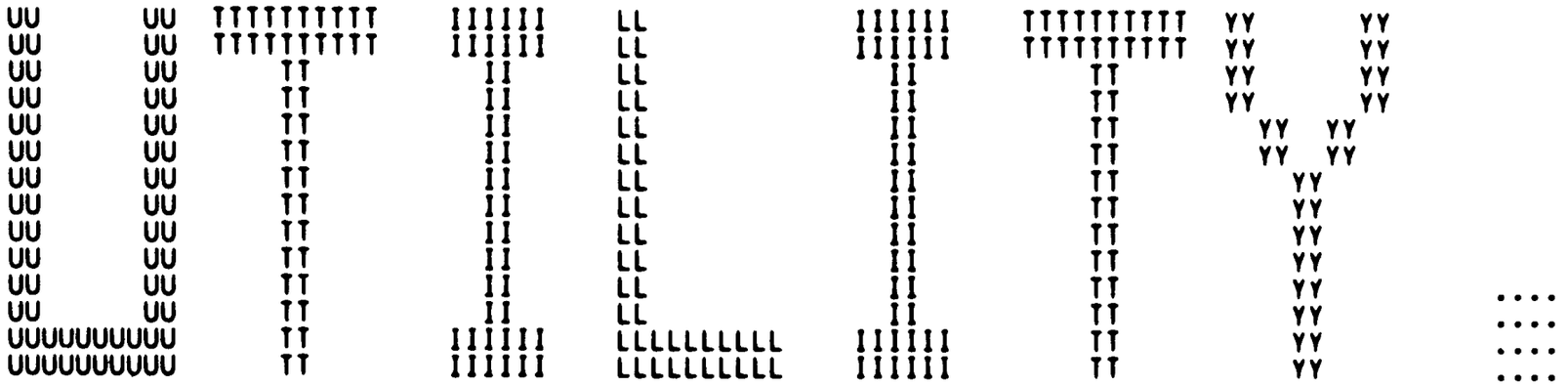
```
BBBBBBBBBBBB        AAAAAAAAA        CCCCCCCCCCC  KKK         KKK  UUU          UUU  PPPPPPPPPPPP
BBBBBBBBBBBB        AAAAAAAAA        CCCCCCCCCCC  KKK         KKK  UUU          UUU  PPPPPPPPPPPP
BBBBBBBBBBBB        AAAAAAAAA        CCCCCCCCCCC  KKK         KKK  UUU          UUU  PPPPPPPPPPPP
BBB         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  UUUUUUUUUUUUUUUU  PPP
BBBBBBBBBBBB     AAA         AAA     CCCCCCCCCCC  KKK          KKK  UUUUUUUUUUUUUUUU  PPP
```


UU UU TTTTTTTTTT IIIIII LL IIIIII TTTTTTTTTT YY YY
UU UU TTTTTTTTTT IIIIII LL IIIIII TTTTTTTTTT YY YY
UU UU TT II LL II TT YY YY
UU UU TT II LL II TT YY YY
UU UU TT II LL II TT YY YY
UU UU TT II LL II TT YY YY
UU UU TT II LL II TT YY
UU UU TT II LL II TT YY
UU UU TT II LL II TT YY
UU UU TT II LL II TT YY
UU UU TT II LL II TT YY
UU UU TT II LL II TT YY
UUUUUUUUUU TT IIIIII LLLLLLLLLL IIIIII TT YY
UUUUUUUUUU TT IIIIII LLLLLLLLLL IIIIII TT YY
LL IIIIII SSSSSSSS
LL IIIIII SSSSSSSS
LL II SS
LL II SS
LL II SS
LL II SS
LL II SSSSSS
LL II SSSSSS
LL II SS
LL II SS
LL II SS
LL II SS
LLLLLLLLLL IIIIII SSSSSSSS
LLLLLLLLLL IIIIII SSSSSSSS

```
;     1        0001  0 MODULE UTILITY  (%TITLE 'Utility routines'
;     2        0002  0                  IDENT = 'V04-000'
;     3        0003  0                  ) =
;     4        0004  1 BEGIN
;     5        0005  1
;     6        0006  1 !
;     7        0007  1 !**************************************************************************
;     8        0008  1 !*                                                                        *
;     9        0009  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                              *
;    10        0010  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                *
;    11        0011  1 !*  ALL RIGHTS RESERVED.                                                  *
;    12        0012  1 !*                                                                        *
;    13        0013  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
;    14        0014  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    15        0015  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER  *
;    16        0016  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
;    17        0017  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY  *
;    18        0018  1 !*  TRANSFERRED.                                                           *
;    19        0019  1 !*                                                                        *
;    20        0020  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE  *
;    21        0021  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
;    22        0022  1 !*  CORPORATION.                                                           *
;    23        0023  1 !*                                                                        *
;    24        0024  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
;    25        0025  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                *
;    26        0026  1 !*                                                                        *
;    27        0027  1 !*                                                                        *
;    28        0028  1 !**************************************************************************
;    29        0029  1 !
;    30        0030  1
;    31        0031  1 !++
;    32        0032  1 ! FACILITY:
;    33        0033  1 !       Backup/Restore
;    34        0034  1 !
;    35        0035  1 ! ABSTRACT:
;    36        0036  1 !       This module contains general utility routines.
;    37        0037  1 !
;    38        0038  1 ! ENVIRONMENT:
;    39        0039  1 !       VAX/VMS user mode.
;    40        0040  1 !--
;    41        0041  1 !
;    42        0042  1 ! AUTHOR: M. Jack, CREATION DATE: 03-Sep-1980
;    43        0043  1 !
;    44        0044  1 ! MODIFIED BY:
;    45        0045  1 !
;    46        0046  1 !       V03-007 LY0510          Larry Yetto             19-JUL-1984 08:48
;    47        0047  1 !               Add DECODE_DEVTYP routine to build an ASCII string from
;    48        0048  1 !               the DEVTYP in the physical volume attribute record.  Increase
;    49        0049  1 !               DEVTYP from 1 to 4 bytes.
;    50        0050  1 !
;    51        0051  1 !       V03-006 LY0484          Larry Yetto             27-APR-1984 08:41
;    52        0052  1 !               FT1 QAR # 2088  -  Add new routines to use to determine if
;    53        0053  1 !               a saveset is encrypted.
;    54        0054  1 !
;    55        0055  1 !       V03-005 ACG0332         Andrew C. Goldstein,    22-Jun-1983  18:29
;    56        0056  1 !               Add file highwater mark and RMS journal flag attributes
;    57        0057  1 !
```

```
;  58          0058  1 ;       V03-004 JEP0003         J. Eric Pollack,        23-Apr-1983  10:53
;  59          0059  1 ;               Add support for encrypted savesets.
;  60          0060  1 ;
;  61          0061  1 ;       V03-003 ACG0313         Andrew C. Goldstein,    12-Feb-1983  17:29
;  62          0062  1 ;               Add routine subtitles
;  63          0063  1 ;
;  64          0064  1 ;       V03-002 LMP0044         L. Mark Pilant,         21-Oct-1982  15:20
;  65          0065  1 ;               Add support for saving and restoring ACL's.
;  66          0066  1 ;
;  67          0067  1 ;       V03-001 ACG0280         Andrew C. Goldstein,    2-Apr-1982  16:45
;  68          0068  1 ;               Include dot on save set name with null type
;  69          0069  1 ;
;  70          0070  1 ;       V02-010 MLJ0081         Martin L. Jack, 26-Feb-1982  15:21
;  71          0071  1 ;               Add RETAINMIN and RETAINMAX attributes to support new home
;  72          0072  1 ;               block fields.
;  73          0073  1 ;
;  74          0074  1 ;       V02-009 MLJ0075         Martin L. Jack, 28-Jan-1982  20:08
;  75          0075  1 ;               Add DIR_VERLIM and VERLIMIT attributes to support version limit
;  76          0076  1 ;               handling.
;  77          0077  1 ;
;  78          0078  1 ;       V02-008 MLJ0062         Martin L. Jack, 3-Dec-1981  12:17
;  79          0079  1 ;               Add DIR_STATUS attribute to support /INCREMENTAL.
;  80          0080  1 ;
;  81          0081  1 ;       V02-007 MLJ0054         Martin L. Jack, 22-Nov-1981  21:32
;  82          0082  1 ;               Integrate GET_VM and FREE_VM jacket routines.
;  83          0083  1 ;
;  84          0084  1 ;       V02-006 MLJ0047         Martin L. Jack, 25-Sep-1981  15:28
;  85          0085  1 ;               Remove INVATTTYP message to allow restores
;  86          0086  1 ;               with earlier versions.
;  87          0087  1 ;
;  88          0088  1 ;       V02-005 MLJ0036         Martin L. Jack, 28-Aug-1981  18:04
;  89          0089  1 ;               Implement parent directory attributes.  Make use of new NAM
;  90          0090  1 ;               block descriptors.
;  91          0091  1 ;
;  92          0092  1 ;       V02-004 MLJ0023         Martin L. Jack, 23-Apr-1981  11:42
;  93          0093  1 ;               Implement placement attribute.
;  94          0094  1 ;
;  95          0095  1 ;       V02-003 MLJ0014         Martin L. Jack, 6-Apr-1981  23:19
;  96          0096  1 ;               Correct and clarify VBN sequence-error messages
;  97          0097  1 ;
;  98          0098  1 ;       V02-002 MLJ0010         Martin L. Jack, 25-Mar-1981  15:50
;  99          0099  1 ;               Reorganize global storage.  Add new attributes for image
; 100          0100  1 ;               restore.  Convert to bitmap for attribute code checking.
; 101          0101  1 ;               Add DEBLOCK parameter to inhibit freeing of buffer.
; 102          0102  1 ;
; 103          0103  1 ;       V02-001 MLJ0006         Martin L. Jack, 26-Feb-1981  17:37
; 104          0104  1 ;               Append filetype in EXTRACT_FILENAME unless it is null
; 105          0105  1 ;
; 106          0106  1 ;**
```

```
;  108          0107  1 REQUIRE 'SRC$:COMMON';
;  109          1213  1 LIBRARY 'SYS$LIBRARY:LIB';
;  110          1214  1 REQUIRE 'LIB$:BACKDEF';
;  111          1664  1
;  112          1665  1
;  113          1666  1 LINKAGE
;  114          1667  1         L_P$=           CALL: GLOBAL(P$=11);
;  115          1668  1
;  116          1669  1
;  117          1670  1 MACRO
;  118          1671  1         L_DECL=         EXTERNAL REGISTER P$ = 11 %;
;  119          1672  1
;  120          1673  1
;  121          1674  1 FORWARD ROUTINE
;  122          1675  1         GET_VM,                             ! Allocate virtual memory
;  123          1676  1         GET_ZERO_VM,                        ! Allocate and zero virtual memory
;  124          1677  1         FREE_VM:        NOVALUE,            ! Free virtual memory
;  125          1678  1         DEBLOCK:        L_P$ NOVALUE,       ! Deblock a save set buffer
;  126          1679  1         DEBLOCK_ATTR:   L_P$ NOVALUE,       ! Deblock an attribute record
;  127          1680  1         FILE_ERROR:     NOVALUE,            ! Signal a file-related error
;  128          1681  1         FIND_BADBLOCK,                      ! Search the bad block table
;  129          1682  1         EXTRACT_FILENAME:
;  130          1683  1                         NOVALUE,            ! Extract file name (and type, if not
;  131          1684  1                                             ! null) from string
;  132          1685  1         EXTRACT_DIR_FILENAME:
;  133          1686  1                         NOVALUE,            ! Extract directory, name, type,
;  134          1687  1                                             ! version from string
;  135          1688  1         GET_ONE_ATTRIBUTE:
;  136          1689  1                         NOVALUE,            ! Store one attribute value
;  137          1690  1         GET_SUMMARY_ATTRIBUTES:
;  138          1691  1                         NOVALUE,            ! Get BACKUP summary attributes
;  139          1692  1         GET_VOLUME_ATTRIBUTES:
;  140          1693  1                         NOVALUE,            ! Get volume summary attributes
;  141          1694  1         GET_FILE_ATTRIBUTES:
;  142          1695  1                         NOVALUE,            ! Get file attributes
;  143          1696  1         GET_FILE_EXT_ATTRIBUTES:
;  144          1697  1                         NOVALUE,            ! Get file attributes from extension record
;  145          1698  1         GET_PHYSVOL_ATTRIBUTES:
;  146          1699  1                         NOVALUE,            ! Get physical volume attributes
;  147          1700  1         CRYPTO_CHKSAV,                      ! Check if saveset is enrypted
;  148          1701  1         SEARCHFOR_BSR_ENC:                  ! Scan records for backup summary
;  149          1702  1                         L_P$ NOVALUE,       !  and determine if saveset is encrypted
;  150          1703  1         DECODE_DEVTYP :
;  151          1704  1                         NOVALUE ;           ! Decode MEDIA_ID
;  152          1705  1
;  153          1706  1
;  154          1707  1 EXTERNAL ROUTINE
;  155          1708  1         FREE_BUFFER:    NOVALUE,            ! Free an I/O buffer
;  156          1709  1         LIB$FREE_VM:    ADDRESSING_MODE(GENERAL),
;  157          1710  1                                             ! Free virtual memory
;  158          1711  1         LIB$GET_VM:     ADDRESSING_MODE(GENERAL),
;  159          1712  1                                             ! Allocate virtual memory
;  160          1713  1         LIB$SIGNAL:     ADDRESSING_MODE(GENERAL);
;  161          1714  1                                             ! Signal a condition
;  162          1715  1
;  163          1716  1
;  164          1717  1 G$DEFINE();              ! Define global common area
```

```
;  165          1718  1
;  166          1719  1
;  167          1720  1 EXTERNAL LITERAL
;  168          1721  1          BACKUP$_VBNERROR,
;  169          1722  1          BACKUP$_VBNMISSING,
;  170          1723  1          BACKUP$_VBNPASTEOF,
;  171          1724  1          BACKUP$_READBLOCK,
;  172          1725  1          BACKUP$_WRITEBLOCK,
;  173          1726  1          BACKUP$_VERIFYERR,
;  174          1727  1          BACKUP$_BADDATA,
;  175          1728  1          BACKUP$_SOFTRERRS,
;  176          1729  1          BACKUP$_SOFTWERRS,
;  177          1730  1          BACKUP$_XORERRS,
;  178          1731  1          BACKUP$_PHYSET,
;  179          1732  1          BACKUP$_NONPHYSET,
;  180          1733  1          BACKUP$_INVATTSTR,
;  181          1734  1          BACKUP$_INVATTSIZ,
;  182          1735  1          BACKUP$_INVRECTYP,
;  183          1736  1          BACKUP$_INVRECSIZ,
;  184          1737  1          BACKUP$_ALLOCMEM,
;  185          1738  1          BACKUP$_FREEMEM;
;  186          1739  1
;  187          1740  1
```

```
;  189          1741  1 %SBTTL 'Attribute validation tables'
;  190          1742  1 ! Define bitvectors of valid attribute codes for each attribute record type.
;  191          1743  1 !
;  192          1744  1 MACRO
;  193        M 1745  1         DEFINE_BITVECTOR_1[A]=
;  194        M 1746  1             [%NAME('BSA$K_', A)] = TRUE %,
;  195          1747  1
;  196          1748  1
;  197        M 1749  1         DEFINE_BITVECTOR(A)=
;  198        M 1750  1             OWN A:
;  199        M 1751  1                 BITVECTOR[BSA$K_NUM_ATRS]
;  200        M 1752  1                 PSECT(CODE)
;  201          1753  1                 PRESET(DEFINE_BITVECTOR_1(%REMAINING)) %;
;  202          1754  1
;  203          1755  1
;  204        P 1756  1 DEFINE_BITVECTOR(MASK_BSR,
;  205        P 1757  1         SSNAME,         COMMAND,        COMMENT,        USERNAME,
;  206        P 1758  1         USERUIC,        DATE,           OPSYS,          SYSVER,
;  207        P 1759  1         NODENAME,       SIR,            DRIVEID,        BACKVER,
;  208        P 1760  1         BLOCKSIZE,      XORSIZE,        BUFFERS,        VOLSETNAM,
;  209          1761  1         NVOLS,          BACKSIZE,       BACKFILES,      CRYPDATKEY);
;  210          1762  1
;  211          1763  1
;  212        P 1764  1 DEFINE_BITVECTOR(MASK_VSR,
;  213        P 1765  1         VOLSTRUCT,      VOLNAME,        OWNERNAME,      FORMAT,
;  214        P 1766  1         RVN,            VOLOWNER,       PROTECT,        FILEPROT,
;  215        P 1767  1         RECPROT,        VOLCHAR,        VOLDATE,        WINDOW,
;  216        P 1768  1         LRU_LIM,        EXTEND,         CLUSTER,        RESFILES,
;  217        P 1769  1         VOLSIZE,        TOTSIZE,        TOTFILES,       MAXFILES,
;  218        P 1770  1         MAXFILNUM,      SERIALNUM,      INDEXLBN,       BOOTBLOCK,
;  219          1771  1         RETAINMIN,      RETAINMAX);
;  220          1772  1
;  221          1773  1
;  222        P 1774  1 DEFINE_BITVECTOR(MASK_FAR,
;  223        P 1775  1         FILENAME,       STRUCLEV,       FID,            BACKLINK,
;  224        P 1776  1         FILESIZE,       UIC,            FPRO,           RPRO,
;  225        P 1777  1         ACLEVEL,        UCHAR,          RECATTR,        REVISION,
;  226        P 1778  1         CREDATE,        REVDATE,        EXPDATE,        BAKDATE,
;  227        P 1779  1         BOOTVBN,        PLACEMENT,      DIR_UIC,        DIR_FPRO,
;  228        P 1780  1         DIR_STATUS,     DIR_VERLIM,     VERLIMIT,       ACLSEGMENT,
;  229          1781  1         HIGHWATER,      JNL_FLAGS);
;  230          1782  1
;  231          1783  1
;  232        P 1784  1 DEFINE_BITVECTOR(MASK_PVA,
;  233        P 1785  1         SECTORS,        TRACKS,         CYLINDERS,      MAXBLOCK,
;  234        P 1786  1         DEVTYP,         SERIAL,         DEVNAM,         LABEL,
;  235          1787  1         BADBLOCK);
;  236          1788  1
;  237          1789  1 !
;  238          1790  1 ! Module wide own storage
;  239          1791  1 !
;  240          1792  1 OWN
;  241          1793  1     SAVESET_ENCR : LONG ;
;  242          1794  1
```

```
;  244          1795  1 %SBTTL 'GET_VM - allocate virtual memory'
;  245          1796  1 GLOBAL ROUTINE GET_VM(SIZE)=
;  246          1797  1
;  247          1798  1 !++
;  248          1799  1 !
;  249          1800  1 ! FUNCTIONAL DESCRIPTION:
;  250          1801  1 !       This routine interfaces to LIB$GET_VM to allocate a block of virtual
;  251          1802  1 !       memory.
;  252          1803  1 !
;  253          1804  1 ! INPUT PARAMETERS:
;  254          1805  1 !       SIZE            - Size in bytes of area.
;  255          1806  1 !
;  256          1807  1 ! IMPLICIT INPUTS:
;  257          1808  1 !       NONE
;  258          1809  1 !
;  259          1810  1 ! OUTPUT PARAMETERS:
;  260          1811  1 !       NONE
;  261          1812  1 !
;  262          1813  1 ! IMPLICIT OUTPUTS:
;  263          1814  1 !       NONE
;  264          1815  1 !
;  265          1816  1 ! ROUTINE VALUE:
;  266          1817  1 !       Address of allocated area.
;  267          1818  1 !
;  268          1819  1 ! SIDE EFFECTS:
;  269          1820  1 !       If allocation fails, a fatal error is signalled.
;  270          1821  1 !
;  271          1822  1 !--
;  272          1823  1
;  273          1824  2 BEGIN
;  274          1825  2 LOCAL
;  275          1826  2         STATUS,                                 ! General status variable
;  276          1827  2         AREA;                                   ! Pointer to allocated area
;  277          1828  2
;  278          1829  2
;  279          1830  2 STATUS = LIB$GET_VM(SIZE, AREA);
;  280          1831  2 IF NOT .STATUS THEN SIGNAL(BACKUP$_ALLOCMEM, 0, .STATUS);
;  281          1832  2 .AREA
;  282          1833  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
```

```
0012C 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
```

```
;  325          1874  1 %SBTTL 'FREE_VM - release virtual memory'
;  326          1875  1 GLOBAL ROUTINE FREE_VM(SIZE,AREA): NOVALUE=
;  327          1876  1
;  328          1877  1 !++
;  329          1878  1 !
;  330          1879  1 ! FUNCTIONAL DESCRIPTION:
;  331          1880  1 !       This routine interfaces to LIB$FREE_VM to free a block of virtual
;  332          1881  1 !       memory.
;  333          1882  1 !
;  334          1883  1 ! INPUT PARAMETERS:
;  335          1884  1 !       SIZE            - Size of allocated area.
;  336          1885  1 !       AREA            - Address of allocated area.
;  337          1886  1 !
;  338          1887  1 ! IMPLICIT INPUTS:
;  339          1888  1 !       NONE
;  340          1889  1 !
;  341          1890  1 ! OUTPUT PARAMETERS:
;  342          1891  1 !       NONE
;  343          1892  1 !
;  344          1893  1 ! IMPLICIT OUTPUTS:
;  345          1894  1 !       NONE
;  346          1895  1 !
;  347          1896  1 ! ROUTINE VALUE:
;  348          1897  1 !       NONE
;  349          1898  1 !
;  350          1899  1 ! SIDE EFFECTS:
;  351          1900  1 !       If deallocation fails, a fatal error is signalled.
;  352          1901  1 !
;  353          1902  1 !--
;  354          1903  1
;  355          1904  2 BEGIN
;  356          1905  2 LOCAL
;  357          1906  2         STATUS;                                 ! General status variable
;  358          1907  2
;  359          1908  2
;  360          1909  2 STATUS = LIB$FREE_VM(SIZE, AREA);
;  361          1910  2 IF NOT .STATUS THEN SIGNAL(BACKUP$_FREEMEM, 0, .STATUS);
;  362          1911  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
```

```
;  364          1912  1 %SBTTL 'DEBLOCK - deblock and free save set buffer'
;  365          1913  1 GLOBAL ROUTINE DEBLOCK(BCB,ROUT,NOFREE): L_P$ NOVALUE=
;  366          1914  1
;  367          1915  1 !++
;  368          1916  1 !
;  369          1917  1 ! FUNCTIONAL DESCRIPTION:
;  370          1918  1 !       This routine deblocks a buffer, and then frees the buffer unless
;  371          1919  1 !       inhibited.  Validity checking of the size and type values is done.
;  372          1920  1 !
;  373          1921  1 ! INPUT PARAMETERS:
;  374          1922  1 !       BCB             - Pointer to buffer control block.
;  375          1923  1 !       ROUT            - Routine to be called with each record.
;  376          1924  1 !       NOFREE          - True if buffer not to be freed (optional).
;  377          1925  1 !
;  378          1926  1 ! IMPLICIT INPUTS:
;  379          1927  1 !       NONE
;  380          1928  1 !
;  381          1929  1 ! OUTPUT PARAMETERS:
;  382          1930  1 !       NONE
;  383          1931  1 !
;  384          1932  1 ! IMPLICIT OUTPUTS:
;  385          1933  1 !       NONE
;  386          1934  1 !
;  387          1935  1 ! ROUTINE VALUE:
;  388          1936  1 !       NONE
;  389          1937  1 !
;  390          1938  1 ! SIDE EFFECTS:
;  391          1939  1 !       The buffer may be released.
;  392          1940  1 !
;  393          1941  1 !--
;  394          1942  1
;  395          1943  2 BEGIN
;  396          1944  2 MAP
;  397          1945  2         BCB:            REF BBLOCK;     ! Pointer to buffer control block
;  398          1946  2 LOCAL
;  399          1947  2         REC:            REF BBLOCK,     ! Pointer to record
;  400          1948  2         BUF:            REF BBLOCK,     ! Pointer to buffer header
;  401          1949  2         LIMIT;                          ! Limit address of buffer
;  402          1950  2 BUILTIN
;  403          1951  2         ACTUALCOUNT;
;  404          1952  2 L_DECL;
;  405          1953  2
;  406          1954  2
;  407          1955  2 ! Initialize.
;  408          1956  2 !
;  409          1957  2 BUF = .BCB[BCB_BUFFER];
;  410          1958  2 LIMIT = .BUF + .BCB[BCB_SIZE];
;  411          1959  2 REC = .BUF + BBH$C_LENGTH;
;  412          1960  2
;  413          1961  2
;  414          1962  2 ! Loop over all records in the buffer.
;  415          1963  2 !
;  416          1964  2 WHILE .REC LEQA .LIMIT - BRH$C_LENGTH DO
;  417          1965  3     BEGIN
;  418          1966  3     LOCAL
;  419          1967  3         RLIMIT;                     ! Limit address of this record
;  420          1968  3
```

```
;  421          1969  3
;  422          1970  3      ! Compute address of following byte and check that it is within the buffer.
;  423          1971  3      !
;  424          1972  3      RLIMIT = .REC + .REC[BRH$W_RSIZE] + BRH$C_LENGTH;
;  425          1973  3      IF .RLIMIT GTRA .LIMIT
;  426          1974  3      THEN
;  427          1975  4          BEGIN
;  428          1976  4          SIGNAL(BACKUP$_INVRECSIZ);
;  429          1977  4          EXITLOOP;
;  430          1978  3          END;
;  431          1979  3
;  432          1980  3
;  433          1981  3      ! Check record type.  Ensure that it is a valid type, and if it is, ensure
;  434          1982  3      ! that it is valid for the type of operation (physical or non-physical)
;  435          1983  3      ! that is in progress.
;  436          1984  3      !
;  437          1985  3      IF .REC[BRH$W_RTYPE] GTRU BRH$K_FILE_EXT
;  438          1986  3      THEN
;  439          1987  3          SIGNAL(BACKUP$_INVRECTYP)
;  440          1988  3      ELSE
;  441          1989  4          BEGIN
;  442          1990  4          IF NOT .COM_VALID_TYPES[.REC[BRH$W_RTYPE]]
;  443          1991  4          THEN
;  444          1992  4              IF .QUAL[QUAL_PHYS]
;  445          1993  4                  THEN SIGNAL(BACKUP$_NONPHYSET)
;  446          1994  4                  ELSE SIGNAL(BACKUP$_PHYSET);
;  447          1995  4
;  448          1996  4
;  449          1997  4          ! Check record size.  If a data record, the size must be a multiple of
;  450          1998  4          ! 512 bytes.  Otherwise, it must be a multiple of 16 bytes.  If all
;  451          1999  4          ! tests are passed, call the action routine to process the record.
;  452          2000  4          !
;  453          2001  4          IF
;  454          2002  5              BEGIN
;  455          2003  5              IF
;  456          2004  6                  (.REC[BRH$W_RTYPE] EQL BRH$K_VBN OR
;  457          2005  6                   .REC[BRH$W_RTYPE] EQL BRH$K_LBN)
;  458          2006  5              THEN
;  459          2007  5                  .(REC[BRH$W_RSIZE])<0,9> NEQ 0
;  460          2008  5              ELSE
;  461          2009  5                  .(REC[BRH$W_RSIZE])<0,4> NEQ 0
;  462          2010  5              END
;  463          2011  4          THEN
;  464          2012  4              SIGNAL(BACKUP$_INVRECSIZ)
;  465          2013  4          ELSE
;  466          2014  4              (.ROUT)(.REC, .BCB);
;  467          2015  3          END;
;  468          2016  3
;  469          2017  3
;  470          2018  3      ! Advance to next record.
;  471          2019  3      !
;  472          2020  3      REC = .RLIMIT;
;  473          2021  2      END;
;  474          2022  2
;  475          2023  2
;  476          2024  2  ! If warranted, free the buffer.
;  477          2025  2  !
```

```
;  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
```

```
;  485          2032  1 %SBTTL 'DEBLOCK_ATTR - deblock and validate attribute record'
;  486          2033  1 GLOBAL ROUTINE DEBLOCK_ATTR(REC,MASK,ROUT): L_P$ NOVALUE=
;  487          2034  1
;  488          2035  1 !++
;  489          2036  1 !
;  490          2037  1 ! FUNCTIONAL DESCRIPTION:
;  491          2038  1 !       This routine deblocks an attribute record.  Validity checking of the
;  492          2039  1 !       size and type values is done.
;  493          2040  1 !
;  494          2041  1 ! INPUT PARAMETERS:
;  495          2042  1 !       REC             - Pointer to attribute record.
;  496          2043  1 !       MASK            - Pointer to bitvector of valid attribute codes,
;  497          2044  1 !                         or zero to consider all defined codes valid.
;  498          2045  1 !       ROUT            - Routine to be called with each attribute.
;  499          2046  1 !
;  500          2047  1 ! IMPLICIT INPUTS:
;  501          2048  1 !       NONE
;  502          2049  1 !
;  503          2050  1 ! OUTPUT PARAMETERS:
;  504          2051  1 !       NONE
;  505          2052  1 !
;  506          2053  1 ! IMPLICIT OUTPUTS:
;  507          2054  1 !       NONE
;  508          2055  1 !
;  509          2056  1 ! ROUTINE VALUE:
;  510          2057  1 !       NONE
;  511          2058  1 !
;  512          2059  1 ! SIDE EFFECTS:
;  513          2060  1 !       NONE
;  514          2061  1 !
;  515          2062  1 !--
;  516          2063  1
;  517          2064  2 BEGIN
;  518          2065  2 MAP
;  519          2066  2         REC:            REF BBLOCK,     ! Pointer to record
;  520          2067  2         MASK:           REF BITVECTOR;  ! Pointer to bit mask
;  521          2068  2 LOCAL
;  522          2069  2         ATT:            REF BBLOCK,     ! Pointer to attribute
;  523          2070  2         LIMIT;                          ! Limit address of record
;  524          2071  2 L_DECL;
;  525          2072  2
;  526          2073  2
;  527          2074  2 ! Initialize.
;  528          2075  2 !
;  529          2076  2 ATT = .REC + BRH$C_LENGTH;
;  530          2077  2 LIMIT = .ATT + .REC[BRH$W_RSIZE];
;  531          2078  2
;  532          2079  2
;  533          2080  2 ! Check the structure level.
;  534          2081  2 !
;  535          2082  2 IF .ATT[BSA$W_STRUCLEV] NEQ BBH$K_LEVEL1 THEN SIGNAL(BACKUP$_INVATTSTR);
;  536          2083  2 ATT = .ATT + 2;
;  537          2084  2
;  538          2085  2
;  539          2086  2 ! Loop over all attribute entries within the record.
;  540          2087  2
;  541          2088  2 WHILE .ATT LEQA .LIMIT - BSA$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
```

```
;  591          2137  1 %SBTTL 'FILE_ERROR - output file related error message'
;  592          2138  1 GLOBAL ROUTINE FILE_ERROR(MESSAGE,FAB,EXTRA1,EXTRA2): NOVALUE=
;  593          2139  1
;  594          2140  1 !++
;  595          2141  1 !
;  596          2142  1 ! FUNCTIONAL DESCRIPTION:
;  597          2143  1 !       This routine signals a file-related message.
;  598          2144  1 !
;  599          2145  1 ! INPUT PARAMETERS:
;  600          2146  1 !       MESSAGE         - Message code for first message
;  601          2147  1 !       FAB             - Pointer to file context area, from which file name
;  602          2148  1 !                         will be obtained
;  603          2149  1 !       Up to two additional input parameters are additional messages --
;  604          2150  1 !       except that if the message is one of the special cases, they are
;  605          2151  1 !       additional FAO arguments.
;  606          2152  1 !
;  607          2153  1 ! IMPLICIT INPUTS:
;  608          2154  1 !       NONE
;  609          2155  1 !
;  610          2156  1 ! OUTPUT PARAMETERS:
;  611          2157  1 !       NONE
;  612          2158  1 !
;  613          2159  1 ! IMPLICIT OUTPUTS:
;  614          2160  1 !       NONE
;  615          2161  1 !
;  616          2162  1 ! ROUTINE VALUE:
;  617          2163  1 !       NONE
;  618          2164  1 !
;  619          2165  1 ! SIDE EFFECTS:
;  620          2166  1 !       The messages are signalled.
;  621          2167  1 !
;  622          2168  1 !--
;  623          2169  1
;  624          2170  2 BEGIN
;  625          2171  2 MAP
;  626          2172  2         FAB:            REF BBLOCK;     ! Pointer to FAB
;  627          2173  2 LOCAL
;  628          2174  2         NAM:            REF BBLOCK,     ! Pointer to NAM block
;  629          2175  2         DESC:           VECTOR[2],      ! Descriptor for file name
;  630          2176  2         PARAM:          VECTOR[6];      ! Signal parameter list
;  631          2177  2 BUILTIN
;  632          2178  2         ACTUALCOUNT,
;  633          2179  2         CALLG;
;  634          2180  2
;  635          2181  2
;  636          2182  2 ! Establish the file name to be printed.  The resultant string, expanded
;  637          2183  2 ! string, and filename string are examined in that order, and the first
;  638          2184  2 ! one that is not null is used.
;  639          2185  2 !
;  640          2186  2 NAM = .FAB[FAB$L_NAM];
;  641          2187  2 IF .NAM[NAM$B_RS[] NEQ 0
;  642          2188  2 THEN
;  643          2189  3     BEGIN
;  644          2190  3     DESC[0] = .NAM[NAM$B_RSL];
;  645          2191  3     DESC[1] = .NAM[NAM$L_RSA];
;  646          2192  3     END
;  647          2193  2 ELSE IF .NAM[NAM$B_ESL] NEQ 0
```

```
;  648          2194  2 THEN
;  649          2195  3     BEGIN
;  650          2196  3     DESC[0] = .NAM[NAM$B_ESL];
;  651          2197  3     DESC[1] = .NAM[NAM$L_ESA];
;  652          2198  3     END
;  653          2199  2 ELSE
;  654          2200  3     BEGIN
;  655          2201  3     DESC[0] = .FAB[FAB$B_FNS];
;  656          2202  3     DESC[1] = .FAB[FAB$L_FNA];
;  657          2203  2     END;
;  658          2204  2
;  659          2205  2
;  660          2206  2 ! Initialize the signal parameter list.
;  661          2207  2 !
;  662          2208  2 PARAM[0] = 3;                                  ! Parameter count
;  663          2209  2 PARAM[1] = .MESSAGE;                          ! First message code
;  664          2210  2 PARAM[2] = 1;                                  ! FAO argument count
;  665          2211  2 PARAM[3] = DESC;                               ! Filename descriptor
;  666          2212  2 IF ACTUALCOUNT() GEQ 3
;  667          2213  2 THEN
;  668          2214  3     BEGIN
;  669          2215  3     PARAM[0] = .PARAM[0] + 1;                  ! Increment parameter count
;  670          2216  3     PARAM[4] = .EXTRA1;                        ! Next message code
;  671          2217  2     END;
;  672          2218  2 IF ACTUALCOUNT() GEQ 4
;  673          2219  2 THEN
;  674          2220  3     BEGIN
;  675          2221  3     PARAM[0] = .PARAM[0] + 1;                  ! Increment parameter count
;  676          2222  3     PARAM[5] = .EXTRA2;                        ! Next message code
;  677          2223  2     END;
;  678          2224  2
;  679          2225  2
;  680          2226  2 ! For these messages, the additional parameters are FAO arguments.
;  681          2227  2 !
;  682          2228  2 IF
;  683          2229  2     .MESSAGE EQL BACKUP$_VBNMISSING OR
;  684          2230  2     .MESSAGE EQL BACKUP$_VBNERROR OR
;  685          2231  2     .MESSAGE EQL BACKUP$_VBNPASTEOF
;  686          2232  2 THEN
;  687          2233  2     PARAM[2] = 3;
;  688          2234  2
;  689          2235  2
;  690          2236  2 ! For these messages, EXTRA1 is an additional FAO argument that appears before
;  691          2237  2 ! the file name.  EXTRA2, if present, continues to be a message code.
;  692          2238  2 !
;  693          2239  2 IF
;  694          2240  2     .MESSAGE EQL BACKUP$_READBLOCK OR
;  695          2241  2     .MESSAGE EQL BACKUP$_WRITEBLOCK OR
;  696          2242  2     .MESSAGE EQL BACKUP$_VERIFYERR OR
;  697          2243  2     .MESSAGE EQL BACKUP$_BADDATA OR
;  698          2244  2     .MESSAGE EQL BACKUP$_SOFTRERRS OR
;  699          2245  2     .MESSAGE EQL BACKUP$_SOFTWERRS OR
;  700          2246  2     .MESSAGE EQL BACKUP$_XORERRS
;  701          2247  2 THEN
;  702          2248  3     BEGIN
;  703          2249  3     PARAM[2] = 2;
;  704          2250  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
```

```
;  714          2259  1 %SBTTL 'FIND_BADBLOCK - find block in bad block table'
;  715          2260  1 GLOBAL ROUTINE FIND_BADBLOCK(TABLE,LBN)=
;  716          2261  1
;  717          2262  1 !++
;  718          2263  1 !
;  719          2264  1 ! FUNCTIONAL DESCRIPTION:
;  720          2265  1 !       This routine searches a specified bad block table for a specified LBN.
;  721          2266  1 !
;  722          2267  1 ! INPUT PARAMETERS:
;  723          2268  1 !       TABLE           - Pointer to bad block table.
;  724          2269  1 !       LBN             - LBN to search for.
;  725          2270  1 !
;  726          2271  1 ! IMPLICIT INPUTS:
;  727          2272  1 !       NONE
;  728          2273  1 !
;  729          2274  1 ! OUTPUT PARAMETERS:
;  730          2275  1 !       NONE
;  731          2276  1 !
;  732          2277  1 ! IMPLICIT OUTPUTS:
;  733          2278  1 !       NONE
;  734          2279  1 !
;  735          2280  1 ! ROUTINE VALUE:
;  736          2281  1 !       True if the LBN is found, false otherwise.
;  737          2282  1 !
;  738          2283  1 ! SIDE EFFECTS:
;  739          2284  1 !       NONE
;  740          2285  1 !
;  741          2286  1 !--
;  742          2287  1
;  743          2288  2 BEGIN
;  744          2289  2 MAP
;  745          2290  2         TABLE:          REF BBLOCK;     ! Pointer to bad block table
;  746          2291  2 LOCAL
;  747          2292  2         DESC:           REF BBLOCK;     ! Pointer to bad block descriptor
;  748          2293  2
;  749          2294  2
;  750          2295  2 ! If there is no table, assume not found.
;  751          2296  2 !
;  752          2297  2 IF .TABLE EQL 0 THEN RETURN FALSE;
;  753          2298  2
;  754          2299  2
;  755          2300  2 ! Loop over all entries of the table.
;  756          2301  2 !
;  757          2302  2 DESC = TABLE[BAD_DESC];
;  758          2303  2 INCR I FROM 0 TO .TABLE[BAD_NUMDESC]-1 DO
;  759          2304  3     BEGIN
;  760          2305  3
;  761          2306  3     ! If the LBN is in the range described by this entry, return found.
;  762          2307  3     !
;  763          2308  3     IF
;  764          2309  3         .LBN GEQU .DESC[BAD_LBN] AND
;  765          2310  3         .LBN LEQU .DESC[BAD_LBN] + .DESC[BAD_COUNT] - 1
;  766          2311  3     THEN
;  767          2312  3         RETURN TRUE;
;  768          2313  3
;  769          2314  3
;  770          2315  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
```

```
;   781          2325  1 %SBTTL 'EXTRACT_FILENAME - get file name from FAB'
;   782          2326  1 GLOBAL ROUTINE EXTRACT_FILENAME(FAB,DESC): NOVALUE=
;   783          2327  1
;   784          2328  1 !++
;   785          2329  1 !
;   786          2330  1 ! FUNCTIONAL DESCRIPTION:
;   787          2331  1 !       This routine extracts the filename, and filetype if it is not null,
;   788          2332  1 !       from the resultant string associated with FAB and returns a
;   789          2333  1 !       descriptor in DESC.
;   790          2334  1 !
;   791          2335  1 ! INPUT PARAMETERS:
;   792          2336  1 !       FAB             - Pointer to the FAB
;   793          2337  1 !       DESC            - Pointer to the output descriptor
;   794          2338  1 !
;   795          2339  1 ! IMPLICIT INPUTS:
;   796          2340  1 !       NONE
;   797          2341  1 !
;   798          2342  1 ! OUTPUT PARAMETERS:
;   799          2343  1 !       NONE
;   800          2344  1 !
;   801          2345  1 ! IMPLICIT OUTPUTS:
;   802          2346  1 !       NONE
;   803          2347  1 !
;   804          2348  1 ! ROUTINE VALUE:
;   805          2349  1 !       NONE
;   806          2350  1 !
;   807          2351  1 ! SIDE EFFECTS:
;   808          2352  1 !       NONE
;   809          2353  1 !
;   810          2354  1 !--
;   811          2355  1
;   812          2356  2 BEGIN
;   813          2357  2 MAP
;   814          2358  2         FAB:            REF BBLOCK,             ! Pointer to FAB
;   815          2359  2         DESC:           REF BBLOCK;             ! Pointer to descriptor
;   816          2360  2
;   817          2361  2
;   818          2362  2 ! Construct a descriptor for the file name and file type, concatenated.
;   819          2363  2 !
;   820          2364  2 DESC[DSC$W_LENGTH] = .BBLOCK[FAB[FC_NAM], NAM$B_NAME]
;   821          2365  2                   + .BBLOCK[FAB[FC_NAM], NAM$B_TYPE];
;   822          2366  2 DESC[DSC$A_POINTER] = .BBLOCK[FAB[FC_NAM], NAM$L_NAME];
;   823          2367  2
;   824          2368  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

```
;   826          2369  1 %SBTTL 'EXTRACT_DIR_FILENAME - get directory and file name from FAB'
;   827          2370  1 GLOBAL ROUTINE EXTRACT_DIR_FILENAME(FAB,DESC): NOVALUE=
;   828          2371  1
;   829          2372  1 !++
;   830          2373  1 !
;   831          2374  1 ! FUNCTIONAL DESCRIPTION:
;   832          2375  1 !       This routine extracts the directory, name, type, and version from the
;   833          2376  1 !       resultant string associated with FAB and returns a descriptor in DESC.
;   834          2377  1 !
;   835          2378  1 ! INPUT PARAMETERS:
;   836          2379  1 !       FAB             - Pointer to the FAB
;   837          2380  1 !       DESC            - Pointer to the output descriptor
;   838          2381  1 !
;   839          2382  1 ! IMPLICIT INPUTS:
;   840          2383  1 !       NONE
;   841          2384  1 !
;   842          2385  1 ! OUTPUT PARAMETERS:
;   843          2386  1 !       NONE
;   844          2387  1 !
;   845          2388  1 ! IMPLICIT OUTPUTS:
;   846          2389  1 !       NONE
;   847          2390  1 !
;   848          2391  1 ! ROUTINE VALUE:
;   849          2392  1 !       NONE
;   850          2393  1 !
;   851          2394  1 ! SIDE EFFECTS:
;   852          2395  1 !       NONE
;   853          2396  1 !
;   854          2397  1 !--
;   855          2398  1
;   856          2399  2 BEGIN
;   857          2400  2 MAP
;   858          2401  2         FAB:            REF BBLOCK,             ! Pointer to FAB
;   859          2402  2         DESC:           REF BBLOCK;             ! Pointer to descriptor
;   860          2403  2
;   861          2404  2
;   862          2405  2 ! Construct a descriptor for the portion of the resultant string beginning
;   863          2406  2 ! with the directory name.
;   864          2407  2 !
;   865          2408  2 DESC[DSC$W_LENGTH] =
;   866          2409  2     .BBLOCK[FAB[FC_NAM], NAM$L_RSA] +
;   867          2410  2     .BBLOCK[FAB[FC_NAM], NAM$B_RSL] -
;   868          2411  2     .BBLOCK[FAB[FC_NAM], NAM$L_DIR];
;   869          2412  2 DESC[DSC$A_POINTER] = .BBLOCK[FAB[FC_NAM], NAM$L_DIR];
;   870          2413  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                     ;
                          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
```

```
;  872          2414  1 %SBTTL 'GET_ONE_ATTRIBUTE - save value of specified attribute'
;  873          2415  1 ROUTINE GET_ONE_ATTRIBUTE(ATT): NOVALUE=
;  874          2416  1
;  875          2417  1 !++
;  876          2418  1 !
;  877          2419  1 ! FUNCTIONAL DESCRIPTION:
;  878          2420  1 !       This routine saves the value of one attribute.
;  879          2421  1 !
;  880          2422  1 ! INPUT PARAMETERS:
;  881          2423  1 !       ATT             - Pointer to attribute entry from save set record.
;  882          2424  1 !
;  883          2425  1 ! IMPLICIT INPUTS:
;  884          2426  1 !       NONE
;  885          2427  1 !
;  886          2428  1 ! OUTPUT PARAMETERS:
;  887          2429  1 !       NONE
;  888          2430  1 !
;  889          2431  1 ! IMPLICIT OUTPUTS:
;  890          2432  1 !       OUTPUT_ATTBUF   - Contains the attribute value.
;  891          2433  1 !
;  892          2434  1 ! ROUTINE VALUE:
;  893          2435  1 !       NONE
;  894          2436  1 !
;  895          2437  1 ! SIDE EFFECTS:
;  896          2438  1 !       NONE
;  897          2439  1 !
;  898          2440  1 !--
;  899          2441  1
;  900          2442  2 BEGIN
;  901          2443  2 MAP
;  902          2444  2         ATT:            REF BBLOCK;     ! Pointer to attribute entry
;  903          2445  2 LOCAL
;  904          2446  2         DLEN,                           ! Length of attribute storage
;  905          2447  2         DST:            REF BBLOCK;     ! Pointer to attribute storage
;  906          2448  2
;  907          2449  2
;  908          2450  2 ! Initially, dispatch on the attribute code to establish where in OUTPUT_ATTBUF
;  909          2451  2 ! the value of the attribute is stored.  Also, establish whether a descriptor
;  910          2452  2 ! for, or the value of, the attribute will be stored.  If the value, establish
;  911          2453  2 ! the length of the attribute storage.
;  912          2454  2 !
;  913          2455  2 IF
;  914          2456  3     BEGIN
;  915          2457  3     CASE .ATT[BSA$W_TYPE] FROM BSA$K_SSNAME TO BSA$K_NUM_ATRS-1 OF
;  916          2458  3         SET
;  917          2459  3
;  918          2460  3         [BSA$K_SSNAME]:
;  919          2461  4             BEGIN
;  920          2462  4             DST = OUTPUT_ATTBUF[BSR_SSNAME];
;  921          2463  4             TRUE
;  922          2464  3             END;
;  923          2465  3
;  924          2466  3         [BSA$K_COMMAND]:
;  925          2467  4             BEGIN
;  926          2468  4             DST = OUTPUT_ATTBUF[BSR_COMMAND];
;  927          2469  4             TRUE
;  928          2470  3             END;
```

```
;   929          2471  3
;   930          2472  3          [BSA$K_COMMENT]:
;   931          2473  4              BEGIN
;   932          2474  4              DST = OUTPUT_ATTBUF[BSR_COMMENT];
;   933          2475  4              TRUE
;   934          2476  3              END;
;   935          2477  3
;   936          2478  3          [BSA$K_USERNAME]:
;   937          2479  4              BEGIN
;   938          2480  4              DST = OUTPUT_ATTBUF[BSR_USERNAME];
;   939          2481  4              TRUE
;   940          2482  3              END;
;   941          2483  3
;   942          2484  3          [BSA$K_USERUIC]:
;   943          2485  4              BEGIN
;   944          2486  4              DST = OUTPUT_ATTBUF[BSR_USERUIC];
;   945          2487  4              DLEN = 4;
;   946          2488  4              FALSE
;   947          2489  3              END;
;   948          2490  3
;   949          2491  3          [BSA$K_DATE]:
;   950          2492  4              BEGIN
;   951          2493  4              DST = OUTPUT_ATTBUF[BSR_DATE];
;   952          2494  4              DLEN = 8;
;   953          2495  4              FALSE
;   954          2496  3              END;
;   955          2497  3
;   956          2498  3          [BSA$K_OPSYS]:
;   957          2499  4              BEGIN
;   958          2500  4              DST = OUTPUT_ATTBUF[BSR_OPSYS];
;   959          2501  4              DLEN = 2;
;   960          2502  4              FALSE
;   961          2503  3              END;
;   962          2504  3
;   963          2505  3          [BSA$K_SYSVER]:
;   964          2506  4              BEGIN
;   965          2507  4              DST = OUTPUT_ATTBUF[BSR_SYSVER];
;   966          2508  4              TRUE
;   967          2509  3              END;
;   968          2510  3
;   969          2511  3          [BSA$K_NODENAME]:
;   970          2512  4              BEGIN
;   971          2513  4              DST = OUTPUT_ATTBUF[BSR_NODENAME];
;   972          2514  4              TRUE
;   973          2515  3              END;
;   974          2516  3
;   975          2517  3          [BSA$K_SIR]:
;   976          2518  4              BEGIN
;   977          2519  4              DST = OUTPUT_ATTBUF[BSR_SIR];
;   978          2520  4              DLEN = 4;
;   979          2521  4              FALSE
;   980          2522  3              END;
;   981          2523  3
;   982          2524  3          [BSA$K_DRIVEID]:
;   983          2525  4              BEGIN
;   984          2526  4              DST = OUTPUT_ATTBUF[BSR_DRIVEID];
;   985          2527  4              TRUE
```

```
;  986          2528  3              END;
;  987          2529  3
;  988          2530  3          [BSA$K_BACKVER]:
;  989          2531  4              BEGIN
;  990          2532  4              DST = OUTPUT_ATTBUF[BSR_BACKVER];
;  991          2533  4              TRUE
;  992          2534  3              END;
;  993          2535  3
;  994          2536  3          [BSA$K_BLOCKSIZE]:
;  995          2537  4              BEGIN
;  996          2538  4              DST = OUTPUT_ATTBUF[BSR_BLOCKSIZE];
;  997          2539  4              DLEN = 4;
;  998          2540  4              FALSE
;  999          2541  3              END;
; 1000          2542  3
; 1001          2543  3          [BSA$K_XORSIZE]:
; 1002          2544  4              BEGIN
; 1003          2545  4              DST = OUTPUT_ATTBUF[BSR_XORSIZE];
; 1004          2546  4              DLEN = 2;
; 1005          2547  4              FALSE
; 1006          2548  3              END;
; 1007          2549  3
; 1008          2550  3          [BSA$K_BUFFERS]:
; 1009          2551  4              BEGIN
; 1010          2552  4              DST = OUTPUT_ATTBUF[BSR_BUFFERS];
; 1011          2553  4              DLEN = 2;
; 1012          2554  4              FALSE
; 1013          2555  3              END;
; 1014          2556  3
; 1015          2557  3          [BSA$K_VOLSETNAM]:
; 1016          2558  4              BEGIN
; 1017          2559  4              DST = OUTPUT_ATTBUF[BSR_VOLSETNAM];
; 1018          2560  4              TRUE
; 1019          2561  3              END;
; 1020          2562  3
; 1021          2563  3          [BSA$K_NVOLS]:
; 1022          2564  4              BEGIN
; 1023          2565  4              DST = OUTPUT_ATTBUF[BSR_NVOLS];
; 1024          2566  4              DLEN = 2;
; 1025          2567  4              FALSE
; 1026          2568  3              END;
; 1027          2569  3
; 1028          2570  3          [BSA$K_BACKSIZE]:
; 1029          2571  4              BEGIN
; 1030          2572  4              DST = OUTPUT_ATTBUF[BSR_BACKSIZE];
; 1031          2573  4              DLEN = 8;
; 1032          2574  4              FALSE
; 1033          2575  3              END;
; 1034          2576  3
; 1035          2577  3          [BSA$K_BACKFILES]:
; 1036          2578  4              BEGIN
; 1037          2579  4              DST = OUTPUT_ATTBUF[BSR_BACKFILES];
; 1038          2580  4              DLEN = 4;
; 1039          2581  4              FALSE
; 1040          2582  3              END;
; 1041          2583  3
; 1042          2584  3          [BSA$K_VOLSTRUCT]:
```

```
;  1043          2585  4              BEGIN
;  1044          2586  4              DST = OUTPUT_ATTBUF[VSR_VOLSTRUCT];
;  1045          2587  4              DLEN = 2;
;  1046          2588  4              FALSE
;  1047          2589  3              END;
;  1048          2590  3
;  1049          2591  3          [BSA$K_VOLNAME]:
;  1050          2592  4              BEGIN
;  1051          2593  4              DST = OUTPUT_ATTBUF[VSR_VOLNAME];
;  1052          2594  4              TRUE
;  1053          2595  3              END;
;  1054          2596  3
;  1055          2597  3          [BSA$K_OWNERNAME]:
;  1056          2598  4              BEGIN
;  1057          2599  4              DST = OUTPUT_ATTBUF[VSR_OWNERNAME];
;  1058          2600  4              TRUE
;  1059          2601  3              END;
;  1060          2602  3
;  1061          2603  3          [BSA$K_FORMAT]:
;  1062          2604  4              BEGIN
;  1063          2605  4              DST = OUTPUT_ATTBUF[VSR_FORMAT];
;  1064          2606  4              TRUE
;  1065          2607  3              END;
;  1066          2608  3
;  1067          2609  3          [BSA$K_RVN]:
;  1068          2610  4              BEGIN
;  1069          2611  4              DST = OUTPUT_ATTBUF[VSR_RVN];
;  1070          2612  4              DLEN = 2;
;  1071          2613  4              FALSE
;  1072          2614  3              END;
;  1073          2615  3
;  1074          2616  3          [BSA$K_VOLOWNER]:
;  1075          2617  4              BEGIN
;  1076          2618  4              DST = OUTPUT_ATTBUF[VSR_VOLOWNER];
;  1077          2619  4              DLEN = 4;
;  1078          2620  4              FALSE
;  1079          2621  3              END;
;  1080          2622  3
;  1081          2623  3          [BSA$K_PROTECT]:
;  1082          2624  4              BEGIN
;  1083          2625  4              DST = OUTPUT_ATTBUF[VSR_PROTECT];
;  1084          2626  4              DLEN = 2;
;  1085          2627  4              FALSE
;  1086          2628  3              END;
;  1087          2629  3
;  1088          2630  3          [BSA$K_FILEPROT]:
;  1089          2631  4              BEGIN
;  1090          2632  4              DST = OUTPUT_ATTBUF[VSR_FILEPROT];
;  1091          2633  4              DLEN = 2;
;  1092          2634  4              FALSE
;  1093          2635  3              END;
;  1094          2636  3
;  1095          2637  3          [BSA$K_RECPROT]:
;  1096          2638  4              BEGIN
;  1097          2639  4              DST = OUTPUT_ATTBUF[VSR_RECPROT];
;  1098          2640  4              DLEN = 2;
;  1099          2641  4              FALSE
```

```
;  1100          2642  3              END;
;  1101          2643  3
;  1102          2644  3          [BSA$K_VOLCHAR]:
;  1103          2645  4              BEGIN
;  1104          2646  4              DST = OUTPUT_ATTBUF[VSR_VOLCHAR];
;  1105          2647  4              DLEN = 2;
;  1106          2648  4              FALSE
;  1107          2649  3              END;
;  1108          2650  3
;  1109          2651  3          [BSA$K_VOLDATE]:
;  1110          2652  4              BEGIN
;  1111          2653  4              DST = OUTPUT_ATTBUF[VSR_VOLDATE];
;  1112          2654  4              DLEN = 8;
;  1113          2655  4              FALSE
;  1114          2656  3              END;
;  1115          2657  3
;  1116          2658  3          [BSA$K_WINDOW]:
;  1117          2659  4              BEGIN
;  1118          2660  4              DST = OUTPUT_ATTBUF[VSR_WINDOW];
;  1119          2661  4              DLEN = 1;
;  1120          2662  4              FALSE
;  1121          2663  3              END;
;  1122          2664  3
;  1123          2665  3          [BSA$K_LRU_LIM]:
;  1124          2666  4              BEGIN
;  1125          2667  4              DST = OUTPUT_ATTBUF[VSR_LRU_LIM];
;  1126          2668  4              DLEN = 1;
;  1127          2669  4              FALSE
;  1128          2670  3              END;
;  1129          2671  3
;  1130          2672  3          [BSA$K_EXTEND]:
;  1131          2673  4              BEGIN
;  1132          2674  4              DST = OUTPUT_ATTBUF[VSR_EXTEND];
;  1133          2675  4              DLEN = 2;
;  1134          2676  4              FALSE
;  1135          2677  3              END;
;  1136          2678  3
;  1137          2679  3          [BSA$K_CLUSTER]:
;  1138          2680  4              BEGIN
;  1139          2681  4              DST = OUTPUT_ATTBUF[VSR_CLUSTER];
;  1140          2682  4              DLEN = 2;
;  1141          2683  4              FALSE
;  1142          2684  3              END;
;  1143          2685  3
;  1144          2686  3          [BSA$K_RESFILES]:
;  1145          2687  4              BEGIN
;  1146          2688  4              DST = OUTPUT_ATTBUF[VSR_RESFILES];
;  1147          2689  4              DLEN = 2;
;  1148          2690  4              FALSE
;  1149          2691  3              END;
;  1150          2692  3
;  1151          2693  3          [BSA$K_VOLSIZE]:
;  1152          2694  4              BEGIN
;  1153          2695  4              DST = OUTPUT_ATTBUF[VSR_VOLSIZE];
;  1154          2696  4              DLEN = 4;
;  1155          2697  4              FALSE
;  1156          2698  3              END;
```

```
;  1157          2699  3
;  1158          2700  3          [BSA$K_TOTSIZE]:
;  1159          2701  4              BEGIN
;  1160          2702  4              DST = OUTPUT_ATTBUF[VSR_TOTSIZE];
;  1161          2703  4              DLEN = 8;
;  1162          2704  4              FALSE
;  1163          2705  3              END;
;  1164          2706  3
;  1165          2707  3          [BSA$K_TOTFILES]:
;  1166          2708  4              BEGIN
;  1167          2709  4              DST = OUTPUT_ATTBUF[VSR_TOTFILES];
;  1168          2710  4              DLEN = 4;
;  1169          2711  4              FALSE
;  1170          2712  3              END;
;  1171          2713  3
;  1172          2714  3          [BSA$K_MAXFILES]:
;  1173          2715  4              BEGIN
;  1174          2716  4              DST = OUTPUT_ATTBUF[VSR_MAXFILES];
;  1175          2717  4              DLEN = 4;
;  1176          2718  4              FALSE
;  1177          2719  3              END;
;  1178          2720  3
;  1179          2721  3          [BSA$K_MAXFILNUM]:
;  1180          2722  4              BEGIN
;  1181          2723  4              DST = OUTPUT_ATTBUF[VSR_MAXFILNUM];
;  1182          2724  4              DLEN = 4;
;  1183          2725  4              FALSE
;  1184          2726  3              END;
;  1185          2727  3
;  1186          2728  3          [BSA$K_SERIALNUM]:
;  1187          2729  4              BEGIN
;  1188          2730  4              DST = OUTPUT_ATTBUF[VSR_SERIALNUM];
;  1189          2731  4              DLEN = 4;
;  1190          2732  4              FALSE
;  1191          2733  3              END;
;  1192          2734  3
;  1193          2735  3          [BSA$K_FILENAME]:
;  1194          2736  4              BEGIN
;  1195          2737  4              DST = OUTPUT_ATTBUF[FAR_FILENAME];
;  1196          2738  4              TRUE
;  1197          2739  3              END;
;  1198          2740  3
;  1199          2741  3          [BSA$K_STRUCLEV]:
;  1200          2742  4              BEGIN
;  1201          2743  4              DST = OUTPUT_ATTBUF[FAR_STRUCLEV];
;  1202          2744  4              DLEN = 2;
;  1203          2745  4              FALSE
;  1204          2746  3              END;
;  1205          2747  3
;  1206          2748  3          [BSA$K_FID]:
;  1207          2749  4              BEGIN
;  1208          2750  4              DST = OUTPUT_ATTBUF[FAR_FID];
;  1209          2751  4              DLEN = 6;
;  1210          2752  4              FALSE
;  1211          2753  3              END;
;  1212          2754  3
;  1213          2755  3          [BSA$K_BACKLINK]:
```

```
;  1214          2756  4              BEGIN
;  1215          2757  4              DST = OUTPUT_ATTBUF[FAR_BACKLINK];
;  1216          2758  4              DLEN = 6;
;  1217          2759  4              FALSE
;  1218          2760  3              END;
;  1219          2761  3
;  1220          2762  3          [BSA$K_FILESIZE]:
;  1221          2763  4              BEGIN
;  1222          2764  4              DST = OUTPUT_ATTBUF[FAR_FILESIZE];
;  1223          2765  4              DLEN = 4;
;  1224          2766  4              FALSE
;  1225          2767  3              END;
;  1226          2768  3
;  1227          2769  3          [BSA$K_UIC]:
;  1228          2770  4              BEGIN
;  1229          2771  4              DST = OUTPUT_ATTBUF[FAR_UIC];
;  1230          2772  4              DLEN = 4;
;  1231          2773  4              FALSE
;  1232          2774  3              END;
;  1233          2775  3
;  1234          2776  3          [BSA$K_FPRO]:
;  1235          2777  4              BEGIN
;  1236          2778  4              DST = OUTPUT_ATTBUF[FAR_FPRO];
;  1237          2779  4              DLEN = 2;
;  1238          2780  4              FALSE
;  1239          2781  3              END;
;  1240          2782  3
;  1241          2783  3          [BSA$K_RPRO]:
;  1242          2784  4              BEGIN
;  1243          2785  4              DST = OUTPUT_ATTBUF[FAR_RPRO];
;  1244          2786  4              DLEN = 2;
;  1245          2787  4              FALSE
;  1246          2788  3              END;
;  1247          2789  3
;  1248          2790  3          [BSA$K_ACLEVEL]:
;  1249          2791  4              BEGIN
;  1250          2792  4              DST = OUTPUT_ATTBUF[FAR_ACLEVEL];
;  1251          2793  4              DLEN = 1;
;  1252          2794  4              FALSE
;  1253          2795  3              END;
;  1254          2796  3
;  1255          2797  3          [BSA$K_UCHAR]:
;  1256          2798  4              BEGIN
;  1257          2799  4              DST = OUTPUT_ATTBUF[FAR_UCHAR];
;  1258          2800  4              DLEN = 4;
;  1259          2801  4              FALSE
;  1260          2802  3              END;
;  1261          2803  3
;  1262          2804  3          [BSA$K_RECATTR]:
;  1263          2805  4              BEGIN
;  1264          2806  4              DST = OUTPUT_ATTBUF[FAR_RECATTR];
;  1265          2807  4              DLEN = 32;
;  1266          2808  4              FALSE
;  1267          2809  3              END;
;  1268          2810  3
;  1269          2811  3          [BSA$K_REVISION]:
;  1270          2812  4              BEGIN
```

```
; 1271          2813  4             DST = OUTPUT_ATTBUF[FAR_REVISION];
; 1272          2814  4             DLEN = 2;
; 1273          2815  4             FALSE
; 1274          2816  3             END;
; 1275          2817  3
; 1276          2818  3         [BSA$K_CREDATE]:
; 1277          2819  4             BEGIN
; 1278          2820  4             DST = OUTPUT_ATTBUF[FAR_CREDATE];
; 1279          2821  4             DLEN = 8;
; 1280          2822  4             FALSE
; 1281          2823  3             END;
; 1282          2824  3
; 1283          2825  3         [BSA$K_REVDATE]:
; 1284          2826  4             BEGIN
; 1285          2827  4             DST = OUTPUT_ATTBUF[FAR_REVDATE];
; 1286          2828  4             DLEN = 8;
; 1287          2829  4             FALSE
; 1288          2830  3             END;
; 1289          2831  3
; 1290          2832  3         [BSA$K_EXPDATE]:
; 1291          2833  4             BEGIN
; 1292          2834  4             DST = OUTPUT_ATTBUF[FAR_EXPDATE];
; 1293          2835  4             DLEN = 8;
; 1294          2836  4             FALSE
; 1295          2837  3             END;
; 1296          2838  3
; 1297          2839  3         [BSA$K_BAKDATE]:
; 1298          2840  4             BEGIN
; 1299          2841  4             DST = OUTPUT_ATTBUF[FAR_BAKDATE];
; 1300          2842  4             DLEN = 8;
; 1301          2843  4             FALSE
; 1302          2844  3             END;
; 1303          2845  3
; 1304          2846  3         [BSA$K_SECTORS]:
; 1305          2847  4             BEGIN
; 1306          2848  4             DST = OUTPUT_ATTBUF[PVA_SECTORS];
; 1307          2849  4             DLEN = 1;
; 1308          2850  4             FALSE
; 1309          2851  3             END;
; 1310          2852  3
; 1311          2853  3         [BSA$K_TRACKS]:
; 1312          2854  4             BEGIN
; 1313          2855  4             DST = OUTPUT_ATTBUF[PVA_TRACKS];
; 1314          2856  4             DLEN = 1;
; 1315          2857  4             FALSE
; 1316          2858  3             END;
; 1317          2859  3
; 1318          2860  3         [BSA$K_CYLINDERS]:
; 1319          2861  4             BEGIN
; 1320          2862  4             DST = OUTPUT_ATTBUF[PVA_CYLINDERS];
; 1321          2863  4             DLEN = 2;
; 1322          2864  4             FALSE
; 1323          2865  3             END;
; 1324          2866  3
; 1325          2867  3         [BSA$K_MAXBLOCK]:
; 1326          2868  4             BEGIN
; 1327          2869  4             DST = OUTPUT_ATTBUF[PVA_MAXBLOCK];
```

```
;  1328     2870  4              DLEN = 4;
;  1329     2871  4              FALSE
;  1330     2872  3              END;
;  1331     2873  3
;  1332     2874  3          [BSA$K_DEVTYP]:
;  1333     2875  4              BEGIN
;  1334     2876  4              DST = OUTPUT_ATTBUF[PVA_DEVTYP];
;  1335     2877  4              DLEN = 4;
;  1336     2878  4              FALSE
;  1337     2879  3              END;
;  1338     2880  3
;  1339     2881  3          [BSA$K_SERIAL]:
;  1340     2882  4              BEGIN
;  1341     2883  4              DST = OUTPUT_ATTBUF[PVA_SERIAL];
;  1342     2884  4              DLEN = 4;
;  1343     2885  4              FALSE
;  1344     2886  3              END;
;  1345     2887  3
;  1346     2888  3          [BSA$K_DEVNAM]:
;  1347     2889  4              BEGIN
;  1348     2890  4              DST = OUTPUT_ATTBUF[PVA_DEVNAM];
;  1349     2891  4              TRUE
;  1350     2892  3              END;
;  1351     2893  3
;  1352     2894  3          [BSA$K_LABEL]:
;  1353     2895  4              BEGIN
;  1354     2896  4              DST = OUTPUT_ATTBUF[PVA_LABEL];
;  1355     2897  4              TRUE
;  1356     2898  3              END;
;  1357     2899  3
;  1358     2900  3          [BSA$K_BADBLOCK]:
;  1359     2901  4              BEGIN
;  1360     2902  4              DST = OUTPUT_ATTBUF[PVA_BADBLOCK];
;  1361     2903  4              TRUE
;  1362     2904  3              END;
;  1363     2905  3
;  1364     2906  3          [BSA$K_INDEXLBN]:
;  1365     2907  4              BEGIN
;  1366     2908  4              DST = OUTPUT_ATTBUF[VSR_INDEXLBN];
;  1367     2909  4              DLEN = 4;
;  1368     2910  4              FALSE
;  1369     2911  3              END;
;  1370     2912  3
;  1371     2913  3          [BSA$K_BOOTBLOCK]:
;  1372     2914  4              BEGIN
;  1373     2915  4              DST = OUTPUT_ATTBUF[VSR_BOOTBLOCK];
;  1374     2916  4              TRUE
;  1375     2917  3              END;
;  1376     2918  3
;  1377     2919  3          [BSA$K_BOOTVBN]:
;  1378     2920  4              BEGIN
;  1379     2921  4              DST = OUTPUT_ATTBUF[FAR_BOOTVBN];
;  1380     2922  4              DLEN = 4;
;  1381     2923  4              FALSE
;  1382     2924  3              END;
;  1383     2925  3
;  1384     2926  3          [BSA$K_PLACEMENT]:
```

```
;  1385          2927  4              BEGIN
;  1386          2928  4              DST = OUTPUT_ATTBUF[FAR_PLACEMENT];
;  1387          2929  4              TRUE
;  1388          2930  3              END;
;  1389          2931  3
;  1390          2932  3          [BSA$K_DIR_UIC]:
;  1391          2933  4              BEGIN
;  1392          2934  4              DST = OUTPUT_ATTBUF[FAR_DIR_UIC];
;  1393          2935  4              DLEN = 4;
;  1394          2936  4              FALSE
;  1395          2937  3              END;
;  1396          2938  3
;  1397          2939  3          [BSA$K_DIR_FPRO]:
;  1398          2940  4              BEGIN
;  1399          2941  4              DST = OUTPUT_ATTBUF[FAR_DIR_FPRO];
;  1400          2942  4              DLEN = 2;
;  1401          2943  4              FALSE
;  1402          2944  3              END;
;  1403          2945  3
;  1404          2946  3          [BSA$K_DIR_STATUS]:
;  1405          2947  4              BEGIN
;  1406          2948  4              DST = OUTPUT_ATTBUF[FAR_DIR_STATUS];
;  1407          2949  4              DLEN = 1;
;  1408          2950  4              FALSE
;  1409          2951  3              END;
;  1410          2952  3
;  1411          2953  3          [BSA$K_DIR_VERLIM]:
;  1412          2954  4              BEGIN
;  1413          2955  4              DST = OUTPUT_ATTBUF[FAR_DIR_VERLIM];
;  1414          2956  4              DLEN = 2;
;  1415          2957  4              FALSE
;  1416          2958  3              END;
;  1417          2959  3
;  1418          2960  3          [BSA$K_VERLIMIT]:
;  1419          2961  4              BEGIN
;  1420          2962  4              DST = OUTPUT_ATTBUF[FAR_VERLIMIT];
;  1421          2963  4              DLEN = 2;
;  1422          2964  4              FALSE
;  1423          2965  3              END;
;  1424          2966  3
;  1425          2967  3          [BSA$K_RETAINMIN]:
;  1426          2968  4              BEGIN
;  1427          2969  4              DST = OUTPUT_ATTBUF[VSR_RETAINMIN];
;  1428          2970  4              DLEN = 8;
;  1429          2971  4              FALSE
;  1430          2972  3              END;
;  1431          2973  3
;  1432          2974  3          [BSA$K_RETAINMAX]:
;  1433          2975  4              BEGIN
;  1434          2976  4              DST = OUTPUT_ATTBUF[VSR_RETAINMAX];
;  1435          2977  4              DLEN = 8;
;  1436          2978  4              FALSE
;  1437          2979  3              END;
;  1438          2980  3
;  1439          2981  3           BSA$K_ACLSEGMENT]:
;  1440          2982  4              BEGIN
;  1441          2983  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        C2  A6  AF 00011     CASEW   2(R6), #1, #81                      ;
    00A5        01D3  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
```

```
;  1529          3069  1 %SBTTL 'GET_VOLUME_ATTRIBUTES - process volume summary record'
;  1530          3070  1 GLOBAL ROUTINE GET_VOLUME_ATTRIBUTES(REC): NOVALUE=
;  1531          3071  1
;  1532          3072  1 !++
;  1533          3073  1 !
;  1534          3074  1 ! FUNCTIONAL DESCRIPTION:
;  1535          3075  1 !       This routine processes a volume summary record, expanding
;  1536          3076  1 !       the attributes it contains into the OUTPUT_ATTBUF area.
;  1537          3077  1 !
;  1538          3078  1 ! INPUT PARAMETERS:
;  1539          3079  1 !       REC             - Pointer to attribute record.
;  1540          3080  1 !
;  1541          3081  1 ! IMPLICIT INPUTS:
;  1542          3082  1 !       NONE
;  1543          3083  1 !
;  1544          3084  1 ! OUTPUT PARAMETERS:
;  1545          3085  1 !       NONE
;  1546          3086  1 !
;  1547          3087  1 ! IMPLICIT OUTPUTS:
;  1548          3088  1 !       OUTPUT_ATTBUF   - Contains the attribute values.
;  1549          3089  1 !
;  1550          3090  1 ! ROUTINE VALUE:
;  1551          3091  1 !       NONE
;  1552          3092  1 !
;  1553          3093  1 ! SIDE EFFECTS:
;  1554          3094  1 !       NONE
;  1555          3095  1 !
;  1556          3096  1 !--
;  1557          3097  1
;  1558          3098  2 BEGIN
;  1559          3099  2 GLOBAL REGISTER P$ = 11;
;  1560          3100  2 CH$FILL(0, VSR_LENGTH, OUTPUT_ATTBUF);
;  1561          3101  2 DEBLOCK_ATTR(.REC, MASK_VSR, GET_ONE_ATTRIBUTE);
;  1562          3102  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
```

```
;  1671          3207  1 %SBTTL 'CRYPTO_CHKSAV - Check to see if saveset is incrypted'
;  1672          3208  1 GLOBAL ROUTINE CRYPTO_CHKSAV(BCB) =
;  1673          3209  1
;  1674          3210  1 !++
;  1675          3211  1 !
;  1676          3212  1 ! FUNCTIONAL DESCRIPTION:
;  1677          3213  1 !       This routine locates the backup summary record in a block and
;  1678          3214  1 !       and determines if the saveset is encrypted.
;  1679          3215  1 !
;  1680          3216  1 ! INPUT PARAMETERS:
;  1681          3217  1 !       BCB = address of block containing the backup summary record.
;  1682          3218  1 !
;  1683          3219  1 ! IMPLICIT INPUTS:
;  1684          3220  1 !       NONE
;  1685          3221  1 !
;  1686          3222  1 ! OUTPUT PARAMETERS:
;  1687          3223  1 !       NONE
;  1688          3224  1 !
;  1689          3225  1 ! IMPLICIT OUTPUTS:
;  1690          3226  1 !
;  1691          3227  1 !
;  1692          3228  1 ! ROUTINE VALUE:
;  1693          3229  1 !       TRUE if the saveset is encrypted
;  1694          3230  1 !       FALSE if not encrypted.
;  1695          3231  1 !
;  1696          3232  1 ! SIDE EFFECTS:
;  1697          3233  1 !
;  1698          3234  1 !
;  1699          3235  1 !--
;  1700          3236  1
;  1701          3237  2 BEGIN
;  1702          3238  2 GLOBAL REGISTER
;  1703          3239  2     PS = 11:    REF VECTOR;             ! linkage for data passed to inner routine
;  1704          3240  2 MAP
;  1705          3241  2     BCB:        REF BBLOCK;             ! parameter points to buffer
;  1706          3242  2
;  1707          3243  2 !
;  1708          3244  2 ! Set flag to indicate whether saveset is encrypted
;  1709          3245  2 !
;  1710          3246  2 SAVESET_ENCR = 0 ;
;  1711          3247  2
;  1712          3248  2 !
;  1713          3249  2 ! Use the BACKUP facility routine DEBLOCK to extract all records
;  1714          3250  2 ! of the buffer.
;  1715          3251  2 !
;  1716          3252  2 DEBLOCK (.BCB, SEARCHFOR_BSR_ENC , TRUE ) ;
;  1717          3253  2
;  1718          3254  2
;  1719          3255  2 IF .SAVESET_ENCR EQL 0
;  1720          3256  2     THEN RETURN FALSE
;  1721          3257  2     ELSE RETURN TRUE ;
;  1722          3258  1 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
```

```
;  1725          3260  1 %SBTTL 'SEARCHFOR_BSR_ENC - Search for Backup Summary in block'
;  1726          3261  1 ROUTINE SEARCHFOR_BSR_ENC(PTR): L_P$ NOVALUE =
;  1727          3262  1
;  1728          3263  1 !++
;  1729          3264  1 !
;  1730          3265  1 ! FUNCTIONAL DESCRIPTION:
;  1731          3266  1 !       This routine simply checks for an encrypted save set
;  1732          3267  1 !
;  1733          3268  1 ! INPUT PARAMETERS:
;  1734          3269  1 !       PTR = address of record to examine
;  1735          3270  1 !
;  1736          3271  1 ! IMPLICIT INPUTS:
;  1737          3272  1 !       NONE
;  1738          3273  1 !
;  1739          3274  1 ! OUTPUT PARAMETERS:
;  1740          3275  1 !       NONE
;  1741          3276  1 !
;  1742          3277  1 ! IMPLICIT OUTPUTS:
;  1743          3278  1 !       NONE
;  1744          3279  1 !
;  1745          3280  1 ! ROUTINE VALUE:
;  1746          3281  1 !       NONE
;  1747          3282  1 !
;  1748          3283  1 ! SIDE EFFECTS:
;  1749          3284  1 !       The listing is produced.
;  1750          3285  1 !
;  1751          3286  1 !--
;  1752          3287  1
;  1753          3288  2 BEGIN
;  1754          3289  2
;  1755          3290  2 EXTERNAL ROUTINE
;  1756          3291  2     GET_SUMMARY_ATTRIBUTES ;
;  1757          3292  2
;  1758          3293  2 GLOBAL REGISTER
;  1759          3294  2     P$ = 11:    REF VECTOR;             ! linkage for data passed to inner routine
;  1760          3295  2
;  1761          3296  2 MAP
;  1762          3297  2     PTR:        REF BBLOCK;             ! address of record from block to examine
;  1763          3298  2
;  1764          3299  2 !
;  1765          3300  2 ! We are only interested in backup summary records
;  1766          3301  2 !
;  1767          3302  2 IF .PTR[BRH$W_RTYPE] NEQU BRH$K_SUMMARY THEN RETURN;
;  1768          3303  2
;  1769          3304  2 GET_SUMMARY_ATTRIBUTES(.PTR);
;  1770          3305  2
;  1771          3306  2 SAVESET_ENCR = .BBLOCK[OUTPUT_ATTBUF[BSR_CRYPTDATA], BSA$B_CRYPTYP] ;
;  1772          3307  2
;  1773          3308  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
```

```
;  1776          3310  1 %SBTTL 'DECODE_DEVTYP - decode the devtyp phys vol attr'
;  1777          3311  1 GLOBAL ROUTINE DECODE_DEVTYP ( MEDIA_ID , NAME_LENGTH,
;  1778          3312  1                                NAME_BUFFER: REF VECTOR[,BYTE],
;  1779          3313  1                                TYPE_LENGTH,
;  1780          3314  1                                TYPE_BUFFER: REF VECTOR[,BYTE] ): NOVALUE=
;  1781          3315  1
;  1782          3316  1 !++
;  1783          3317  1 !
;  1784          3318  1 ! FUNCTIONAL DESCRIPTION:
;  1785          3319  1 !       This routine takes the longword value and interprets it as a
;  1786          3320  1 !       UCB MEDIA_ID and converts it to ascii
;  1787          3321  1 !
;  1788          3322  1 ! INPUT PARAMETERS:
;  1789          3323  1 !       MEDIA_ID        - Longword value
;  1790          3324  1 !       NAME_LENGTH     - Address of longword containing the name buffer length
;  1791          3325  1 !       TYPE_LENGTH     - Address of longword containing the type buffer length
;  1792          3326  1 !
;  1793          3327  1 ! IMPLICIT INPUTS:
;  1794          3328  1 !       NONE
;  1795          3329  1 !
;  1796          3330  1 ! OUTPUT PARAMETERS:
;  1797          3331  1 !       NAME_LENGTH     - address of longword to recieve name string length
;  1798          3332  1 !       NAME_BUFFER     - address of buffer to receive name string
;  1799          3333  1 !       TYPE_LENGTH     - address of longword to recieve type string length
;  1800          3334  1 !       TYPE_BUFFER     - address of buffer to receive type string
;  1801          3335  1 !
;  1802          3336  1 ! IMPLICIT OUTPUTS:
;  1803          3337  1 !
;  1804          3338  1 ! ROUTINE VALUE:
;  1805          3339  1 !       NONE
;  1806          3340  1 !
;  1807          3341  1 ! SIDE EFFECTS:
;  1808          3342  1 !       NONE
;  1809          3343  1 !
;  1810          3344  1 !--
;  1811          3345  1
;  1812          3346  2 BEGIN
;  1813          3347  2
;  1814          3348  2 FIELD
;  1815          3349  2     ! Field definitions for UCB$L_MEDIA_ID
;  1816          3350  2     !
;  1817          3351  2     MEDIA_ID_FIELDS =
;  1818          3352  2         SET
;  1819          3353  2         MEDIA_ID_T0 = [UCB$V_MEDIA_ID_T0],
;  1820          3354  2         MEDIA_ID_T1 = [UCB$V_MEDIA_ID_T1],
;  1821          3355  2         MEDIA_ID_N0 = [UCB$V_MEDIA_ID_N0],
;  1822          3356  2         MEDIA_ID_N1 = [UCB$V_MEDIA_ID_N1],
;  1823          3357  2         MEDIA_ID_N2 = [UCB$V_MEDIA_ID_N2],
;  1824          3358  2         MEDIA_ID_NN = [UCB$V_MEDIA_ID_NN]
;  1825          3359  2         TES ;
;  1826          3360  2
;  1827          3361  2 ! We only reserve a byte for this number.
;  1828          3362  2 !
;  1829          3363  2 $ASSUME (%FIELDEXPAND(MEDIA_ID_NN,2), LEQ, 8) ;
;  1830          3364  2
;  1831          3365  2 STRUCTURE
;  1832          3366  2         ! Define a structure to ignore the offset.
```

```
;  1833          3367  2          ! This way we can use the UCB symbols that are already
;  1834          3368  2          ! defined.
;  1835          3369  2          !
;  1836          3370  2          MEDIA_ID_STRUCT[O,P,S,E;N]=
;  1837          3371  2              [N]
;  1838          3372  2              (MEDIA_ID_STRUCT)<P,S,E>;
;  1839          3373  2
;  1840          3374  2  MAP
;  1841          3375  2      MEDIA_ID : MEDIA_ID_STRUCT FIELD (MEDIA_ID_FIELDS) ;
;  1842          3376  2
;  1843          3377  2  BIND
;  1844          3378  2      NAME_INLEN = .NAME_LENGTH : LONG ,
;  1845          3379  2      TYPE_INLEN = .TYPE_LENGTH : LONG ;
;  1846          3380  2
;  1847          3381  2  LOCAL
;  1848          3382  2      OUTLEN    : LONG            INITIAL (0) ,
;  1849          3383  2      CTRL_DESC : BBLOCK[8]       PRESET (
;  1850          3384  2                                    [DSC$W_LENGTH] = %CHARCOUNT('!2ZB'),
;  1851          3385  2                                    [DSC$B_DTYPE]  = DSC$K_DTYPE_Z,
;  1852          3386  2                                    [DSC$B_CLASS]  = DSC$K_CLASS_S,
;  1853          3387  2                                    [DSC$A_POINTER]= UPLIT BYTE(%!2ZB')),
;  1854          3388  2      OUT_DESC  : BBLOCK[8]       PRESET (
;  1855          3389  2                                    [DSC$W_LENGTH] = 0,
;  1856          3390  2                                    [DSC$B_DTYPE]  = DSC$K_DTYPE_Z,
;  1857          3391  2                                    [DSC$B_CLASS]  = DSC$K_CLASS_S,
;  1858          3392  2                                    [DSC$A_POINTER]= 0),
;  1859          3393  2      NUMBER    : BYTE            INITIAL (0) ;
;  1860          3394  2
;  1861          3395  2  ! We must have at least 3 bytes in the name buffer
;  1862          3396  2  !
;  1863          3397  2  IF NAME_INLEN NEQ 0
;  1864          3398  2  THEN
;  1865          3399  3      BEGIN
;  1866          3400  3      IF (.NAME_INLEN GEQ 3) AND
;  1867          3401  4         (.NAME_BUFFER NEQ 0)
;  1868          3402  3      THEN
;  1869          3403  4          BEGIN
;  1870          3404  4
;  1871          3405  4          OUTLEN = 0 ;
;  1872          3406  4
;  1873          3407  4          ! Each character is assigned a number from 1 to 26 corresponding to A-z
;  1874          3408  4          !
;  1875          3409  4          IF .MEDIA_ID[MEDIA_ID_N0] NEQ 0
;  1876          3410  4          THEN
;  1877          3411  5              BEGIN
;  1878          3412  5              NAME_BUFFER[.OUTLEN] = %X'40' + .MEDIA_ID[MEDIA_ID_N0] ;
;  1879          3413  5              OUTLEN = .OUTLEN + 1 ;
;  1880          3414  4              END ;
;  1881          3415  4          IF .MEDIA_ID[MEDIA_ID_N1] NEQ 0
;  1882          3416  4          THEN
;  1883          3417  5              BEGIN
;  1884          3418  5              NAME_BUFFER[.OUTLEN] = %X'40' + .MEDIA_ID[MEDIA_ID_N1] ;
;  1885          3419  5              OUTLEN = .OUTLEN + 1 ;
;  1886          3420  4              END ;
;  1887          3421  4          IF .MEDIA_ID[MEDIA_ID_N2] NEQ 0
;  1888          3422  4          THEN
;  1889          3423  5              BEGIN
```

```
;1890        3424  5              NAME_BUFFER[.OUTLEN] = %X'40' + .MEDIA_ID[MEDIA_ID_N2] ;
;1891        3425  5              OUTLEN = .OUTLEN + 1 ;
;1892        3426  4              END ;
;1893        3427  4
;1894        3428  4          ! The last field is a number
;1895        3429  4          !
;1896        3430  4          NUMBER = .MEDIA_ID[MEDIA_ID_NN] ;
;1897        3431  4          OUT_DESC[DSC$W_LENGTH] = .NAME_INLEN - .OUTLEN ;
;1898        3432  4          OUT_DESC[DSC$A_POINTER] = NAME_BUFFER[.OUTLEN] ;
;1899      P 3433  4          $FAO (CTRL_DESC,                    ! CTRSTR
;1900      P 3434  4                OUT_DESC[DSC$W_LENGTH],       ! OUTLEN
;1901      P 3435  4                OUT_DESC,                     ! OUTBUF
;1902        3436  4                .NUMBER ) ;                   ! P1
;1903        3437  4          OUTLEN = .OUTLEN + .OUT_DESC[DSC$W_LENGTH] ;
;1904        3438  4
;1905        3439  4          NAME_INLEN = .OUTLEN ;
;1906        3440  4          END
;1907        3441  3      ELSE
;1908        3442  3          NAME_INLEN = 0 ;
;1909        3443  2      END ;
;1910        3444  2
;1911        3445  2  ! We must have at least 2 bytes in the type buffer
;1912        3446  2  !
;1913        3447  2  IF TYPE_INLEN NEQ 0
;1914        3448  2  THEN
;1915        3449  3      BEGIN
;1916        3450  3      IF (.TYPE_INLEN GEQ 2) AND
;1917        3451  4         (.TYPE_BUFFER NEQ 0)
;1918        3452  3      THEN
;1919        3453  4          BEGIN
;1920        3454  4
;1921        3455  4          OUTLEN = 0 ;
;1922        3456  4
;1923        3457  4          ! Each character is assigned a number from 1 to 26 corresponding to A-z
;1924        3458  4          !
;1925        3459  4          IF .MEDIA_ID[MEDIA_ID_T0] NEQ 0
;1926        3460  4          THEN
;1927        3461  5              BEGIN
;1928        3462  5              TYPE_BUFFER[.OUTLEN] = %X'40' + .MEDIA_ID[MEDIA_ID_T0] ;
;1929        3463  5              OUTLEN = .OUTLEN + 1 ;
;1930        3464  4              END ;
;1931        3465  4          IF .MEDIA_ID[MEDIA_ID_T1] NEQ 0
;1932        3466  4          THEN
;1933        3467  5              BEGIN
;1934        3468  5              TYPE_BUFFER[.OUTLEN] = %X'40' + .MEDIA_ID[MEDIA_ID_T1] ;
;1935        3469  5              OUTLEN = .OUTLEN + 1 ;
;1936        3470  4              END ;
;1937        3471  4
;1938        3472  4          TYPE_INLEN = .OUTLEN ;
;1939        3473  4          END
;1940        3474  3      ELSE
;1941        3475  3          TYPE_INLEN = 0 ;
;1942        3476  2      END ;
;1943        3477  2
;1944        3478  2  RETURN ;
;1945        3479  2
;1946        3480  1  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

STAINVOL
LIS

UTILITY
LIS

STATOSYS
LIS

STAPARSE
LIS

BADBLOCKS
LIS

BAD

STABAD
MAP

BADDATA
LIS

ANALYZBAD
MAP

ANALYZEMD
CLD

BADDEF
SDL

WRITESAVE
LIS

BADINIT
LIS

ANALYZCMD
LIS