F 1

```
DDDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBBBB    UUU         UUU     GGGGGGGGGGGG
DDDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBBBB    UUU         UUU     GGGGGGGGGGGG
DDDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBBBB    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  EEE              BBB         BBB  UUU         UUU  GGG
DDD         DDD  EEEEEEEEEEEE     BBBBBBBBBBBB     UUU         UUU  GGG
DDD         DDD  EEEEEEEEEEEE     BBBBBBBBBBBB     UUU         UUU  GGG
DDD         DDD  EEEEEEEEEEEE     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     UUUUUUUUUUUUUUU     GGGGGGGGG
DDDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUUUUUUUUUUUUUU     GGGGGGGGG
DDDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUUUUUUUUUUUUUU     GGGGGGGGG
```

FILEID**RSTTYPES   L 8

```
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
```

```
MODULE RSTTYPES (IDENT = 'V04-000') =

BEGIN

!****************************************************************************
!*                                                                          *
!*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
!*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
!*  ALL RIGHTS RESERVED.                                                    *
!*                                                                          *
!*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
!*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
!*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
!*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
!*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
!*  TRANSFERRED.                                                            *
!*                                                                          *
!*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
!*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
!*  CORPORATION.                                                            *
!*                                                                          *
!*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
!*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
!*                                                                          *
!*                                                                          *
!****************************************************************************

! WRITTEN BY
!       Bert Beander    August, 1980
!       Ken Nappa       August, 1980
!
! MODULE FUNCTION
!       This module contains symbol table access routines for types.  These
!       routines accept a Type ID and return information about the correspond-
!       ing data type.
!
! MODIFIED BY
!       Ping Sager      May 1982        Correct the way to get BITSIZE value
!                                       in DBG$STA_TYP_VARIANT.
!       Ping Sager      Oct 1982        Added DBG$TYPEID_FOR_SET routine
!       Rich Title      Nov 1982        Added DBG$TYPEID_FOR_TPTR routine
!       Ping Sager      Jan 1983        Handled more cases for DST has SEPTYP
!       Rich Title      Jan 1982        Added DBG$TYPEID_FOR_ARRAY routine
! 006   Walter Carrell  Oct 1983        Added support for continuation DST records
!

REQUIRE 'SRC$:DBGPROLOG.REQ';

FORWARD ROUTINE
    DBG$FIND_SEPTYP,                    ! Get Separate Type Specification
                                        !       from a given DST record
    DBG$GET_BITSIZE_FROM_DESC,          ! Get the size from the descriptor
    DBG$GET_BITSIZE_FROM_TYPESPEC,      ! Get the size from Type Spec.
    DBG$STA_SYMTYPE:    NOVALUE,        ! Return a symbol's TYPEID and FCODE
    DBG$STA_SYMSIZE:    NOVALUE,        ! Return the bit size of a data type
    DBG$STA_TYPEFCODE,                  ! Return the type format code
    DBG$STA_TYP_AREA: NOVALUE,          ! Return the info for an area type
```

```
;   1   DBG$STA_TYP_ARRAY:  NOVALUE,        ! Return info for an array type
;   1   DBG$STA_TYP_ATOMIC: NOVALUE,        ! Return type code for an atomic type
;   1   DBG$STA_TYP_DESCR:  NOVALUE,        ! Return descriptor for descriptor type
;   1   DBG$STA_TYP_ENUM:   NOVALUE,        ! Return info for an enumeration type
;   1   DBG$STA_TYP_FILE:   NOVALUE,        ! Return info for a file type
;   1   DBG$STA_TYP_OFFSET: NOVALUE,        ! Return info for a offset type
;   1   DBG$STA_TYP_PICT:   NOVALUE,        ! Return info for a picture type
;   1   DBG$STA_TYP_TYPEDPTR: NOVALUE,      ! Return info for a typed pointer type
;   1   DBG$STA_TYP_RECORD: NOVALUE,        ! Return info for a record type
;   1   DBG$STA_TYP_SET:    NOVALUE,        ! Return info for a set type
;   1   DBG$STA_TYP_SUBRNG: NOVALUE,        ! Return info for a subrange type
;   1   DBG$STA_TYP_VARIANT:NOVALUE,        ! Return info for a variant type
;   1   DBG$STA_TYP_VARIANT_COMP: NOVALUE,  ! XXX
;   1   DBG$TRANS_TYPE_CODE,                ! Translates DST fcodes to RST fcodes
;   1   DBG$TYPEID_FOR_ARRAY,               ! Return Type ID for array
;   1   DBG$TYPEID_FOR_ATOMIC,              ! Return Type ID for atomic type
;   1   DBG$TYPEID_FOR_SET,                 ! Return Type ID for set type
;   1   DBG$TYPEID_FOR_TPTR,                ! Return Type ID for TPTR type
;   1   TYPEID_FOR_COB_HACK,                ! Return Type ID for COBOL Hack record
;   1   TYPEID_FOR_DESCR,                   ! Return Type ID for descriptor type
;   1   TYPEID_FROM_DST_TYPESPEC,           ! Return Type ID from a Type Speci-
;   1                                       !       fication embedded in the DST
;   1   TYPEID_FROM_DST_RECORD,
;   1   DESC_OR_ARRAY_OR_ATOM,              ! Determines the nature of the type.
;   1   FIND_RST_TYPE_ENTRY,
;   1   FIND_TYPE_RECORD,                   ! Returns symid of type info
;   1   FIND_TYPESPEC,                      ! Get Type Spec from Sep Type
;   1   FIND_TYPREC_FROM_TSPEC,
;   1   GET_BITSIZE_FROM_DTYPE,             ! Get the size from Dtype
;   1   GET_BITSIZE_FROM_BLIDST;            ! Get the size from BLI DST
```

```
1 EXTERNAL ROUTINE
1     DBG$DATA_LENGTH,                          ! Obtain bitsize from VMS descriptor
1     DBG$GET_MEMORY,                           ! Get a permanent memory block
1     DBG$GET_TEMPMEM,                          ! Get a "temporary" memory block
1     DBG$READ_ACCESS: NOVALUE,                 !
1     DBG$RST_DST_PTR,                          ! Return the real DST pointer
1     DBG$RST_NEXT_DST,                         ! Return the next DST given a DST
1     DBG$STA_SYMKIND,                          ! Get a specified symbol's kind
1     DBG$STA_SYMNAME: NOVALUE,                 ! Obtain symbol name.
1     DBG$STA_SYMVALUE: NOVALUE,                ! Evaluate a specified symbol's value
1     DBG$STA_VALSPEC: NOVALUE;                 ! Evaluate a DST Value Spec
1
1 EXTERNAL
1     DST$BEGIN_ADDR,                           ! The start address of the DST in memory
1     RST$START_ADDR: REF RST$ENTRY,            ! Pointer to RST anonymous module
1     RST$TEMP_LIST;                            ! Pointer to Temporary RST Entry List
1
1 OWN
1     NOVEL_LENGTH_FLAG;                        ! This flag is used to indicate
1                                               !   we have encounted a novel length
1                                               !   dst, we need to take the size
1                                               !   specificed in this dst to override
1                                               !   the original length.  Especially,
1                                               !   when a data has novel length dst,
1                                               !   and this data item has a typeid,
1                                               !   (typid has its own size), in
1                                               !   this case we need to modify the
1                                               !   typeid to reflect the novel
1                                               !   length by making a copy of
1                                               !   the typeid with new novel length
1                                               !   for the data.
```

```
1 GLOBAL ROUTINE DBG$FIND_SEPTYP(DST_PTR, TYPE_PTR, TYPE_SPEC) =
1 !
1 ! FUNCTION
1 !       Given Separate Type Specification DST Record, Figure out its Type
1 !       Specification and the size of the described object is returned.
1 !
1 ! INPUTS
1 !
1 !       DST_PTR         - Pointer to DST Record.
1 !
1 !       TYPE_PTR        - Pointer to DST Type Spec fields.
1 !
1 !       TYPE_SPEC       - Pointer to DST Type Spec.
1 !
1 ! OUTPUTS
1 !       Size of the object in Type Spec is returned.
1 !
1
2     BEGIN
2
2     MAP
2       DST_PTR: REF DST$RECORD,              ! Pointer to DST Type Spec Record
2       TYPE_PTR: REF VECTOR[,LONG],          ! Pointer to DST Type Spec fields
2       TYPE_SPEC: REF VECTOR[,LONG];         ! Pointer to DST Type Spec
2
2     LOCAL
2       ENUMBEG: REF DST$RECORD,              ! Pointer to Enum. begin record
2       RECBEG: REF DST$RECBEG_TRLR,          ! Pointer to Rec. begin trailer record
2       NAMEPTR: REF VECTOR[,BYTE],           ! Pointer to name of the record
2       TYPEPTR: REF DST$RECORD,              ! Pointer to DST Type Spec fields
2       TYPESPEC: REF DST$TYPE_SPEC,          ! Pointer to DST Type Spec
2       SIZE;                                 ! Size of the Object
2
2     SIZE = 0;
2 !    TYPEPTR = DST_PTR[DST$A_NEXT] + .DST_PTR[DST$B_LENGTH];                  ! D006
2     TYPEPTR = DBG$RST_NEXT_DST( .DST_PTR );                                  ! A006
2     TYPESPEC =
2       TYPEPTR[DST$A_TYPSPEC_TS_ADDR] + .TYPEPTR[DST$B_TYPSPEC_NAME];
2
2     WHILE TRUE DO
3       BEGIN
3       SELECTONE .TYPEPTR[DST$B_TYPE] OF
3           SET
3           [DST$K_SEPTYP]:
4               BEGIN
4               TYPEPTR = DBG$RST_NEXT_DST( .TYPEPTR );                        ! M006
4               TYPESPEC = .TYPEPTR[DST$B_TYPSPEC_NAME] +
4                       TYPEPTR[DST$A_TYPSPEC_TS_ADDR];
3               END;
3
3           [DST$K_GLOBNXT]:
4               BEGIN
4 !             TYPEPTR = .TYPEPTR + 2;                                        ! A006
4               TYPEPTR = DBG$RST_NEXT_DST( .TYPEPTR );                        ! A006
4               TYPESPEC = .TYPEPTR[DST$B_TYPSPEC_NAME] +
4                       TYPEPTR[DST$A_TYPSPEC_TS_ADDR];
3               END;
```

```
; 3
; 3               [DST$K_TYPSPEC]:
; 4                   BEGIN
; 4                   TYPESPEC = FIND_TYPESPEC(.TYPESPEC, SIZE, TYPEPTR);
; 4                   EXITLOOP;
; 3                   END;
; 3
; 3               [DST$K_ENUMBEG]:
; 4                   BEGIN
; 4                   ENUMBEG = .TYPEPTR;
; 4                   SIZE = .TYPEPTR[DST$B_ENUMBEG_LENG];
; 4                   EXITLOOP;
; 3                   END;
; 3
; 3               [DST$K_RECBEG]:
; 4                   BEGIN
; 4
; 4
; 4                   ! Advance the pointer to the name field.
; 4                   ! (1 byte length, 1 byte type, 1 byte vflags, 4 bytes value).
; 4                   !
; 4                   RECBEG = .TYPEPTR + 1 + 1 + 1 + 4;
; 4                   NAMEPTR = .RECBEG;
; 4
; 4                   ! Advance the pointer to the trailer field.
; 4                   !
; 4                   RECBEG = .RECBEG + .NAMEPTR[0] + 1;
; 4                   SIZE = .RECBEG[DST$L_RECBEG_SIZE];
; 4                   EXITLOOP;
; 3                   END;
; 3
; 3               [OTHERWISE]:
; 3                   EXITLOOP;
; 3
; 3               TES;
; 3
; 2           END;                    ! End of WHILE Loop.
; 2
; 2       IF .SIZE EQL 0 THEN SIZE = DBG$GET_BITSIZE_FROM_TYPESPEC(.TYPESPEC);
; 2       TYPE_PTR[0] = .TYPEPTR;
; 2       TYPE_SPEC[0] = .TYPESPEC;
; 2       RETURN .SIZE;
; 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
```

```
GLOBAL ROUTINE DBG$STA_SYMTYPE(SYMID, FCODE, TYPEID): NOVALUE =
!
! FUNCTION
!       This routine takes a SYMID and returns a TYPEID and FCODE for the
!       type associated with the input SYMID.  In the case where the SYMID
!       defines a type item, the returned TYPEID is identical to the SYMID.
!
! INPUTS
!       SYMID   - The SYMID for the symbol or type whose TYPEID and  Format Code
!                 are to be returned.
!
!       FCODE   - The address of a longword location to receive the FCODE
!                 associated with the given SYMID.
!
!       TYPEID  - The address of a longword location to receive the
!                 TYPEID describing the type associated with SYMID.
!
! OUTPUTS
!       FCODE   - The format code associated with SYMID is returned to FCODE.
!
!       TYPEID  - A TYPEID defining the type associated with the input
!                 SYMID is returned to TYPEID.
!

    BEGIN

    MAP
        SYMID: REF RST$ENTRY,           ! Pointer to input symbol's RST entry
        FCODE: REF VECTOR[1],           ! Address where FCODE is returned
        TYPEID: REF VECTOR[1];          ! Address where TYPEID is returned

    LOCAL
        SIZE;                           ! Bit size of a data item of the type



    ! Find the Type RST Entry associated with the input SYMID.  Return its
    ! address to TYPEID[0] and its Format Code to FCODE[0].  Signal an internal
    ! error if the TYPEID turns out to be zero--that should never happen.
    !
    TYPEID[0] = FIND_TYPE_RECORD(.SYMID, FCODE[0], SIZE);
    IF .TYPEID[0] EQL 0
    THEN
        $DBG_ERROR('RSTTYPES\SYMTYPE');

    RETURN;

    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
```

```
0402  1 GLOBAL ROUTINE DBG$STA_SYMSIZE(SYMID, SIZE): NOVALUE =
0403  1 !
0404  1 ! FUNCTION
0405  1 !       This routine takes a SYMID and returns the size in bits associated
0406  1 !       with that input SYMID.
0407  1 !
0408  1 ! INPUTS
0409  1 !       SYMID   - The SYMID for the symbol or type whose bitsize is to be
0410  1 !                 returned.
0411  1 !
0412  1 !       SIZE    - The address of a longword location to receive the size in bits
0413  1 !                 of the item described by the given SYMID.
0414  1 !
0415  1 ! OUTPUTS
0416  1 !       SIZE    - The size in bits of the input item is returned to SIZE.
0417  1 !
0418  1 !       No value is returned.
0419  1 !
0420  1
0421  2     BEGIN
0422  2
0423  2     MAP
0424  2         SYMID: REF RST$ENTRY,               ! Pointer to input symbol's RST entry
0425  2         SIZE: REF VECTOR[1];                ! Address where size is to be returned
0426  2
0427  2     LOCAL
0428  2         TYPEID: REF RST$ENTRY,              ! The symbol's Type ID
0429  2         FCODE;                              ! The symbol's Type Format Code
0430  2
0431  2
0432  2
0433  2     ! Get to the Type DST record and pick up the size information.  If no type
0434  2     ! info was found, set the bit size to zero (wich means size is unknown).
0435  2     !
0436  2     TYPEID = FIND_TYPE_RECORD(.SYMID, FCODE, SIZE[0]);
0437  2     IF .TYPEID EQL 0 THEN SIZE[0] = 0
0438  2     ELSE IF (.SIZE[0] EQL 0) AND
0439  3        ((.FCODE EQL RST$K_TYPE_PTR) OR (.FCODE EQL RST$K_TYPE_TPTR))
0440  2         THEN SIZE[0] = 32;
0441  2     RETURN;
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
```

```
1 GLOBAL ROUTINE DBG$STA_TYP_ARRAY(TYPEID, DSCADDR, CELLTYPE,
1                                   NDIMS, DIMVECPTR, BITSIZE): NOVALUE =
1 !
1 ! FUNCTION
1 !       This routine returns the type information associated with an Array data
1 !       type.  The type of interest is identified by a Type ID, and the returned
1 !       information includes the array descriptor, the cell type, the number of
1 !       dimensions, the size in bits of the array, and information about each
1 !       specific dimension (the subscript type, the lower bound, and the upper
1 !       bound).
1 !
1 ! INPUTS
1 !       TYPEID  - The Type ID of the Array data type about which information is
1 !                 desired.  Its Format Code must be RST$K_TYPE_ARRAY.
1 !
1 !       DSCADDR - The address of a longword location to receive a pointer to the
1 !                 array's descriptor.
1 !
1 !       CELLTYPE - The address of a longword location to receive the Type ID of
1 !                 the data type of the individual array elements.
1 !
1 !       NDIMS   - The address of a longword location to receive the number of
1 !                 dimensions in the array.
1 !
1 !       DIMVECPTR - The address of a longword location to receive a pointer to a
1 !                 vector of subscript information.
1 !
1 !       BITSIZE - The address of a longword location to receive the size in bits
1 !                 of an array of this type.
1 !
1 ! OUTPUTS
1 !       DSCADDR - The address of the array descriptor is returned to DSCADDR.
1 !                 This may be a contiguous or noncontiguous array descriptor.
1 !                 The descriptor may disappear at the end of the current Debug
1 !                 command.
1 !
1 !       CELLTYPE - The Type ID of the individual array elements' data type is
1 !                 returned to CELLTYPE.
1 !
1 !       NDIMS   - The number of array dimensions is returned to NDIMS.
1 !
1 !       DIMVECPTR - A pointer to a subscript vector is returned to
1 !                 DIMVECPTR.  This vector contains one longword
1 !                 per dimension, giving the subscript Type ID.
1 !                 The vector is allocated in temporary storage--it disappears
1 !                 at the end of the current Debug command.
1 !
1 !       BITSIZE - The size in bits is returned to BITSIZE.
1 !
1 !       No value is returned.
1 !
1 ! WARNING:
1 !       The five output parameters must be distinct locations in the
1 !       callers address space. For example, you cannot do:
1 !       DBG$STA_TYP_ARRAY (.TYPEID, JUNK, CELLTYPE, JUNK, JUNK, JUNK)
1 !       to just find out the celltype. The problem is that the routine
1 !       is using the addresses passed in (.JUNK in this case) as local
```

```
!       storage, which it really should not be doing, and stores into
!       one .JUNK will trip up reads from another. So you must pass
!       in distinct parameters,
!       DBG$STA_TYP_ARRAY (.TYPEID, JUNK1, CELLTYPE, JUNK2, JUNK3, JUNK4)
!       I got tripped up on this so I thought I'd warn others.
!       R. Title Nov 1982.
!

    BEGIN

    MAP
        TYPEID : REF RST$ENTRY;             ! The input TYPEID for the array type

    LOCAL
        I,
        BITS_USED,
        DATA_TYPE,
        SIZE,
        FCODE,
        VALKIND,
        TYPE_SPEC : REF DST$TYPE_SPEC,
        TYPE_SPEC1 : REF DST$TYPE_SPEC,
        DST_PTR : REF DST$RECORD,
        TYPEID_PTR : REF RST$ENTRY,
        DESC_PTR : REF BLOCK[,BYTE],
        SUBSCR_VECT : REF VECTOR,
        VAL_VECT : VECTOR[3],
        VAL_PTR : REF DST$VAL_SPEC;



    TYPEID_PTR = FIND_TYPE_RECORD(.TYPEID, FCODE, SIZE);


    ! TYPEID_PTR -> RST Entry for this array type.  Verify the format code.
    !
    IF .FCODE NEQ RST$K_TYPE_ARRAY THEN $DBG_ERROR('RSTTYPES\TYP_ARRAY 10');
    DST_PTR = .TYPEID_PTR[RST$L_DSTPTR];
    TYPE_SPEC = DST_PTR[DST$A_TYPSPEC_TS_ADDR] + .DST_PTR[DST$B_TYPSPEC_NAME];
    TYPE_SPEC = FIND_TYPESPEC(.TYPE_SPEC, SIZE, DST_PTR);


    ! For array, Typeid does not carray the size information.  However
    ! FIND_TYPESPEC routine does go further into the DST to dig out
    ! the size of the array for contiguous array.
    !
    .BITSIZE = .SIZE;


    ! TYPE_SPEC -> Type Specification field for this array type.
    ! If old format array spec, get info from descriptor
    !
    IF .TYPE_SPEC[DST$B_TS_KIND] EQL DST$K_TS_DSC
    THEN
        BEGIN

        ! Obtain a descriptor by evaluating the value specification.
```

```
3          !
3          VAL_PTR = TYPE_SPEC[DST$A_TS_DSC_VSPEC_ADDR];
3          DBG$STA_VALSPEC(.VAL_PTR, VAL_VECT, VALKIND);
3          IF .VALKIND NEQ DBG$K_VAL_DESCR
3          THEN
3              $DBG_ERROR('RSTTYPES\TYP_ARRAY 20');
3          DESC_PTR = .VAL_VECT[0];
3          DBG$READ_ACCESS(.DESC_PTR, 12);
3          .DSCADDR = .DESC_PTR;
3
3          ! Obtain the dtype of an element and the bitsize of an element.
3          ! Note that DBG$DATA_LENGTH returns the bitsize of an element
3          ! because it just looks at the dtype field and the length field,
3          ! not the class field.
3          !
3          DATA_TYPE = .DESC_PTR[DSC$B_DTYPE];
3          BITS_USED = DBG$DATA_LENGTH(.DESC_PTR);
3
3          ! Build a typeid for the array element.
3          !
3          IF (.BITS_USED NEQ 0)                 AND
3             (.DESC_PTR[DSC$B_SCALE] EQL 0) AND
4             (.DESC_PTR[DSC$B_DIGITS] EQL 0)
3          THEN
3              .CELLTYPE = DBG$TYPEID_FOR_ATOMIC(.DATA_TYPE,.BITS_USED,TRUE)
3
3          ELSE
3              .CELLTYPE = TYPEID_FOR_DESCR(.DESC_PTR,RST$K_TYPE_DESCR,TRUE);
3
3          ! Use the descriptor to fill in the remaining output parameters.
3          !
3          .NDIMS = .DESC_PTR[DSC$B_DIMCT];
3          SUBSCR_VECT = DBG$GET_TEMPMEM(..NDIMS);
3          .DIMVECPTR = .SUBSCR_VECT;
3          INCR I FROM 0 TO ..NDIMS - 1 DO
3              SUBSCR_VECT[.I] = 0;
3          END
3
3
3      ! New format type specification for this array.
3      !
2      ELSE
3          BEGIN
3          IF .TYPE_SPEC[DST$B_TS_KIND] EQL DST$K_TS_ARRAY
3          THEN
4              BEGIN
4              LOCAL
4                  FLAGS_PTR : REF BITVECTOR;
4
4              .NDIMS = .TYPE_SPEC[DST$B_TS_ARRAY_DIM];
4              FLAGS_PTR = TYPE_SPEC[DST$A_TS_ARRAY_FLAGS_ADDR];
4
4
4              ! FLAGS_PTR -> Bitvector of optional Type-Spec indicators.
4              !
4              VAL_PTR = .FLAGS_PTR + (..NDIMS/8) + 1;
4              DBG$STA_VALSPEC(.VAL_PTR, VAL_VECT, VALKIND);
```

```
IF .VALKIND NEQ DBG$K_VAL_DESCR
THEN
    $DBG_ERROR('RSTTYPES\TYP_ARRAY 30');

DESC_PTR = .VAL_VECT[0];
.DSCADDR = .DESC_PTR;
IF .VAL_PTR[DST$B_VS_VFLAGS] EQL DST$K_VS_FOLLOWS
THEN
    TYPE_SPEC1 = .VAL_PTR + .VAL_PTR[DST$W_VS_LENGTH] + 3

ELSE
    TYPE_SPEC1 = .VAL_PTR + 5;


! TYPE_SPEC1 -> First optional Type Spec within this record.
! TYPE_SPEC1 will move along through the optional Type Specs.
!
IF .FLAGS_PTR[0]
THEN
    BEGIN
    ! Optional cell type is present.
    .CELLTYPE = FIND_TYPREC_FROM_TSPEC(.TYPE_SPEC1);
    TYPE_SPEC1 = .TYPE_SPEC1 + 2 + .TYPE_SPEC1[DST$W_TS_LENGTH];
    END

ELSE
    BEGIN
    ! no optional cell type; use cell type from descriptor.
    DBG$READ_ACCESS(.DESC_PTR, 12);

    ! Obtain the dtype of an element and the bitsize of an element.
    ! Note that DBG$DATA_LENGTH returns the bitsize of an element
    ! because it just looks at the dtype field and the length field,
    ! not the class field.
    !
    DATA_TYPE = .DESC_PTR[DSC$B_DTYPE];
    BITS_USED = DBG$DATA_LENGTH(.DESC_PTR);

    IF (.BITS_USED NEQ 0)                AND
       (.DESC_PTR[DSC$B_SCALE] EQL 0) AND
       (.DESC_PTR[DSC$B_DIGITS] EQL 0)
    THEN
        .CELLTYPE = DBG$TYPEID_FOR_ATOMIC(.DATA_TYPE,.BITS_USED,TRUE)

    ELSE
        .CELLTYPE = TYPEID_FOR_DESCR(.DESC_PTR,RST$K_TYPE_DESCR,TRUE);

    END;


! Now build DIMVECPTR.  Each element receives either a TYPEID or
! zero, depending on the appropriate bit in the flags vector.
!
SUBSCR_VECT = DBG$GET_TEMPMEM(..NDIMS);
.DIMVECPTR = .SUBSCR_VECT;
INCR I FROM 1 TO ..NDIMS DO
    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           RETURN;
;  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$STA_TYP_ARRAY, Save R2,R3,R4,R5,R6,R7,- ; 0474
                                                               R8,R9,R10,R11                              ;
             5B 00000000G   00   9E 00002          MOVAB   DBG$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(R1), 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 0012A        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, @DSCADDR                          ; 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, @CELLTYPE                               ;
                   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
```

```
GLOBAL ROUTINE DBG$STA_TYP_ATOMIC(TYPEID, TYPECODE, BITSIZE): NOVALUE =
!
! FUNCTION
!       This routine returns the VAX standard type code ("one-byte type code")
!       and the bit length for a specified atomic data type.
!
! INPUTS
!       TYPEID   - The Type ID of an Atomic data type whose VAX standard type
!                  code is to be returned.  This data type must have the Format
!                  Code RST$K_TYPE_ATOMIC.
!
!       TYPECODE - The address of a longword location to receive the type code.
!
!       BITSIZE  - The address of a longword location to receive the bit length
!                  of an item of this type.
!
! OUTPUTS
!       TYPECODE - The VAX standard type code for the specified data type is
!                  returned to TYPECODE.
!
!       BITSIZE  - The bit length is returned to BITSIZE.
!
!       No value is returned.
!

    BEGIN

    MAP
        TYPEID: REF RST$ENTRY;          ! The input TYPEID for the atomic type

    LOCAL
        FCODE,
        SIZE,
        DST_PTR : REF DST$RECORD,
        TYPE_SPEC : REF DST$TYPE_SPEC,
        TYPEID_PTR : REF RST$ENTRY;



    TYPEID_PTR = FIND_TYPE_RECORD(.TYPEID, FCODE, SIZE);
    IF .FCODE NEQ RST$K_TYPE_ATOMIC
    THEN
        $DBG_ERROR('RSTTYPES\TYP_ATOMIC');

    DST_PTR = .TYPEID_PTR[RST$L_DSTPTR];
    TYPE_SPEC = DST_PTR[DST$A_TYPSPEC_TS_ADDR] + .DST_PTR[DST$B_TYPSPEC_NAME];
    .TYPECODE = .TYPE_SPEC[DST$B_TS_ATOM_TYP];
    .BITSIZE = .SIZE;
    IF (..TYPECODE EQL DSC$K_DTYPE_T) AND (.SIZE EQL 0) THEN .BITSIZE = %BPUNIT;
    RETURN;

    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
```

```
GLOBAL ROUTINE DBG$STA_TYP_DESCR(TYPEID, DSCADDR): NOVALUE =
!
! FUNCTION
!       This routine returns the address of a VAX standard descriptor for a
!       specified descriptor-defined data type.
!
! INPUTS
!       TYPEID  - The Type ID of the data type whose descriptor is to be
!                 returned.  Its Format Code must be RST$K_TYPE_DESCR.
!
!       DSCADDR - The address of a longword location to receive the descriptor
!                 address.
!
! OUTPUTS
!       DSCADDR - A pointer to the data type's VAX standard descriptor is
!                 returned to DSCADDR.  This descriptor may not survive past
!                 the end of the current command.
!
!       No value is returned.
!

    BEGIN

    MAP
        TYPEID : REF RST$ENTRY;         ! The input TYPEID for the descriptor
                                        !       data type

    LOCAL
        SIZE,
        FCODE,
        DSC_PTR,
        TYPE_SPEC : REF DST$TYPE_SPEC,
        DST_PTR : REF DST$RECORD,
        TYPEID_PTR : REF RST$ENTRY,
        VALKIND,
        VAL_VECT : VECTOR[3],
        VAL_PTR : REF DST$VAL_SPEC;



    TYPEID_PTR = FIND_TYPE_RECORD(.TYPEID, FCODE, SIZE);
    IF .FCODE NEQ RST$K_TYPE_DESCR THEN $DBG_ERROR('RSTTYPES\TYP_DESCR 10');
    DST_PTR = .TYPEID_PTR[RST$L_DSTPTR];
    TYPE_SPEC = DST_PTR[DST$A_TYPSPEC_TS_ADDR] + .DST_PTR[DST$B_TYPSPEC_NAME];
    TYPE_SPEC = FIND_TYPESPEC(.TYPE_SPEC, SIZE, DST_PTR);
    VAL_PTR = TYPE_SPEC[DST$A_TS_DSC_VSPEC_ADDR];
    DBG$STA_VALSPEC(.VAL_PTR, VAL_VECT, VALKIND);
    IF .VALKIND NEQ DBG$K_VAL_DESCR THEN $DBG_ERROR('RSTTYPES\TYP_DESCR 20');
    .DSCADDR = .VAL_VECT[0];
    RETURN;

    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
```

```
1 GLOBAL ROUTINE DBG$STA_TYP_ENUM(TYPEID, NELTS, ELTVECPTR, BITSIZE): NOVALUE =
1 !
1 ! FUNCTION
1 !       This routine returns information about a specified enumeration type.
1 !       The type is identified by a Type ID, and the returned information
1 !       includes the number of enumeration elements, a vector of SYMIDs
1 !       for all the elements, and the bit length of an element of this type.
1 !       (Each "element" is a constant of the enumeration type in this terminology.)
1 !
1 ! INPUTS
1 !       TYPEID    - The Type ID of the Enumeration type whose information is to
1 !                   be returned.  Its Format Code must be RST$K_TYPE_ENUM.
1 !
1 !       NELTS     - The address of a longword location to receive the number of
1 !                   constants there are of the enumeration type.
1 !
1 !       ELTVECPTR - The address of a longword location to receive a pointer to
1 !                   a vector of SYMIDs for the enumeration type elements.
1 !
1 !       BITSIZE - The address of a longword location to receive the bit length of
1 !                   an item of this type.
1 !
1 ! OUTPUTS
1 !       NELTS     - The number of enumeration type "elements" (i.e., the number of
1 !                   distinct values of the type) is returned to NELTS.
1 !
1 !       ELTVECPTR - A pointer to a vector of SYMIDs for the enumeration type
1 !                   elements is returned to ELTVECPTR.  The elements are stored
1 !                   in order of their values.  Each element's SYMID can then be
1 !                   used to extract the element's name and value.  This vector
1 !                   disappears at the end of the current Debug command.
1 !
1 !       BITSIZE - The bit length is returned to BITSIZE.
1 !
1 !       No value is returned.
1 !
1
2     BEGIN
2
2     MAP
2         TYPEID : REF RST$ENTRY;           ! The input TYPEID for the enumeration
2                                           !     type
2
2     LOCAL
2         TYPEID_PTR : REF RST$ENTRY,
2         FCODE,
2         SIZE;
2
2
2
2     TYPEID_PTR = FIND_TYPE_RECORD(.TYPEID, FCODE, SIZE);
2     IF .FCODE NEQ RST$K_TYPE_ENUM THEN $DBG_ERROR('RSTTYPES\TYP_ENUM');
2     .NELTS = .TYPEID_PTR[RST$L_TYPCOMPCNT];
2     .ELTVECPTR = TYPEID_PTR[RST$A_TYPCOMPLST];
2     .BITSIZE = .SIZE;
2     RETURN;
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
```

```
GLOBAL ROUTINE DBG$STA_TYP_PICT(TYPEID, LANGCODE, PICTPTR, PICTVAL, PSCALE): NOVALUE =
!
! FUNCTION
!       This routine returns information about a Picture data type (as used in
!       Cobol and PL/I).  The input data type is represented by a Type ID, and
!       the picture's language code and picture string are returned as output.
!
! INPUTS
!       TYPEID   - The Type ID of the Picture data type whose information is to
!                  be returned.  Its Format Code must be RST$K_TYPE_PICT.
!
!       LANGCODE - The address of a longword location to receive the language
!                  code.
!
!       PICTPTR  - The address of a longword location to receive a pointer to the
!                  picture itself.
!
!       PICTVAL  - The address of a longword location to receive a pointer to the
!                  language-specific picture encoding.
!
!       PSCALE   - (Optional) The address of a longword to receive the
!                   scale factor and digit-count for this picture.
! OUTPUTS
!       LANGCODE - The picture language code is returned to LANGCODE.
!
!       PICTPTR  - A pointer to the picture itself, represented as a Counted
!                  ASCII string, is returned to PICTPTR.
!
!       PICTVAL  - A pointer to the language-specific picture encoding is returned
!                  to PICTVAL; if no such encoding exists, zero is returned.
!
!       No value is returned.
!

    BEGIN

    BUILTIN
        ACTUALCOUNT;

    MAP
        TYPEID : REF RST$ENTRY;         ! The input TYPEID for the picture type

    LOCAL
        TYPEID_PTR : REF RST$ENTRY,
        FCODE,
        DST_PTR : REF DST$RECORD,
        SIZE,
        VALKIND,
        VAL_SPEC,
        VAL_VECT : VECTOR[3],
        TYPE_SPEC : REF DST$TYPE_SPEC;



    TYPEID_PTR = FIND_TYPE_RECORD(.TYPEID, FCODE, SIZE);
    IF .FCODE NEQ RST$K_TYPE_PICT THEN $DBG_ERROR('RSTTYPES\TYP_PICT');
    DST_PTR = .TYPEID_PTR[RST$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, VA[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$STA_TYP_PICT, Save R2,R3          ; 0884
                    5E               1C   C2 00002           SUBL2   #28, SP
                          08    AE   9F 00005           PUSHAB  SIZE                                  ; 0938
                          04    AE   9F 00008           PUSHAB  FCODE                                 ;
                          04    AC   DD 0000B           PUSHL   TYPEID                                ;
             0000V   CF         03   FB 0000E           CALLS   #3, FIND_TYPE_RECORD
                     52         50   D0 00013           MOVL    R0, TYPEID_PTR
                     05         6E   D1 00016           CMPL    FCODE, #5                             ; 0939
                                15   13 00019           BEQL    1$                                    ;
                 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$SIGNAL                        ;
                 04  AE   0C    A2   D0 00030 1$:       MOVL    12(TYPEID_PTR), DST_PTR               ; 0940
                     51   04    AE   D0 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   D0 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
```

```
1 GLOBAL ROUTINE DBG$STA_TYP_TYPEDPTR(TYPEID, REFTYPEID): NOVALUE =
1 !
1 ! FUNCTION
1 !       This routine accepts as input the Type ID of a Typed Pointer data type
1 !       and returns the Type ID of the data type it points to.
1 !
1 ! INPUTS
1 !       TYPEID  - The Type ID of the Typed Pointer type whose referenced type
1 !                 is to be returned.  Its Format Code must be RST$K_TYPE_TPTR.
1 !
1 !       REFTYPEID - The address of a longword location to receive the pointed-to
1 !                 data type's Type ID.
1 !
1 ! OUTPUTS
1 !       REFTYPEID - The Type ID of the data type which is pointed to by the
1 !                 TYPEID Typed Pointer type is returned to REFTYPEID.
1 !
1 !       No value is returned.
1 !
1
2     BEGIN
2
2     MAP
2         TYPEID: REF RST$ENTRY;              ! The input TYPEID of the Typed Pointer
2                                             !       data type
2
2     LOCAL
2         FCODE,
2         TYPEID_PTR: REF RST$ENTRY,
2         SIZE,
2         DST_PTR: REF DST$RECORD,
2         TYPE_SPEC: REF DST$TYPE_SPEC;
2
2
2
2     TYPEID_PTR = FIND_TYPE_RECORD(.TYPEID, FCODE, SIZE);
2     IF .FCODE NEQ RST$K_TYPE_TPTR THEN $DBG_ERROR('RSTTYPES\TYP_TYPEDPTR');
2     DST_PTR = .TYPEID_PTR[RST$L_DSTPTR];
2     TYPE_SPEC = DST_PTR[DST$A_TYPSPEC_TS_ADDR] + .DST_PTR[DST$B_TYPSPEC_NAME];
2     TYPE_SPEC = FIND_TYPESPEC(.TYPE_SPEC, SIZE, DST_PTR);
2     TYPE_SPEC = TYPE_SPEC[DST$A_TS_TPTR_TSPEC_ADDR];
2     .REFTYPEID = FIND_TYPREC_FROM_TSPEC(.TYPE_SPEC);
2     RETURN;
2
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$STA_TYP_TYPEDPTR, Save R2            : 0963
           5E             0C  C2 00002              SUBL2   #12, SP                                  :
                    08    AE  9F 00005              PUSHAB  SIZE                                     : 0998
                    04    AE  9F 00008              PUSHAB  FCODE                                    :
                    04    AC  DD 0000B              PUSHL   TYPEID                                   :
      0000V  CF           03  FB 0000E              CALLS   #3, FIND_TYPE_RECORD                     :
           52             50  D0 00013              MOVL    R0, TYPEID_PTR                           :
           06             6E  D1 00016              CMPL    FCODE, #6                                : 0999
                          15  13 00019              BEQL    1$                                       :
              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$SIGNAL                           :
         04  AE      0C   A2  D0 00030  1$:         MOVL    12(TYPEID_PTR), DST_PTR                  : 1000
             51      04   AE  D0 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  C0 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  D0 00059              MOVL    R0, @REFTYPEID                           :
                          04     0005D              RET                                              : 1007

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

```
1 GLOBAL ROUTINE DBG$STA_TYP_FILE(TYPEID, LANGUAGE, RECTYPEID): NOVALUE =
1 !
1 ! FUNCTION
1 !       This routine accepts as input the Type ID of a File data type
1 !       and returns the associated language code and record-type typeid.
1 !
1 ! INPUTS
1 !       TYPEID  - The Type ID of the File type.
1 !                 Its Format Code must be RST$K_TYPE_FILE.
1 !
1 !       LANGUAGE - The address of a longword location to receive the File
1 !                  type's LANGUAGE code.
1 !
1 !       RECTYPEID - The address of a longword location to receive
1 !                   the typeid for the File's record type.
1 !
1 ! OUTPUTS
1 !       LANGUAGE - The File type's language code.
1 !
1 !       RECTYPEID - The File's record-type typeid is returned to RECTYPEID.
1 !
1 !       No value is returned.
1 !
1
2     BEGIN
2
2     MAP
2         TYPEID: REF RST$ENTRY;            ! The input TYPEID of the file type
2
2     LOCAL
2         DST_PTR: REF DST$RECORD,
2         TYPE_SPEC: REF DST$TYPE_SPEC,
2         TYPEID_PTR: REF RST$ENTRY,
2         SIZE,
2         FCODE;
2
2
2
2     TYPEID_PTR = FIND_TYPE_RECORD(.TYPEID, FCODE, SIZE);
2     IF .FCODE NEQ RST$K_TYPE_FILE THEN $DBG_ERROR('RSTTYPES\TYP_FILE');
2     DST_PTR = .TYPEID_PTR[RST$L_DSTPTR];
2     TYPE_SPEC = DST_PTR[DST$A_TYPSPEC_TS_ADDR] + .DST_PTR[DST$B_TYPSPEC_NAME];
2     TYPE_SPEC = FIND_TYPESPEC(.TYPE_SPEC, SIZE, DST_PTR);
2     .LANGUAGE = .TYPE_SPEC[DST$B_TS_FILE_LANG];
2     IF .TYPE_SPEC[DST$W_TS_LENGTH] GTR 2
2     THEN
3         BEGIN
3         TYPE_SPEC = TYPE_SPEC[DST$A_TS_FILE_RCRD_TYP];
3         .RECTYPEID = FIND_TYPREC_FROM_TSPEC(.TYPE_SPEC);
3         END
3
2     ELSE
2         .RECTYPEID = 0;
2
2     RETURN;
2
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$STA_TYP_FILE, Save R2                   ; 1008
                        5E            0C   C2  00002         SUBL2   #12, SP                                     ;
                              08      AE   9F  00005         PUSHAB  SIZE                                        ; 1046
                              04      AE   9F  00008         PUSHAB  FCODE                                       ;
                              04      AC   DD  0000B         PUSHL   TYPEID                                      ;
                  0000V  CF           03   FB  0000E         CALLS   #3, FIND_TYPE_RECORD                        ;
                         52           50   D0  00013         MOVL    R0, TYPEID_PTR                              ;
                         0F           6E   D1  00016         CMPL    FCODE, #15                                  ; 1047
                                      15   13  00019         BEQL    1$                                          ;
                          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
```

```
GLOBAL ROUTINE DBG$STA_TYP_RECORD(TYPEID, NCOMPS, COMPVECPTR, BITSIZE): NOVALUE =
!
! FUNCTION
!       This routine accepts as input a Type ID for a record (or "structure")
!       data type, and returns the number of record components, a vector of
!       record component SYMIDs, and the bit length of the record as output.
!       The record component SYMIDs can then be used as input to other symbol
!       table access routines to extract the component names, types, and values.
!
! INPUTS
!       TYPEID  - The Type ID of the Record data type whose component informa-
!                 tion is to be returned.  The Format Code of this data type
!                 must be RST$K_TYPE_RECORD.
!
!       NCOMPS  - The address of a longword location to receive the number of
!                 record components associated with the specified data type.
!
!       COMPVECPTR - The address of a longword location to receive a pointer to
!                 a vector of component information.
!
!       BITSIZE - The address of a longword location to receive the bit length
!                 of the record.
!
! OUTPUTS
!       NCOMPS  - The number of record components is returned to NCOMPS.
!
!       COMPVECPTR - A pointer to a vector of record component SYMIDs is re-
!                 turned to COMPVECPTR.  There is one SYMID for each record
!                 component in the vector; this SYMID can then be used to
!                 extract the component's data type, name, or value (offset
!                 from the start of the record).  This vector disappears at
!                 the end of the current Debug command.
!
!       BITSIZE - The record's bit length is returned to BITSIZE.
!
!       No value is returned.
!

    BEGIN

    MAP
        TYPEID: REF RST$ENTRY;          ! The input TYPEID of the record type

    LOCAL
        TYPEID_PTR: REF RST$ENTRY,
        FCODE,
        SIZE;



    TYPEID_PTR = FIND_TYPE_RECORD(.TYPEID, FCODE, SIZE);
    IF .FCODE NEQ RST$K_TYPE_RECORD THEN $DBG_ERROR('RSTTYPES\TYP_RECORD');
    .NCOMPS = .TYPEID_PTR[RST$L_TYPCOMPCNT];
    .COMPVECPTR = TYPEID_PTR[RST$A_TYPCOMPLST];
    .BITSIZE = .SIZE;
    RETURN;
```

```
; 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
```

```
1 GLOBAL ROUTINE DBG$STA_TYP_SET(TYPEID, PARENT_TYPE, BITSIZE, OFFSET): NOVALUE =
1 !
1 ! FUNCTION
1 !       This routine accepts as input the Type ID of a Set data type (as in
1 !       Pascal, for example), and returns the set's length and parent data
1 !       type as output.  A set is always assumed to consist of a bit string
1 !       where each bit represents one member of the parent type.  The parent
1 !       data type, i.e. the data type of the set elements, can be any ordinal
1 !       type (such as integer, enumeration type, character, etc.).
1 !
1 ! INPUTS
1 !       TYPEID  - The Type ID of the Set type whose attributes are to be
1 !                 returned.  Its Format Code must be RST$K_TYPE_SET.
1 !
1 !       PARENT_TYPE - The address of a longword location to receive the Type ID
1 !                 of the set's parent type.
1 !
1 !       BITSIZE - The address of a longword location to receive the length of
1 !                 the set.
1 !
1 !       OFFSET  - (optional parameter) The address of a longword location to
1 !                 receive the virtual bit index of the first allocated bit
1 !
1 ! OUTPUTS
1 !       PARENT_TYPE - The Type ID of the set's parent type (i.e., the data type
1 !                 of the set elements) is returned to PARENT_TYPE.
1 !
1 !       BITSIZE - The length of the set type (in bits used to represent the
1 !                 set) is returned to BITSIZE.
1 !
1 !       OFFSET  - If this optional parameter was specified, the bit number of
1 !                 the first allocated bit is returned to OFFSET.  Currently
1 !                 all SETs are allocated starting at bit zero - this parameter
1 !                 is provided for future expansion.
1 !
1 !       No value is returned.
1 !
1
2     BEGIN
2
2     BUILTIN
2         ACTUALCOUNT;
2
2     MAP
2         TYPEID : REF RST$ENTRY;         ! The input TYPEID of the set type
2
2     LOCAL
2         FCODE,
2         TYPEID_PTR: REF RST$ENTRY,
2         DST_PTR: REF DST$RECORD,
2         SIZE,
2         SIZETMP,
2         TYPE_SPEC: REF DST$TYPE_SPEC;
2
2
2
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
```

```
1 GLOBAL ROUTINE DBG$STA_TYP_SUBRNG(TYPEID, PARENT_TYPE, LOWPTR, HIGHPTR,
1                                           BITSIZE): NOVALUE =
1 !
1 ! FUNCTION
1 !       This routine accepts the Type ID of a Subrange data type (as in Pascal,
1 !       for instance) as input, and returns that type's parent type, subrange
1 !       interval, and bit length.  The parent type can be any data type with
1 !       values of reasonable size (e.g., anything that can be represented as an
1 !       integer or real value), and the subrange interval is returned as
1 !       pointers to the lower bound value and the upper bound value.  Both
1 !       values are assumed to be of the parent type.
1 !
1 ! INPUTS
1 !       TYPEID  - The Type ID of the Subrange data type whose attributes are to
1 !                 be returned.  Its Format Code must be RST$K_TYPE_SUBRNG.
1 !
1 !       PARENT_TYPE - The address of a longword location to receive the returned
1 !                 Type ID of subrange's parent type.
1 !
1 !       LOWPTR  - The address of a longword location to receive a pointer to the
1 !                 lower bound value of the subrange.
1 !
1 !       HIGHPTR - The address of a longword location to receive a pointer to the
1 !                 upper bound value of the subrange.
1 !
1 !       BITSIZE - The address of a longword location to receive the length in bits
1 !                 of an item of this type.
1 !
1 ! OUTPUTS
1 !       PARENT_TYPE - The Type ID of the subrange's parent type (i.e., the data
1 !                 type of which this is a subrange) is returned to PARENT_TYPE.
1 !
1 !       LOWPTR  - A pointer to the lower bound value of the subrange is returned
1 !                 to LOWPTR.  This value must be interpreted as being of the
1 !                 parent type.  The value disappears at the end of the current
1 !                 Debug command.
1 !
1 !       HIGHPTR - A pointer to the upper bound value of the subrange is returned
1 !                 to HIGHPTR.  This value must be intepreted as being of the
1 !                 parent type.  The value disappears at the end of the current
1 !                 Debug command.
1 !
1 !       BITSIZE - The length in bits is returned to BITSIZE.
1 !
1 !       No value is returned.
1 !
1
2     BEGIN
2
2     MAP
2         TYPEID: REF RST$ENTRY;          ! The input TYPEID of the subrange type
2
2     LOCAL
2         SIZE,
2         SIZETMP,
2         TYPEID_PTR: REF RST$ENTRY,
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

                                0000C 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
```

```
GLOBAL ROUTINE DBG$STA_TYP_OFFSET(TYPEID, AREA_ADDR, AREA_LEN): NOVALUE =
!
! FUNCTION
!       This routine accepts as input the Type ID of an Offset data type and
!       it returns the address and length of the associated Area.  Areas and
!       Offsets are PL/I data types.
!
! INPUTS
!       TYPEID  - The Type ID of the Offset data type whose attributes are to be
!                 returned.  Its Format Code must be RST$K_TYPE_OFFSET.
!
!       AREA_ADDR - The address of a longword location to receive the byte
!                 address of the Area associated with this Offset.
!
!       AREA_LEN - The address of a longword location to receive the byte
!                 length of the Area associated with this Offset.
!
! OUTPUTS
!       AREA_ADDR - The byte address of the Area associated with this Offset
!                 is returned to AREA_ADDR.
!
!       AREA_LEN - The byte length of the Area associated with this Offset
!                 is returned to AREA_LEN.
!
!       No routine value is returned.
!
!
    BEGIN

    MAP
        TYPEID: REF RST$ENTRY,              ! Pointer to the input Type RST Entry
        AREA_ADDR: REF VECTOR[1],           ! Address where we return Area address
        AREA_LEN: REF VECTOR[1];            ! Address where we return Area length

    LOCAL
        DSTPTR: REF DST$RECORD,             ! Pointer to Offset Type Spec DST record
        FCODE,                              ! The Format Code for this data type
        SIZE,                               ! The bit size of this data object
        TSPTR: REF DST$TYPE_SPEC,           ! Pointer to DST Type Spec for Offset
        TYPEPTR: REF RST$ENTRY,             ! Pointer to the offset Type RST Entry
        VALKIND,                            ! The kind of value returned by the
                                            !       DBG$STA_VALSPEC routine
        VALPTR: REF VECTOR[,LONG],          ! Pointer to the Area length value
        VALVECTOR: VECTOR[3,LONG],          ! Three-longword value vector returned
                                            !       by the DBG$STA_VALSPEC routine
        VSPTR: REF DST$VAL_SPEC;            ! Pointer to Value Specs in the DST Type
                                            !       Spec for Offset



    ! In case the caller passed in a SYMID for the Offset symbol instead of its
    ! Type ID, we get the symbol's actual Type ID here.  We also make sure we
    ! really have an Offset data type.
    !
    TYPEPTR = FIND_TYPE_RECORD(.TYPEID, FCODE, SIZE);
    IF .FCODE NEQ RST$K_TYPE_OFFSET
    THEN
```

```
2          $DBG_ERROR('RSTTYPES\TYP_OFFSET');
2
2
2      ! Get a pointer to the Type Spec DST Record and get a pointer to the actual
2      ! Type Spec within it (possibly after Type Spec indirection).
2      !
2      DSTPTR = .TYPEPTR[RST$L_DSTPTR];
2      TSPTR = DSTPTR[DST$A_TYPSPEC_TS_ADDR] + .DSTPTR[DST$B_TYPSPEC_NAME];
2      TSPTR = FIND_TYPESPEC(.TSPTR, SIZE, DSTPTR);
2
2
2      ! Evaluate the first Value Spec in the Offset Type Spec.  Return the result
2      ! (i.e., the address of the Area) to the AREA_ADDR parameter.
2      !
2      VSPTR = TSPTR[DST$A_TS_OFFSET_VALSPEC];
2      DBG$STA_VALSPEC(.VSPTR, VALVECTOR, VALKIND);
2      AREA_ADDR[0] = .VALVECTOR[0];
2
2
2      ! Get to the second Value Spec in the Offset Type Spec and evaluate it.
2      ! Return its result (i.e., the length of the Area) to AREA_LEN.
2      !
2      IF .VSPTR[DST$B_VS_VFLAGS] EQL DST$K_VS_FOLLOWS
2      THEN
2          VSPTR = VSPTR[DST$B_VS_ALLOC] + .VSPTR[DST$W_VS_LENGTH]
2
2      ELSE
2          VSPTR = .VSPTR + 5;
2
2      DBG$STA_VALSPEC(.VSPTR, VALVECTOR, VALKIND);
2      VALPTR = .VALVECTOR[0];
2      AREA_LEN[0] = .VALPTR[0];
2
2
2      ! We are all done--return to the caller.
2      !
2      RETURN;
2
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.AA0:  .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
```

```
GLOBAL ROUTINE DBG$STA_TYP_AREA(TYPEID, BYTE_LEN): NOVALUE =
!
! FUNCTION
!       This routine accepts as input the Type ID of an AREA data type
!       and returns the byte length of the AREA.
!
! INPUTS
!       TYPEID   - The Type ID of the AREA data type whose attributes are to be
!                  returned.  Its Format Code must be RST$K_TYPE_AREA.
!
!       BYTE_LEN - The address of a longword location to receive a pointer
!                  to the byte length of the AREA.
!
! OUTPUTS
!       BYTE_LEN - A pointer to the byte length of the AREA is returned to
!                  BYTE_LEN
!
!
!       No value is returned.
!
    BEGIN

    MAP
        TYPEID: REF RST$ENTRY;          ! The input TYPEID of the PL/I AREA type

    LOCAL
        TYPEID_PTR: REF RST$ENTRY,
        DST_PTR: REF DST$RECORD,
        TYPE_SPEC: REF DST$TYPE_SPEC,
        VAL_SPEC: REF DST$VAL_SPEC,
        VALKIND,
        FCODE,
        VAL_VECT: VECTOR[3],
        SIZE;


    TYPEID_PTR = FIND_TYPE_RECORD(.TYPEID, FCODE, SIZE);
    IF .FCODE NEQ RST$K_TYPE_AREA THEN $DBG_ERROR('RSTTYPES\TYP_AREA');
    DST_PTR = .TYPEID_PTR[RST$L_DSTPTR];
    TYPE_SPEC = DST_PTR[DST$A_TYPSPEC_TS_ADDR] + .DST_PTR[DST$B_TYPSPEC_NAME];
    TYPE_SPEC = FIND_TYPESPEC(.TYPE_SPEC, SIZE, DST_PTR);
    IF .SIZE EQL 0
    THEN
        BEGIN
        VAL_SPEC = TYPE_SPEC[DST$A_TS_AREA_BYTE_LEN];
        DBG$STA_VALSPEC(.VAL_SPEC, VAL_VECT, VALKIND);
        .BYTE_LEN = .VAL_VECT[0];
        END

    ELSE
        .BYTE_LEN = .SIZE;

    RETURN;

    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  0013B                                                       ;

                                                              .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
```

```
1 GLOBAL ROUTINE DBG$STA_TYP_VARIANT(TYPEID, NCOMPS, COMPVECPTR, TAG,
1                                                  BITSIZE): NOVALUE =
1 !
1 ! FUNCTION
1 !       This routine accepts as input the Type ID of a Variant data type or the
1 !       SYMID of a symbol of variant type, and it returns the number of variant
1 !       components, the SYMID of each such component, and the SYMID of the vari-
1 !       ant's tag field.
1 !
1 ! INPUTS
1 !       TYPEID  - The Type ID of the Variant data type whose attributes and var-
1 !                 iants are to be returned.  The Kind of this data type
1 !                 must be RST$K_VARIANT.
1 !
1 !       NCOMPS  - The address of a longword location to receive the number of
1 !                 variant components of the TYPEID data type.
1 !
1 !       COMPVECPTR - The address of a longword location to receive a pointer to
1 !                 a vector of variant component SYMIDs and tag values.
1 !
1 !       TAG     - The address of a longword location to receive the SYMID of the
1 !                 Variant data type's tag field.
1 !
1 !       BITSIZE - The address of a longword location to receive the size in bits
1 !                 of an item of this type.
1 !
1 ! OUTPUTS
1 !       NCOMPS  - The number of components of the Variant data type (the number
1 !                 of different "variants") is returned to NCOMPS.
1 !
1 !       COMPVECPTR - A pointer to a vector of variant component information is
1 !                 returned to COMPVECPTR.  In this vector, there is a two-long-
1 !                 word entry for each variant component; each such entry gives
1 !                 the SYMID and a pointer to the tag value, respectively, of
1 !                 the corresponding component.  The SYMID can then be used to
1 !                 extract the component's name, data type, and value.  The tag
1 !                 value must be intepreted as a value of tag field's data type.
1 !                 This vector disappears at the end of the current command.
1 !
1 !       TAG     - The SYMID of the variant tag field is returned to TAG.  This
1 !                 SYMID can be used to extract the tag field name, data type,
1 !                 and value.
1 !
1 !       BITSIZE - The size in bits of an item of this type is returned to
1 !                 BITSIZE.
1 !
1 !       No value is returned.
1 !
1
2     BEGIN
2
2     MAP
2       TYPEID : REF RST$ENTRY;           ! Pointer to Variant RST Entry
2
2     LOCAL
2       DSTPTR: REF DST$RECORD,
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_VARSETCNT];
; 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
```

```
GLOBAL ROUTINE DBG$STA_TYP_VARIANT_COMP(LIST_ID, NCOMPS, COMPVECPTR,
                                        NTAGS, TAGVECPTR, SIZE): NOVALUE =
!
! FUNCTION
!       This routine accepts as input a pointer to a set of Variant fields,
!       and it returns the number of fields, the SYMID for each field, the number
!       of tag blocks, a value range for each tag block, and the bit length of
!       the variant.
!
! INPUTS
!       LIST_ID - A pointer to a list of variant fields; this list was produced
!                 by DBG$STA_TYP_VARIANT and was pointed to by one element of
!                 the COMPVECPTR it returned.
!
!       NCOMPS  - The address of a longword location to receive the number of
!                 fields in this variant.
!
!       COMPVECPTR - The address of a longword location to receive a pointer to
!                 a vector of field SYMIDs.
!
!       NTAGS   - The address of a longword location to receive the number of
!                 tag blocks.
!
!       TAGVECPTR - The address of a longword location to receive a pointer to a
!                   vector of tag value ranges. An element of this vector is the
!                   address of a three longword block; each block has the following
!                   format:
!                       longword 1: dst$k_varval_single = single tag value
!                                                     or
!                                   dst$k_varval_range = range of tag values
!                       longword 2: pointer to low bound value of tag range
!                       longword 3: pointer to high bound value of tag range
!                                   (or zero if single value)
!
!       SIZE    - The address of a longword location to receive the size in bits
!                 of an item of this type.
!
! OUTPUTS
!       NCOMPS  - The number of fields is returned to NCOMPS.
!
!       COMPVECPTR - A pointer to a vector of field SYMIDs is returned to
!                 COMPVECPTR.
!                 This vector disappears at the end of the current command.
!
!       NTAGS   - The number of tag blocks is returned to NTAGS.
!
!       SIZE    - The size in bits of this variant is returned to SIZE.
!
!       No value is returned.
!

    BEGIN

    MAP
        LIST_ID: REF RST$VAR_ENTRY;     ! Pointer to Variant Component List Entry

    LOCAL
```

```
2        TAG_LIST: REF RST$TAG_LIST,
2        TAG_PTR: REF VECTOR[,BYTE],
2        DST_PTR: REF DST$RECORD,
2        VALUE_SPEC: REF DST$VAL_SPEC,
2        VAL_LNGTH,
2        VAL_VECT: VECTOR[3],
2        VALKIND;
2
2
2
2    DST_PTR = .LIST_ID[RST$L_VAR_DSTPTR];
2    IF .DST_PTR[DST$B_TYPE] NEQ DST$K_VARVAL
2    THEN
2        $DBG_ERROR('RSTTYPES\VARIANT_COMP');
2
2    .NCOMPS = .LIST_ID[RST$L_VAR_COMPCNT];
2    .NTAGS = .DST_PTR[DST$W_VARVAL_COUNT];
2    .SIZE = .DST_PTR[DST$L_VARVAL_SIZE];
2    IF ..NCOMPS NEQ 0
2    THEN
2        .COMPVECPTR = LIST_ID[RST$A_VAR_COMPLST]
2    ELSE
2        .COMPVECPTR = 0;
2
2    TAG_LIST = DBG$GET_TEMPMEM(RST$K_TAG_BLOCK_SIZE * ..NTAGS);
2    TAG_PTR = DST_PTR[DST$A_VARVAL_RNGSPEC];
2    INCR I FROM 0 TO ..NTAGS - 1 DO
3        BEGIN
3        VALUE_SPEC = TAG_PTR[1];
3        TAG_LIST[.I,RST$L_TAG_NUMVALS] = .TAG_PTR[0];
3        DBG$STA_VALSPEC(.VALUE_SPEC, VAL_VECT, VALKIND);
3        TAG_LIST[.I,RST$L_TAG_LOWBOUND] = .VAL_VECT[0];
3        VAL_LNGTH = 5;
3        IF .VALUE_SPEC[DST$B_VS_VFLAGS] EQL DST$K_VS_FOLLOWS
3        THEN
3            VAL_LNGTH = .VALUE_SPEC[DST$W_VS_LENGTH] + 3;
3
3        IF .TAG_PTR[0] EQL DST$K_VARVAL_RANGE
3        THEN
4            BEGIN
4            VALUE_SPEC = .TAG_PTR + .VAL_LNGTH + 1;
4            DBG$STA_VALSPEC(.VALUE_SPEC, VAL_VECT, VALKIND);
4            TAG_LIST[.I,RST$L_TAG_HIGHBOUND] = .VAL_VECT[0];
4            VAL_LNGTH = 5;
4            IF .VALUE_SPEC[DST$B_VS_VFLAGS] EQL DST$K_VS_FOLLOWS
4            THEN
4                VAL_LNGTH = .VALUE_SPEC[DST$W_VS_LENGTH] + 3;
4
4            TAG_PTR = .TAG_PTR + .VAL_LNGTH;
4            END
4
3        ELSE
3            TAG_PTR = .TAG_PTR + .VAL_LNGTH + 1;
3
2        END;
2
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$STA_VALSPEC            ;
                      08 A645  9F 000AC       PUSHAB  8(R6)[TAG_LIST]                ; 1603
                  9E  08   AE  D0 000B0       MOVL    VAL_VECT, @(SP)+               ;
                  57       05  D0 000B4       MOVL    #5, VAL_LNGTH                  ; 1604
              FD  8F       62  91 000B7       CMPB    (VALUE_SPEC), #253             ; 1605
                           07  12 000BB       BNEQ    6$                             ;
                  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

```
1 GLOBAL ROUTINE DBG$TRANS_TYPE_CODE(CODE) =
1 !
1 ! FUNCTION
1 !       This routine takes a DST type code and translates it to the
1 !       corresponding RST type code.
1 !
1 ! INPUTS
1 !       CODE    - The DST type code to be translated.
1 !
1 ! OUTPUTS
1 !       The corresponding RST type code is returned as the routine's value.
1 !
1
2     BEGIN
2
2     CASE .CODE FROM DST$K_TS_DTYPE_LOWEST TO DST$K_TS_DTYPE_HIGHEST OF
2         SET
2
2         [DST$K_TS_ATOM]:
2             RETURN RST$K_TYPE_ATOMIC;
2
2         [DST$K_TS_DSC]:
2             RETURN RST$K_TYPE_DESCR;
2
2         [DST$K_TS_TPTR]:
2             RETURN RST$K_TYPE_TPTR;
2
2         [DST$K_TS_PTR]:
2             RETURN RST$K_TYPE_PTR;
2
2         [DST$K_TS_PIC]:
2             RETURN RST$K_TYPE_PICT;
2
2         [DST$K_TS_ARRAY]:
2             RETURN RST$K_TYPE_ARRAY;
2
2         [DST$K_TS_SET]:
2             RETURN RST$K_TYPE_SET;
2
2         [DST$K_TS_SUBRANGE]:
2             RETURN RST$K_TYPE_SUBRNG;
2
2         [DST$K_TS_FILE]:
2             RETURN RST$K_TYPE_FILE;
2
2         [DST$K_TS_SELF_REL_LABEL]:
2             RETURN RST$K_TYPE_SELF_REL_LAB;
2
2         [DST$K_TS_RFA]:
2             RETURN RST$K_TYPE_RFA;
2
2         [DST$K_TS_AREA]:
2             RETURN RST$K_TYPE_AREA;
2
2         [DST$K_TS_OFFSET]:
2             RETURN RST$K_TYPE_OFFSET;
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$-1$,-                                    ;
      004B      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
```

```
1 GLOBAL ROUTINE DBG$TYPEID_FOR_ARRAY(TYPEID, ADDRESS) =
1 !
1 ! FUNCTION
1 !       This routine is called from DBGPARSER in the process of parsing
1 !       a C expression of the form PTR[n], where PTR was declared as
1 !       a pointer. Since arrays and pointers are treated the same in C,
1 !       we want to convert the pointer typeid to an equivalent array
1 !       typeid.
1 !
1 ! INPUT
1 !       TYPEID  - The RST entry describing a pointer data type
1 !       ADDRESS - The value of the pointer (which will be the base
1 !                 address of the array).
1 !
1 ! OUTPUT
1 !       A new typeid is constructed out of temporary memory, and
1 !       a pointer to this typeid is returned.
1 !
1 ! Example:
1 !       REAL *RPTR;
1 !       If RPTR is declared as a pointer to a floating number,
1 !       then if we pass in the TYPEID for RPTR, this routine returns
1 !       a typeid for an array of floating numbers (e.g., as if we
1 !       had declared an object
1 !       REAL RARRAY[n];
1 !
2     BEGIN
2     MAP
2       TYPEID: REF RST$ENTRY;            ! The input TYPEID for the Type Pointer
2                                         !         data type
2
2     LOCAL
2       ADDRPTR,                          ! Pointer into array descriptor
2       ARG_DSTPTR: REF DST$RECORD,       ! Pointer to DST record for argument
2       ARG_TYPEID: REF RST$ENTRY,        ! Pointer to RST record for pointed-to
2                                         !     object.
2       BITSIZE,                          ! Bitsize of pointed-to object
2       BOUNDVEC: REF VECTOR[],           ! Pointer to array bounds
2       DSCADDR: REF BLOCK[,BYTE],        ! Pointer to array descriptor
2       DSTPTR: REF DST$RECORD,           ! Pointer to dummy DST record
2       FLAGS_PTR: REF BITVECTOR[],       ! "flags" field withing array type spec
2       RSTPTR: REF RST$ENTRY,            ! Pointer to new RST record
2       TSPTR: REF DST$TYPE_SPEC,         ! Pointer to type spec within DST record
2       TSPTR1: REF DST$TYPE_SPEC,        ! Pointer to type spec within DST record
2       TSPTR2: REF DST$TYPE_SPEC,        ! Pointer to type spec within DST record
2       VSPTR: REF DST$VAL_SPEC;          ! Pointer to val spec within DST record
2
2
2     ! Obtain a TYPEID and a DSTPTR for the pointed-to object.
2     !
2     DBG$STA_TYP_TYPEDPTR (.TYPEID, ARG_TYPEID);
2     DBG$STA_SYMSIZE (.ARG_TYPEID, BITSIZE);
2     ARG_DSTPTR = .ARG_TYPEID[RST$L_DSTPTR];
2
2     ! Allocate a memory block large enough to hold the newly created
2     ! RST record, the dummy DST record that it will point to, and the
2     ! array descriptor for this dummy array we are creating.
```

```
!
RSTPTR = DBG$GET_MEMORY (RST$K_TYPENTSIZ + 14);
DSTPTR = .RSTPTR + 4*RST$K_TYPENTSIZ;

! Fill in some of the fields of the RST record.
!
RSTPTR[RST$L_DSTPTR] = .DSTPTR;
RSTPTR[RST$L_UPSCOPEPTR] = .RST$START_ADDR;
RSTPTR[RST$B_KIND] = RST$K_TYPE;
RSTPTR[RST$B_FCODE] = RST$K_TYPE_ARRAY;

! Put the RST record on the temporary RST list. This will ensure that
! the space gets released when there are no more references to it.
!
RSTPTR[RST$L_HASH_FLINK] = .RST$TEMP_LIST;
RST$TEMP_LIST = .RSTPTR;

! Fill in the header of the dummy DST record.
!
DSTPTR[DST$B_LENGTH] = 23;
DSTPTR[DST$B_TYPE] = DST$K_TYPSPEC;
DSTPTR[DST$B_TYPSPEC_NAME] = 0;

! Fill in the Type Spec record within the dummy DST record.
! This describes the array as a whole.
!
TSPTR = .DSTPTR + 3;
TSPTR[DST$W_TS_LENGTH] = 19;
TSPTR[DST$B_TS_KIND] = DST$K_TS_ARRAY;
TSPTR[DST$B_TS_ARRAY_DIM] = 1;
FLAGS_PTR = TSPTR[DST$A_TS_ARRAY_FLAGS_ADDR];
FLAGS_PTR[0] = TRUE;

! Fill in the Val Spec record within the dummy DST record.
! This points to the array descriptor that we will construct
! below this dummy DST record.
!
VSPTR = .FLAGS_PTR + 1;
VSPTR[DST$B_VS_VFLAGS] = DST$K_VFLAGS_DSC;
VSPTR[DST$L_VS_DSC_OFFS] = 11;

! Fill in the celltype type spec record within the dummy DST record.
! This describes the type of each element of the array. This type
! should be the same as the type of the pointed-to object, so
! we use an indirect type specification, and point to the DST
! we obtained from the argument.
!
TSPTR1 = .VSPTR + 5;
TSPTR1[DST$W_TS_LENGTH] = 5;
TSPTR1[DST$B_TS_KIND] = DST$K_TS_IND;
TSPTR1[DST$L_TS_IND_PTR] = DBG$RST_DST_PTR( .ARG_DSTPTR - .DST$BEGIN_ADDR );


! Fill in the type specification for the type of the subscript.
! This type will just be longword integer.
!
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$TEMP_LIST, (RSTPTR)                  ; 1758
       65             53    D0 0004F    MOVL    RSTPTR, RST$TEMP_LIST                    ; 1759
       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 0008B        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
```

```
1 GLOBAL ROUTINE DBG$TYPEID_FOR_ATOMIC(TYPECODE, BITSIZE, DSC_FLAG) =
1 !
1 ! FUNCTION
1 !       This routine returns the Type ID associated with a specified VAX stand-
1 !       ard type code (a "one-byte type code").  The Type ID is a pointer to an
1 !       RST entry for the data type.  This RST entry is added to the Temporary
1 !       RST Entry List so that it disappears as soon as there are no longer any
1 !       references to it.
1 !
1 !       The routine builds a Data Type RST Entry and a Type Specification DST
1 !       record, both in the same memory block.  The RST entry's FCODE is set to
1 !       be atomic and its DST pointer points to the DST record.  The DST record
1 !       is built to contain a DST Type Specification for the specified atomic
1 !       data type.
1 !
1 ! INPUTS
1 !       TYPECODE - The VAX standard type code ("one-byte" type code) whose Type
1 !                  ID is to be returned.
1 !
1 !       BITSIZE  - The size in bits of an item of this type.
1 !
1 !       DSC_FLAG - A flag which, when set, indicates that we are processing a
1 !                  descriptor item; thus, we will signal INVALID ARRAY DESC
1 !                  in certain error situations.
1 !
1 ! OUTPUTS
1 !       The corresponding Type ID (Type RST Entry pointer) is returned as the
1 !                  routine value.
1 !
1
2    BEGIN
2
2    LOCAL
2       DSTPTR: REF DST$RECORD,          ! Pointer to the generated DST record
2       RSTPTR: REF RST$ENTRY,           ! Pointer to the Type RST Entry
2       TSPTR: REF DST$TYPE_SPEC;        ! Pointer to Type Spec in DST record
2
2
2
2    ! Make sure the VAX standard type code is in the valid range.
2    !
3    IF ((.TYPECODE LSS DSC$K_DTYPE_LOWEST) OR
2       (.TYPECODE GTR DSC$K_DTYPE_HIGHEST)) AND
2      (.TYPECODE NEQ DST$K_BOOL) AND
3      (.TYPECODE NEQ DSC$K_DTYPE_TF)
2    THEN
3       BEGIN
3       IF .DSC_FLAG
3       THEN
3           SIGNAL(DBG$_INVARRDSC)
3
3       ELSE
3           $DBG_ERROR('RSTTYPES\TYPEID_FOR_ATOMIC');
3
2       END;
2
2
```

```
2       ! Get a memory block large enough to contain both the RST entry and the
2       ! dummy DST record.
2       !
2       RSTPTR = DBG$GET_MEMORY(RST$K_TYPENTSIZ + 2);
2       DSTPTR = .RSTPTR + 4*RST$K_TYPENTSIZ;
2
2
2       ! Build the Data Type RST Entry for the atomic type.
2       !
2       RSTPTR[RST$L_DSTPTR] = .DSTPTR;
2       RSTPTR[RST$L_UPSCOPEPTR] = .RST$START_ADDR;
2       RSTPTR[RST$B_KIND] = RST$K_TYPE;
2       RSTPTR[RST$B_FCODE] = RST$K_TYPE_ATOMIC;
2       RSTPTR[RST$L_BITSIZE] = .BITSIZE;
2
2
2       ! Put this Data Type RST Entry on the Temporary RST Entry List.  This will
2       ! cause the entry to be released at the end of a DEBUG command when there
2       ! are no references to it (when it is not locked).
2       !
2       RSTPTR[RST$L_HASH_FLINK] = .RST$TEMP_LIST;
2       RST$TEMP_LIST = .RSTPTR;
2
2
2       ! Build the dummy Type Specification DST entry for the atomic type.
2       !
2       DSTPTR[DST$B_LENGTH] = 6;
2       DSTPTR[DST$B_TYPE] = DST$K_TYPSPEC;
2       DSTPTR[DST$B_TYPSPEC_NAME] = 0;
2       TSPTR = .DSTPTR + 3;
2       TSPTR[DST$W_TS_LENGTH] = 2;
2       TSPTR[DST$B_TS_KIND] = DST$K_TS_ATOM;
2       TSPTR[DST$B_TS_ATOM_TYP] = .TYPECODE;
2
2
2       ! Return the type's Type ID (Data Type RST Entry address) to the caller.
2       !
2       RETURN .RSTPTR;
2
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
```

```
GLOBAL ROUTINE DBG$TYPEID_FOR_SET(ARG1, FCODE, BITSIZE) =
!
! FUNCTION
!       This routine builds a Data Type RST Entry for a set data type.
!       This routine is normally called when a Type ID (i.e., an RST
!       pointer to a Type RST Entry) must be produced for a set constant
!       data type.  The RST entry is added to the Temporary RST Entry List so
!       that it is released back to the memory pool once there are no longer
!       any references to it.
!
!       The routine builds a Data Type RST Entry and a dummy DST record in the
!       same memory block.  The DST record is a Type Specification DST record
!       with the null name and an Set Type Specification.  Parent Type
!       Specification in the Set Type Specification can be either an atomic
!       type or an Indirect Type Specification pointing to the DST Type
!       Specification given as the input parameter.
!
! INPUTS
!       ARG1    - DTYPE or DSTADDR
!                 Dtype is vax standard data type.
!                 Dstaddr is the address of the DST Parent Type Specification
!                 for which a Set Type ID is to be generated.
!
!       FCODE   - The type Format Code associated with the Type Specification.
!
!       BITSIZE - The size in bits of an item of this type.
!
!       Optional 4th argument - If presents, that means the 1st argument is
!                               passed in as Dtype.  An Atomic parent type
!                               specification will be generated.
!
! OUTPUTS
!       The data type's Type ID (i.e., a pointer to the Type RST Entry) is
!                 returned as the routine's value.
!

    BEGIN

    BUILTIN
        ACTUALCOUNT;

    LOCAL
        DSTADDR,                        ! Parent type DST pointer
        DTYPE: BYTE,                    ! Data type
        DSTPTR: REF DST$RECORD,         ! Pointer to generated DST record
        PARENT_TSPTR: REF DST$TYPE_SPEC,! Pointer to Parent Type Spec in Set
                                        !       DST record
        RSTPTR: REF RST$ENTRY,          ! Pointer to generated Type RST Entry
        TSPTR: REF DST$TYPE_SPEC;       ! Pointer to Type Spec in DST record


    ! Depending on whether we have a fourth argument to this routine or not,
    ! the first argument is either a DTYPE or a DSTADDR.  Pick up the Data
    ! Type or the Parent type DST pointer accordingly.
    !
    IF ACTUALCOUNT() GTR 3 THEN DTYPE = .ARG1 ELSE DSTADDR = .ARG1;
```

```
2
2
2       ! Get a memory block large enough to accommodate both the RST entry and
2       ! the dummy DST record.
2       !
2       RSTPTR = DBG$GET_MEMORY(RST$K_TYPENTSIZ + 5);
2       DSTPTR = .RSTPTR + 4*RST$K_TYPENTSIZ;
2
2
2       ! Build the Data Type RST Entry for the input type.
2       !
2       RSTPTR[RST$L_DSTPTR] = .DSTPTR;
2       RSTPTR[RST$L_UPSCOPEPTR] = .RST$START_ADDR;
2       RSTPTR[RST$B_KIND] = RST$K_TYPE;
2       RSTPTR[RST$B_FCODE] = .FCODE;
2       RSTPTR[RST$L_BITSIZE] = .BITSIZE;
2
2
2       ! Put this Data Type RST Entry on the Temporary RST Entry List.  This will
2       ! cause the entry to be released at the end of a DEBUG command when there
2       ! are no references to it (when it is not locked).
2       !
2       RSTPTR[RST$L_HASH_FLINK] = .RST$TEMP_LIST;
2       RST$TEMP_LIST = .RSTPTR;
2
2
2       ! Build the dummy DST record for the type.
2       !
2       DSTPTR[DST$B_LENGTH] = 4 ;
2       DSTPTR[DST$B_TYPE] = DST$K_TYPSPEC;
2       DSTPTR[DST$B_TYPSPEC_NAME] = 0;
2       TSPTR = .DSTPTR + 3;
2       TSPTR[DST$W_TS_LENGTH] = 5;
2       TSPTR[DST$B_TS_KIND] = DST$K_TS_SET;
2       TSPTR[DST$L_TS_SET_LENG] = .BITSIZE;
2       PARENT_TSPTR = .TSPTR + 7;
2       IF ACTUALCOUNT() GTR 3
2       THEN
3           BEGIN
3           TSPTR[DST$W_TS_LENGTH] = .TSPTR[DST$W_TS_LENGTH] + DST$K_TS_ATOM_LENG;
3           PARENT_TSPTR[DST$W_TS_LENGTH] = 2;
3           PARENT_TSPTR[DST$B_TS_KIND] = DST$K_TS_ATOM;
3           PARENT_TSPTR[DST$B_TS_ATOM_TYP] = .DTYPE;
3           END
3
2       ELSE
3           BEGIN
3           TSPTR[DST$W_TS_LENGTH] = .TSPTR[DST$W_TS_LENGTH] + DST$K_TS_IND_LENG;
3           PARENT_TSPTR[DST$W_TS_LENGTH] = 5;
3           PARENT_TSPTR[DST$B_TS_KIND] = DST$K_TS_IND;
3           PARENT_TSPTR[DST$L_TS_IND_PTR] = DBG$RST_DST_PTR( .DSTADDR - .DST$BEGIN_ADDR );
3           END;
2
2       DSTPTR[DST$B_LENGTH] = .DSTPTR[DST$B_LENGTH] + .TSPTR[DST$W_TS_LENGTH];
2
2
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
```

```
GLOBAL ROUTINE DBG$TYPEID_FOR_TPTR(TYPEID) =
!
! FUNCTION
!       Given a TYPEID describing an arbitrary data type, this routine
!       constructs a TYPEID for a Typed Pointer data type, where the
!       typed pointer points to the given data type.
!
!       This is needed for the & operator in C.  For example, if X is data
!       of type T, then &X returns the address of X, and the type of &X is
!       "pointer to data of type T".  This routine is then used to transform
!       the TYPEID for X into the TYPEID for &X.
!
! INPUTS
!       TYPEID  - A TYPEID for the object to which the & operator is being
!                 applied.
!
! OUTPUTS
!       Return Value: A TYPEID is returned, as described above.
!

    BEGIN

    MAP
        TYPEID: REF RST$ENTRY;              ! The input TYPEID for the Type Pointer
                                            !       data type

    LOCAL
        ARG_DSTPTR: REF DST$RECORD,         ! Pointer to DST record for argument
        ARG_TSPTR: REF DST$TYPE_SPEC,       ! Pointer to Type Spec for argument
        DSTPTR: REF DST$RECORD,             ! Pointer to dummy DST record
        RSTPTR: REF RST$ENTRY,              ! Pointer to new RST record
        TSPTR: REF DST$TYPE_SPEC;           ! Pointer to type spec within DST record


    ! Obtain the DST pointer and Type Spec pointer for the argument.
    !
    ARG_DSTPTR = .TYPEID[RST$L_DSTPTR];
    ARG_TSPTR =  ARG_DSTPTR[DST$A_TYPSPEC_TS_ADDR] +
                .ARG_DSTPTR[DST$B_TYPSPEC_NAME];


    ! Allocate a memory block large enough to hold both the newly created
    ! RST record and the dummy DST record that it will point to.
    !
    RSTPTR = DBG$GET_MEMORY (RST$K_TYPENTSIZ + 4);
    DSTPTR = .RSTPTR + 4*RST$K_TYPENTSIZ;


    ! Fill in some of the fields of the RST record.
    !
    RSTPTR[RST$L_DSTPTR] = .DSTPTR;
    RSTPTR[RST$L_UPSCOPEPTR] = .RST$START_ADDR;
    RSTPTR[RST$B_KIND] = RST$K_TYPE;
    RSTPTR[RST$B_FCODE] = RST$K_TYPE_TPTR;
    RSTPTR[RST$L_BITSIZE] = 32;

```

```
2
2          ! Put the RST record on the temporary RST list. This will ensure that
2          ! the space gets released when there are no more references to it.
2          !
2          RSTPTR[RST$L_HASH_FLINK] = .RST$TEMP_LIST;
2          RST$TEMP_LIST = .RSTPTR;
2
2
2          ! Fill in the dummy DST record.
2          !
2          DSTPTR[DST$B_LENGTH] = 12;
2          DSTPTR[DST$B_TYPE] = DST$K_TYPSPEC;
2          DSTPTR[DST$B_TYPSPEC_NAME] = 0;
2
2
2          ! Fill in the Type Spec record within the dummy DST record.
2          !
2          TSPTR = .DSTPTR + 3;
2          TSPTR[DST$W_TS_LENGTH] = 8;
2          TSPTR[DST$B_TS_KIND] = DST$K_TS_TPTR;
2          TSPTR = TSPTR[DST$A_TS_TPTR_TSPEC_ADDR];
2
2
2          ! Fill in the Type Spec for the pointed-to object.
2          ! Atomic types are filled in in-line, other types use the
2          ! "indirect" Type Spec to point back to to argument DST.
2          !
2          IF .ARG_TSPTR[DST$B_TS_KIND] EQL DST$K_TS_ATOM
2          THEN
3              BEGIN
3              TSPTR[DST$W_TS_LENGTH] = 2;
3              TSPTR[DST$B_TS_KIND] = DST$K_TS_ATOM;
3              TSPTR[DST$B_TS_ATOM_TYP] = .ARG_TSPTR[DST$B_TS_ATOM_TYP];
3              END
3
2          ELSE
3              BEGIN
3              TSPTR[DST$W_TS_LENGTH] = 5;
3              TSPTR[DST$B_TS_KIND] = DST$K_TS_IND;
3              TSPTR[DST$L_TS_IND_PTR] = DBG$RST_DST_PTR( .ARG_DSTPTR - .DST$BEGIN_ADDR );
2              END;
2
2
2          ! Return the pointer to the RST record.
2          !
2          RETURN .RSTPTR;
2
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
```

```
ROUTINE TYPEID_FOR_COB_HACK(CH_DSTPTR) =
!
! FUNCTION
!       This routine builds an appropriate Data Type RST Entry for the data
!       type of an object described by a Cobol Hack DST Record.  It accepts
!       as input a pointer to the DST record and it builds and returns the
!       address of a Data Type RST Entry for the corresponding type.  The
!       memory block which contains the Type RST Entry also contains a dummy
!       Type Spec DST Record for the data type in question.
!
!       Cobol Hack records are obsolete, but are still found in programs com-
!       piled with the Version 1 COBOL compiler.  This routine therefore con-
!       verts such DST records into more "regular" DST structures so that on-
!       ly this routine needs to understand the obsolete Cobol Hack Record.
!
! INPUTS
!       CH_DSTPTR - A pointer to the Cobol Hack DST Record whose data type is
!                   to be extracted.
!
! OUTPUTS
!       This routine returns as its value a pointer to a Data Type RST Entry
!                   for the data type specified in the Cobol Hack DST Record.
!

    BEGIN

    MAP
        CH_DSTPTR: REF DST$RECORD;      ! Pointer to Cobol Hack DST Record

    LOCAL
        BITSIZ,                         ! The bitsize of an atomic data type
        CH_TRLR_PTR: REF DST$CH_TRLR,   ! Pointer to Cobol Hack record trailer
        DESCADDR: REF BLOCK[,BYTE],     ! Pointer to VAX standard descriptor
        DSTPTR: REF DST$RECORD,         ! Pointer to dummy Type Spec DST Record
                                        !       built by this routine
        RSTPTR: REF RST$ENTRY,          ! Pointer to Type RST Entry built by
                                        !       by this routine for data item
        TSPTR: REF DST$TYPE_SPEC,       ! Pointer to Type Spec in Type Spec DST
                                        !       record built by this routine
        VSPTR: REF DST$VAL_SPEC;        ! Pointer to Value Spec in Type Spec
                                        !       DST record built by this routine



    ! If the descriptor address in the DST record is zero, this is an atomic
    ! data type defined by the DST$CH_TYPE field.  Build a Type RST Entry for
    ! it and return that entry's address as a TYPEID.
    !
    IF .CH_DSTPTR[DST$L_VALUE] EQL 0
    THEN
        BEGIN
        CH_TRLR_PTR = CH_DSTPTR[DST$A_COBHACK_TRLR] + .CH_DSTPTR[DST$B_NAME];
        BITSIZ = GET_BITSIZE_FROM_DTYPE(.CH_TRLR_PTR[DST$B_CH_TYPE]);
        RSTPTR = DBG$TYPEID_FOR_ATOMIC(.CH_TRLR_PTR[DST$B_CH_TYPE], .BITSIZ, FALSE);
        RETURN .RSTPTR;
        END;
```

```
	! We have a descriptor-specified data type.  Compute the address of the
	! descriptor.
	!
	IF .CH_DSTPTR[DST$B_VFLAGS] EQL DST$K_VFLAGS_DSC
	THEN
	    DESCADDR = CH_DSTPTR[DST$A_DSC_BASE] + .CH_DSTPTR[DST$L_DSC_OFFS]

	ELSE
	    SIGNAL(DBG$_INVDSTREC);


	! If the descriptor is an array descriptor, we build an Array Type RST Entry
	! and the corresponding dummy Type Spec DST Record.  We also return the
	! address of the Type RST Entry as a TYPEID.
	!
	IF (.DESCADDR[DSC$B_CLASS] EQL DSC$K_CLASS_A)    OR
	   (.DESCADDR[DSC$B_CLASS] EQL DSC$K_CLASS_NCA) OR
	   (.DESCADDR[DSC$B_CLASS] EQL DSC$K_CLASS_UBA) OR
	   (.DESCADDR[DSC$B_CLASS] EQL DSC$K_CLASS_VSA)
	THEN
	    BEGIN


	    ! Get a memory block to hold both the Type RST Entry and the Type Spec
	    ! DST Record for the array.
	    !
	    RSTPTR = DBG$GET_MEMORY(RST$K_TYPENTSIZ + 4);
	    DSTPTR = .RSTPTR + 4*RST$K_TYPENTSIZ;


	    ! Build the Data Type RST Entry.
	    !
	    RSTPTR[RST$L_DSTPTR] = .DSTPTR;
	    RSTPTR[RST$L_UPSCOPEPTR] = .RST$START_ADDR;
	    RSTPTR[RST$B_KIND] = RST$K_TYPE;
	    RSTPTR[RST$B_FCODE] = RST$K_TYPE_ARRAY;


	    ! Put this Data Type RST Entry on the Temporary RST Entry List.  This
	    ! will cause the entry to be released at the end of a DEBUG command
	    ! when there are no references to it.
	    !
	    RSTPTR[RST$L_HASH_FLINK] = .RST$TEMP_LIST;
	    RST$TEMP_LIST = .RSTPTR;


	    ! Then build the Type Specification DST Record.  This contains a Type
	    ! Spec containing a Value Spec which contains the descriptor address.
	    !
	    DSTPTR[DST$B_LENGTH] = 12;
	    DSTPTR[DST$B_TYPE] = DST$K_TYPSPEC;
	    TSPTR = DSTPTR[DST$A_TYPSPEC_TS_ADDR];
	    TSPTR[DST$W_TS_LENGTH] = 8;
	    TSPTR[DST$B_TS_KIND] = DST$K_TS_ARRAY;
	    TSPTR[DST$B_TS_ARRAY_DIM] = .DESCADDR[DSC$B_DIMCT];
	    VSPTR = TSPTR[DST$A_TS_ARRAY_FLAGS_ADDR] + 1;
```

```
        VSPTR[DST$B_VS_VFLAGS] = DST$K_VALKIND_DESC;
        VSPTR[DST$L_VS_VALUE] = .DESCADDR;


        ! Return the address of the Type RST Entry we built (the TYPEID).
        !
        RETURN .RSTPTR;
        END;


    ! It is not an array descriptor.  We therefore generate a Descriptor Type
    ! RST Entry and the corresponding dummy Type Spec DST Record.
    !
    RSTPTR = DBG$GET_MEMORY(RST$K_TYPENTSIZ + 3);
    DSTPTR = .RSTPTR + 4*RST$K_TYPENTSIZ;


    ! Build the Data Type RST Entry.
    !
    RSTPTR[RST$L_DSTPTR] = .DSTPTR;
    RSTPTR[RST$L_UPSCOPEPTR] = .RST$START_ADDR;
    RSTPTR[RST$B_KIND] = RST$K_TYPE;
    RSTPTR[RST$B_FCODE] = RST$K_TYPE_DESCR;


    ! Put the Data Type RST Entry on the Temporary RST Entry List.  This will
    ! cause the entry to be released when no longer referenced.
    !
    RSTPTR[RST$L_HASH_FLINK] = .RST$TEMP_LIST;
    RST$TEMP_LIST = .RSTPTR;


    ! Then build the Type Specification DST Record.  This will contain a Type
    ! Spec containing a Value Spec containing the descriptor address.
    !
    DSTPTR[DST$B_LENGTH] = 10;
    DSTPTR[DST$B_TYPE] = DST$K_TYPSPEC;
    TSPTR = DSTPTR[DST$A_TYPSPEC_TS_ADDR];
    TSPTR[DST$W_TS_LENGTH] = 6;
    TSPTR[DST$B_TS_KIND] = DST$K_TS_DSC;
    VSPTR = TSPTR[DST$A_TS_DSC_VSPEC_ADDR];
    VSPTR[DST$B_VS_VFLAGS] = DST$K_VALKIND_DESC;
    VSPTR[DST$L_VS_VALUE] = .DESCADDR;


    ! Return the address of the Type RST Entry we build, i.e. the TYPEID.
    !
    RETURN .RSTPTR;

    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
```

```
ROUTINE TYPEID_FOR_DESCR(DESCADDR, FCODE) =
!
! FUNCTION
!       This routine returns the Type ID associated with a VAX standard descrip-
!       tor type.  The routine builds a Data Type RST Entry for the data type
!       and returns its address as the Type ID.  This RST entry points to a dum-
!       my Type Specification DST Record, also built by this routine, which con-
!       tains a descriptor type specification which in turn contains a value
!       specification which points to the specified descriptor.  The RST entry
!       and the DST record are both built in the same memory block.  This block
!       is put on the Temporary RST Entry List so that it disappears as soon as
!       there no longer are any references to it.
!
! INPUTS
!       DESCADDR - The address of the descriptor for which a Type ID is to be
!                  returned.
!
!       FCODE    - The format code of the resultant Type ID.
!
!       ARRFLAG - (Optional). If this argument is TRUE, DESCADDR is assumed
!                 to be an ARRAY descriptor,  and a VAX standard descriptor
!                 is built to describe an element of the array.  The descrip-
!                 tor is also placed in the same memory block.
!
! OUTPUTS
!       The corresponding Type ID is returned as the routine value.
!

    BEGIN

    MAP
        DESCADDR: REF BLOCK[,BYTE];

    BUILTIN
        ACTUALCOUNT,
        ACTUALPARAMETER;

    LOCAL
        ARRFLAG,                        ! Flag set to TRUE if the input descrip-
                                        !       tor is an array descriptor
        DSTPTR: REF DST$RECORD,         ! Pointer to the type's DST record
        ELEMENT: REF DBG$STG_DESC,      ! Pointer to array element descriptor
                                        !       in constructed DST entry
        RSTPTR: REF RST$ENTRY,          ! Pointer to the type's RST entry
        SIZE,
        TSPTR: REF DST$TYPE_SPEC,       ! Pointer to the DST Type Specification
        VSPTR: REF DST$VAL_SPEC;        ! Pointer to the DST Value Specification


    ! *** How am I supposed to get this definition properly?
    !
    MACRO
        DSC$B_DIMCNT = 8, 24, 8, 0%;

    ! If the input descriptor is an array descriptor, as declared by a third
    ! argument (the ARRFLAG argument), then we set the ARRFLAG flag to TRUE.
    ! Otherwise, it is not an array descriptor and ARRFLAG is set to FALSE.
```

```
        !
        ARRFLAG = (IF ACTUALCOUNT() GTR 2 THEN ACTUALPARAMETER(3) ELSE FALSE);

        ! Figure out how big the descriptor is.
        ! Assume 12 to start with.
        ! Then check for array descriptors, which are bigger.
        ! In some cases, there may be some slop (SIZE may be bigger
        ! than the descriptor really is), but that is OK.
        !
        SIZE = 12;
        IF .ARRFLAG
        THEN
            SIZE = 10
        ELSE
            IF (.DESCADDR[DSC$B_CLASS] EQL DSC$K_CLASS_A) OR
               (.DESCADDR[DSC$B_CLASS] EQL DSC$K_CLASS_NCA) OR
               (.DESCADDR[DSC$B_CLASS] EQL DSC$K_CLASS_UBA) OR
               (.DESCADDR[DSC$B_CLASS] EQL DSC$K_CLASS_VSA)
            THEN
                SIZE = 20 + (12 * .DESCADDR[DSC$B_DIMCNT]);

        ! Get a memory block to hold both the RST entry and the DST record,
        ! and the descriptor. Copy the descriptor into the end of this
        ! memory block, and reset DESCADDR to point to this copy. (The
        ! reason for this is so that the descriptor resides in permanent
        ! memory, and in fact in the same permanent memory block as the
        ! typeid. This makes sure that the descriptor does not outlive
        ! the typeid or vice-versa.
        !
        RSTPTR = DBG$GET_MEMORY(RST$K_TYPENTSIZ + 3 + (.SIZE+3)/4);
        DSTPTR = .RSTPTR + 4*RST$K_TYPENTSIZ;
        ELEMENT = .DSTPTR+12;
        CH$MOVE(.SIZE,.DESCADDR,.ELEMENT);
        DESCADDR = .ELEMENT;

        ! For array descriptors we change the class field so it describes the
        ! element and not the entire array.
        !
        IF .ARRFLAG
        THEN
            BEGIN
            ELEMENT[DSC$B_CLASS] = (IF .ELEMENT[DSC$B_DTYPE] EQL DSC$K_DTYPE_T
                                    THEN DSC$K_CLASS_S ELSE DSC$K_CLASS_SD);
            ELEMENT[DSC$A_POINTER] = 0;
            END;

        ! Build the Data Type RST Entry for the descriptor type.
        !
        RSTPTR[RST$L_DSTPTR] = .DSTPTR;
        RSTPTR[RST$L_UPSCOPEPTR] = .RST$START_ADDR;
        RSTPTR[RST$B_KIND] = RST$K_TYPE;
        RSTPTR[RST$B_FCODE] = .FCODE;


        ! Put this Data Type RST Entry on the Temporary RST Entry List.  This will
        ! cause the entry to be released at the end of a DEBUG command when there
        ! are no references to it (when it is not locked).
```

```
2       !
2       RSTPTR[RST$L_HASH_FLINK] = .RST$TEMP_LIST;
2       RST$TEMP_LIST = .RSTPTR;
2
2
2       ! Then build the corresponding Type Specification DST Record.  This contains
2       ! a type spec which contains a value spec containing the descriptor address.
2       !
2       DSTPTR[DST$B_LENGTH] = 10;
2       DSTPTR[DST$B_TYPE] = DST$K_TYPSPEC;
2       DSTPTR[DST$B_TYPSPEC_NAME] = 0;
2       TSPTR = DSTPTR[DST$A_TYPSPEC_TS_ADDR];
2       TSPTR[DST$W_TS_LENGTH] = 6;
2       TSPTR[DST$B_TS_KIND] = DST$K_TS_DSC;
2       VSPTR = TSPTR[DST$A_TS_DSC_VSPEC_ADDR];
2       VSPTR[DST$B_VS_VFLAGS] = 0;
2       VSPTR[DST$V_VS_VALKIND] = DST$K_VALKIND_DESC;
2       VSPTR[DST$L_VS_VALUE] = .DESCADDR;
2
2
2       ! Return the address of the Data Type RST Entry as the Type ID.
2       !
2       RETURN .RSTPTR;
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
```

```
ROUTINE TYPEID_FROM_DST_TYPESPEC(DSTADDR, FCODE, BITSIZE) =
!
! FUNCTION
!       This routine builds a Data Type RST Entry for a data type specified by a
!       DST Type Specification embedded in some other symbol's DST record.  For
!       example, the cell type of an array or the referenced type of a typed
!       pointer is specified by a Type Specification in the DST record for the
!       array or the typed pointer, but does not have a separate DST record of
!       its own.  This routine is normally called when a Type ID (i.e., an RST
!       pointer to a Type RST Entry) must be produced for such a data type.
!       The RST entry is added to the Temporary RST Entry List so that it is
!       released back to the memory pool once there are no longer any references
!       to it.
!
!       The routine builds a Data Type RST Entry and a dummy DST record in the
!       same memory block.  The DST record is a Type Specification DST record
!       with the null name and an Indirect Type Specification pointing to the
!       DST Type Specification given as the input parameter.  The RST entry is
!       then set to point to this dummy DST record.
!
! INPUTS
!       DSTADDR - The address of the DST Type Specification for which a Type ID
!                 is to be generated.
!
!       FCODE   - The type Format Code associated with the Type Specification.
!
!       BITSIZE - The size in bits of an item of this type.
!
! OUTPUTS
!       The data type's Type ID (i.e., a pointer to the Type RST Entry) is
!                 returned as the routine's value.
!

    BEGIN

    LOCAL
        DSTPTR: REF DST$RECORD,          ! Pointer to generated DST record
        RSTPTR: REF RST$ENTRY,           ! Pointer to generated Type RST Entry
        TSPTR: REF DST$TYPE_SPEC;        ! Pointer to Type Spec in DST record



    ! Get a memory block large enough to accommodate both the RST entry and
    ! the dummy DST record.
    !
    RSTPTR = DBG$GET_MEMORY(RST$K_TYPENTSIZ + 3);
    DSTPTR = .RSTPTR + 4*RST$K_TYPENTSIZ;


    ! Build the Data Type RST Entry for the input type.
    !
    RSTPTR[RST$L_DSTPTR] = .DSTPTR;
    RSTPTR[RST$L_UPSCOPEPTR] = .RST$START_ADDR;
    RSTPTR[RST$B_KIND] = RST$K_TYPE;
    RSTPTR[RST$B_FCODE] = .FCODE;
    RSTPTR[RST$L_BITSIZE] = .BITSIZE;
```

```
        ! Put this Data Type RST Entry on the Temporary RST Entry List.  This will
        ! cause the entry to be released at the end of a DEBUG command when there
        ! are no references to it (when it is not locked).
        !
        RSTPTR[RST$L_HASH_FLINK] = .RST$TEMP_LIST;
        RST$TEMP_LIST = .RSTPTR;


        ! Build the dummy DST record for the type.
        !
        DSTPTR[DST$B_LENGTH] = 9;
        DSTPTR[DST$B_TYPE] = DST$K_TYPSPEC;
        DSTPTR[DST$B_TYPSPEC_NAME] = 0;
        TSPTR = .DSTPTR + 3;
        TSPTR[DST$W_TS_LENGTH] = 5;
        TSPTR[DST$B_TS_KIND] = DST$K_TS_IND_TSPEC;
        TSPTR[DST$L_TS_IND_PTR] = DBG$RST_DST_PTR( .DSTADDR - .DST$BEGIN_ADDR );


        ! Return the RST pointer to the caller.
        !
        RETURN .RSTPTR;

        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
```

```
ROUTINE TYPEID_FROM_DST_RECORD(DSTADDR, FCODE, BITSIZE) =
!
! FUNCTION
!       This routine returns a Type ID for the item described by the DST record
!       pointed to by DSTADDR.
!
! INPUTS
!       DSTADDR - The DST address of the record defining this item.
!
!       FCODE   - The Format Code which corresponds to the type of this item.
!
!       BITSIZE - The size in bits of this item.
!
! OUTPUTS
!       A Type ID for the type of the item is returned as the routine's value.
!


    BEGIN

    MAP
        DSTADDR: REF DST$RECORD;            ! Pointer to DST record from which we
                                            !       will construct a Type ID

    LOCAL
        RSTPTR: REF RST$ENTRY;              ! Pointer to generated Type RST Entry



    ! If the data item is atomic, we call DBG$TYPEID_FOR_ATOMIC to build the
    ! desired Data Type RST Entry.  We return its output directly.
    !
    IF .FCODE EQL RST$K_TYPE_ATOMIC
    THEN
        RETURN DBG$TYPEID_FOR_ATOMIC(.DSTADDR[DST$B_TYPE], .BITSIZE, FALSE);


    ! Get a memory block to hold the Data Type RST Entry.
    !
    RSTPTR = DBG$GET_MEMORY(RST$K_TYPENTSIZ);


    ! Build the Data Type RST Entry for the input type.
    !
    RSTPTR[RST$L_DSTPTR] = .DSTADDR;
    RSTPTR[RST$L_UPSCOPEPTR] = .RST$START_ADDR;
    RSTPTR[RST$B_KIND] = RST$K_TYPE;
    RSTPTR[RST$B_FCODE] = .FCODE;
    RSTPTR[RST$L_BITSIZE] = .BITSIZE;


    ! Put this Data Type RST Entry on the Temporary RST Entry List.  This will
    ! cause the entry to be released at the end of a DEBUG command when there
    ! are no references to it (when it is not locked).
    !
    RSTPTR[RST$L_HASH_FLINK] = .RST$TEMP_LIST;
    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
```

```
ROUTINE FIND_TYPE_RECORD(SYMID, FCODE, BITSIZE) =
!
! FUNCTION
!       This routine takes a symid, and tracks down the type spec which
!       describes it. In the process, it determines the Format Code and
!       bit size of the item.
!
! INPUTS
!       SYMID   - The SYMID of the item whose type we want.  (Note that the
!                 SYMID may itself represent a named type.)
!
!       FCODE   - The address of a longword to receive the format code of
!                 the specified item.
!
!       BITSIZE - The address of a longword to receive the size in bits of
!                 the specified item.
!
! OUTPUTS
!       FCODE   - The format code of the item is returned to FCODE.
!
!       BITSIZE - The size in bits of the item is returned to BITSIZE.
!
!       The data type's Type ID (i.e., a pointer to the Type RST Entry) is
!                 returned as the routine's value; zero is returned if no
!                 type exists.
!

    BEGIN

    MAP
        SYMID: REF RST$ENTRY;

    LOCAL
        CLASS,
        DESC_PTR: REF BLOCK[,BYTE],
        DST_PTR: REF DST$RECORD,
        HAVE_TYPE,
        KIND,
        LENGTH,
        SAVEFCODE,
        SAVEID: REF RST$ENTRY,
        SIZE,
        TYPEID: REF RST$ENTRY,
        TYPE_PTR: REF DST$RECORD,
        TYPE_SPEC: REF DST$TYPE_SPEC,
        VALKIND,
        VAL_PTR,
        VAL_VECT: VECTOR[3];


    NOVEL_LENGTH_FLAG = FALSE;
    DBG$STA_SYMKIND(.SYMID, KIND);
    SELECTONE .KIND OF
        SET
        [RST$K_TYPE]:
            BEGIN
            .FCODE = .SYMID[RST$B_FCODE];
```

```
            .BITSIZE = .SYMID[RST$L_BITSIZE];
            RETURN .SYMID;
            END;

        [RST$K_TYPCOMP, RST$K_DATA]:
            BEGIN
            TYPEID = .SYMID[RST$L_TYPEPTR];
            IF .TYPEID NEQ 0
            THEN
                BEGIN
                HAVE_TYPE = TRUE;
                SAVEFCODE = .TYPEID[RST$B_FCODE];
                SAVEID = .TYPEID;
                END

            ELSE
                HAVE_TYPE = FALSE;

            DST_PTR = .SYMID[RST$L_DSTPTR];
            SELECTONE .DST_PTR[DST$B_TYPE] OF
                SET
                [DST$K_SEPTYP]:
                    BEGIN

                    ! Normally, TYPE_SPEC returned from DBG$FIND_SEPTYP does
                    ! not mean anything if TYPE_PTR[DST$B_TYPE] EQL
                    ! DST$K_ENUMBEG or DST$K_RECBEG.  In here this case
                    ! is caught early by HAVE_TYPE set to TRUE.  In other
                    ! words, in this routine, DST$K_ENUMBEG or DST$K_RECBEG
                    ! always comes in to have TYPEID. (We hope).
                    !
                    SIZE = DBG$FIND_SEPTYP(.DST_PTR, TYPE_PTR, TYPE_SPEC);
                    .BITSIZE = .SIZE;
                    IF .HAVE_TYPE
                    THEN
                        BEGIN
                        .FCODE = .SAVEFCODE;

                        ! The novel length flags gets set earlier, in
                        ! FIND_TYPSPEC, if we encounterd a novel length
                        ! type spec on the way to our "real" type_spec.
                        ! If this flag is set, and if the "novel" size
                        ! is different from the size we have in the
                        ! typeid, then we build a new typeid with
                        ! the novel length in it.
                        !
                        IF .NOVEL_LENGTH_FLAG AND
                           (.SAVEFCODE EQL RST$K_TYPE_ENUM) AND
                           (.SIZE NEQ .TYPEID[RST$L_BITSIZE])
                        THEN
                            BEGIN
                            OWN
                                DUMMY: VECTOR[2];
                            LOCAL
                                LAST_SYMID: REF RST$ENTRY,
```

```
6                   TEMP_SYMID: REF RST$ENTRY;
6
6               LENGTH = RST$K_TYPENTSIZ + .TYPEID[RST$L_TYPCOMPCNT];
6               SAVEID = DBG$GET_MEMORY(.LENGTH + 3);
6
6
6               ! Make the copy.
6               !
6               CH$MOVE(.LENGTH*4, .TYPEID, .SAVEID);
6
6
6               ! Modify the typeid. We create a "dummy" hash
6               ! chain for it.
6               !
6               SAVEID[RST$L_UPSCOPEPTR] = .RST$START_ADDR;
6               SAVEID[RST$L_BITSIZE] = .SIZE;
6               DUMMY[0] = .SAVEID;
6               DUMMY[1] = .SAVEID;
6               SAVEID[RST$L_HASH_FLINK] = .DUMMY;
6               SAVEID[RST$L_HASH_BLINK] = .DUMMY;
6               SAVEID[RST$L_SYMCHNPTR] = 0;
6
6               ! We put the new SAVEID typeid on the symchain
6               ! for this module so it gets freed up when we
6               ! cancel the module.
6               !
6               TEMP_SYMID = .SYMID;
6               WHILE .TEMP_SYMID NEQ 0 DO
7                   BEGIN
7                   LAST_SYMID = .TEMP_SYMID;
7                   TEMP_SYMID = .TEMP_SYMID[RST$L_SYMCHNPTR];
6                   END;
6               LAST_SYMID[RST$L_SYMCHNPTR] = .SAVEID;
6               SYMID[RST$L_TYPEPTR] = .SAVEID;
5               END;
5
5           RETURN .SAVEID;
4           END;
4
4       ! If we have a descriptor type, then we need to check
4       ! whether this is an array descriptor.
4       !
5       IF  (.TYPE_SPEC[DST$B_TS_KIND] EQL DST$K_TS_DSC)
4       THEN
5           BEGIN
5           VAL_PTR = TYPE_SPEC[DST$A_TS_DSC_VSPEC_ADDR];
5           DBG$STA_VALSPEC(.VAL_PTR, VAL_VECT, VAL_KIND);
5           IF .VAL_KIND NEQ DBG$K_VAL_DESCR
5           THEN
5               $DBG_ERROR('RSTTYPES\FIND_TYPE_RECORD 20');
5
5           DESC_PTR = .VAL_VECT[0];
5           DBG$READ_ACCESS(.DESC_PTR, 12);
5           CLASS = .DESC_PTR[DSC$B_CLASS];
5           IF (.CLASS EQL DSC$K_CLASS_A)   OR
5               (.CLASS EQL DSC$K_CLASS_NCA) OR
5               (.CLASS EQL DSC$K_CLASS_UBA) OR
```

```
                    (.CLASS EQL DSC$K_CLASS_VSA)
                THEN
                    BEGIN
                    .FCODE = RST$K_TYPE_ARRAY;
                    RETURN TYPEID_FOR_DESCR(.DESC_PTR, ..FCODE);
                    END
                ELSE
                    BEGIN
                    .FCODE = RST$K_TYPE_DESCR;
                    RETURN TYPEID_FOR_DESCR(.DESC_PTR, ..FCODE);
                    END;
                END;

            .FCODE = DBG$TRANS_TYPE_CODE(.TYPE_SPEC[DST$B_TS_KIND]);
            IF ..FCODE EQL RST$K_TYPE_ATOMIC
            THEN
                RETURN DBG$TYPEID_FOR_ATOMIC(
                        .TYPE_SPEC[DST$B_TS_ATOM_TYP], .SIZE, FALSE)

            ELSE
                RETURN TYPEID_FROM_DST_RECORD(.TYPE_PTR,..FCODE,.SIZE);

            END;

        [DST$K_BLI]:
            BEGIN
            .FCODE = RST$K_TYPE_BLIDATA;
            .BITSIZE = GET_BITSIZE_FROM_BLIDST(.SYMID[RST$L_DSTPTR]);
            RETURN DBG$TYPEID_FOR_ATOMIC(RST$K_TYPE_BLIDATA,
                    ..BITSIZE, FALSE);
            END;

        [DST$K_ENUMELT, DST$K_RECBEG]:
            BEGIN
            .FCODE = .SAVEID[RST$B_FCODE];
            .BITSIZE = .SAVEID[RST$L_BITSIZE];
            RETURN .SAVEID;
            END;


        ! Handle the COBOL Hack DST Record.  Here we call a routine
        ! which converts that DST record into a more regular Type RST
        ! Entry.  We then get the FCODE from that RST entry and re-
        ! turn the entry's address as a Type ID.
        !
        [DST$K_COB_HACK]:
            BEGIN
            TYPEID = TYPEID_FOR_COB_HACK(.DST_PTR);
            .FCODE = .TYPEID[RST$B_FCODE];
            .BITSIZE = 0;
            RETURN .TYPEID;
            END;

        [DST$K_BLIFLD]:
            BEGIN
            .FCODE = RST$K_TYPE_BLIFLD;
            .BITSIZE = 0;
```

```
                        RETURN DBG$TYPEID_FOR_ATOMIC(RST$K_TYPE_BLIFLD,0,FALSE);
                        END;

                    [DST$K_LBLORLIT]:
                        BEGIN
                        .FCODE = RST$K_TYPE_ATOMIC;
                        .BITSIZE = 32;
                        RETURN DBG$TYPEID_FOR_ATOMIC(DSC$K_DTYPE_LU,32,FALSE);
                        END;

                    [OTHERWISE]:
                        BEGIN
                        IF (.DST_PTR[DST$B_TYPE] LSS DSC$K_DTYPE_LOWEST OR
                                .DST_PTR[DST$B_TYPE] GTR DSC$K_DTYPE_HIGHEST) AND
                            (.DST_PTR[DST$B_TYPE] NEQ DST$K_BOOL) AND
                            (.DST_PTR[DST$B_TYPE] NEQ DSC$K_DTYPE_TF)
                        THEN
                            $DBG_ERROR('RSTTYPES\FIND_TYPE_RECORD');

                        .FCODE = DESC_OR_ARRAY_OR_ATOM(.SYMID, DESC_PTR);
                        IF ..FCODE EQL RST$K_TYPE_ATOMIC
                        THEN
                            BEGIN
                            .BITSIZE = GET_BITSIZE_FROM_DTYPE(.DST_PTR[DST$B_TYPE]);
                            RETURN DBG$TYPEID_FOR_ATOMIC(.DST_PTR[DST$B_TYPE],
                                        ..BITSIZE,FALSE);
                            END

                        ELSE
                            BEGIN
                            .BITSIZE = 0;
                            RETURN TYPEID_FOR_DESCR(.DESC_PTR, ..FCODE);
                            END;

                        END;

                    TES;

                END;

            [RST$K_VARIANT]:
                BEGIN
                LOCAL
                    VAR_PTR: REF DST$VARBEG_TRAILER;

                .FCODE = RST$K_VARIANT;
                DST_PTR = .SYMID[RST$L_DSTPTR];
                VAR_PTR = DST_PTR[DST$B_NAME] + 1 + .DST_PTR[DST$B_NAME];
                .BITSIZE = .VAR_PTR[DST$L_VARBEG_SIZE];
                RETURN .SYMID;
                END;

            [OTHERWISE]:
                RETURN 0;

            TES;
```

```
; 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, #163                                       ; 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  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 0009B           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

```
ROUTINE DESC_OR_ARRAY_OR_ATOM(SYMID, DESC_PTR) =
!
! FUNCTION
!       This routine takes a SYMID and determines whether it is of type atomic,
!       array, or descriptor.
!
! INPUTS
!       SYMID    - The SYMID in question.
!
!       DESC_PTR - The address of a longword location to receive the address
!                  of the item's descriptor (if present).
!
! OUTPUTS
!       DESC_PTR - For return values of RST$K_TYPE_DESCR and RST$K_TYPE_ARRAY,
!                  DESC_PTR receives the address of the item's descriptor.
!
!       The routine's value will be one of RST$K_TYPE_ATOMIC, RST$K_TYPE_DESCR,
!                  or RST$K_TYPE_ARRAY.
!

    BEGIN
    BUILTIN
        PROBER;
    MAP
        DESC_PTR: REF BLOCK[,BYTE],
        SYMID: REF RST$ENTRY;

    LOCAL
        CLASS,
        DESC_ADDR: REF BLOCK[,BYTE],
        DST_PTR: REF DST$RECORD,
        VALPTR: VECTOR[3],
        VKIND;



    DST_PTR = .SYMID[RST$L_DSTPTR];
    IF .DST_PTR[DST$V_VALKIND] NEQ DST$K_VALKIND_DESC
    THEN
        RETURN RST$K_TYPE_ATOMIC

    ELSE
        BEGIN
        IF .DST_PTR[DST$B_VFLAGS] EQL DST$K_VFLAGS_DSC
        THEN
            DESC_ADDR = .DST_PTR[DST$L_DSC_OFFS] + DST_PTR[DST$A_DSC_BASE]

        ELSE
            BEGIN
            DBG$STA_SYMVALUE(.SYMID,VALPTR,VKIND);
            IF .VKIND NEQ DBG$K_VAL_DESCR
            THEN
                $DBG_ERROR('RSTTYPES\DESC_OR_ARRAY_OR_ATOM');

            DESC_ADDR = .VALPTR[0];
            END;
```

```
; 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                                                          ;
                                                        4D  4F  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 00059 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
```

```
ROUTINE FIND_TYPESPEC(TYPE_SPEC, BITSIZE, DST_PTR) =
!
! FUNCTION
!       This routine takes a starting TYPE_SPEC and traces through the DST
!       to find the destination type specification; in this process, the
!       routine is sensitive to DST$K_TS_IND, DST$K_SEPTYP, DST$K_GLOBNXT,
!       and DST$K_TS_NOV_LENG. (All of these types indicate that the actual
!       type specification is found elsewhere.) When a DST$K_TS_NOV_LENG is
!       detected, BITSIZE is updated to reflect the novel length.
!
!       The routine loops until a true Type Spec is located.
!
! INPUTS
!       TYPE_SPEC - The DST address of the initial type-specification.
!
!       BITSIZE   - The address of a longword location to receive the size in
!                   bits of the specified type.
!
!       DST_PTR   - The address of a longword location to receive the DST
!                   address of the DST record which contains the desired
!                   type specification.
!
! OUTPUTS
!       The routine's value is the DST address of the desired type-specification.
!
!       BITSIZE - The size in bits of an item of this type is returned to
!                 BITSIZE. If a novel length record was detected, BITSIZE
!                 reflects the novel length; otherwise, BITSIZE receives the
!                 default size for the indicated data type.
!
!       DST_PTR - The address of the DST record which contains the desired
!                 type-spec is returned to DST_PTR. The routine's value is
!                 always of type DST$TYPE_SPEC; DST_PTR enables the calling
!                 routine to know the start address of the containing DST record.
!

    BEGIN

    LOCAL
        RECBEG_TRLR: REF DST$RECBEG_TRLR,
        SIZE,
        TEMP_TS: REF DST$TYPE_SPEC,
        TYPE_PTR: REF DST$RECORD;


    TEMP_TS = .TYPE_SPEC;
    SIZE = 0;
    TYPE_PTR = 0;
    WHILE TRUE DO
        BEGIN
        SELECTONE .TEMP_TS[DST$B_TS_KIND] OF
            SET
            [DST$K_TS_IND]:
                BEGIN
                TYPE_PTR = DBG$RST_DST_PTR( .DST$BEGIN_ADDR + .TEMP_TS[DST$L_TS_IND_PTR] );
                IF .TYPE_PTR[DST$B_TYPE] EQL DST$K_SEPTYP
                THEN
```

```
                    BEGIN
                    TYPE_PTR = DBG$RST_NEXT_DST( .TYPE_PTR );
                    IF .TYPE_PTR[DST$B_TYPE] EQL DST$K_GLOBNXT
                    THEN
                        TYPE_PTR = DBG$RST_NEXT_DST( .TYPE_PTR );

                    END;

                IF (.TYPE_PTR[DST$B_TYPE] EQL DST$K_ENUMBEG) OR
                   (.TYPE_PTR[DST$B_TYPE] EQL DST$K_RECBEG)
                THEN
                    EXITLOOP;

                TEMP_TS = TYPE_PTR[DST$A_TYPSPEC_TS_ADDR] +
                        .TYPE_PTR[DST$B_TYPSPEC_NAME];
                END;


            ! Handle the kind of indirect pointer which points directly to
            ! a DST Type Spec (as opposed to a Type Spec DST record).
            !
            [DST$K_TS_IND_TSPEC]:
                TEMP_TS = DBG$RST_DST_PTR( .DST$BEGIN_ADDR + .TEMP_TS[DST$L_TS_IND_PTR] );


            ! Handle the Novel Length Type Spec.  Here we pick up the length
            ! of data objects of this type and indirect to the Type Spec.
            !
            [DST$K_TS_NOV_LENG]:
                BEGIN


                ! Make a note for the fact, we have encounted a novel
                ! length dst.
                !
                NOVEL_LENGTH_FLAG = TRUE;
                SIZE = .TEMP_TS[DST$L_TS_NOV_LENG];
                TYPE_PTR = DBG$RST_DST_PTR( .DST$BEGIN_ADDR + .TEMP_TS[DST$L_TS_NOV_LENG_PAR_TSPEC] );
                IF .TYPE_PTR[DST$B_TYPE] EQL DST$K_SEPTYP
                THEN
                    BEGIN
                    TYPE_PTR = DBG$RST_NEXT_DST( .TYPE_PTR );
                    IF .TYPE_PTR[DST$B_TYPE] EQL DST$K_GLOBNXT
                    THEN
                        TYPE_PTR = DBG$RST_NEXT_DST( .TYPE_PTR );

                    END;

                IF (.TYPE_PTR[DST$B_TYPE] EQL DST$K_ENUMBEG) OR
                   (.TYPE_PTR[DST$B_TYPE] EQL DST$K_RECBEG)
                THEN
                    EXITLOOP;

                TEMP_TS = TYPE_PTR[DST$A_TYPSPEC_TS_ADDR] +
                        .TYPE_PTR[DST$B_TYPSPEC_NAME];
                END;
```

```
3                      ! In all other cases, just exit the loop.
3                      !
3                      [OTHERWISE]:
3                          EXITLOOP;
3                      TES;
3
2                  END;
2
2          IF .TYPE_PTR NEQ 0
2          THEN
3              BEGIN
3              .DST_PTR = .TYPE_PTR;
3
3
3              ! Note TEMP_TS does not mean anything for TYPE_PTR[DST$B_TYPE] EQL
3              ! DST$K_ENUMBEG or DST$K_RECBEG.
3              !
3              IF .TYPE_PTR[DST$B_TYPE] EQL DST$K_RECBEG
3              THEN
4                  BEGIN
4                  RECBEG_TRLR = TYPE_PTR[DST$B_NAME] + 1 + .TYPE_PTR[DST$B_NAME];
4
4
4                  ! Get size only if there is no novel length overrides.
4                  !
4                  IF .SIZE EQL 0 THEN SIZE = .RECBEG_TRLR[DST$L_RECBEG_SIZE];
4                  .BITSIZE = .SIZE;
4                  RETURN .TEMP_TS;
4                  END
4
3              ELSE
4                  BEGIN
4                  IF .TYPE_PTR[DST$B_TYPE] EQL DST$K_ENUMBEG
4                  THEN
5                      BEGIN
5
5
5                      ! Get size only if there is no novel length overrides.
5                      !
5                      IF .SIZE EQL 0 THEN SIZE = .TYPE_PTR[DST$B_ENUMBEG_LENG];
5                      .BITSIZE = .SIZE;
5                      RETURN .TEMP_TS;
4                      END;
4
3                  END;
3
2              END;
2
2          IF .SIZE EQL 0 THEN SIZE = DBG$GET_BITSIZE_FROM_TYPESPEC(.TEMP_TS);
2          .BITSIZE = .SIZE;
2          RETURN .TEMP_TS;
2
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$RST_NEXT_DST                            ;
          52            50  D0 0003F            MOVL    R0, TYPE_PTR                                    ;
     A0   8F         01 A2  91 00042            CMPB    1(TYPE_PTR), #160                               ; 3018
                        45  13 00047            BEQL    5$                                              ;
                        4B  11 00049            BRB     6$                                              ; 3024
          0F            50  91 0004B 2$:        CMPB    R0, #15                                         ; 3037
                        0D  12 0004E            BNEQ    4$                                              ;
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
```

```
1 ROUTINE FIND_TYPREC_FROM_TSPEC( TYPE_SPEC ) =
1 !
1 ! FUNCTION
1 !       This routine takes a TYPE SPECIFICATION and returns the (possibly)
1 !       constructed TYPEID which describes the type.
1 !
1 ! INPUTS
1 !       TYPE_SPEC - The type specification for which we need a TYPEID.
1 !
1 ! OUTPUTS
1 !       The address of a TYPEID describing type_spec.
1 !
1
2 BEGIN
2
2     LOCAL
2         FCODE,
2         SIZE,
2         TEMP_TS: REF DST$TYPE_SPEC,
2         TYPEID: REF RST$ENTRY,
2         TYPE_PTR: REF DST$RECORD;
2
2
2     SIZE = 0;
2     TYPE_PTR = 0;
2     TEMP_TS = FIND_TYPESPEC(.TYPE_SPEC, SIZE, TYPE_PTR);
2     IF .TYPE_PTR NEQ 0
2     THEN
3         BEGIN
3         IF (.TYPE_PTR[DST$B_TYPE] EQL DST$K_ENUMBEG) OR
4            (.TYPE_PTR[DST$B_TYPE] EQL DST$K_RECBEG)
3         THEN
4             BEGIN
4             TYPEID = FIND_RST_TYPE_ENTRY(.TYPE_PTR);
4             IF .TYPEID NEQ 0 THEN RETURN .TYPEID;
3             END;
2         END;
2
2     ! If we have a descriptor type, then we need to check
2     ! whether this is an array descriptor.
2     !
3     IF  (.TEMP_TS[DST$B_TS_KIND] EQL DST$K_TS_DSC)
2     THEN
3         BEGIN
3         LOCAL
3             CLASS,
3             DESC_PTR: REF BLOCK[,BYTE],
3             VALKIND,
3             VAL_PTR,
3             VAL_VECT: VECTOR[3];
3
3         VAL_PTR = TEMP_TS[DST$A_TS_DSC_VSPEC_ADDR];
3         DBG$STA_VALSPEC(.VAL_PTR, VAL_VECT, VALKIND);
3         IF .VALKIND NEQ DBG$K_VAL_DESCR
3         THEN
3             $DBG_ERROR('RSTTYPES\FIND_TYPREC_FROM_TSPEC 20');
3
```

```
            3          DESC_PTR = .VAL_VECT[0];
            3          DBG$READ_ACCESS(.DESC_PTR, 12);
            3          CLASS = .DESC_PTR[DSC$B_CLASS];
            3          IF (.CLASS EQL DSC$K_CLASS_A)    OR
            3             (.CLASS EQL DSC$K_CLASS_NCA) OR
            3             (.CLASS EQL DSC$K_CLASS_UBA) OR
            4             (.CLASS EQL DSC$K_CLASS_VSA)
            3          THEN
            4              BEGIN
            4              FCODE = RST$K_TYPE_ARRAY;
            4              RETURN TYPEID_FOR_DESCR(.DESC_PTR, .FCODE);
            3              END;
            2          END;
            2
            2      FCODE = DBG$TRANS_TYPE_CODE(.TEMP_TS[DST$B_TS_KIND]);
            2      IF .FCODE EQL RST$K_TYPE_ATOMIC
            2      THEN
            2          RETURN DBG$TYPEID_FOR_ATOMIC(.TEMP_TS[DST$B_TS_ATOM_TYP],.SIZE, FALSE);
            2
            2      IF .TYPE_PTR NEQ 0
            2      THEN
            2          RETURN TYPEID_FROM_DST_RECORD(.TYPE_PTR, .FCODE, .SIZE);
            2
            2      RETURN TYPEID_FROM_DST_TYPESPEC(.TEMP_TS,.FCODE,.SIZE);
            2
            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                                   ;
          FAD0    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

```
1 GLOBAL ROUTINE DBG$GET_BITSIZE_FROM_DESC(DESC_PTR): =
1 !
1 ! FUNCTION
1 !       Given pointer to VAX Standard Descriptor, get its size.
1 !
1 ! INPUT
1 !       DESC_PTR - Pointer to VAX Standard Descriptor.
1 !
1 ! OUTPUT
1 !       Size in bits is returned, it can be zero for non-contiguous array.
1 !
1
2     BEGIN
2
2     MAP
2         DESC_PTR: REF BLOCK[,BYTE];
2
2     LOCAL
2         SIZE;
2
2
2     ! Check for read access.
2     !
2     DBG$READ_ACCESS(.DESC_PTR, 12);
2
2     CASE .DESC_PTR[DSC$B_CLASS] FROM DSC$K_CLASS_S TO DSC$K_CLASS_UBA OF
2         SET
2
2         ! Scalar types, scaled decimal types, and unaligned bit strings.
2         ! The DBG$DATA_LENGTH routine computes a bit length by looking
2         ! at the DSC$W_LENGTH field, and then looking at the dtype field
2         ! to figure out whether the length is in bits, nibbles, or bytes.
2         !
2         [DSC$K_CLASS_S, DSC$K_CLASS_D, DSC$K_CLASS_SD, DSC$K_CLASS_UBS]:
3             BEGIN
3             SIZE = DBG$DATA_LENGTH(.DESC_PTR);
2             END;
2
2         ! Arrays. The length is given in the ARSIZE field. It is either
2         ! in bytes or nibbles, depending on the dtype.
2         !
2         [DSC$K_CLASS_A]:
3             BEGIN
3             IF .DESC_PTR[DSC$B_DTYPE] EQL DSC$K_DTYPE_P
3             THEN
3                 SIZE = .DESC_PTR[DSC$L_ARSIZE] * 4
3             ELSE
3                 SIZE = .DESC_PTR[DSC$L_ARSIZE] * 8;
2             END;
2
2         ! Non-contiguous arrays. The ARSIZE field contains a length,
2         ! but that length may not be meaningful. Return 0 since we
2         ! don't really know what the length is.
2         !
2         [DSC$K_CLASS_NCA, DSC$K_CLASS_UBA, DSC$K_CLASS_VSA]:
3             BEGIN
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$GET_BITSIZE_FROM_DESC, Save R2      ; 3210
                                      0C   DD 00002           PUSHL   #12                                     ; 3233
                        52       04   AC   D0 00004           MOVL    DESC_PTR, R2                            ;
                                      52   DD 00008           PUSHL   R2                                      ;
               00000000G  00              02   FB 0000A           CALLS   #2, DBG$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$                                   ;
                     00028F50       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
```

```
ROUTINE GET_BITSIZE_FROM_DTYPE(DTYPE) =
!
! FUNCTION
!       This routine takes an SRM DTYPE and returns the default bitsize
!       defined for that type.
!
! INPUTS
!       DTYPE   - The VAX Standard Data Type whose bit size we want.
!
! OUTPUTS
!       The corresponding bit size is returned as the routine's value.
!

    BEGIN

    BIND
        LENGTH_TBL = UPLIT BYTE(0, 1, 1,2,4,8, 1,2,4,8, 4,8,8,16,
                                REP 11 OF (0), 16,16,8,16,16,32,
                                12,8,8,0,0,0,0): VECTOR[,BYTE];

L       %IF DSC$K_DTYPE_HIGHEST GTR 37  ! If the LENGTH_TBL PLIT is too small,
U       %THEN                           !       generate a compile-time error
U           %ERROR('Error:  Must expand PLIT in Routine GET_BITSIZE_FROM_DTYPE')
        %FI;



    ! Return the bitsize corresponding to the input data type.  Note that
    ! we treat Boolean (as in PASCAL) as a special case.
    !
    IF .DTYPE EQL DST$K_BOOL OR .DTYPE EQL DSC$K_DTYPE_TF THEN RETURN 8;
    RETURN 8*.LENGTH_TBL[.DTYPE];

    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
```

```
1 GLOBAL ROUTINE DBG$GET_BITSIZE_FROM_TYPESPEC( TYPE_SPEC ) =
1 !
1 ! FUNCTION
1 !       This routine takes a type specification and returns the corresponding
1 !       default bit size.
1 !
1 ! INPUTS
1 !       type_spec - The type specification whose default size we want.
1 !
1 ! OUTPUTS
1 !       The type's default bit size is returned as the routine's value.
1 !
1
2     BEGIN
2
2     MAP
2         TYPE_SPEC: REF DST$TYPE_SPEC;
2
2     LOCAL
2         DESCPTR: REF BLOCK[,BYTE],
2         FLAGS_PTR: REF BITVECTOR,
2         NDIMS,
2         SIZE,
2         VAL_VECT: VECTOR[3],
2         VALKIND,
2         VALPTR: REF DST$VAL_SPEC;
2
2
2     SELECTONE .TYPE_SPEC[DST$B_TS_KIND] OF
2         SET
2
2         [DST$K_TS_ATOM]:
2             SIZE = GET_BITSIZE_FROM_DTYPE(.TYPE_SPEC[DST$B_TS_ATOM_TYP]);
2
2         [DST$K_TS_DSC]:
3             BEGIN
3             VALPTR = TYPE_SPEC[DST$A_TS_DSC_VSPEC_ADDR];
3             DBG$STA_VALSPEC(.VALPTR, VAL_VECT, VALKIND);
3             IF .VALKIND NEQ DBG$K_VAL_DESCR
3             THEN
3                 $DBG_ERROR('RSTTYPES\DBG$GET_BITSIZE_FROM_TYPESPEC');
3
3             DESCPTR = .VAL_VECT[0];
3             SIZE = DBG$GET_BITSIZE_FROM_DESC(.DESCPTR);
3             END;
2
2         [DST$K_TS_TPTR, DST$K_TS_PTR, DST$K_TS_FILE]:
2             SIZE = 32;
2
2         [DST$K_TS_PIC]:
2             SIZE = .TYPE_SPEC[DST$B_TS_PIC_DLENG]*%BPUNIT;
2
2         [DST$K_TS_ARRAY]:
3             BEGIN
3             NDIMS = .TYPE_SPEC[DST$B_TS_ARRAY_DIM];
3             FLAGS_PTR = TYPE_SPEC[DST$A_TS_ARRAY_FLAGS_ADDR];
3             VALPTR = .FLAGS_PTR + (.NDIMS/8) + 1;
```

```
3                      DBG$STA_VALSPEC(.VALPTR, VAL_VECT, VALKIND);
3                      IF .VALKIND NEQ DBG$K_VAL_DESCR
3                      THEN
3                          $DBG_ERROR('RSTTYPES\DBG$GET_BITSIZE_FROM_TYPESPEC');
3
3                      DESCPTR = .VAL_VECT[0];
3                      SIZE = DBG$GET_BITSIZE_FROM_DESC(.DESCPTR);
2                      END;
2
2                  [DST$K_TS_SET]:
2                      SIZE = .TYPE_SPEC[DST$L_TS_SET_LENG];
2
2                  [DST$K_TS_SUBRANGE]:
2                      SIZE = .TYPE_SPEC[DST$L_TS_SUBR_LENG];
2
2                  [DST$K_TS_NOV_LENG]:
2                      SIZE = .TYPE_SPEC[DST$L_TS_NOV_LENG];
2
2                  [DST$K_TS_SELF_REL_LABEL]:
2                      SIZE = .TYPE_SPEC[DST$L_TS_SELF_LENG] * 8;
2
2                  [DST$K_TS_RFA]:
2                      SIZE = 6 * 8;
2
2                  [DST$K_TS_TASK]:
2                      SIZE = 32;
2
2                  [OTHERWISE]:
2                      SIZE = 0;
2
2                  TES;
2
2              RETURN .SIZE;
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$:        BRW     14$                                 ;
          0B              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
```

```
ROUTINE FIND_RST_TYPE_ENTRY(DSTPTR) =
!
! FUNCTION
!       This routine takes the address of a DST record, and tries to find the
!       RST record which it defined.  In particular, this routine serves to
!       find the TYPEID for those types which were built into the RST at SET
!       MODULE time (Record and Enumeration Types), but are now being referenced
!       via a DST pointer.  The routine simply passes through the RST looking
!       for an RST$L_DSTPTR which matches the input DSTPTR.
!
! INPUTS
!       DSTPTR  - The address of the DST record of interest.
!
! OUTPUTS
!       The TYPEID of the Record or Enumeration Type we are seeking is returned
!                 as the routine value.  If no such TYPEID exists, zero is re-
!                 turned as the routine value.
!

    BEGIN

    LOCAL
        MODPTR: REF RST$ENTRY,          ! Pointer to current Module RST Entry
        SYMID: REF RST$ENTRY;           ! Pointer to current symbol RST entry
                                        !       within the current module



    ! Start up a loop through the RST Module Chain which scans all modules.
    ! Note that we skip over the "anonymous module" at the start of the chain;
    ! this module contains all the Global Symbols and is not of interest.
    !
    MODPTR = .RST$START_ADDR[RST$L_NXTMODPTR];
    WHILE .MODPTR NEQ 0 DO
        BEGIN


        ! Now search the Symbol Chain for the current module, looking for an
        ! RST Entry whose DSTPTR is the same as the given DST pointer.  When
        ! such an RST Entry is found, return its address as the symbol SYMID.
        !
        SYMID = .MODPTR[RST$L_SYMCHNPTR];
        WHILE .SYMID NEQ 0 DO
            BEGIN
            IF .SYMID[RST$L_DSTPTR] EQL .DSTPTR THEN RETURN .SYMID;
            SYMID = .SYMID[RST$L_SYMCHNPTR];
            END;


        ! The DST pointer was not found in this module.  Link on to the next
        ! module and search its symbol chain.
        !
        MODPTR = .MODPTR[RST$L_NXTMODPTR];
        END;


    ! 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
```

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

RSTTYPES
LIS

0101 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

SSIK
LIS

DELTA
LIS

DELETE

DELEMAIN
LIS

SSITAB
LIS

DELETE
MAP

DELTA

DELTA
MAP

SSIU
LIS

SSIUW
LIS

PURGE
LIS

STRUCDEF
LIS

SSIDISP
LIS

S0DELTA
MAP

DELETE
REQ