PTR[DST$B_TYPE] EQL DST$K_RECBEG)
;  3057          3158  3         THEN
;  3058          3159  4             BEGIN
;  3059          3160  4             TYPEID = FIND_RST_TYPE_ENTRY(.TYPE_PTR);
;  3060          3161  4             IF .TYPEID NEQ 0 THEN RETURN .TYPEID;
;  3061          3162  3             END;
;  3062          3163  2         END;
;  3063          3164  2
;  3064          3165  2     ! If we have a descriptor type, then we need to check
;  3065          3166  2     ! whether this is an array descriptor.
;  3066          3167  2     !
;  3067          3168  3     IF  (.TEMP_TS[DST$B_TS_KIND] EQL DST$K_TS_DSC)
;  3068          3169  2     THEN
;  3069          3170  3         BEGIN
;  3070          3171  3         LOCAL
;  3071          3172  3             CLASS,
;  3072          3173  3             DESC_PTR: REF BLOCK[,BYTE],
;  3073          3174  3             VALKIND,
;  3074          3175  3             VAL_PTR,
;  3075          3176  3             VAL_VECT: VECTOR[3];
;  3076          3177  3
;  3077          3178  3         VAL_PTR = TEMP_TS[DST$A_TS_DSC_VSPEC_ADDR];
;  3078          3179  3         DBG$STA_VALSPEC(.VAL_PTR, VAL_VECT, VALKIND);
;  3079          3180  3         IF .VALKIND NEQ DBG$K_VAL_DESCR
;  3080          3181  3         THEN
;  3081          3182  3             $DBG_ERROR('RSTTYPES\FIND_TYPREC_FROM_TSPEC 20');
;  3082          3183  3
```

```
; 3083          3184  3             DESC_PTR = .VAL_VECT[0];
; 3084          3185  3             DBG$READ_ACCESS(.DESC_PTR, 12);
; 3085          3186  3             CLASS = .DESC_PTR[DSC$B_CLASS];
; 3086          3187  3             IF (.CLASS EQL DSC$K_CLASS_A)    OR
; 3087          3188  3                (.CLASS EQL DSC$K_CLASS_NCA) OR
; 3088          3189  3                (.CLASS EQL DSC$K_CLASS_UBA) OR
; 3089          3190  4                (.CLASS EQL DSC$K_CLASS_VSA)
; 3090          3191  3             THEN
; 3091          3192  4                 BEGIN
; 3092          3193  4                 FCODE = RST$K_TYPE_ARRAY;
; 3093          3194  4                 RETURN TYPEID_FOR_DESCR(.DESC_PTR, .FCODE);
; 3094          3195  3                 END;
; 3095          3196  2             END;
; 3096          3197  2
; 3097          3198  2         FCODE = DBG$TRANS_TYPE_CODE(.TEMP_TS[DST$B_TS_KIND]);
; 3098          3199  2         IF .FCODE EQL RST$K_TYPE_ATOMIC
; 3099          3200  2         THEN
; 3100          3201  2             RETURN DBG$TYPEID_FOR_ATOMIC(.TEMP_TS[DST$B_TS_ATOM_TYP],.SIZE, FALSE);
; 3101          3202  2
; 3102          3203  2         IF .TYPE_PTR NEQ 0
; 3103          3204  2         THEN
; 3104          3205  2             RETURN TYPEID_FROM_DST_RECORD(.TYPE_PTR, .FCODE, .SIZE);
; 3105          3206  2
; 3106          3207  2         RETURN TYPEID_FROM_DST_TYPESPEC(.TEMP_TS,.FCODE,.SIZE);
; 3107          3208  2
; 3108          3209  1         END;


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

5F 44 4E 49 46 5C 53 45 50 59 54 54 53 52 22 001DA P.AAW:  .ASCII  \"RSTTYPES\<92>\FIND_TYPREC_FROM_TSPEC 2\   ;
50 53 54 5F 4D 4F 52 46 5F 43 45 52 50 59 54 001E9                                                           ;
                                    32 20 43 45 001F8                                                           ;
                                                30 001FC         .ASCII  \0\                                    ;


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

                                      007C 00000 FIND_TYPREC_FROM_TSPEC:
                                                         .WORD   Save R2,R3,R4,R5,R6                    ; 3127
                      5E          10  C2 00002           SUBL2   #16, SP                                 ;
                                  7E  7C 00005           CLRQ    TYPE_PTR                                ; 3151
                                  5E  DD 00007           PUSHL   SP                                      ; 3152
                             08   AE  9F 00009           PUSHAB  SIZE                                    ;
                             04   AC  DD 0000C           PUSHL   TYPE_SPEC                               ;
                 FEF5   CF        03  FB 0000F           CALLS   #3, FIND_TYPESPEC                       ;
                        54        50  D0 00014           MOVL    R0, TEMP_TS                             ;
                        55        6E  D0 00017           MOVL    TYPE_PTR, R5                            ; 3153
                                  56  D4 0001A           CLRL    R6                                      ;
                                  55  D5 0001C           TSTL    R5                                      ;
                                  1C  13 0001E           BEQL    2$                                      ;
                                  56  D6 00020           INCL    R6                                      ;
                   A5   8F   01   A5  91 00022           CMPB    1(R5), #165                             ; 3156
                                  07  13 00027           BEQL    1$                                      ;
                   AB   8F   01   A5  91 00029           CMPB    1(R5), #171                             ; 3157
```

```
                                 0C  12 0002E          BNEQ    2$                                      :
                                 55  DD 00030 1$:      PUSHL   R5                                      : 3160
          0000V  CF              01  FB 00032          CALLS   #1, FIND_RST_TYPE_ENTRY                 :
                                 50  D5 00037          TSTL    TYPEID                                  : 3161
                                 01  13 00039          BEQL    2$                                      :
                                     04 0003B          RET                                             :
                02        02     A4  91 0003C 2$:      CMPB    2(TEMP_TS), #2                          : 3168
                                 60  12 00040          BNEQ    5$                                      :
                50        03     A4  9E 00042          MOVAB   3(R4), VAL_PTR                          : 3178
                             08  AE  9F 00046          PUSHAB  VALKIND                                 : 3179
                             10  AE  9F 00049          PUSHAB  VAL_VECT                                :
                                 50  DD 0004C          PUSHL   VAL_PTR                                 :
    00000000G   00               03  FB 0004E          CALLS   #3, DBG$STA_VALSPEC                     :
                03           08  AE  D1 00055          CMPL    VALKIND, #3                             : 3180
                                 15  13 00059          BEQL    3$                                      :
                    00000000'    EF  9F 0005B          PUSHAB  P.AAW                                   : 3182
                                 01  DD 00061          PUSHL   #1                                      :
                    00028362     8F  DD 00063          PUSHL   #164706                                 :
    00000000G   00               03  FB 00069          CALLS   #3, LIB$SIGNAL                          :
                52           0C  AE  D0 00070 3$:      MOVL    VAL_VECT, DESC_PTR                      : 3184
                             0C  DD 00074          PUSHL   #12                                     : 3185
                                 52  DD 00076          PUSHL   DESC_PTR                                :
    00000000G   00               02  FB 00078          CALLS   #2, DBG$READ_ACCESS                     :
                50           03  A2  9A 0007F          MOVZBL  3(DESC_PTR), CLASS                      : 3186
                04               50  D1 00083          CMPL    CLASS, #4                               : 3187
                                 0F  13 00086          BEQL    4$                                      :
                0A               50  D1 00088          CMPL    CLASS, #10                              : 3188
                                 0A  13 0008B          BEQL    4$                                      :
                0E               50  D1 0008D          CMPL    CLASS, #14                              : 3189
                                 05  13 00090          BEQL    4$                                      :
                0C               50  D1 00092          CMPL    CLASS, #12                              : 3190
                                 0B  12 00095          BNEQ    5$                                      :
                53               01  D0 00097 4$:      MOVL    #1, FCODE                               : 3193
                                 0C  BB 0009A          PUSHR   #^M<R2,R3>                              : 3194
          F9DA  CF               02  FB 0009C          CALLS   #2, TYPEID_FOR_DESCR                    :
                                     04 000A1          RET                                             :
                7E        02     A4  9A 000A2 5$:      MOVZBL  2(TEMP_TS), -(SP)                       : 3198
          F5F1  CF               01  FB 000A6          CALLS   #1, DBG$TRANS_TYPE_CODE                 :
                53               50  D0 000AB          MOVL    R0, FCODE                               :
                02               53  D1 000AE          CMPL    FCODE, #2                               : 3199
                                 0F  12 000B1          BNEQ    6$                                      :
                                 7E  D4 000B3          CLRL    -(SP)                                   : 3201
                             08  AE  DD 000B5          PUSHL   SIZE                                    :
                7E           03  A4  9A 000B8          MOVZBL  3(TEMP_TS), -(SP)                       :
          F713  CF               03  FB 000BC          CALLS   #3, DBG$TYPEID_FOR_ATOMIC               :
                                     04 000C1          RET                                             :
                0D               56  E9 000C2 6$:      BLBC    R6, 7$                                  : 3203
                             04  AE  DD 000C5          PUSHL   SIZE                                    : 3204
                                 53  DD 000C8          PUSHL   FCODE                                   :
                                 55  DD 000CA          PUSHL   R5                                      :
          FADO  CF               03  FB 000CC          CALLS   #3, TYPEID_FROM_DST_RECORD              :
                                     04 000D1          RET                                             :
                             04  AE  DD 000D2 7$:      PUSHL   SIZE                                    : 3207
                                 53  DD 000D5          PUSHL   FCODE                                   :
                                 54  DD 000D7          PUSHL   TEMP_TS                                 :
          FA5F  CF               03  FB 000D9          CALLS   #3, TYPEID_FROM_DST_TYPESPEC            :
                                     04 000DE          RET                                             : 3209
```

; Routine Size: 223 bytes,   Routine Base: DBG$CODE + 1250

```
;  3110          3210  1 GLOBAL ROUTINE DBG$GET_BITSIZE_FROM_DESC(DESC_PTR): =
;  3111          3211  1 !
;  3112          3212  1 ! FUNCTION
;  3113          3213  1 !       Given pointer to VAX Standard Descriptor, get its size.
;  3114          3214  1 !
;  3115          3215  1 ! INPUT
;  3116          3216  1 !       DESC_PTR - Pointer to VAX Standard Descriptor.
;  3117          3217  1 !
;  3118          3218  1 ! OUTPUT
;  3119          3219  1 !       Size in bits is returned, it can be zero for non-contiguous array.
;  3120          3220  1 !
;  3121          3221  1
;  3122          3222  2     BEGIN
;  3123          3223  2
;  3124          3224  2     MAP
;  3125          3225  2         DESC_PTR: REF BLOCK[,BYTE];
;  3126          3226  2
;  3127          3227  2     LOCAL
;  3128          3228  2         SIZE;
;  3129          3229  2
;  3130          3230  2
;  3131          3231  2     ! Check for read access.
;  3132          3232  2     !
;  3133          3233  2     DBG$READ_ACCESS(.DESC_PTR, 12);
;  3134          3234  2
;  3135          3235  2     CASE .DESC_PTR[DSC$B_CLASS] FROM DSC$K_CLASS_S TO DSC$K_CLASS_UBA OF
;  3136          3236  2         SET
;  3137          3237  2
;  3138          3238  2         ! Scalar types, scaled decimal types, and unaligned bit strings.
;  3139          3239  2         ! The DBG$DATA_LENGTH routine computes a bit length by looking
;  3140          3240  2         ! at the DSC$W_LENGTH field, and then looking at the dtype field
;  3141          3241  2         ! to figure out whether the length is in bits, nibbles, or bytes.
;  3142          3242  2         !
;  3143          3243  2         [DSC$K_CLASS_S, DSC$K_CLASS_D, DSC$K_CLASS_SD, DSC$K_CLASS_UBS]:
;  3144          3244  3             BEGIN
;  3145          3245  3             SIZE = DBG$DATA_LENGTH(.DESC_PTR);
;  3146          3246  2             END;
;  3147          3247  2
;  3148          3248  2         ! Arrays. The length is given in the ARSIZE field. It is either
;  3149          3249  2         ! in bytes or nibbles, depending on the dtype.
;  3150          3250  2         !
;  3151          3251  2         [DSC$K_CLASS_A]:
;  3152          3252  3             BEGIN
;  3153          3253  3             IF .DESC_PTR[DSC$B_DTYPE] EQL DSC$K_DTYPE_P
;  3154          3254  3             THEN
;  3155          3255  3                 SIZE = .DESC_PTR[DSC$L_ARSIZE] * 4
;  3156          3256  3             ELSE
;  3157          3257  3                 SIZE = .DESC_PTR[DSC$L_ARSIZE] * 8;
;  3158          3258  2             END;
;  3159          3259  2
;  3160          3260  2         ! Non-contiguous arrays. The ARSIZE field contains a length,
;  3161          3261  2         ! but that length may not be meaningful. Return 0 since we
;  3162          3262  2         ! don't really know what the length is.
;  3163          3263  2         !
;  3164          3264  2         [DSC$K_CLASS_NCA, DSC$K_CLASS_UBA, DSC$K_CLASS_VSA]:
;  3165          3265  3             BEGIN
;  3166          3266  3             SIZE = 0;
```

```
;  3167          3267  2              END;
;  3168          3268  2
;  3169          3269  2          ! Varying string descriptor. The maximum string length in
;  3170          3270  2          ! bytes is in the MAXSTRLEN field.
;  3171          3271  2          !
;  3172          3272  2          [DSC$K_CLASS_VS]:
;  3173          3273  3              BEGIN
;  3174          3274  3              SIZE = .DESC_PTR[DSC$W_MAXSTRLEN] * 8;
;  3175          3275  2              END;
;  3176          3276  2
;  3177          3277  2          [INRANGE, OUTRANGE]:
;  3178          3278  2              SIGNAL(DBG$_DESCNOTSET);
;  3179          3279  2
;  3180          3280  2          TES;
;  3181          3281  2
;  3182          3282  2      RETURN .SIZE;
;  3183          3283  1      END;


                                                    0004 00000          .ENTRY  DBG$GET_BITSIZE_FROM_DESC, Save R2          ; 3210
                                                 0C DD 00002            PUSHL   #12                                         ; 3233
                                      52      04 AC DO 00004            MOVL    DESC_PTR, R2                                ;
                                                 52 DD 00008            PUSHL   R2                                          ;
                          00000000G   00         02 FB 0000A            CALLS   #2, DBG$READ_ACCESS                         ;
                                0D            01      03 A2 8F 00011    CASEB   3(R2), #1, #T3                              ; 3235
                0034          001C              002A       002A 00016 1$: .WORD   3$-1$,-                                     ;
                001C          001C              001C       001C 0001E     3$-1$,-                                     ;
                0046          0049              0046       002A 00026     2$-1$,-                                     ;
                                                0046       002A 0002E     4$-1$,-                                     ;
                                                                          2$-1$,-                                     ;
                                                                          2$-1$,-                                     ;
                                                                          2$-1$,-                                     ;
                                                                          2$-1$,-                                     ;
                                                                          3$-1$,-                                     ;
                                                                          6$-1$,-                                     ;
                                                                          7$-1$,-                                     ;
                                                                          6$-1$,-                                     ;
                                                                          3$-1$,-                                     ;
                                                                          6$-1$                                       ;
                                                00028F 50  8F DD 00032 2$: PUSHL   #167760                             ; 3278
                          00000000G   00            01 FB 00038         CALLS   #1, LIB$SIGNAL                          ;
                                                    04 0003F            RET                                             ;
                                                    52 DD 00040 3$:     PUSHL   R2                                      ; 3245
                          00000000G   00            01 FB 00042         CALLS   #1, DBG$DATA_LENGTH                     ;
                                                    04 00049            RET                                             ; 3235
                                      15      02    A2 91 0004A 4$:     CMPB    2(R2), #21                              ; 3253
                                                    06 12 0004E         BNEQ    5$                                      ;
                          50          0C    A2      02 78 00050         ASHL    #2, 12(R2), SIZE                        ; 3255
                                                    04 00055            RET                                             ;
                          50          0C    A2      03 78 00056 5$:     ASHL    #3, 12(R2), SIZE                        ; 3257
                                                    04 0005B            RET                                             ; 3235
                                                    50 D4 0005C 6$:     CLRL    SIZE                                    ; 3266
                                                    04 0005E            RET                                             ; 3235
                                      50            62 3C 0005F 7$:     MOVZWL  (R2), SIZE                              ; 3274
```

```
                50        08  C4 00062        MULL2   #8, SIZE                ;
                              04 00065        RET                             ; 3283

; Routine Size:  102 bytes,    Routine Base:  DBG$CODE + 132F
```

```
;  3185          3284  1 ROUTINE GET_BITSIZE_FROM_DTYPE(DTYPE) =
;  3186          3285  1 !
;  3187          3286  1 ! FUNCTION
;  3188          3287  1 !       This routine takes an SRM DTYPE and returns the defau.t bitsize
;  3189          3288  1 !       defined for that type.
;  3190          3289  1 !
;  3191          3290  1 ! INPUTS
;  3192          3291  1 !       DTYPE   - The VAX Standard Data Type whose bit size we want.
;  3193          3292  1 !
;  3194          3293  1 ! OUTPUTS
;  3195          3294  1 !       The corresponding bit size is returned as the routine's value.
;  3196          3295  1 !
;  3197          3296  1
;  3198          3297  2     BEGIN
;  3199          3298  2
;  3200          3299  2     BIND
;  3201          3300  2         LENGTH_TBL = UPLIT BYTE(0, 1, 1,2,4,8, 1,2,4,8, 4,8,8,16,
;  3202          3301  2                                REP 11 OF (0), 16,16,8,16,16,32,
;  3203          3302  2                                12,8,8,0,0,0,0): VECTOR[,BYTE];
;  3204          3303  2
;  3205        L 3304  2         %IF DSC$K_DTYPE_HIGHEST GTR 37  ! If the LENGTH_TBL PLIT is too small,
;  3206        U 3305  2         %THEN                           !     generate a compile-time error
;  3207        U 3306  2             %ERROR('Error:  Must expand PLIT in Routine GET_BITSIZE_FROM_DTYPE')
;  3208          3307  2         %FI;
;  3209          3308  2
;  3210          3309  2
;  3211          3310  2
;  3212          3311  2     ! Return the bitsize corresponding to the input data type.  Note that
;  3213          3312  2     ! we treat Boolean (as in PASCAL) as a special case.
;  3214          3313  2     !
;  3215          3314  2     IF .DTYPE EQL DST$K_BOOL OR .DTYPE EQL DSC$K_DTYPE_TF THEN RETURN 8;
;  3216          3315  2     RETURN 8*.LENGTH_TBL[.DTYPE];
;  3217          3316  2
;  3218          3317  1     END;

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

    10  08  08  04  08  04  02  01  08  04  02  01  01  00  001FD P.AAX:  .BYTE   0, 1, 1, 2, 4, 8, 1, 2, 4, 8, 4, 8, 8, 16   ;
                                                        00# 0020B       .BYTE   0[11]                                       ;
        00  00  00  00  08  08  0C  20  10  10  08  10  10  00216       .BYTE   16, 16, 8, 16, 16, 32, 12, 8, 8, 0, 0, 0, - ;
                                                                        0                                                   ;

                                                LENGTH_TBL=         P.AAX

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

                                                    0000 00000 GET_BITSIZE_FROM_DTYPE:
                                                               .WORD    Save nothing                                    ; 3284
                                0000009E    8F      04    AC  D1 00002               CMPL    DTYPE, #158                 ; 3314
                                                          06  13 0000A               BEQL    1$                          ;
                                            28      04    AC  D1 0000C               CMPL    DTYPE, #40                  ;
                                                          04  12 00010               BNEQ    2$                          ;
                                            50            08  D0 00012 1$:           MOVL    #8, R0                      ;
                                                              04 00015               RET                                 ;
```

```
                50 00000000'  EF  9E 00016 2$:     MOVAB   LENGTH_TBL, R0                  ; 3315
                50        04 BC40  9A 0001D        MOVZBL  @DTYPE[R0], R0                  ;
                50           08  C4 00022        MULL2   #8, R0                          ;
                                 04 00025        RET                                     ; 3317

; Routine Size:  38 bytes,    Routine Base:  DBG$CODE + 1395
```

```
;  3220          3318  1 GLOBAL ROUTINE DBG$GET_BITSIZE_FROM_TYPESPEC( TYPE_SPEC ) =
;  3221          3319  1 !
;  3222          3320  1 ! FUNCTION
;  3223          3321  1 !       This routine takes a type specification and returns the corresponding
;  3224          3322  1 !       default bit size.
;  3225          3323  1 !
;  3226          3324  1 ! INPUTS
;  3227          3325  1 !       type_spec - The type specification whose default size we want.
;  3228          3326  1 !
;  3229          3327  1 ! OUTPUTS
;  3230          3328  1 !       The type's default bit size is returned as the routine's value.
;  3231          3329  1 !
;  3232          3330  1
;  3233          3331  2     BEGIN
;  3234          3332  2
;  3235          3333  2     MAP
;  3236          3334  2         TYPE_SPEC: REF DST$TYPE_SPEC;
;  3237          3335  2
;  3238          3336  2     LOCAL
;  3239          3337  2         DESCPTR: REF BLOCK[,BYTE],
;  3240          3338  2         FLAGS_PTR: REF BITVECTOR,
;  3241          3339  2         NDIMS,
;  3242          3340  2         SIZE,
;  3243          3341  2         VAL_VECT: VECTOR[3],
;  3244          3342  2         VALKIND,
;  3245          3343  2         VALPTR: REF DST$VAL_SPEC;
;  3246          3344  2
;  3247          3345  2
;  3248          3346  2     SELECTONE .TYPE_SPEC[DST$B_TS_KIND] OF
;  3249          3347  2         SET
;  3250          3348  2
;  3251          3349  2         [DST$K_TS_ATOM]:
;  3252          3350  2             SIZE = GET_BITSIZE_FROM_DTYPE(.TYPE_SPEC[DST$B_TS_ATOM_TYP]);
;  3253          3351  2
;  3254          3352  2         [DST$K_TS_DSC]:
;  3255          3353  3             BEGIN
;  3256          3354  3             VALPTR = TYPE_SPEC[DST$A_TS_DSC_VSPEC_ADDR];
;  3257          3355  3             DBG$STA_VALSPEC(.VALPTR, VAL_VECT, VALKIND);
;  3258          3356  3             IF .VALKIND NEQ DBG$K_VAL_DESCR
;  3259          3357  3             THEN
;  3260          3358  3                 $DBG_ERROR('RSTTYPES\DBG$GET_BITSIZE_FROM_TYPESPEC');
;  3261          3359  3
;  3262          3360  3             DESCPTR = .VAL_VECT[0];
;  3263          3361  3             SIZE = DBG$GET_BITSIZE_FROM_DESC(.DESCPTR);
;  3264          3362  2             END;
;  3265          3363  2
;  3266          3364  2         [DST$K_TS_TPTR, DST$K_TS_PTR, DST$K_TS_FILE]:
;  3267          3365  2             SIZE = 32;
;  3268          3366  2
;  3269          3367  2         [DST$K_TS_PIC]:
;  3270          3368  2             SIZE = .TYPE_SPEC[DST$B_TS_PIC_DLENG]*%BPUNIT;
;  3271          3369  2
;  3272          3370  2         [DST$K_TS_ARRAY]:
;  3273          3371  3             BEGIN
;  3274          3372  3             NDIMS = .TYPE_SPEC[DST$B_TS_ARRAY_DIM];
;  3275          3373  3             FLAGS_PTR = TYPE_SPEC[DST$A_TS_ARRAY_FLAGS_ADDR];
;  3276          3374  3             VALPTR = .FLAGS_PTR + (.NDIMS/8) + 1;
```

```
;  3277     3375  3              DBG$STA_VALSPEC(.VALPTR, VAL_VECT, VALKIND);
;  3278     3376  3              IF .VALKIND NEQ DBG$K_VAL_DESCR
;  3279     3377  3              THEN
;  3280     3378  3                  $DBG_ERROR('RSTTYPES\DBG$GET_BITSIZE_FROM_TYPESPEC');
;  3281     3379  3
;  3282     3380  3              DESCPTR = .VAL_VECT[0];
;  3283     3381  3              SIZE = DBG$GET_BITSIZE_FROM_DESC(.DESCPTR);
;  3284     3382  2              END;
;  3285     3383  2
;  3286     3384  2          [DST$K_TS_SET]:
;  3287     3385  2              SIZE = .TYPE_SPEC[DST$L_TS_SET_LENG];
;  3288     3386  2
;  3289     3387  2          [DST$K_TS_SUBRANGE]:
;  3290     3388  2              SIZE = .TYPE_SPEC[DST$L_TS_SUBR_LENG];
;  3291     3389  2
;  3292     3390  2          [DST$K_TS_NOV_LENG]:
;  3293     3391  2              SIZE = .TYPE_SPEC[DST$L_TS_NOV_LENG];
;  3294     3392  2
;  3295     3393  2          [DST$K_TS_SELF_REL_LABEL]:
;  3296     3394  2              SIZE = .TYPE_SPEC[DST$L_TS_SELF_LENG] * 8;
;  3297     3395  2
;  3298     3396  2          [DST$K_TS_RFA]:
;  3299     3397  2              SIZE = 6 * 8;
;  3300     3398  2
;  3301     3399  2          [DST$K_TS_TASK]:
;  3302     3400  2              SIZE = 32;
;  3303     3401  2
;  3304     3402  2          [OTHERWISE]:
;  3305     3403  2              SIZE = 0;
;  3306     3404  2
;  3307     3405  2          TES;
;  3308     3406  2
;  3309     3407  2      RETURN .SIZE;
;  3310     3408  2
;  3311     3409  1      END;


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

47 24 47 42 44 5C 53 45 50 59 54 54 53 52 26  00223 P.AAY:   .ASCII  \&RSTTYPES\<92>\DBG$GET_BITSIZE_FROM_TYP\   ;
4D 4F 52 46 5F 45 5A 49 53 54 49 42 5F 54 45  00232                                                           ;
                              50 59 54 5F     00241                                                           ;
                           43 45 50 53 45     00245           .ASCII  \ESPEC\                                 ;
47 24 47 42 44 5C 53 45 50 59 54 54 53 52 26  0024A P.AAZ:   .ASCII  \&RSTTYPES\<92>\DBG$GET_BITSIZE_FROM_TYP\   ;
4D 4F 52 46 5F 45 5A 49 53 54 49 42 5F 54 45  00259                                                           ;
                              50 59 54 5F     00268                                                           ;
                           43 45 50 53 45     0026C           .ASCII  \ESPEC\                                 ;


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

                                    007C 00000                      .ENTRY  DBG$GET_BITSIZE_FROM_TYPESPEC, Save R2,R3,- ; 3318
                                                                            R4,R5,R6                                    ;
                 56 00000000G   00   9E 00002                      MOVAB   DBG$STA_VALSPEC, R6                         ;
                 5E             10   C2 00009                      SUBL2   #16, SP                                     ;
```

```
                53        04   AC  D0 0000C          MOVL    TYPE_SPEC, R3                       ; 3346
                54        02   A3  9A 00010          MOVZBL  2(R3), R4
                01             54  91 00014          CMPB    R4, #1                              ; 3349
                               0B  12 00017          BNEQ    1$
                7E        03   A3  9A 00019          MOVZBL  3(R3), -(SP)                        ; 3350
             B9 AF              01  FB 0001D          CALLS   #1, GET_BITSIZE_FROM_DTYPE
                             0084  31 00021          BRW     8$
                02             54  91 00024 1$:      CMPB    R4, #2                              ; 3352
                               1B  12 00027          BNEQ    2$
                52        03   A3  9E 00029          MOVAB   3(R3), VALPTR                       ; 3354
                               5E  DD 0002D          PUSHL   SP                                  ; 3355
                          08   AE  9F 0002F          PUSHAB  VAL_VECT
                               52  DD 00032          PUSHL   VALPTR
                66             03  FB 00034          CALLS   #3, DBG$STA_VALSPEC
                03             6E  D1 00037          CMPL    VALKIND, #3                         ; 3356
                               61  13 0003A          BEQL    7$
                    00000000'  EF  9F 0003C          PUSHAB  P.AAY                               ; 3358
                               4A  11 00042          BRB     6$
                04             54  91 00044 2$:      CMPB    R4, #4                              ; 3364
                               08  1F 00047          BLSSU   4$
                05             54  91 00049          CMPB    R4, #5
                               03  1A 0004C          BGTRU   4$
                             008C  31 0004E 3$:      BRW     14$
                08             54  91 00051 4$:      CMPB    R4, #11
                               F8  13 00054          BEQL    3$
                06             54  91 00056          CMPB    R4, #6                              ; 3367
                               09  12 00059          BNEQ    5$
                55        03   A3  9A 0005B          MOVZBL  3(R3), SIZE                         ; 3368
                55             08  C4 0005F          MULL2   #8, SIZE
                               7C  11 00062          BRB     15$
                07             54  91 00064 5$:      CMPB    R4, #7                              ; 3370
                               44  12 00067          BNEQ    9$
                51        03   A3  9A 00069          MOVZBL  3(R3), NDIMS                        ; 3372
                50        04   A3  9E 0006D          MOVAB   4(R3), FLAGS_PTR                    ; 3373
                51             08  C6 00071          DIVL2   #8, R1                              ; 3374
                52        01 A140  9E 00074          MOVAB   1(R1)[FLAGS_PTR], VALPTR
                               5E  DD 00079          PUSHL   SP                                  ; 3375
                          08   AE  9F 0007B          PUSHAB  VAL_VECT
                               52  DD 0007E          PUSHL   VALPTR
                66             03  FB 00080          CALLS   #3, DBG$STA_VALSPEC
                03             6E  D1 00083          CMPL    VALKIND, #3                         ; 3376
                               15  13 00086          BEQL    7$
                    00000000'  EF  9F 00088          PUSHAB  P.AAZ                               ; 3378
                               01  DD 0008E 6$:      PUSHL   #1
                    00028362   8F  DD 00090          PUSHL   #164706
            00000000G  00          03  FB 00096      CALLS   #3, LIB$SIGNAL
                53        04   AE  D0 0009D 7$:      MOVL    VAL_VECT, DESCPTR                   ; 3380
                               53  DD 000A1          PUSHL   DESCPTR                             ; 3381
                FECC   CF          01  FB 000A3      CALLS   #1, DBG$GET_BITSIZE_FROM_DESC
                55             50  D0 000A8 8$:      MOVL    R0, SIZE
                               37  11 000AB          BRB     17$                                 ; 3346
                08             54  91 000AD 9$:      CMPB    R4, #8                              ; 3384
                               0A  13 000B0          BEQL    10$
                09             54  91 000B2          CMPB    R4, #9                              ; 3387
                               05  13 000B5          BEQL    10$
                0E             54  91 000B7          CMPB    R4, #14                             ; 3390
                               06  12 000BA          BNEQ    11$
```

```
                        55      03      A3  D0 000BC 10$:   MOVL    3(R3), SIZE                     ; 3391
                                        22  11 000C0         BRB     17$                             ;
                        10              54  91 000C2 11$:   CMPB    R4, #16                         ; 3393
                                        07  12 000C5         BNEQ    12$                             ;
                55      03      A3      03  78 000C7         ASHL    #3, 3(R3), SIZE                 ; 3394
                                        16  11 000CC         BRB     17$                             ;
                        11              54  91 000CE 12$:   CMPB    R4, #17                         ; 3396
                                        05  12 000D1         BNEQ    13$                             ;
                        55              30  D0 000D3         MOVL    #48, SIZE                       ; 3397
                                        0C  11 000D6         BRB     17$                             ;
                        12              54  91 000D8 13$:   CMPB    R4, #18                         ; 3399
                                        05  12 000DB         BNEQ    16$                             ;
                        55              20  D0 000DD 14$:   MOVL    #32, SIZE                       ; 3400
                                        02  11 000E0 15$:   BRB     17$                             ;
                                        55  D4 000E2 16$:   CLRL    SIZE                            ; 3403
                        50              55  D0 000E4 17$:   MOVL    SIZE, R0                        ; 3407
                                            04 000E7         RET                                     ; 3409

; Routine Size:  232 bytes,    Routine Base:  DBG$CODE + 13BB
```

```
;  3313          3410  1 ROUTINE GET_BITSIZE_FROM_BLIDST(DSTPTR) =
;  3314          3411  1 !
;  3315          3412  1 ! FUNCTION
;  3316          3413  1 !       This routine takes the address of a BLISS type zero DST record, and
;  3317          3414  1 !       determines the bit size of the data item.
;  3318          3415  1 !
;  3319          3416  1 ! INPUTS
;  3320          3417  1 !       DSTPTR  - The address of the DST record.
;  3321          3418  1 !
;  3322          3419  1 ! OUTPUTS
;  3323          3420  1 !       The bit size of the item is returned as the routine's value.
;  3324          3421  1 !
;  3325          3422  1
;  3326          3423  2     BEGIN
;  3327          3424  2
;  3328          3425  2     MAP
;  3329          3426  2         DSTPTR : REF DST$RECORD;
;  3330          3427  2
;  3331          3428  2     LOCAL
;  3332          3429  2         SIZE,
;  3333          3430  2         BPTR1: REF DST$BLI_TRAILER1,
;  3334          3431  2         BPTR2: REF DST$BLI_TRAILER2;
;  3335          3432  2
;  3336          3433  2
;  3337          3434  2
;  3338          3435  2     IF .DSTPTR[DST$B_BLI_VFLAGS] NEQ 1 THEN RETURN 0;
;  3339          3436  2     BPTR1 = DSTPTR[DST$A_BLI_TRLR1] + .DSTPTR[DST$B_BLI_LNG];
;  3340          3437  2     BPTR2 = BPTR1[DST$A_BLI_TRLR2] + .BPTR1[DST$B_BLI_NAME];
;  3341          3438  2     SIZE = .BPTR2[DST$L_BLI_SIZE];
;  3342          3439  2     IF .DSTPTR[DST$V_BLI_REF] THEN SIZE = 4;
;  3343          3440  2     RETURN .SIZE;
;  3344          3441  2
;  3345          3442  1     END;


                              0004 00000 GET_BITSIZE_FROM_BLIDST:
                                         .WORD   Save R2                                 ;  3410
52      04   AC  D0 00002               MOVL    DSTPTR, R2                              ;  3435
01      04   A2  91 00006               CMPB    4(R2), #1                               ;
                 1E  12 0000A           BNEQ    1$                                      ;
50      02   A2  9A 0000C               MOVZBL  2(R2), R0                               ;  3436
50      03 A240  9E 00010               MOVAB   3(R2)[R0], BPTR1                        ;
51      04   A0  9A 00015               MOVZBL  4(BPTR1), R1                            ;  3437
50      05 A140  9E 00019               MOVAB   5(R1)[BPTR1], BPTR2                     ;
50           60  D0 0001E               MOVL    (BPTR2), SIZE                           ;  3438
          05 A2  95 00021               TSTB    5(R2)                                   ;  3439
                 06  18 00024           BGEQ    2$                                      ;
50               04  D0 00026           MOVL    #4, SIZE                                ;
                     04 00029           RET                                             ;  3440
                 50  D4 0002A 1$:       CLRL    R0                                      ;  3442
                     04 0002C 2$:       RET                                             ;

; Routine Size:  45 bytes,    Routine Base:  DBG$CODE + 14A3
```

```
;  3347          3443  1 ROUTINE FIND_RST_TYPE_ENTRY(DSTPTR) =
;  3348          3444  1 !
;  3349          3445  1 ! FUNCTION
;  3350          3446  1 !       This routine takes the address of a DST record, and tries to find the
;  3351          3447  1 !       RST record which it defined.  In particular, this routine serves to
;  3352          3448  1 !       find the TYPEID for those types which were built into the RST at SET
;  3353          3449  1 !       MODULE time (Record and Enumeration Types), but are now being referenced
;  3354          3450  1 !       via a DST pointer.  The routine simply passes through the RST looking
;  3355          3451  1 !       for an RST$L_DSTPTR which matches the input DSTPTR.
;  3356          3452  1 !
;  3357          3453  1 ! INPUTS
;  3358          3454  1 !       DSTPTR  - The address of the DST record of interest.
;  3359          3455  1 !
;  3360          3456  1 ! OUTPUTS
;  3361          3457  1 !       The TYPEID of the Record or Enumeration Type we are seeking is returned
;  3362          3458  1 !                 as the routine value.  If no such TYPEID exists, zero is re-
;  3363          3459  1 !                 turned as the routine value.
;  3364          3460  1 !
;  3365          3461  1
;  3366          3462  2     BEGIN
;  3367          3463  2
;  3368          3464  2     LOCAL
;  3369          3465  2         MODPTR: REF RST$ENTRY,          ! Pointer to current Module RST Entry
;  3370          3466  2         SYMID: REF RST$ENTRY;           ! Pointer to current symbol RST entry
;  3371          3467  2                                         !      within the current module
;  3372          3468  2
;  3373          3469  2
;  3374          3470  2
;  3375          3471  2     ! Start up a loop through the RST Module Chain which scans all modules.
;  3376          3472  2     ! Note that we skip over the "anonymous module" at the start of the chain;
;  3377          3473  2     ! this module contains all the Global Symbols and is not of interest.
;  3378          3474  2     !
;  3379          3475  2     MODPTR = .RST$START_ADDR[RST$L_NXTMODPTR];
;  3380          3476  2     WHILE .MODPTR NEQ 0 DO
;  3381          3477  3         BEGIN
;  3382          3478  3
;  3383          3479  3
;  3384          3480  3         ! Now search the Symbol Chain for the current module, looking for an
;  3385          3481  3         ! RST Entry whose DSTPTR is the same as the given DST pointer.  When
;  3386          3482  3         ! such an RST Entry is found, return its address as the symbol SYMID.
;  3387          3483  3         !
;  3388          3484  3         SYMID = .MODPTR[RST$L_SYMCHNPTR];
;  3389          3485  3         WHILE .SYMID NEQ 0 DO
;  3390          3486  4             BEGIN
;  3391          3487  4             IF .SYMID[RST$L_DSTPTR] EQL .DSTPTR THEN RETURN .SYMID;
;  3392          3488  4             SYMID = .SYMID[RST$L_SYMCHNPTR];
;  3393          3489  3             END;
;  3394          3490  3
;  3395          3491  3
;  3396          3492  3         ! The DST pointer was not found in this module.  Link on to the next
;  3397          3493  3         ! module and search its symbol chain.
;  3398          3494  3         !
;  3399          3495  3         MODPTR = .MODPTR[RST$L_NXTMODPTR];
;  3400          3496  2         END;
;  3401          3497  2
;  3402          3498  2
;  3403          3499  2     ! We have search the whole RST without finding the desired DST pointer.
```

```
;  3404          3500  2      ! Return zero to indicate that the search failed.
;  3405          3501  2      !
;  3406          3502  2      RETURN 0;
;  3407          3503  2
;  3408          3504  1      END;


                                        0000 00000 FIND_RST_TYPE_ENTRY:
                                                           .WORD   Save nothing                                    ; 3443
                          50 00000000G  00   D0 00002              MOVL    RST$START_ADDR, R0                     ; 3475
                          51       10   A0   D0 00009              MOVL    16(R0), MODPTR                         ;
                                        19   13 0000D 1$:          BEQL    4$                                     ; 3476
                          50       08   A1   D0 0000F              MOVL    8(MODPTR), SYMID                       ; 3484
                                        0D   13 00013 2$:          BEQL    3$                                     ; 3485
                       04 AC       0C   A0   D1 00015              CMPL    12(SYMID), DSTPTR                      ; 3487
                                        0E   13 0001A              BEQL    5$                                     ;
                          50       08   A0   D0 0001C              MOVL    8(SYMID), SYMID                        ; 3488
                                        F1   11 00020              BRB     2$                                     ; 3485
                          51       10   A1   D0 00022 3$:          MOVL    16(MODPTR), MODPTR                     ; 3495
                                        E5   11 00026              BRB     1$                                     ; 3476
                                        50   D4 00028 4$:          CLRL    R0                                     ; 3502
                                             04 0002A 5$:          RET                                            ; 3504

;  Routine Size:  43 bytes,    Routine Base:  DBG$CODE + 14D0

;  3409          3505  1
;  3410          3506  0 END ELUDOM


                                                                   .EXTRN  LIB$SIGNAL

;                                  PSECT SUMMARY
;
;       Name                            Bytes                           Attributes
;
;  DBG$OWN                               12  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  DBG$CODE                            5371  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)
;  DBG$PLIT                             625  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        24         0        1000      00:01.8
;  _$255$DUA28:[DEBUG.OBJ]STRUCDEF.L32;1        32         0         0           7      00:00.1
;  _$255$DUA28:[DEBUG.OBJ]DBGLIB.L32;1        1545        92         5          97      00:01.8
;  _$255$DUA28:[DEBUG.OBJ]DSTRECRDS.L32;1
```

```
;                                      418       200        47        31          00:00.3
;  _$255$DUA28:[DEBUG.OBJ]DBGMSG.L32;1 386         5         1        22          00:00.3



;                                  COMMAND QUALIFIERS

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

; Size:         5371 code + 637 data bytes
; Run Time:        01:35.7
; Elapsed Time:    01:45.7
; Lines/CPU Min:    2197
; Lexemes/CPU-Min: 16567
; Memory Used:  281 pages
; Compilation Complete
```

0100 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

RSTTYPES
LIS

SSIK
LIS

DELETE

DELEMAIN
LIS

DELTA
LIS

SSITAB
LIS

DELETE
MAP

DELTA

DELTA
MAP

SSIU
LIS

SSIUW
LIS

PURGE
LIS

STRUCDEF
LIS

SSIDISP
LIS

SØDELTA
MAP

DELETE
REQ