```
LLL               BBBBBBBBBBBB      RRRRRRRRRRRR
LLL               BBBBBBBBBBBBB     RRRRRRRRRRRRR
LLL               BBBBBBBBBBBBB     RRRRRRRRRRRRR
LLL               BBB          BBB  RR           RRR
LLL               BBB          BBB  RRR          RRR
LLL               BBB          BBB  RRR          RRR
LLL               BBB          BBB  RRR          RRR
LLL               BBB          BBB  RRR          RRR
LLL               BBB          BBB  RRR          RRR
LLL               BBBBBBBBBBBBB     RRRRRRRRRRRRR
LLL               BBBBBBBBBBBBB     RRRRRRRRRRRRR
LLL               BBBBBBBBBBBBB     RRRRRRRRRRRRR
LLL               BBB          BBB  RRR   RRR
LLL               BBB          BBB  RRR   RRR
LLL               BBB          BBB  RRR   RRR
LLL               BBB          BBB  RRR      RRR
LLL               BBB          BBB  RRR      RRR
LLL               BBB          BBB  RRR      RRR
LLLLLLLLLLLLLLLL  BBBBBBBBBBBB      RRR         RRR
LLLLLLLLLLLLLLLL  BBBBBBBBBBBBB     RRR         RRR
LLLLLLLLLLLLLLLL  BBBBBBBBBBBBB     RRR         RRR
LLLLLLLLLLLLLLLL  BBBBBBBBBBBBB     RRR         RRR
```

```
_$2

Val
---
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
00F
00F
00F
00F
00F
00F
00F
00F
00F
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
```

FILEID**SUBS

```
  SSSSSSSS  UU      UU  BBBBBBBB      SSSSSSSS
  SSSSSSSS  UU      UU  BBBBBBBB      SSSSSSSS
SS          UU      UU  BB      BB  SS
SS          UU      UU  BB      BB  SS
SS          UU      UU  BB      BB  SS
SS          UU      UU  BB      BB  SS
  SSSSSS    UU      UU  BBBBBBBB      SSSSSS
  SSSSSS    UU      UU  BBBBBBBB      SSSSSS
        SS  UU      UU  BB      BB          SS
        SS  UU      UU  BB      BB          SS
        SS  UU      UU  BB      BB          SS      ....
        SS  UU      UU  BB      BB          SS      ....
SSSSSSSS    UUUUUUUUUU  BBBBBBBB  SSSSSSSS        ....
SSSSSSSS    UUUUUUUUUU  BBBBBBBB  SSSSSSSS        ....

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

```
;     1          0001  0 MODULE LBR_SUBS (                       ! General library procedure routines
;     2          0002  0                 LANGUAGE (BLISS32),
;     3          0003  0                 IDENT = 'V04-000'
;     4          0004  0                 ) =
;     5          0005  1 BEGIN
;     6          0006  1
;     7          0007  1 !
;     8          0008  1 !*****************************************************************************
;     9          0009  1 !*                                                                           *
;    10          0010  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;    11          0011  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
;    12          0012  1 !*  ALL RIGHTS RESERVED.                                                      *
;    13          0013  1 !*                                                                           *
;    14          0014  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
;    15          0015  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
;    16          0016  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
;    17          0017  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
;    18          0018  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
;    19          0019  1 !*  TRANSFERRED.                                                              *
;    20          0020  1 !*                                                                           *
;    21          0021  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
;    22          0022  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT     *
;    23          0023  1 !*  CORPORATION.                                                              *
;    24          0024  1 !*                                                                           *
;    25          0025  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
;    26          0026  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
;    27          0027  1 !*                                                                           *
;    28          0028  1 !*                                                                           *
;    29          0029  1 !*****************************************************************************
;    30          0030  1
;    31          0031  1 !++
;    32          0032  1 !
;    33          0033  1 ! FACILITY:  Library access procedures
;    34          0034  1 !
;    35          0035  1 ! ABSTRACT:
;    36          0036  1 !
;    37          0037  1 !       The VAX/VMS librarian procedures implement a standard access method
;    38          0038  1 !       to libraries through a shared, common procedure set.
;    39          0039  1 !
;    40          0040  1 ! ENVIRONMENT:
;    41          0041  1 !
;    42          0042  1 !       VAX native, user mode.
;    43          0043  1 !
;    44          0044  1 !--
;    45          0045  1 !
;    46          0046  1 !
;    47          0047  1 ! AUTHOR:  Tim Halvorsen, Benn Schreiber
;    48          0048  1 !
;    49          0049  1 ! CREATION DATE: June, 1979
;    50          0050  1 !
;    51          0051  1 ! MODIFIED BY:
;    52          0052  1 !
;    53          0053  1 !       V03-003         JWT0101         Jim Teague      20-Apr-1983
;    54          0054  1 !               Correct check for maximum number of control indexes.
;    55          0055  1 !               Control indexes run from 1 to 16, inclusive.
;    56          0056  1 !
;    57          0057  1 !--
```

```
;    58          0058  1
;    59          0059  1
```

```
;    61          0060  1 %SBTTL  'Declarations';
;    62          0061  1
;    63          0062  1 LIBRARY
;    64          0063  1         'SYS$LIBRARY:STARLET.L32';      ! System macros
;    65          0064  1 REQUIRE
;    66          0065  1         'PREFIX';                       ! Librarian general definitions
;    67          0204  1 REQUIRE
;    68          0205  1         'LBRDEF';                       ! Librarian structure definitions
;    69          0796  1
;    70          0797  1 EXTERNAL
;    71          0798  1     lbr$gl_maxread,                     ! Max blocks to read at once
;    72          0799  1     lbr$gl_hictl,                       ! Highest control block allocated
;    73          0800  1     lbr$gl_rmsstv,                      ! Returns STV on RMS errors
;    74          0801  1     lbr$gl_control: REF BBLOCK,         ! Pointer to current control block
;    75          0802  1     lbr$al_ctltab: VECTOR [lbr$c_maxctl]; ! Pointers to control blocks
;    76          0803  1
;    77          0804  1 EXTERNAL ROUTINE
;    78          0805  1     remove_cache : JSB_1,               ! Remove entry from cache
;    79          0806  1     lookup_cache : JSB_2,               ! Lookup entry in disk cache
;    80          0807  1     add_cache : JSB_2,                  ! Add entry to cache
;    81          0808  1     lbr$get_vm : JSB_2,                 ! Allocate memory
;    82          0809  1     lbr$free_vm: JSB_2;                 ! Deallocate memory
;    83          0810  1
;    84          0811  1 EXTERNAL LITERAL
;    85          0812  1     lbr$_illctl,
;    86          0813  1     lbr$_invkey,
;    87          0814  1     lbr$_keynotfnd,
;    88          0815  1     lbr$_libnotopn;
;    89          0816  1
;    90          0817  1 FORWARD ROUTINE
;    91          0818  1     get_mem : JSB_2,                    ! Allocate dynamic memory
;    92          0819  1     dealloc_mem : JSB_2,                ! Deallocate dynamic memory
;    93          0820  1     get_zmem : JSB_2,                   ! Allocated zeroed memory
;    94          0821  1     validate_ctl : JSB_1,               ! Validate control table
;    95          0822  1     alloc_block : JSB_2,                ! Allocate disk block
;    96          0823  1     dealloc_block : JSB_1,              ! Deallocate disk block                    ; Rc
;    97          0824  1     write_block : JSB_2,                ! Write disk block
;    98          0825  1     read_block : JSB_2,                 ! Read disk block
;    99          0826  1     read_n_block : JSB_2,               ! Read multiple disk blocks
;   100          0827  1     find_block : JSB_3,                 ! Read vbn from disk and cache if not there
;   101          0828  1     make_upper_case : JSB_3,            ! Guarantee name all upper case
;   102          0829  1     moveto_upper_case : JSB_3,          ! Upcase and move string
;   103          0830  1     incr_rfa: JSB_2 NOVALUE;            ! Increment RFA by byte count
;   104          0831  1
```

```
;  106     0832  1 %SBTTL  'MAKE_UPPER_CASE';
;  107     0833  1
;  108     0834  1 GLOBAL ROUTINE make_upper_case (idesc, odesc, upcase) : JSB_3 =
;  109     0835  2 BEGIN
;  110     0836  2 !++
;  111     0837  2 !
;  112     0838  2 ! Upper case the name described by string descriptor idesc
;  113     0839  2 ! Put the name at descriptor found by odesc. Length may be
;  114     0840  2 ! modified if trailing blanks.
;  115     0841  2 !
;  116     0842  2 ! Inputs:
;  117     0843  2 !
;  118     0844  2 !       idesc   string descriptor for the name or address of binary value
;  119     0845  2 !       odesc   string descriptor for output name or address of binary value
;  120     0846  2 !       upcase  if true then raise case, else just check length and copy
;  121     0847  2 !
;  122     0848  2 ! Outputs:
;  123     0849  2 !
;  124     0850  2 !       string converted to upper case. trailing blanks eliminated and
;  125     0851  2 !       odesc [dsc$w_length] modified to reflect this.
;  126     0852  2 !
;  127     0853  2 !       If binary keys are passed, the value is merely copied.
;  128     0854  2 !
;  129     0855  2 ! Return value:
;  130     0856  2 !
;  131     0857  2 !       true            name upper cased and length ok
;  132     0858  2 !       lbr$_invkey     name is 0-length
;  133     0859  2 !       lbr$_keynotfnd  name is too long
;  134     0860  2 !--
;  135     0861  2 MAP
;  136     0862  2     idesc : REF BBLOCK,
;  137     0863  2     odesc : REF BBLOCK;
;  138     0864  2
;  139     0865  2 BIND
;  140     0866  2     namlen = idesc [dsc$w_length] : WORD,
;  141     0867  2     iname = idesc [dsc$a_pointer] : REF VECTOR [,BYTE],
;  142     0868  2     index_desc = .lbr$gl_control  [lbr$l_hdrptr] + lhd$c_idxdesc
;  143     0869  3                         + (.lbr$gl_control [lbr$l_curidx] - 1)
;  144     0870  2                             * idd$c_length : BBLOCK,        !Point to index desc.
;  145     0871  2     oname = .odesc [dsc$a_pointer] : VECTOR [,BYTE];        !Output vector
;  146     0872  2
;  147     0873  2 IF NOT .index_desc [idd$v_ascii]                            !If binary keys
;  148     0874  3 THEN BEGIN
;  149     0875  3     .odesc = ..idesc;                                       ! then just copy the value
;  150     0876  3     RETURN true;
;  151     0877  2     END;
;  152     0878  2 !
;  153     0879  2 ! Handle ascii keys differently
;  154     0880  2 !
;  155     0881  2 IF .namlen EQL 0                                            !0-length name is illegal
;  156     0882  2         THEN RETURN lbr$_invkey;
;  157     0883  2 !
;  158     0884  2 ! Name length is ok, convert to upper case
;  159     0885  2 !
;  160     0886  2 INCRU i FROM 0 TO .namlen-1 DO
;  161     0887  3     BEGIN
;  162     0888  3     IF .iname [.i] EQL %ASCII ' '                           !If character is a space
```

```
;  163          0889  3       OR .iname [.i] EQL 0                    ! or a null
;  164          0890  3     THEN
;  165          0891  4         BEGIN                                 ! then adjust output descriptor and exit
;  166          0892  4         odesc [dsc$w_length] = .i;
;  167          0893  4         EXITLOOP;
;  168          0894  4         END
;  169          0895  3     ELSE
;  170          0896  4         IF (.iname [.i] GEQU %ASCII'a'        !copy name and convert to upper case
;  171          0897  4           AND .iname [.i] LEQU %ASCII'z'
;  172          0898  4           AND .upcase)                        ! if not upcasing then just copy
;  173          0899  4         THEN oname [.i] = .iname [.i] - (%ASCII'a' - %ASCII'A')
;  174          0900  3         ELSE oname [.i] = .iname [.i];
;  175          0901  2     END;
;  176          0902  2 IF .odesc [dsc$w_length] GTRU .index_desc [idd$w_keylen] !Check for name too long
;  177          0903  2     THEN RETURN [br$_keynotfnd
;  178          0904  2     ELSE RETURN true
;  179          0905  1 END;                                          !Of make_upper_case


                                .TITLE  LBR_SUBS
                                .IDENT  \V04-000\

                                .EXTRN  LBR$GL_MAXREAD, LBR$GL_HICTL
                                .EXTRN  LBR$GL_RMSSTV, LBR$GL_CONTROL
                                .EXTRN  LBR$AL_CTLTAB, REMOVE_CACHE
                                .EXTRN  LOOKUP_CACHE, ADD_CACHE
                                .EXTRN  LBR$GET_VM, LBR$FREE_VM
                                .EXTRN  LBR$_ILLCTL, LBR$_INVKEY
                                .EXTRN  LBR$_KEYNOTFND, LBR$_LIBNOTOPN

                                .PSECT  $CODE$,NOWRT,2

        00F8   8F   BB 00000 MAKE_UPPER_CASE::
                             PUSHR   #^M<R3,R4,R5,R6,R7>                ;  0834
53      0000G  CF   D0 00004         MOVL    LBR$GL_CONTROL, R3         ;  0868
54        12   A3   D0 00009         MOVL    18(R3), R4                 ;  0869
56        0A B344   7E 0000D         MOVAQ   @10(R3)[R4], R6            ;
56      00BC   C6   9E 00012         MOVAB   188(R6), R6                ;
54        04   A1   D0 00017         MOVL    4(ODESC), R4               ;  0871
05             66   E8 0001B         BLBS    (R6), 1$                   ;  0873
6?             60   D0 0001E         MOVL    (IDESC), (ODESC)           ;  0875
               59   11 00021         BRB     10$                        ;  0876
               60   B5 00023 1$:     TSTW    (IDESC)                    ;  0881
               09   12 00025         BNEQ    2$                         ;
50 00000000G   8F   D0 00027         MOVL    #LBR$_INVKEY, R0           ;  0882
               4F   11 0002E         BRB     11$                        ;
57             60   3C 00030 2$:     MOVZWL  (IDESC), R7                ;  0886
               57   D7 00033         DECL    R7                         ;
               53   D4 00035         CLRL    I                          ;
               2F   11 00037         BRB     8$                         ;
55        04 B043   9A 00039 3$:     MOVZBL  @4(IDESC)[I], R5           ;  0888
20             55   91 0003E         CMPB    R5, #32                    ;
               04   13 00041         BEQL    4$                         ;
               55   D5 00043         TSTL    R5                         ;  0889
               05   12 00045         BNEQ    5$                         ;
61             53   B0 00047 4$:     MOVW    I, (ODESC)                 ;  0892
               21   11 0004A         BRB     9$                         ;  0891
```

```
                61   8F              55  91 0004C 5$:      CMPB    R5, #97                                 ; 0896
                                     10  1F 00050          BLSSU   6$                                      ;
                7A   8F              55  91 00052          CMPB    R5, #122                                ; 0897
                                     0A  1A 00056          BGTRU   6$                                      ;
                     07              52  E9 00058          BLBC    UPCASE, 6$                              ; 0898
          6344       55              20  83 0005B          SUBB3   #32, R5, (I)[R4]                        ; 0899
                                     04  11 00060          BRB     7$                                      ;
             6344    55              55  90 00062 6$:      MOVB    R5, (I)[R4]                             ; 0900
                                     53  D6 00066 7$:      INCL    I                                       ; 0886
                     57              53  D1 00068 8$:      CMPL    I, R7                                   ;
                                     CC  1B 0006B          BLEQU   3$                                      ;
                02   A6              61  B1 0006D 9$:      CMPW    (ODESC), 2(R6)                          ; 0902
                                     09  1B 00071          BLEQU   10$                                     ;
                     50 00000000G    8F  D0 00073          MOVL    #LBR$_KEYNOTFND, R0                     ; 0904
                                     03  11 0007A          BRB     11$                                     ;
                     50              01  D0 0007C 10$:     MOVL    #1, R0                                  ;
                           00F8      8F  BA 0007F 11$:     POPR    #^M<R3,R4,R5,R6,R7>                     ; 0905
                                     05 00083              RSB                                             ;

; Routine Size:  132 bytes,    Routine Base:  $CODE$ + 0000
```

```
;  181          0906  1 %SBTTL  'MOVETO_UPPER_CASE';
;  182          0907  1
;  183          0908  1 GLOBAL ROUTINE moveto_upper_case (len, instring, outstring) : JSB_3 =
;  184          0909  2 BEGIN
;  185          0910  2 LOCAL
;  186          0911  2     indesc : BBLOCK [dsc$c_s_bln],
;  187          0912  2     outdesc : BBLOCK [dsc$c_s_bln],
;  188          0913  2     status;
;  189          0914  2
;  190          0915  2 indesc [dsc$w_length] = .len;
;  191          0916  2 indesc [dsc$a_pointer] = .instring;
;  192          0917  2 outdesc [dsc$a_pointer] = .outstring;
;  193          0918  2 status = make_upper_case (indesc, outdesc, true);
;  194          0919  2 RETURN .status;
;  195          0920  1 END;


                                    52  DD 00000 MOVETO_UPPER_CASE::
                                                 PUSHR   R2                                      ; 0908
                                 5E     10  C2 00002         SUBL2   #16, SP                     ;
                           08    AE     50  B0 00005         MOVW    LEN, INDESC                 ; 0915
                           0C    AE     51  D0 00009         MOVL    INSTRING, INDESC+4          ; 0916
                           04    AE     52  D0 0000D         MOVL    OUTSTRING, OUTDESC+4        ; 0917
                                 51     6E  9E 00011         MOVAB   OUTDESC, R1                 ; 0918
                                 50     08  AE 9E 00014      MOVAB   INDESC, R0                  ;
                                 52         01 D0 00018      MOVL    #1, R2                      ;
                                        FF5E 30 0001B        BSBW    MAKE_UPPER_CASE             ;
                                 5E         10 C0 0001E      ADDL2   #16, SP                     ; 0920
                                            04 BA 00021      POPR    #^M<R2>                     ;
                                               05 00023      RSB                                 ;

; Routine Size:  36 bytes,    Routine Base:  $CODE$ + 0084
```

```
;  197          0921  1 %SBTTL 'GET_MEM';
;  198          0922  1
;  199          0923  1 GLOBAL ROUTINE get_mem (bytes, retadr) : JSB_2 =
;  200          0924  1
;  201          0925  1 !---
;  202          0926  1 !
;  203          0927  1 !       This routine allocates virtual memory for the library access
;  204          0928  1 !       routines.  The memory will not be zeroed.
;  205          0929  1 !
;  206          0930  1 ! Inputs:
;  207          0931  1 !
;  208          0932  1 !       bytes = number of bytes to allocate
;  209          0933  1 !       retadr = address to receieve address of memory
;  210          0934  1 !
;  211          0935  1 ! Routine value:
;  212          0936  1 !
;  213          0937  1 !       lib$_insvirmem  insufficient virtual memory
;  214          0938  1 !       lib$_badblosiz  bad block size
;  215          0939  1 !       true            success
;  216          0940  1 !--
;  217          0941  1
;  218          0942  1 LBR$GET_VM (.bytes, .retadr);          ! Allocate the memory



                     0000G 31 00000 GET_MEM::
                                        BRW     LBR$GET_VM                      ; 0942

; Routine Size:  3 bytes,    Routine Base:  $CODE$ + 00A8
```

```
;  220          0943  1 %SBTTL 'DEALLOC_MEM';
;  221          0944  1
;  222          0945  1 GLOBAL ROUTINE dealloc_mem (bytes, address) : JSB_2 =
;  223          0946  1
;  224          0947  1 !---
;  225          0948  1 !
;  226          0949  1 !       Deallocate dynamic memory.
;  227          0950  1 !
;  228          0951  1 ! Inputs:
;  229          0952  1 !
;  230          0953  1 !       bytes = Number of bytes to deallocate.
;  231          0954  1 !       address = Starting address to deallocate.
;  232          0955  1 !
;  233          0956  1 ! Outputs:
;  234          0957  1 !
;  235          0958  1 !       None
;  236          0959  1 !---
;  237          0960  1
;  238          0961  1 LBR$FREE_VM (.bytes, .address);         ! Deallocate memory



                     0000G 31 00000 DEALLOC_MEM::
                                    BRW     LBR$FREE_VM                     ; 0961

; Routine Size:  3 bytes,    Routine Base:  $CODE$ + 00AB
```

```
;  240          0962  1 %SBTTL 'GET_ZMEM';
;  241          0963  1
;  242          0964  1 GLOBAL ROUTINE get_zmem (bytes, retadr) : JSB_2 =
;  243          0965  1
;  244          0966  1 !---
;  245          0967  1 !
;  246          0968  1 !       This routine obtains dynamic virtual memory
;  247          0969  1 !       and zeros the memory as well before returning.
;  248          0970  1 !
;  249          0971  1 ! Inputs:
;  250          0972  1 !
;  251          0973  1 !       bytes = Number of bytes to obtain
;  252          0974  1 !       retadr = Address to return address of memory
;  253          0975  1 !
;  254          0976  1 ! Outputs:
;  255          0977  1 !
;  256          0978  1 !       The memory is zeroed.
;  257          0979  1 !---
;  258          0980  1
;  259          0981  2 BEGIN
;  260          0982  2
;  261          0983  2 LOCAL
;  262          0984  2     status;
;  263          0985  2
;  264          0986  2 perform (get_mem (.bytes, .retadr));    ! Get the memory
;  265          0987  2
;  266          0988  2 CH$FILL (0,.bytes,..retadr);    ! Zero the memory
;  267          0989  2
;  268          0990  2 RETURN true
;  269          0991  1 END;


                                3C  BB 00000 GET_ZMEM::
                                                PUSHR   #^M<R2,R3,R4,R5>                        ; 0964
                           52       51  D0 00002         MOVL    R1, R2                          ;
                           53       50  D0 00005         MOVL    R0, R3                          ;
                           51       52  D0 00008         MOVL    RETADR, R1                      ; 0986
                           50       53  D0 0000B         MOVL    BYTES, R0                       ;
                                    EA  10 0000E         BSBB    GET_MEM                         ;
                           0A       50  E9 00010         BLBC    STATUS, 1$                      ;
                53         00    6E       00  2C 00013         MOVC5   #0, (SP), #0, BYTES, @0(RETADR) ; 0988
                                       00    82     00018                                         ;
                           50       01  D0 0001A         MOVL    #1, R0                          ; 0990
                                    3C  BA 0001D 1$:     POPR    #^M<R2,R3,R4,R5>                ; 0991
                                        05 0001F         RSB                                     ;

; Routine Size:  32 bytes,    Routine Base:  $CODE$ + 00AE
```

```
;  271          0992  1 %SBTTL 'FIND_BLOCK';
;  272          0993  1
;  273          0994  1 GLOBAL ROUTINE find_block (vbn, blockaddr, entry) : JSB_3 =
;  274          0995  2 BEGIN
;  275          0996  2 !
;  276          0997  2 ! This routine reads the given VBN from the disk and caches it
;  277          0998  2 ! in memory (if it is not already there).
;  278          0999  2 !
;  279          1000  2
;  280          1001  2 LOCAL
;  281          1002  2     cachentry : REF BBLOCK;
;  282          1003  2
;  283          1004  2 IF lookup_cache (.vbn, cachentry)
;  284          1005  2 THEN .blockaddr = .cachentry [cache$l_address]
;  285          1006  3 ELSE BEGIN
;  286          1007  3     perform (read_block (.vbn, .blockaddr));
;  287          1008  3     perform (add_cache (.vbn, cachentry));
;  288          1009  3     cachentry [cache$l_address] = ..blockaddr;
;  289          1010  2     END;
;  290          1011  2 .entry = .cachentry;
;  291          1012  2 RETURN true
;  292          1013  1 END;                                    !Of find_block


                                 18  BB 00000 FIND_BLOCK::
                                             PUSHR   #^M<R3,R4>                      ; 0994
                       5E            04  C2 00002             SUBL2   #4, SP
                       53            51  D0 00005             MOVL    R1, R3
                       54            50  D0 00008             MOVL    R0, R4
                       51            6E  9E 0000B             MOVAB   CACHENTRY, R1          ; 1004
                       50            54  D0 0000E             MOVL    VBN, R0
                                 0000G 30 00011             BSBW    LOOKUP_CACHE
                       09            50  E9 00014             BLBC    R0, 1$
                       50            6E  D0 00017             MOVL    CACHENTRY, R0          ; 1005
                       63        08  A0  D0 0001A             MOVL    8(R0), (BLOCKADDR)
                                     1F  11 0001E             BRB     2$
                       51            53  D0 00020 1$:         MOVL    BLOCKADDR, R1          ; 1007
                       50            54  D0 00023             MOVL    VBN, R0
                                 0000V 30 00026             BSBW    READ_BLOCK
                       19            50  E9 00029             BLBC    STATUS, 3$
                       51            6E  9E 0002C             MOVAB   CACHENTRY, R1          ; 1008
                       50            54  D0 0002F             MOVL    VBN, R0
                                 0000G 30 00032             BSBW    ADD_CACHE
                       0D            50  E9 00035             BLBC    STATUS, 3$
                       50            6E  D0 00038             MOVL    CACHENTRY, R0          ; 1009
                   08  A0            63  D0 0003B             MOVL    (BLOCKADDR), 8(R0)
                       62            6E  D0 0003F 2$:         MOVL    CACHENTRY, (ENTRY)     ; 1011
                       50            01  D0 00042             MOVL    #1, R0                 ; 1012
                       5E            04  C0 00045 3$:         ADDL2   #4, SP                 ; 1013
                                     18  BA 00048             POPR    #^M<R3,R4>
                                         05 0004A             RSB

; Routine Size:  75 bytes,    Routine Base:  $CODE$ + 00CE
```

```
;  294          1014  1 %SBTTL 'VALIDATE_CTL';
;  295          1015  1
;  296          1016  1 GLOBAL ROUTINE validate_ctl (index) : JSB_1 =
;  297          1017  1
;  298          1018  1 !---
;  299          1019  1 !
;  300          1020  1 !       This routine performs most of the initial checking done
;  301          1021  1 !       during argument validation for the processing routines.
;  302          1022  1 !       Its checks a given control table index for validity and
;  303          1023  1 !       also ensures that the library file has already been opened.
;  304          1024  1 !
;  305          1025  1 ! Inputs:
;  306          1026  1 !
;  307          1027  1 !       index = Control table index
;  308          1028  1 !
;  309          1029  1 ! Outputs:
;  310          1030  1 !
;  311          1031  1 !       lbr$_libnotopn - library not open
;  312          1032  1 !       lbr$_illctl - illegal control table index
;  313          1033  1 !---
;  314          1034  1
;  315          1035  2 BEGIN
;  316          1036  2
;  317          1037  2 lbr$gl_rmsstv = 0;                        ! Preset to no error
;  318          1038  2 IF .index GTRU lbr$c_maxctl               ! If greater than maximum,
;  319          1039  2     OR .lbr$al_ctltab [.index-1] EQL 0    ! Or if uninitialized slot,
;  320          1040  2 THEN
;  321          1041  2     RETURN lbr$_illctl;                   ! return with error
;  322          1042  2
;  323          1043  2 lbr$gl_control = .lbr$al_ctltab [.index-1]; ! Set current index
;  324          1044  2
;  325          1045  2 IF NOT .lbr$gl_control [lbr$v_open]      ! If library not open,
;  326          1046  2 THEN
;  327          1047  2     RETURN lbr$_libnotopn;                ! then return with error
;  328          1048  2
;  329          1049  2 RETURN true;
;  330          1050  2
;  331          1051  1 END;


                        0000G  CF   D4 00000 VALIDATE_CTL::
                                             CLRL    LBR$GL_RMSSTV                                  ; 1037
                    10          50   D1 00004         CMPL    INDEX, #16                             ; 1038
                                07   1A 00007         BGTRU   1$
                        0000GCF40   D5 00009         TSTL    LBR$AL_CTLTAB-4[INDEX]                 ; 1039
                                08   12 0000E         BNEQ    2$
                    50 00000000G   8F   D0 00010 1$:  MOVL    #LBR$_ILLCTL, R0                       ; 1041
                                05   00017           RSB
              0000G CF      0000GCF40   D0 00018 2$:  MOVL    LBR$AL_CTLTAB-4[INDEX], LBR$GL_CONTROL ; 1043
                    50      0000G CF    D0 00020      MOVL    LBR$GL_CONTROL, R0                     ; 1045
          08     06   A0              01   E0 00025      BBS     #1, 6(R0), 3$
                    50 00000000G   8F   D0 0002A         MOVL    #LBR$_LIBNOTOPN, R0                    ; 1047
                                05   00031           RSB
                    50          01   D0 00032 3$:     MOVL    #1, R0                                 ; 1049
```

```
                                05 00035        RSB                                     ; 1051

; Routine Size:  54 bytes,    Routine Base:  $CODE$ + 0119
```

```
;  333          1052  1 %SBTTL 'ALLOC_BLOCK';
;  334          1053  1
;  335          1054  1 GLOBAL ROUTINE alloc_block (vbn, address) : JSB_2 =
;  336          1055  1
;  337          1056  1 !---
;  338          1057  1 !
;  339          1058  1 !       Allocate an unused disk block in the file and return
;  340          1059  1 !       the address of the block in memory.
;  341          1060  1 !
;  342          1061  1 ! Inputs:
;  343          1062  1 !
;  344          1063  1 !       vbn = Longword to receive VBN of allocated block
;  345          1064  1 !       address = Longword to receive address of block in memory
;  346          1065  1 !
;  347          1066  1 ! Outputs:
;  348          1067  1 !
;  349          1068  1 !       vbn = VBN of newly allocated block
;  350          1069  1 !       address = Address of block in memory
;  351          1070  1 !---
;  352          1071  1
;  353          1072  2 BEGIN
;  354          1073  2
;  355          1074  2 BIND
;  356          1075  2     context = .lbr$gl_control [lbr$l_ctxptr]: BBLOCK,   ! Context block
;  357          1076  2     header = .lbr$gl_control [lbr$l_hdrptr]: BBLOCK;    ! Header block
;  358          1077  2
;  359          1078  2 LOCAL
;  360          1079  2     cachentry : REF BBLOCK;
;  361          1080  2
;  362          1081  2 IF .header [lhd$l_freevbn] NEQ 0        ! If logically deleted space,
;  363          1082  2 THEN
;  364          1083  3     BEGIN
;  365        P 1084  3     perform (find_block(                ! Read 1st free block
;  366          1085  3                 .header [lhd$l_freevbn], .address, cachentry));
;  367          1086  3     .vbn = .header [lhd$l_freevbn];     ! Return VBN of block
;  368          1087  3     header [lhd$l_freevbn] = ...address;! Dequeue 1st block from chain
;  369          1088  3     header [lhd$l_freeblk] = .header [lhd$l_freeblk] -1;
;  370          1089  3     perform (remove_cache (..vbn));     ! Remove entry from cache so caller may
;  371          1090  3                                         !  reinsert if desired
;  372          1091  3     END
;  373          1092  2 ELSE
;  374          1093  3     BEGIN
;  375          1094  3     BIND
;  376          1095  3         next_vbn = header [lhd$l_nextvbn];      ! End of library
;  377          1096  3
;  378          1097  3     perform (get_zmem (lbr$c_pagesize,.address));       ! Allocate space for block
;  379          1098  3     .vbn = .next_vbn;                               ! Return VBN of block
;  380          1099  3     next_vbn = .next_vbn + 1;                       !next vbn is next block
;  381          1100  2     END;
;  382          1101  2
;  383          1102  2 context [ctx$v_hdrdirty] = true;        ! Mark header modified
;  384          1103  2
;  385          1104  2 RETURN true;
;  386          1105  1 END;
```

```
                 00FC   8F   BB 00000 ALLOC_BLOCK::
                                      PUSHR   #^M<R2,R3,R4,R5,R6,R7>                  ; 1054
                     5E        04   C2 00004          SUBL2   #4, SP                  ;
                     55        51   D0 00007          MOVL    R1, R5                  ;
                     57        50   D0 0000A          MOVL    R0, R7                  ;
                     50   0000G  CF   D0 0000D          MOVL    LBR$GL_CONTROL, R0      ; 1075
                     56     0E  A0   D0 00012          MOVL    14(R0), R6              ;
                     53     0A  A0   D0 00016          MOVL    10(R0), R3              ; 1076
                     54     44  A3   D0 0001A          MOVL    68(R3), R4              ; 1081
                                 22   13 0001E          BEQL    1$                      ;
                     52        6E   9E 00020          MOVAB   CACHENTRY, R2           ; 1085
                     50        54   7D 00023          MOVQ    R4, R0                  ;
                            FF56   30 00026          BSBW    FIND_BLOCK              ;
                     32        50   E9 00029          BLBC    STATUS, 3$              ;
                     67        54   D0 0002C          MOVL    R4, (VBN)               ; 1086
                 44  A3     00   B5   D0 0002F          MOVL    @0(ADDRESS), 68(R3)     ; 1087
                           48  A3   D7 00034          DECL    72(R3)                  ; 1088
                     50        67   D0 00037          MOVL    (VBN), R0               ; 1089
                            0000G   30 0003A          BSBW    REMOVE_CACHE            ;
                     17        50   E8 0003D          BLBS    STATUS, 2$              ;
                               1C   11 00040          BRB     3$                      ;
                     51        55   D0 00042 1$:      MOVL    ADDRESS, R1             ; 1097
                     50   0200   8F   3C 00045          MOVZWL  #512, R0                ;
                            FF12   30 0004A          BSBW    GET_ZMEM                ;
                     0E        50   E9 0004D          BLBC    STATUS, 3$              ;
                     67        52   A3   D0 00050          MOVL    82(R3), (VBN)           ; 1098
                           52  A3   D6 00054          INCL    82(R3)                  ; 1099
                 04  A6           08   88 00057 2$:      BISB2   #8, 4(R6)               ; 1102
                     50        01   D0 0005B          MOVL    #1, R0                  ; 1104
                     5E        04   C0 0005E 3$:      ADDL2   #4, SP                  ; 1105
                 00FC   8F   BA 00061          POPR    #^M<R2,R3,R4,R5,R6,R7>          ;
                            05 00065          RSB                                     ;

; Routine Size:  102 bytes,    Routine Base:  $CODE$ + 014F
```

```
;  388          1106  1 %SBTTL 'DEALLOC_BLOCK';
;  389          1107  1
;  390          1108  1 GLOBAL ROUTINE dealloc_block (vbn) : JSB_1 =
;  391          1109  1
;  392          1110  1 !---
;  393          1111  1 !
;  394          1112  1 !       Deallocate a disk block in the file and add the block
;  395          1113  1 !       to the logically deleted disk block list.
;  396          1114  1 !
;  397          1115  1 ! Inputs:
;  398          1116  1 !
;  399          1117  1 !       vbn = Longword to receive VBN of allocated block
;  400          1118  1 !
;  401          1119  1 ! Outputs:
;  402          1120  1 !
;  403          1121  1 !       None
;  404          1122  1 !---
;  405          1123  1
;  406          1124  2 BEGIN
;  407          1125  2
;  408          1126  2 BIND
;  409          1127  2     context = .lbr$gl_control [lbr$l_ctxptr]: BBLOCK,    ! Context block
;  410          1128  2     header = .lbr$gl_control [lbr$l_hdrptr]: BBLOCK;     ! Header block
;  411          1129  2
;  412          1130  2 LOCAL
;  413          1131  2     blockaddr : REF VECTOR [,LONG],
;  414          1132  2     cachentry : REF BBLOCK,
;  415          1133  2     thisblock : REF VECTOR [,LONG],
;  416          1134  2     thiscache : REF BBLOCK,
;  417          1135  2     prevblock : REF VECTOR [,LONG],
;  418          1136  2     prevcache : REF BBLOCK,
;  419          1137  2     thisvbn,
;  420          1138  2     prevbn;
;  421          1139  2
;  422          1140  2 perform (find_block (.vbn, blockaddr, cachentry)); !Find block in memory
;  423          1141  2 thisvbn = .header [lhd$l_freevbn];       ! Get free block listhead
;  424          1142  2 perform (find_block (.thisvbn, thisblock, thiscache)); !Find first block
;  425          1143  2 prevbn = 0;
;  426          1144  2 prevblock = 0;
;  427          1145  2 prevcache = 0;
;  428          1146  2 !
;  429          1147  2 ! Loop through the free blocks looking for the place to insert
;  430          1148  2 ! this block.
;  431          1149  2 !
;  432          1150  2 WHILE .vbn GTRU .thisvbn
;  433          1151  2     AND .thisvbn NEQ 0
;  434          1152  3 DO BEGIN
;  435          1153  3     perform (find_block (.thisvbn, thisblock, thiscache)); ! Locate the block
;  436          1154  3     prevbn = .thisvbn;                      ! Remember previous block
;  437          1155  3     thisvbn = .thisblock [0];               ! Link to next
;  438          1156  3     prevblock = .thisblock;                 ! Remember the location of it
;  439          1157  3     prevcache = .thiscache;
;  440          1158  2     END;
;  441          1159  2 IF .prevbn NEQ 0                           ! If it goes in the list somewhere
;  442          1160  3 THEN BEGIN
;  443          1161  3     blockaddr [0] = .prevblock [0];         ! Copy forward link of previous block
;  444          1162  3     prevblock [0] = .vbn;                   ! And insert into the list
```

```
3       cachentry [cache$v_dirty] = true;   ! Mark blocks as dirty
3       cachentry [cache$v_data] = false;   ! No longer a data block
3       prevcache [cache$v_dirty] = true;
3       END
3   ELSE BEGIN
3       blockaddr [0] = .header [lhd$l_freevbn]; ! Insert into free chain
3       header  [lhd$l_freevbn] = .vbn;      ! Insert into free chain
3       cachentry [cache$v_dirty] = true;   ! Mark the block as dirty
3       cachentry [cache$v_data] = false;   ! No longer a data block
2       END;
2
2   header [lhd$l_freeblk] = .header [lhd$l_freeblk] +1; ! Count another free block
2
2   context [ctx$v_hdrdirty] = true;        ! Mark header modified
2
2   RETURN true;
1   END;
```

```
                 03FC   8F  BB 00000 DEALLOC_BLOCK::
                                               PUSHR   #^M<R2,R3,R4,R5,R6,R7,R8,R9>                 ; 1108
            5E          10  C2 00004           SUBL2   #16, SP                                      ;
            59          50  D0 00007           MOVL    R0, R9                                       ;
            50   0000G  CF  D0 0000A           MOVL    LBR$GL_CONTROL, R0                           ; 1127
            58      0E  A0  D0 0000F           MOVL    14(R0), R8                                   ;
            53      0A  A0  D0 00013           MOVL    10(R0), R3                                   ; 1128
            52          6E  9E 00017           MOVAB   CACHENTRY, R2                                ; 1140
            51      04  AE  9E 0001A           MOVAB   BLOCKADDR, R1                                ;
            50          59  D0 0001E           MOVL    VBN, R0                                      ;
                      FEF5  30 00021           BSBW    FIND_BLOCK                                   ;
            77          50  E9 00024           BLBC    STATUS, 5$                                   ;
            54      44  A3  D0 00027           MOVL    68(R3), THISVBN                              ; 1141
            52      08  AE  9E 0002B           MOVAB   THISCACHE, R2                                ; 1142
            51      0C  AE  9E 0002F           MOVAB   THISBLOCK, R1                                ;
            50          54  D0 00033           MOVL    THISVBN, R0                                  ;
                      FEE0  30 00036           BSBW    FIND_BLOCK                                   ;
            62          50  E9 00039           BLBC    STATUS, 5$                                   ;
                        56  7C 0003C           CLRQ    PREVBLOCK                                    ; 1144
                        55  D4 0003E           CLRL    PREVCACHE                                    ; 1145
            54          59  D1 00040 1$:       CMPL    VBN, THISVBN                                 ; 1150
                        22  1B 00043           BLEQU   2$                                           ;
                        54  D5 00045           TSTL    THISVBN                                      ; 1151
                        1E  13 00047           BEQL    2$                                           ;
            52      08  AE  9E 00049           MOVAB   THISCACHE, R2                                ; 1153
            51      0C  AE  9E 0004D           MOVAB   THISBLOCK, R1                                ;
            50          54  D0 00051           MOVL    THISVBN, R0                                  ;
                      FEC2  30 00054           BSBW    FIND_BLOCK                                   ;
            44          50  E9 00057           BLBC    STATUS, 5$                                   ;
            57          54  D0 0005A           MOVL    THISVBN, PREVBN                              ; 1154
            54      0C  BE  D0 0005D           MOVL    @THISBLOCK, THISVBN                          ; 1155
            55      08  AE  7D 00061           MOVQ    THISCACHE, PREVCACHE                         ; 1157
                        D9  11 00065           BRB     1$                                           ; 1150
            50          6E  D0 00067 2$:       MOVL    CACHENTRY, R0                                ; 1163
                        57  D5 0006A           TSTL    PREVBN                                       ; 1159
```

```
                               15  13 0006C        BEQL    3$                                      ;
       04    BE                66  D0 0006E        MOVL    (PREVBLOCK), @BLOCKADDR                 ; 1161
             66                59  D0 00072        MOVL    VBN, (PREVBLOCK)                        ; 1162
       0C    A0                01  88 00075        BISB2   #1, 12(R0)                              ; 1163
       0C    A0                02  8A 00079        BICB2   #2, 12(R0)                              ; 1164
       0C    A5                01  88 0007D        BISB2   #1, 12(PREVCACHE)                       ; 1165
                               11  11 00081        BRB     4$                                      ; 1159
       04    BE         44     A3  D0 00083 3$:    MOVL    68(R3), @BLOCKADDR                      ; 1168
       44    A3                59  D0 00088        MOVL    VBN, 68(R3)                             ; 1169
       0C    A0                01  88 0008C        BISB2   #1, 12(R0)                              ; 1170
       0C    A0                02  8A 00090        BICB2   #2, 12(R0)                              ; 1171
                        48     A3  D6 00094 4$:    INCL    72(R3)                                  ; 1174
       04    A8                08  88 00097        BISB2   #8, 4(R8)                               ; 1176
             50                01  D0 0009B        MOVL    #1, R0                                  ; 1178
             5E                10  C0 0009E 5$:    ADDL2   #16, SP                                 ; 1179
                      03FC     8F  BA 000A1        POPR    #^M<R2,R3,R4,R5,R6,R7,R8,R9>            ;
                                   05 000A5        RSB                                             ;

; Routine Size:  166 bytes,    Routine Base:  $CODE$ + 01B5
```

```
1 %SBTTL 'WRITE_BLOCK';
1
1 GLOBAL ROUTINE write_block (addr, vbn) : JSB_2 =
1
1 !---
1 !
1 !       This routine writes a block in memory to the library
1 !       file at the specified block number.
1 !
1 ! Inputs:
1 !
1 !       addr = Address of block in memory
1 !       vbn = disk block number
1 !
1 ! Outputs:
1 !
1 !       None
1 !---
1
2 BEGIN
2
2 BIND
2     context = .lbr$gl_control [lbr$l_ctxptr]: BBLOCK,
2     rab   = .context [ctx$l_recrab]: BBLOCK;
2
2 LOCAL
2     status;
2
2 rab [rab$l_bkt] = .vbn;          ! Set block number to write
2 rab [rab$l_rbf] = .addr;         !Set address of buffer for write
2 rab [rab$w_rsz] = lbr$c_pagesize;        !And its length
2 status = $WRITE (RAB=rab);       !Write the record
2 IF NOT .status
2 THEN lbr$gl_rmsstv = .rab [rab$l_stv];
2 RETURN .status;
2
1 END;
```

```
                                                    .EXTRN  SYS$WRITE

                                 52  DD 00000 WRITE_BLOCK::
                                                    PUSHL   R2                              ; 1182
                  52      0000G  CF  D0 00002       MOVL    LBR$GL_CONTROL, R2              ; 1202
                  52         0E  A2  D0 00007       MOVL    14(R2), R2                      ;
                  52         0C  A2  D0 0000B       MOVL    12(R2), R2                      ; 1203
             38   A2             51  D0 0000F       MOVL    VBN, 56(R2)                     ; 1208
             28   A2             50  D0 00013       MOVL    ADDR, 40(R2)                    ; 1209
             22   A2       0200  8F  B0 00017       MOVW    #512, 34(R2)                    ; 1210
                                 52  DD 0001D       PUSHL   R2                              ; 1211
      00000000G   00             01  FB 0001F       CALLS   #1, SYS$WRITE                   ;
                  06             50  E8 00026       BLBS    STATUS, 1$                      ; 1212
          0000G   CF         0C  A2  D0 00029       MOVL    12(R2), LBR$GL_RMSSTV           ; 1213
                                 04  BA 0002F 1$:   POPR    #^M<R2>                         ; 1216
                                     05 00031       RSB                                     ;
```

; Routine Size: 50 bytes,   Routine Base: $CODE$ + 025B

```
1 %SBTTL 'READ_BLOCK';
1
1 GLOBAL ROUTINE read_block (vbn, addr) : JSB_2 =
1
1 !---
1 !
1 !       This routine reads a specified block and returns
1 !       the address of the block in memory.
1 !
1 ! Inputs:
1 !
1 !       vbn = block number in file
1 !       addr = Longword to receive address of block
1 !
1 ! Outputs:
1 !
1 !       addr = Address of block read from disk
1 !---
1
2 BEGIN
2
2 BIND
2     context = .lbr$gl_control [lbr$l_ctxptr]: BBLOCK,
2     rab   = .context [ctx$l_recrab]: BBLOCK;
2 LOCAL
2     status;
2
2 perform (get_mem (lbr$c_pagesize,rab [rab$l_ubf]));      ! Allocate storage for block
2 rab [rab$w_usz] = lbr$c_pagesize;                ! Set size of buffer
2 rab [rab$l_bkt] = .vbn;            ! Set block number to read
2 status = $READ (RAB=rab);          ! Read the block into memory
2 IF NOT .status
2 THEN lbr$gl_rmsstv = .rab [rab$l_stv];
2 .addr = .rab [rab$l_rbf];                  ! Return address
2
2 RETURN .status;
2
1 END;


                                                  .EXTRN  SYS$READ

                           1C  BB 00000 READ_BLOCK::
                                                  PUSHR   #^M<R2,R3,R4>                   ; 1219
              53           50  7D 00002           MOVQ    R0, R3                          ;
              50    0000G  CF  D0 00005           MOVL    LBR$GL_CONTROL, R0              ; 1239
              50       0E  A0  D0 0000A           MOVL    14(R0), R0                      ;
              52       0C  A0  D0 0000E           MOVL    12(R0), R2                      ; 1240
              51       24  A2  9E 00012           MOVAB   36(R2), R1                      ; 1244
              50     0200  8F  3C 00016           MOVZWL  #512, R0                        ;
                         FDFD  30 0001B           BSBW    GET_MEM                         ;
              20           50  E9 0001E           BLBC    STATUS, 2$                      ;
         20   A2     0200  8F  B0 00021           MOVW    #512, 32(R2)                    ; 1245
         38   A2           53  D0 00027           MOVL    VBN, 56(R2)                     ; 1246
                           52  DD 0002B           PUSHL   R2                              ; 1247
  00000000G   00           01  FB 0002D           CALLS   #1, SYS$READ                    ;
```

```
                         06          50  E8 00034          BLBS    STATUS, 1$                              ; 1248
                  0000G  CF      0C   A2  D0 00037          MOVL    12(R2), LBR$GL_RMSSTV                   ; 1249
                         64      28   A2  D0 0003D 1$:      MOVL    40(R2), (ADDR)                          ; 1250
                                      1C  BA 00041 2$:      POPR    #^M<R2,R3,R4>                           ; 1254
                                          05 00043          RSB                                             ;

; Routine Size:  68 bytes,    Routine Base:  $CODE$ + 028D
```

```
;  540          1255  1 %SBTTL 'READ_N_BLOCK';
;  541          1256  1
;  542          1257  1 GLOBAL ROUTINE read_n_block (start_block, num_blocks) : JSB_2 =
;  543          1258  2 BEGIN
;  544          1259  2 !
;  545          1260  2 ! This routine reads the given number of blocks, starting with start_block,
;  546          1261  2 ! and caches them in the disk block cache.
;  547          1262  2 !
;  548          1263  2 BIND
;  549          1264  2     context = .lbr$gl_control [lbr$l_ctxptr] : BBLOCK,
;  550          1265  2     librab = .context [ctx$l_recrab] : BBLOCK;
;  551          1266  2
;  552          1267  2 LOCAL
;  553          1268  2     nblocks,
;  554          1269  2     rblocks,
;  555          1270  2     blkaddr,
;  556          1271  2     cachentry : REF BBLOCK,
;  557          1272  2     all_blocks,
;  558          1273  2     rmsstv,
;  559          1274  2     status;
;  560          1275  2 !
;  561          1276  2 ! Allocate the memory
;  562          1277  2 !
;  563          1278  2 IF .num_blocks EQL 0
;  564          1279  2     THEN RETURN true;
;  565          1280  2 all_blocks = true;
;  566          1281  2 nblocks = MINU (.lbr$gl_maxread, .num_blocks);
;  567          1282  2 IF .nblocks LSS .num_blocks
;  568          1283  2     THEN all_blocks = false;
;  569          1284  2 INCRU i FROM 0 TO .nblocks - 1
;  570          1285  2 DO IF lookup_cache (.start_block+.i, cachentry)
;  571          1286  3     THEN BEGIN
;  572          1287  3         nblocks = .i;
;  573          1288  3         EXITLOOP;
;  574          1289  2         END;
;  575          1290  2 IF .nblocks LEQ 0
;  576          1291  2     THEN RETURN true;
;  577          1292  3 DO  (status = get_mem (.nblocks*lbr$c_pagesize, blkaddr))
;  578          1293  2 UNTIL .status
;  579          1294  2     OR ((nblocks = .nblocks - 1) LEQ 0);
;  580          1295  2 !
;  581          1296  2 ! Memory is allocated, read the blocks
;  582          1297  2 !
;  583          1298  2 IF .nblocks GTR 0
;  584          1299  3 THEN BEGIN
;  585          1300  3     librab [rab$l_bkt] = .start_block;                  !Set starting vbn
;  586          1301  3     librab [rab$l_ubf] = .blkaddr;                      !and buffer address
;  587          1302  3     librab [rab$w_usz] = .nblocks*lbr$c_pagesize;       !and byte count
;  588          1303  3     status = $READ (RAB=librab);                        !Read the blocks
;  589          1304  3     rmsstv = .librab[rab$l_stv];                        !Save the STV
;  590          1305  3     rblocks = .librab [rab$w_rsz]/lbr$c_pagesize;       !Compute number of blocks read
;  591          1306  3     IF .rblocks LSSU .nblocks                           ! and if we did not fill the buffer
;  592          1307  3     THEN dealloc_mem ((.nblocks-.rblocks)*lbr$c_pagesize, !Then deallocate the memory
;  593          1308  3                       .blkaddr+ (.rblocks*lbr$c_pagesize));
;  594          1309  3     IF .rblocks LEQ 0                                   ! No blocks read
;  595          1310  4     THEN BEGIN
;  596          1311  4         IF NOT .status
```

```
4                THEN lbr$gl_rmsstv = .rmsstv;
4            RETURN .status;
3            END;
3
3        INCRU i FROM 0 TO .rblocks-1                        !Loop over all read
4        DO BEGIN
4            IF NOT add_cache (.start_block+.i, cachentry)   !Try to add to cache
4            THEN dealloc_mem (lbr$c_pagesize, .blkaddr)
5            ELSE BEGIN
5                cachentry [cache$l_address] = .blkaddr;
4                END;
4            blkaddr = .blkaddr + lbr$c_pagesize;
3            END;
3        IF .nblocks LSSU .num_blocks                        !If we did not read all requested
3            AND .status NEQ rms$_eof                        ! and we did not read to end of file
3            AND NOT .all_blocks
3            THEN read_n_block (.start_block+.nblocks, .num_blocks-.nblocks);
3        IF .status EQL rms$_eof
3            THEN RETURN true
4            ELSE BEGIN
4                IF NOT .status
4                    THEN lbr$gl_rmsstv = .rmsstv;
4                RETURN .status;
3                END;
2        END;
2    RETURN .status                                          !Could not allocate memory
1    END;                           !Of read_n_block



              03FC   8F  BB 00000 READ_N_BLOCK::
                                  PUSHR   #^M<R2,R3,R4,R5,R6,R7,R8,R9>            ; 1257
         5E          08  C2 00004         SUBL2   #8, SP                                  ;
         58          51  D0 00007         MOVL    R1, R8                                  ;
         55          50  D0 0000A         MOVL    R0, R5                                  ;
         50   0000G  CF  D0 0000D         MOVL    LBR$GL_CONTROL, R0                      ; 1264
         50      0E  A0  D0 00012         MOVL    14(R0), R0                              ;
         52      0C  A0  D0 00016         MOVL    12(R0), R2                              ; 1265
                     58  D5 0001A         TSTL    NUM_BLOCKS                              ; 1278
                     40  13 0001C         BEQL    7$                                      ;
         59          01  D0 0001E         MOVL    #1, ALL_BLOCKS                          ; 1280
         50   0000G  CF  D0 00021         MOVL    LBR$GL_MAXREAD, R0                      ; 1281
         58          50  D1 00026         CMPL    R0, NUM_BLOCKS                          ;
                     03  1B 00029         BLEQU   1$                                      ;
         50          58  D0 0002B         MOVL    NUM_BLOCKS, R0                          ;
         53          50  D0 0002E 1$:     MOVL    R0, NBLOCKS                             ;
         58          53  D1 00031         CMPL    NBLOCKS, NUM_BLOCKS                     ; 1282
                     02  18 00034         BGEQ    2$                                      ;
                     59  D4 00036         CLRL    ALL_BLOCKS                              ; 1283
         56      FF  A3  9E 00038 2$:     MOVAB   -1(R3), R6                              ; 1284
                     54  D4 0003C         CLRL    I                                       ; 1285
                     15  11 0003E         BRB     5$                                      ;
         51      04  AE  9E 00040 3$:     MOVAB   CACHENTRY, R1                           ;
 50      55          54  C1 00044         ADDL3   I, START_BLOCK, R0                      ;
                  0000G  30 00048         BSBW    LOOKUP_CACHE                            ;
```

```
                                05              50  E9 0004B          BLBC    R0, 4$                                          ;
                                53              54  D0 0004E          MOVL    I, NBLOCKS                                      ; 1287
                                                07  11 00051          BRB     6$                                              ; 1286
                                                54  D6 00053 4$:      INCL    I                                               ; 1285
                                56              54  D1 00055 5$:      CMPL    I, R6                                           ;
                                                E6  1B 00058          BLEQU   3$                                              ;
                                                53  D5 0005A 6$:      TSTL    NBLOCKS                                         ; 1290
                                                03  14 0005C          BGTR    8$                                              ;
                                             00BE   31 0005E 7$:      BRW     18$                                             ;
                                51              6E  9E 00061 8$:      MOVAB   BLKADDR, R1                                     ; 1292
                        50                      53  09 78 00064          ASHL    #9, NBLOCKS, R0                                 ;
                                              FD6C  30 00068          BSBW    GET_MEM                                         ;
                                54              50  D0 0006B          MOVL    R0, STATUS                                      ;
                                03              54  E8 0006E          BLBS    STATUS, 9$                                      ; 1293
                                ED              53  F5 00071          SOBGTR  NBLOCKS, 8$                                     ; 1294
                                                53  D5 00074 9$:      TSTL    NBLOCKS                                         ; 1298
                                                49  15 00076          BLEQ    11$                                             ;
                             38 A2              55  D0 00078          MOVL    START_BLOCK, 56(R2)                             ; 1300
                             24 A2              6E  D0 0007C          MOVL    BLKADDR, 36(R2)                                 ; 1301
                        20 A2                   53      0200  8F A5 00080  MULW3   #512, NBLOCKS, 32(R2)                      ; 1302
                                                52  DD 00087          PUSHL   R2                                              ; 1303
                            00000000G           00  01 FB 00089          CALLS   #1, SYS$READ                                    ;
                                                54  50 D0 00090          MOVL    R0, STATUS                                      ;
                                56          0C  A2  D0 00093          MOVL    12(R2), RMSSTV                                  ; 1304
                                52          22  A2  3C 00097          MOVZWL  34(R2), RBLOCKS                                 ; 1305
                                52  00000200   8F  C6 0009B          DIVL2   #512, RBLOCKS                                   ;
                                53              52  D1 000A2          CMPL    RBLOCKS, NBLOCKS                                ; 1306
                                                13  1E 000A5          BGEQU   10$                                             ;
                        50                      52  09 78 000A7          ASHL    #9, RBLOCKS, R0                                 ; 1308
                        51                      50  6E C1 000AB          ADDL3   BLKADDR, R0, R1                                 ;
                        50                      53  52 C3 000AF          SUBL3   RBLOCKS, NBLOCKS, R0                            ; 1307
                        50                      50  09 78 000B3          ASHL    #9, R0, R0                                      ;
                                                  FD20  30 000B7          BSBW    DEALLOC_MEM                                     ;
                                                52  D5 000BA 10$:      TSTL    RBLOCKS                                         ; 1309
                                                05  14 000BC          BGTR    12$                                             ;
                                66              54  E9 000BE          BLBC    STATUS, 20$                                     ; 1311
                                                69  11 000C1 11$:      BRB     21$                                             ; 1313
                                                52  D7 000C3 12$:      DECL    R2                                              ; 1316
                                                57  D4 000C5          CLRL    I                                               ;
                                                2C  11 000C7          BRB     16$                                             ;
                                51          04  AE  9E 000C9 13$:      MOVAB   CACHENTRY, R1                                   ; 1318
                        50                      57  C1 000CD          ADDL3   I, START_BLOCK, R0                              ;
                                              0000G 30 000D1          BSBW    ADD_CACHE                                       ;
                                0D              50  E8 000D4          BLBS    R0, 14$                                         ;
                                51              6E  D0 000D7          MOVL    BLKADDR, R1                                     ; 1319
                                50      0200    8F  3C 000DA          MOVZWL  #512, R0                                        ;
                                              FCF8  30 000DF          BSBW    DEALLOC_MEM                                     ;
                                                08  11 000E2          BRB     15$                                             ;
                                50          04  AE  D0 000E4 14$:      MOVL    CACHENTRY, R0                                   ; 1321
                             08 A0              6E  D0 000E8          MOVL    BLKADDR, 8(R0)                                  ;
                                6E  00000200    8F  C0 000EC 15$:      ADDL2   #512, BLKADDR                                   ; 1323
                                                57  D6 000F3          INCL    I                                               ; 1316
                                52              57  D1 000F5 16$:      CMPL    I, R2                                           ;
                                                CF  1B 000F8          BLEQU   13$                                             ;
                                58              53  D1 000FA          CMPL    NBLOCKS, NUM_BLOCKS                             ; 1325
                                                17  1E 000FD          BGEQU   17$                                             ;
                         0001827A   8F          54  D1 000FF          CMPL    STATUS, #98938                                  ; 1326
```

```
                                    0E  13 00106          BEQL    17$                                                     ;
                                 0B     59  E8 00108          BLBS    ALL_BLOCKS, 17$                                         ; 1327
                              51        58  C3 0010B          SUBL3   NBLOCKS, NUM_BLOCKS, R1                                 ; 1328
                              50        55  C1 0010F          ADDL3   NBLOCKS, START_BLOCK, R0                                ;
                                    FEEA  30 00113          BSBW    READ_N_BLOCK                                            ;
                          0001827A  8F    54  D1 00116 17$:    CMPL    STATUS, #98938                                          ; 1329
                                    05  12 0011D          BNEQ    19$                                                     ;
                                 50     01  D0 0011F 18$:    MOVL    #1, R0                                                  ; 1331
                                    0B  11 00122          BRB     22$                                                     ;
                                 05     54  E8 00124 19$:    BLBS    STATUS, 21$                                             ; 1332
                             0000G  CF   56  D0 00127 20$:    MOVL    RMSSTV, LBR$GL_RMSSTV                                   ; 1333
                                 50     54  D0 0012C 21$:    MOVL    STATUS, R0                                              ; 1337
                                 5E     08  C0 0012F 22$:    ADDL2   #8, SP                                                  ; 1338
                                    03FC  8F  BA 00132          POPR    #^M<R2,R3,R4,R5,R6,R7,R8,R9>                            ;
                                          05  00136          RSB                                                             ;

; Routine Size:  311 bytes,    Routine Base:  $CODE$ + 02D1

;  624          1339  1
```

```
;  626      1340  1 %SBTTL 'INCR_RFA';
;  627      1341  1
;  628      1342  1 GLOBAL ROUTINE incr_rfa (incrmnt, rfadr) : JSB_2 NOVALUE =
;  629      1343  2 BEGIN
;  630      1344  2 !
;  631      1345  2 ! This routine updates the specified rfa by incrementing by
;  632      1346  2 ! the given number of bytes.
;  633      1347  2 !
;  634      1348  2 !
;  635      1349  2 MAP
;  636      1350  2     rfadr : REF BBLOCK;
;  637      1351  2
;  638      1352  2 LOCAL
;  639      1353  2     offset;
;  640      1354  2
;  641      1355  2 offset = (.rfadr [rfa$w_offset] + .incrmnt + 1) AND NOT 1; !Compute total # bytes
;  642      1356  2 rfadr [rfa$w_offset] = .offset AND (data$c_length - 1); !Compute new offset
;  643      1357  2 rfadr [rfa$l_vbn] = .rfadr [rfa$l_vbn] +
;  644      1358  2                 (.offset/data$c_length);
;  645      1359  2 RETURN
;  646      1360  1 END;                                   ! Of incr_rfa



                                         52  DD 00000 INCR_RFA::
                                                            PUSHL   R2                              ; 1342
                             52        04  A1  3C 00002     MOVZWL  4(RFADR), R2                    ; 1355
                             50     01 A042  9E 00006       MOVAB   1(INCRMNT)[R2], R0              ;
                             50            01  8A 0000B     BICB2   #1, OFFSET                      ;
       52          50        09            00  EF 0000E     EXTZV   #0, #9, OFFSET, R2              ; 1356
                       04    A1            52  B0 00013     MOVW    R2, 4(RFADR)                    ;
                             50 00000200   8F  C6 00017     DIVL2   #512, R0                        ; 1358
                             61            50  C0 0001E     ADDL2   R0, (RFADR)                     ;
                                           04  BA 00021     POPR    #^M<R2>                         ; 1360
                                               05 00023     RSB                                     ;

; Routine Size:  36 bytes,    Routine Base:  $CODE$ + 0408
```

_$2
Sym
CLI
CLI
CLI
CLI
CLI
CLI
CLI
CLI
CLI
CLI
CLI
CLI
CLI
CLI
COM
COM
CRE
CRE
CRF
CRF
CRF
DAT
DAT
DCL
DCX
FIN
GET
GET
LBR
LBR
LBR
LBR
LBR
LBR
LBR
LBR
LBR
LBR
LBR
LBR
LBR
LBR
LBR

```
;  648          1361  1 END                                  . Of module
;  649          1362  0 ELUDOM
```

```
;                                  PSECT SUMMARY

;       Name                    Bytes                           Attributes

;  $CODE$                         1068  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL   CON,NOPIC,ALIGN(2)
```

```
;                               Library Statistics

;                                        -------- Symbols --------      Pages       Processing
;       File                             Total    Loaded   Percent      Mapped      Time

;  _$255$DUA28:[SYSLIB]STARLET.L32;1      9776        25         0       581          00:01.1
```

```
;                                   COMMAND QUALIFIERS

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

; Size:          1068 code + 0 data bytes
; Run Time:         00:26.8
; Elapsed Time:     01:07.6
; Lines/CPU Min:     3049
; Lexemes/CPU-Min: 25688
; Memory Used:  172 pages
; Compilation Complete
```

TRANSFER
LIS

DATABASE
LIS

PUTCACHE
LIS

LIBRAR

PREFIX
REQ

LIBRARIAN
MAP

CROSS
LIS

SUBS
LIS

FILEIO
LIS

PADLBR
LIS

COMPRESS
LIS

EXTRACT
LIS

LIB
MDL

DELETE
LIS