F 7

```
BBBBBBBBBBBB        AAAAAAAAA        CCCCCCCCCCC   KKK         KKK  UUU         UUU  PPPPPPPPPPPP
BBBBBBBBBBBB        AAAAAAAAA        CCCCCCCCCCC   KKK         KKK  UUU         UUU  PPPPPPPPPPPP
BBBBBBBBBBBB        AAAAAAAAA        CCCCCCCCCCC   KKK         KKK  UUU         UUU  PPPPPPPPPPPP
BBB         BBB  AAA         AAA  CCC              KKK         KKK  UUU         UUU  PPP         PPP
BBB         BBB  AAA         AAA  CCC              KKK         KKK  UUU         UUU  PPP         PPP
BBB         BBB  AAA         AAA  CCC              KKK         KKK  UUU         UUU  PPP         PPP
BBB         BBB  AAA         AAA  CCC              KKK      KKK     UUU         UUU  PPP         PPP
BBB         BBB  AAA         AAA  CCC              KKK      KKK     UUU         UUU  PPP         PPP
BBB         BBB  AAA         AAA  CCC              KKK      KKK     UUU         UUU  PPP         PPP
BBBBBBBBBBBB     AAA         AAA  CCC              KKKKKKKKK        UUU         UUU  PPPPPPPPPPPP
BBBBBBBBBBBB     AAA         AAA  CCC              KKKKKKKKK        UUU         UUU  PPPPPPPPPPPP
BBBBBBBBBBBB     AAA         AAA  CCC              KKKKKKKKK        UUU         UUU  PPPPPPPPPPPP
BBB         BBB  AAAAAAAAAAAAAAA  CCC              KKK      KKK     UUU         UUU  PPP
BBB         BBB  AAAAAAAAAAAAAAA  CCC              KKK      KKK     UUU         UUU  PPP
BBB         BBB  AAAAAAAAAAAAAAA  CCC              KKK      KKK     UUU         UUU  PPP
BBB         BBB  AAA         AAA  CCC              KKK         KKK  UUU         UUU  PPP
BBB         BBB  AAA         AAA  CCC              KKK         KKK  UUU         UUU  PPP
BBB         BBB  AAA         AAA  CCC              KKK         KKK  UUU         UUU  PPP
BBBBBBBBBBBB     AAA         AAA     CCCCCCCCCCC   KKK         KKK  UUUUUUUUUUUUUUU  PPP
BBBBBBBBBBBB     AAA         AAA     CCCCCCCCCCC   KKK         KKK  UUUUUUUUUUUUUUU  PPP
BBBBBBBBBBBB     AAA         AAA     CCCCCCCCCCC   KKK         KKK  UUUUUUUUUUUUUUU  PPP
```

FILEID**UTILITY

UTILITY.

LIS

```
0 MODULE UTILITY  (%TITLE 'Utility routines'
0                 IDENT = 'V04-000'
0                 ) =
1 BEGIN
1
1 !
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1 !
1
1 !++
1 ! FACILITY:
1 !       Backup/Restore
1 !
1 ! ABSTRACT:
1 !       This module contains general utility routines.
1 !
1 ! ENVIRONMENT:
1 !       VAX/VMS user mode.
1 !--
1 !
1 ! AUTHOR: M. Jack, CREATION DATE: 03-Sep-1980
1 !
1 ! MODIFIED BY:
1 !
1 !       V03-007 LY0510          Larry Yetto             19-JUL-1984 08:48
1 !               Add DECODE_DEVTYP routine to build an ASCII string from
1 !               the DEVTYP in the physical volume attribute record.  Increase
1 !               DEVTYP from 1 to 4 bytes.
1 !
1 !       V03-006 LY0484          Larry Yetto             27-APR-1984 08:41
1 !               FT1 QAR # 2088  -  Add new routines to use to determine if
1 !               a saveset is encrypted.
1 !
1 !       V03-005 ACG0332         Andrew C. Goldstein,    22-Jun-1983  18:29
1 !               Add file highwater mark and RMS journal flag attributes
1 !
```

```
1 !
1 !          V03-004 JEP0003           J. Eric Pollack,         23-Apr-1983  10:53
1 !                  Add support for encrypted savesets.
1 !
1 !          V03-003 ACG0313           Andrew C. Goldstein,     12-Feb-1983  17:29
1 !                  Add routine subtitles
1 !
1 !          V03-002 LMP0044           L. Mark Pilant,          21-Oct-1982  15:20
1 !                  Add support for saving and restoring ACL's.
1 !
1 !          V03-001 ACG0280           Andrew C. Goldstein,     2-Apr-1982  16:45
1 !                  Include dot on save set name with null type
1 !
1 !          V02-010 MLJ0081           Martin L. Jack, 26-Feb-1982  15:21
1 !                  Add RETAINMIN and RETAINMAX attributes to support new home
1 !                  block fields.
1 !
1 !          V02-009 MLJ0075           Martin L. Jack, 28-Jan-1982  20:08
1 !                  Add DIR_VERLIM and VERLIMIT attributes to support version limit
1 !                  handling.
1 !
1 !          V02-008 MLJ0062           Martin L. Jack, 3-Dec-1981  12:17
1 !                  Add DIR_STATUS attribute to support /INCREMENTAL.
1 !
1 !          V02-007 MLJ0054           Martin L. Jack, 22-Nov-1981  21:32
1 !                  Integrate GET_VM and FREE_VM jacket routines.
1 !
1 !          V02-006 MLJ0047           Martin L. Jack, 25-Sep-1981  15:28
1 !                  Remove INVATTTYP message to allow restores
1 !                  with earlier versions.
1 !
1 !          V02-005 MLJ0036           Martin L. Jack, 28-Aug-1981  18:04
1 !                  Implement parent directory attributes.  Make use of new NAM
1 !                  block descriptors.
1 !
1 !          V02-004 MLJ0023           Martin L. Jack, 23-Apr-1981  11:42
1 !                  Implement placement attribute.
1 !
1 !          V02-003 MLJ0014           Martin L. Jack, 6-Apr-1981  23:19
1 !                  Correct and clarify VBN sequence-error messages
1 !
1 !          V02-002 MLJ0010           Martin L. Jack, 25-Mar-1981  15:50
1 !                  Reorganize global storage.  Add new attributes for image
1 !                  restore.  Convert to bitmap for attribute code checking.
1 !                  Add DEBLOCK parameter to inhibit freeing of buffer.
1 !
1 !          V02-001 MLJ0006           Martin L. Jack, 26-Feb-1981  17:37
1 !                  Append filetype in EXTRACT_FILENAME unless it is null
1 !
1 !**
```

```
1 REQUIRE 'SRC$:COMMON';
1 LIBRARY 'SYS$LIBRARY:LIB';
1 REQUIRE 'LIB$:BACKDEF';
1
1
1 LINKAGE
1         L_P$=            CALL: GLOBAL(P$=11);
1
1
1 MACRO
1         L_DECL=          EXTERNAL REGISTER P$ = 11 %;
1
1
1 FORWARD ROUTINE
1         GET_VM,                                  ! Allocate virtual memory
1         GET_ZERO_VM,                             ! Allocate and zero virtual memory
1         FREE_VM:         NOVALUE,                ! Free virtual memory
1         DEBLOCK:         L_P$ NOVALUE,           ! Deblock a save set buffer
1         DEBLOCK_ATTR:    L_P$ NOVALUE,           ! Deblock an attribute record
1         FILE_ERROR:      NOVALUE,                ! Signal a file-related error
1         FIND_BADBLOCK,                           ! Search the bad block table
1         EXTRACT_FILENAME:
1                          NOVALUE,                ! Extract file name (and type, if not
1                                                  ! null) from string
1         EXTRACT_DIR_FILENAME:
1                          NOVALUE,                ! Extract directory, name, type,
1                                                  ! version from string
1         GET_ONE_ATTRIBUTE:
1                          NOVALUE,                ! Store one attribute value
1         GET_SUMMARY_ATTRIBUTES:
1                          NOVALUE,                ! Get BACKUP summary attributes
1         GET_VOLUME_ATTRIBUTES:
1                          NOVALUE,                ! Get volume summary attributes
1         GET_FILE_ATTRIBUTES:
1                          NOVALUE,                ! Get file attributes
1         GET_FILE_EXT_ATTRIBUTES:
1                          NOVALUE,                ! Get file attributes from extension record
1         GET_PHYSVOL_ATTRIBUTES:
1                          NOVALUE,                ! Get physical volume attributes
1         CRYPTO_CHKSAV,                           ! Check if saveset is enrypted
1         SEARCHFOR_BSR_ENC:                       ! Scan records for backup summary
1                          L_P$ NOVALUE,           !  and determine if saveset is encrypted
1         DECODE_DEVTYP :
1                          NOVALUE ;               ! Decode MEDIA_ID
1
1
1 EXTERNAL ROUTINE
1         FREE_BUFFER:     NOVALUE,                ! Free an I/O buffer
1         LIB$FREE_VM:     ADDRESSING_MODE(GENERAL),
1                                                  ! Free virtual memory
1         LIB$GET_VM:      ADDRESSING_MODE(GENERAL),
1                                                  ! Allocate virtual memory
1         LIB$SIGNAL:      ADDRESSING_MODE(GENERAL);
1                                                  ! Signal a condition
1
1
1 G$DEFINE();              ! Define global common area
```

```
1
1
1 EXTERNAL LITERAL
1           BACKUP$_VBNERROR,
1           BACKUP$_VBNMISSING,
1           BACKUP$_VBNPASTEOF,
1           BACKUP$_READBLOCK,
1           BACKUP$_WRITEBLOCK,
1           BACKUP$_VERIFYERR,
1           BACKUP$_BADDATA,
1           BACKUP$_SOFTRERRS,
1           BACKUP$_SOFTWERRS,
1           BACKUP$_XORERRS,
1           BACKUP$_PHYSET,
1           BACKUP$_NONPHYSET,
1           BACKUP$_INVATTSTR,
1           BACKUP$_INVATTSIZ,
1           BACKUP$_INVRECTYP,
1           BACKUP$_INVRECSIZ,
1           BACKUP$_ALLOCMEM,
1           BACKUP$_FREEMEM;
1
1
```

```
%SBTTL 'Attribute validation tables'
! Define bitvectors of valid attribute codes for each attribute record type.
!
MACRO
        DEFINE_BITVECTOR_1[A]=
            [%NAME('BSA$K_', A)] = TRUE %,


        DEFINE_BITVECTOR(A)=
            OWN A:
                BITVECTOR[BSA$K_NUM_ATRS]
                PSECT(CODE)
                PRESET(DEFINE_BITVECTOR_1(%REMAINING)) %;


DEFINE_BITVECTOR(MASK_BSR,
        SSNAME,         COMMAND,        COMMENT,        USERNAME,
        USERUIC,        DATE,           OPSYS,          SYSVER,
        NODENAME,       SIR,            DRIVEID,        BACKVER,
        BLOCKSIZE,      XORSIZE,        BUFFERS,        VOLSETNAM,
        NVOLS,          BACKSIZE,       BACKFILES,      CRYPDATKEY);


DEFINE_BITVECTOR(MASK_VSR,
        VOLSTRUCT,      VOLNAME,        OWNERNAME,      FORMAT,
        RVN,            VOLOWNER,       PROTECT,        FILEPROT,
        RECPROT,        VOLCHAR,        VOLDATE,        WINDOW,
        LRU_LIM,        EXTEND,         CLUSTER,        RESFILES,
        VOLSIZE,        TOTSIZE,        TOTFILES,       MAXFILES,
        MAXFILNUM,      SERIALNUM,      INDEXLBN,       BOOTBLOCK,
        RETAINMIN,      RETAINMAX);


DEFINE_BITVECTOR(MASK_FAR,
        FILENAME,       STRUCLEV,       FID,            BACKLINK,
        FILESIZE,       UIC,            FPRO,           RPRO,
        ACLEVEL,        UCHAR,          RECATTR,        REVISION,
        CREDATE,        REVDATE,        EXPDATE,        BAKDATE,
        BOOTVBN,        PLACEMENT,      DIR_UIC,        DIR_FPRO,
        DIR_STATUS,     DIR_VERLIM,     VERLIMIT,       ACLSEGMENT,
        HIGHWATER,      JNL_FLAGS);


DEFINE_BITVECTOR(MASK_PVA,
        SECTORS,        TRACKS,         CYLINDERS,      MAXBLOCK,
        DEVTYP,         SERIAL,         DEVNAM,         LABEL,
        BADBLOCK);

!
! Module wide own storage
!
OWN
    SAVESET_ENCR : LONG ;
```

```
1 %SBTTL 'GET_VM - allocate virtual memory'
1 GLOBAL ROUTINE GET_VM(SIZE)=
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !       This routine interfaces to LIB$GET_VM to allocate a block of virtual
1 !       memory.
1 !
1 ! INPUT PARAMETERS:
1 !       SIZE            - Size in bytes of area.
1 !
1 ! IMPLICIT INPUTS:
1 !       NONE
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       Address of allocated area.
1 !
1 ! SIDE EFFECTS:
1 !       If allocation fails, a fatal error is signalled.
1 !
1 !--
1
2 BEGIN
2 LOCAL
2       STATUS,                                 ! General status variable
2       AREA;                                   ! Pointer to allocated area
2
2
2 STATUS = LIB$GET_VM(SIZE, AREA);
2 IF NOT .STATUS THEN SIGNAL(BACKUP$_ALLOCMEM, 0, .STATUS);
2 .AREA
1 END;


                                        .TITLE  UTILITY Utility routines
                                        .IDENT  \V04-000\

                                        .PSECT  COMMON,NOEXE,  OVR,2

                        00000 GLOBAL_BASE:
                                        .BLKB   0
                        00000 FREE_LIST:
                                        .BLKB   8
                        00008 INPUT_WAIT:
                                        .BLKB   8
                        00010 REREAD_WAIT:
                                        .BLKB   8
                        00018 OUTPUT_WAIT:
                                        .BLKB   8
                        00020 JPI_UIC:.BLKB     4
```

```
00024 JPI_USERNAME:
                .BLKB   12
00030 JPI_DATE:
                .BLKB   8
00038 JPI_NODE_DESC:
                .BLKB   8
00040 JPI_CURPRIV:
                .BLKB   8
00048 SYI_VERSION:
                .BLKB   4
0004C SYI_SID:.BLKB   4
00050 RWSV_HOLD_LIST:
                .BLKB   8
00058 RWSV_CRC16:
                .BLKB   64
00098 RWSV_AUTODIN:
                .BLKB   64
000D8 RWSV_FILESET_ID:
                .BLKB   8
000E0 RWSV_VOLUME_ID:
                .BLKB   12
000EC RWSV_VOL_NUMBER:
                .BLKB   2
000EE RWSV_SEG_NUMBER:
                .BLKB   2
000F0 RWSV_FILE_NUMBER:
                .BLKB   4
000F4 RWSV_SAVE_QUAL:
                .BLKB   4
000F8 RWSV_SAVE_FAB:
                .BLKB   4
000FC RWSV_CHAN:
                .BLKB   4
00100 RWSV_XOR_BCB:
                .BLKB   4
00104 RWSV_IN_SEQ:
                .BLKB   4
00108 RWSV_IN_SEQ_0:
                .BLKB   4
0010C RWSV_IN_XOR_SEQ:
                .BLKB   4
00110 RWSV_IN_XOR_RFA:
                .BLKB   6
00116 RWSV_LOOKAHEAD:
                .BLKB   1
00117 RWSV_XORSIZE:
                .BLKB   1
00118 RWSV_IN_GROUP_SIZE:
                .BLKB   4
0011C RWSV_IN_ERRORS:
                .BLKB   2
0011E RWSV_IN_XORUSE:
                .BLKB   2
00120 RWSV_IN_ORGERR:
                .BLKB   8
00128 RWSV_IN_VBN:
                .BLKB   4
```

```
00012C RWSV_IN_VBN_0:
                .BLKB   4
00130 RWSV_ALLOC:
                .BLKB   4
00134 RWSV_EOF:
                .BLKB   4
00138 RWSV_OUT_SEQ:
                .BLKB   4
0013C RWSV_OUT_VBN:
                .BLKB   4
00140 RWSV_OUT_BLOCK_COUNT:
                .BLKB   4
00144 RWSV_OUT_ERRORS:
                .BLKB   2
00146 RWSV_SEQ_ERRORS:
                .BLKB   2
00148 RWSV_OUT_GROUP_COUNT:
                .BLKB   1
00149 RWSV_PADDING:
                .BLKB   3
0014C QUAL:     .BLKB   112
001BC COM_SSNAME:
                .BLKB   8
001C4 COM_VALID_TYPES:
                .BLKB   2
001C6 COM_FLAGS:
                .BLKB   2
001C8 COM_PADDING:
                .BLKB   1
001C9 COM_BUFF_COUNT:
                .BLKB   1
001CA COM_I_SETCOUNT:
                .BLKB   1
001CB COM_O_SETCOUNT:
                .BLKB   1
001CC COM_I_STRUCNAME:
                .BLKB   12
001D8 COM_O_STRUCNAME:
                .BLKB   12
001E4 COM_O_BSRDATE:
                .BLKB   8
001EC ALT_SSNAME:
                .BLKB   32
0020C INPUT_FUNC:
                .BLKB   1
0020D INPUT_RTYPE:
                .BLKB   1
0020E OUTPUT_FUNC:
                .BLKB   1
0020F FAST_STRUCLEV:
                .BLKB   1
00210 INPUT_BEG:
                .BLKB   0
00210 INPUT_CHAN:
                .BLKB   4
00214 INPUT_FLAGS:
                .BLKB   2
```

```
00216 INPUT_PADDING:
                .BLKB   2
00218 INPUT_FAB:
                .BLKB   4
0021C INPUT_NAM:
                .BLKB   4
00220 INPUT_BCB:
                .BLKB   4
00224 INPUT_QUAL:
                .BLKB   4
00228 INPUT_BAD:
                .BLKB   4
0022C INPUT_BLOCK:
                .BLKB   4
00230 INPUT_MAXBLOCK:
                .BLKB   4
00234 INPUT_MEDIA_ID:
                .BLKB   4
00238 INPUT_NAMEDESC:
                .BLKB   8
00240 INPUT_STATBLK:
                .BLKB   8
00248 INPUT_HDR_BEG:
                .BLKB   0
00248 INPUT_CREDATE:
                .BLKB   8
00250 INPUT_REVDATE:
                .BLKB   8
00258 INPUT_EXPDATE:
                .BLKB   8
00260 INPUT_BAKDATE:
                .BLKB   8
00268 INPUT_FILEOWNER:
                .BLKB   4
0026C INPUT_FILECHAR:
                .BLKB   4
00270 INPUT_RECATTR:
                .BLKB   32
00290 INPUT_HDR_END:
                .BLKB   0
00290 INPUT_END:
                .BLKB   0
00290 INPUT_PROC_LIST:
                .BLKB   4
00294 INPUT_PLACEMENT:
                .BLKB   8
0029C INPUT_VBN_LIST:
                .BLKB   8
002A4 INPUT_PLACE_LEN:
                .BLKB   2
002A6 INPUT_PADDING_2:
                .BLKB   2
002A8 OUTPUT_BEG:
                .BLKB   0
002A8 OUTPUT_CHAN:
                .BLKB   4
002AC OUTPUT_FLAGS:
```

```
                    .BLKB   2
002AE OUTPUT_PADDING:
                    .BLKB   2
002B0 OUTPUT_FAB:
                    .BLKB   4
002B4 OUTPUT_NAM:
                    .BLKB   4
002B8 OUTPUT_BCB:
                    .BLKB   4
002BC OUTPUT_QUAL:
                    .BLKB   4
002C0 OUTPUT_BAD:
                    .BLKB   4
002C4 OUTPUT_BLOCK:
                    .BLKB   4
002C8 OUTPUT_MAXBLOCK:
                    .BLKB   4
002CC OUTPUT_DEVGEOM:
                    .BLKB   8
002D4 OUTPUT_ATTBUF:
                    .BLKB   144
00364 OUTPUT_END:
                    .BLKB   0
00364 LIST_TOTFILES:
                    .BLKB   4
00368 LIST_TOTSIZE:
                    .BLKB   4
0036C VERIFY_FAB:
                    .BLKB   4
00370 VERIFY_USE_COUNT:
                    .BLKB   4
00374 VERIFY_QUAL:
                    .BLKB   4
00378 COMPARE_BCB:
                    .BLKB   4
0037C FAST_BUFFER:
                    .BLKB   4
00380 FAST_BUFFER_SIZE:
                    .BLKB   4
00384 FAST_RVN:
                    .BLKB   1
00385 FAST_PADDING:
                    .BLKB   1
00386 DIR_VERLIMIT:
                    .BLKB   2
00388 FAST_VOL_BEG:
                    .BLKB   0
00388 FAST_IMAP_SIZE:
                    .BLKB   4
0038C FAST_IMAP:
                    .BLKB   4
00390 FAST_HDR_OFFSET:
                    .BLKB   4
00394 FAST_BOOT_LBN:
                    .BLKB   4
00398 FAST_VOL_END:
                    .BLKB   0
```

```
00398 JOUR_BUFFER:
                .BLKB   4
0039C JOUR_DIR:
                .BLKB   4
003A0 JOUR_HIBLK:
                .BLKB   4
003A4 JOUR_EFBLK:
                .BLKB   4
003A8 JOUR_INBLK:
                .BLKB   4
003AC JOUR_FFBYTE:
                .BLKB   2
003AE JOUR_INBYTE:
                .BLKB   2
003B0 JOUR_STRUCT_LEV:
                .BLKB   2
003B2 JOUR_COUNT:
                .BLKB   1
003B3 JOUR_REVERSE:
                .BLKB   1
003B4 JOUR_EXSZ:
                .BLKB   2
003B6 JOUR_PADDING:
                .BLKB   2
003B8 CHKPT_HIGH_SP:
                .BLKB   4
003BC CHKPT_LOW_SP:
                .BLKB   4
003C0 CHKPT_STACK:
                .BLKB   4
003C4 CHKPT_VARS:
                .BLKB   4
003C8 CHKPT_STATUS:
                .BLKB   4
003CC DIR_BEG:.BLKB     0
003CC DIR_CHAN:
                .BLKB   4
003D0 DIR_NAM:.BLKB     4
003D4 DIR_DEV_DESC:
                .BLKB   4
003D8 DIR_SEL_DIR:
                .BLKB   8
003E0 DIR_SEL_NTV:
                .BLKB   8
003E8 DIR_STRUCLEV:
                .BLKB   1
003E9 DIR_LEVELS:
                .BLKB   1
003EA DIR_FLAGS:
                .BLKB   1
003EB DIR_STATUS:
                .BLKB   1
003EC DIR_STRING:
                .BLKB   320
0052C DIR_STACK:
                .BLKB   612
00790 DIR_SP: .BLKB     4
```

```
                                00794 DIR_SEL_LATEST:
                                              .BLKB    4
                                00798 DIR_END:.BLKB    0
                                00798 DIR_SCANLIMIT:
                                              .BLKB    36
                                007BC INPUT_MTL:
                                              .BLKB    4
                                007C0 OUTPUT_MTL:
                                              .BLKB    4
                                007C4 CURRENT_MTL:
                                              .BLKB    4
                                007C8 CURRENT_VCB:
                                              .BLKB    4
                                007CC CURRENT_WCB:
                                              .BLKB    4
                                007D0 ACL_FIB_DESCR:
                                              .BLKB    8
                                007D8 ACL_FIB:.BLKB    64
                                00818 ACL_LENGTH:
                                              .BLKB    4
                                0081C ACL_BUFFER:
                                              .BLKB    4
                                00820 CRYP_IN_CONTEXT:
                                              .BLKB    4
                                00824 CRYP_OU_CONTEXT:
                                              .BLKB    4
                                00828 CRYP_DA_CONTEXT:
                                              .BLKB    4
                                0082C CRYP_DATA_ENCIV:
                                              .BLKB    8
                                00834 CRYP_DATA_CODE:
                                              .BLKB    4
                                00838 CRYP_DATA_KEY:
                                              .BLKB    8
                                00840 CRYP_DATA_IV:
                                              .BLKB    8
                                00848 CRYP_DATA_CKSM:
                                              .BLKB    4

                                              .PSECT   DATA,NOEXE,2

                                00000 SAVESET_ENCR:
                                              .BLKB    4

                                              .PSECT   CODE,NOWRT,2

             0F    FF    FE     00000 MASK_BSR:
                                              .BYTE    -2, -1, 15
                         00#    00003         .BYTE    0[7]
                         02     0000A         .BYTE    2
                                0000B         .BLKB    1
                         00#    0000C MASK_VSR:
                                              .BYTE    0[2]
      03     FF    FF    F0     0000E         .BYTE    -16, -1, -1, 3
                         00#    00012         .BYTE    0[2]
                   30    18     00014         .BYTE    24, 48
                                00016         .BLKB    1
```

```
                                         00017           .BLKB    1
                                     00# 00018 MASK_FAR:
                                                         .BYTE    0[5]
                    01  CF  E0  03  FF  FC  0001D        .BYTE    -4, -1, 3, -32, -49, 1                       ;
                                         00023           .BLKB    1
                                     00# 00024 MASK_PVA:
                                                         .BYTE    0[7]                                         ;
                                07  FC  0002B            .BYTE    -4, 7                                        ;
                                         0002D           .BLKB    2

                                                         .EXTRN   FREE_BUFFER, LIB$FREE_VM
                                                         .EXTRN   LIB$GET_VM, LIB$SIGNAL
                                                         .EXTRN   BACKUP$_VBNERROR
                                                         .EXTRN   BACKUP$_VBNMISSING
                                                         .EXTRN   BACKUP$_VBNPASTEOF
                                                         .EXTRN   BACKUP$_READBLOCK
                                                         .EXTRN   BACKUP$_WRITEBLOCK
                                                         .EXTRN   BACKUP$_VERIFYERR
                                                         .EXTRN   BACKUP$_BADDATA
                                                         .EXTRN   BACKUP$_SOFTRERRS
                                                         .EXTRN   BACKUP$_SOFTWERRS
                                                         .EXTRN   BACKUP$_XORERRS
                                                         .EXTRN   BACKUP$_PHYSET, BACKUP$_NONPHYSET
                                                         .EXTRN   BACKUP$_INVATTSTR
                                                         .EXTRN   BACKUP$_INVATTSIZ
                                                         .EXTRN   BACKUP$_INVRECTYP
                                                         .EXTRN   BACKUP$_INVRECSIZ
                                                         .EXTRN   BACKUP$_ALLOCMEM
                                                         .EXTRN   BACKUP$_FREEMEM

                                    0000 00000           .ENTRY   GET_VM, Save nothing                         ; 1796
                  5E                 04  C2 00002        SUBL2    #4, SP
                                     5E  DD 00005        PUSHL    SP                                           ; 1830
                             04      AC  9F 00007        PUSHAB   SIZE                                         ;
         00000000G  00               02  FB 0000A        CALLS    #2, LIB$GET_VM                               ;
                    11               50  E8 00011        BLBS     STATUS, 1$                                   ; 1831
                                     50  DD 00014        PUSHL    STATUS                                       ;
                                     7E  D4 00016        CLRL     -(SP)                                        ;
                         00000000G   8F  DD 00018        PUSHL    #BACKUP$_ALLOCMEM                            ;
         00000000G  00               03  FB 0001E        CALLS    #3, LIB$SIGNAL                               ;
                    50               6E  D0 00025 1$:    MOVL     AREA, R0                                     ; 1833
                                         04 00028        RET                                                   ;

; Routine Size:  41 bytes,    Routine Base:  CODE + 002F
```

```
1 %SBTTL 'GET_ZERO_VM - allocate and zero virtual memory'
1 GLOBAL ROUTINE GET_ZERO_VM(SIZE)=
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !       This routine interfaces to LIB$GET_VM to allocate a block of virtual
1 !       memory and clears the allocated area.
1 !
1 ! INPUT PARAMETERS:
1 !       SIZE            - Size in bytes of area.
1 !
1 ! IMPLICIT INPUTS:
1 !       NONE
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       Address of allocated area.
1 !
1 ! SIDE EFFECTS:
1 !       If allocation fails, a fatal error is signalled.
1 !
1 !--
1
2 BEGIN
2 LOCAL
2       STATUS,                         ! General status variable
2       AREA;                           ! Pointer to allocated area
2
2
2 STATUS = LIB$GET_VM(SIZE, AREA);
2 IF NOT .STATUS THEN SIGNAL(BACKUP$_ALLOCMEM, 0, .STATUS);
2 CH$FILL(0, .SIZE, .AREA);
2 .AREA
1 END;



                                003C 00000           .ENTRY  GET_ZERO_VM, Save R2,R3,R4,R5          ; 1835
                  5E         04   C2 00002           SUBL2   #4, SP
                             5E   DD 00005           PUSHL   SP                                     ; 1869
                       04    AC   9F 00007           PUSHAB  SIZE
          00000000G  00      02   FB 0000A           CALLS   #2, LIB$GET_VM
                  11         50   E8 00011           BLBS    STATUS, 1$                             ; 1870
                             50   DD 00014           PUSHL   STATUS
                             7E   D4 00016           CLRL    -(SP)
                  00000000G  8F   DD 00018           PUSHL   #BACKUP$_ALLOCMEM
          00000000G  00      03   FB 0001E           CALLS   #3, LIB$SIGNAL
  04   AC      00    6E      00   2C 00025  1$:      MOVC5   #0, (SP), #0, SIZE, @AREA              ; 1871
                       00    BE      0002B
```

```
                  50        6E  D0 0002D          MOVL     AREA, R0                                  ; 1873
                                04 00030          RET                                                ;

; Routine Size:  49 bytes,    Routine Base:  CODE + 0058
```

```
1 %SBTTL 'FREE_VM - release virtual memory'
1 GLOBAL ROUTINE FREE_VM(SIZE,AREA): NOVALUE=
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !       This routine interfaces to LIB$FREE_VM to free a block of virtual
1 !       memory.
1 !
1 ! INPUT PARAMETERS:
1 !       SIZE                - Size of allocated area.
1 !       AREA                - Address of allocated area.
1 !
1 ! IMPLICIT INPUTS:
1 !       NONE
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !       If deallocation fails, a fatal error is signalled.
1 !
1 !--
1
2 BEGIN
2 LOCAL
2       STATUS;                                   ! General status variable
2
2
2 STATUS = LIB$FREE_VM(SIZE, AREA);
2 IF NOT .STATUS THEN SIGNAL(BACKUP$_FREEMEM, 0, .STATUS);
1 END;
```

```
                                  0000 00000          .ENTRY  FREE_VM, Save nothing           ; 1875
                        08    AC    9F 00002          PUSHAB  AREA                            ; 1909
                        04    AC    9F 00005          PUSHAB  SIZE                            ;
        00000000G  00         02    FB 00008          CALLS   #2, LIB$FREE_VM                 ;
                   11         50    E8 0000F          BLBS    STATUS, 1$                      ; 1910
                              50    DD 00012          PUSHL   STATUS                          ;
                              7E    D4 00014          CLRL    -(SP)                           ;
                   00000000G  8F    DD 00016          PUSHL   #BACKUP$_FREEMEM                ;
        00000000G  00         03    FB 0001C          CALLS   #3, LIB$SIGNAL                  ;
                                    04 00023 1$:      RET                                     ; 1911

; Routine Size:  36 bytes,    Routine Base:  CODE + 0089
```

```
1 %SBTTL 'DEBLOCK - deblock and free save set buffer'
1 GLOBAL ROUTINE DEBLOCK(BCB,ROUT,NOFREE): L_P$ NOVALUE=
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !	This routine deblocks a buffer, and then frees the buffer unless
1 !	inhibited.  Validity checking of the size and type values is done.
1 !
1 ! INPUT PARAMETERS:
1 !	BCB		- Pointer to buffer control block.
1 !	ROUT		- Routine to be called with each record.
1 !	NOFREE		- True if buffer not to be freed (optional).
1 !
1 ! IMPLICIT INPUTS:
1 !	NONE
1 !
1 ! OUTPUT PARAMETERS:
1 !	NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !	NONE
1 !
1 ! ROUTINE VALUE:
1 !	NONE
1 !
1 ! SIDE EFFECTS:
1 !	The buffer may be released.
1 !
1 !--
1
2 BEGIN
2 MAP
2	BCB:		REF BBLOCK;	! Pointer to buffer control block
2 LOCAL
2	REC:		REF BBLOCK,	! Pointer to record
2	BUF:		REF BBLOCK,	! Pointer to buffer header
2	LIMIT;				! Limit address of buffer
2 BUILTIN
2	ACTUALCOUNT;
2 L_DECL;
2
2
2 ! Initialize.
2 !
2 BUF = .BCB[BCB_BUFFER];
2 LIMIT = .BUF + .BCB[BCB_SIZE];
2 REC = .BUF + BBH$C_LENGTH;
2
2
2 ! Loop over all records in the buffer.
2 !
2 WHILE .REC LEQA .LIMIT - BRH$C_LENGTH DO
3     BEGIN
3     LOCAL
3	RLIMIT;				! Limit address of this record
3
```

```

    ! Compute address of following byte and check that it is within the buffer.
    !
    RLIMIT = .REC + .REC[BRH$W_RSIZE] + BRH$C_LENGTH;
    IF .RLIMIT GTRA .LIMIT
    THEN
        BEGIN
        SIGNAL(BACKUP$_INVRECSIZ);
        EXITLOOP;
        END;


    ! Check record type.  Ensure that it is a valid type, and if it is, ensure
    ! that it is valid for the type of operation (physical or non-physical)
    ! that is in progress.
    !
    IF .REC[BRH$W_RTYPE] GTRU BRH$K_FILE_EXT
    THEN
        SIGNAL(BACKUP$_INVRECTYP)
    ELSE
        BEGIN
        IF NOT .COM_VALID_TYPES[.REC[BRH$W_RTYPE]]
        THEN
            IF .QUAL[QUAL_PHYS]
                THEN SIGNAL(BACKUP$_NONPHYSET)
                ELSE SIGNAL(BACKUP$_PHYSET);


        ! Check record size.  If a data record, the size must be a multiple of
        ! 512 bytes.  Otherwise, it must be a multiple of 16 bytes.  If all
        ! tests are passed, call the action routine to process the record.
        !
        IF
            BEGIN
            IF
                (.REC[BRH$W_RTYPE] EQL BRH$K_VBN OR
                 .REC[BRH$W_RTYPE] EQL BRH$K_LBN)
            THEN
                .(REC[BRH$W_RSIZE])<0,9> NEQ 0
            ELSE
                .(REC[BRH$W_RSIZE])<0,4> NEQ 0
            END
        THEN
            SIGNAL(BACKUP$_INVRECSIZ)
        ELSE
            (.ROUT)(.REC, .BCB);
        END;


    ! Advance to next record.
    !
    REC = .RLIMIT;
    END;


! If warranted, free the buffer.
!
```

```
;   478          2026  2 IF
;   479          2027  2      .BCB[BCB_STATE] NEQ BCB_S_WRITE AND
;   480          2028  3      NOT (IF ACTUALCOUNT() LSSU 3 THEN FALSE ELSE .NOFREE)
;   481          2029  2 THEN
;   482          2030  2      FREE_BUFFER(.BCB);
;   483          2031  1 END;



                                    01FC 00000          .ENTRY  DEBLOCK, Save R2,R3,R4,R5,R6,R7,R8          ; 1913
         58 00000000G     8F   D0 00002          MOVL    #BACKUP$_INVRECSIZ, R8                       ;
         57 00000000G     00   9E 00009          MOVAB   LIB$SIGNAL, R7                               ;
         54          04   AC   D0 00010          MOVL    BCB, R4                                      ; 1957
         52          0C   A4   D0 00014          MOVL    12(R4), BUF                                  ;
         53          08   A4   3C 00018          MOVZWL  8(R4), LIMIT                                 ; 1958
         53               52   C0 0001C          ADDL2   BUF, LIMIT                                   ;
         52        0100   C2   9E 0001F          MOVAB   256(R2), REC                                 ; 1959
         56          F0   A3   9E 00024          MOVAB   -16(R3), R6                                  ; 1964
         56               52   D1 00028 1$:      CMPL    REC, R6                                      ;
                          71   1A 0002B          BGTRU   13$                                          ;
         50               62   3C 0002D          MOVZWL  (REC), R0                                    ; 1972
         55       10 A042      9E 00030          MOVAB   16(R0)[REC], RLIMIT                          ;
         53               55   D1 00035          CMPL    RLIMIT, LIMIT                                ; 1973
                          07   1B 00038          BLEQU   2$                                           ;
                          58   DD 0003A          PUSHL   R8                                           ; 1976
         67               01   FB 0003C          CALLS   #1, LIB$SIGNAL                               ;
                          5D   11 0003F          BRB     13$                                          ; 1975
         08          02   A2   B1 00041 2$:      CMPW    2(REC), #8                                   ; 1985
                          08   1B 00045          BLEQU   3$                                           ;
            00000000G     8F   DD 00047          PUSHL   #BACKUP$_INVRECTYP                           ; 1987
                          3F   11 0004D          BRB     10$                                          ;
         50          02   A2   3C 0004F 3$:      MOVZWL  2(REC), R0                                   ; 1990
 19 00000000'     EF      50   E0 00053          BBS     R0, COM_VALID_TYPES, 6$                      ;
 08 00000000'     EF      05   E1 0005B          BBC     #5, QUAL+12, 4$                              ; 1992
            00000000G     8F   DD 00063          PUSHL   #BACKUP$_NONPHYSET                           ; 1993
                          06   11 00069          BRB     5$                                           ;
            00000000G     8F   DD 0006B 4$:      PUSHL   #BACKUP$_PHYSET                              ; 1994
         67               01   FB 00071 5$:      CALLS   #1, LIB$SIGNAL                               ;
         04          02   A2   B1 00074 6$:      CMPW    2(REC), #4                                   ; 2004
                          06   13 00078          BEQL    7$                                           ;
         06          02   A2   B1 0007A          CMPW    2(REC), #6                                   ; 2005
                          07   12 0007E          BNEQ    8$                                           ;
      01FF  8F            62   B3 00080 7$:      BITW    (REC), #511                                  ; 2007
                          03   11 00085          BRB     9$                                           ;
         0F               62   93 00087 8$:      BITB    (REC), #15                                   ; 2009
                          07   13 0008A 9$:      BEQL    11$                                          ;
                          58   DD 0008C          PUSHL   R8                                           ; 2012
         67               01   FB 0008E 10$:     CALLS   #1, LIB$SIGNAL                               ;
                          06   11 00091          BRB     12$                                          ;
                          14   BB 00093 11$:     PUSHR   #^M<R2,R4>                                   ; 2014
        08  BC            02   FB 00095          CALLS   #2, @ROUT                                    ;
         52               55   D0 00099 12$:     MOVL    RLIMIT, REC                                  ; 2020
                          8A   11 0009C          BRB     1$                                           ; 1964
         02          0A   A4   91 0009E 13$:     CMPB    10(R4), #2                                   ; 2027
                          12   13 000A2          BEQL    15$                                          ;
```

```
                              03          6C  91 000A4          CMPB    (AP), #3                              ; 2028
                                          04  1F 000A7          BLSSU   14$                                   ;
                              09     0C   AC  E8 000A9          BLBS    NOFREE, 15$                           ;
                                          54  DD 000AD 14$:     PUSHL   R4                                    ; 2030
                     00000000G  00        01  FB 000AF          CALLS   #1, FREE_BUFFER                       ;
                                              04 000B6 15$:     RET                                           ; 2031

; Routine Size:  183 bytes,    Routine Base:  CODE + 00AD
```

```
1 %SBTTL 'DEBLOCK_ATTR - deblock and validate attribute record'
1 GLOBAL ROUTINE DEBLOCK_ATTR(REC,MASK,ROUT): L_P$ NOVALUE=
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !       This routine deblocks an attribute record.  Validity checking of the
1 !       size and type values is done.
1 !
1 ! INPUT PARAMETERS:
1 !       REC             - Pointer to attribute record.
1 !       MASK            - Pointer to bitvector of valid attribute codes,
1 !                         or zero to consider all defined codes valid.
1 !       ROUT            - Routine to be called with each attribute.
1 !
1 ! IMPLICIT INPUTS:
1 !       NONE
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !       NONE
1 !
1 !--
1
2 BEGIN
2 MAP
2       REC:            REF BBLOCK,     ! Pointer to record
2       MASK:           REF BITVECTOR;  ! Pointer to bit mask
2 LOCAL
2       ATT:            REF BBLOCK,     ! Pointer to attribute
2       LIMIT;                          ! Limit address of record
2 L_DECL;
2
2
2 ! Initialize.
2 !
2 ATT = .REC + BRH$C_LENGTH;
2 LIMIT = .ATT + .REC[BRH$W_RSIZE];
2
2
2 ! Check the structure level.
2 !
2 IF .ATT[BSA$W_STRUCLEV] NEQ BBH$K_LEVEL1 THEN SIGNAL(BACKUP$_INVATTSTR);
2 ATT = .ATT + 2;
2
2
2 ! Loop over all attribute entries within the record.
2 !
2 WHILE .ATT LEQA .LIMIT - BSA$C_LENGTH DO
```

```
3          BEGIN
3          LOCAL
3              ALIMIT;                              ! Limit address of this attribute
3
3
3          ! Compute address of following byte and check that it is within the record.
3          !
3          ALIMIT = .ATT + .ATT[BSA$W_SIZE] + BSA$C_LENGTH;
3          IF .ALIMIT GTRA .LIMIT
3          THEN
4              BEGIN
4              SIGNAL(BACKUP$_INVATTSIZ);
4              EXITLOOP;
3              END;
3
3
3          ! If the type code is zero, exit the loop.
3          !
3          IF .ATT[BSA$W_TYPE] EQL 0
3          THEN
3              EXITLOOP;
3
3
3          ! Ensure that the attribute code is a valid code.  If it is, ensure that it
3          ! is valid for this type of attribute record.  If all tests are passed,
3          ! call the action routine to process the attribute.
3          !
3          IF
4              BEGIN
4              IF .ATT[BSA$W_TYPE] GEQU BSA$K_NUM_ATRS
4              THEN
4                  FALSE
4              ELSE
4                  IF .MASK EQL 0
4                  THEN
4                      TRUE
4                  ELSE
4                      .MASK[.ATT[BSA$W_TYPE]]
4              END
3          THEN
3              (.ROUT)(.ATT);
3
3
3          ! Advance to next attribute entry.
3          !
3          ATT = .ALIMIT;
2          END;
1      END;


                                  007C 00000        .ENTRY  DEBLOCK_ATTR, Save R2,R3,R4,R5,R6       ; 2033
                     56 00000000G  00  9E 00002        MOVAB   LIB$SIGNAL, R6                            ;
        53       04  AC            10  C1 00009        ADDL3   #16, REC, ATT                             ; 2076
                     52        04  BC  3C 0000E        MOVZWL  @REC, LIMIT                               ; 2077
```

```
                    52          53  C0 00012         ADDL2   ATT, LIMIT                          ;
              0101  8F          63  B1 00015         CMPW    (ATT), #257                         ; 2082
                    09          13  0001A            BEQL    1$                                  ;
                 00000000G      8F  DD 0001C         PUSHL   #BACKUP$_INVATTSTR                  ;
                    66          01  FB 00022         CALLS   #1, LIB$SIGNAL                      ;
                    53          02  C0 00025 1$:     ADDL2   #2, ATT                             ; 2083
                    55      FC  A2  9E 00028         MOVAB   -4(R2), R5                          ; 2088
                    55          53  D1 0002C 2$:     CMPL    ATT, R5                             ;
                                39  1A 0002F         BGTRU   6$                                  ;
                    50          63  3C 00031         MOVZWL  (ATT), R0                           ; 2096
                    54    04 A043   9E 00034         MOVAB   4(R0)[ATT], ALIMIT                  ;
                    52          54  D1 00039         CMPL    ALIMIT, LIMIT                       ; 2097
                                0A  1B 0003C         BLEQU   3$                                  ;
                 00000000G      8F  DD 0003E         PUSHL   #BACKUP$_INVATTSIZ                  ; 2100
                    66          01  FB 00044         CALLS   #1, LIB$SIGNAL                      ;
                                04  00047            RET                                         ; 2099
                    50      02  A3  3C 00048 3$:     MOVZWL  2(ATT), R0                          ; 2107
                                1C  13 0004C         BEQL    6$                                  ;
              0053  8F          50  B1 0004E         CMPW    R0, #83                             ; 2118
                                10  1E 00053         BGEQU   5$                                  ;
                            08  AC  D5 00055         TSTL    MASK                                ; 2122
                                05  13 00058         BEQL    4$                                  ;
        06          08  BC      50  E1 0005A         BBC     R0, @MASK, 5$                       ; 2126
                                53  DD 0005F 4$:     PUSHL   ATT                                 ; 2129
                    0C  BC      01  FB 00061         CALLS   #1, @ROUT                           ;
                    53          54  D0 00065 5$:     MOVL    ALIMIT, ATT                         ; 2134
                                C2  11 00068         BRB     2$                                  ; 2088
                                04  0006A 6$:        RET                                         ; 2136

; Routine Size:  107 bytes,    Routine Base:  CODE + 0164
```

```
%SBTTL 'FILE_ERROR - output file related error message'
GLOBAL ROUTINE FILE_ERROR(MESSAGE,FAB,EXTRA1,EXTRA2): NOVALUE=

!++
!
! FUNCTIONAL DESCRIPTION:
!       This routine signals a file-related message.
!
! INPUT PARAMETERS:
!       MESSAGE         - Message code for first message
!       FAB             - Pointer to file context area, from which file name
!                         will be obtained
!       Up to two additional input parameters are additional messages --
!       except that if the message is one of the special cases, they are
!       additional FAO arguments.
!
! IMPLICIT INPUTS:
!       NONE
!
! OUTPUT PARAMETERS:
!       NONE
!
! IMPLICIT OUTPUTS:
!       NONE
!
! ROUTINE VALUE:
!       NONE
!
! SIDE EFFECTS:
!       The messages are signalled.
!
!--

BEGIN
MAP
        FAB:            REF BBLOCK;     ! Pointer to FAB
LOCAL
        NAM:            REF BBLOCK,     ! Pointer to NAM block
        DESC:           VECTOR[2],      ! Descriptor for file name
        PARAM:          VECTOR[6];      ! Signal parameter list
BUILTIN
        ACTUALCOUNT,
        CALLG;


! Establish the file name to be printed.  The resultant string, expanded
! string, and filename string are examined in that order, and the first
! one that is not null is used.
!
NAM = .FAB[FAB$L_NAM];
IF .NAM[NAM$B_RSL] NEQ 0
THEN
    BEGIN
    DESC[0] = .NAM[NAM$B_RSL];
    DESC[1] = .NAM[NAM$L_RSA];
    END
ELSE IF .NAM[NAM$B_ESL] NEQ 0
```

```
2 THEN
3     BEGIN
3     DESC[0] = .NAM[NAM$B_ESL];
3     DESC[1] = .NAM[NAM$L_ESA];
3     END
2 ELSE
3     BEGIN
3     DESC[0] = .FAB[FAB$B_FNS];
3     DESC[1] = .FAB[FAB$L_FNA];
2     END;
2
2
2 ! Initialize the signal parameter list.
2 !
2 PARAM[0] = 3;                         ! Parameter count
2 PARAM[1] = .MESSAGE;                  ! First message code
2 PARAM[2] = 1;                         ! FAO argument count
2 PARAM[3] = DESC;                      ! Filename descriptor
2 IF ACTUALCOUNT() GEQ 3
2 THEN
3     BEGIN
3     PARAM[0] = .PARAM[0] + 1;         ! Increment parameter count
3     PARAM[4] = .EXTRA1;               ! Next message code
2     END;
2 IF ACTUALCOUNT() GEQ 4
2 THEN
3     BEGIN
3     PARAM[0] = .PARAM[0] + 1;         ! Increment parameter count
3     PARAM[5] = .EXTRA2;               ! Next message code
2     END;
2
2
2 ! For these messages, the additional parameters are FAO arguments.
2 !
2 IF
2     .MESSAGE EQL BACKUP$_VBNMISSING OR
2     .MESSAGE EQL BACKUP$_VBNERROR OR
2     .MESSAGE EQL BACKUP$_VBNPASTEOF
2 THEN
2     PARAM[2] = 3;
2
2
2 ! For these messages, EXTRA1 is an additional FAO argument that appears before
2 ! the file name.  EXTRA2, if present, continues to be a message code.
2 !
2 IF
2     .MESSAGE EQL BACKUP$_READBLOCK OR
2     .MESSAGE EQL BACKUP$_WRITEBLOCK OR
2     .MESSAGE EQL BACKUP$_VERIFYERR OR
2     .MESSAGE EQL BACKUP$_BADDATA OR
2     .MESSAGE EQL BACKUP$_SOFTRERRS OR
2     .MESSAGE EQL BACKUP$_SOFTWERRS OR
2     .MESSAGE EQL BACKUP$_XORERRS
2 THEN
3     BEGIN
3     PARAM[2] = 2;
3     PARAM[3] = .EXTRA1;
```

```
;  705     2251  3       PARAM[4] = DESC;
;  706     2252  2       END;
;  707     2253  2
;  708     2254  2
;  709     2255  2   ! Finally, signal the messages.
;  710     2256  2   !
;  711     2257  2   CALLG(PARAM, LIB$SIGNAL);
;  712     2258  1   END;


                                       0000 00000          .ENTRY  FILE_ERROR, Save nothing          ; 2138
                  5E              20    C2 00002           SUBL2   #32, SP                           ;
                  51        08    AC    D0 00005           MOVL    FAB, R1                           ; 2186
                  50        28    A1    D0 00009           MOVL    40(R1), NAM                       ;
                            03    A0    95 0000D           TSTB    3(NAM)                            ; 2187
                                  0C    13 00010           BEQL    1$                                ;
            18    AE        03    A0    9A 00012           MOVZBL  3(NAM), DESC                      ; 2190
            1C    AE        04    A0    D0 00017           MOVL    4(NAM), DESC+4                    ; 2191
                                  1B    11 0001C           BRB     3$                                ; 2187
                            0B    A0    95 0001E 1$:       TSTB    11(NAM)                           ; 2193
                                  0C    13 00021           BEQL    2$                                ;
            18    AE        0B    A0    9A 00023           MOVZBL  11(NAM), DESC                     ; 2196
            1C    AE        0C    A0    D0 00028           MOVL    12(NAM), DESC+4                   ; 2197
                                  0A    11 0002D           BRB     3$                                ; 2193
            18    AE        34    A1    9A 0002F 2$:       MOVZBL  52(R1), DESC                      ; 2201
            1C    AE        2C    A1    D0 00034           MOVL    44(R1), DESC+4                    ; 2202
                  6E              03    D0 00039 3$:       MOVL    #3, PARAM                         ; 2208
                  50        04    AC    D0 0003C           MOVL    MESSAGE, R0                       ; 2209
            04    AE              50    D0 00040           MOVL    R0, PARAM+4                       ;
            08    AE              01    D0 00044           MOVL    #1, PARAM+8                       ; 2210
            0C    AE        18    AE    9E 00048           MOVAB   DESC, PARAM+12                    ; 2211
                  03              6C    91 0004D           CMPB    (AP), #3                          ; 2212
                                  07    1F 00050           BLSSU   4$                                ;
                                  6E    D6 00052           INCL    PARAM                             ; 2215
            10    AE        0C    AC    D0 00054           MOVL    EXTRA1, PARAM+16                  ; 2216
                  04              6C    91 00059 4$:       CMPB    (AP), #4                          ; 2218
                                  07    1F 0005C           BLSSU   5$                                ;
                                  6E    D6 0005E           INCL    PARAM                             ; 2221
            14    AE        10    AC    D0 00060           MOVL    EXTRA2, PARAM+20                  ; 2222
     00000000G    8F              50    D1 00065 5$:       CMPL    R0, #BACKUP$_VBNMISSING           ; 2229
                                  12    13 0006C           BEQL    6$                                ;
     00000000G    8F              50    D1 0006E           CMPL    R0, #BACKUP$_VBNERROR             ; 2230
                                  09    13 00075           BEQL    6$                                ;
     00000000G    8F              50    D1 00077           CMPL    R0, #BACKUP$_VBNPASTEOF           ; 2231
                                  04    12 0007E           BNEQ    7$                                ;
            08    AE              03    D0 00080 6$:       MOVL    #3, PARAM+8                       ; 2233
     00000000G    8F              50    D1 00084 7$:       CMPL    R0, #BACKUP$_READBLOCK            ; 2240
                                  36    13 0008B           BEQL    8$                                ;
     00000000G    8F              50    D1 0008D           CMPL    R0, #BACKUP$_WRITEBLOCK           ; 2241
                                  2D    13 00094           BEQL    8$                                ;
     00000000G    8F              50    D1 00096           CMPL    R0, #BACKUP$_VERIFYERR            ; 2242
                                  24    13 0009D           BEQL    8$                                ;
     00000000G    8F              50    D1 0009F           CMPL    R0, #BACKUP$_BADDATA              ; 2243
                                  1B    13 000A6           BEQL    8$                                ;
```

```
         00000000G  8F          50  D1 000A8       CMPL    R0, #BACKUP$_SOFTRERRS         ; 2244
                                12  13 000AF       BEQL    8$                             ;
         00000000G  8F          50  D1 000B1       CMPL    R0, #BACKUP$_SOFTWERRS         ; 2245
                                09  13 000B8       BEQL    8$                             ;
         00000000G  8F          50  D1 000BA       CMPL    R0, #BACKUP$_XORERRS           ; 2246
                                0E  12 000C1       BNEQ    9$                             ;
                08  AE          02  D0 000C3 8$:   MOVL    #2, PARAM+8                    ; 2249
                0C  AE     0C   AC  D0 000C7       MOVL    EXTRA1, PARAM+12               ; 2250
                10  AE     18   AE  9E 000CC       MOVAB   DESC, PARAM+16                 ; 2251
         00000000G  00          6E  FA 000D1 9$:   CALLG   PARAM, LIB$SIGNAL              ; 2257
                                    04 000D8       RET                                    ; 2258

; Routine Size:  217 bytes,    Routine Base:  CODE + 01CF
```

```
1 %SBTTL 'FIND_BADBLOCK - find block in bad block table'
1 GLOBAL ROUTINE FIND_BADBLOCK(TABLE,LBN)=
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !       This routine searches a specified bad block table for a specified LBN.
1 !
1 ! INPUT PARAMETERS:
1 !       TABLE           - Pointer to bad block table.
1 !       LBN             - LBN to search for.
1 !
1 ! IMPLICIT INPUTS:
1 !       NONE
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       True if the LBN is found, false otherwise.
1 !
1 ! SIDE EFFECTS:
1 !       NONE
1 !
1 !--
1
2 BEGIN
2 MAP
2       TABLE:          REF BBLOCK;     ! Pointer to bad block table
2 LOCAL
2       DESC:           REF BBLOCK;     ! Pointer to bad block descriptor
2
2
2 ! If there is no table, assume not found.
2 !
2 IF .TABLE EQL 0 THEN RETURN FALSE;
2
2
2 ! Loop over all entries of the table.
2 !
2 DESC = TABLE[BAD_DESC];
2 INCR I FROM 0 TO .TABLE[BAD_NUMDESC]-1 DO
3     BEGIN
3
3     ! If the LBN is in the range described by this entry, return found.
3     !
3     IF
3         .LBN GEQU .DESC[BAD_LBN] AND
3         .LBN LEQU .DESC[BAD_LBN] + .DESC[BAD_COUNT] - 1
3     THEN
3         RETURN TRUE;
3
3
3     ! Advance to next entry.
```

```
;  771          2316  3          !
;  772          2317  3          DESC = .DESC + BAD_S_DESC;
;  773          2318  2          END;
;  774          2319  2
;  775          2320  2
;  776          2321  2 ! All entries examined, return not found.
;  777          2322  2 !
;  778          2323  2 FALSE
;  779          2324  1 END;


                                          000C 00000          .ENTRY  FIND_BADBLOCK, Save R2,R3              ; 2260
                          52        04  AC  D0 00002          MOVL    TABLE, R2                              ; 2297
                                    27  13 00006              BEQL    4$
                          50        08  A2  9E 00008          MOVAB   8(R2), DESC                            ; 2302
                          53            01  CE 0000C          MNEGL   #1, I                                  ; 2309
                                        1A  11 0000F          BRB     3$                                     ;
                          60        08  AC  D1 00011 1$:      CMPL    LBN, (DESC)                            ;
                                        11  1F 00015          BLSSU   2$                                     ;
                51        60        04  A0  C1 00017          ADDL3   4(DESC), (DESC), R1                    ; 2310
                                        51  D7 0001C          DECL    R1                                     ;
                          51        08  AC  D1 0001E          CMPL    LBN, R1                                ;
                                        04  1A 00022          BGTRU   2$                                     ;
                          50            01  D0 00024          MOVL    #1, R0                                 ; 2312
                                            04 00027          RET                                            ;
                          50            08  C0 00028 2$:      ADDL2   #8, DESC                               ; 2317
                E2        53            62  F2 0002B 3$:      AOBLSS  (R2), I, 1$                            ; 2303
                                        50  D4 0002F 4$:      CLRL    R0                                     ; 2324
                                            04 00031          RET                                            ;

; Routine Size:  50 bytes,    Routine Base:  CODE + 02A8
```

```
1 %SBTTL 'EXTRACT_FILENAME - get file name from FAB'
1 GLOBAL ROUTINE EXTRACT_FILENAME(FAB,DESC): NOVALUE=
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !       This routine extracts the filename, and filetype if it is not null,
1 !       from the resultant string associated with FAB and returns a
1 !       descriptor in DESC.
1 !
1 ! INPUT PARAMETERS:
1 !       FAB             - Pointer to the FAB
1 !       DESC            - Pointer to the output descriptor
1 !
1 ! IMPLICIT INPUTS:
1 !       NONE
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !       NONE
1 !
1 !--
1
2 BEGIN
2 MAP
2       FAB:            REF BBLOCK,             ! Pointer to FAB
2       DESC:           REF BBLOCK;             ! Pointer to descriptor
2
2
2 ! Construct a descriptor for the file name and file type, concatenated.
2 !
2 DESC[DSC$W_LENGTH] = .BBLOCK[FAB[FC_NAM], NAM$B_NAME]
2                   + .BBLOCK[FAB[FC_NAM], NAM$B_TYPE];
2 DESC[DSC$A_POINTER] = .BBLOCK[FAB[FC_NAM], NAM$L_NAME];
2
1 END;
```

```
                                000C 00000      .ENTRY  EXTRACT_FILENAME, Save R2,R3       ; 2326
                  50      04  AC  7D 00002      MOVQ    FAB, R0                            ; 2364
                  52    00CF  C0  9A 00006      MOVZBL  207(R0), R2                        ; 2365
                  53    00D0  C0  9A 0000B      MOVZBL  208(R0), R3                        ;
     61           52          53  A1 00010      ADDW3   R3, R2, (R1)                       ;
            04    A1    00E0  C0  D0 00014      MOVL    224(R0), 4(R1)                     ; 2366
                                   04 0001A     RET                                        ; 2368
```

```
; Routine Size:  27 bytes,    Routine Base:  CODE + 02DA
```

```
1 %SBTTL 'EXTRACT_DIR_FILENAME - get directory and file name from FAB'
1 GLOBAL ROUTINE EXTRACT_DIR_FILENAME(FAB,DESC): NOVALUE=
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !       This routine extracts the directory, name, type, and version from the
1 !       resultant string associated with FAB and returns a descriptor in DESC.
1 !
1 ! INPUT PARAMETERS:
1 !       FAB             - Pointer to the FAB
1 !       DESC            - Pointer to the output descriptor
1 !
1 ! IMPLICIT INPUTS:
1 !       NONE
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !       NONE
1 !
1 !--
1
2 BEGIN
2 MAP
2       FAB:            REF BBLOCK,             ! Pointer to FAB
2       DESC:           REF BBLOCK;             ! Pointer to descriptor
2
2
2 ! Construct a descriptor for the portion of the resultant string beginning
2 ! with the directory name.
2 !
2 DESC[DSC$W_LENGTH] =
2     .BBLOCK[FAB[FC_NAM], NAM$L_RSA] +
2     .BBLOCK[FAB[FC_NAM], NAM$B_RSL] -
2     .BBLOCK[FAB[FC_NAM], NAM$L_DIR];
2 DESC[DSC$A_POINTER] = .BBLOCK[FAB[FC_NAM], NAM$L_DIR];
1 END;


                          0004 00000        .ENTRY  EXTRACT_DIR_FILENAME, Save R2    ; 2370
        50          04  AC   7D 00002        MOVQ    FAB, R0                          ; 2409
        52        0097  C0   9A 00006        MOVZBL  151(R0), R2                      ; 2410
        52        0098  C0   C0 0000B        ADDL2   152(R0), R2                      ;
  61    52        00DC  C0   A3 00010        SUBW3   220(R0), R2, (R1)                ; 2411
     04 A1        00DC  C0   D0 00016        MOVL    220(R0), 4(R1)                   ; 2412
                             04 0001C        RET                                      ; 2413
```

```
; Routine Size:  29 bytes,    Routine Base:  CODE + 02F5
```

```
1 %SBTTL 'GET_ONE_ATTRIBUTE - save value of specified attribute'
1 ROUTINE GET_ONE_ATTRIBUTE(ATT): NOVALUE=
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !       This routine saves the value of one attribute.
1 !
1 ! INPUT PARAMETERS:
1 !       ATT             - Pointer to attribute entry from save set record.
1 !
1 ! IMPLICIT INPUTS:
1 !       NONE
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       OUTPUT_ATTBUF   - Contains the attribute value.
1 !
1 ! ROUTINE VALUE:
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !       NONE
1 !
1 !--
1
2 BEGIN
2 MAP
2       ATT:            REF BBLOCK;     ! Pointer to attribute entry
2 LOCAL
2       DLEN,                           ! Length of attribute storage
2       DST:            REF BBLOCK;     ! Pointer to attribute storage
2
2
2 ! Initially, dispatch on the attribute code to establish where in OUTPUT_ATTBUF
2 ! the value of the attribute is stored.  Also, establish whether a descriptor
2 ! for, or the value of, the attribute will be stored.  If the value, establish
2 ! the length of the attribute storage.
2 !
2 IF
3     BEGIN
3     CASE .ATT[BSA$W_TYPE] FROM BSA$K_SSNAME TO BSA$K_NUM_ATRS-1 OF
3         SET
3
3         [BSA$K_SSNAME]:
4             BEGIN
4             DST = OUTPUT_ATTBUF[BSR_SSNAME];
4             TRUE
3             END;
3
3         [BSA$K_COMMAND]:
4             BEGIN
4             DST = OUTPUT_ATTBUF[BSR_COMMAND];
4             TRUE
3             END;
```

```
        [BSA$K_COMMENT]:
            BEGIN
            DST = OUTPUT_ATTBUF[BSR_COMMENT];
            TRUE
            END;

        [BSA$K_USERNAME]:
            BEGIN
            DST = OUTPUT_ATTBUF[BSR_USERNAME];
            TRUE
            END;

        [BSA$K_USERUIC]:
            BEGIN
            DST = OUTPUT_ATTBUF[BSR_USERUIC];
            DLEN = 4;
            FALSE
            END;

        [BSA$K_DATE]:
            BEGIN
            DST = OUTPUT_ATTBUF[BSR_DATE];
            DLEN = 8;
            FALSE
            END;

        [BSA$K_OPSYS]:
            BEGIN
            DST = OUTPUT_ATTBUF[BSR_OPSYS];
            DLEN = 2;
            FALSE
            END;

        [BSA$K_SYSVER]:
            BEGIN
            DST = OUTPUT_ATTBUF[BSR_SYSVER];
            TRUE
            END;

        [BSA$K_NODENAME]:
            BEGIN
            DST = OUTPUT_ATTBUF[BSR_NODENAME];
            TRUE
            END;

        [BSA$K_SIR]:
            BEGIN
            DST = OUTPUT_ATTBUF[BSR_SIR];
            DLEN = 4;
            FALSE
            END;

        [BSA$K_DRIVEID]:
            BEGIN
            DST = OUTPUT_ATTBUF[BSR_DRIVEID];
            TRUE
```

```
            END;

        [BSA$K_BACKVER]:
            BEGIN
            DST = OUTPUT_ATTBUF[BSR_BACKVER];
            TRUE
            END;

        [BSA$K_BLOCKSIZE]:
            BEGIN
            DST = OUTPUT_ATTBUF[BSR_BLOCKSIZE];
            DLEN = 4;
            FALSE
            END;

        [BSA$K_XORSIZE]:
            BEGIN
            DST = OUTPUT_ATTBUF[BSR_XORSIZE];
            DLEN = 2;
            FALSE
            END;

        [BSA$K_BUFFERS]:
            BEGIN
            DST = OUTPUT_ATTBUF[BSR_BUFFERS];
            DLEN = 2;
            FALSE
            END;

        [BSA$K_VOLSETNAM]:
            BEGIN
            DST = OUTPUT_ATTBUF[BSR_VOLSETNAM];
            TRUE
            END;

        [BSA$K_NVOLS]:
            BEGIN
            DST = OUTPUT_ATTBUF[BSR_NVOLS];
            DLEN = 2;
            FALSE
            END;

        [BSA$K_BACKSIZE]:
            BEGIN
            DST = OUTPUT_ATTBUF[BSR_BACKSIZE];
            DLEN = 8;
            FALSE
            END;

        [BSA$K_BACKFILES]:
            BEGIN
            DST = OUTPUT_ATTBUF[BSR_BACKFILES];
            DLEN = 4;
            FALSE
            END;

        [BSA$K_VOLSTRUCT]:
```

```
4              BEGIN
4              DST = OUTPUT_ATTBUF[VSR_VOLSTRUCT];
4              DLEN = 2;
4              FALSE
3              END;
3
3          [BSA$K_VOLNAME]:
4              BEGIN
4              DST = OUTPUT_ATTBUF[VSR_VOLNAME];
4              TRUE
3              END;
3
3          [BSA$K_OWNERNAME]:
4              BEGIN
4              DST = OUTPUT_ATTBUF[VSR_OWNERNAME];
4              TRUE
3              END;
3
3          [BSA$K_FORMAT]:
4              BEGIN
4              DST = OUTPUT_ATTBUF[VSR_FORMAT];
4              TRUE
3              END;
3
3          [BSA$K_RVN]:
4              BEGIN
4              DST = OUTPUT_ATTBUF[VSR_RVN];
4              DLEN = 2;
4              FALSE
3              END;
3
3          [BSA$K_VOLOWNER]:
4              BEGIN
4              DST = OUTPUT_ATTBUF[VSR_VOLOWNER];
4              DLEN = 4;
4              FALSE
3              END;
3
3          [BSA$K_PROTECT]:
4              BEGIN
4              DST = OUTPUT_ATTBUF[VSR_PROTECT];
4              DLEN = 2;
4              FALSE
3              END;
3
3          [BSA$K_FILEPROT]:
4              BEGIN
4              DST = OUTPUT_ATTBUF[VSR_FILEPROT];
4              DLEN = 2;
4              FALSE
3              END;
3
3          [BSA$K_RECPROT]:
4              BEGIN
4              DST = OUTPUT_ATTBUF[VSR_RECPROT];
4              DLEN = 2;
4              FALSE
```

```
            END;

        [BSA$K_VOLCHAR]:
            BEGIN
            DST = OUTPUT_ATTBUF[VSR_VOLCHAR];
            DLEN = 2;
            FALSE
            END;

        [BSA$K_VOLDATE]:
            BEGIN
            DST = OUTPUT_ATTBUF[VSR_VOLDATE];
            DLEN = 8;
            FALSE
            END;

        [BSA$K_WINDOW]:
            BEGIN
            DST = OUTPUT_ATTBUF[VSR_WINDOW];
            DLEN = 1;
            FALSE
            END;

        [BSA$K_LRU_LIM]:
            BEGIN
            DST = OUTPUT_ATTBUF[VSR_LRU_LIM];
            DLEN = 1;
            FALSE
            END;

        [BSA$K_EXTEND]:
            BEGIN
            DST = OUTPUT_ATTBUF[VSR_EXTEND];
            DLEN = 2;
            FALSE
            END;

        [BSA$K_CLUSTER]:
            BEGIN
            DST = OUTPUT_ATTBUF[VSR_CLUSTER];
            DLEN = 2;
            FALSE
            END;

        [BSA$K_RESFILES]:
            BEGIN
            DST = OUTPUT_ATTBUF[VSR_RESFILES];
            DLEN = 2;
            FALSE
            END;

        [BSA$K_VOLSIZE]:
            BEGIN
            DST = OUTPUT_ATTBUF[VSR_VOLSIZE];
            DLEN = 4;
            FALSE
            END;
```

```

        [BSA$K_TOTSIZE]:
            BEGIN
            DST = OUTPUT_ATTBUF[VSR_TOTSIZE];
            DLEN = 8;
            FALSE
            END;

        [BSA$K_TOTFILES]:
            BEGIN
            DST = OUTPUT_ATTBUF[VSR_TOTFILES];
            DLEN = 4;
            FALSE
            END;

        [BSA$K_MAXFILES]:
            BEGIN
            DST = OUTPUT_ATTBUF[VSR_MAXFILES];
            DLEN = 4;
            FALSE
            END;

        [BSA$K_MAXFILNUM]:
            BEGIN
            DST = OUTPUT_ATTBUF[VSR_MAXFILNUM];
            DLEN = 4;
            FALSE
            END;

        [BSA$K_SERIALNUM]:
            BEGIN
            DST = OUTPUT_ATTBUF[VSR_SERIALNUM];
            DLEN = 4;
            FALSE
            END;

        [BSA$K_FILENAME]:
            BEGIN
            DST = OUTPUT_ATTBUF[FAR_FILENAME];
            TRUE
            END;

        [BSA$K_STRUCLEV]:
            BEGIN
            DST = OUTPUT_ATTBUF[FAR_STRUCLEV];
            DLEN = 2;
            FALSE
            END;

        [BSA$K_FID]:
            BEGIN
            DST = OUTPUT_ATTBUF[FAR_FID];
            DLEN = 6;
            FALSE
            END;

        [BSA$K_BACKLINK]:
```

```
4               BEGIN
4               DST = OUTPUT_ATTBUF[FAR_BACKLINK];
4               DLEN = 6;
4               FALSE
3               END;
3
3       [BSA$K_FILESIZE]:
4               BEGIN
4               DST = OUTPUT_ATTBUF[FAR_FILESIZE];
4               DLEN = 4;
4               FALSE
3               END;
3
3       [BSA$K_UIC]:
4               BEGIN
4               DST = OUTPUT_ATTBUF[FAR_UIC];
4               DLEN = 4;
4               FALSE
3               END;
3
3       [BSA$K_FPRO]:
4               BEGIN
4               DST = OUTPUT_ATTBUF[FAR_FPRO];
4               DLEN = 2;
4               FALSE
3               END;
3
3       [BSA$K_RPRO]:
4               BEGIN
4               DST = OUTPUT_ATTBUF[FAR_RPRO];
4               DLEN = 2;
4               FALSE
3               END;
3
3       [BSA$K_ACLEVEL]:
4               BEGIN
4               DST = OUTPUT_ATTBUF[FAR_ACLEVEL];
4               DLEN = 1;
4               FALSE
3               END;
3
3       [BSA$K_UCHAR]:
4               BEGIN
4               DST = OUTPUT_ATTBUF[FAR_UCHAR];
4               DLEN = 4;
4               FALSE
3               END;
3
3       [BSA$K_RECATTR]:
4               BEGIN
4               DST = OUTPUT_ATTBUF[FAR_RECATTR];
4               DLEN = 32;
4               FALSE
3               END;
3
3       [BSA$K_REVISION]:
4               BEGIN
```

```
4                   DST = OUTPUT_ATTBUF[FAR_REVISION];
4                   DLEN = 2;
4                   FALSE
3                   END;
3
3               [BSA$K_CREDATE]:
4                   BEGIN
4                   DST = OUTPUT_ATTBUF[FAR_CREDATE];
4                   DLEN = 8;
4                   FALSE
3                   END;
3
3               [BSA$K_REVDATE]:
4                   BEGIN
4                   DST = OUTPUT_ATTBUF[FAR_REVDATE];
4                   DLEN = 8;
4                   FALSE
3                   END;
3
3               [BSA$K_EXPDATE]:
4                   BEGIN
4                   DST = OUTPUT_ATTBUF[FAR_EXPDATE];
4                   DLEN = 8;
4                   FALSE
3                   END;
3
3               [BSA$K_BAKDATE]:
4                   BEGIN
4                   DST = OUTPUT_ATTBUF[FAR_BAKDATE];
4                   DLEN = 8;
4                   FALSE
3                   END;
3
3               [BSA$K_SECTORS]:
4                   BEGIN
4                   DST = OUTPUT_ATTBUF[PVA_SECTORS];
4                   DLEN = 1;
4                   FALSE
3                   END;
3
3               [BSA$K_TRACKS]:
4                   BEGIN
4                   DST = OUTPUT_ATTBUF[PVA_TRACKS];
4                   DLEN = 1;
4                   FALSE
3                   END;
3
3               [BSA$K_CYLINDERS]:
4                   BEGIN
4                   DST = OUTPUT_ATTBUF[PVA_CYLINDERS];
4                   DLEN = 2;
4                   FALSE
3                   END;
3
3               [BSA$K_MAXBLOCK]:
4                   BEGIN
4                   DST = OUTPUT_ATTBUF[PVA_MAXBLOCK];
```

```
                DLEN = 4;
                FALSE
                END;

            [BSA$K_DEVTYP]:
                BEGIN
                DST = OUTPUT_ATTBUF[PVA_DEVTYP];
                DLEN = 4;
                FALSE
                END;

            [BSA$K_SERIAL]:
                BEGIN
                DST = OUTPUT_ATTBUF[PVA_SERIAL];
                DLEN = 4;
                FALSE
                END;

            [BSA$K_DEVNAM]:
                BEGIN
                DST = OUTPUT_ATTBUF[PVA_DEVNAM];
                TRUE
                END;

            [BSA$K_LABEL]:
                BEGIN
                DST = OUTPUT_ATTBUF[PVA_LABEL];
                TRUE
                END;

            [BSA$K_BADBLOCK]:
                BEGIN
                DST = OUTPUT_ATTBUF[PVA_BADBLOCK];
                TRUE
                END;

            [BSA$K_INDEXLBN]:
                BEGIN
                DST = OUTPUT_ATTBUF[VSR_INDEXLBN];
                DLEN = 4;
                FALSE
                END;

            [BSA$K_BOOTBLOCK]:
                BEGIN
                DST = OUTPUT_ATTBUF[VSR_BOOTBLOCK];
                TRUE
                END;

            [BSA$K_BOOTVBN]:
                BEGIN
                DST = OUTPUT_ATTBUF[FAR_BOOTVBN];
                DLEN = 4;
                FALSE
                END;

            [BSA$K_PLACEMENT]:
```

```
4               BEGIN
4               DST = OUTPUT_ATTBUF[FAR_PLACEMENT];
4               TRUE
3               END;
3
3           [BSA$K_DIR_UIC]:
4               BEGIN
4               DST = OUTPUT_ATTBUF[FAR_DIR_UIC];
4               DLEN = 4;
4               FALSE
3               END;
3
3           [BSA$K_DIR_FPRO]:
4               BEGIN
4               DST = OUTPUT_ATTBUF[FAR_DIR_FPRO];
4               DLEN = 2;
4               FALSE
3               END;
3
3           [BSA$K_DIR_STATUS]:
4               BEGIN
4               DST = OUTPUT_ATTBUF[FAR_DIR_STATUS];
4               DLEN = 1;
4               FALSE
3               END;
3
3           [BSA$K_DIR_VERLIM]:
4               BEGIN
4               DST = OUTPUT_ATTBUF[FAR_DIR_VERLIM];
4               DLEN = 2;
4               FALSE
3               END;
3
3           [BSA$K_VERLIMIT]:
4               BEGIN
4               DST = OUTPUT_ATTBUF[FAR_VERLIMIT];
4               DLEN = 2;
4               FALSE
3               END;
3
3           [BSA$K_RETAINMIN]:
4               BEGIN
4               DST = OUTPUT_ATTBUF[VSR_RETAINMIN];
4               DLEN = 8;
4               FALSE
3               END;
3
3           [BSA$K_RETAINMAX]:
4               BEGIN
4               DST = OUTPUT_ATTBUF[VSR_RETAINMAX];
4               DLEN = 8;
4               FALSE
3               END;
3
3           [BSA$K_ACLSEGMENT]:
4               BEGIN
4               DST = OUTPUT_ATTBUF[FAR_ACLSEGMENT];
```

```
4                TRUE
3                END;
3
3            [BSA$K_HIGHWATER]:
4                BEGIN
4                DST = OUTPUT_ATTBUF[FAR_HIGHWATER];
4                DLEN = 4;
4                FALSE
3                END;
3
3            [BSA$K_JNL_FLAGS]:
4                BEGIN
4                DST = OUTPUT_ATTBUF[FAR_JNL_FLAGS];
4                DLEN = 2;
4                FALSE
3                END;
3
3            [BSA$K_CRYPDATKEY]:
4                BEGIN
4                DST = OUTPUT_ATTBUF[BSR_CRYPTDATA];
4                DLEN = BSA$S_CRYPDATKEY;
4                FALSE
3                END;
3
3            [INRANGE,OUTRANGE]:
3                RETURN;
3
3            TES
3        END
2    THEN
3        BEGIN
3        !
3        ! Descriptor-valued attribute.  Put a descriptor at location DST.
3        !
3        DST[DSC$W_LENGTH] = .ATT[BSA$W_SIZE];
3        DST[DSC$A_POINTER] = ATT[BSA$C_LENGTH,0,0,0];
3        END
2    ELSE
3        BEGIN
3        !
3        ! Not a descriptor.  Zero-extend to length DLEN at location DST.
3        ! For some special cases, copy the attribute value to a global variable
3        ! for the READSAVE routines.
3        !
3        CH$COPY(.ATT[BSA$W_SIZE], ATT[BSA$C_LENGTH,0,0,0], 0, .DLEN, .DST);
3        IF .ATT[BSA$W_TYPE] EQL BSA$K_BUFFERS
3            THEN RWSV_LOOKAHEAD = .OUTPUT_ATTBUF[BSR_BUFFERS];
3        IF .ATT[BSA$W_TYPE] EQL BSA$K_XORSIZE
3            THEN RWSV_XORSIZE = .OUTPUT_ATTBUF[BSR_XORSIZE];
2        END;
1    END;
```

```
00FC 00000 GET_ONE_ATTRIBUTE:
```

```
                                                                  .WORD    Save R2,R3,R4,R5,R6,R7                       ; 2415
                                57 00000000'   EF  9E 00002       MOVAB    OUTPUT_ATTBUF+24, R7                         ;
                                52        04   AC  D0 00009       MOVL     ATT, R2                                      ; 3019
                                56        04   AC  D0 0000D       MOVL     ATT, R6                                      ; 2457
               0051    8F       01        02   A6  AF 00011       CASEW    2(R6), #1, #81                               ;
     00A5          01D5         01E7          01CF    00018 1$:   .WORD    64$-1$,-                                     ;
     00BC          00B6         00B0          00AA    00020                69$-1$,-                                     ;
     00D4          00CE         00C8          00C2    00028                65$-1$,-                                     ;
     00EC          00E6         00E0          00DA    00030                2$-1$,-                                      ;
     0105          00FF         00F9          00F3    00038                3$-1$,-                                      ;
     010B          01D5         01E7          01CF    00040                4$-1$,-                                      ;
     0123          011D         0117          0111    00048                5$-1$,-                                      ;
     013A          0134         012F          0129    00050                6$-1$,-                                      ;
     0152          014C         0146          0140    00058                7$-1$,-                                      ;
     016A          0164         015E          0158    00060                8$-1$,-                                      ;
     0176          00B6         01CF          0170    00068                9$-1$,-                                      ;
     00E0          00C8         00AA          017C    00070                10$-1$,-                                     ;
     018B          00DA         0185          00E6    00078                11$-1$,-                                     ;
     01A0          019A         0194          00F3    00080                12$-1$,-                                     ;
     01B8          01B2         01AC          01A6    00088                13$-1$,-                                     ;
     01CF          01C9         01C3          01BE    00090                14$-1$,-                                     ;
     01E1          01DB         01D5          01E7    00098                16$-1$,-                                     ;
     01F3          01ED         01E7          00FF    000A0                17$-1$,-                                     ;
     020E          0208         0202          01F9    000A8                18$-1$,-                                     ;
     022C          0223         021D          0214    000B0                19$-1$,-                                     ;
                                0268          0235    000B8                64$-1$,-                                     ;
                                                                           69$-1$,-                                     ;
                                                                           65$-1$,-                                     ;
                                                                           20$-1$,-                                     ;
                                                                           21$-1$,-                                     ;
                                                                           23$-1$,-                                     ;
                                                                           24$-1$,-                                     ;
                                                                           25$-1$,-                                     ;
                                                                           26$-1$,-                                     ;
                                                                           28$-1$,-                                     ;
                                                                           30$-1$,-                                     ;
                                                                           31$-1$,-                                     ;
                                                                           32$-1$,-                                     ;
                                                                           33$-1$,-                                     ;
                                                                           34$-1$,-                                     ;
                                                                           36$-1$,-                                     ;
                                                                           38$-1$,-                                     ;
                                                                           40$-1$,-                                     ;
                                                                           41$-1$,-                                     ;
                                                                           42$-1$,-                                     ;
                                                                           43$-1$,-                                     ;
                                                                           64$-1$,-                                     ;
                                                                           5$-1$,-                                      ;
                                                                           45$-1$,-                                     ;
                                                                           46$-1$,-                                     ;
                                                                           3$-1$,-                                      ;
                                                                           8$-1$,-                                      ;
                                                                           12$-1$,-                                     ;
                                                                           13$-1$,-                                     ;
                                                                           48$-1$,-                                     ;
                                                                           11$-1$,-                                     ;
                                                                           49$-1$,-                                     ;
```

```
                                                                16$-1$,-                                     ;
                                                                50$-1$,-                                     ;
                                                                51$-1$,-                                     ;
                                                                52$-1$,-                                     ;
                                                                53$-1$,-                                     ;
                                                                55$-1$,-                                     ;
                                                                56$-1$,-                                     ;
                                                                58$-1$,-                                     ;
                                                                60$-1$,-                                     ;
                                                                61$-1$,-                                     ;
                                                                62$-1$,-                                     ;
                                                                64$-1$,-                                     ;
                                                                69$-1$,-                                     ;
                                                                65$-1$,-                                     ;
                                                                66$-1$,-                                     ;
                                                                68$-1$,-                                     ;
                                                                18$-1$,-                                     ;
                                                                69$-1$,-                                     ;
                                                                70$-1$,-                                     ;
                                                                72$-1$,-                                     ;
                                                                73$-1$,-                                     ;
                                                                76$-1$,-                                     ;
                                                                77$-1$,-                                     ;
                                                                78$-1$,-                                     ;
                                                                79$-1$,-                                     ;
                                                                81$-1$,-                                     ;
                                                                82$-1$,-                                     ;
                                                                84$-1$,-                                     ;
                                                                86$-1$,-                                     ;
                                                                90$-1$                                       ;
                            04  000BC          RET                                                           ; 3009
     50              67     9E  000BD  2$:     MOVAB    OUTPUT_ATTBUF+24, DST                                ; 2480
                     46     11  000C0          BRB      15$                                                  ;
     50      44      A7     9E  000C2  3$:     MOVAB    OUTPUT_ATTBUF+92, DST                                ; 2486
                     65     11  000C6          BRB      22$                                                  ; 2487
     50      0C      A7     9E  000C8  4$:     MOVAB    OUTPUT_ATTBUF+36, DST                                ; 2493
                     7C     11  000CC          BRB      29$                                                  ; 2494
     50      54      A7     9E  000CE  5$:     MOVAB    OUTPUT_ATTBUF+108, DST                               ; 2500
                     71     11  000D2          BRB      27$                                                  ; 2501
     50      14      A7     9E  000D4  6$:     MOVAB    OUTPUT_ATTBUF+44, DST                                ; 2507
                     2E     11  000D8          BRB      15$                                                  ;
     50      1C      A7     9E  000DA  7$:     MOVAB    OUTPUT_ATTBUF+52, DST                                ; 2513
                     28     11  000DE          BRB      15$                                                  ;
     50      48      A7     9E  000E0  8$:     MOVAB    OUTPUT_ATTBUF+96, DST                                ; 2519
                     47     11  000E4          BRB      22$                                                  ; 2520
     50      24      A7     9E  000E6  9$:     MOVAB    OUTPUT_ATTBUF+60, DST                                ; 2526
                     1C     11  000EA          BRB      15$                                                  ;
     50      2C      A7     9E  000EC  10$:    MOVAB    OUTPUT_ATTBUF+68, DST                                ; 2532
                     16     11  000F0          BRB      15$                                                  ;
     50      4C      A7     9E  000F2  11$:    MOVAB    OUTPUT_ATTBUF+100, DST                               ; 2538
                     76     11  000F6          BRB      37$                                                  ; 2539
     50      56      A7     9E  000F8  12$:    MOVAB    OUTPUT_ATTBUF+110, DST                               ; 2545
                     6A     11  000FC          BRB      35$                                                  ; 2546
     50      58      A7     9E  000FE  13$:    MOVAB    OUTPUT_ATTBUF+112, DST                               ; 2552
                     64     11  00102          BRB      35$                                                  ; 2553
     50      34      A7     9E  00104  14$:    MOVAB    OUTPUT_ATTBUF+76, DST                                ; 2559
                   014B     31  00108  15$:    BRW      87$                                                  ;
```

```
50      5A  A7  9E 0010B 16$:    MOVAB   OUTPUT_ATTBUF+114, DST          ; 2565
            57  11 0010F         BRB     35$                             ; 2566
50      3C  A7  9E 00111 17$:    MOVAB   OUTPUT_ATTBUF+84, DST           ; 2572
            5D  11 00115         BRB     39$                             ; 2573
50      50  A7  9E 00117 18$:    MOVAB   OUTPUT_ATTBUF+104, DST          ; 2579
            6F  11 0011B         BRB     44$                             ; 2580
50      28  A7  9E 0011D 19$:    MOVAB   OUTPUT_ATTBUF+64, DST           ; 2586
            45  11 00121         BRB     35$                             ; 2587
50      2A  A7  9E 00123 20$:    MOVAB   OUTPUT_ATTBUF+66, DST           ; 2611
            3F  11 00127         BRB     35$                             ; 2612
50      10  A7  9E 00129 21$:    MOVAB   OUTPUT_ATTBUF+40, DST           ; 2618
            5D  11 0012D 22$:    BRB     44$                             ; 2619
50      2C  A7  9E 0012F 23$:    MOVAB   OUTPUT_ATTBUF+68, DST           ; 2625
            33  11 00133         BRB     35$                             ; 2626
50      2E  A7  9E 00135 24$:    MOVAB   OUTPUT_ATTBUF+70, DST           ; 2632
            2D  11 00139         BRB     35$                             ; 2633
50      30  A7  9E 0013B 25$:    MOVAB   OUTPUT_ATTBUF+72, DST           ; 2639
            27  11 0013F         BRB     35$                             ; 2640
50      32  A7  9E 00141 26$:    MOVAB   OUTPUT_ATTBUF+74, DST           ; 2646
            21  11 00145 27$:    BRB     35$                             ; 2647
50          67  9E 00147 28$:    MOVAB   OUTPUT_ATTBUF+24, DST           ; 2653
            76  11 0014A 29$:    BRB     54$                             ; 2654
50      3A  A7  9E 0014C 30$:    MOVAB   OUTPUT_ATTBUF+82, DST           ; 2660
            7C  11 00150         BRB     57$                             ; 2661
50      3B  A7  9E 00152 31$:    MOVAB   OUTPUT_ATTBUF+83, DST           ; 2667
            76  11 00156         BRB     57$                             ; 2668
50      34  A7  9E 00158 32$:    MOVAB   OUTPUT_ATTBUF+76, DST           ; 2674
            76  11 0015C         BRB     59$                             ; 2675
50      36  A7  9E 0015E 33$:    MOVAB   OUTPUT_ATTBUF+78, DST           ; 2681
            70  11 00162         BRB     59$                             ; 2682
50      38  A7  9E 00164 34$:    MOVAB   OUTPUT_ATTBUF+80, DST           ; 2688
            6A  11 00168 35$:    BRB     59$                             ; 2689
50      14  A7  9E 0016A 36$:    MOVAB   OUTPUT_ATTBUF+44, DST           ; 2695
            75  11 0016E 37$:    BRB     63$                             ; 2696
50      08  A7  9E 00170 38$:    MOVAB   OUTPUT_ATTBUF+32, DST           ; 2702
            4C  11 00174 39$:    BRB     54$                             ; 2703
50      18  A7  9E 00176 40$:    MOVAB   OUTPUT_ATTBUF+48, DST           ; 2709
            7B  11 0017A         BRB     67$                             ; 2710
50      1C  A7  9E 0017C 41$:    MOVAB   OUTPUT_ATTBUF+52, DST           ; 2716
            75  11 00180         BRB     67$                             ; 2717
50      20  A7  9E 00182 42$:    MOVAB   OUTPUT_ATTBUF+56, DST           ; 2723
            6F  11 00186         BRB     67$                             ; 2724
50      24  A7  9E 00188 43$:    MOVAB   OUTPUT_ATTBUF+60, DST           ; 2730
            7B  11 0018C 44$:    BRB     71$                             ; 2731
50      38  A7  9E 0018E 45$:    MOVAB   OUTPUT_ATTBUF+80, DST           ; 2750
            04  11 00192         BRB     47$                             ; 2751
50      3E  A7  9E 00194 46$:    MOVAB   OUTPUT_ATTBUF+86, DST           ; 2757
51          06  D0 00198 47$:    MOVL    #6, DLEN                        ; 2758
            7B  11 0019B         BRB     75$                             ;
50      62  A7  9E 0019D 48$:    MOVAB   OUTPUT_ATTBUF+122, DST          ; 2792
            72  11 001A1         BRB     74$                             ; 2793
50      F8  A7  9E 001A3 49$:    MOVAB   OUTPUT_ATTBUF+16, DST           ; 2806
51          20  D0 001A7         MOVL    #32, DLEN                       ; 2807
            6C  11 001AA         BRB     75$                             ;
50      18  A7  9E 001AC 50$:    MOVAB   OUTPUT_ATTBUF+48, DST           ; 2820
            7E  11 001B0         BRB     80$                             ; 2821
50      20  A7  9E 001B2 51$:    MOVAB   OUTPUT_ATTBUF+56, DST           ; 2827
```

```
                          78  11 001B6           BRB     80$                                            ; 2828
            50      28    A7  9E 001B8  52$:     MOVAB   OUTPUT_ATTBUF+64, DST                          ; 2834
                          72  11 001BC           BRB     80$                                            ; 2835
            50      30    A7  9E 001BE  53$:     MOVAB   OUTPUT_ATTBUF+72, DST                          ; 2841
                          6C  11 001C2  54$:     BRB     80$                                            ; 2842
            50      0A    A7  9E 001C4  55$:     MOVAB   OUTPUT_ATTBUF+34, DST                          ; 2848
                          4B  11 001C8           BRB     74$                                            ; 2849
            50      0B    A7  9E 001CA  56$:     MOVAB   OUTPUT_ATTBUF+35, DST                          ; 2855
                          45  11 001CE  57$:     BRB     74$                                            ; 2856
            50      08    A7  9E 001D0  58$:     MOVAB   OUTPUT_ATTBUF+32, DST                          ; 2862
                          72  11 001D4  59$:     BRB     85$                                            ; 2863
            50            67  9E 001D6  60$:     MOVAB   OUTPUT_ATTBUF+24, DST                          ; 2869
                          64  11 001D9           BRB     83$                                            ; 2870
            50      0C    A7  9E 001DB  61$:     MOVAB   OUTPUT_ATTBUF+36, DST                          ; 2876
                          5E  11 001DF           BRB     83$                                            ; 2877
            50      04    A7  9E 001E1  62$:     MOVAB   OUTPUT_ATTBUF+28, DST                          ; 2883
                          58  11 001E5  63$:     BRB     83$                                            ; 2884
            50      E8    A7  9E 001E7  64$:     MOVAB   OUTPUT_ATTBUF, DST                             ; 2890
                          69  11 001EB           BRB     87$                                            ;
            50      F8    A7  9E 001ED  65$:     MOVAB   OUTPUT_ATTBUF+16, DST                          ; 2902
                          63  11 001F1           BRB     87$                                            ;
            50      3C    A7  9E 001F3  66$:     MOVAB   OUTPUT_ATTBUF+84, DST                          ; 2908
                          46  11 001F7  67$:     BRB     83$                                            ; 2909
            50      40    A7  9E 001F9  68$:     MOVAB   OUTPUT_ATTBUF+88, DST                          ; 2915
                          57  11 001FD           BRB     87$                                            ;
            50      F0    A7  9E 001FF  69$:     MOVAB   OUTPUT_ATTBUF+8, DST                           ; 2928
                          51  11 00203           BRB     87$                                            ;
            50      5C    A7  9E 00205  70$:     MOVAB   OUTPUT_ATTBUF+116, DST                         ; 2934
                          34  11 00209  71$:     BRB     83$                                            ; 2935
            50      60    A7  9E 0020B  72$:     MOVAB   OUTPUT_ATTBUF+120, DST                         ; 2941
                          37  11 0020F           BRB     85$                                            ; 2942
            50      63    A7  9E 00211  73$:     MOVAB   OUTPUT_ATTBUF+123, DST                         ; 2948
            51            01  D0 00215  74$:     MOVL    #1, DLEN                                       ; 2949
                          46  11 00218  75$:     BRB     88$                                            ;
            50      64    A7  9E 0021A  76$:     MOVAB   OUTPUT_ATTBUF+124, DST                         ; 2955
                          28  11 0021E           BRB     85$                                            ; 2956
            50      66    A7  9E 00220  77$:     MOVAB   OUTPUT_ATTBUF+126, DST                         ; 2962
                          22  11 00224           BRB     85$                                            ; 2963
            50      48    A7  9E 00226  78$:     MOVAB   OUTPUT_ATTBUF+96, DST                          ; 2969
                          04  11 0022A           BRB     80$                                            ; 2970
            50      50    A7  9E 0022C  79$:     MOVAB   OUTPUT_ATTBUF+104, DST                         ; 2976
            51            08  D0 00230  80$:     MOVL    #8, DLEN                                       ; 2977
                          2B  11 00233           BRB     88$                                            ;
            50      68    A7  9E 00235  81$:     MOVAB   OUTPUT_ATTBUF+128, DST                         ; 2983
                          1B  11 00239           BRB     87$                                            ;
            50      70    A7  9E 0023B  82$:     MOVAB   OUTPUT_ATTBUF+136, DST                         ; 2989
            51            04  D0 0023F  83$:     MOVL    #4, DLEN                                       ; 2990
                          1C  11 00242           BRB     88$                                            ;
            50      74    A7  9E 00244  84$:     MOVAB   OUTPUT_ATTBUF+140, DST                         ; 2996
            51            02  D0 00248  85$:     MOVL    #2, DLEN                                       ; 2997
                          13  11 0024B           BRB     88$                                            ;
            50      5C    A7  9E 0024D  86$:     MOVAB   OUTPUT_ATTBUF+116, DST                         ; 3003
            51            18  D0 00251           MOVL    #24, DLEN                                      ; 3004
                          0A  11 00254           BRB     88$                                            ;
            60      04    BC  B0 00256  87$:     MOVW    @ATT, (DST)                                    ; 3018
      04    A0      04    A2  9E 0025A           MOVAB   4(R2), 4(DST)                                  ; 3019
                              04 0025F           RET                                                    ; 2455
```

```
        51         00      04  A2        04  BC  2C 00260 88$:    MOVC5   @ATT, 4(R2), #0, DLEN, (DST)               ; 3028
                                              60    00267                                                              ;
                               0F        02  A6  B1 00268         CMPW    2(R6), #15                                 ; 3029
                                              06  12 0026C         BNEQ    89$                                        ;
                         FE2A  C7        58  A7  90 0026E         MOVB    OUTPUT_ATTBUF+112, RWSV_LOOKAHEAD          ; 3030
                               0E        02  A6  B1 00274 89$:    CMPW    2(R6), #14                                 ; 3031
                                              06  12 00278         BNEQ    90$                                        ;
                         FE2B  C7        56  A7  90 0027A         MOVB    OUTPUT_ATTBUF+110, RWSV_XORSIZE            ; 3032
                                                  04 00280 90$:    RET                                                ; 3034

; Routine Size:  641 bytes,     Routine Base:  CODE + 0312
```

```
; 1494      3035  1 %SBTTL 'GET_SUMMARY_ATTRIBUTES - process BACKUP summary record'
; 1495      3036  1 GLOBAL ROUTINE GET_SUMMARY_ATTRIBUTES(REC): NOVALUE=
; 1496      3037  1
; 1497      3038  1 !++
; 1498      3039  1 !
; 1499      3040  1 ! FUNCTIONAL DESCRIPTION:
; 1500      3041  1 !       This routine processes a BACKUP summary attribute record, expanding
; 1501      3042  1 !       the attributes it contains into the OUTPUT_ATTBUF area.
; 1502      3043  1 !
; 1503      3044  1 ! INPUT PARAMETERS:
; 1504      3045  1 !       REC             - Pointer to attribute record.
; 1505      3046  1 !
; 1506      3047  1 ! IMPLICIT INPUTS:
; 1507      3048  1 !       NONE
; 1508      3049  1 !
; 1509      3050  1 ! OUTPUT PARAMETERS:
; 1510      3051  1 !       NONE
; 1511      3052  1 !
; 1512      3053  1 ! IMPLICIT OUTPUTS:
; 1513      3054  1 !       OUTPUT_ATTBUF   - Contains the attribute values.
; 1514      3055  1 !
; 1515      3056  1 ! ROUTINE VALUE:
; 1516      3057  1 !       NONE
; 1517      3058  1 !
; 1518      3059  1 ! SIDE EFFECTS:
; 1519      3060  1 !       NONE
; 1520      3061  1 !
; 1521      3062  1 !--
; 1522      3063  1
; 1523      3064  2 BEGIN
; 1524      3065  2 GLOBAL REGISTER P$ = 11;
; 1525      3066  2 CH$FILL(0, BSR_LENGTH, OUTPUT_ATTBUF);
; 1526      3067  2 DEBLOCK_ATTR(.REC, MASK_BSR, GET_ONE_ATTRIBUTE);
; 1527      3068  1 END;



                                              083C 00000          .ENTRY   GET_SUMMARY_ATTRIBUTES, Save R2,R3,R4,R5,-   ; 3036
                                                                           R11                                          ;
  008C   8F          00          6E       00   2C 00002          MOVC5    #0, (SP), #0, #140, OUTPUT_ATTBUF            ; 3066
                              00000000'   EF      00009                                                                 ;
                                   FD6D   CF   9F 0000E          PUSHAB   GET_ONE_ATTRIBUTE                            ; 3067
                                   FA57   CF   9F 00012          PUSHAB   MASK_BSR                                     ;
                                     04   AC   DD 00016          PUSHL    REC                                          ;
                     FBB3   CF            03   FB 00019          CALLS    #3, DEBLOCK_ATTR                             ;
                                          04      0001E          RET                                                   ; 3068

; Routine Size:  31 bytes,    Routine Base:  CODE + 0593
```

```
1 %SBTTL 'GET_VOLUME_ATTRIBUTES - process volume summary record'
1 GLOBAL ROUTINE GET_VOLUME_ATTRIBUTES(REC): NOVALUE=
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !       This routine processes a volume summary record, expanding
1 !       the attributes it contains into the OUTPUT_ATTBUF area.
1 !
1 ! INPUT PARAMETERS:
1 !       REC             - Pointer to attribute record.
1 !
1 ! IMPLICIT INPUTS:
1 !       NONE
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       OUTPUT_ATTBUF   - Contains the attribute values.
1 !
1 ! ROUTINE VALUE:
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !       NONE
1 !
1 !--
1
2 BEGIN
2 GLOBAL REGISTER P$ = 11;
2 CH$FILL(0, VSR_LENGTH, OUTPUT_ATTBUF);
2 DEBLOCK_ATTR(.REC, MASK_VSR, GET_ONE_ATTRIBUTE);
1 END;
```

```
                                           083C 00000         .ENTRY  GET_VOLUME_ATTRIBUTES, Save R2,R3,R4,R5,R11  ; 3070
   0070   8F          00            6E    00  2C 00002         MOVC5   #0, (SP), #0, #112, OUTPUT_ATTBUF            ; 3100
                                 00000000'  EF    00009                                                                ;
                                      FD4E  CF  9F 0000E         PUSHAB  GET_ONE_ATTRIBUTE                             ; 3101
                                      FA44  CF  9F 00012         PUSHAB  MASK_VSR                                      ;
                                        04  AC  DD 00016         PUSHL   REC                                           ;
                       FB94   CF              03  FB 00019         CALLS   #3, DEBLOCK_ATTR                              ;
                                              04 0001E         RET                                                   ; 3102
```

; Routine Size: 31 bytes,   Routine Base: CODE + 05B2

```
1 %SBTTL 'GET_FILE_ATTRIBUTES - process file attribute record'
1 GLOBAL ROUTINE GET_FILE_ATTRIBUTES(REC): NOVALUE=
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !       This routine processes a file attribute record, expanding
1 !       the attributes it contains into the OUTPUT_ATTBUF area.
1 !
1 ! INPUT PARAMETERS:
1 !       REC             - Pointer to attribute record.
1 !
1 ! IMPLICIT INPUTS:
1 !       NONE
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       OUTPUT_ATTBUF   - Contains the attribute values.
1 !
1 ! ROUTINE VALUE:
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !       NONE
1 !
1 !--
1
2 BEGIN
2 GLOBAL REGISTER P$ = 11;
2 CH$FILL(0, FAR_LENGTH, OUTPUT_ATTBUF);
2 DEBLOCK_ATTR(.REC, MASK_FAR, GET_ONE_ATTRIBUTE);
1 END;


                                           083C 00000        .ENTRY  GET_FILE_ATTRIBUTES, Save R2,R3,R4,R5,R11    ; 3104
  0090  8F         00         6E      00     2C 00002        MOVC5   #0, (SP), #0, #144, OUTPUT_ATTBUF            ; 3134
                              00000000'   EF    00009                                                              ;
                                   FD2F   CF 9F 0000E        PUSHAB  GET_ONE_ATTRIBUTE                            ; 3135
                                   FA31   CF 9F 00012        PUSHAB  MASK_FAR                                     ;
                                     04   AC DD 00016        PUSHL   REC                                          ;
                      FB75   CF           03 FB 00019        CALLS   #3, DEBLOCK_ATTR                             ;
                                             04 0001E        RET                                                  ; 3136

; Routine Size:  31 bytes,    Routine Base:  CODE + 05D1
```

```
; 1599          3137  1 %SBTTL 'GET_PHYSVOL_ATTRIBUTES - process physical volume record'
; 1600          3138  1 %SBTTL 'GET_FILE_EXT_ATTRIBUTES - process file attribute extension record'
; 1601          3139  1 GLOBAL ROUTINE GET_FILE_EXT_ATTRIBUTES(REC): NOVALUE=
; 1602          3140  1
; 1603          3141  1 !++
; 1604          3142  1 !
; 1605          3143  1 ! FUNCTIONAL DESCRIPTION:
; 1606          3144  1 !       This routine processes a file attribute extension record.
; 1607          3145  1 !       The record processing is the same as GET_FILE_ATTRIBUTES,
; 1608          3146  1 !       except that the OUTPUT_ATTBUF is not cleared before the
; 1609          3147  1 !       record is deblocked.
; 1610          3148  1 !
; 1611          3149  1 ! INPUT PARAMETERS:
; 1612          3150  1 !       REC             - Pointer to attribute record.
; 1613          3151  1 !
; 1614          3152  1 ! IMPLICIT INPUTS:
; 1615          3153  1 !       NONE
; 1616          3154  1 !
; 1617          3155  1 ! OUTPUT PARAMETERS:
; 1618          3156  1 !       NONE
; 1619          3157  1 !
; 1620          3158  1 ! IMPLICIT OUTPUTS:
; 1621          3159  1 !       NONE
; 1622          3160  1 !
; 1623          3161  1 ! ROUTINE VALUE:
; 1624          3162  1 !       NONE
; 1625          3163  1 !
; 1626          3164  1 ! SIDE EFFECTS:
; 1627          3165  1 !       NONE
; 1628          3166  1 !
; 1629          3167  1 !--
; 1630          3168  1
; 1631          3169  2 BEGIN
; 1632          3170  2 GLOBAL REGISTER P$ = 11;
; 1633          3171  2 DEBLOCK_ATTR(.REC, MASK_FAR, GET_ONE_ATTRIBUTE);
; 1634          3172  1 END;


                                    0800 00000          .ENTRY  GET_FILE_EXT_ATTRIBUTES, Save R11              ; 3139
                         FD1C   CF   9F 00002          PUSHAB  GET_ONE_ATTRIBUTE                              ; 3171
                         FA1E   CF   9F 00006          PUSHAB  MASK_FAR                                       ;
                           04   AC   DD 0000A          PUSHL   REC                                            ;
           FB62   CF             03   FB 0000D          CALLS   #3, DEBLOCK_ATTR                               ;
                                  04 00012          RET                                                    ; 3172

; Routine Size:  19 bytes,    Routine Base:  CODE + 05F0
```

```
; 1636          3173  1 GLOBAL ROUTINE GET_PHYSVOL_ATTRIBUTES(REC): NOVALUE=
; 1637          3174  1
; 1638          3175  1 !++
; 1639          3176  1 !
; 1640          3177  1 ! FUNCTIONAL DESCRIPTION:
; 1641          3178  1 !       This routine processes a physical volume attribute record, expanding
; 1642          3179  1 !       the attributes it contains into the OUTPUT_ATTBUF area.
; 1643          3180  1 !
; 1644          3181  1 ! INPUT PARAMETERS:
; 1645          3182  1 !       REC             - Pointer to attribute record.
; 1646          3183  1 !
; 1647          3184  1 ! IMPLICIT INPUTS:
; 1648          3185  1 !       NONE
; 1649          3186  1 !
; 1650          3187  1 ! OUTPUT PARAMETERS:
; 1651          3188  1 !       NONE
; 1652          3189  1 !
; 1653          3190  1 ! IMPLICIT OUTPUTS:
; 1654          3191  1 !       OUTPUT_ATTBUF   - Contains the attribute values.
; 1655          3192  1 !
; 1656          3193  1 ! ROUTINE VALUE:
; 1657          3194  1 !       NONE
; 1658          3195  1 !
; 1659          3196  1 ! SIDE EFFECTS:
; 1660          3197  1 !       NONE
; 1661          3198  1 !
; 1662          3199  1 !--
; 1663          3200  1
; 1664          3201  2 BEGIN
; 1665          3202  2 GLOBAL REGISTER P$ = 11;
; 1666          3203  2 CH$FILL(0, PVA_LENGTH, OUTPUT_ATTBUF);
; 1667          3204  2 DEBLOCK_ATTR(.REC, MASK_PVA, GET_ONE_ATTRIBUTE);
; 1668          3205  1 END;


                                        083C 00000          .ENTRY  GET_PHYSVOL_ATTRIBUTES, Save R2,R3,R4,R5,-   ; 3173
                                                                    R11
        28          00          6E    00  2C 00002          MOVC5   #0, (SP), #0, #40, OUTPUT_ATTBUF             ; 3203
                          00000000'   EF     00007                                                              ;
                               FCFF   CF  9F 0000C          PUSHAB  GET_ONE_ATTRIBUTE                            ; 3204
                               FA0D   CF  9F 00010          PUSHAB  MASK_PVA                                     ;
                                 04   AC  DD 00014          PUSHL   REC                                          ;
                    FB45    CF        03  FB 00017          CALLS   #3, DEBLOCK_ATTR                             ;
                                          04 0001C          RET                                                  ; 3205

; Routine Size:  29 bytes,    Routine Base:  CODE + 0603


; 1669          3206  1
```

```
%SBTTL 'CRYPTO_CHKSAV - Check to see if saveset is incrypted'
GLOBAL ROUTINE CRYPTO_CHKSAV(BCB) =

!++
!
! FUNCTIONAL DESCRIPTION:
!       This routine locates the backup summary record in a block and
!       and determines if the saveset is encrypted.
!
! INPUT PARAMETERS:
!       BCB = address of block containing the backup summary record.
!
! IMPLICIT INPUTS:
!       NONE
!
! OUTPUT PARAMETERS:
!       NONE
!
! IMPLICIT OUTPUTS:
!
!
! ROUTINE VALUE:
!       TRUE if the saveset is encrypted
!       FALSE if not encrypted.
!
! SIDE EFFECTS:
!
!
!--

BEGIN
GLOBAL REGISTER
    P$ = 11:    REF VECTOR;             ! linkage for data passed to inner routine
MAP
    BCB:        REF BBLOCK;             ! parameter points to buffer

!
! Set flag to indicate whether saveset is encrypted
!
SAVESET_ENCR = 0 ;

!
! Use the BACKUP facility routine DEBLOCK to extract all records
! of the buffer.
!
DEBLOCK (.BCB, SEARCHFOR_BSR_ENC , TRUE ) ;


IF .SAVESET_ENCR EQL 0
    THEN RETURN FALSE
    ELSE RETURN TRUE ;
END;
```

```
                                  0804 00000         .ENTRY  CRYPTO_CHKSAV, Save R2,R11              ; 3208
              52 00000000'   EF   9E 00002           MOVAB   SAVESET_ENCR, R2                        ;
                             62   D4 00009           CLRL    SAVESET_ENCR                            ; 3246
                             01   DD 0000B           PUSHL   #1                                      ; 3252
                      0000V  CF   9F 0000D           PUSHAB  SEARCHFOR_BSR_ENC                       ;
                         04  AC   DD 00011           PUSHL   BCB                                     ;
         FA74    CF          03   FB 00014           CALLS   #3, DEBLOCK                             ;
                             62   D5 00019           TSTL    SAVESET_ENCR                            ; 3255
                             03   12 0001B           BNEQ    1$                                      ;
                             50   D4 0001D           CLRL    R0                                      ; 3257
                                  04 0001F           RET                                             ;
              50             01   D0 00020 1$:       MOVL    #1, R0                                  ;
                                  04 00023           RET                                             ; 3258

; Routine Size:  36 bytes,    Routine Base:  CODE + 0620


; 1723          3259  1
```

```
1 %SBTTL 'SEARCHFOR_BSR_ENC - Search for Backup Summary in block'
1 ROUTINE SEARCHFOR_BSR_ENC(PTR): L_P$ NOVALUE =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !       This routine simply checks for an encrypted save set
1 !
1 ! INPUT PARAMETERS:
1 !       PTR = address of record to examine
1 !
1 ! IMPLICIT INPUTS:
1 !       NONE
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !       The listing is produced.
1 !
1 !--
1
2 BEGIN
2
2 EXTERNAL ROUTINE
2     GET_SUMMARY_ATTRIBUTES ;
2
2 GLOBAL REGISTER
2     P$ = 11:    REF VECTOR;              ! linkage for data passed to inner routine
2
2 MAP
2     PTR:        REF BBLOCK;              ! address of record from block to examine
2
2 !
2 ! We are only interested in backup summary records
2 !
2 IF .PTR[BRH$W_RTYPE] NEQU BRH$K_SUMMARY THEN RETURN;
2
2 GET_SUMMARY_ATTRIBUTES(.PTR);
2
2 SAVESET_ENCR = .BBLOCK[OUTPUT_ATTBUF[BSR_CRYPTDATA], BSA$B_CRYPTYP] ;
2
1 END;
```

```
                              0000 00000 SEARCHFOR_BSR_ENC:
                                                .WORD   Save nothing                  ; 3261
         50       04   AC   D0 00002            MOVL    PTR, R0                       ; 3302
```

```
                             01        02   A0  B1 00006          CMPW    2(R0), #1                           ;
                                            14  12 0000A          BNEQ    1$                                  ;
                                            50  DD 0000C          PUSHL   R0                                  ; 3304
                   00000000G  00            01  FB 0000E          CALLS   #1, GET_SUMMARY_ATTRIBUTES          ;
                   00000000'  EF 00000000'  EF  9A 00015          MOVZBL  OUTPUT_ATTBUF+116, SAVESET_ENCR     ; 3306
                                                04 00020 1$:      RET                                         ; 3308

; Routine Size:  33 bytes,    Routine Base:  CODE + 0644


; 1774          3309  1
```

```
%SBTTL 'DECODE_DEVTYP - decode the devtyp phys vol attr'
GLOBAL ROUTINE DECODE_DEVTYP ( MEDIA_ID , NAME_LENGTH,
                               NAME_BUFFER: REF VECTOR[,BYTE],
                               TYPE_LENGTH,
                               TYPE_BUFFER: REF VECTOR[,BYTE] ): NOVALUE=

!++
!
! FUNCTIONAL DESCRIPTION:
!       This routine takes the longword walue and interprets it as a
!       UCB MEDIA_ID and converts it to ascii
!
! INPUT PARAMETERS:
!       MEDIA_ID        - Longword value
!       NAME_LENGTH     - Address of longword containing the name buffer length
!       TYPE_LENGTH     - Address of longword containing the type buffer length
!
! IMPLICIT INPUTS:
!       NONE
!
! OUTPUT PARAMETERS:
!       NAME_LENGTH     - address of longword to recieve name string length
!       NAME_BUFFER     - address of buffer to receive name string
!       TYPE_LENGTH     - address of longword to recieve type string length
!       TYPE_BUFFER     - address of buffer to receive type string
!
! IMPLICIT OUTPUTS:
!
! ROUTINE VALUE:
!       NONE
!
! SIDE EFFECTS:
!       NONE
!
!--

BEGIN

FIELD
    ! Field definitions for UCB$L_MEDIA_ID
    !
    MEDIA_ID_FIELDS =
        SET
        MEDIA_ID_T0 = [UCB$V_MEDIA_ID_T0],
        MEDIA_ID_T1 = [UCB$V_MEDIA_ID_T1],
        MEDIA_ID_N0 = [UCB$V_MEDIA_ID_N0],
        MEDIA_ID_N1 = [UCB$V_MEDIA_ID_N1],
        MEDIA_ID_N2 = [UCB$V_MEDIA_ID_N2],
        MEDIA_ID_NN = [UCB$V_MEDIA_ID_NN]
        TES ;

! We only reserve a byte for this number.
!
$ASSUME (%FIELDEXPAND(MEDIA_ID_NN,2), LEQ, 8) ;

STRUCTURE
        ! Define a structure to ignore the offset.
```

```
2           ! This way we can use the UCB symbols that are already
2           ! defined.
2           !
2           MEDIA_ID_STRUCT[O,P,S,E;N]=
2               [N]
2               (MEDIA_ID_STRUCT)<P,S,E>;
2
2 MAP
2     MEDIA_ID : MEDIA_ID_STRUCT FIELD (MEDIA_ID_FIELDS) ;
2
2 BIND
2     NAME_INLEN = .NAME_LENGTH : LONG ,
2     TYPE_INLEN = .TYPE_LENGTH : LONG ;
2
2 LOCAL
2     OUTLEN     : LONG             INITIAL (0) ,
2     CTRL_DESC : BBLOCK[8]         PRESET (
2                                       [DSC$W_LENGTH] = %CHARCOUNT('!2ZB'),
2                                       [DSC$B_DTYPE]  = DSC$K_DTYPE_Z,
2                                       [DSC$B_CLASS]  = DSC$K_CLASS_S,
2                                       [DSC$A_POINTER]= UPLIT BYTE('!2ZB')),
2     OUT_DESC  : BBLOCK[8]         PRESET (
2                                       [DSC$W_LENGTH] = 0,
2                                       [DSC$B_DTYPE]  = DSC$K_DTYPE_Z,
2                                       [DSC$B_CLASS]  = DSC$K_CLASS_S,
2                                       [DSC$A_POINTER]= 0),
2     NUMBER     : BYTE             INITIAL (0) ;
2
2 ! We must have at least 3 bytes in the name buffer
2 !
2 IF NAME_INLEN NEQ 0
2 THEN
3     BEGIN
3     IF (.NAME_INLEN GEQ 3) AND
4        (.NAME_BUFFER NEQ 0)
3     THEN
4         BEGIN
4
4         OUTLEN = 0 ;
4
4         ! Each character is assigned a number from 1 to 26 corresponding to A-z
4         !
4         IF .MEDIA_ID[MEDIA_ID_NO] NEQ 0
4         THEN
5             BEGIN
5             NAME_BUFFER[.OUTLEN] = %X'40' + .MEDIA_ID[MEDIA_ID_NO] ;
5             OUTLEN = .OUTLEN + 1 ;
4             END ;
4         IF .MEDIA_ID[MEDIA_ID_N1] NEQ 0
4         THEN
5             BEGIN
5             NAME_BUFFER[.OUTLEN] = %X'40' + .MEDIA_ID[MEDIA_ID_N1] ;
5             OUTLEN = .OUTLEN + 1 ;
4             END ;
4         IF .MEDIA_ID[MEDIA_ID_N2] NEQ 0
4         THEN
5             BEGIN
```

```
                    NAME_BUFFER[.OUTLEN] = %X'40' + .MEDIA_ID[MEDIA_ID_N2] ;
                    OUTLEN = .OUTLEN + 1 ;
                    END ;

                ! The last field is a number
                !
                NUMBER = .MEDIA_ID[MEDIA_ID_NN] ;
                OUT_DESC[DSC$W_LENGTH] = .NAME_INLEN - .OUTLEN ;
                OUT_DESC[DSC$A_POINTER] = NAME_BUFFER[.OUTLEN] ;
                $FAO (CTRL_DESC,                        ! CTRSTR
                      OUT_DESC[DSC$W_LENGTH],           ! OUTLEN
                      OUT_DESC,                         ! OUTBUF
                      .NUMBER ) ;                       ! P1
                OUTLEN = .OUTLEN + .OUT_DESC[DSC$W_LENGTH] ;

                NAME_INLEN = .OUTLEN ;
                END
            ELSE
                NAME_INLEN = 0 ;
            END ;

        ! We must have at least 2 bytes in the type buffer
        !
        IF TYPE_INLEN NEQ 0
        THEN
            BEGIN
            IF (.TYPE_INLEN GEQ 2) AND
               (.TYPE_BUFFER NEQ 0)
            THEN
                BEGIN

                OUTLEN = 0 ;

                ! Each character is assigned a number from 1 to 26 corresponding to A-z
                !
                IF .MEDIA_ID[MEDIA_ID_T0] NEQ 0
                THEN
                    BEGIN
                    TYPE_BUFFER[.OUTLEN] = %X'40' + .MEDIA_ID[MEDIA_ID_T0] ;
                    OUTLEN = .OUTLEN + 1 ;
                    END ;
                IF .MEDIA_ID[MEDIA_ID_T1] NEQ 0
                THEN
                    BEGIN
                    TYPE_BUFFER[.OUTLEN] = %X'40' + .MEDIA_ID[MEDIA_ID_T1] ;
                    OUTLEN = .OUTLEN + 1 ;
                    END ;

                TYPE_INLEN = .OUTLEN ;
                END
            ELSE
                TYPE_INLEN = 0 ;
            END ;

        RETURN ;

        END ;
```

```
                              42  5A  32  21  00665 P.AAA:  .ASCII  \!2ZB\                                        ;

                                                            .EXTRN  SYS$FAO

                                        0004 00000          .ENTRY  DECODE_DEVTYP, Save R2                        ; 3311
                     5E           0C  C2 00002              SUBL2   #12, SP                                       ;
                                  52  D4 00005              CLRL    OUTLEN                                        ; 3379
              04     AE 01000004  8F  D0 00007              MOVL    #16777220, CTRL_DESC                          ; 3387
              08     AE       EA  AF  9E 0000F              MOVAB   P.AAA, CTRL_DESC+4                            ;
                        01000000  8F  DD 00014              PUSHL   #16777216                                     ; 3392
                              04  AE  D4 0001A              CLRL    OUT_DESC+4                                    ;
                                  50  94 0001D              CLRB    NUMBER                                        ;
                              08  AC  D5 0001F              TSTL    NAME_LENGTH                                   ; 3397
                                  03  12 00022              BNEQ    1$                                            ;
                                0086  31 00024              BRW     6$                                            ;
                     03       08  BC  D1 00027 1$:          CMPL    @NAME_LENGTH, #3                              ; 3400
                                  7D  19 0002B              BLSS    5$                                            ;
                              0C  AC  D5 0002D              TSTL    NAME_BUFFER                                   ; 3401
                                  78  13 00030              BEQL    5$                                            ;
                                  52  D4 00032              CLRL    OUTLEN                                        ; 3405
                     3E       06  AC  93 00034              BITB    MEDIA_ID+2, #62                               ; 3409
                                  0F  13 00038              BEQL    2$                                            ;
 51         06   AC  05           01  EF 0003A              EXTZV   #1, #5, MEDIA_ID+2, R1                        ; 3412
            0C BC42  51       40  8F  81 00040              ADDB3   #64, R1, @NAME_BUFFER[OUTLEN]                 ;
                                  52  D6 00047              INCL    OUTLEN                                        ; 3413
              01F0   8F       05  AC  B3 00049 2$:          BITW    MEDIA_ID+1, #496                              ; 3415
                                  0F  13 0004F              BEQL    3$                                            ;
 51         05   AC  05           04  EF 00051              EXTZV   #4, #5, MEDIA_ID+1, R1                        ; 3418
            0C BC42  51       40  8F  81 00057              ADDB3   #64, R1, @NAME_BUFFER[OUTLEN]                 ;
                                  52  D6 0005E              INCL    OUTLEN                                        ; 3419
              0F80   8F       04  AC  B3 00060 3$:          BITW    MEDIA_ID, #3968                               ; 3421
                                  0F  13 00066              BEQL    4$                                            ;
 51         04   AC  05           07  EF 00068              EXTZV   #7, #5, MEDIA_ID, R1                          ; 3424
            0C BC42  51       40  8F  81 0006E              ADDB3   #64, R1, @NAME_BUFFER[OUTLEN]                 ;
                                  52  D6 00075              INCL    OUTLEN                                        ; 3425
 51         04   AC  07           00  EF 00077 4$:          EXTZV   #0, #7, MEDIA_ID, R1                          ; 3430
                     50           51  90 0007D              MOVB    R1, NUMBER                                    ;
                 6E  08  BC       52  A3 00080              SUBW3   OUTLEN, @NAME_LENGTH, OUT_DESC                ; 3431
                     04  AE  0C BC42  9E 00085              MOVAB   @NAME_BUFFER[OUTLEN], OUT_DESC+4              ; 3432
                         7E       50  9A 0008B              MOVZBL  NUMBER, -(SP)                                 ; 3436
                              04  AE  9F 0008E              PUSHAB  OUT_DESC                                      ;
                              08  AE  9F 00091              PUSHAB  OUT_DESC                                      ;
                              14  AE  9F 00094              PUSHAB  CTRL_DESC                                     ;
           00000000G 00           04  FB 00097              CALLS   #4, SYS$FAO                                   ;
                     50           6E  3C 0009E              MOVZWL  OUT_DESC, R0                                  ; 3437
                     52           50  C0 000A1              ADDL2   R0, OUTLEN                                    ;
              08     BC           52  D0 000A4              MOVL    OUTLEN, @NAME_LENGTH                          ; 3439
                                  03  11 000A8              BRB     6$                                            ; 3400
                              08  BC  D4 000AA 5$:          CLRL    @NAME_LENGTH                                  ; 3442
                              10  AC  D5 000AD 6$:          TSTL    TYPE_LENGTH                                   ; 3447
                                  42  13 000B0              BEQL    10$                                           ;
                     02       10  BC  D1 000B2              CMPL    @TYPE_LENGTH, #2                              ; 3450
                                  39  19 000B6              BLSS    9$                                            ;
                              14  AC  D5 000B8              TSTL    TYPE_BUFFER                                   ; 3451
```

```
                                34  13 000BB          BEQL    9$                               ;
                                52  D4 000BD          CLRL    OUTLEN                           ; 3455
                F8   8F     07  AC  93 000BF          BITB    MEDIA_ID+3, #248                 ; 3459
                                0F  13 000C4          BEQL    7$                               ;
   50     07  AC         05     03  EF 000C6          EXTZV   #3, #5, MEDIA_ID+3, R0           ; 3462
          14 BC42        50  40 8F  81 000CC          ADDB3   #64, R0, @TYPE_BUFFER[OUTLEN]    ;
                                52  D6 000D3          INCL    OUTLEN                           ; 3463
              07C0   8F     06  AC  B3 000D5 7$:      BITW    MEDIA_ID+2, #1984                ; 3465
                                0F  13 000DB          BEQL    8$                               ;
   50     06  AC         05     06  EF 000DD          EXTZV   #6, #5, MEDIA_ID+2, R0           ; 3468
          14 BC42        50  40 8F  81 000E3          ADDB3   #64, R0, @TYPE_BUFFER[OUTLEN]    ;
                                52  D6 000EA          INCL    OUTLEN                           ; 3469
                10   BC         52  D0 000EC 8$:      MOVL    OUTLEN, @TYPE_LENGTH             ; 3472
                                    04 000F0          RET                                      ; 3450
                            10  BC  D4 000F1 9$:      CLRL    @TYPE_LENGTH                     ; 3475
                                    04 000F4 10$:     RET                                      ; 3480

; Routine Size:  245 bytes,    Routine Base:  CODE + 0669

; 1947         3481  1
```

```
; 1949      3482  1 END
; 1950      3483  0 ELUDOM



                                                        .EXTRN  LIB$SIGNAL

;                               PSECT SUMMARY
;
;       Name                 Bytes                    Attributes
;
;   COMMON                     2124  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  OVR,NOPIC,ALIGN(2)
;   CODE                       1886  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;   DATA                          4  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)


;                           Library Statistics
;
;                                          -------- Symbols --------      Pages      Processing
;       File                               Total    Loaded  Percent      Mapped     Time
;
;   _$255$DUA28:[SYSLIB]LIB.L32;1          18619       28        0        1000        00:02.2



;                               COMMAND QUALIFIERS

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

; Size:           1835 code + 2179 data bytes
; Run Time:          00:57.1
; Elapsed Time:      03:11.2
; Lines/CPU Min:      3662
; Lexemes/CPU-Min:  28500
; Memory Used:  525 pages
; Compilation Complete
```

TAPEUTIL
LIS

STAINIVOL
LIS

UTILITY
LIS

STAIOSYS
LIS

STAPARSE
LIS

0017 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

BADBLOCKS
LIS

BAD

STABAD
MAP

BADDATA
LIS

ANALYZBAD
MAP

ANALYZCMD
CLD

BADDEF
SDL

WRITESAVE
LIS

BADINIT
LIS

ANALYZCMD
LIS