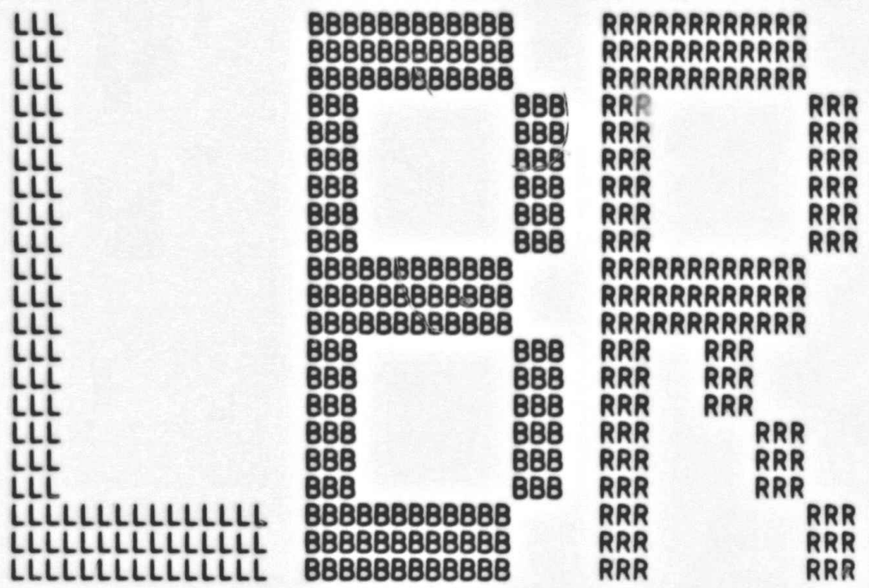

_$2

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
00F
00F
00F
00F
00F
00F
00F
00F
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF

FILEID**GETPUT

```
  GGGGGGGG  EEEEEEEEEE  TTTTTTTTTT  PPPPPPPP    UU      UU  TTTTTTTTTT
  GGGGGGGG  EEEEEEEEEE  TTTTTTTTTT  PPPPPPPPP   UU      UU  TTTTTTTTTT
GG          EE              TT      PP      PP  UU      UU      TT
GG          EE              TT      PP      PP  UU      UU      TT
GG          EE              TT      PP      PP  UU      UU      TT
GG          EE              TT      PP      PP  UU      UU      TT
GG          EEEEEEEE        TT      PPPPPPPPP   UU      UU      TT
GG          EEEEEEEE        TT      PPPPPPPP    UU      UU      TT
GG  GGGGGG  EE              TT      PP          UU      UU      TT
GG  GGGGGG  EE              TT      PP          UU      UU      TT
GG      GG  EE              TT      PP          UU      UU      TT          ....
GG      GG  EE              TT      PP          UU      UU      TT          ....
  GGGGGG    EEEEEEEEEE      TT      PP          UUUUUUUUUU      TT          ....
  GGGGGG    EEEEEEEEEE      TT      PP          UUUUUUUUUU      TT          ....

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_GETPUT (                        ! GET and PUT routines for librarian
;     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 ! AUTHOR:  Benn Schreiber
;    47          0047  1 !
;    48          0048  1 ! CREATION DATE: June, 1979
;    49          0049  1 !
;    50          0050  1 ! MODIFIED BY:
;    51          0051  1 !
;    52          0052  1 !       V03-006 GJA0094         Greg Awdziewicz         7-Aug-1984
;    53          0053  1 !               - Make the buffers for DCX reduced records larger so that
;    54          0054  1 !               records already near the maximum record size can still be
;    55          0055  1 !               "reduced" even if they actually get larger because of
;    56          0056  1 !               widely disparate modules (eg, adding a message pointer
;    57          0057  1 !               object module to a library of normal object modules).
```

```
;    58          0058  1 !                - Replace obj$c_maxrecsiz with lbr$c_maxrecsiz.
;    59          0059  1 !
;    60          0060  1 !        V03-005 GJA0086         Greg Awdziewicz         14-May-1984
;    61          0061  1 !                Record length variable bound to history discriptor
;    62          0062  1 !                corrected to be a word (not longword).
;    63          0063  1 !
;    64          0064  1 !        V03-004 JWT0114         Jim Teague              20-Apr-1983
;    65          0065  1 !                Activate DCXSHR dynamically when needed.
;    66          0066  1 !
;    67          0067  1 !        V03-003 JWT0064         Jim Teague              11-Nov-1982
;    68          0068  1 !                Enlarged space allocated for DCX records.
;    69          0069  1 !
;    70          0070  1 !        V03-002 JWT0062         Jim Teague              28-Oct-1982
;    71          0071  1 !                Made DCX record descriptors static.
;    72          0072  1 !
;    73          0073  1 !        V03-001 JWT0056         Jim Teague              16-Sep-1982
;    74          0074  1 !                Equipped LBRSHR with DCX interface.
;    75          0075  1 !
;    76          0076  1 !        V02-118         RPG0118         Bob Grosso      02-Feb-1982
;    77          0077  1 !                Fix decr_refs deallocation bug.
;    78          0078  1 !
;    79          0079  1 !        V02-117         RPG0117         Bob Grosso      25-Jan-1982
;    80          0080  1 !                Complete random access by record rfa.
;    81          0081  1 !
;    82          0082  1 !        V02-116         RPG0116         Bob Grosso      15-Jan-1982
;    83          0083  1 !                Random access by record rfa.
;    84          0084  1 !                Fix history record boundary problem.
;    85          0085  1 !
;    86          0086  1 !        V02-115         RPG00115        Bob Grosso      17-Dec-1981
;    87          0087  1 !                Enhance update history deletion.
;    88          0088  1 !
;    89          0089  1 !        V02-114         RPG00114        Bob Grosso      16-Nov-1981
;    90          0090  1 !                Change lbr$get_record to support locate mode.
;    91          0091  1 !
;    92          0092  1 !        V02-113         RPG00113        Bob Grosso      25-Aug-1981
;    93          0093  1 !                Add messages to lbr$get_history and lbr$put_history.
;    94          0094  1 !
;    95          0095  1 !        V02-012         RPG00052        Bob Grosso      30-Jul-1981
;    96          0096  1 !                Correct the setting of control indexes.
;    97          0097  1 !                Convert messages.
;    98          0098  1 !
;    99          0099  1 !        V02-008         RPG00044        Bob Grosso      18-Jun-1981
;   100          0100  1 !                Replace lbr$c_maxluhlen with lbr$c_maxrecsiz
;   101          0101  1 !                Fix delete_data for multiple block spanning records.
;   102          0102  1 !
;   103          0103  1 !        V02-007         RPG00043        Bob Grosso      12-Jun-1981
;   104          0104  1 !                Comment history code.
;   105          0105  1 !
;   106          0106  1 !        V02-006         RPG00042        Bob Grosso      2-Jun-1981
;   107          0107  1 !                Correct delete_data to avoid looping on RFA past EOF.
;   108          0108  1 !
;   109          0109  1 !        V02-005         RPG00041        Bob Grosso      8-May-1981
;   110          0110  1 !                Refine lbr$get_history and lbr$put_history.
;   111          0111  1 !
;   112          0112  1 !        V02-004         RPG00035        Bob Grosso      22-Apr-1981
;   113          0113  1 !                Add lbr$put_history and lbr$get_history.
;   114          0114  1 !                Remove lbr_rkcache reference.
```

```
;  115          0115  1 !
;  116          0116  1 !       V02-003         RPG00006        Bob Grosso      5-Jan-1981
;  117          0117  1 !               Correct the BUILTIN declaration
;  118          0118  1 !
;  119          0119  1 !       V02-002         RPG34250        Bob Grosso      16-Dec-1980
;  120          0120  1 !               Correct the conversion of module insertion dates
;  121          0121  1 !               entered prior to Version 2 Librarian.
;  122          0122  1 !
;  123          0123  1 !--
;  124          0124  1
;  125          0125  1
```

```
;  127     0126  1 %SBTTL  'Declarations';
;  128     0127  1 LIBRARY
;  129     0128  1         'SYS$LIBRARY:STARLET.L32';      ! System data structures
;  130     0129  1 REQUIRE
;  131     0130  1         'PREFIX';
;  132     0269  1 REQUIRE
;  133     0270  1         'LBRDEF';
;  134     0861  1 REQUIRE
;  135     0862  1         'OLDFMTDEF';                    !Old format library structure definitions
;  136     0958  1
;  137     0959  1 EXTERNAL ROUTINE
;  138     0960  1     lbr$load_dcx,                       ! load dcxshr if not already loaded
;  139     0961  1     SYS$FAO : ADDRESSING_MODE (GENERAL), !Formatted ascii output
;  140     0962  1     lookup_cache : JSB_2,               ! Lookup disk vbn in cache table
;  141     0963  1     add_cache : JSB_2,                  ! Add vbn to cache table
;  142     0964  1     validate_ctl : JSB_1,               ! Validate library control index
;  143     0965  1     get_mem : JSB_2,                    ! Allocate dynamic memory
;  144     0966  1     dealloc_mem : JSB_2,                ! Deallocate dynamic memory
;  145     0967  1     find_key,                           ! Find key in index and return position
;  146     0968  1     mark_dirty,                         ! Mark block dirty
;  147     0969  1     alloc_block : JSB_2,                ! Allocate a disk block
;  148     0970  1     dealloc_block : JSB_1,              ! Deallocate a disk block
;  149     0971  1     read_block : JSB_2,                 ! Read disk block
;  150     0972  1     incr_rfa : JSB_2 NOVALUE,           ! Update RFA
;  151     0973  1     find_block : JSB_3,                 ! Locate a block and cache it if not there already
;  152     0974  1     get_zmem  : JSB_2;                  ! Allocate VM and zero it
;  153     0975  1
;  154     0976  1 FORWARD ROUTINE
;  155     0977  1     update_nextrfa : JSB_1,             ! Update next RFA in library header
;  156     0978  1     incr_refcnt,                        ! Increment module reference count
;  157     0979  1     decr_refcnt,                        ! Decrement module reference count
;  158     0980  1     set_module,                         ! Read and optionally update module header
;  159     0981  1     map_blk_to_mem,                     ! Find/allocate block and map into memory
;  160     0982  1     delete_data,                        ! Delete data
;  161     0983  1     write_record,                       ! Write record to library
;  162     0984  1     read_old_record : JSB_2,            ! Read record from old format library
;  163     0985  1     read_record : JSB_2,                ! Read record from library
;  164     0986  1     add_luhrecord,                      ! Store the LUH record
;  165     0987  1     delete_luhrecord;                   ! Skip first luh record and return any freed blocks
;  166     0988  1
;  167     0989  1 EXTERNAL
;  168     0990  1     dcxshr_address,
;  169     0991  1     dcx_compress_data,
;  170     0992  1     dcx_expand_data,
;  171     0993  1     mem$l_maxblk,
;  172     0994  1     lbr$gl_maxread,                     ! Max number blocks to read at once
;  173     0995  1     lbr$gl_rmsstv,                      ! Return STV on errors here
;  174     0996  1     lbr$gt_eotdesc : VECTOR [4,BYTE], ! End of text ASCIC record
;  175     0997  1     lbr$gl_control: REF BBLOCK;  ! Pointer to control block
;  176     0998  1
;  177     0999  1 EXTERNAL LITERAL
;  178     1000  1     lbr$_emptyhist,
;  179     1001  1     lbr$_hdrtrunc,
;  180     1002  1     lbr$_illop,
;  181     1003  1     lbr$_intrnlerr,
;  182     1004  1     lbr$_invrfa,
;  183     1005  1     lbr$_lkpnotdon,
```

```
;  184          1006  1      lbr$_nohistory,
;  185          1007  1      lbr$_normal,
;  186          1008  1      lbr$_recIng,
;  187          1009  1      lbr$_rectrunc,
;  188          1010  1      lbr$_rfapasteof,
;  189          1011  1      lbr$_stillkeys;
;  190          1012  1
;  191          1013  1  ! Replacing uses of obj$c_maxrecsiz with lbr$c_maxrecsiz requires that
;  192          1014  1  ! they have the same value.  Also, provide a larger value for DCX
;  193          1015  1  ! encoded records since they may in fact grow when they are "reduced" --
;  194          1016  1  ! e.g., adding a message pointer object module to an object library.
;  195          1017  1  !
;  196        U 1018  1  %IF lbr$c_maxrecsiz NEQ obj$c_maxrecsiz %THEN
;  197        U 1019  1  %ERROR ('lbr$c_maxrecsiz is not equivalent to obj$c_maxrecsiz')
;  198          1020  1  %FI
;  199          1021  1
;  200          1022  1  LITERAL
;  201          1023  1      lbr_dcx$c_maxrecsiz= 2 * lbr$c_maxrecsiz;    ! Allow DCX maximum record size
;  202          1024  1                                                   ! to be larger than normal.
;  203          1025  1
;  204          1026  1  PSECT OWN = $CODE$;                              !Own data is all shareable
;  205          1027  1
;  206          1028  1  OWN
;  207          1029  1      fao_old2newdate : countedstring ('!ZW-!AC-19!ZW 00:00:00'),
;  208          1030  1      jan :    countedstring ('JAN'),     !ASCIC strings for months **MUST BE ONLY 3 BYTES LONG TO FIT IN A WO
;  209          1031  1      feb :    countedstring ('FEB'),
;  210          1032  1      mar :    countedstring ('MAR'),
;  211          1033  1      apr :    countedstring ('APR'),
;  212          1034  1      may :    countedstring ('MAY'),
;  213          1035  1      jun :    countedstring ('JUN'),
;  214          1036  1      jul :    countedstring ('JUL'),
;  215          1037  1      aug :    countedstring ('AUG'),
;  216          1038  1      sep :    countedstring ('SEP'),
;  217          1039  1      oct :    countedstring ('OCT'),
;  218          1040  1      nov :    countedstring ('NOV'),
;  219          1041  1      dec :    countedstring ('DEC');
;  220          1042  1
;  221          1043  1  BIND
;  222          1044  1      months = jan : VECTOR [,LONG];      !Months of year table
```

```
;  224          1045  1 %SBTTL 'LBR$FIND';
;  225          1046  1 GLOBAL ROUTINE lbr$find (control_index, txtrfa) =
;  226          1047  2 BEGIN
;  227          1048  2 !++
;  228          1049  2 ! FUNCTIONAL DESCRIPTION:
;  229          1050  2 !
;  230          1051  2 !       This routine performs a lookup on a module given the RFA
;  231          1052  2 !
;  232          1053  2 ! Inputs:
;  233          1054  2 !
;  234          1055  2 !       control_index   is the address of a longword containing the
;  235          1056  2 !                       control index for th library.
;  236          1057  2 !       txtrfa          is the address of a 6-byte buffer containing
;  237          1058  2 !                       the module RFA to find.
;  238          1059  2 !
;  239          1060  2 ! Outputs:
;  240          1061  2 !
;  241          1062  2 !       The file is positioned to read the module's text
;  242          1063  2 !
;  243          1064  2 !--
;  244          1065  2
;  245          1066  2 MAP
;  246          1067  2     txtrfa: REF BBLOCK;                    ! Pointer to RFA
;  247          1068  2
;  248          1069  2 LOCAL
;  249          1070  2     descrip : BBLOCK [dsc$c_s_bln];
;  250          1071  2
;  251          1072  2 BIND
;  252          1073  2     length = descrip [dsc$w_length] : WORD,
;  253          1074  2     addr = descrip [dsc$a_pointer] : REF BBLOCK;
;  254          1075  2
;  255          1076  2 perform (validate_ctl (..control_index));       ! Validate control table index
;  256          1077  2
;  257          1078  3 BEGIN
;  258          1079  3     BIND
;  259          1080  3         header = .lbr$gl_control [lbr$l_hdrptr] : BBLOCK, ! Pointer to header
;  260          1081  3         context = .lbr$gl_control [lbr$l_ctxptr] : BBLOCK, ! Pointer to context block
;  261          1082  3         eomodrfa = context [ctx$b_eomodrfa] : BBLOCK,   ! End of module RFA
;  262          1083  3         readrfa = context [ctx$b_readrfa] : BBLOCK;     ! Next RFA for read
;  263          1084  3
;  264          1085  3     IF .context [ctx$v_oldlib]                          ! If old format library
;  265          1086  3     THEN
;  266          1087  4         BEGIN
;  267          1088  4             CH$MOVE (rfa$c_length, .txtrfa, readrfa);   ! Set RFA for reading
;  268          1089  4             eomodrfa [rfa$l_vbn] = 0;                   ! Disable end of module
;  269          1090  4             eomodrfa [rfa$w_offset] = 0;                ! until after header read
;  270          1091  4             perform (read_old_record (readrfa, descrip)); ! Read and skip header
;  271          1092  4             IF .length NEQ omh$c_size
;  272          1093  4                 THEN RETURN lbr$_invrfa
;  273          1094  4                 ELSE
;  274          1095  5                     BEGIN
;  275          1096  5                         BIND
;  276          1097  5                             modsizwords = addr [omh$l_modsiz] : VECTOR [,WORD];
;  277          1098  5
;  278          1099  5                         CH$MOVE (rfa$c_length, .txtrfa, eomodrfa);
;  279          1100  5                         incr_rfa (.modsizwords [1] + .modsizwords [0] *
;  280          1101  5                                                 %X'10000', eomodrfa);
```

```
;  281          1102  5                        END
;  282          1103  4                    END
;  283          1104  3                ELSE
;  284          1105  4                    BEGIN
;  285          1106  4                        CH$MOVE (rfa$c_length, .txtrfa, readrfa);
;  286          1107  4                        perform (read_record (readrfa, descrip)); ! Read module header to skip it
;  287          1108  4                        IF .length NEQ mhd$c_mhdlen+.header [lhd$b_mhdusz] ! If module header not correct length
;  288          1109  4                        OR .addr [mhd$l_refcnt] EQL 0                      !  or ref count is 0
;  289          1110  4                        THEN RETURN lbr$_invrfa;                           !  then RFA is bad
;  290          1111  3                        END;
;  291          1112  3                context [ctx$v_lkpdon] = true;              ! Indicate lookup_key done
;  292          1113  2                END;
;  293          1114  2
;  294          1115  2            RETURN true;
;  295          1116  2
;  296          1117  1            END;


                                                                .TITLE  LBR_GETPUT
                                                                .IDENT  \V04-000\

                                                                .PSECT  $CODE$,NOWRT,2

                                                        16  00000 FAO_OLD2NEWDATE:
                                                                .BYTE   22
30  20  57  5A  21  39  31  2D  43  41  21  2D  57  5A  21  00001          .ASCII  \!ZW-!AC-19!ZW 00:00:00\                        ;
                                30  30  3A  30  30  3A  30  00010                                                                  ;
                                                            00017          .BLKB   1
                                                            03  00018 JAN:      .BYTE   3                                       ;
                                                    4E  41  4A  00019          .ASCII  \JAN\                                       ;
                                                            03  0001C FEB:      .BYTE   3                                       ;
                                                    42  45  46  0001D          .ASCII  \FEB\                                       ;
                                                            03  00020 MAR:      .BYTE   3                                       ;
                                                    52  41  4D  00021          .ASCII  \MAR\                                       ;
                                                            03  00024 APR:      .BYTE   3                                       ;
                                                    52  50  41  00025          .ASCII  \APR\                                       ;
                                                            03  00028 MAY:      .BYTE   3                                       ;
                                                    59  41  4D  00029          .ASCII  \MAY\                                       ;
                                                            03  0002C JUN:      .BYTE   3                                       ;
                                                    4E  55  4A  0002D          .ASCII  \JUN\                                       ;
                                                            03  00030 JUL:      .BYTE   3                                       ;
                                                    4C  55  4A  00031          .ASCII  \JUL\                                       ;
                                                            03  00034 AUG:      .BYTE   3                                       ;
                                                    47  55  41  00035          .ASCII  \AUG\                                       ;
                                                            03  00038 SEP:      .BYTE   3                                       ;
                                                    50  45  53  00039          .ASCII  \SEP\                                       ;
                                                            03  0003C OCT:      .BYTE   3                                       ;
                                                    54  43  4F  0003D          .ASCII  \OCT\                                       ;
                                                            03  00040 NOV:      .BYTE   3                                       ;
                                                    56  4F  4E  00041          .ASCII  \NOV\                                       ;
                                                            03  00044 DEC:      .BYTE   3                                       ;
                                                    43  45  44  00045          .ASCII  \DEC\                                       ;

                                                MONTHS=                             JAN
                                                                .EXTRN  LBR$LOAD_DCX, SYS$FAO
                                                                .EXTRN  LOOKUP_CACHE, ADD_CACHE
                                                                .EXTRN  VALIDATE_CTL, GET_MEM
```

```
                                                        .EXTRN  DEALLOC_MEM, FIND_KEY
                                                        .EXTRN  MARK_DIRTY, ALLOC_BLOCK
                                                        .EXTRN  DEALLOC_BLOCK, READ_BLOCK
                                                        .EXTRN  INCR_RFA, FIND_BLOCK
                                                        .EXTRN  GET_ZMEM, DCXSRR_ADDRESS
                                                        .EXTRN  DCX_COMPRESS_DATA
                                                        .EXTRN  DCX_EXPAND_DATA
                                                        .EXTRN  MEM$L_MAXBLK, LBR$GL_MAXREAD
                                                        .EXTRN  LBR$GL_RMSSTV, LBR$GT_EOTDESC
                                                        .EXTRN  LBR$GL_CONTROL, LBR$_EMPTYHIST
                                                        .EXTRN  LBR$_HDRTRUNC, LBR$_ILLOP
                                                        .EXTRN  LBR$_INTRNLERR, LBR$_INVRFA
                                                        .EXTRN  LBR$_LKPNOTDON, LBR$_NOHISTORY
                                                        .EXTRN  LBR$_NORMAL, LBR$_RECLNG
                                                        .EXTRN  LBR$_RECTRUNC, LBR$_RFAPASTEOF
                                                        .EXTRN  LBR$_STILLKEYS

                                    0FFC 00000          .ENTRY  LBR$FIND, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,-; 1046
                                                                R11
                                 5E      08  C2 00002          SUBL2   #8, SP
                                 50  04  BC  D0 00005          MOVL    @CONTROL_INDEX, R0                    ; 1076
                                        0000G 30 00009          BSBW    VALIDATE_CTL
                                 63      50  E9 0000C          BLBC    STATUS, 2$
                                 50  0000G CF  D0 0000F          MOVL    LBR$GL_CONTROL, R0                    ; 1080
                                 57    0A  A0  D0 00014          MOVL    10(R0), R7                            ;
                                 56    0E  A0  D0 00018          MOVL    14(R0), R6                            ; 1081
                                 58    22  A6  9E 0001C          MOVAB   34(R6), R8                            ; 1082
                              3D      04  A6     05  E1 00020    BBC     #5, 4(R6), 1$                         ; 1085
                           28 A6      08  BC     06  28 00025    MOVC3   #6, @TXTRFA, 40(R6)                   ; 1088
                                          68     D4  0002B       CLRL    (R8)                                  ; 1089
                                       04 A8     B4  0002D       CLRW    4(R8)                                 ; 1090
                                 51       6E     9E  00030       MOVAB   DESCRIP, R1                           ; 1091
                                 50    28 A6     9E  00033       MOVAB   40(R6), R0                            ;
                                       0000V     30  00037       BSBW    READ_OLD_RECORD                       ;
                                 5E       50     E9  0003A       BLBC    STATUS, 5$                            ;
                                 1C       6E     B1  0003D       CMPW    LENGTH, #28                           ; 1092
                                          4A     12  00040       BNEQ    3$                                    ;
                              57      04  AE     02  C1 00042    ADDL3   #2, ADDR, R7                          ; 1097
                              68      08  BC     06  28 00047    MOVC3   #6, @TXTRFA, (R8)                     ; 1099
                                 50    02 A7     3C  0004C       MOVZWL  2(R7), R0                             ; 1100
                                 57       67     3C  00050       MOVZWL  (R7), R7                              ;
                              57         57     10  78 00053    ASHL    #16, R7, R7                           ;
                                 50       57     C0  00057       ADDL2   R7, R0                                ;
                                 51       58     D0  0005A       MOVL    R8, R1                                ;
                                       0000G     30  0005D       BSBW    INCR_RFA                              ;
                                          32     11  00060       BRB     4$                                    ; 1092
                           28 A6      08  BC     06  28 00062 1$: MOVC3   #6, @TXTRFA, 40(R6)                   ; 1106
                                 51       6E     9E  00068       MOVAB   DESCRIP, R1                           ; 1107
                                 50    28 A6     9E  0006B       MOVAB   40(R6), R0                            ;
                                       0000V     30  0006F       BSBW    READ_RECORD                           ;
                                 26       50     E9  00072 2$:   BLBC    STATUS, 5$                            ;
                                 50    3C A7     9A  00075       MOVZBL  60(R7), R0                            ; 1108
                                 50       10     C0  00079       ADDL2   #16, R0                               ;
                           50               6E    10  00     ED 0007C    CMPZV   #0, #16, LENGTH, R0           ;
                                          09     12  00081       BNEQ    3$                                    ;
                                 50    04 AE     D0  00083       MOVL    ADDR, R0                              ; 1109
                                       04 A0     D5  00087       TSTL    4(R0)                                 ;
```

```
                                08  12 0008A          BNEQ    4$                                      ;
                      50 00000000G  8F  D0 0008C 3$:  MOVL    #LBR$_INVRFA, R0                        ; 1110
                                    04  00093          RET                                             ;
                   04   A6          02  88 00094 4$:  BISB2   #2, 4(R6)                               ; 1112
                        50          01  D0 00098       MOVL    #1, R0                                  ; 1115
                                        04  0009B 5$:  RET                                             ; 1117

; Routine Size:  156 bytes,    Routine Base:  $CODE$ + 0048
```

```
;  298          1118  1 %SBTTL  'LBR$PUT_RECORD';
;  299          1119  1 GLOBAL ROUTINE lbr$put_record (control_index, bufdesc, txtrfa) =
;  300          1120  2 BEGIN
;  301          1121  2
;  302          1122  2 !++
;  303          1123  2 !
;  304          1124  2 ! FUNCTIONAL DESCRIPTION:
;  305          1125  2 !
;  306          1126  2 !       This routine writes the record passed to it out to the library.
;  307          1127  2 !
;  308          1128  2 !
;  309          1129  2 ! CALLING SEQUENCE:
;  310          1130  2 !
;  311          1131  2 !       status = lbr$put_record (control_index, bufdesc, txtrfa)
;  312          1132  2 !
;  313          1133  2 ! INPUT PARAMETERS:
;  314          1134  2 !
;  315          1135  2 !       control_index           is the index returned from lbr$ini_control
;  316          1136  2 !       bufdesc                 is the string descriptor for the record
;  317          1137  2 !                                to be output
;  318          1138  2 !
;  319          1139  2 !
;  320          1140  2 ! OUTPUT PARAMETERS:
;  321          1141  2 !
;  322          1142  2 !       txtrfa                  is a pointer to a two-longword array that
;  323          1143  2 !                                is filled in with the RFA of the record
;  324          1144  2 !                                (i.e. the module header if first PUT)
;  325          1145  2 !
;  326          1146  2 !--
;  327          1147  2 MAP
;  328          1148  2     bufdesc : REF BBLOCK,                      !Pointer to string descriptor
;  329          1149  2     txtrfa  : REF BBLOCK;                      !Pointer to array
;  330          1150  2
;  331          1151  2 LOCAL
;  332          1152  2     reduce_record,
;  333          1153  2     localrfa : BBLOCK [dsc$c_s_bln];
;  334          1154  2
;  335          1155  2 perform (validate_ctl (..control_index));
;  336          1156  2
;  337          1157  3 BEGIN
;  338          1158  3     BIND
;  339          1159  3         context = .lbr$gl_control [lbr$l_ctxptr] : BBLOCK, !Point to context block
;  340          1160  3         header = .lbr$gl_control [lbr$l_hdrptr] : BBLOCK, !and header
;  341          1161  3         nxtputrfa = context [ctx$b_nxtputrfa] : BBLOCK,  !RFA for next PUT
;  342          1162  3         hdrnxtrfa = header [lhd$b_nextrfa] : BBLOCK; !Name next RFA
;  343          1163  3
;  344          1164  3     IF .context [ctx$v_oldlib]                 !Cannot write to old library
;  345          1165  3         OR .context [ctx$v_ronly]              ! or read only library
;  346          1166  3     THEN RETURN lbr$_illop;
;  347          1167  3
;  348          1168  3     IF .bufdesc [dsc$w_length] GTRU lbr$c_maxrecsiz     !If record length illegal
;  349          1169  3     THEN RETURN lbr$_reclng;                   ! then return with error
;  350          1170  3
;  351          1171  3     reduce_record = .header[lhd$l_dcxmapvbn] NEQ 0;
;  352          1172  3 !
;  353          1173  3 !       Create the module header record if this is the first put.
;  354          1174  3 !
```

```
;  355          1175  3     CH$MOVE (rfa$c_length, nxtputrfa, localrfa);
;  356          1176  3     IF NOT .context [ctx$v_mhdout]                !If module header needs to be written
;  357          1177  4     THEN BEGIN
;  358          1178  4         BIND
;  359          1179  4             mhdlen = .header [lhd$b_mhdusz] + mhd$c_mhdlen; !Length of module header
;  360          1180  4
;  361          1181  4         LOCAL
;  362          1182  4             mhdrec : BBLOCK [lbr$c_maxhdrsiz]; !buffer for module header
;  363          1183  4
;  364          1184  4         CH$FILL (0, lbr$c_maxhdrsiz, mhdrec);    !Zero the module header
;  365          1185  4         mhdrec [mhd$b_id] = mhd$c_mhdid;         !Set ident
;  366          1186  4         $GETTIM (TIMADR = mhdrec [mhd$l_datim]);!Set in time of insertion
;  367          1187  4         header [lhd$l_updtim] = .mhdrec [mhd$l_datim]; !Set new time into header
;  368          1188  4         header [lhd$l_updtim] + 4 = .(mhdrec [mhd$l_datim] + 4);
;  369          1189  4         CH$MOVE (rfa$c_length, hdrnxtrfa, localrfa);
;  370          1190  4         perform (write_record (mhdlen, mhdrec, localrfa, false, .txtrfa)); !write the header
;  371          1191  4         context [ctx$v_mhdout] = true;           !No longer need module header
;  372          1192  4         header [lhd$l_modhdrs] = .header [lhd$l_modhdrs] + 1; !Count another module header
;  373          1193  4         update_nextrfa (localrfa);                          !Update next RFA
;  374          1194  3         END;
;  375          1195  3
;  376          1196  3     IF .reduce_record
;  377          1197  3     THEN
;  378          1198  4         BEGIN
;  379          1199  4         BIND
;  380          1200  4             compress_desc = context[ctx$l_dcxrecdsc]: BBLOCK[dsc$c_s_bln];
;  381          1201  4         if .dcxshr_address eql 0
;  382          1202  4         then
;  383          1203  4             perform (lbr$load_dcx());
;  384          1204  4         compress_desc[dsc$w_length] = lbr_dcx$c_maxrecsiz;
;  385          1205  4         bufdesc[dsc$b_class] = dsc$k_class_s;
;  386          1206  4         bufdesc[dsc$b_dtype] = dsc$k_dtype_t;
;  387          1207  4         perform ((.dcx_compress_data) ( context[ctx$l_dcxctx], .bufdesc, compress_desc, compress_desc[dsc$w_
;  388        P 1208  4         perform (write_record (.compress_desc[dsc$w_length], .compress_desc[dsc$a_pointer],
;  389          1209  4                               localrfa, false));
;  390          1210  4         END
;  391          1211  3     ELSE
;  392        P 1212  3         perform (write_record (.bufdesc[dsc$w_length], .bufdesc[dsc$a_pointer],
;  393          1213  3                               localrfa, false));
;  394          1214  3
;  395          1215  3     update_nextrfa (localrfa);                              !Update next RFA
;  396          1216  3     CH$MOVE (rfa$c_length, localrfa, nxtputrfa);
;  397          1217  3     context [ctx$v_hdrdirty] = true;                !Flag header is dirty
;  398          1218  3     RETURN ss$_normal
;  399          1219  3     END
;  400          1220  1 END;                                    ! Of lbr$put_record


                                                    .EXTRN  SYS$GETTIM

                                    0FFC 00000      .ENTRY  LBR$PUT_RECORD, Save R2,R3,R4,R5,R6,R7,R8,- ; 1119
                                                            R9,R10,R11                                  ;
                              5E    FF78   CE  9E 00002     MOVAB   -136(SP), SP                        ;
                              50      04   BC  D0 00007     MOVL    @CONTROL_INDEX, R0                  ; 1155
                                  0000G 30 0000B          BSBW    VALIDATE_CTL                        ;
                              01        50  E8 0000E      BLBS    STATUS, 1$                          ;
```

```
                                            04 00011          RET                                              ;
                  50      0000G   CF        D0 00012 1$:      MOVL    LBR$GL_CONTROL, R0                       ; 1159
                  58         0E   A0        D0 00017          MOVL    14(R0), R8                               ;
                  56         0A   A0        D0 0001B          MOVL    10(R0), R6                               ; 1160
                  5A         04   A8        9E 0001F          MOVAB   4(R8), R10                               ; 1164
        04        6A              05        E0 00023          BBS     #5, (R10), 2$                            ;
                                  6A        95 00027          TSTB    (R10)                                    ; 1165
                                  08        18 00029          BGEQ    3$                                       ;
                  50 00000000G    8F        D0 0002B 2$:      MOVL    #LBR$_ILLOP, R0                          ; 1166
                                            04 00032          RET                                              ;
                  57         08   AC        D0 00033 3$:      MOVL    BUFDESC, R7                              ; 1168
             0800 8F              67        B1 00037          CMPW    (R7), #2048                              ;
                                  08        1B 0003C          BLEQU   4$                                       ;
                  50 00000000G    8F        D0 0003E          MOVL    #LBR$_RECLNG, R0                         ; 1169
                                            04 00045          RET                                              ;
                                  50        D4 00046 4$:      CLRL    R0                                       ; 1171
                          008C    C6        D5 00048          TSTL    140(R6)                                  ;
                                  02        13 0004C          BEQL    5$                                       ;
                                  50        D6 0004E          INCL    R0                                       ;
                  5B              50        D0 00050 5$:      MOVL    R0, REDUCE_RECORD                        ;
     F8   AD      3E   A8         06        28 00053          MOVC3   #6, 62(R8), LOCALRFA                     ; 1175
          4B      6A              04        E0 00059          BBS     #4, (R10), 6$                            ; 1176
                  59         3C   A6        9A 0005D          MOVZBL  60(R6), R9                               ; 1179
                  59              10        C0 00061          ADDL2   #16, R9                                  ;
0080  8F          00      6E      00        2C 00064          MOVC5   #0, (SP), #0, #128, MHDREC               ; 1184
                                  6E           0006B                                                           ;
                  01      AE      AD        8F 90 0006C       MOVB    #-83, MHDREC+1                           ; 1185
                             08   AE        9F 00071          PUSHAB  MHDREC+8                                 ; 1186
           00000000G      00      01        FB 00074          CALLS   #1, SYS$GETTIM                           ;
                  34      A6 08   AE        7D 0007B          MOVQ    MHDREC+8, 52(R6)                         ; 1187
     F8   AD      4C      A6      06        28 00080          MOVC3   #6, 76(R6), LOCALRFA                     ; 1189
                             0C   AC        DD 00086          PUSHL   TXTRFA                                   ; 1190
                                  7E        D4 00089          CLRL    -(SP)                                    ;
                             F8   AD        9F 0008B          PUSHAB  LOCALRFA                                 ;
                             0C   AE        9F 0008E          PUSHAB  MHDREC                                   ;
                                  59        DD 00091          PUSHL   R9                                       ;
                  0000V   CF      05        FB 00093          CALLS   #5, WRITE_RECORD                         ;
                          71      50        E9 00098          BLBC    STATUS, 10$                              ;
                          6A      10        88 0009B          BISB2   #16, (R10)                               ; 1191
                             74   A6        D6 0009E          INCL    116(R6)                                  ; 1192
                  50         F8   AD        9E 000A1          MOVAB   LOCALRFA, R0                             ; 1193
                               0000V        30 000A5          BSBW    UPDATE_NEXTRFA                           ;
                  3B              5B        E9 000A8 6$:      BLBC    REDUCE_RECORD, 8$                        ; 1196
                  52         5A   A8        9E 000AB          MOVAB   90(R8), R2                               ; 1200
                          0000G   CF        D5 000AF          TSTL    DCXSHR_ADDRESS                           ; 1201
                                  08        12 000B3          BNEQ    7$                                       ;
                  0000G   CF      00        FB 000B5          CALLS   #0, LBR$LOAD_DCX                         ; 1203
                          4F      50        E9 000BA          BLBC    STATUS, 10$                              ;
                          62 1000 8F        B0 000BD 7$:      MOVW    #4096, (R2)                              ; 1204
                  02      A7 010E 8F        B0 000C2          MOVW    #270, 2(R7)                              ; 1206
                                  52        DD 000C8          PUSHL   R2                                       ; 1207
                                  52        DD 000CA          PUSHL   R2                                       ;
                                  57        DD 000CC          PUSHL   R7                                       ;
                             52   A8        9F 000CE          PUSHAB  82(R8)                                   ;
                  0000G   DF      04        FB 000D1          CALLS   #4, @DCX_COMPRESS_DATA                   ;
                          33      50        E9 000D6          BLBC    STATUS, 10$                              ;
                                  7E        D4 000D9          CLRL    -(SP)                                    ; 1209
```

```
                                F8  AD  9F 000DB          PUSHAB  LOCALRFA                                ;
                                04  A2  DD 000DE          PUSHL   4(R2)                                   ;
                             7E     62  3C 000E1          MOVZWL  (R2), -(SP)                             ;
                                    0B  11 000E4          BRB     9$                                      ;
                                    7E  D4 000E6 8$:      CLRL    -(SP)                                   ; 1213
                                F8  AD  9F 000E8          PUSHAB  LOCALRFA                                ;
                                04  A7  DD 000EB          PUSHL   4(R7)                                   ;
                             7E     67  3C 000EE          MOVZWL  (R7), -(SP)                             ;
                       0000V CF     04  FB 000F1 9$:      CALLS   #4, WRITE_RECORD                        ;
                             13     50  E9 000F6          BLBC    STATUS, 10$                             ;
                             50     F8  AD  9E 000F9      MOVAB   LOCALRFA, R0                            ; 1215
                                0000V 30 000FD          BSBW    UPDATE_NEXTRFA                          ;
                 3E  A8         F8  AD     06  28 00100  MOVC3   #6, LOCALRFA, 62(R8)                   ; 1216
                             6A     08  88 00106          BISB2   #8, (R10)                               ; 1217
                             50     01  D0 00109          MOVL    #1, R0                                  ; 1218
                                    04 0010C 10$:         RET                                             ; 1220

; Routine Size:  269 bytes,    Routine Base:  $CODE$ + 00E4
```

```
;  402          1221  1 %SBTTL  'LBR$PUT_END';
;  403          1222  1 GLOBAL ROUTINE lbr$put_end (control_index) =
;  404          1223  2 BEGIN
;  405          1224  2 !++
;  406          1225  2 !
;  407          1226  2 ! FUNCTIONAL DESCRIPTION:
;  408          1227  2 !
;  409          1228  2 !       This routine is called to finish putting text into the library.
;  410          1229  2 !
;  411          1230  2 !
;  412          1231  2 ! CALLING SEQUENCE:
;  413          1232  2 !
;  414          1233  2 !       status = lbr$put_end (control_index)
;  415          1234  2 !
;  416          1235  2 ! INPUT PARAMETERS:
;  417          1236  2 !
;  418          1237  2 !       control_index           is the control index returned from lbr$ini_control
;  419          1238  2 !
;  420          1239  2 ! IMPLICIT OUTPUTS:
;  421          1240  2 !
;  422          1241  2 !       An end of text record is written.
;  423          1242  2 !
;  424          1243  2 !--
;  425          1244  2
;  426          1245  2 LOCAL
;  427          1246  2     localrfa : BBLOCK [dsc$c_s_bln];
;  428          1247  2
;  429          1248  2 perform (validate_ctl (..control_index));              !Validate control index
;  430          1249  3 BEGIN
;  431          1250  3     BIND
;  432          1251  3         context = .lbr$gl_control [lbr$l_ctxptr] : BBLOCK, !Get context block address
;  433          1252  3         header = .lbr$gl_control [lbr$l_hdrptr] : BBLOCK,  !Get header address
;  434          1253  3         nxtputrfa = context [ctx$b_nxtputrfa] : BBLOCK;
;  435          1254  3
;  436          1255  3     IF .context [ctx$v_oldlib]                      !Error if old library
;  437          1256  3         OR .context [ctx$v_ronly]
;  438          1257  3     THEN RETURN lbr$_illop;
;  439          1258  3
;  440          1259  3     CH$MOVE (rfa$c_length, nxtputrfa, localrfa);
;  441        P 1260  3     perform (write_record (.lbr$gt_eotdesc [0], lbr$gt_eotdesc [1],
;  442          1261  3                            localrfa, false));
;  443          1262  3     update_nextrfa (localrfa);                          !Update next RFA
;  444          1263  3     nxtputrfa [rfa$l_vbn] = 0;                      !Zero next put RFA
;  445          1264  3     context [ctx$v_mhdout] = false;                 !Need module header next PUT
;  446          1265  3     context [ctx$v_hdrdirty] = true;                !Flag header is dirty
;  447          1266  2     END;
;  448          1267  2 RETURN ss$_normal
;  449          1268  1 END;                                  ! Of lbr$put_end


                                0FFC 00000          .ENTRY  LBR$PUT_END, Save R2,R3,R4,R5,R6,R7,R8,R9,- ; 1222
                                                            R10,R11                                     ;
                      5E        08   C2 00002       SUBL2   #8, SP                                      ;
                      50   04   BC   D0 00005       MOVL    @CONTROL_INDEX, R0                          ; 1248
```

```
                                    0000G 30 00009          BSBW    VALIDATE_CTL                                    ;
                                 4A       50 E9 0000C       BLBC    STATUS, 3$                                      ;
                                 50 0000G CF D0 0000F       MOVL    LBR$GL_CONTROL, R0                              ; 1251
                                 56   0E  A0 D0 00014       MOVL    14(R0), R6                                      ;
                          05     04  A6    05 E0 00018       BBS     #5, 4(R6), 1$                                   ; 1255
                                       04  A6 95 0001D       TSTB    4(R6)                                           ; 1256
                                           08 18 00020       BGEQ    2$                                              ;
                                 50 00000000G 8F D0 00022 1$:  MOVL    #LBR$_ILLOP, R0                                 ; 1257
                                              04 00029       RET                                                     ;
                          6E     3E  A6       06 28 0002A 2$:  MOVC3   #6, 62(R6), LOCALRFA                            ; 1259
                                              7E D4 0002F       CLRL    -(SP)                                           ; 1261
                                        04    AE 9F 00031       PUSHAB  LOCALRFA                                        ;
                                    0000G CF 9F 00034       PUSHAB  LBR$GT_EOTDESC+1                                ;
                                 7E 0000G CF 9A 00038       MOVZBL  LBR$GT_EOTDESC, -(SP)                           ;
                            0000V CF                 04 FB 0003D       CALLS   #4, WRITE_RECORD                        ;
                                 14                  50 E9 00042       BLBC    STATUS, 3$                              ;
                                 50                  6E 9E 00045       MOVAB   LOCALRFA, R0                            ; 1262
                                                0000V 30 00048       BSBW    UPDATE_NEXTRFA                          ;
                                              3E    A6 D4 0004B       CLRL    62(R6)                                  ; 1263
                                        04    A6          10 8A 0004E       BICB2   #16, 4(R6)                      ; 1264
                                        04    A6          08 88 00052       BISB2   #8, 4(R6)                       ; 1265
                                 50                       01 D0 00056       MOVL    #1, R0                          ; 1267
                                                             04 00059 3$:  RET                                     ; 1268

; Routine Size:  90 bytes,    Routine Base:  $CODE$ + 01F1
```

```
;  451     1269  1 %SBTTL  'LBR$GET_RECORD';
;  452     1270  1 GLOBAL ROUTINE lbr$get_record (control_index, inbufdesc, outbufdesc, txtrfa) =
;  453     1271  2 BEGIN
;  454     1272  2
;  455     1273  2 !++
;  456     1274  2 !
;  457     1275  2 ! FUNCTIONAL DESCRIPTION:
;  458     1276  2 !
;  459     1277  2 !       Read a record from the library
;  460     1278  2 !
;  461     1279  2 ! INPUT PARAMETERS:
;  462     1280  2 !
;  463     1281  2 !       control_index   Address of longword containing valid control index
;  464     1282  2 !       inbufdesc       Address of string descriptor for user-supplied buffer
;  465     1283  2 !       outbufdesc      (optional) Address of string descriptor for record if locate mode
;  466     1284  2 !       txtrfa          (optional) Address of rfa.
;  467     1285  2 !                           If empty then return rfa of retrieved record
;  468     1286  2 !                           If non-empty then retrieve record located by it.
;  469     1287  2 !
;  470     1288  2 ! IMPLICIT INPUTS:
;  471     1289  2 !
;  472     1290  2 !       An lbr$lookup_key or lbr$find must have been done to position to the module
;  473     1291  2 !
;  474     1292  2 !--
;  475     1293  2
;  476     1294  2 MAP
;  477     1295  2     inbufdesc : REF BBLOCK,
;  478     1296  2     outbufdesc : REF BBLOCK;
;  479     1297  2
;  480     1298  2 LOCAL
;  481     1299  2     status,
;  482     1300  2     use_call_rfa,                          ! remember whether caller supplied an rfa
;  483     1301  2     descrip : BBLOCK [dsc$c_s_bln];
;  484     1302  2
;  485     1303  2 BIND
;  486     1304  2     context = .lbr$gl_control[lbr$l_ctxptr]: BBLOCK,
;  487     1305  2     call_rfa = .txtrfa : BBLOCK,           ! caller supplied rfa, or slot to return rfa
;  488     1306  2     reclen = descrip [dsc$w_length] : WORD,
;  489     1307  2     recaddr = descrip [dsc$a_pointer];
;  490     1308  2
;  491     1309  2 BUILTIN
;  492     1310  2     NULLPARAMETER;              ! True if parameter not specified
;  493     1311  2
;  494     1312  2 perform (validate_ctl (..control_index));          !Validate control index
;  495     1313  2
;  496     1314  4 use_call_rfa = (IF (NOT NULLPARAMETER (4) )        ! if caller has supplied a non-zero rfa then use it.
;  497     1315  4                  THEN (.call_rfa [rfa$l_vbn] NEQ 0)
;  498     1316  2                  ELSE false);
;  499     1317  3 BEGIN
;  500     1318  3     BIND
;  501     1319  3         context = .lbr$gl_control [lbr$l_ctxptr] : BBLOCK, !Name context block
;  502     1320  3         lrab = .context [ctx$l_recrab] : BBLOCK,
;  503     1321  3         readrfa = context [ctx$b_readrfa] : BBLOCK,
;  504     1322  3         header = .lbr$gl_control [lbr$l_hdrptr] : BBLOCK;
;  505     1323  3
;  506     1324  3     IF .use_call_rfa
;  507     1325  3     THEN
```

```
;  508          1326  4          BEGIN
;  509          1327  4          context [ctx$v_lkpdon] = true;
;  510          1328  4          readrfa [rfa$l_vbn] = .call_rfa [rfa$l_vbn];
;  511          1329  4          readrfa [rfa$w_offset] = .call_rfa [rfa$w_offset];
;  512          1330  4          END
;  513          1331  3      ELSE
;  514          1332  4          BEGIN
;  515          1333  5          IF (NOT .context [ctx$v_lkpdon])
;  516          1334  4          THEN RETURN lbr$_lkpnotdon;
;  517          1335  4          IF NOT NULLPARAMETER (4)
;  518          1336  4          THEN
;  519          1337  5              BEGIN       ! return rfa of retrieved record
;  520          1338  5              call_rfa [rfa$l_vbn] = .readrfa [rfa$l_vbn];
;  521          1339  5              call_rfa [rfa$w_offset] = .readrfa [rfa$w_offset];
;  522          1340  4              END;
;  523          1341  3          END;
;  524          1342  3
;  525          1343  4      status = (IF NOT .context [ctx$v_oldlib]
;  526          1344  4                  THEN read_record ( readrfa, descrip)
;  527          1345  3                  ELSE read_old_record ( readrfa, descrip) );
;  528          1346  3
;  529          1347  3      IF .header[lhd$l_dcxmapvbn] NEQ 0  AND  .status
;  530          1348  3      THEN
;  531          1349  4          BEGIN
;  532          1350  4          BIND
;  533          1351  4              expand_desc = context[ctx$l_dcxrecdsc] : BBLOCK [dsc$c_s_bln];
;  534          1352  4          if .dcxshr_address eql 0
;  535          1353  4          then
;  536          1354  4              perform(lbr$load_dcx());
;  537          1355  4          expand_desc[dsc$w_length] = lbr$c_maxrecsiz;
;  538          1356  4          descrip[dsc$b_dtype] = dsc$k_dtype_t;
;  539          1357  4          descrip[dsc$b_class] = dsc$k_class_s;
;  540        P 1358  4          perform ( (.dcx_expand_data) ( context[ctx$l_dcxctx], descrip, expand_desc,
;  541          1359  4                                         reclen));
;  542          1360  4          recaddr = .expand_desc[dsc$a_pointer];
;  543          1361  3          END;
;  544          1362  3
;  545          1363  3      IF .status                                      !If successful read
;  546          1364  4      THEN BEGIN
;  547          1365  4          IF .lbr$gl_control [lbr$v_locate]           !Locate mode?
;  548          1366  4          THEN
;  549          1367  5              BEGIN
;  550          1368  5              IF NOT NULLPARAMETER (3)                    !Want buffer length?
;  551          1369  5              THEN
;  552          1370  6                  BEGIN
;  553          1371  6                  outbufdesc [dsc$w_length] = .reclen;    !yes--update descriptor
;  554          1372  6                  outbufdesc [dsc$a_pointer] = .recaddr;
;  555          1373  5                  END;
;  556          1374  5              END
;  557          1375  5          ELSE BEGIN
;  558          1376  5
;  559          1377  5              CH$MOVE (MIN (.reclen, .inbufdesc [dsc$w_length]),
;  560          1378  5                                  .recaddr, .inbufdesc [dsc$a_pointer]);
;  561          1379  5              IF .reclen GTR .inbufdesc [dsc$w_length]
;  562          1380  5                  THEN status = lbr$_rectrunc;
;  563          1381  5              IF NOT NULLPARAMETER (3)                    !Want buffer length?
;  564          1382  6                  THEN BEGIN
```

```
;  565          1383  6                    outbufdesc [dsc$w_length] = .reclen;
;  566          1384  6                    outbufdesc [dsc$a_pointer] = .inbufdesc [dsc$a_pointer];
;  567          1385  5                    END;
;  568          1386  4                END;          ! if move mode
;  569          1387  4
;  570          1388  4            END               ! if successful read
;  571          1389  3        ELSE IF .status EQL rms$_eof                !Otherwise, if end of module
;  572          1390  3                THEN context [ctx$v_lkpdon] = false;
;  573          1391  3
;  574          1392  2        END;
;  575          1393  2 RETURN .status
;  576          1394  1 END;                                               ! Of lbr$get_record


                                        0FFC 00000          .ENTRY  LBR$GET_RECORD, Save R2,R3,R4,R5,R6,R7,R8,- ; 1270
                                                                    R9,R10,R11                                      ;
                                    5E      08  C2 00002            SUBL2   #8, SP                                  ;
                                    52  10  AC  D0 00005            MOVL    TXTRFA, R2                              ; 1305
                                    50  04  BC  D0 00009            MOVL    @CONTROL_INDEX, R0                      ; 1312
                                        0000G 30 0000D              BSBW    VALIDATE_CTL                            ;
                                    01      50  E8 00010            BLBS    STATUS, T$                              ;
                                                04 00013            RET                                             ;
                                    04          6C  91 00014 1$:    CMPB    (AP), #4                                ; 1314
                                                12  1F 00017         BLSSU   3$                                      ;
                                            10  AC  D5 00019         TSTL    16(AP)                                  ;
                                                0D  13 0001C         BEQL    3$                                      ;
                                                50  D4 0001E         CLRL    R0                                      ; 1315
                                                62  D5 00020         TSTL    (R2)                                    ;
                                                02  13 00022         BEQL    2$                                      ;
                                                50  D6 00024         INCL    R0                                      ;
                                        54      50  D0 00026 2$:    MOVL    R0, USE_CALL_RFA                        ;
                                                02  11 00029         BRB     4$                                      ;
                                                54  D4 0002B 3$:    CLRL    USE_CALL_RFA                            ; 1314
                                        51  0000G   CF  D0 0002D 4$:    MOVL    LBR$GL_CONTROL, R1                  ; 1319
                                        53  0E      A1  D0 00032         MOVL    14(R1), R3                          ;
                                        50  28      A3  9E 00036         MOVAB   40(R3), R0                          ; 1321
                                        55  0A      A1  D0 0003A         MOVL    10(R1), R5                          ; 1322
                                        11          54  E9 0003E         BLBC    USE_CALL_RFA, 5$                    ; 1324
                                        54  04      A3  9E 00041         MOVAB   4(R3), R4                           ; 1327
                                        64          02  88 00045         BISB2   #2, (R4)                            ;
                                        60          62  D0 00048         MOVL    (R2), (R0)                          ; 1328
                                04  A0      04      A2  B0 0004B         MOVW    4(R2), 4(R0)                        ; 1329
                                                22  11 00050         BRB     7$                                      ; 1324
                                        54  04      A3  9E 00052 5$:    MOVAB   4(R3), R4                           ; 1333
                            08          64          01  E0 00056         BBS     #1, (R4), 6$                        ;
                                        50  00000000G 8F  D0 0005A         MOVL    #LBR$_LKPNOTDON, R0                 ; 1334
                                                04  00061         RET                                             ;
                                        04          6C  91 00062 6$:    CMPB    (AP), #4                            ; 1335
                                                0D  1F 00065         BLSSU   7$                                      ;
                                            10  AC  D5 00067         TSTL    16(AP)                                  ;
                                                08  13 0006A         BEQL    7$                                      ;
                                        62          60  D0 0006C         MOVL    (R0), (R2)                          ; 1338
                                04  A2      04      A0  B0 0006F         MOVW    4(R0), 4(R2)                        ; 1339
                            08          64          05  E0 00074 7$:    BBS     #5, (R4), 8$                        ; 1343
```

```
                   51              6E  9E 00078        MOVAB   DESCRIP, R1                              ; 1344
                                0000V  30 0007B        BSBW    READ_RECORD
                                   06  11 0007E        BRB     9$
                   51              6E  9E 00080  8$:   MOVAB   DESCRIP, R1                              ; 1345
                                0000V  30 00083        BSBW    READ_OLD_RECORD
                   57              50  D0 00086  9$:   MOVL    R0, STATUS
                           008C    C5  D5 00089        TSTL    140(R5)                                  ; 1347
                                   37  13 0008D        BEQL    12$
                   34              57  E9 0008F        BLBC    STATUS, 12$
                   52        5A    A3  9E 00092        MOVAB   90(R3), R2                               ; 1351
                          0000G    CF  D5 00096        TSTL    DCXSHR_ADDRESS                           ; 1352
                                   08  12 0009A        BNEQ    10$
         0000G     CF              00  FB 0009C        CALLS   #0, LBR$LOAD_DCX                         ; 1354
                   1A              50  E9 000A1        BLBC    STATUS, 11$
                   62      0800    8F  B0 000A4 10$:   MOVW    #2048, (R2)                              ; 1355
            02     AE      010E    8F  B0 000A9        MOVW    #270, DESCRIP+2                          ; 1356
                           4004    8F  BB 000AF        PUSHR   #^M<R2,SP>                               ; 1359
                             08    AE  9F 000B3        PUSHAB  DESCRIP
                             52    A3  9F 000B6        PUSHAB  82(R3)
         0000G     DF              04  FB 000B9        CALLS   #4, @DCX_EXPAND_DATA
                   71              50  E9 000BE 11$:   BLBC    STATUS, 18$
            04     AE        04    A2  D0 000C1        MOVL    4(R2), RECADDR                           ; 1360
                   5A              57  E9 000C6 12$:   BLBC    STATUS, 16$                              ; 1363
                   50     0000G    CF  D0 000C9        MOVL    LBR$GL_CONTROL, R0                       ; 1365
                   18        06    A0  E9 000CE        BLBC    6(R0), 13$
                   03              6C  91 000D2        CMPB    (AP), #3                                 ; 1368
                                   58  1F 000D5        BLSSU   17$
                             0C    AC  D5 000D7        TSTL    12(AP)
                                   53  13 000DA        BEQL    17$
                   50        0C    AC  D0 000DC        MOVL    OUTBUFDESC, R0                           ; 1371
                   60              6E  B0 000E0        MOVW    RECLEN, (R0)
            04     A0        04    AE  D0 000E3        MOVL    RECADDR, 4(R0)                           ; 1372
                                   45  11 000E8        BRB     17$                                      ; 1365
                   56        08    AC  D0 000EA 13$:   MOVL    INBUFDESC, R6                            ; 1377
                   50              6E  3C 000EE        MOVZWL  RECLEN, R0
                   50              66  B1 000F1        CMPW    (R6), R0
                                   03  1E 000F4        BGEQU   14$
                   50              66  3C 000F6        MOVZWL  (R6), R0
    04    B6       04        BE    50  28 000F9 14$:   MOVC3   R0, @RECADDR, @4(R6)                     ; 1378
                   66              6E  B1 000FF        CMPW    RECLEN, (R6)                             ; 1379
                                   07  1B 00102        BLEQU   15$
                   57 00000000G    8F  D0 00104        MOVL    #LBR$_RECTRUNC, STATUS                   ; 1380
                   03              6C  91 0010B 15$:   CMPB    (AP), #3                                 ; 1381
                                   1F  1F 0010E        BLSSU   17$
                             0C    AC  D5 00110        TSTL    12(AP)
                                   1A  13 00113        BEQL    17$
                   50        0C    AC  D0 00115        MOVL    OUTBUFDESC, R0                           ; 1383
                   60              6E  B0 00119        MOVW    RECLEN, (R0)
            04     A0        04    A6  D0 0011C        MOVL    4(R6), 4(R0)                             ; 1384
                                   0C  11 00121        BRB     17$                                      ; 1363
      0001827A     8F              57  D1 00123 16$:   CMPL    STATUS, #98938                           ; 1389
                                   03  12 0012A        BNEQ    17$
                   64              02  8A 0012C        BICB2   #2, (R4)                                 ; 1390
                   50              57  D0 0012F 17$:   MOVL    STATUS, R0                               ; 1393
                                       04 00132 18$:   RET                                              ; 1394

; Routine Size:  307 bytes,    Routine Base:  $CODE$ + 024B
```

```
;  578          1395  1 %SBTTL  'LBR$DELETE_DATA';
;  579          1396  1 GLOBAL ROUTINE lbr$delete_data (control_index, txtrfa) =
;  580          1397  2 BEGIN
;  581          1398  2 !++
;  582          1399  2 !
;  583          1400  2 ! FUNCTIONAL DESCRIPTION:
;  584          1401  2 !
;  585          1402  2 !       Delete a text module from the library
;  586          1403  2 !
;  587          1404  2 ! INPUT PARAMETERS:
;  588          1405  2 !
;  589          1406  2 !       control_index           Address of valid control index
;  590          1407  2 !       txtrfa                  Pointer to RFA of text to delete
;  591          1408  2 !
;  592          1409  2 ! IMPLICIT OUTPUTS:
;  593          1410  2 !
;  594          1411  2 !       text is deleted
;  595          1412  2 !--
;  596          1413  2
;  597          1414  2 perform (validate_ctl (..control_index));
;  598          1415  2
;  599          1416  3 BEGIN
;  600          1417  3     BIND
;  601          1418  3         context = .lbr$gl_control [lbr$l_ctxptr] : BBLOCK;
;  602          1419  3
;  603          1420  3     IF .context [ctx$v_oldlib]                  !Can't delete in old library
;  604          1421  3         OR .context [ctx$v_ronly]               ! or read only
;  605          1422  3     THEN RETURN lbr$_illop;
;  606          1423  2     END;
;  607          1424  2
;  608          1425  2 perform (delete_data (.txtrfa));
;  609          1426  2 RETURN true;
;  610          1427  1 END;                                           !OF lbr$delete_data


                                        0FFC 00000          .ENTRY  LBR$DELETE_DATA, Save R2,R3,R4,R5,R6,R7,R8,-; 1396
                                                                    R9,R10,R11                                    ;
                            50      04  BC  D0 00002         MOVL    @CONTROL_INDEX, R0                        ; 1414
                                  0000G 30  00006         BSBW    VALIDATE_CTL                                  ;
                            29          50  E9 00009         BLBC    STATUS, 3$                                ;
                            50    0000G CF  D0 0000C         MOVL    LBR$GL_CONTROL, R0                        ; 1418
                            50      0E  A0  D0 00011         MOVL    14(R0), R0                                ;
                     05     04  A0          05  E0 00015         BBS     #5, 4(R0), 1$                             ; 1420
                                        04  A0  95 0001A         TSTB    4(R0)                                     ; 1421
                                            08  18 0001D         BGEQ    2$                                        ;
                            50 00000000G 8F  D0 0001F 1$:       MOVL    #LBR$_ILLOP, R0                           ; 1422
                                            04  00026         RET                                               ;
                                        08  AC  DD 00027 2$:     PUSHL   TXTRFA                                    ; 1425
                          0000V CF          01  FB 0002A         CALLS   #1, DELETE_DATA                           ;
                            03          50  E9 0002F         BLBC    STATUS, 3$                                ;
                            50          01  D0 00032         MOVL    #1, R0                                    ; 1426
                                            04  00035 3$:     RET                                               ; 1427

; Routine Size:  54 bytes,    Routine Base:  $CODE$ + 037E
```

```
;  612          1428  1 %SBTTL  'delete_data';
;  613          1429  1 GLOBAL ROUTINE delete_data (txtrfa) =
;  614          1430  2 BEGIN
;  615          1431  2 !
;  616          1432  2 ! Delete data starting with the given RFA
;  617          1433  2 !
;  618          1434  2
;  619          1435  2 ROUTINE decr_refs (startrfa, endrfa) =
;  620          1436  3 BEGIN
;  621          1437  3 !
;  622          1438  3 ! Local routine to decrement record count for a given vbn.  If
;  623          1439  3 ! record count goes to zero, deallocate the block.
;  624          1440  3 !
;  625          1441  3 MAP
;  626          1442  3     startrfa : REF BBLOCK,
;  627          1443  3     endrfa : REF BBLOCK;
;  628          1444  3
;  629          1445  3
;  630          1446  3 LOCAL
;  631          1447  3     cachentry : REF BBLOCK,
;  632          1448  3     blkadr : REF BBLOCK,
;  633          1449  3     link;
;  634          1450  3
;  635          1451  3 perform (lookup_cache (.startrfa [rfa$l_vbn], cachentry));!Find the block
;  636          1452  3 blkadr = .cachentry [cache$l_address];                    !Point to it
;  637          1453  3 blkadr [data$b_recs] = .blkadr [data$b_recs] - 1;        !Count one less
;  638          1454  3 cachentry [cache$v_dirty] = true;                        !Mark block as dirty
;  639          1455  3 link = .blkadr [data$l_link];                           ! Save link to next block
;  640          1456  3 IF .blkadr [data$b_recs] EQL 0                         ! and if all gone
;  641          1457  3         THEN dealloc_block (.startrfa [rfa$l_vbn]);     ! then deallocate the block
;  642          1458  3
;  643          1459  4 IF (.startrfa [rfa$l_vbn] NEQ .endrfa [rfa$l_vbn])     !If record spans multiple blocks
;  644          1460  3 THEN
;  645          1461  4     IF (.link NEQ .endrfa [rfa$l_vbn])  ! Spans more than two blocks
;  646          1462  3     THEN
;  647          1463  4         BEGIN
;  648          1464  4         LOCAL
;  649          1465  4             start_rfa : BBLOCK [rfa$c_length];
;  650          1466  4         start_rfa [rfa$l_vbn] = .link;
;  651          1467  4         decr_refs (start_rfa, .endrfa);
;  652          1468  4         END
;  653          1469  3     ELSE                                                ! Spans two blocks
;  654          1470  3         IF .endrfa [rfa$w_offset] NEQ data$c_data        ! and does not end at end of previous block
;  655          1471  3         THEN decr_refs (.endrfa, .endrfa);              ! then decrement ref count in ending block
;  656          1472  3 RETURN true;
;  657          1473  2 END;                                                   !Of dec_recs


                                OFFC 00000 DECR_REFS:
                                           .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11        ; 1435
                           5E       0C  C2 00002    SUBL2   #12, SP                            ;
                           51       6E  9E 00005    MOVAB   CACHENTRY, R1                      ; 1451
                           50   04  BC  D0 00008    MOVL    @STARTRFA, R0                      ;
                                0000G 30 0000C     BSBW    LOOKUP_CACHE                       ;
```

```
                                47              50  E9 0000F          BLBC    STATUS, 5$                                      ;
                                50              6E  D0 00012          MOVL    CACHENTRY, R0                                   ; 1452
                                51        08    A0  D0 00015          MOVL    8(R0), BLKADR                                   ;
                                                61  97 00019          DECB    (BLKADR)                                        ; 1453
                           0C   A0              01  88 0001B          BISB2   #1, 12(R0)                                      ; 1454
                                53        02    A1  D0 0001F          MOVL    2(BLKADR), LINK                                 ; 1455
                                                61  95 00023          TSTB    (BLKADR)                                        ; 1456
                                                07  12 00025          BNEQ    1$                                              ;
                                50        04    BC  D0 00027          MOVL    @STARTRFA, R0                                   ; 1457
                                          0000G 30  0002B             BSBW    DEALLOC_BLOCK                                   ;
                                52        08    AC  D0 0002E 1$:      MOVL    ENDRFA, R2                                      ; 1459
                                62        04    BC  D1 00032          CMPL    @STARTRFA, (R2)                                 ;
                                                1E  13 00036          BEQL    4$                                              ;
                                62              53  D1 00038          CMPL    LINK, (R2)                                      ; 1461
                                                0B  13 0003B          BEQL    2$                                              ;
                           04   AE              53  D0 0003D          MOVL    LINK, START_RFA                                 ; 1466
                                                52  DD 00041          PUSHL   R2                                              ; 1467
                                          08    AE  9F 00043          PUSHAB  START_RFA                                       ;
                                                0A  11 00046          BRB     3$                                              ;
                                06        04    A2  B1 00048 2$:      CMPW    4(R2), #6                                       ; 1470
                                                08  13 0004C          BEQL    4$                                              ;
                                                52  DD 0004E          PUSHL   R2                                              ; 1471
                                                52  DD 00050          PUSHL   R2                                              ;
                           AA   AF                  02  FB 00052 3$:      CALLS   #2, DECR_REFS                               ;
                                50                  01  D0 00056 4$:      MOVL    #1, R0                                      ; 1472
                                                        04 00059 5$:      RET                                                 ; 1473

; Routine Size:  90 bytes,    Routine Base:  $CODE$ + 03B4

;  658          1474  2
;  659          1475  2 !
;  660          1476  2 ! Main body of delete_data
;  661          1477  2 !
;  662          1478  2 MAP
;  663          1479  2     txtrfa : REF BBLOCK;
;  664          1480  2 LOCAL
;  665          1481  2     read_status,
;  666          1482  2     localrfa : BBLOCK [rfa$c_length],
;  667          1483  2     recrfa   : BBLOCK [rfa$c_length],
;  668          1484  2     cachentry : REF BBLOCK,
;  669          1485  2     descrip : BBLOCK [dsc$c_s_bln];
;  670          1486  2
;  671          1487  2 BIND
;  672          1488  2     header = .lbr$gl_control [lbr$l_hdrptr] : BBLOCK,
;  673          1489  2     hdrnxtrfa = header [lhd$b_nextrfa] : BBLOCK,
;  674          1490  2     context = .lbr$gl_control [lbr$l_ctxptr] : BBLOCK,
;  675          1491  2     length = descrip [dsc$w_length] : WORD,
;  676          1492  2     addr = descrip [dsc$a_pointer] : REF BBLOCK;
;  677          1493  2
;  678          1494  2 IF .context [ctx$v_oldlib]                          !Can't delete text
;  679          1495  2     THEN RETURN lbr$_illop;                         ! from old library
;  680          1496  2
;  681          1497  2 CH$MOVE (rfa$c_length, .txtrfa, localrfa);
;  682          1498  2 CH$MOVE (rfa$c_length, .txtrfa, recrfa);
;  683          1499  2 perform (read_record (localrfa, descrip));          !read module header
;  684          1500  2 IF .addr [mhd$b_id] NEQ mhd$c_mhdid                !check that it really is
```

```
;  685          1501  2     THEN RETURN lbr$_invrfa;                          !and return error if not
;  686          1502  2 IF .addr [mhd$l_refcnt] NEQ 0                        !There should be no other keys
;  687          1503  2     THEN RETURN lbr$_stillkeys;                       ! still pointing at the data
;  688          1504  2 decr_refs (recrfa, localrfa);                        !Decrement record counts
;  689          1505  2 !
;  690          1506  2 ! Read the text until end, deleteing empty blocks
;  691          1507  2 !
;  692          1508  2 CH$MOVE (rfa$c_length, localrfa, recrfa);            !Save RFA of first data record
;  693          1509  2 WHILE (read_status = read_record (localrfa, descrip)) NEQ rms$_eof
;  694          1510  3 DO BEGIN
;  695          1511  3     IF NOT .read_status THEN RETURN .read_status;     !Avoid looping on read error
;  696          1512  3     decr_refs (recrfa, localrfa);                     !Decrement record counts
;  697          1513  3     CH$MOVE (rfa$c_length, localrfa, recrfa);         !Copy RFA of next record
;  698          1514  2     END;
;  699          1515  2
;  700          1516  2 decr_refs (recrfa, localrfa);                        !Discount end of file record too
;  701          1517  2
;  702          1518  2 header [lhd$l_modhdrs] = .header [lhd$l_modhdrs] - 1;   !One less module header
;  703          1519  2 IF .header [lhd$l_modhdrs] EQL 0                      !If that was the last one,
;  704          1520  3 THEN BEGIN
;  705          1521  3     hdrnxtrfa [rfa$l_vbn] = .header [lhd$l_hipreal] + 1;!Reset next VBN
;  706          1522  3     hdrnxtrfa [rfa$w_offset] = 0;                       !And offset
;  707          1523  2     END;
;  708          1524  2 context [ctx$v_hdrdirty] = true;                      !flag header is dirty
;  709          1525  2 RETURN true
;  710          1526  1 END;                                                   ! Of delete_data

                                    0FFC 00000          .ENTRY  DELETE_DATA, Save R2,R3,R4,R5,R6,R7,R8,R9,- ; 1429
                                                                R10,R11
                                5E      18  C2 00002            SUBL2   #24, SP                             ;
                                50  0000G   CF  D0 00005            MOVL    LBR$GL_CONTROL, R0              ; 1488
                                56    0A    A0  7D 0000A            MOVQ    10(R0), R6                      ;
                                59    4C    A6  9E 0000E            MOVAB   76(R6), R9                      ; 1489
                          08    04  A7      05  E1 00012            BBC     #5, 4(R7), 1$                   ; 1494
                                50 00000000G  8F  D0 00017            MOVL    #LBR$_ILLOP, R0                 ; 1495
                                            04  00 0001E            RET                                     ;
                      10  AE    04  BC      06  28 0001F 1$:        MOVC3   #6, @TXTRFA, LOCALRFA           ; 1497
                      08  AE    04  BC      06  28 00025            MOVC3   #6, @TXTRFA, RECRFA             ; 1498
                                51          6E  9E 0002B            MOVAB   DESCRIP, R1                     ; 1499
                                50    10    AE  9E 0002E            MOVAB   LOCALRFA, R0                    ;
                                      0000V 30  00032            BSBW    READ_RECORD                     ;
                                6D          50  E9 00035            BLBC    STATUS, 6$                      ;
                                50    04    AE  D0 00038            MOVL    ADDR, R0                        ; 1500
                          AD    8F    01    A0  91 0003C            CMPB    1(R0), #173                     ;
                                            08  13 00041            BEQL    2$                              ;
                                50 00000000G  8F  D0 00043            MOVL    #LBR$_INVRFA, R0                ; 1501
                                            04  00 0004A            RET                                     ;
                                      04    A0  D5 0004B 2$:        TSTL    4(R0)                           ; 1502
                                            08  13 0004E            BEQL    3$                              ;
                                50 00000000G  8F  D0 00050            MOVL    #LBR$_STILLKEYS, R0             ; 1503
                                            04  00 00057            RET                                     ;
                                      10    AE  9F 00058 3$:        PUSHAB  LOCALRFA                        ; 1504
                                      0C    AE  9F 0005B            PUSHAB  RECRFA                          ;
```

```
                    FF43   CF          02  FB 0005E        CALLS   #2, DECR_REFS                       ;
          08   AE     10   AE          06  28 00063        MOVC3   #6, LOCALRFA, RECRFA                ; 1508
                           51          6E  9E 00069        MOVAB   DESCRIP, R1                         ; 1509
                           50      10  AE  9E 0006C        MOVAB   LOCALRFA, R0                        ;
                                    0000V  30 00070        BSBW    READ_RECORD                         ;
                           58          50  D0 00073        MOVL    R0, READ_STATUS                     ;
                0001827A   8F          58  D1 00076        CMPL    READ_STATUS, #98938                 ;
                                       07  13 0007D        BEQL    4$                                  ;
                           D6          58  E8 0007F        BLBS    READ_STATUS, 3$                     ; 1511
                           50          58  D0 00082        MOVL    READ_STATUS, R0                     ;
                                           04 00085        RET                                         ;
                                   10  AE  9F 00086 4$:    PUSHAB  LOCALRFA                            ; 1516
                                   0C  AE  9F 00089        PUSHAB  RECRFA                              ;
                    FF15   CF          02  FB 0008C        CALLS   #2, DECR_REFS                       ;
                                   74  A6  D7 00091        DECL    116(R6)                             ; 1518
                                       08  12 00094        BNEQ    5$                                  ; 1519
               69     5E   A6          01  C1 00096        ADDL3   #1, 94(R6), (R9)                    ; 1521
                                   04  A9  B4 0009B        CLRW    4(R9)                               ; 1522
                      04   A7          08  88 0009E 5$:    BISB2   #8, 4(R7)                           ; 1524
                           50          01  D0 000A2        MOVL    #1, R0                              ; 1525
                                           04 000A5 6$:    RET                                         ; 1526

; Routine Size:  166 bytes,    Routine Base:  $CODE$ + 040E
```

```
;  712          1527  1 %SBTTL 'write_record';
;  713          1528  1 GLOBAL ROUTINE write_record (bytcnt, addr, writerfa, rewrite, retrfa) =
;  714          1529  2 BEGIN
;  715          1530  2 !
;  716          1531  2 ! This routine does the actual output to the library
;  717          1532  2 ! Inputs:
;  718          1533  2 !
;  719          1534  2 !       bytcnt = Number of bytes in record
;  720          1535  2 !       addr = Address of record
;  721          1536  2 !       writerfa = RFA to store record in file
;  722          1537  2 !       rewrite = true if rewriting previous record
;  723          1538  2 !       retrfa (optional) = Address to receieve RFA of record
;  724          1539  2 !                           (the requested RFA may be modified)
;  725          1540  2 !
;  726          1541  2 ROUTINE next_block (lastblkadr, rfa, rewrite, newblkadr) =
;  727          1542  3 BEGIN
;  728          1543  3 !
;  729          1544  3 ! Local routine to map the next block into memory and
;  730          1545  3 ! handle the links.
;  731          1546  3 !
;  732          1547  3 MAP
;  733          1548  3     lastblkadr : REF BBLOCK,
;  734          1549  3     rfa : REF BBLOCK,
;  735          1550  3     newblkadr : REF BBLOCK;
;  736          1551  3
;  737          1552  3 LOCAL
;  738          1553  3     newblock : REF BBLOCK,
;  739          1554  3     cachentry : REF BBLOCK;
;  740          1555  3
;  741          1556  3 IF .rewrite                                  !If rewriting the record
;  742          1557  4 THEN BEGIN
;  743          1558  4     rfa [rfa$l_vbn] = .lastblkadr [data$l_link];!link to next block
;  744          1559  4     rfa [rfa$w_offset] = data$c_data;
;  745          1560  3     END;
;  746          1561  3 update_nextrfa (.rfa);                      !Update next RFA
;  747        P 1562  3 perform (map_blk_to_mem (.rfa, .rewrite, .newblkadr, !Bring block into memory
;  748          1563  3                        cachentry));
;  749          1564  3 newblock = ..newblkadr;                     !Get memory address
;  750          1565  3 IF NOT .rewrite                             !If writing (not rewriting)
;  751          1566  3 THEN newblock [data$b_recs] = 1;           ! then this is first record in block
;  752          1567  3 update_nextrfa (.rfa);                     !Update next RFA (map_blk_to_mem
;  753          1568  3                                           ! may modify RFA if needed)
;  754          1569  3 cachentry [cache$v_dirty] = true;          !Mark block as dirty
;  755          1570  3 IF NOT .rewrite                             !Unless rewriting the block
;  756          1571  3 THEN lastblkadr [data$l_link] = .cachentry [cache$l_vbn];!Then set the link in last block
;  757          1572  3 RETURN true
;  758          1573  2 END;                                       !Of next_block


                                OFFC 00000 NEXT_BLOCK:
                                           .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11        ; 1541
                     5E                04  C2 00002              SUBL2   #4, SP                      ;
                     0C          0C    AC  E9 00005              BLBC    REWRITE, 1$                 ; 1556
                     50          04    AC  7D 00009              MOVQ    LASTBLKADR, R0              ; 1558
```

```
               61      02  A0  D0 0000D       MOVL    2(R0), (R1)                      ;
          04   A1          06  B0 00011       MOVW    #6, 4(R1)                        ; 1559
               50      08  AC  D0 00015 1$:   MOVL    RFA, R0                          ; 1561
                      0000V  30 00019         BSBW    UPDATE_NEXTRFA                   ;
                          5E  DD 0001C        PUSHL   SP                               ; 1563
               7E      0C  AC  7D 0001E       MOVQ    REWRITE, -(SP)                   ;
                       08  AC  DD 00022       PUSHL   RFA                              ;
       0000V   CF          04  FB 00025       CALLS   #4, MAP_BLK_TO_MEM               ;
               29          50  E9 0002A       BLBC    STATUS, 4$                       ;
               50      10  BC  D0 0002D       MOVL    @NEWBLKADR, NEWBLOCK             ; 1564
               03      0C  AC  E8 00031       BLBS    REWRITE, 2$                      ; 1565
               60          01  90 00035       MOVB    #1, (NEWBLOCK)                   ; 1566
               50      08  AC  D0 00038 2$:   MOVL    RFA, R0                          ; 1567
                      0000V  30 0003C         BSBW    UPDATE_NEXTRFA                   ;
               51          6E  D0 0003F       MOVL    CACHENTRY, R1                    ; 1569
          0C   A1          01  88 00042       BISB2   #1, 12(R1)                       ;
               09      0C  AC  E8 00046       BLBS    REWRITE, 3$                      ; 1570
               50      04  AC  D0 0004A       MOVL    LASTBLKADR, R0                   ; 1571
          02   A0      04  A1  D0 0004E       MOVL    4(R1), 2(R0)                     ;
               50          01  D0 00053 3$:   MOVL    #1, R0                           ; 1572
                           04 00056 4$:       RET                                      ; 1573

; Routine Size:  87 bytes,    Routine Base:  $CODE$ + 04B4

;  2 !
;  2 ! Main body of write_record
;  2 !
;  2 MAP
;  2     writerfa : REF BBLOCK;          !Pointer to RFA to write at
;  2 LOCAL
;  2     bytes,
;  2     blkadr : REF BBLOCK,                    !Pointer to disk block in memory
;  2     movecount,
;  2     cachentry : REF BBLOCK,
;  2     bufptr;
;  2
;  2 BIND
;  2     blkvector = blkadr : REF VECTOR [,BYTE],
;  2     header = .lbr$gl_control [lbr$l_hdrptr] : BBLOCK, !point to the header
;  2     hdrnxtrfa = header [lhd$b_nextrfa] : BBLOCK,      !name next RFA part
;  2     context = .lbr$gl_control [lbr$l_ctxptr] : BBLOCK;
;  2
;  2 BUILTIN
;  2     NULLPARAMETER;                  ! True if parameter not specified
;  2
;  2 bytes = .bytcnt;                            !and the byte count
;  2 bufptr = .addr;                             !Point to the data buffer
;  2 IF .writerfa [rfa$l_vbn] GTRU .hdrnxtrfa [rfa$l_vbn] !Check for illegal vbn request
;  2     THEN RETURN lbr$_rfapasteof;
;  2 perform (map_blk_to_mem (.writerfa, .rewrite, blkadr, cachentry)); !Map block
;  2 cachentry [cache$v_dirty] = true;         !Mark block as dirty
;  2 IF NOT .rewrite                             !Unless rewriting the record
;  2 THEN blkadr [data$b_recs] = .blkadr [data$b_recs] + 1; ! then count another record in block
;  2 update_nextrfa (.writerfa);                 !Update next RFA
;  2
;  3 DO BEGIN
```

```
;  791          1606  3
;  792          1607  3     IF .bytes EQL .bytcnt                    !If this is first time in here
;  793          1608  4     THEN BEGIN                               !then we need to set the byte count
;  794          1609  4         IF .writerfa [rfa$w_offset] EQL 0        !If just went to new page
;  795          1610  4         THEN perform (next_block (.blkadr, .writerfa, .rewrite, blkadr)); ! then get next block in
;  796          1611  4         IF NOT NULLPARAMETER (5)                 ! If retrfa specified,
;  797          1612  4         THEN                                     ! then return to caller
;  798          1613  4             CH$MOVE (rfa$c_length, .writerfa, .retrfa);
;  799          1614  5         BEGIN
;  800          1615  5         BIND
;  801          1616  5             bytecount = blkvector [.writerfa [rfa$w_offset]] : WORD; !Name the spot where it goes
;  802          1617  5         bytecount = .bytcnt;                     !Set the byte count
;  803          1618  4         END;
;  804          1619  4         incr_rfa (2, .writerfa);                 !Bump the RFA
;  805          1620  4         update_nextrfa (.writerfa);              !Update next RFA
;  806          1621  4         IF .writerfa [rfa$w_offset] EQL 0        !gone to new block?
;  807          1622  4         THEN perform (next_block (.blkadr, .writerfa, .rewrite, blkadr)); !yes--bring in the block
;  808          1623  3         END;                                     !bytes eql bytcnt
;  809          1624  3     movecount = MINU (.bytes, data$c_length - .writerfa [rfa$w_offset]); !Figure length of move
;  810          1625  3     CH$MOVE (.movecount, .bufptr, blkvector [.writerfa [rfa$w_offset]]); !and move it in
;  811          1626  3     incr_rfa (.movecount, .writerfa);            !increment RFA
;  812          1627  3     update_nextrfa (.writerfa);                  !Update next RFA
;  813          1628  3     bufptr = .bufptr + .movecount;               !update the pointer
;  814          1629  3     bytes = .bytes - .movecount;                 !and bytes to go
;  815          1630  3     IF .writerfa [rfa$w_offset] EQL 0            !going to new page?
;  816          1631  4         THEN BEGIN
;  817        P 1632  4             perform (next_block (.blkadr,        !yes--bring next page in
;  818          1633  4                 .writerfa, .rewrite, blkadr));
;  819          1634  4             IF .bytes EQL 0                      !However, if done with record
;  820          1635  4                 AND NOT .rewrite                 ! and not rewriting record
;  821          1636  4                 THEN blkadr [data$b_recs] = 0;   ! then really no records in there yet
;  822          1637  3             END;
;  823          1638  3 END                                              !End of repeat loop
;  824          1639  2 UNTIL .bytes EQL 0;                              !End of repeat loop
;  825          1640  2
;  826          1641  2 RETURN true
;  827          1642  1 END;                                  !Of write_record


                                    0FFC 00000          .ENTRY  WRITE_RECORD, Save R2,R3,R4,R5,R6,R7,R8,R9,-; 1528
                                                                R10,R11                                      ;
                              5B      0000V CF  9E 00002        MOVAB   UPDATE_NEXTRFA, R11                  ;
                              5E            08  C2 00007        SUBL2   #8, SP                               ;
                              50      0000G CF  D0 0000A        MOVL    LBR$GL_CONTROL, R0                   ; 1588
                    51        0A    A0 0000004C 8F  C1 0000F  ADDL3   #76, 10(R0), R1                     ; 1589
                              56        04  AC  D0 00018        MOVL    BYTCNT, BYTES                        ; 1595
                              5A        08  AC  D0 0001C        MOVL    ADDR, BUFPTR                         ; 1596
                              57        0C  AC  D0 00020        MOVL    WRITERFA, R7                         ; 1597
                              61            67  D1 00024        CMPL    (R7), (R1)                           ;
                                            08  1B 00027        BLEQU   1$                                   ;
                              50 00000000G  8F  D0 00029        MOVL    #LBR$_RFAPASTEOF, R0                 ; 1598
                                            04  00030        RET                                          ;
                                        5E  DD 00031 1$:    PUSHL   SP                                   ; 1599
                                    08  AE  9F 00033        PUSHAB  BLKADR                               ;
```

```
                  59       10   AC   D0 00036       MOVL    REWRITE, R9                      ;
                         0280   8F   BB 0003A       PUSHR   #^M<R7,R9>                       ;
          0000V   CF       04   FB 0003E            CALLS   #4, MAP_BLK_TO_MEM               ;
                  6C       50   E9 00043            BLBC    STATUS, 6$                       ;
                  50       6E   D0 00046            MOVL    CACHENTRY, R0                    ; 1600
             0C   A0       01   88 00049            BISB2   #1, 12(R0)                       ;
                  03       59   E8 0004D            BLBS    R9, 2$                           ; 1601
                         04   BE   96 00050         INCB    @BLKADR                          ; 1602
                  50       57   D0 00053  2$:       MOVL    R7, R0                           ; 1603
                           6B   16 00056            JSB     UPDATE_NEXTRFA                   ;
             04   AC       56   D1 00058  3$:       CMPL    BYTES, BYTCNT                    ; 1607
                           57   12 0005C            BNEQ    7$                               ;
                         04   A7   B5 0005E         TSTW    4(R7)                            ; 1609
                           12   12 00061            BNEQ    4$                               ;
                         04   AE   9F 00063         PUSHAB  BLKADR                           ; 1610
                       0280   8F   BB 00066         PUSHR   #^M<R7,R9>                       ;
                         10   AE   DD 0006A         PUSHL   BLKADR                           ;
          FF37    CF       04   FB 0006D            CALLS   #4, NEXT_BLOCK                   ;
                  3D       50   E9 00072            BLBC    STATUS, 6$                       ;
                  05       6C   91 00075  4$:       CMPB    (AP), #5                         ; 1611
                           0A   1F 00078            BLSSU   5$                               ;
                         14   AC   D5 0007A         TSTL    20(AP)                           ;
                           05   13 0007D            BEQL    5$                               ;
    14   BC       67       06   28 0007F            MOVC3   #6, (R7), @RETRFA                ; 1613
                  50  04   A7   3C 00084  5$:       MOVZWL  4(R7), R0                        ; 1616
                  50  04   AE   C0 00088            ADDL2   BLKVECTOR, R0                    ;
                  60  04   AC   B0 0008C            MOVW    BYTCNT, (R0)                     ; 1617
                  51       57   D0 00090            MOVL    R7, R1                           ; 1619
                  50       02   D0 00093            MOVL    #2, R0                           ;
                      0000G   30 00096              BSBW    INCR_RFA                         ;
                  50       57   D0 00099            MOVL    R7, R0                           ; 1620
                           6B   16 0009C            JSB     UPDATE_NEXTRFA                   ;
                         04   A7   B5 0009E         TSTW    4(R7)                            ; 1621
                           12   12 000A1            BNEQ    7$                               ;
                         04   AE   9F 000A3         PUSHAB  BLKADR                           ; 1622
                       0280   8F   BB 000A6         PUSHR   #^M<R7,R9>                       ;
                         10   AE   DD 000AA         PUSHL   BLKVECTOR                        ;
          FEF7    CF       04   FB 000AD            CALLS   #4, NEXT_BLOCK                   ;
                  65       50   E9 000B2  6$:       BLBC    STATUS, 11$                      ;
                  51  04   A7   3C 000B5  7$:       MOVZWL  4(R7), R1                        ; 1624
     51 00000200  8F       51   C3 000B9            SUBL3   R1, #512, R1                     ;
                  50       56   D0 000C1            MOVL    BYTES, R0                        ;
                  51       50   D1 000C4            CMPL    R0, R1                           ;
                           03   1B 000C7            BLEQU   8$                               ;
                  50       51   D0 000C9            MOVL    R1, R0                           ;
                  58       50   D0 000CC  8$:       MOVL    R0, MOVECOUNT                    ;
                  50  04   A7   3C 000CF            MOVZWL  4(R7), R0                        ; 1625
                  50  04   AE   C0 000D3            ADDL2   BLKVECTOR, R0                    ;
         60       6A       58   28 000D7            MOVC3   MOVECOUNT, (BUFPTR), (R0)        ;
                  51       57   D0 000DB            MOVL    R7, R1                           ; 1626
                  50       58   D0 000DE            MOVL    MOVECOUNT, R0                    ;
                      0000G   30 000E1              BSBW    INCR_RFA                         ;
                  50       57   D0 000E4            MOVL    R7, R0                           ; 1627
                           6B   16 000E7            JSB     UPDATE_NEXTRFA                   ;
                  5A       58   C0 000E9            ADDL2   MOVECOUNT, BUFPTR                ; 1628
                  56       58   C2 000EC            SUBL2   MOVECOUNT, BYTES                 ; 1629
                         04   A7   B5 000EF         TSTW    4(R7)                            ; 1630
```

```
                                 1C  12 000F2          BNEQ    9$                                                      ;
                              04 AE  9F 000F4          PUSHAB  BLKADR                                                  ; 1633
                            0280 8F  BB 000F7          PUSHR   #^M<R7,R9>                                              ;
                              10 AE  DD 000FB          PUSHL   BLKVECTOR                                               ;
                       FEA6   CF     04 FB 000FE       CALLS   #4, NEXT_BLOCK                                          ;
                              14     50 E9 00103       BLBC    STATUS, T1$                                             ;
                                     56 D5 00106       TSTL    BYTES                                                   ; 1634
                                     06 12 00108       BNEQ    9$                                                      ;
                              03     59 E8 0010A       BLBS    R9, 9$                                                  ; 1635
                                 04  BE 94 0010D       CLRB    @BLKADR                                                 ; 1636
                                     56 D5 00110 9$:   TSTL    BYTES                                                   ; 1639
                                     03 13 00112       BEQL    10$                                                     ;
                                 FF41 31 00114         BRW     3$                                                      ;
                              50     01 D0 00117 10$:  MOVL    #1, R0                                                  ; 1641
                                        04 0011A 11$:  RET                                                           ; 1642

; Routine Size:  283 bytes,    Routine Base:  $CODE$ + 050B
```

```
;   829     1643  1 %SBTTL  'read_record';
;   830     1644  1 GLOBAL ROUTINE read_record (readrfa, descrip) : JSB_2 =
;   831     1645  2 BEGIN
;   832     1646  2 !++
;   833     1647  2 ! This routine does the actual input from the library
;   834     1648  2 !
;   835     1649  2 ! Inputs:
;   836     1650  2 !
;   837     1651  2 !       readrfa         Address of RFA to read from
;   838     1652  2 !       descrip         address of string descriptor to return record description
;   839     1653  2 !
;   840     1654  2 ! Outputs:
;   841     1655  2 !
;   842     1656  2 !       record is read, descriptor returned in descrip
;   843     1657  2 !       readrfa is updated
;   844     1658  2 !
;   845     1659  2 !--
;   846     1660  2
;   847     1661  2 MAP
;   848     1662  2     readrfa : REF BBLOCK,
;   849     1663  2     descrip : REF BBLOCK;
;   850     1664  2
;   851     1665  2 LOCAL
;   852     1666  2     blkadr : REF BBLOCK,                        !Pointer to disk block in memory
;   853     1667  2     cachentry : REF BBLOCK,             !Pointer to cache entry for block
;   854     1668  2     movecount,
;   855     1669  2     bytcnt,
;   856     1670  2     bufptr;
;   857     1671  2
;   858     1672  2 BIND
;   859     1673  2     blkvector = blkadr : REF VECTOR [,BYTE],
;   860     1674  2     context = lbr$gl_control [lbr$l_ctxptr] : REF BBLOCK;
;   861     1675  2
;   862     1676  2 perform (map_blk_to_mem (.readrfa, true, blkadr, cachentry));
;   863     1677  2 IF .readrfa [rfa$w_offset] EQL 0                                    !Starting new block?
;   864     1678  2     THEN readrfa [rfa$w_offset] = data$c_data;                      !start at top of block
;   865     1679  2
;   866     1680  3     BEGIN
;   867     1681  3     BIND
;   868     1682  3         header = .lbr$gl_control[lbr$l_hdrptr]: BLOCK [, BYTE],
;   869     1683  3         bytecount = blkvector [.readrfa [rfa$w_offset]] : WORD; !Name bytecount
;   870     1684  3     LOCAL
;   871     1685  3         maxrecsiz;                                  ! Maximum record size.
;   872     1686  3     descrip [dsc$w_length] = .bytecount;            ! Return byte count to caller.
;   873     1687  3     IF .header[lhd$l_dcxmapvbn] EQL 0 THEN          ! If not a DCX library
;   874     1688  3         maxrecsiz = lbr$c_maxrecsiz                 !   use normal maxrecsize,
;   875     1689  3     ELSE                                            ! if DCX
;   876     1690  3         maxrecsiz = lbr_dcx$c_maxrecsiz;            !   use larger value.
;   877     1691  3     IF .bytecount GTRU .maxrecsiz                   !Make sure it's really a record
;   878     1692  3         THEN RETURN lbr$_invrfa;                    ! and return error if not
;   879     1693  3     IF .bytecount+.readrfa [rfa$w_offset] + 2 LEQU data$c_length !If record on one block
;   880     1694  4     THEN BEGIN
;   881     1695  4         descrip [dsc$a_pointer] = blkvector [.readrfa [rfa$w_offset]] + 2; !return the address
;   882     1696  4         incr_rfa (.descrip [dsc$w_length] + 2, .readrfa);       !increment RFA
;   883     1697  4         IF .readrfa [rfa$w_offset] EQL 0                !If went to next block
;   884     1698  5         THEN BEGIN
;   885     1699  5             readrfa [rfa$l_vbn] = .blkadr [data$l_link]; !Link to next block
```

```
;  886          1700  5            readrfa [rfa$w_offset] = data$c_data;
;  887          1701  4            END;
;  888          1702  4        END
;  889          1703  4    !
;  890          1704  4    ! Record is split across multiple blocks
;  891          1705  4    !
;  892          1706  4      ELSE BEGIN
;  893          1707  4          incr_rfa (2, .readrfa);                 !skip the byte count
;  894          1708  4          IF .readrfa [rfa$w_offset] EQL 0        ! and if went to new block
;  895          1709  5          THEN BEGIN
;  896          1710  5              readrfa [rfa$l_vbn] = .blkadr [data$l_link];    !Link to next block
;  897          1711  5              readrfa [rfa$w_offset] = data$c_data;
;  898          1712  4              END;
;  899          1713  4
;  900          1714  4          IF .context [ctx$l_readbuf] EQL 0               !If no buffer allocated
;  901          1715  4          THEN perform (get_mem (.maxrecsiz, context [ctx$l_readbuf]));
;  902          1716  4          descrip [dsc$a_pointer] = .context [ctx$l_readbuf];     !Return address to caller
;  903          1717  4          bufptr = .context [ctx$l_readbuf];      !Init buffer pointer
;  904          1718  4          bytcnt = .bytecount;                    !Set up byte count
;  905          1719  5          DO BEGIN                                !Read whole record into buffer
;  906          1720  5              perform (map_blk_to_mem (.readrfa, true, blkadr, cachentry)); !Map into memory
;  907          1721  5              movecount = MINU (.bytcnt, data$c_length - .readrfa [rfa$w_offset]); !Compute length of move
;  908          1722  5              bufptr = CH$MOVE (.movecount,
;  909          1723  5                  blkvector [.readrfa [rfa$w_offset]], .bufptr); !Copy partial record
;  910          1724  5              bytcnt = .bytcnt - .movecount;      !Update bytes left to go
;  911          1725  5              incr_rfa (.movecount, .readrfa);    !Update RFA
;  912          1726  5              IF .readrfa [rfa$w_offset] EQL 0    !If went to new page
;  913          1727  6                  THEN BEGIN
;  914          1728  6                      readrfa [rfa$l_vbn] = .blkadr [data$l_link]; !next block
;  915          1729  6                      readrfa [rfa$w_offset] = data$c_data;
;  916          1730  5                      END;
;  917          1731  5                  END
;  918          1732  4              UNTIL .bytcnt EQL 0;
;  919          1733  3              END;
;  920          1734  2        END;
;  921          1735  2    !
;  922          1736  2    ! Check to see if this is the end of text record, and return
;  923          1737  2    ! rms$_eof if so.
;  924          1738  2    !
;  925          1739  2
;  926          1740  2    IF .descrip [dsc$w_length] EQL .lbr$gt_eotdesc [0]              !If the length is correct
;  927          1741  2    AND CH$EQL  (.descrip [dsc$w_length], .descrip [dsc$a_pointer], ! and its an eof record
;  928          1742  2                .lbr$gt_eotdesc [0], lbr$gt_eotdesc [1])        !
;  929          1743  2    THEN RETURN rms$_eof                                        !then it is end of file
;  930          1744  2    ELSE RETURN true                                            !otherwise return good record
;  931          1745  1    END;                            ! Of read_record


                                0FFC    8F  BB 00000 READ_RECORD::
                                                      PUSHR   #^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>        ; 1644
                              5E            08  C2 00004       SUBL2   #8, SP                            ;
                              57            51  D0 00007       MOVL    R1, R7                            ;
                              5A            50  D0 0000A       MOVL    R0, R10                           ;
                        55    0000G   CF      0E  C1 0000D       ADDL3   #14, LBR$GL_CONTROL, R5           ; 1674
```

```
                                5E  DD 00013        PUSHL   SP                                  ; 1676
                         08     AE  9F 00015        PUSHAB  BLKADR                              ;
                                01  DD 00018        PUSHL   #1                                  ;
                                5A  DD 0001A        PUSHL   READRFA                             ;
                0000V   CF      04  FB 0001C        CALLS   #4, MAP_BLK_TO_MEM                  ;
                        42      50  E9 00021        BLBC    STATUS, 4$                          ;
                        58  04  AA  9E 00024        MOVAB   4(READRFA), R8                      ; 1677
                                68  B5 00028        TSTW    (R8)                                ;
                                03  12 0002A        BNEQ    1$                                  ;
                        68      06  B0 0002C        MOVW    #6, (R8)                            ; 1678
                        50  0000G  CF  D0 0002F 1$: MOVL    LBR$GL_CONTROL, R0                  ; 1682
                        51  0A  A0  D0 00034        MOVL    10(R0), R1                          ;
                        52  04  AE  D0 00038        MOVL    BLKVECTOR, R2                       ; 1683
                        50      68  3C 0003C        MOVZWL  (R8), R0                            ;
               54       50      52  C1 0003F        ADDL3   R2, R0, R4                          ;
                        67      64  B0 00043        MOVW    (R4), (DESCRIP)                     ; 1686
                           008C C1  D5 00046        TSTL    140(R1)                             ; 1687
                                07  12 0004A        BNEQ    2$                                  ;
                        53  0800 8F 3C 0004C        MOVZWL  #2048, MAXRECSIZ                    ; 1688
                                05  11 00051        BRB     3$                                  ;
                        53  1000 8F 3C 00053 2$:    MOVZWL  #4096, MAXRECSIZ                    ; 1690
       53      64       10      00  ED 00058 3$:    CMPZV   #0, #16, (R4), MAXRECSIZ            ; 1691
                                0A  1B 0005D        BLEQU   5$                                  ;
                        50 00000000G 8F D0 0005F    MOVL    #LBR$_INVRFA, R0                    ; 1692
                           00E1 31 00066 4$:        BRW     15$                                 ;
                        51      64  3C 00069 5$:    MOVZWL  (R4), R1                            ; 1693
                        56      68  3C 0006C        MOVZWL  (R8), R6                            ;
                        51      56  C0 0006F        ADDL2   R6, R1                              ;
                        51      02  C0 00072        ADDL2   #2, R1                              ;
            00000200    8F      51  D1 00075        CMPL    R1, #512                            ;
                                20  1A 0007C        BGTRU   7$                                  ;
               04       A7   02 A240 9E 0007E       MOVAB   2(R2)[R0], 4(DESCRIP)               ; 1695
                        50      67  3C 00084        MOVZWL  (DESCRIP), R0                       ; 1696
                        50      02  C0 00087        ADDL2   #2, R0                              ;
                        51      5A  D0 0008A        MOVL    READRFA, R1                         ;
                           0000G 30 0008D           BSBW    INCR_RFA                            ;
                                68  B5 00090        TSTW    (R8)                                ; 1697
                                07  12 00092        BNEQ    6$                                  ;
                        6A  02  A2  D0 00094        MOVL    2(R2), (READRFA)                    ; 1699
                        68      06  B0 00098        MOVW    #6, (R8)                            ; 1700
                           0086 31 0009B 6$:        BRW     13$                                 ; 1693
                        51      5A  D0 0009E 7$:    MOVL    READRFA, R1                         ; 1707
                        50      02  D0 000A1        MOVL    #2, R0                              ;
                           0000G 30 000A4           BSBW    INCR_RFA                            ;
                                68  B5 000A7        TSTW    (R8)                                ; 1708
                                07  12 000A9        BNEQ    8$                                  ;
                        6A  02  A2  D0 000AB        MOVL    2(R2), (READRFA)                    ; 1710
                        68      06  B0 000AF        MOVW    #6, (R8)                            ; 1711
                        50      65  D0 000B2 8$:    MOVL    (R5), R0                            ; 1714
                        52  2E  A0  9E 000B5        MOVAB   46(R0), R2                          ;
                                62  D5 000B9        TSTL    (R2)                                ;
                                0C  12 000BB        BNEQ    9$                                  ;
                        51      52  D0 000BD        MOVL    R2, R1                              ; 1715
                        50      53  D0 000C0        MOVL    MAXRECSIZ, R0                       ;
                           0000G 30 000C3           BSBW    GET_MEM                             ;
                        9D      50  E9 000C6        BLBC    STATUS, 4$                          ;
               04       A7      62  D0 000C9 9$:    MOVL    (R2), 4(DESCRIP)                    ; 1716
```

```
                                53              62  D0 000CD          MOVL    (R2), BUFPTR                                    ; 1717
                                56              64  3C 000D0          MOVZWL  (R4), BYTCNT                                    ; 1718
                                                5E  DD 000D3 10$:     PUSHL   SP                                              ; 1720
                                          08    AE  9F 000D5          PUSHAB  BLKADR                                          ;
                                                01  DD 000D8          PUSHL   #1                                              ;
                                                5A  DD 000DA          PUSHL   READRFA                                         ;
                                     0000V      CF  04 FB 000DC       CALLS   #4, MAP_BLK_TO_MEM                              ;
                                                66  50 E9 000E1       BLBC    STATUS, 15$                                     ;
                                                51  68 3C 000E4       MOVZWL  (R8), R1                                        ; 1721
                                51 00000200     8F  51 C3 000E7       SUBL3   R1, #512, R1                                    ;
                                                50  56 D0 000EF       MOVL    BYTCNT, R0                                      ;
                                                51  50 D1 000F2       CMPL    R0, R1                                          ;
                                                    03 1B 000F5       BLEQU   11$                                             ;
                                                50  51 D0 000F7       MOVL    R1, R0                                          ;
                                                5B  50 D0 000FA 11$:  MOVL    R0, MOVECOUNT                                   ;
                                                59  04 AE D0 000FD    MOVL    BLKVECTOR, R9                                   ; 1723
                                                50  68 3C 00101       MOVZWL  (R8), R0                                        ;
                                63         6940     5B 28 00104       MOVC3   MOVECOUNT, (R9)[R0], (BUFPTR)                   ;
                                                56  5B C2 00109       SUBL2   MOVECOUNT, BYTCNT                               ; 1724
                                                51  5A D0 0010C       MOVL    READRFA, R1                                     ; 1725
                                                50  5B D0 0010F       MOVL    MOVECOUNT, R0                                   ;
                                                  0000G 30 00112       BSBW    INCR_RFA                                        ;
                                                    68 B5 00115       TSTW    (R8)                                            ; 1726
                                                    07 12 00117       BNEQ    12$                                             ;
                                                6A  02 A9 D0 00119       MOVL    2(R9), (READRFA)                                ; 1728
                                                68      06 B0 0011D       MOVW    #6, (R8)                                        ; 1729
                                                        56 D5 00120 12$:  TSTL    BYTCNT                                          ; 1732
                                                        AF 12 00122       BNEQ    10$                                             ;
                                                50      0000G CF 9A 00124 13$:  MOVZBL  LBR$GT_EOTDESC, R0                      ; 1740
                                                67          50 B1 00129       CMPW    R0, (DESCRIP)                                   ;
                                                            19 12 0012C       BNEQ    14$                                             ;
                                                50      0000G CF 9A 0012E       MOVZBL  LBR$GT_EOTDESC, R0                      ; 1742
                50              00      04  B7              67 2D 00133       CMPC5   (DESCRIP), @4(DESCRIP), #0, R0, -               ;
                                                0000G CF        00139             LBR$GT_EOTDESC+1                                ;
                                                            09 12 0013C       BNEQ    14$                                             ;
                                                50 0001827A 8F  D0 0013E       MOVL    #98938, R0                              ; 1744
                                                            03 11 00145       BRB     15$                                             ;
                                                50              01 D0 00147 14$:  MOVL    #1, R0                                          ;
                                                5E              08 C0 0014A 15$:  ADDL2   #8, SP                                  ; 1745
                                                    0FFC    8F  BA 0014D       POPR    #^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>            ;
                                                                05 00151       RSB                                             ;

; Routine Size:  338 bytes,    Routine Base:  $CODE$ + 0626
```

```
;  933          1746  1 %SBTTL  'read_old_record';
;  934          1747  1 GLOBAL ROUTINE read_old_record (readrfa, descrip) : JSB_2 =
;  935          1748  2 BEGIN
;  936          1749  2 !++
;  937          1750  2 ! This routine does the actual input from the library for old format libraries
;  938          1751  2 !
;  939          1752  2 ! Inputs:
;  940          1753  2 !
;  941          1754  2 !       readrfa         Address of RFA to start reading from
;  942          1755  2 !       descrip         Address of string descriptor to fill in
;  943          1756  2 !
;  944          1757  2 ! Outputs:
;  945          1758  2 !
;  946          1759  2 !       Record is read, descrip filled in, readrfa updated
;  947          1760  2 !
;  948          1761  2 !--
;  949          1762  2
;  950          1763  2 MAP
;  951          1764  2     readrfa : REF BBLOCK,
;  952          1765  2     descrip : REF BBLOCK;
;  953          1766  2
;  954          1767  2 LOCAL
;  955          1768  2     blkadr : REF VECTOR [,BYTE],                    !Pointer to disk block in memory
;  956          1769  2     cachentry : REF BBLOCK,             !Pointer to cache entry for block
;  957          1770  2     movecount,
;  958          1771  2     bytcnt,
;  959          1772  2     bufptr;
;  960          1773  2
;  961          1774  2 LITERAL
;  962          1775  2      bsize = 2;
;  963          1776  2
;  964          1777  2 BIND
;  965          1778  2     context = .lbr$gl_control [lbr$l_ctxptr] : BBLOCK,
;  966          1779  2     eofrfa = context [ctx$b_eomodrfa] : BBLOCK;
;  967          1780  2
;  968          1781  2 !
;  969          1782  2 ! Check for end of module
;  970          1783  2 !
;  971          1784  2 IF .eofrfa [rfa$l_vbn] NEQ 0
;  972          1785  2 AND .readrfa [rfa$l_vbn] EQL .eofrfa [rfa$l_vbn]
;  973          1786  2 AND .readrfa [rfa$w_offset] EQL .eofrfa [rfa$w_offset]
;  974          1787  3 THEN BEGIN
;  975          1788  3     eofrfa [rfa$l_vbn] = 0;
;  976          1789  3     RETURN rms$_eof;
;  977          1790  2     END;
;  978          1791  2
;  979          1792  2 perform (map_blk_to_mem (.readrfa, true, blkadr, cachentry));
;  980          1793  3 BEGIN
;  981          1794  3     BIND
;  982          1795  3         bytecount = blkadr [.readrfa [rfa$w_offset]] : WORD;     !Name bytecount
;  983          1796  3     descrip [dsc$w_length] = .bytecount;                         !and return it to caller
;  984          1797  3     IF .bytecount GTRU lbr$c_maxrecsiz          !Make sure it's really a record
;  985          1798  3         THEN RETURN lbr$_invrfa;                ! and return error if not
;  986          1799  3     IF .bytecount+.readrfa [rfa$w_offset]+bsize LEQU data$c_length !If record on one block
;  987          1800  4     THEN BEGIN
;  988          1801  4         descrip [dsc$a_pointer] = blkadr [.readrfa [rfa$w_offset]]+bsize; !return the address
;  989          1802  4         incr_rfa (.descrip [dsc$w_length] +bsize, .readrfa);                    !increment RFA
```

```
;  990          1803  4          RETURN true
;  991          1804  4          END
;  992          1805  4  !
;  993          1806  4  ! Record is split across multiple blocks
;  994          1807  4  !
;  995          1808  4      ELSE BEGIN
;  996          1809  4          IF .lbr$gl_control [lbr$b_func] EQL lbr$c_read  !If reading the library
;  997          1810  4          AND .context [ctx$l_rdbufr] NEQ 0               ! and read buffer is allocated
;  998          1811  5          THEN BEGIN
;  999          1812  5  !
; 1000          1813  5  ! See if whole record is in the read buffer
; 1001          1814  5  !
; 1002          1815  5              LOCAL
; 1003          1816  5                  endrfa : BBLOCK [rfa$c_length];
; 1004          1817  5
; 1005          1818  5              CH$MOVE (rfa$c_length, .readrfa, endrfa);
; 1006          1819  5              incr_rfa (.descrip [dsc$w_length] + bsize, endrfa);         !Compute ending rfa
; 1007          1820  5              IF .endrfa [rfa$l_vbn] LSSU                     !If whole record in buffer
; 1008          1821  5                  .context [ctx$l_rdvbn1] + .context [ctx$l_rdblks]
; 1009          1822  6              THEN BEGIN
; 1010          1823  6                  descrip [dsc$a_pointer] = blkadr [.readrfa [rfa$w_offset]]+bsize; !Return address to caller
; 1011          1824  6                  incr_rfa (.descrip [dsc$w_length] + bsize, .readrfa);   !Update rfa
; 1012          1825  6                  RETURN true
; 1013          1826  5                  END;
; 1014          1827  4              END;
; 1015          1828  4
; 1016          1829  4          incr_rfa (bsize, .readrfa);                 !skip the byte count
; 1017          1830  4
; 1018          1831  4          IF .context [ctx$l_readbuf] EQL 0                   !If no buffer allocated
; 1019          1832  4          THEN perform (get_mem (lbr$c_maxrecsiz, context [ctx$l_readbuf]));
; 1020          1833  4          descrip [dsc$a_pointer] = .context [ctx$l_readbuf];     !Return address to caller
; 1021          1834  4          bufptr = .context [ctx$l_readbuf];          !Init buffer pointer
; 1022          1835  4          bytcnt = .bytecount;                        !Set up byte count
; 1023          1836  5          DO BEGIN                                    !Read whole record into buffer
; 1024          1837  5              perform (map_blk_to_mem (.readrfa, true, blkadr, cachentry)); !Map into memory
; 1025          1838  5              movecount = MINU (.bytcnt, data$c_length - .readrfa [rfa$w_offset]); !Compute length of move
; 1026          1839  5              bufptr = CH$MOVE (.movecount, blkadr [.readrfa [rfa$w_offset]], .bufptr); !Copy partial record
; 1027          1840  5              bytcnt = .bytcnt - .movecount;          !Update bytes left to go
; 1028          1841  5              incr_rfa (.movecount, .readrfa);        !Update RFA
; 1029          1842  5              END
; 1030          1843  4          UNTIL .bytcnt EQL 0;
; 1031          1844  3          END;
; 1032          1845  2      END;
; 1033          1846  2 RETURN true                         ! return good record
; 1034          1847  1 END;                                ! Of read_record


                                03FC   8F  BB 00000 READ_OLD_RECORD::
                                                    PUSHR   #^M<R2,R3,R4,R5,R6,R7,R8,R9>                ; 1747
                              5E          10  C2 00004      SUBL2   #16, SP                             ;
                              57          51  D0 00007      MOVL    R1, R7                              ;
                              58          50  D0 0000A      MOVL    R0, R8                              ;
                              50    0000G CF  D0 0000D      MOVL    LBR$GL_CONTROL, R0                  ; 1778
                              56      0E  A0  D0 00012      MOVL    14(R0), R6                          ;
```

```
                        50        22    A6  9E 00016          MOVAB   34(R6), R0                                  ; 1779
                                        60  D5 0001A          TSTL    (R0)                                       ; 1784
                                        17  13 0001C          BEQL    1$                                         ;
                        60              68  D1 0001E          CMPL    (READRFA), (R0)                            ; 1785
                                        12  12 00021          BNEQ    1$                                         ;
                     04 A0        04    A8  B1 00023          CMPW    4(READRFA), 4(R0)                          ; 1786
                                        0B  12 00028          BNEQ    1$                                         ;
                                        60  D4 0002A          CLRL    (R0)                                       ; 1788
                        50 0001827A     8F  D0 0002C          MOVL    #98938, R0                                 ; 1789
                                        2A  11 00033          BRB     2$                                         ;
                                        5E  DD 00035 1$:      PUSHL   SP                                         ; 1792
                                  08    AE  9F 00037          PUSHAB  BLKADR                                     ;
                                        01  DD 0003A          PUSHL   #1                                         ;
                                        58  DD 0003C          PUSHL   READRFA                                    ;
                     0000V CF           04  FB 0003E          CALLS   #4, MAP_BLK_TO_MEM                         ;
                           19           50  E9 00043          BLBC    STATUS, 2$                                 ;
                           59     04    A8  3C 00046          MOVZWL  4(READRFA), R9                             ; 1795
                           59     04    AE  C0 0004A          ADDL2   BLKADR, R9                                 ;
                           67           69  B0 0004E          MOVW    (R9), (DESCRIP)                            ; 1796
                     0800  8F           69  B1 00051          CMPW    (R9), #2048                                ; 1797
                                        0A  1B 00056          BLEQU   3$                                         ;
                        50 00000000G    8F  D0 00058          MOVL    #LBR$_INVRFA, R0                           ; 1798
                                     00CA   31  0005F 2$:      BRW     10$                                        ;
                        50              69  3C 00062 3$:      MOVZWL  (R9), R0                                   ; 1799
                        51        04    A8  3C 00065          MOVZWL  4(READRFA), R1                             ;
                        50              51  C0 00069          ADDL2   R1, R0                                     ;
                        50              02  C0 0006C          ADDL2   #2, R0                                     ;
                 00000200  8F           50  D1 0006F          CMPL    R0, #512                                   ;
                                        2E  1B 00076          BLEQU   4$                                         ;
                        50       0000G  CF  D0 00078          MOVL    LBR$GL_CONTROL, R0                         ; 1809
                        01         03   A0  91 0007D          CMPB    3(R0), #1                                  ;
                                        36  12 00081          BNEQ    5$                                         ;
                                   32   A6  D5 00083          TSTL    50(R6)                                     ; 1810
                                        31  13 00086          BEQL    5$                                         ;
                  08 AE                 68  06 28 00088          MOVC3   #6, (READRFA), ENDRFA                      ; 1818
                        51         08   AE  9E 0008D          MOVAB   ENDRFA, R1                                 ; 1819
                        50              67  3C 00091          MOVZWL  (DESCRIP), R0                              ;
                        50              02  C0 00094          ADDL2   #2, R0                                     ;
                                     0000G  30  00097          BSBW    INCR_RFA                                   ;
                     50       36    3A   A6  C1 0009A          ADDL3   58(R6), 54(R6), R0                         ; 1821
                        50         08   AE  D1 000A0          CMPL    ENDRFA, R0                                 ;
                                        13  1E 000A4          BGEQU   5$                                         ;
                              04    A7   02  A9  9E 000A6 4$:      MOVAB   2(R9), 4(DESCRIP)                     ; 1823
                        50              67  3C 000AB          MOVZWL  (DESCRIP), R0                              ; 1824
                        50              02  C0 000AE          ADDL2   #2, R0                                     ;
                        51              58  D0 000B1          MOVL    READRFA, R1                                ;
                                     0000G  30  000B4          BSBW    INCR_RFA                                   ;
                                        70  11 000B7          BRB     9$                                         ;
                        51              58  D0 000B9 5$:      MOVL    READRFA, R1                                ; 1829
                        50              02  D0 000BC          MOVL    #2, R0                                     ;
                                     0000G  30  000BF          BSBW    INCR_RFA                                   ;
                                   2E   A6  D5 000C2          TSTL    46(R6)                                     ; 1831
                                        0F  12 000C5          BNEQ    6$                                         ;
                        51         2E   A6  9E 000C7          MOVAB   46(R6), R1                                 ; 1832
                        50       0800   8F  3C 000CB          MOVZWL  #2048, R0                                  ;
                                     0000G  30  000D0          BSBW    GET_MEM                                    ;
                        56              50  E9 000D3          BLBC    STATUS, 10$                                ;
```

```
          04  A7     2E  A6  D0 000D6 6$:     MOVL    46(R6), 4(DESCRIP)                  ; 1833
              53     2E  A6  D0 000DB         MOVL    46(R6), BUFPTR                      ; 1834
              57         69  3C 000DF         MOVZWL  (R9), BYTCNT                        ; 1835
                         5E  DD 000E2 7$:     PUSHL   SP                                  ; 1837
                     08  AE  9F 000E4         PUSHAB  BLKADR                              ;
                         01  DD 000E7         PUSHL   #1                                  ;
                         58  DD 000E9         PUSHL   READRFA                             ;
       0000V  CF         04  FB 000EB         CALLS   #4, MAP_BLK_TO_MEM                  ;
              39         50  E9 000F0         BLBC    STATUS, 10$                         ;
              51     04  A8  3C 000F3         MOVZWL  4(READRFA), R1                      ; 1838
   51 00000200  8F       51  C3 000F7         SUBL3   R1, #512, R1                        ;
              50         57  D0 000FF         MOVL    BYTCNT, R0                          ;
              51         50  D1 00102         CMPL    R0, R1                              ;
                         03  1B 00105         BLEQU   8$                                  ;
              50         51  D0 00107         MOVL    R1, R0                              ;
              56         50  D0 0010A 8$:     MOVL    R0, MOVECOUNT                       ;
              50     04  A8  3C 0010D         MOVZWL  4(READRFA), R0                      ; 1839
              50     04  AE  C0 00111         ADDL2   BLKADR, R0                          ;
     63       60         56  28 00115         MOVC3   MOVECOUNT, (R0), (BUFPTR)           ;
              57         56  C2 00119         SUBL2   MOVECOUNT, BYTCNT                   ; 1840
              51         58  D0 0011C         MOVL    READRFA, R1                         ; 1841
              50         56  D0 0011F         MOVL    MOVECOUNT, R0                       ;
                      0000G  30 00122         BSBW    INCR_RFA                            ;
                         57  D5 00125         TSTL    BYTCNT                              ; 1843
                         B9  12 00127         BNEQ    7$                                  ;
              50         01  D0 00129 9$:     MOVL    #1, R0                              ; 1846
              5E         10  C0 0012C 10$:    ADDL2   #16, SP                             ; 1847
                  03FC   8F  BA 0012F         POPR    #^M<R2,R3,R4,R5,R6,R7,R8,R9>        ;
                         05 00133             RSB                                         ;

; Routine Size:  308 bytes,    Routine Base:  $CODE$ + 0778
```

```
;  1036          1848  1 %SBTTL  'map_blk_to_mem';
;  1037          1849  1 ROUTINE map_blk_to_mem (rfadr, reading, blkadr, cachentry) =
;  1038          1850  2 BEGIN
;  1039          1851  2 !++
;  1040          1852  2 !
;  1041          1853  2 ! Find block in memory, given RFA
;  1042          1854  2 !
;  1043          1855  2 ! Inputs:
;  1044          1856  2 !
;  1045          1857  2 !       rfadr           Address of RFA to find
;  1046          1858  2 !       reading         true if reading/updateing, otherwise false
;  1047          1859  2 !
;  1048          1860  2 ! Outputs:
;  1049          1861  2 !
;  1050          1862  2 !       blkadr          Address of block if found
;  1051          1863  2 !       cachentry       Address of cache entry for block
;  1052          1864  2 !
;  1053          1865  2 !       RFA requested may be modified if writing.
;  1054          1866  2 !
;  1055          1867  2 !--
;  1056          1868  2 MAP
;  1057          1869  2     rfadr : REF BBLOCK,
;  1058          1870  2     cachentry : REF BBLOCK;
;  1059          1871  2
;  1060          1872  2 BIND
;  1061          1873  2     context = .lbr$gl_control [lbr$l_ctxptr] : BBLOCK,
;  1062          1874  2     diskvbn = rfadr [rfa$l_vbn],
;  1063          1875  2     offset = rfadr [rfa$w_offset] : WORD,
;  1064          1876  2     header = .lbr$gl_control [lbr$l_hdrptr] : BBLOCK,
;  1065          1877  2     next_vbn = header [lhd$l_nextvbn];          ! Library end of file
;  1066          1878  2
;  1067          1879  2 LOCAL
;  1068          1880  2     status,
;  1069          1881  2     newvbn,
;  1070          1882  2     cacheaddr : REF BBLOCK;
;  1071          1883  2
;  1072          1884  2 !
;  1073          1885  2 ! If just reading the file, use a block buffer instead. Allocate it now if needed
;  1074          1886  2 !
;  1075          1887  2 IF .lbr$gl_control [lbr$b_func] EQL lbr$c_read          !Reading the library?
;  1076          1888  2 !***AND .context [ctx$v_o[dlib]                        ! and its old format
;  1077          1889  3 THEN BEGIN
;  1078          1890  3     IF .context [ctx$l_rdbufr] EQL 0                    !Need a buffer?
;  1079        P 1891  3         THEN perform (get_mem (.lbr$gl_maxread * lbr$c_pagesize, ! then allocate one
;  1080          1892  3                                context [ctx$l_rdbufr]));
;  1081          1893  3     IF .diskvbn GEQU .context [ctx$l_rdvbn1]            !Is block in the buffer?
;  1082          1894  3         AND .diskvbn LSSU .context [ctx$l_rdvbn1] + .context [ctx$l_rdblks]
;  1083          1895  4         THEN BEGIN
;  1084          1896  4             .blkadr = .context [ctx$l_rdbufr] +      !Yes! return block address
;  1085          1897  4                       (.diskvbn - .context [ctx$l_rdvbn1]) * lbr$c_pagesize;
;  1086          1898  4             RETURN true;
;  1087          1899  4             END
;  1088          1900  4         ELSE BEGIN
;  1089          1901  4             BIND
;  1090          1902  4                 lrab = .context [ctx$l_recrab] : BBLOCK; !RAB for I/O
;  1091          1903  4             LOCAL
;  1092          1904  4                 status;
```

```
;  1093          1905  4
;  1094          1906  4              lrab [rab$l_bkt] = .diskvbn;                    !Set starting block
;  1095          1907  4              lrab [rab$l_ubf] = .context [ctx$l_rdbufr]; !and buffer address
;  1096          1908  4              lrab [rab$w_usz] = .lbr$gl_maxread * lbr$c_pagesize; !Set buffer size
;  1097          1909  5              IF (status = $READ (RAB = [rab))                !If good read
;  1098          1910  4                  OR .status EQL rms$_eof                     ! or we read to eof
;  1099          1911  5                  THEN BEGIN                                  !Then things look good
;  1100          1912  5                      .blkadr = .context [ctx$l_rdbufr];      !Return buffer address
;  1101          1913  5                      context [ctx$l_rdblks] = .lrab [rab$w_rsz] / lbr$c_pagesize;
;  1102          1914  5                      context [ctx$l_rdvbn1] = .diskvbn;      !Set vbn into context block
;  1103          1915  5                      RETURN true;
;  1104          1916  5                      END
;  1105          1917  5                  ELSE BEGIN
;  1106          1918  5                      lbr$gl_rmsstv = .lrab [rab$l_stv];      !Return stv on error
;  1107          1919  5                      RETURN .status;
;  1108          1920  4                      END;
;  1109          1921  4              END
;  1110          1922  3          END
;  1111          1923  3      ELSE BEGIN                                              ! Also writing, so cache disk blocks
;  1112          1924  3      IF .diskvbn LSSU .next_vbn                              !Disk block already allocated?
;  1113          1925  3      OR .context [ctx$v_old[ib]                              ! or an old format library (always!)
;  1114          1926  4      THEN BEGIN                                              !Yes--look in cache first
;  1115          1927  5          IF (status = lookup_cache (.diskvbn, cacheaddr))    ! and if it is found
;  1116          1928  4              AND .cacheaddr [cache$v_data]                   ! and it is a data block
;  1117          1929  5              AND (.reading OR (.offset NEQ 0))               ! and we are reading or writing and
;  1118          1930  5                                                              !  not just starting the block
;  1119          1931  5              THEN BEGIN
;  1120          1932  5                  .blkadr = .cacheaddr [cache$l_address];      ! then use it
;  1121          1933  5                  .cachentry = .cacheaddr;
;  1122          1934  5                  RETURN true;
;  1123          1935  5                  END
;  1124          1936  4              ELSE IF NOT .reading                            !Not found--if writing the record
;  1125          1937  5              THEN BEGIN                                      ! then allocate a new block
;  1126          1938  5                  alloc_block (newvbn, .blkadr);              !allocate a new block
;  1127          1939  5                  offset = data$c_data;                       !Set offset
;  1128          1940  5                  CH$FILL (0, data$c_data, ..blkadr);         !Zero info at start of block
;  1129          1941  5                  diskvbn = .newvbn;                          !Fill in block allocated
;  1130          1942  5                  END
;  1131          1943  5              ELSE BEGIN                                      !Otherwise, read it from the disk
;  1132          1944  5                  perform (read_block (.diskvbn, .blkadr));
;  1133          1945  4                  END;
;  1134          1946  4          END
;  1135          1947  3      ELSE IF .diskvbn GTRU .next_vbn                         !Not allocated--is this a bad call?
;  1136          1948  3      THEN RETURN lbr$_rfapasteof                             !yes, return error
;  1137          1949  4      ELSE BEGIN
;  1138          1950  4          IF .offset EQL 0                                    !Just starting the new block?
;  1139          1951  4          AND NOT .reading                                    ! and writing
;  1140          1952  5              THEN BEGIN
;  1141          1953  5                  alloc_block (newvbn, .blkadr);              !yes--allocate it
;  1142          1954  5                  offset = data$c_data;                       !Set correct offset
;  1143          1955  5                  CH$FILL (0, data$c_data, ..blkadr);         !Zero info in block
;  1144          1956  5                  diskvbn = .newvbn;                          !update vbn gotten
;  1145          1957  5                  END
;  1146          1958  5              ELSE BEGIN                                      !We've already touched the block
;  1147          1959  5                  IF lookup_cache (.diskvbn, cacheaddr)       !So find the cache entry
;  1148          1960  6                  THEN BEGIN
;  1149          1961  6                      .blkadr = .cacheaddr [cache$l_address]; !Get the data block address
```

```
;  1150          1962  6                 .cachentry = .cacheaddr;
;  1151          1963  6                 RETURN true;
;  1152          1964  6                 END
;  1153          1965  6             ELSE BEGIN                                  !It's not in memory, read it in
;  1154          1966  6                 perform (read_block (.diskvbn, .blkadr)); !it wasn't so read it in
;  1155          1967  5                 END;
;  1156          1968  4             END;
;  1157          1969  3     END;
;  1158          1970  3 perform (add_cache (.diskvbn, cacheaddr));              !Insert into disk block cache
;  1159          1971  3 .cachentry = .cacheaddr;                               !Return cache entry address to caller
;  1160          1972  3 cacheaddr [cache$l_address] = ..blkadr;
;  1161          1973  3 cacheaddr [cache$v_data] = true;
;  1162          1974  3 RETURN true
;  1163          1975  2 END;
;  1164          1976  1 END;                                                    !Of map_blk_to_mem


                                                        .EXTRN  SYS$READ

                                        0FFC 00000 MAP_BLK_TO_MEM:
                                                   .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11        ; 1849
                                    5E          08  C2 00002           SUBL2   #8, SP                  ;
                                    50    0000G CF  D0 00005           MOVL    LBR$GL_CONTROL, R0      ; 1873
                                    53      0E  A0  D0 0000A           MOVL    14(R0), R3              ;
                                    56      04  AC  D0 0000E           MOVL    RFADR, R6               ; 1874
                              52    0A  A0 00000052 8F  C1 00012       ADDL3   #82, 10(R0), R2         ; 1877
                                    01      03  A0  91 0001B           CMPB    3(R0), #1               ; 1887
                                                03  13 0001F           BEQL    1$                      ;
                                              0083  31 00021           BRW     7$                      ;
                                           32   A3  D5 00024 1$:        TSTL    50(R3)                  ; 1890
                                                11  12 00027           BNEQ    2$                      ;
                                        51      32  A3  9E 00029       MOVAB   50(R3), R1              ; 1892
                                  50      0000G CF      09  78 0002D   ASHL    #9, LBR$GL_MAXREAD, R0  ;
                                                0000G 30  00033       BSBW    GET_MEM                 ;
                                        01          50  E8 00036       BLBS    STATUS, 2$              ;
                                                    04  00039       RET                             ;
                                           36   A3      66  D1 0003A 2$:     CMPL    (R6), 54(R3)            ; 1893
                                                    1C  1F 0003E       BLSSU   3$                      ;
                                  50        36  A3  3A  A3  C1 00040   ADDL3   58(R3), 54(R3), R0      ; 1894
                                        50          66  D1 00046       CMPL    (R6), R0                ;
                                                    11  1E 00049       BGEQU   3$                      ;
                                  50            66  36  A3  C3 0004B   SUBL3   54(R3), (R6), R0        ; 1897
                                  50            50      09  78 00050   ASHL    #9, R0, R0              ;
                                           0C   BC      32 B340  9E 00054   MOVAB   @50(R3)[R0], @BLKADR      ;
                                                    41  11 0005A       BRB     5$                      ; 1902
                                        52      0C  A3      D0 0005C 3$:     MOVL    12(R3), R2              ;
                                           38   A2      66  D0 00060       MOVL    (R6), 56(R2)            ; 1906
                                           24   A2  32  A3  D0 00064       MOVL    50(R3), 36(R2)          ; 1907
                              20    A2    0000G CF  0200  8F  A5 00069   MULW3   #512, LBR$GL_MAXREAD, 32(R2) ; 1908
                                                    52  DD 00072       PUSHL   R2                      ; 1909
                                    00000000G   00      01  FB 00074   CALLS   #1, SYS$READ             ;
                                                09      50  E8 0007B   BLBS    STATUS, 4$              ;
                                    0001827A    8F      50  D1 0007E   CMPL    STATUS, #98938          ; 1910
                                                    19  12 00085       BNEQ    6$                      ;
                                           0C   BC  32  A3  D0 00087 4$:     MOVL    50(R3), @BLKADR         ; 1912
                                                50  22  A2  3C 0008C       MOVZWL  34(R2), R0              ; 1913
```

```
                3A   A3          50 00000200   8F   C7 00090          DIVL3   #512, R0, 58(R3)                                    ;
                          36    A3            66   D0 00099          MOVL    (R6), 54(R3)                                        ; 1914
                                              00AD 31 0009D 5$:      BRW     17$                                                 ; 1917
                     0000G  CF          0C    A2   D0 000A0 6$:      MOVL    12(R2), LBR$GL_RMSSTV                               ; 1918
                                              04   000A6              RET                                                         ; 1923
                          62                  66   D1 000A7 7$:      CMPL    (R6), (R2)                                          ; 1924
                                              05   1F 000AA          BLSSU   8$                                                  ;
                     25         04    A3      05   E1 000AC          BBC     #5, 4(R3), 10$                                      ; 1925
                     51              04    AE   9E 000B1 8$:      MOVAB   CACHEADDR, R1                                       ; 1927
                     50                    66   D0 000B5          MOVL    (R6), R0                                            ;
                                         0000G 30 000B8          BSBW    LOOKUP_CACHE                                        ;
                     12                    50   E9 000BB          BLBC    STATUS, 9$                                          ;
                     50              04    AE   D0 000BE          MOVL    CACHEADDR, R0                                       ; 1928
                     09         0C   A0      01   E1 000C2          BBC     #1, 12(R0), 9$                                      ;
                     48              08    AC   E8 000C7          BLBS    READING, 14$                                        ; 1929
                                     04    A6   B5 000CB          TSTW    4(R6)                                               ;
                                           43   12 000CE          BNEQ    14$                                                 ;
                     4E              08    AC   E8 000D0 9$:      BLBS    READING, 15$                                        ; 1936
                                           13   11 000D4          BRB     12$                                                 ; 1938
                                           08   1B 000D6 10$:     BLEQU   11$                                                 ; 1947
                     50 00000000G        8F   D0 000D8          MOVL    #LBR$_RFAPASTEOF, R0                                ; 1948
                                              04   000DF          RET                                                         ;
                                     04    A6   B5 000E0 11$:     TSTW    4(R6)                                               ; 1950
                                           21   12 000E3          BNEQ    13$                                                 ;
                     1D              08    AC   E8 000E5          BLBS    READING, 13$                                        ; 1951
                     50                    6E   9E 000E9 12$:     MOVAB   NEWVBN, R0                                          ; 1953
                     51              0C    AC   D0 000EC          MOVL    BLKADR, R1                                          ;
                                         0000G 30 000F0          BSBW    ALLOC_BLOCK                                         ;
                          04    A6           06   B0 000F3          MOVW    #6, 4(R6)                                           ; 1954
                     50              0C    BC   D0 000F7          MOVL    @BLKADR, R0                                         ; 1955
          06         00              6E         00   2C 000FB          MOVC5   #0, (SP), #0, #6, (R0)                              ;
                                                60   00100                                                                      ;
                     66                    6E   D0 00101          MOVL    NEWVBN, (R6)                                        ; 1956
                                           29   11 00104          BRB     16$                                                 ; 1950
                     51              04    AE   9E 00106 13$:     MOVAB   CACHEADDR, R1                                       ; 1959
                     50                    66   D0 0010A          MOVL    (R6), R0                                            ;
                                         0000G 30 0010D          BSBW    LOOKUP_CACHE                                        ;
                     0F                    50   E9 00110          BLBC    R0, 15$                                             ;
                     50              04    AE   D0 00113 14$:     MOVL    CACHEADDR, R0                                       ; 1961
               0C    BC              08    A0   D0 00117          MOVL    8(R0), @BLKADR                                      ;
               10    BC                    50   D0 0011C          MOVL    R0, @CACHENTRY                                      ; 1962
                                           2B   11 00120          BRB     17$                                                 ; 1963
                     51              0C    AC   D0 00122 15$:     MOVL    BLKADR, R1                                          ; 1966
                     50                    66   D0 00126          MOVL    (R6), R0                                            ;
                                         0000G 30 00129          BSBW    READ_BLOCK                                          ;
                     21                    50   E9 0012C          BLBC    STATUS, 18$                                         ;
                     51              04    AE   9E 0012F 16$:     MOVAB   CACHEADDR, R1                                       ; 1970
                     50                    66   D0 00133          MOVL    (R6), R0                                            ;
                                         0000G 30 00136          BSBW    ADD_CACHE                                           ;
                     14                    50   E9 00139          BLBC    STATUS, 18$                                         ;
                     50              04    AE   D0 0013C          MOVL    CACHEADDR, R0                                       ; 1971
               10    BC                    50   D0 00140          MOVL    R0, @CACHENTRY                                      ;
               08    A0              0C    BC   D0 00144          MOVL    @BLKADR, 8(R0)                                      ; 1972
               0C    A0                    02   88 00149          BISB2   #2, 12(R0)                                          ; 1973
                     50                    01   D0 0014D 17$:     MOVL    #1, R0                                              ; 1974
                                               04   00150 18$:     RET                                                         ; 1976
```

```
; Routine Size:  337 bytes,    Routine Base:  $CODE$ + 08AC
```

```
;  1166          1977  1 %SBTTL  'update_nextrfa';
;  1167          1978  1 ROUTINE update_nextrfa (rfa) : JSB_1 =
;  1168          1979  2 BEGIN
;  1169          1980  2 !++
;  1170          1981  2 ! Update the next RFA location (LHD$B_NEXTRFA) in library header if
;  1171          1982  2 ! needed.
;  1172          1983  2 !
;  1173          1984  2 ! Inputs:
;  1174          1985  2 !
;  1175          1986  2 !       rfa             Address of new rfa
;  1176          1987  2 !
;  1177          1988  2 ! Outputs:
;  1178          1989  2 !
;  1179          1990  2 !       nextrfa in header updated if new rfa is greater.
;  1180          1991  2 !
;  1181          1992  2 !--
;  1182          1993  2
;  1183          1994  2 MAP
;  1184          1995  2     rfa : REF BBLOCK;
;  1185          1996  2
;  1186          1997  2 BIND
;  1187          1998  2     header = .lbr$gl_control [lbr$l_hdrptr] : BBLOCK,
;  1188          1999  2     hdrnxtrfa = header [lhd$b_nextrfa] : BBLOCK;
;  1189          2000  2
;  1190          2001  2 IF .rfa [rfa$l_vbn] GTRU .hdrnxtrfa [rfa$l_vbn]
;  1191          2002  4     OR ((.rfa [rfa$l_vbn] EQL .hdrnxtrfa [rfa$l_vbn])
;  1192          2003  3     AND (.rfa [rfa$w_offset] GTRU .hdrnxtrfa [rfa$w_offset]))
;  1193          2004  2 THEN
;  1194          2005  2     CH$MOVE (rfa$c_length, .rfa, hdrnxtrfa);
;  1195          2006  2
;  1196          2007  2 RETURN true;
;  1197          2008  1 END;


                                        3C  BB 00000 UPDATE_NEXTRFA:
                                                     PUSHR   #^M<R2,R3,R4,R5>                    ; 1978
                              51    0000G CF  D0 00002         MOVL    LBR$GL_CONTROL, R1        ; 1998
                    51    0A    A1 0000004C 8F  C1 00007         ADDL3   #76, 10(R1), R1           ; 1999
                              61          60  D1 00010         CMPL    (RFA), (R1)               ; 2001
                                          09  1A 00013         BGTRU   1$                        ;
                                          0B  12 00015         BNEQ    2$                        ; 2002
                          04  A1      04  A0  B1 00017         CMPW    4(RFA), 4(R1)             ; 2003
                                          04  1B 0001C         BLEQU   2$                        ;
                    61          60        06  28 0001E 1$:     MOVC3   #6, (RFA), (R1)           ; 2005
                              50          01  D0 00022 2$:     MOVL    #1, R0                    ; 2007
                                          3C  BA 00025         POPR    #^M<R2,R3,R4,R5>          ; 2008
                                              05 00027         RSB                               ;

; Routine Size:  40 bytes,    Routine Base:  $CODE$ + 09FD
```

```
;  1199     2009  1 %SBTTL  'incr_refcnt';
;  1200     2010  1 GLOBAL ROUTINE incr_refcnt (txtrfa) =
;  1201     2011  2 BEGIN
;  1202     2012  2 !++
;  1203     2013  2 ! Increment the module reference count in the module header
;  1204     2014  2 !
;  1205     2015  2 ! Inputs:
;  1206     2016  2 !
;  1207     2017  2 !       txtrfa          Address of rfa for module header
;  1208     2018  2 !
;  1209     2019  2 ! Outputs:
;  1210     2020  2 !
;  1211     2021  2 !       Reference count in module header is incremented.
;  1212     2022  2 !
;  1213     2023  2 !--
;  1214     2024  2
;  1215     2025  2 MAP
;  1216     2026  2     txtrfa : REF BBLOCK;
;  1217     2027  2
;  1218     2028  2 LOCAL
;  1219     2029  2     header : BBLOCK [lbr$c_maxhdrsiz],
;  1220     2030  2     hdrdesc : BBLOCK [dsc$c_s_bln],
;  1221     2031  2     hdrlen,
;  1222     2032  2     blockaddr : REF VECTOR [,BYTE],
;  1223     2033  2     cachentry : REF BBLOCK,
;  1224     2034  2     localrfa : BBLOCK [rfa$c_length];
;  1225     2035  2
;  1226     2036  2 CH$MOVE (rfa$c_length, .txtrfa, localrfa);
;  1227     2037  2 perform (map_blk_to_mem (localrfa, true, blockaddr, cachentry));
;  1228     2038  2 IF (.txtrfa [rfa$w_offset] + mhd$c_reflng + 2) LEQU data$c_length
;  1229     2039  3 THEN BEGIN
;  1230     2040  3     BIND
;  1231     2041  3         libhdr = .lbr$gl_control [lbr$l_hdrptr] : BBLOCK,       !Library header
;  1232     2042  3         reclen = blockaddr [.txtrfa [rfa$w_offset]] : WORD,     !Length of record
;  1233     2043  3         refcnt = blockaddr [.txtrfa [rfa$w_offset] + mhd$c_reflng - 2];
;  1234     2044  3
;  1235     2045  3     IF .reclen NEQ mhd$c_mhdlen + .libhdr [lhd$b_mhdusz]
;  1236     2046  3         THEN RETURN lbr$_invrfa;
;  1237     2047  3     refcnt = .refcnt + 1;
;  1238     2048  3     cachentry [cache$v_dirty] = true;           !Mark block dirty
;  1239     2049  3     END
;  1240     2050  2 !
;  1241     2051  2 ! Module header is split across blocks
;  1242     2052  2 !
;  1243     2053  2 ELSE BEGIN
;  1244     2054  3     hdrdesc [dsc$w_length] = lbr$c_maxhdrsiz;
;  1245     2055  3     hdrdesc [dsc$a_pointer] = header;
;  1246     2056  3     perform (set_module (.txtrfa, hdrdesc, hdrlen));
;  1247     2057  3     header [mhd$l_refcnt] = .header [mhd$l_refcnt] + 1;
;  1248     2058  3     CH$MOVE (rfa$c_length, .txtrfa, localrfa);
;  1249     2059  3     perform (write_record (.hdrlen, header, localrfa, true));
;  1250     2060  2     END;
;  1251     2061  2
;  1252     2062  2 RETURN true
;  1253     2063  1 END;
```

```
                                 007C 00000          .ENTRY  INCR_REFCNT, Save R2,R3,R4,R5,R6                ; 2010
                          5E     FF64 CE   9E 00002  MOVAB   -156(SP), SP                                    ;
                          56       04 AC   D0 00007  MOVL    TXTRFA, R6                                      ; 2036
                    0C    AE         66   06   28 0000B  MOVC3   #6, (R6), LOCALRFA                         ;
                                        5E   DD 00010  PUSHL   SP                                          ; 2037
                                     08 AE   9F 00012  PUSHAB  BLOCKADDR                                   ;
                                        01   DD 00015  PUSHL   #1                                          ;
                                     18 AE   9F 00017  PUSHAB  LOCALRFA                                    ;
                              FE68   CF         04   FB 0001A  CALLS   #4, MAP_BLK_TO_MEM                   ;
                                     7F         50   E9 0001F  BLBC    STATUS, 3$                           ;
                                     50       04 A6   3C 00022  MOVZWL  4(R6), R0                           ; 2038
                                     50          0A   C0 00026  ADDL2   #10, R0                             ;
                           00000200  8F          50   D1 00029  CMPL    R0, #512                           ;
                                               3D   1A 00030  BGTRU   2$                                  ;
                                     50     0000G CF   D0 00032  MOVL    LBR$GL_CONTROL, R0                 ; 2041
                                     51       0A A0   D0 00037  MOVL    10(R0), R1                         ;
                                     52       04 A6   3C 0003B  MOVZWL  4(R6), R2                          ; 2042
                                     52       04 AE   C0 0003F  ADDL2   BLOCKADDR, R2                      ;
                                     50       04 A6   3C 00043  MOVZWL  4(R6), R0                          ; 2043
                                     50       04 AE   C0 00047  ADDL2   BLOCKADDR, R0                      ;
                                     50          06   C0 0004B  ADDL2   #6, R0                             ;
                                     51       3C A1   9A 0004E  MOVZBL  60(R1), R1                         ; 2045
                                     51          10   C0 00052  ADDL2   #16, R1                            ;
                          51               62   10   00   ED 00055  CMPZV   #0, #16, (R2), R1               ;
                                               08   13 0005A  BEQL    1$                                  ;
                                  50 00000000G  8F   D0 0005C  MOVL    #LBR$_INVRFA, R0                    ; 2046
                                                04 00063  RET                                              ;
                                               60   D6 00064  1$:    INCL    (R0)                          ; 2047
                                     50           6E   D0 00066  MOVL    CACHENTRY, R0                       ; 2048
                                  0C A0           01   88 00069  BISB2   #1, 12(R0)                         ;
                                               35   11 0006D  BRB     4$                                  ; 2038
                                  14 AE       80   8F   9B 0006F  2$:    MOVZBW  #128, HDRDESC               ; 2054
                                  18 AE       1C   AE   9E 00074  MOVAB   HEADER, HDRDESC+4                  ; 2055
                                              08   AE   9F 00079  PUSHAB  HDRLEN                             ; 2056
                                              18   AE   9F 0007C  PUSHAB  HDRDESC                           ;
                                              56   DD 0007F  PUSHL   R6                                     ;
                                   0000V      CF         03   FB 00081  CALLS   #3, SET_MODULE              ;
                                              1E         50   E9 00086  BLBC    STATUS, 5$                  ;
                                              20   AE   D6 00089  INCL    HEADER+4                            ; 2057
                          0C    AE                66   06   28 0008C  MOVC3   #6, (R6), LOCALRFA             ; 2058
                                                     01   DD 00091  PUSHL   #1                               ; 2059
                                              10   AE   9F 00093  PUSHAB  LOCALRFA                          ;
                                              24   AE   9F 00096  PUSHAB  HEADER                            ;
                                              14   AE   DD 00099  PUSHL   HDRLEN                            ;
                                   FA45       CF         04   FB 0009C  CALLS   #4, WRITE_RECORD            ;
                                              03         50   E9 000A1  3$:    BLBC    STATUS, 5$          ;
                                              50         01   D0 000A4  4$:    MOVL    #1, R0              ; 2062
                                                         04   00 000A7  5$:    RET                         ; 2063

; Routine Size:  168 bytes,    Routine Base:  $CODE$ + 0A25
```

```
;  1255          2064  1 %SBTTL  'decr_refcnt';
;  1256          2065  1 GLOBAL ROUTINE decr_refcnt (txtrfa) =
;  1257          2066  2 BEGIN
;  1258          2067  2 !++
;  1259          2068  2 ! Decrement the module reference count in the module header
;  1260          2069  2 !
;  1261          2070  2 ! Inputs:
;  1262          2071  2 !
;  1263          2072  2 !       txtrfa          Address of rfa of module header
;  1264          2073  2 !
;  1265          2074  2 ! Outputs:
;  1266          2075  2 !
;  1267          2076  2 !       reference count in module header is decremented.
;  1268          2077  2 !
;  1269          2078  2 !--
;  1270          2079  2
;  1271          2080  2 MAP
;  1272          2081  2     txtrfa : REF BBLOCK;
;  1273          2082  2
;  1274          2083  2 LOCAL
;  1275          2084  2     header : BBLOCK [lbr$c_maxhdrsiz],
;  1276          2085  2     hdrdesc : BBLOCK [dsc$c_s_bln],
;  1277          2086  2     hdrlen,
;  1278          2087  2     blockaddr : REF VECTOR [,BYTE],
;  1279          2088  2     cachentry : REF BBLOCK,
;  1280          2089  2     localrfa : BBLOCK [rfa$c_length];
;  1281          2090  2
;  1282          2091  2 CH$MOVE (rfa$c_length, .txtrfa, localrfa);
;  1283          2092  2 perform (map_blk_to_mem (localrfa, true, blockaddr, cachentry));
;  1284          2093  2 IF (.txtrfa [rfa$w_offset] + mhd$c_reflng + 2) LEQU data$c_length
;  1285          2094  3 THEN BEGIN
;  1286          2095  3     BIND
;  1287          2096  3         libhdr = .lbr$gl_control [lbr$l_hdrptr] : BBLOCK,       !Library header
;  1288          2097  3         reclen = blockaddr [.txtrfa [rfa$w_offset]] : WORD,     !Length of record
;  1289          2098  3         refcnt = blockaddr [.txtrfa [rfa$w_offset] + mhd$c_reflng - 2];
;  1290          2099  3
;  1291          2100  3     IF .reclen NEQ mhd$c_mhdlen + .libhdr [lhd$b_mhdusz]
;  1292          2101  3         THEN RETURN lbr$_invrfa;
;  1293          2102  3
;  1294          2103  3     refcnt = .refcnt - 1;
;  1295          2104  3     cachentry [cache$v_dirty] = true;
;  1296          2105  3     END
;  1297          2106  3 !
;  1298          2107  3 ! Module header is split across blocks
;  1299          2108  3 !
;  1300          2109  3 ELSE BEGIN
;  1301          2110  3     hdrdesc [dsc$w_length] = lbr$c_maxhdrsiz;
;  1302          2111  3     hdrdesc [dsc$a_pointer] = header;
;  1303          2112  3     perform (set_module (.txtrfa, hdrdesc, hdrlen));
;  1304          2113  3     header [mhd$l_refcnt] = .header [mhd$l_refcnt] - 1;
;  1305          2114  3     CH$MOVE (rfa$c_length, .txtrfa, localrfa);
;  1306          2115  3     perform (write_record (.hdrlen, header, localrfa, true));
;  1307          2116  2     END;
;  1308          2117  2
;  1309          2118  2 RETURN true
;  1310          2119  1 END;
```

```
                                 007C 00000          .ENTRY  DECR_REFCNT, Save R2,R3,R4,R5,R6                ; 2065
                          5E  FF64  CE   9E 00002    MOVAB   -156(SP), SP                                    ;
                          56    04  AC   D0 00007    MOVL    TXTRFA, R6                                      ; 2091
                    0C    AE        66   28 0000B    MOVC3   #6, (R6), LOCALRFA                              ;
                                    5E   DD 00010    PUSHL   SP                                              ; 2092
                                 08 AE   9F 00012    PUSHAB  BLOCKADDR                                       ;
                                 01      DD 00015    PUSHL   #1                                              ;
                                 18 AE   9F 00017    PUSHAB  LOCALRFA                                        ;
                        FDC0     CF      04   FB 0001A    CALLS   #4, MAP_BLK_TO_MEM                         ;
                          7F             50   E9 0001F    BLBC    STATUS, 3$                                 ;
                          50    04  A6   3C 00022    MOVZWL  4(R6), R0                                       ; 2093
                          50        0A   C0 00026    ADDL2   #10, R0                                         ;
                    00000200    8F        50   D1 00029    CMPL    R0, #512                                  ;
                                          3D   1A 00030    BGTRU   2$                                        ;
                          50  0000G CF      D0 00032    MOVL    LBR$GL_CONTROL, R0                          ; 2096
                          51    0A  A0      D0 00037    MOVL    10(R0), R1                                  ;
                          52    04  A6      3C 0003B    MOVZWL  4(R6), R2                                   ; 2097
                          52    04  AE      C0 0003F    ADDL2   BLOCKADDR, R2                               ;
                          50    04  A6      3C 00043    MOVZWL  4(R6), R0                                   ; 2098
                          50    04  AE      C0 00047    ADDL2   BLOCKADDR, R0                               ;
                          50        06      C0 0004B    ADDL2   #6, R0                                      ;
                          51    3C  A1      9A 0004E    MOVZBL  60(R1), R1                                  ; 2100
                          51        10      C0 00052    ADDL2   #16, R1                                     ;
                51        62        10      00   ED 00055    CMPZV   #0, #16, (R2), R1                       ;
                                          08   13 0005A    BEQL    1$                                      ;
                          50 00000000G    8F   D0 0005C    MOVL    #LBR$_INVRFA, R0                        ; 2101
                                          04   00063      RET                                             ;
                                          60   D7 00064 1$: DECL    (R0)                                    ; 2103
                          50              6E   D0 00066    MOVL    CACHENTRY, R0                           ; 2104
                       0C AO              01   88 00069    BISB2   #1, 12(R0)                              ;
                                          35   11 0006D    BRB     4$                                      ; 2093
                       14 AE      80   8F   9B 0006F 2$:  MOVZBW  #128, HDRDESC                           ; 2110
                       18 AE      1C   AE   9E 00074      MOVAB   HEADER, HDRDESC+4                       ; 2111
                                  08   AE   9F 00079      PUSHAB  HDRLEN                                  ; 2112
                                  18   AE   9F 0007C      PUSHAB  HDRDESC                                 ;
                                          56   DD 0007F    PUSHL   R6                                      ;
                      0000V       CF        03   FB 00081    CALLS   #3, SET_MODULE                        ;
                                  1E        50   E9 00086    BLBC    STATUS, 5$                            ;
                                  20   AE   D7   00089      DECL    HEADER+4                              ; 2113
                    0C    AE            66   28 0008C    MOVC3   #6, (R6), LOCALRFA                      ; 2114
                                          01   DD 00091    PUSHL   #1                                      ; 2115
                                  10   AE   9F   00093    PUSHAB  LOCALRFA                                ;
                                  24   AE   9F   00096    PUSHAB  HEADER                                  ;
                                  14   AE   DD   00099    PUSHL   HDRLEN                                  ;
                      F99D        CF        04   FB 0009C    CALLS   #4, WRITE_RECORD                      ;
                                  03        50   E9 000A1 3$: BLBC    STATUS, 5$                            ;
                                  50        01   D0 000A4 4$: MOVL    #1, R0                              ; 2118
                                            04   000A7 5$:   RET                                           ; 2119

; Routine Size:  168 bytes,    Routine Base:  $CODE$ + 0ACD
```

```
;  1312          2120  1 %SBTTL  'LBR$INSERT_TIME';
;  1313          2121  1 GLOBAL ROUTINE lbr$insert_time (control_index, txtrfa, newtime) =
;  1314          2122  2 BEGIN
;  1315          2123  2 !++
;  1316          2124  2 ! Replace the module inserted date/time with the provided newtime
;  1317          2125  2 !
;  1318          2126  2 ! Inputs:
;  1319          2127  2 !
;  1320          2128  2 !       control_index           Address of control index for library
;  1321          2129  2 !       txtrfa                  Address of rfa for module header
;  1322          2130  2 !       newtime                 Address of quadword containing new time to set in header
;  1323          2131  2 !
;  1324          2132  2 !--
;  1325          2133  2
;  1326          2134  2 MAP
;  1327          2135  2     newtime : REF VECTOR,
;  1328          2136  2     txtrfa : REF BBLOCK;
;  1329          2137  2
;  1330          2138  2 LOCAL
;  1331          2139  2     header : BBLOCK [lbr$c_maxhdrsiz],
;  1332          2140  2     hdrdesc : BBLOCK [dsc$c_s_bln],
;  1333          2141  2     hdrlen,
;  1334          2142  2     blockaddr : REF VECTOR [,BYTE],
;  1335          2143  2     cachentry : REF BBLOCK,
;  1336          2144  2     localrfa : BBLOCK [rfa$c_length];
;  1337          2145  2
;  1338          2146  2 perform (validate_ctl (..control_index));
;  1339          2147  2 CH$MOVE (rfa$c_length, .txtrfa, localrfa);
;  1340          2148  2 perform (map_blk_to_mem (localrfa, true, blockaddr, cachentry));
;  1341          2149  2 IF (.txtrfa [rfa$w_offset] + mhd$c_instime + 10) LEQU data$c_length
;  1342          2150  3 THEN BEGIN
;  1343          2151  3     BIND
;  1344          2152  3         libhdr = .lbr$gl_control [lbr$l_hdrptr] : BBLOCK,       !Library header
;  1345          2153  3         reclen = blockaddr [.txtrfa [rfa$w_offset]] : WORD,     !Length of record
;  1346          2154  3         daytime = blockaddr [.txtrfa [rfa$w_offset] + mhd$c_instime + 2];
;  1347          2155  3
;  1348          2156  3     IF .reclen NEQ mhd$c_mhdlen + .libhdr [lhd$b_mhdusz]
;  1349          2157  3         THEN RETURN lbr$_invrfa;
;  1350          2158  3
;  1351          2159  3     CH$MOVE (8, .newtime, daytime);                 !Set new time
;  1352          2160  3     cachentry [cache$v_dirty] = true;               !Mark block dirty
;  1353          2161  3     END
;  1354          2162  3 ELSE BEGIN
;  1355          2163  3     hdrdesc [dsc$w_length] = lbr$c_maxhdrsiz;
;  1356          2164  3     hdrdesc [dsc$a_pointer] = header;
;  1357          2165  3     perform (set_module (.txtrfa, hdrdesc, hdrlen));
;  1358          2166  3     CH$MOVE (8, .newtime, header [mhd$l_datim]); !Set new time
;  1359          2167  3     CH$MOVE (rfa$c_length, .txtrfa, localrfa);
;  1360          2168  3     perform (write_record (.hdrlen, header, localrfa, true));
;  1361          2169  2     END;
;  1362          2170  2
;  1363          2171  2 RETURN true
;  1364          2172  1 END;
```

```
                                                 OFFC 00000          .ENTRY  LBR$INSERT_TIME, Save R2,R3,R4,R5,R6,R7,R8,-: 2121
                                                                             R9,R10,R11                                    :
                                           5E    FF64   CE  9E 00002        MOVAB   -156(SP), SP                          :
                                           50      04   BC  D0 00007        MOVL    @CONTROL_INDEX, R0                    : 2146
                                                  0000G 30  0000B        BSBW    VALIDATE_CTL                          :
                                           18           50  E9 0000E        BLBC    STATUS, 1$                            :
                                           56      08   AC  D0 00011        MOVL    TXTRFA, R6                            : 2147
                                     0C    AE        66   06  28 00015        MOVC3   #6, (R6), LOCALRFA                    :
                                                       5E  DD 0001A        PUSHL   SP                                    : 2148
                                                     08    AE  9F 0001C        PUSHAB  BLOCKADDR                             :
                                                     01    DD  9F 0001F        PUSHL   #1                                    :
                                                     18    AE  9F 00021        PUSHAB  LOCALRFA                              :
                                           FDOE      CF        04  FB 00024        CALLS   #4, MAP_BLK_TO_MEM                    :
                                           67              50  E9 00029 1$:    BLBC    STATUS, 4$                            :
                                           50      04   A6  3C 0002C        MOVZWL  4(R6), R0                             : 2149
                                           50           12  C0 00030        ADDL2   #18, R0                               :
                                  00000200   8F              50  D1 00033        CMPL    R0, #512                              :
                                                             40  1A 0003A        BGTRU   3$                                    :
                                           50    0000G  CF    D0 0003C        MOVL    LBR$GL_CONTROL, R0                    : 2152
                                           51      0A   A0  D0 00041        MOVL    10(R0), R1                            : 2153
                                           52      04   A6  3C 00045        MOVZWL  4(R6), R2                             :
                                           52      04   AE  C0 00049        ADDL2   BLOCKADDR, R2                         : 2154
                                           50      04   A6  3C 0004D        MOVZWL  4(R6), R0                             :
                                           50      04   AE  C0 00051        ADDL2   BLOCKADDR, R0                         :
                                           50      0A   C0  00055        ADDL2   #10, R0                               :
                                           51      3C   A1  9A 00058        MOVZBL  60(R1), R1                            : 2156
                                           51           10  C0 0005C        ADDL2   #16, R1                               :
                                51         62           10  00  ED 0005F        CMPZV   #0, #16, (R2), R1                     :
                                                             08  13 00064        BEQL    2$                                    :
                                           50 00000000G  8F    D0 00066        MOVL    #LBR$_INVRFA, R0                      : 2157
                                                             04  00 0006D        RET                                           :
                                     60          0C   BC        08  28 0006E 2$:    MOVC3   #8, @NEWTIME, (R0)            : 2159
                                                 50            6E  D0 00073        MOVL    CACHENTRY, R0                 : 2160
                                                 0C    A0      01  88 00076        BISB2   #1, 12(R0)                    :
                                                             38  11 0007A        BRB     5$                                    : 2149
                                                 14    AE  80  8F  9B 0007C 3$:    MOVZBW  #128, HDRDESC                 : 2163
                                                 18    AE  1C  AE  9E 00081        MOVAB   HEADER, HDRDESC+4             : 2164
                                                         08  AE  9F 00086        PUSHAB  HDRLEN                        : 2165
                                                         18  AE  9F 00089        PUSHAB  HDRDESC                       :
                                                             56  DD 0008C        PUSHL   R6                            :
                                                 0000V   CF        03  FB 0008E        CALLS   #3, SET_MODULE                :
                                                             21    50  E9 00093 4$:    BLBC    STATUS, 6$                    :
                                     24    AE        0C   BC        08  28 00096        MOVC3   #8, @NEWTIME, HEADER+8        : 2166
                                     0C    AE          66          06  28 0009C        MOVC3   #6, (R6), LOCALRFA            : 2167
                                                             01  DD 000A1        PUSHL   #1                            : 2168
                                                         10  AE  9F 000A3        PUSHAB  LOCALRFA                      :
                                                         24  AE  9F 000A6        PUSHAB  HEADER                        :
                                                         14  AE  DD 000A9        PUSHL   HDRLEN                        :
                                                 F8E5    CF        04  FB 000AC        CALLS   #4, WRITE_RECORD              :
                                                             03    50  E9 000B1        BLBC    STATUS, 6$                    :
                                                             50    01  D0 000B4 5$:    MOVL    #1, R0                        : 2171
                                                                   04  00 000B7 6$:    RET                               : 2172

; Routine Size:  184 bytes,    Routine Base:  $CODE$ + 0B75
```

```
;  1366          2173  1 %SBTTL  'LBR$SET_MODULE';
;  1367          2174  1 GLOBAL ROUTINE lbr$set_module (control_index, txtrfa,
;  1368          2175  1                 bufdesc, buflen, updatedesc) =
;  1369          2176  2 BEGIN
;  1370          2177  2 !++
;  1371          2178  2 !
;  1372          2179  2 ! FUNCTIONAL DESCRIPTION:
;  1373          2180  2 !
;  1374          2181  2 !       This routine reads and optionally updates the module header
;  1375          2182  2 !       associated with the given RFA.
;  1376          2183  2 !
;  1377          2184  2 !
;  1378          2185  2 ! CALLING SEQUENCE:
;  1379          2186  2 !
;  1380          2187  2 !       status = lbr$set_module (control_index, txtrfa[,bufdesc,buflen,updatedesc])
;  1381          2188  2 !
;  1382          2189  2 ! INPUT PARAMETERS:
;  1383          2190  2 !
;  1384          2191  2 !       control_index           Address of library control index
;  1385          2192  2 !       txtrfa                  Address of rfa for module header
;  1386          2193  2 !       bufdesc                 Address of string descriptor for return
;  1387          2194  2 !       buflen                  Address to return length of header
;  1388          2195  2 !       updatedesc              Address of string descriptor to update module header user data
;  1389          2196  2 !
;  1390          2197  2 !--
;  1391          2198  2
;  1392          2199  2 BUILTIN
;  1393          2200  2     NULLPARAMETER;              ! True if parameter not specified
;  1394          2201  2
;  1395          2202  2 perform (validate_ctl (..control_index));             !Validate control index
;  1396          2203  2
;  1397          2204  3 BEGIN
;  1398          2205  3     BIND
;  1399          2206  3         context = .lbr$gl_control [lbr$l_ctxptr] : BBLOCK;
;  1400          2207  3
;  1401          2208  3     IF NOT NULLPARAMETER (5)                            !If updating header
;  1402          2209  4         AND (.context [ctx$v_oldlib]
;  1403          2210  4                 OR .context [ctx$v_ronly])
;  1404          2211  3         THEN RETURN lbr$_illop;
;  1405          2212  2     END;
;  1406          2213  2
;  1407        P 2214  2 perform (set_module (.txtrfa, (IF NOT NULLPARAMETER (3) THEN .bufdesc
;  1408        P 2215  2                                                ELSE 0),
;  1409        P 2216  2                               (IF NOT NULLPARAMETER (4) THEN .buflen
;  1410        P 2217  2                                                ELSE 0),
;  1411        P 2218  2                               (IF NOT NULLPARAMETER (5) THEN .updatedesc
;  1412          2219  2                                                ELSE 0)));
;  1413          2220  2 RETURN true
;  1414          2221  1 END;


                                    0FFC 00000          .ENTRY  LBR$SET_MODULE, Save R2,R3,R4,R5,R6,R7,R8,- ; 2174
                                                                R9,R10,R11                                      ;
                          50    04  BC  D0 00002          MOVL    @CONTROL_INDEX, R0                          ; 2202
```

```
                                 0000G 30 00006          BSBW    VALIDATE_CTL                                    ;
                              66   50 E9 00009          BLBC    STATUS, 9$                                      ;
                              50 0000G CF D0 0000C       MOVL    LBR$GL_CONTROL, R0                              ; 2206
                              50   0E A0 D0 00011       MOVL    14(R0), R0                                      ;
                              05      6C 91 00015       CMPB    (AP), #5                                        ; 2208
                                      17 1F 00018       BLSSU   2$                                              ;
                                   14 AC D5 0001A       TSTL    20(AP)                                          ;
                                      12 13 0001D       BEQL    2$                                              ;
                        05        04 A0 05 E0 0001F       BBS     #5, 4(R0), 1$                                   ; 2209
                                   04 A0 95 00024       TSTB    4(R0)                                           ; 2210
                                      08 18 00027       BGEQ    2$                                              ;
                              50 00000000G 8F D0 00029 1$:  MOVL    #LBR$_ILLOP, R0                                 ; 2211
                                         04 00030       RET                                                     ;
                              05      6C 91 00031 2$:   CMPB    (AP), #5                                        ; 2219
                                      0A 1F 00034       BLSSU   3$                                              ;
                                   14 AC D5 00036       TSTL    20(AP)                                          ;
                                      05 13 00039       BEQL    3$                                              ;
                                   14 AC DD 0003B       PUSHL   UPDATEDESC                                      ;
                                      02 11 0003E       BRB     4$                                              ;
                                      7E D4 00040 3$:   CLRL    -(SP)                                           ;
                              04      6C 91 00042 4$:   CMPB    (AP), #4                                        ;
                                      0A 1F 00045       BLSSU   5$                                              ;
                                   10 AC D5 00047       TSTL    16(AP)                                          ;
                                      05 13 0004A       BEQL    5$                                              ;
                                   10 AC DD 0004C       PUSHL   BUFLEN                                          ;
                                      02 11 0004F       BRB     6$                                              ;
                                      7E D4 00051 5$:   CLRL    -(SP)                                           ;
                              03      6C 91 00053 6$:   CMPB    (AP), #3                                        ;
                                      0A 1F 00056       BLSSU   7$                                              ;
                                   0C AC D5 00058       TSTL    12(AP)                                          ;
                                      05 13 0005B       BEQL    7$                                              ;
                                   0C AC DD 0005D       PUSHL   BUFDESC                                         ;
                                      02 11 00060       BRB     8$                                              ;
                                      7E D4 00062 7$:   CLRL    -(SP)                                           ;
                                   08 AC DD 00064 8$:   PUSHL   TXTRFA                                          ;
                        0000V CF         04 FB 00067       CALLS   #4, SET_MODULE                                  ;
                              03         50 E9 0006C       BLBC    STATUS, 9$                                      ;
                              50         01 D0 0006F       MOVL    #1, R0                                          ; 2220
                                         04 00072 9$:      RET                                                     ; 2221

; Routine Size:  115 bytes,    Routine Base:  $CODE$ + 0C2D
```

```
;  1416          2222  1 %SBTTL 'set_module';
;  1417          2223  1 GLOBAL ROUTINE set_module (txtrfa, bufdesc, buflen, updatedesc) =
;  1418          2224  2 BEGIN
;  1419          2225  2 !
;  1420          2226  2 ! Read and optionally update module header
;  1421          2227  2 !
;  1422          2228  2 MAP
;  1423          2229  2     txtrfa : REF BBLOCK,
;  1424          2230  2     bufdesc : REF BBLOCK,
;  1425          2231  2     updatedesc : REF BBLOCK;
;  1426          2232  2
;  1427          2233  2 LOCAL
;  1428          2234  2     recdesc : BBLOCK [dsc$c_s_bln],
;  1429          2235  2     header : REF BBLOCK,
;  1430          2236  2     descptr : REF BBLOCK,
;  1431          2237  2     faodesc : BBLOCK [dsc$c_s_bln],
;  1432          2238  2     localrfa : BBLOCK [rfa$c_length],
;  1433          2239  2     myheader : BBLOCK [lbr$c_maxhdrsiz],
;  1434          2240  2     mydesc   : BBLOCK [dsc$c_s_bln];
;  1435          2241  2
;  1436          2242  2 BUILTIN
;  1437          2243  2     NULLPARAMETER;
;  1438          2244  2
;  1439          2245  2 BIND
;  1440          2246  2     context = .lbr$gl_control [lbr$l_ctxptr] : BBLOCK, !Context block
;  1441          2247  2     reclen = recdesc [dsc$w_length] : WORD,
;  1442          2248  2     recaddr = recdesc [dsc$a_pointer] : REF BBLOCK;
;  1443          2249  2
;  1444          2250  2 IF NOT NULLPARAMETER (4)
;  1445          2251  2 THEN IF .context [ctx$v_oldlib]
;  1446          2252  2         OR .context [ctx$v_ronly]
;  1447          2253  2         THEN RETURN lbr$_illop;
;  1448          2254  2
;  1449          2255  2 CH$MOVE (rfa$c_length, .txtrfa, localrfa);
;  1450          2256  2 header = .lbr$gl_control [lbr$l_hdrptr];
;  1451          2257  2 IF NOT NULLPARAMETER (2)                              !bufdesc passed by caller?
;  1452          2258  2 THEN descptr = .bufdesc
;  1453          2259  3 ELSE BEGIN
;  1454          2260  3     mydesc [dsc$w_length] = lbr$c_maxhdrsiz;
;  1455          2261  3     mydesc [dsc$a_pointer] = myheader;
;  1456          2262  3     descptr = mydesc;
;  1457          2263  2     END;
;  1458          2264  2 IF .context [ctx$v_oldlib]
;  1459          2265  3 THEN BEGIN
;  1460          2266  3     BIND
;  1461          2267  3         eomodrfa = context [ctx$b_eomodrfa] : BBLOCK;
;  1462          2268  3
;  1463          2269  3     LOCAL
;  1464          2270  3         savendrfa : BBLOCK [rfa$c_length];
;  1465          2271  3
;  1466          2272  3     CH$MOVE (rfa$c_length, eomodrfa, savendrfa);         !Save end of module RFA in case reading
;  1467          2273  3     eomodrfa [rfa$l_vbn] = 0;                            !Disable end of module check
;  1468          2274  3     perform (read_old_record (localrfa, recdesc));
;  1469          2275  3     CH$MOVE (rfa$c_length, savendrfa, eomodrfa);         !Restore end of module RFA
;  1470          2276  3     IF .reclen NEQ omh$c_size                            !Must be the right length
;  1471          2277  3     THEN RETURN lbr$_invrfa;
;  1472          2278  3     reclen = mhd$c_objident+ofl$c_maxsymlng;             !Adjust record length
```

```
;  1473          2279  3      END
;  1474          2280  3  ELSE BEGIN
;  1475          2281  3      perform (read_record (localrfa, recdesc));          !Read the module header
;  1476          2282  3      IF .reclen NEQ mhd$c_mhdlen+.header [lhd$b_mhdusz]  !If header the wrong size
;  1477          2283  3      OR .recaddr [mhd$b_id] NEQ mhd$c_mhdid              ! or it doesn't look like a header
;  1478          2284  3      THEN RETURN lbr$_invrfa;
;  1479          2285  2      END;
;  1480          2286  2  IF NOT NULLPARAMETER (3)                             !Want header length returned?
;  1481          2287  2  THEN .buflen = .reclen;
;  1482          2288  2  CH$COPY (MINU (.reclen, .descptr [dsc$w_length]),  .recaddr, 0, !Copy header with 0 fill
;  1483          2289  2                .descptr [dsc$w_length], .descptr [dsc$a_pointer]);
;  1484          2290  2  IF .context [ctx$v_oldlib]                            !Old format library?
;  1485          2291  3  THEN BEGIN
;  1486          2292  3      LOCAL
;  1487          2293  3          datebuffer : BBLOCK [20],
;  1488          2294  3          datedesc : BBLOCK [dsc$c_s_bln],
;  1489          2295  3          datelen;
;  1490          2296  3      BIND
;  1491          2297  3          recptr = .descptr [dsc$a_pointer] : BBLOCK,
;  1492          2298  3          insertdate = recaddr [omh$t_insdate] : VECTOR [,WORD]; !Name old fmt insert date
;  1493          2299  3
;  1494          2300  3      CH$MOVE (.recptr [omh$b_midlng] + 1, recptr [omh$b_midlng], !Convert to new format
;  1495          2301  3          recptr [mhd$b_objidlng]);
;  1496          2302  3      recptr [mhd$b_objstat] = .recptr [omh$b_modatr];    !Copy module attributes
;  1497          2303  3      datedesc [dsc$w_length] = 20;
;  1498          2304  3      datedesc [dsc$a_pointer] = datebuffer;
;  1499          2305  3      datelen = 0;
;  1500          2306  3
;  1501          2307  3      faodesc [dsc$w_length] = .fao_old2newdate [0];
;  1502          2308  3      faodesc [dsc$a_pointer] = fao_old2newdate [1];
;  1503          2309  3
;  1504          2310  3  !     $FAO (CTRSTR = faodesc, OUTLEN = datelen,
;  1505          2311  3  !                 OUTBUF = datedesc, P1 = .insertdate [2],
;  1506          2312  3  !                 P2 = .months [(.insertdate [1] - 1) * 2],
;  1507          2313  3  !                 P3 = .insertdate [0]);
;  1508          2314  3      SYS$FAO (faodesc, datelen,
;  1509          2315  3                  datedesc, .insertdate [2],
;  1510          2316  3                  months [ (.insertdate [1] - ,)],
;  1511          2317  3                  .insertdate [0]);
;  1512          2318  3      datedesc [dsc$w_length] = .datelen;                 !Update descriptor
;  1513          2319  3      $BINTIM (TIMBUF = datedesc, TIMADR = recptr [mhd$l_datim]); !Now convert to binary
;  1514          2320  3      recptr [mhd$l_refcnt] = %X'FFFFFFFF';               !Set ref. count to a lot
;  1515          2321  2      END;
;  1516          2322  2  IF NOT NULLPARAMETER (4)                             !Updating the module header?
;  1517          2323  2  AND NOT .context [ctx$v_oldlib]                      ! and not old format library
;  1518          2324  3  THEN BEGIN
;  1519          2325  3      BIND
;  1520          2326  3          mhdusrdat = .descptr [dsc$a_pointer] + mhd$c_usrdat;
;  1521          2327  3      CH$COPY (MINU (.header [lhd$b_mhdusz], .updatedesc [dsc$w_length]),
;  1522          2328  3                .updatedesc [dsc$a_pointer], 0, .header [lhd$b_mhdusz], mhdusrdat);
;  1523          2329  3      CH$MOVE (rfa$c_length, .txtrfa, localrfa);  !Refresh RFA
;  1524          2330  3      perform (write_record (.reclen, .descptr [dsc$a_pointer], localrfa, true)); !Rewrite the header
;  1525          2331  2      END;
;  1526          2332  2  IF .reclen GTR .descptr [dsc$w_length]
;  1527          2333  2      THEN RETURN lbr$_hdrtrunc
;  1528          2334  2      ELSE RETURN true
;  1529          2335  1  END;                                                  !Of lbr$set_module
```

```
                                                        .EXTRN  SYS$BINTIM

                                        0FFC 00000      .ENTRY  SET_MODULE, Save R2,R3,R4,R5,R6,R7,R8,R9,-       ; 2223
                                                                R10,R11
                                5B      FAD2    CF  9E 00002    MOVAB   READ_OLD_RECORD, R11                     ;
                                5E      FF40    CE  9E 00007    MOVAB   -192(SP), SP                             ;
                                56      0000G   CF  D0 0000C    MOVL    LBR$GL_CONTROL, R6                       ; 2246
                                59        0E    A6  D0 00011    MOVL    14(R6), R9                               ;
                                04              6C  91 00015    CMPB    (AP), #4                                 ; 2250
                                                17  1F 00018    BLSSU   2$                                       ;
                                          10    AC  D5 0001A    TSTL    16(AP)                                   ;
                                                12  13 0001D    BEQL    2$                                       ;
                    05      04  A9              05  E0 0001F    BBS     #5, 4(R9), 1$                            ; 2251
                                          04    A9  95 00024    TSTB    4(R9)                                    ; 2252
                                                08  18 00027    BGEQ    2$                                       ;
                                50 00000000G    8F  D0 00029 1$:MOVL    #LBR$_ILLOP, R0                          ; 2253
                                                    04 00030    RET                                              ;
                E8  AD      04  BC              06  28 00031 2$:MOVC3   #6, @TXTRFA, LOCALRFA                    ; 2255
                                56        0A    A6  D0 00037    MOVL    10(R6), HEADER                           ; 2256
                                02              6C  91 0003B    CMPB    (AP), #2                                 ; 2257
                                                0B  1F 0003E    BLSSU   3$                                       ;
                                          08    AC  D5 00040    TSTL    8(AP)                                    ;
                                                06  13 00043    BEQL    3$                                       ;
                                5A        08    AC  D0 00045    MOVL    BUFDESC, DESCPTR                         ; 2258
                                                0E  11 00049    BRB     4$                                       ;
                          20    AE        80    8F  9B 0004B 3$:MOVZBW  #128, MYDESC                            ; 2260
                          24    AE        28    AE  9E 00050    MOVAB   MYHEADER, MYDESC+4                       ; 2261
                                5A        20    AE  9E 00055    MOVAB   MYDESC, DESCPTR                          ; 2262
                    28      04  A9              05  E1 00059 4$:BBC     #5, 4(R9), 5$                            ; 2264
                18  AE      22  A9              06  28 0005E    MOVC3   #6, 34(R9), SAVENDRFA                    ; 2272
                                          22    A9  D4 00064    CLRL    34(R9)                                   ; 2273
                                51        F8    AD  9E 00067    MOVAB   RECDESC, R1                              ; 2274
                                50        E8    AD  9E 0006B    MOVAB   LOCALRFA, R0                             ;
                                                6B  16 0006F    JSB     READ_OLD_RECORD                          ;
                                                50  E9 00071    BLBC    STATUS, 6$                               ;
                22  A9          18  AE          06  28 00074    MOVC3   #6, SAVENDRFA, 34(R9)                    ; 2275
                                          1C    F8  AD  B1 0007A    CMPW    RECLEN, #28                          ; 2276
                                                30  12 0007E    BNEQ    8$                                       ;
                          F8    AD              21  B0 00080    MOVW    #33, RECLEN                              ; 2278
                                                32  11 00084    BRB     9$                                       ; 2264
                                51        F8    AD  9E 00086 5$:MOVAB   RECDESC, R1                              ; 2281
                                50        E8    AD  9E 0008A    MOVAB   LOCALRFA, R0                             ;
                                        FEAE    CB  16 0008E    JSB     READ_RECORD                              ;
                                01              50  E9 00092 6$:BLBS    STATUS, 7$                               ;
                                                    04 00095    RET                                              ;
                                50        3C    A6  9A 00096 7$:MOVZBL  60(HEADER), R0                           ; 2282
                                50              10  C0 0009A    ADDL2   #16, R0                                  ;
 50             F8  AD          10              00  ED 0009D    CMPZV   #0, #16, RECLEN, R0                      ;
                                                0B  12 000A3    BNEQ    8$                                       ;
                                50        FC    AD  D0 000A5    MOVL    RECADDR, R0                              ; 2283
                          AD    8F        01    A0  91 000A9    CMPB    1(R0), #173                              ;
                                                08  13 000AE    BEQL    9$                                       ;
                                50 00000000G    8F  D0 000B0 8$:MOVL    #LBR$_INVRFA, R0                         ; 2284
                                                    04 000B7    RET                                              ;
```

```
                          03        6C  91 000B8 9$:     CMPB    (AP), #3                                  ; 2286
                                    0A  1F 000BB         BLSSU   10$                                       ;
                               0C   AC  D5 000BD         TSTL    12(AP)                                    ;
                                    05  13 000C0         BEQL    10$                                       ;
                   0C     BC   F8   AD  3C 000C2         MOVZWL  RECLEN, @BUFLEN                           ; 2287
                          50   F8   AD  3C 000C7 10$:    MOVZWL  RECLEN, R0                                ; 2288
                          50        6A  B1 000CB         CMPW    (DESCPTR), R0                             ;
                                    03  1E 000CE         BGEQU   11$                                       ;
                          50        6A  3C 000D0         MOVZWL  (DESCPTR), R0                             ;
                          57   04   AA  D0 000D3 11$:    MOVL    4(DESCPTR), R7                            ; 2289
     6A        00         FC   BD   50  2C 000D7         MOVC5   R0, @RECADDR, #0, (DESCPTR), (R7)         ;
                                    67     000DD                                                           ;
               62         04   A9   05  E1 000DE         BBC     #5, 4(R9), 12$                            ; 2290
               58         FC   AD   06  C1 000E3         ADDL3   #6, RECADDR, R8                           ; 2298
                          50   0C   A7  9A 000E8         MOVZBL  12(R7), R0                                ; 2300
                                    50  D6 000EC         INCL    R0                                        ;
          11   A7         0C   A7   50  28 000EE         MOVC3   R0, 12(R7), 17(R7)                        ; 2301
                          10   A7   01  A7  90 000F4     MOVB    1(R7), 16(R7)                             ; 2302
                          04   AE       14  B0 000F9     MOVW    #20, DATEDESC                             ; 2303
                          08   AE   0C  AE  9E 000FD     MOVAB   DATEBUFFER, DATEDESC+4                    ; 2304
                                    6E  D4 00102         CLRL    DATELEN                                   ; 2305
                    F0    AD  F258  CF  9B 00104         MOVZBW  FAO_OLD2NEWDATE, FAODESC                  ; 2307
                    F4    AD  F253  CF  9E 0010A         MOVAB   FAO_OLD2NEWDATE+1, FAODESC+4              ; 2308
                          7E        68  3C 00110         MOVZWL  (R8), -(SP)                               ; 2317
                          50   02   A8  3C 00113         MOVZWL  2(R8), R0                                 ; 2316
                         F258  CF40     DF 00117         PUSHAL  MONTHS-4[R0]                              ;
                          7E   04   A8  3C 0011C         MOVZWL  4(R8), -(SP)                              ;
                               10   AE  9F 00120         PUSHAB  DATEDESC                                  ; 2314
                               10   AE  9F 00123         PUSHAB  DATELEN                                   ;
                               F0   AD  9F 00126         PUSHAB  FAODESC                                   ;
              00000000G   00        06  FB 00129         CALLS   #6, SYS$FAO                               ; 2316
                    04    AE        6E  B0 00130         MOVW    DATELEN, DATEDESC                         ; 2318
                               08   A7  9F 00134         PUSHAB  8(R7)                                     ; 2319
                               08   AE  9F 00137         PUSHAB  DATEDESC                                  ;
              00000000G   00        02  FB 0013A         CALLS   #2, SYS$BINTIM                            ;
                    04    A7        01  CE 00141         MNEGL   #1, 4(R7)                                 ; 2320
                          04        6C  91 00145 12$:    CMPB    (AP), #4                                  ; 2322
                                    3F  1F 00148         BLSSU   14$                                       ;
                               10   AC  D5 0014A         TSTL    16(AP)                                    ;
                                    3A  13 0014D         BEQL    14$                                       ;
               35         04   A9   05  E0 0014F         BBS     #5, 4(R9), 14$                            ; 2323
                          50   10   AC  D0 00154         MOVL    UPDATEDESC, R0                            ; 2327
                          51   3C   A6  9A 00158         MOVZBL  60(HEADER), R1                            ;
                          51        60  B1 0015C         CMPW    (R0), R1                                  ;
                                    03  1E 0015F         BGEQU   13$                                       ;
                          51        60  3C 00161         MOVZWL  (R0), R1                                  ;
                          52   3C   A6  9A 00164 13$:    MOVZBL  60(HEADER), R2                            ; 2328
     52        00         04   B0   51  2C 00168         MOVC5   R1, @4(R0), #0, R2, 16(R7)                ; 2327
                               10   A7     0016E                                                           ;
          E8   AD         04   BC   06  28 00170         MOVC3   #6, @TXTRFA, LOCALRFA                     ; 2329
                                    01  DD 00176         PUSHL   #1                                        ; 2330
                               E8   AD  9F 00178         PUSHAB  LOCALRFA                                  ;
                                    57  DD 0017B         PUSHL   R7                                        ;
                          7E   F8   AD  3C 0017D         MOVZWL  RECLEN, -(SP)                             ;
                    FD93  CB        04  FB 00181         CALLS   #4, WRITE_RECORD                          ;
                          11        50  E9 00186         BLBC    STATUS, 16$                               ;
                          6A   F8   AD  B1 00189 14$:    CMPW    RECLEN, (DESCPTR)                         ; 2332
```

```
                                 08  1B 0018D          BLEQU   15$                                                    ;
                      50 00000000G  8F  D0 0018F          MOVL    #LBR$_HDRTRUNC, R0                                     ; 2334
                                    04 00196            RET                                                            ;
                      50            01  D0 00197 15$:    MOVL    #1, R0                                                 ;
                                    04 0019A 16$:       RET                                                            ; 2335

; Routine Size:  411 bytes,    Routine Base:  $CODE$ + 0CA0
```

```
;  1531          2336  1 %SBTTL  'LBR$PUT_HISTORY';
;  1532          2337  1 GLOBAL ROUTINE lbr$put_history (control_index, record_desc) =
;  1533          2338  2 BEGIN
;  1534          2339  2 !+++
;  1535          2340  2 !
;  1536          2341  2 !   FUNCTIONAL DESCRIPTION:
;  1537          2342  2 !
;  1538          2343  2 !        Add an update history record to the end of the update history list.
;  1539          2344  2 !        If the list is full, delete the oldest record before the addition.
;  1540          2345  2 !
;  1541          2346  2 !
;  1542          2347  2 !   CALLING SEQUENCE:
;  1543          2348  2 !
;  1544          2349  2 !        status = lbr$put_history (control_index, record_desc)
;  1545          2350  2 !
;  1546          2351  2 !
;  1547          2352  2 !   INPUT PARAMETERS:
;  1548          2353  2 !
;  1549          2354  2 !        control_index   is the index returned from lbr$ini_control
;  1550          2355  2 !        record_desc     is the address of string descriptor for the
;  1551          2356  2 !                         record to be added to the library update history
;  1552          2357  2 !
;  1553          2358  2 !   ROUTINE VALUE:
;  1554          2359  2 !
;  1555          2360  2 !        lbr$_illop              Illegal operation for access requested
;  1556          2361  2 !        lbr$_intrnlerr          Internal librarian error
;  1557          2362  2 !        lbr$_normal             Normal exit
;  1558          2363  2 !        lbr$_nohistory          This library does not have an update history
;  1559          2364  2 !        lbr$_reclng             Record length was greater than lbr$c_maxrecsiz
;  1560          2365  2 !
;  1561          2366  2 !---
;  1562          2367  2 perform (validate_ctl (..control_index));       ! Validate the control index
;  1563          2368  3 BEGIN
;  1564          2369  3 BIND
;  1565          2370  3     header = .lbr$gl_control [lbr$l_hdrptr] : BBLOCK;
;  1566          2371  3
;  1567          2372  3 IF .header [lhd$w_maxluhrec] EQL 0      ! History not maintained for this library
;  1568          2373  3 THEN RETURN lbr$_nohistory;
;  1569          2374  3 IF lbr$gl_control [lbr$b_func] EQL lbr$c_read   ! Shouldn't be here on read
;  1570          2375  3 THEN RETURN lbr$_illop;
;  1571          2376  3 IF .header [lhd$w_numluhrec] GTR .header [lhd$w_maxluhrec]
;  1572          2377  3 THEN RETURN lbr$_intrnlerr;              ! somehow there are more than allowed
;  1573          2378  3
;  1574          2379  3 IF .header [lhd$w_numluhrec] EQL .header [lhd$w_maxluhrec]
;  1575          2380  3 THEN perform ( delete_luhrecord () );   ! History full, so drop oldest record
;  1576          2381  3
;  1577          2382  3 perform (add_luhrecord ( .record_desc));
;  1578          2383  3
;  1579          2384  3 RETURN lbr$_normal;                     ! return success
;  1580          2385  2 END;
;  1581          2386  1 END;    ! lbr$put_history


                                0FFC 00000        .ENTRY  LBR$PUT_HISTORY, Save R2,R3,R4,R5,R6,R7,R8,-; 2337
```

```
                                                         R9,R10,R11                    ;
      50          04   BC  D0 00002           MOVL       @CONTROL_INDEX, R0            ; 2367
                    0000G  30 00006           BSBW       VALIDATE_CTL                  ;
      51              50   E9 00009           BLBC       STATUS, 5$                    ;
      51        0000G  CF  D0 0000C           MOVL       LBR$GL_CONTROL, R1            ; 2370
      50          0A   A1  D0 00011           MOVL       10(R1), R0                    ;
      52          7C   A0  3C 00015           MOVZWL     124(R0), R2                   ; 2372
                      08   12 00019           BNEQ       1$                            ;
      50 00000000G     8F  D0 0001B           MOVL       #LBR$_NOHISTORY, R0           ; 2373
                           04 00022           RET                                      ;
      51              03   C0 00023 1$:       ADDL2      #3, R1                        ; 2374
      01              51   D1 00026           CMPL       R1, #1                        ;
                      08   12 00029           BNEQ       2$                            ;
      50 00000000G     8F  D0 0002B           MOVL       #LBR$_ILLOP, R0               ; 2375
                           04 00032           RET                                      ;
      52          7E   A0  B1 00033 2$:       CMPW       126(R0), R2                   ; 2376
                      08   1B 00037           BLEQU      3$                            ;
      50 00000000G     8F  D0 00039           MOVL       #LBR$_INTRNLERR, R0           ; 2377
                           04 00040           RET                                      ;
                      08   12 00041 3$:       BNEQ       4$                            ; 2379
0000V  CF             00   FB 00043           CALLS      #0, DELETE_LUHRECORD          ; 2380
       12             50   E9 00048           BLBC       STATUS, 5$                    ;
                  08   AC  DD 0004B 4$:       PUSHL      RECORD_DESC                   ; 2382
0000V  CF             01   FB 0004E           CALLS      #1, ADD_LUHRECORD             ;
       07             50   E9 00053           BLBC       STATUS, 5$                    ;
      50 00000000G     8F  D0 00056           MOVL       #LBR$_NORMAL, R0              ; 2384
                           04 0005D 5$:       RET                                      ; 2386

; Routine Size:  94 bytes,    Routine Base:  $CODE$ + 0E3B

; 1582          2387  1
```

```
;  1584          2388  1 %SBTTL  'add_luhrecord';
;  1585          2389  1 ROUTINE add_luhrecord ( rec_desc ) =
;  1586          2390  2 BEGIN
;  1587          2391  2 !+++
;  1588          2392  2 !
;  1589          2393  2 !       This routine copies the library update history record from the
;  1590          2394  2 !       descriptor at address rec_desc to the end of the linked list of
;  1591          2395  2 !       library update history records.
;  1592          2396  2 !
;  1593          2397  2 !---
;  1594          2398  2 MAP
;  1595          2399  2     rec_desc : REF BBLOCK;      ! caller's descriptor for LUH record
;  1596          2400  2 BIND
;  1597          2401  2     context = .lbr$gl_control [lbr$l_ctxptr] : BBLOCK,  ! Context block
;  1598          2402  2     header = .lbr$gl_control [lbr$l_hdrptr] : BBLOCK,   ! library header block
;  1599          2403  2     endrfa = header [lhd$b_endluhrfa] : BBLOCK,         ! rfa of end of youngest LUH record in list
;  1600          2404  2     endluhvbn = endrfa [rfa$l_vbn],                     ! VBN of block containing end of luh list
;  1601          2405  2     endoffset = endrfa [rfa$w_offset] : WORD,           ! offset to end of LUH list
;  1602          2406  2     recrdlen = rec_desc [dsc$w_length]: WORD,           ! length of LUH record
;  1603          2407  2     recrd = rec_desc [dsc$a_pointer] : BBLOCK;          ! starting location of LUH record
;  1604          2408  2 LOCAL
;  1605          2409  2     cache_entry : REF BBLOCK,   ! cache entry of new block
;  1606          2410  2     cpyrecadr,                  ! how much of the record is left to copy into LUH block
;  1607          2411  2     endblkadr : REF BBLOCK,     ! address of cached end LUH block
;  1608          2412  2     endvbn,                     ! VBN of first free space in history blocks
;  1609          2413  2     offset,                     ! offset to first available space
;  1610          2414  2     rec : REF BBLOCK,           ! address where LUH record will be stored
;  1611          2415  2     reclenlft;                  ! address of remainder of record to be copied in.
;  1612          2416  2
;  1613          2417  2 IF .recrdlen GTR lbr$c_maxrecsiz       ! record too long
;  1614          2418  2 THEN RETURN lbr$_reclng;
;  1615          2419  2
;  1616          2420  2 endvbn = .endluhvbn;
;  1617          2421  2 offset = .endoffset;
;  1618          2422  2 IF .header [lhd$w_numluhrec] EQL 0
;  1619          2423  2 THEN
;  1620          2424  3     BEGIN       ! Get some space to store record
;  1621          2425  3     BIND
;  1622          2426  3         begluhrfa = header [lhd$b_begluhrfa] : BBLOCK,
;  1623          2427  3         begvbn = begluhrfa [rfa$l_vbn],
;  1624          2428  3         begoffset = begluhrfa [rfa$w_offset] : WORD;
;  1625          2429  3     LOCAL
;  1626          2430  3         newvbn,
;  1627          2431  3         newblkadr;
;  1628          2432  3     IF .begvbn OR .endluhvbn THEN RETURN lbr$_intrnlerr; ! both of these should be 0
;  1629          2433  3                                                 ! logic error may result in some blocks being lost
;  1630          2434  3     !
;  1631          2435  3     !    Get a free block, cache it and set header pointers to it's vbn.
;  1632          2436  3     !
;  1633          2437  3     perform ( alloc_block (newvbn, newblkadr) );
;  1634          2438  3     CH$FILL (0, luh$c_length, .newblkadr);
;  1635          2439  3     add_cache (.newvbn, cache_entry);
;  1636          2440  3     cache_entry [cache$l_address] = .newblkadr;
;  1637          2441  3     cache_entry [cache$v_data] = true;
;  1638          2442  3     cache_entry [cache$v_dirty] = true;
;  1639          2443  3     endblkadr = .newblkadr;
;  1640          2444  3     endvbn = .newvbn;
```

```
;  1641     2445  3     begvbn = .newvbn;
;  1642     2446  3     begoffset = 0;
;  1643     2447  3     END
;  1644     2448  2 ELSE
;  1645     2449  2     !
;  1646     2450  2     !    Find the last block in the chain of history records and cache
;  1647     2451  2     !
;  1648     2452  2     BEGIN
;  1649     2453  2     perform ( find_block (.endvbn, endblkadr, cache_entry) );   ! Cache end of history block
;  1650     2454  2     cache_entry [cache$v_data] = true;  ! Mark as data
;  1651     2455  2     cache_entry [cache$v_dirty] = true; ! Mark to write
;  1652     2456  2     END;
;  1653     2457  2
;  1654     2458  2 IF .offset GTR luh$c_datfldlen THEN RETURN lbr$_intrnlerr;      ! Offset can't point beyond end of record
;  1655     2459  2 IF luh$c_rechdrlen GTR luh$c_datfldlen - .offset        ! if there isn't enough room left for record header
;  1656     2460  2 THEN
;  1657     2461  2     BEGIN       ! not enough room left for the record length so get new block
;  1658     2462  2     LOCAL
;  1659     2463  2         newvbn,
;  1660     2464  2         newblkadr;
;  1661     2465  2     perform ( alloc_block (newvbn, newblkadr) );
;  1662     2466  2     CH$FILL (0, luh$c_length, .newblkadr);       ! zero out whole block
;  1663     2467  2     add_cache (.newvbn, cache_entry);            ! cache it
;  1664     2468  2     cache_entry [cache$l_address] = .newblkadr; ! fill in cache entry
;  1665     2469  2     cache_entry [cache$v_data] = true;
;  1666     2470  2     cache_entry [cache$v_dirty] = true;
;  1667     2471  2     endblkadr[luh$l_nxtluhblk] = .newvbn;        ! link it in to list of LUH record blocks
;  1668     2472  2     endblkadr = .newblkadr;                      ! Update rfa of free space.
;  1669     2473  2     endvbn = .newvbn;
;  1670     2474  3     offset = 0;
;  1671     2475  2     END;
;  1672     2476  2
;  1673     2477  2 !
;  1674     2478  2 !    Each update history record starts with a word to mark it for error checking
;  1675     2479  2 !    followed by a word containing the length of the record.
;  1676     2480  2 !
;  1677     2481  2 rec = .endblkadr + luh$c_data + .offset;          ! New record begins at end of last
;  1678     2482  2 rec [luh$w_rechdr] = luh$c_rechdrmrk;             ! Mark the new record
;  1679     2483  2 rec [luh$w_reclen] = .recrdlen;                  ! Store the length
;  1680     2484  2 reclenlft = .recrdlen;                          ! Set length to copy entire record
;  1681     2485  2 offset = .offset + luh$c_rechdrlen;             ! Bump offset to account for mark and length words
;  1682     2486  2 cpyrecadr = .recrd;                             ! Begin copy from start of record
;  1683     2487  2 WHILE ( .reclenlft GTR 0 ) DO                   ! While there is more to copy
;  1684     2488  3     BEGIN
;  1685     2489  3     LOCAL
;  1686     2490  3         cpylen;                      ! How much to copy with each move
;  1687     2491  4     If ( (.offset EQL luh$c_datfldlen) AND (.reclenlft GTR 0) )
;  1688     2492  3     THEN
;  1689     2493  4         BEGIN   ! used up last of that block, get next ready
;  1690     2494  4         LOCAL
;  1691     2495  4             newvbn,
;  1692     2496  4             newblkadr;
;  1693     2497  4         perform ( alloc_block (newvbn, newblkadr) );
;  1694     2498  4         CH$FILL (0, luh$c_length, .newblkadr);
;  1695     2499  4         add_cache (.newvbn, cache_entry);
;  1696     2500  4         cache_entry [cache$l_address] = .newblkadr;
;  1697     2501  4         cache_entry [cache$v_data] = true;
```

```
;1698          2502  4          cache_entry [cache$v_dirty] = true;
;1699          2503  4          endblkadr[luh$l_nxtluhblk] = .newvbn;   ! Link the new block to the last
;1700          2504  4          endblkadr = .newblkadr;                 ! Use new block
;1701          2505  4          endvbn = .newvbn;
;1702          2506  4          offset = 0;                             ! Reset offset to beginning of new block
;1703          2507  3          END;
;1704          2508  3      cpylen = MIN( luh$c_datfldlen - .offset, .reclenlft);       ! Either copy enough record to fill the rest
;1705          2509  3                                                                 ! or copy to the end of the record if it wil
;1706          2510  3      CH$MOVE( .cpylen, .cpyrecadr, .endblkadr + luh$c_data +.offset);    ! Copy record
;1707          2511  3      cpyrecadr = .cpyrecadr + .cpylen;
;1708          2512  3      reclenlft = .reclenlft - .cpylen;
;1709          2513  3      offset = .offset + .cpylen;
;1710          2514  2      END;        ! WHILE copying record
;1711          2515  2
;1712          2516  2 endoffset = .offset;             ! Update the header to point to end of record
;1713          2517  2 endluhvbn = .endvbn;             ! Update header to point to the last block in the linked list
;1714          2518  2 header [lhd$w_numluhrec] = .header [lhd$w_numluhrec] + 1;       ! There is one more record in the history
;1715          2519  2 context [ctx$v_hdrdirty] = true;        ! Make sure header is written out when cache is deallocated
;1716          2520  2 RETURN true;
;1717          2521  1 END;    ! add_luhrecord


                                OFFC 00000 ADD_LUHRECORD:
                                           .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11        ; 2389
                          5E         24  C2 00002         SUBL2   #36, SP                     ;
                          50   0000G  CF  D0 00005         MOVL    LBR$GL_CONTROL, R0          ; 2401
                                0E   A0  DD 0000A         PUSHL   14(R0)                      ;
                          57    0A   A0  D0 0000D         MOVL    10(R0), R7                  ; 2402
                          59  0086   C7  9E 00011         MOVAB   134(R7), R9                 ; 2403
                    58      04   AC      04  C1 00016     ADDL3   #4, REC_DESC, R8            ; 2407
                        0800   8F      04  BC B1 0001B     CMPW    @REC_DESC, #2048            ; 2417
                                        08  1B 00021     BLEQU   1$                          ;
                          50 00000000G  8F  D0 00023     MOVL    #LBR$_RECLNG, R0            ; 2418
                                        04  0002A     RET                                 ;
                          5B            69  D0 0002B 1$:  MOVL    (R9), ENDVBN                ; 2420
                          56      04    A9  3C 0002E     MOVZWL  4(R9), OFFSET               ; 2421
                                7E      A7  B5 00032     TSTW    126(R7)                     ; 2422
                                        4C  12 00035     BNEQ    2$                          ;
                          5A  0080      C7  9E 00037     MOVAB   128(R7), R10                ; 2426
                          66            6A  E8 0003C     BLBS    (R10), 4$                   ; 2432
                          63            69  E8 0003F     BLBS    (R9), 4$                    ;
                          51      08    AE  9E 00042     MOVAB   NEWBLKADR, R1               ; 2437
                          50      0C    AE  9E 00046     MOVAB   NEWVBN, R0                  ;
                                  0000G 30  00004A     BSBW    ALLOC_BLOCK                 ;
                          75            50  E9 0004D     BLBC    STATUS, 6$                  ;
0200    8F        00        6E          00  2C 00050     MOVC5   #0, (SP), #0, #512, @NEWBLKADR      ; 2438
                                    08  BE       00057
                          51      24    AE  9E 00059     MOVAB   CACHE_ENTRY, R1             ; 2439
                          50      0C    AE  D0 0005D     MOVL    NEWVBN, R0                  ;
                                  0000G 30  00061     BSBW    ADD_CACHE                   ;
                          50      24    AE  D0 00064     MOVL    CACHE_ENTRY, R0             ; 2440
                    08    A0      08    AE  D0 00068     MOVL    NEWBLKADR, 8(R0)            ;
                    0C    A0            03  88 0006D     BISB2   #3, 12(R0)                  ; 2442
                    10    AE      08    AE  D0 00071     MOVL    NEWBLKADR, ENDBLKADR        ; 2443
```

```
                                5B        0C   AE   D0 00076          MOVL    NEWVBN, ENDVBN                          ; 2444
                                6A        0C   AE   D0 0007A          MOVL    NEWVBN, (R10)                           ; 2445
                                          04   AA   B4 0007E          CLRW    4(R10)                                  ; 2446
                                          19        11 00081          BRB     3$                                      ; 2422
                                52        24   AE   9E 00083 2$:     MOVAB   CACHE_ENTRY, R2                         ; 2453
                                51        10   AE   9E 00087          MOVAB   ENDBLKADR, R1                           ;
                                50              5B   D0 0008B          MOVL    ENDVBN, R0                              ;
                                                0000G 30 0008E          BSBW    FIND_BLOCK                              ;
                                31              50   E9 00091          BLBC    STATUS, 6$                              ;
                                50        24   AE   D0 00094          MOVL    CACHE_ENTRY, R0                         ; 2454
                             0C AO              03   88 00098          BISB2   #3, 12(R0)                              ; 2455
                      000001FA  8F              56   D1 0009C 3$:     CMPL    OFFSET, #506                            ; 2458
                                                08   15 000A3          BLEQ    5$                                      ;
                                50 00000000G   8F   D0 000A5 4$:     MOVL    #LBR$_INTRNLERR, R0                     ;
                                                04   000AC          RET                                             ;
                                50        04   A6   9E 000AD 5$:     MOVAB   4(R6), R0                               ; 2459
                      000001FA  8F              50   D1 000B1          CMPL    R0, #506                                ;
                                                3F   15 000B8          BLEQ    7$                                      ;
                                51        14   AE   9E 000BA          MOVAB   NEWBLKADR, R1                           ; 2465
                                50        18   AE   9E 000BE          MOVAB   NEWVBN, R0                              ;
                                                0000G 30 000C2          BSBW    ALLOC_BLOCK                             ;
                                6D              50   E9 000C5 6$:     BLBC    STATUS, 9$                              ;
  0200   8F          00            6E              00   2C 000C8          MOVC5   #0, (SP), #0, #512, @NEWBLKADR          ; 2466
                                          14   BE        000CF          ;                                               ;
                                51        24   AE   9E 000D1          MOVAB   CACHE_ENTRY, R1                         ; 2467
                                50        18   AE   D0 000D5          MOVL    NEWVBN, R0                              ;
                                                0000G 30 000D9          BSBW    ADD_CACHE                               ;
                                50        24   AE   D0 000DC          MOVL    CACHE_ENTRY, R0                         ; 2468
                             08 A0        14   AE   D0 000E0          MOVL    NEWBLKADR, 8(R0)                        ;
                             0C A0              03   88 000E5          BISB2   #3, 12(R0)                              ; 2470
                             10 BE        18   AE   D0 000E9          MOVL    NEWVBN, @ENDBLKADR                      ; 2471
                             10 AE        14   AE   D0 000EE          MOVL    NEWBLKADR, ENDBLKADR                    ; 2472
                                5B        18   AE   D0 000F3          MOVL    NEWVBN, ENDVBN                          ; 2473
                                          56   D4   000F7          CLRL    OFFSET                                  ; 2474
                          50              56   10   AE   C1 000F9 7$:     ADDL3   ENDBLKADR, OFFSET, R0                   ; 2481
                                50        06   C0   000FE          ADDL2   #6, REC                                 ;
                                60      ABBA   8F   B0 00101          MOVW    #-21574, (REC)                          ; 2482
                             02 A0        04   BC   B0 00106          MOVW    @REC_DESC, 2(REC)                       ; 2483
                                5A        04   BC   3C 0010B          MOVZWL  @REC_DESC, RECLENLFT                    ; 2484
                                56        04   C0   0010F          ADDL2   #4, OFFSET                              ; 2485
                             04 AE              68   D0 00112          MOVL    (R8), CPYRECADR                         ; 2486
                                                50   D4 00116 8$:     CLRL    R0                                      ; 2487
                                                5A   D5 00118          TSTL    RECLENLFT                               ;
                                                77   15 0011A          BLEQ    12$                                     ;
                                                50   D6 0011C          INCL    R0                                      ;
                          000001FA  8F              56   D1 0011E          CMPL    OFFSET, #506                            ; 2491
                                                42   12 00125          BNEQ    10$                                     ;
                                3F              50   E9 00127          BLBC    R0, 10$                                 ;
                                51        1C   AE   9E 0012A          MOVAB   NEWBLKADR, R1                           ; 2497
                                50        20   AE   9E 0012E          MOVAB   NEWVBN, R0                              ;
                                                0000G 30 00132          BSBW    ALLOC_BLOCK                             ;
                                6F              50   E9 00135 9$:     BLBC    STATUS, 13$                             ;
  0200   8F          00            6E              00   2C 00138          MOVC5   #0, (SP), #0, #512, @NEWBLKADR          ; 2498
                                          1C   BE        0013F          ;                                               ;
                                51        24   AE   9E 00141          MOVAB   CACHE_ENTRY, R1                         ; 2499
                                50        20   AE   D0 00145          MOVL    NEWVBN, R0                              ;
                                                0000G 30 00149          BSBW    ADD_CACHE                               ;
```

```
                                50        24   AE   D0 0014C          MOVL    CACHE_ENTRY, R0                                  ; 2500
                             08 A0        1C   AE   D0 00150          MOVL    NEWBLKADR, 8(R0)                                 ;
                             0C A0             03   88 00155          BISB2   #3, 12(R0)                                       ; 2502
                             10 BE        20   AE   D0 00159          MOVL    NEWVBN, @ENDBLKADR                               ; 2503
                             10 AE        1C   AE   D0 0015E          MOVL    NEWBLKADR, ENDBLKADR                             ; 2504
                                5B        20   AE   D0 00163          MOVL    NEWVBN, ENDVBN                                   ; 2505
                                          56   D4   00167          CLRL    OFFSET                                           ; 2506
                          50 000001FA     8F             56   C3 00169 10$:   SUBL3   OFFSET, #506, R0                        ; 2508
                                5A                       50   D1 00171          CMPL    R0, RECLENLFT                            ;
                                                         03   15 00174          BLEQ    11$                                      ;
                                50                       5A   D0 00176          MOVL    RECLENLFT, R0                            ;
                                58                       50   D0 00179 11$:    MOVL    R0, CPYLEN                               ;
                             50                          56   C1 0017C          ADDL3   ENDBLKADR, OFFSET, R0                    ; 2510
                          06 A0        04   BE             58   28 00181          MOVC3   CPYLEN, @CPYRECADR, 6(R0)                ;
                             04 AE                       58   C0 00187          ADDL2   CPYLEN, CPYRECADR                        ; 2511
                                5A                       58   C2 0018B          SUBL2   CPYLEN, RECLENLFT                        ; 2512
                                56                       58   C0 0018E          ADDL2   CPYLEN, OFFSET                           ; 2513
                                                         83   11 00191          BRB     8$                                       ; 2487
                             04 A9                       56   B0 00193 12$:    MOVW    OFFSET, 4(R9)                            ; 2516
                                69                       5B   D0 00197          MOVL    ENDVBN, (R9)                             ; 2517
                                                   7E    A7   B6 0019A          INCW    126(R7)                                  ; 2518
                             50                    6E         04   C1 0019D          ADDL3   #4, (SP), R0                             ; 2519
                                60                            08   88 001A1          BISB2   #8, (R0)                                 ;
                                50                            01   D0 001A4          MOVL    #1, R0                                   ; 2520
                                                              04   001A7 13$:    RET                                              ; 2521

; Routine Size:  424 bytes,    Routine Base:  $CODE$ + 0E99

; 1718          2522  1
```

```
;1720     2523  1 %SBTTL  'delete_luhrecord';
;1721     2524  1 ROUTINE delete_luhrecord =
;1722     2525  2 BEGIN
;1723     2526  2 !+++
;1724     2527  2 !
;1725     2528  2 !       Remove the oldest LUH record by moving offset to bypass it.  If record
;1726     2529  2 !       crosses block boundaries than return freed blocks to library header
;1727     2530  2 !       free list.  If there is only one record in the history then the history
;1728     2531  2 !       is completely emptied with all blocks returned and all pointers zeroed.
;1729     2532  2 !
;1730     2533  2 !---
;1731     2534  2 BIND
;1732     2535  2     context = .lbr$gl_control [lbr$l_ctxptr] : BBLOCK,  ! Context block
;1733     2536  2     header = .lbr$gl_control [lbr$l_hdrptr] : BBLOCK,
;1734     2537  2     begluhrfa = header [lhd$b_begluhrfa] : BBLOCK,
;1735     2538  2     begvbn = begluhrfa [rfa$l_vbn],
;1736     2539  2     begoffset = begluhrfa [rfa$w_offset] : WORD;
;1737     2540  2
;1738     2541  2 !
;1739     2542  2 !   Check if there is only one record in history.
;1740     2543  2 !
;1741     2544  2 IF .header [lhd$w_numluhrec] EQL 1
;1742     2545  2 THEN
;1743     2546  3     BEGIN       ! Return all blocks in history
;1744     2547  3     BIND
;1745     2548  3         endluhrfa = header [lhd$b_endluhrfa] : BBLOCK,
;1746     2549  3         endoffset = endluhrfa [rfa$w_offset] : WORD;
;1747     2550  3     LOCAL
;1748     2551  3         blkadr : REF BBLOCK,
;1749     2552  3         cache_entry : REF BBLOCK,
;1750     2553  3         vbn;
;1751     2554  3
;1752     2555  3     vbn = .begvbn;      ! First vbn in history linked list
;1753     2556  3     DO          ! As long as there are more luh blocks in list
;1754     2557  4         BEGIN                       ! keep deallocating them.
;1755     2558  4         LOCAL
;1756     2559  4             ret_vbn;
;1757     2560  4         ret_vbn = .vbn;                             ! Block to deallocate
;1758     2561  4         perform ( find_block (.vbn, blkadr, cache_entry)); ! Cache it
;1759     2562  4         cache_entry [cache$v_data] = true;
;1760     2563  4         cache_entry [cache$v_dirty] = true;
;1761     2564  4         vbn = .blkadr [luh$l_nxtluhblk];            ! Follow link to next block
;1762     2565  4         perform ( dealloc_block ( .ret_vbn ));  ! return it to free list
;1763     2566  4         END
;1764     2567  3     UNTIL .vbn EQL 0;                               ! End of list
;1765     2568  3     !
;1766     2569  3     !    Zero all header pointers and offsets to mark history empty
;1767     2570  3     !
;1768     2571  3     begluhrfa = 0;
;1769     2572  3     begoffset = 0;
;1770     2573  3     endluhrfa = 0;
;1771     2574  3     endoffset = 0;
;1772     2575  3     END
;1773     2576  2 ELSE            ! There was more than one record in history, so remove the
;1774     2577  3     BEGIN       ! oldest, or first in the list
;1775     2578  3     LOCAL
;1776     2579  3         cache_entry : REF BBLOCK,       ! location in cache of luhblk
```

```
;  1777     2580  3          blkadr : REF BBLOCK,                ! address of VBN in cache
;  1778     2581  3          reclenlft,                          ! length of the LUH record
;  1779     2582  3          rec : REF BBLOCK,                   ! address of record within luhblk
;  1780     2583  3          offset,
;  1781     2584  3          vbn;
;  1782     2585  3
;  1783     2586  3      vbn = .begvbn;
;  1784     2587  3      offset = .begoffset;
;  1785     2588  3      perform ( find_block (.begvbn, blkadr, cache_entry) );      ! ensure the block is in cache.
;  1786     2589  3      cache_entry [cache$v_data] = true;
;  1787     2590  3      cache_entry [cache$v_dirty] = true;
;  1788     2591  3      rec = .blkadr + luh$c_data + .offset;       ! compute address of record start
;  1789     2592  3      !
;  1790     2593  3      !    Check mark word in header
;  1791     2594  3      !
;  1792     2595  3      IF .rec [luh$w_rechdr] NEQ luh$c_rechdrmrk THEN RETURN lbr$_intrnlerr;
;  1793     2596  3      !
;  1794     2597  3      !    To delete the record, the offset and beginning vbn pointer are reset to point to
;  1795     2598  3      !    the second record.  This is done a block at a time.  If any blocks are freed in
;  1796     2599  3      !    the process, they are returned to the free-list.
;  1797     2600  3      !
;  1798     2601  3      reclenlft = .rec [luh$w_reclen] + luh$c_rechdrlen;  ! Length of record not yet skipped over
;  1799     2602  3      WHILE .reclenlft GTR 0 DO                           ! While there is still part of the record left
;  1800     2603  4          BEGIN
;  1801     2604  5          IF ( .offset + .reclenlft ) LEQ ( luh$c_datfldlen - luh$c_rechdrlen)
;  1802     2605  4          THEN                ! the record is entirely contained in this block
;  1803     2606  5              BEGIN           ! Set offset to end of record and don't return the block cause next record is in it
;  1804     2607  5              offset = .offset + .reclenlft;
;  1805     2608  5              reclenlft = 0;      ! skipped past entire record
;  1806     2609  5              END
;  1807     2610  4          ELSE
;  1808     2611  5              BEGIN       ! The record fills or overflows this block so deallocate block
;  1809     2612  5              Local
;  1810     2613  5                  ret_vbn;
;  1811     2614  5              reclenlft = .reclenlft - (luh$c_datfldlen - .offset);
;  1812     2615  5              offset = 0;
;  1813     2616  5              ret_vbn = .vbn;
;  1814     2617  5              vbn = .blkadr [luh$l_nxtluhblk];
;  1815     2618  5              perform ( dealloc_block ( .ret_vbn ));
;  1816     2619  5              perform ( find_block ( .vbn, blkadr, cache_entry) );
;  1817     2620  5              cache_entry [cache$v_data] = true;
;  1818     2621  5              cache_entry [cache$v_dirty] = true;
;  1819     2622  4              END;
;  1820     2623  3          END;
;  1821     2624  3      begvbn = .vbn;                  ! Second record is now first
;  1822     2625  3      begoffset = .offset;
;  1823     2626  2      END;
;  1824     2627  2  header [lhd$w_numluhrec] = .header [lhd$w_numluhrec] - 1;
;  1825     2628  2  context [ctx$v_hdrdirty] = true;        ! Make sure header is written out
;  1826     2629  2  RETURN true;
;  1827     2630  1  END;    ! routine delete_luhrecord
```

```
                                                  .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11          ; 2524
             59      0000G  CF  9E 00002          MOVAB   FIND_BLOCK, R9                                ;
             5E         10  C2     00007          SUBL2   #16, SP                                       ;
             50      0000G  CF  D0 0000A          MOVL    LBR$GL_CONTROL, R0                            ; 2535
             58         0E  A0  D0 0000F          MOVL    14(R0), R8                                    ;
             54         0A  A0  D0 00013          MOVL    10(R0), R4                                    ; 2536
             56       0080  C4  9E 00017          MOVAB   128(R4), R6                                   ; 2537
             01         7E  A4  B1 0001C          CMPW    126(R4), #1                                   ; 2544
                            3D  12 00020          BNEQ    2$                                            ;
             53             66  D0 00022          MOVL    (R6), VBN                                     ; 2555
             55             53  D0 00025 1$:      MOVL    VBN, RET_VBN                                  ; 2560
             52             6E  9E 00028          MOVAB   CACHE_ENTRY, R2                               ; 2561
             51         04  AE  9E 0002B          MOVAB   BLKADR, R1                                    ;
             50             53  D0 0002F          MOVL    VBN, R0                                       ;
                            69  16 00032          JSB     FIND_BLOCK                                    ;
             3C             50  E9 00034          BLBC    STATUS, 3$                                    ;
             50             6E  D0 00037          MOVL    CACHE_ENTRY, R0                               ; 2562
          0C A0             03  88 0003A          BISB2   #3, 12(R0)                                    ; 2563
             53         04  BE  D0 0003E          MOVL    @BLKADR, VBN                                  ; 2564
             50             55  D0 00042          MOVL    RET_VBN, R0                                   ; 2565
                         0000G  30 00045          BSBW    DEALLOC_BLOCK                                 ;
             7B             50  E9 00048          BLBC    STATUS, 7$                                    ;
                            53  D5 0004B          TSTL    VBN                                           ; 2567
                            D6  12 0004D          BNEQ    1$                                            ;
                            66  D4 0004F          CLRL    (R6)                                          ; 2571
                        04  A6  B4 00051          CLRW    4(R6)                                         ; 2572
                      0086  C4  D4 00054          CLRL    134(R4)                                       ; 2573
                      008A  C4  B4 00058          CLRW    138(R4)                                       ; 2574
                          008B  31 0005C          BRW     9$                                            ; 2544
             57             66  D0 0005F 2$:      MOVL    (R6), VBN                                     ; 2586
             53         04  A6  3C 00062          MOVZWL  4(R6), OFFSET                                 ; 2587
             52         08  AE  9E 00066          MOVAB   CACHE_ENTRY, R2                               ; 2588
             51         0C  AE  9E 0006A          MOVAB   BLKADR, R1                                    ;
             50             66  D0 0006E          MOVL    (R6), R0                                      ;
                            69  16 00071          JSB     FIND_BLOCK                                    ;
             7E             50  E9 00073 3$:      BLBC    STATUS, 10$                                   ;
             50         08  AE  D0 00076          MOVL    CACHE_ENTRY, R0                               ; 2589
          0C A0             03  88 0007A          BISB2   #3, 12(R0)                                    ; 2590
   50        53         0C  AE  C1 0007E          ADDL3   BLKADR, OFFSET, R0                            ; 2591
             50             06  C0 00083          ADDL2   #6, REC                                       ;
        ABBA 8F             60  B1 00086          CMPW    (REC), #43962                                 ; 2595
                            08  13 0008B          BEQL    4$                                            ;
             50  00000000G  8F  D0 0008D          MOVL    #LBR$_INTRNLERR, R0                           ;
                                04 00094          RET                                                   ;
             55         02  A0  3C 00095 4$:      MOVZWL  2(REC), RECLENLFT                             ; 2601
             55             04  C0 00099          ADDL2   #4, RECLENLFT                                 ;
                            55  D5 0009C 5$:      TSTL    RECLENLFT                                     ; 2602
                            43  15 0009E          BLEQ    8$                                            ;
   50        53             55  C1 000A0          ADDL3   RECLENLFT, OFFSET, R0                         ; 2604
    000001F6 8F             50  D1 000A4          CMPL    R0, #502                                      ;
                            07  14 000AB          BGTR    6$                                            ;
             53             55  C0 000AD          ADDL2   RECLENLFT, OFFSET                             ; 2607
                            55  D4 000B0          CLRL    RECLENLFT                                     ; 2608
                            E8  11 000B2          BRB     5$                                            ; 2604
             55      FE06 C345  9E 000B4 6$:      MOVAB   -506(OFFSET)[RECLENLFT], RECLENLFT            ; 2614
                            53  D4 000BA          CLRL    OFFSET                                        ; 2615
             50             57  D0 000BC          MOVL    VBN, RET_VBN                                  ; 2616
```

```
                                57      0C   BE   D0 000BF             MOVL    @BLKADR, VBN                            ; 2617
                                      0000G  30   000C3                 BSBW    DEALLOC_BLOCK                           ; 2618
                                2B          50   E9 000C6 7$:          BLBC    STATUS, 10$                             ;
                                52      08   AE   9E 000C9             MOVAB   CACHE_ENTRY, R2                         ; 2619
                                51      0C   AE   9E 000CD             MOVAB   BLKADR, R1                              ;
                                50          57   D0 000D1             MOVL    VBN, R0                                 ;
                                            69   16 000D4             JSB     FIND_BLOCK                              ;
                                1B          50   E9 000D6             BLBC    STATUS, 10$                             ;
                                50      08   AE   D0 000D9             MOVL    CACHE_ENTRY, R0                         ; 2620
                            0C  A0          03   88 000DD             BISB2   #3, 12(R0)                              ; 2621
                                            B9   11 000E1             BRB     5$                                      ; 2602
                                66          57   D0 000E3 8$:          MOVL    VBN, (R6)                               ; 2624
                            04  A6          53   B0 000E6             MOVW    OFFSET, 4(R6)                           ; 2625
                                7E      A4   B7   00EA 9$:             DECW    126(R4)                                 ; 2627
                            04  A8          08   88 000ED             BISB2   #8, 4(R8)                               ; 2628
                                50          01   D0 000F1             MOVL    #1, R0                                  ; 2629
                                            04   000F4 10$:           RET                                             ; 2630

; Routine Size:  245 bytes,    Routine Base:  $CODE$ + 1041

; 1828          2631  1
```

```
;  1830     2632  1 %SBTTL  'LBR$GET_HISTORY';
;  1831     2633  1 GLOBAL ROUTINE lbr$get_history (control_index, action_routine) =
;  1832     2634  2 BEGIN
;  1833     2635  2 !+++
;  1834     2636  2 !
;  1835     2637  2 !   FUNCTIONAL DESCRIPTION:
;  1836     2638  2 !
;  1837     2639  2 !       For each Library Update History record copy the record to a buffer
;  1838     2640  2 !       and call the action_routine with a descriptor for the buffer.
;  1839     2641  2 !
;  1840     2642  2 !
;  1841     2643  2 !   CALLING SEQUENCE:
;  1842     2644  2 !
;  1843     2645  2 !       status = lbr$get_history (control_index, action_routine)
;  1844     2646  2 !
;  1845     2647  2 !
;  1846     2648  2 !   INPUT PARAMETERS:
;  1847     2649  2 !
;  1848     2650  2 !       control_index   is the index returned from lbr$ini_control
;  1849     2651  2 !       action_routine  is a user supplied routine which is called for each
;  1850     2652  2 !                       LUH record, being passed a descriptor for the buffer
;  1851     2653  2 !                       containing a copy of the record.
;  1852     2654  2 !
;  1853     2655  2 !   ROUTINE VALUE:
;  1854     2656  2 !
;  1855     2657  2 !       lbr$_intrnlerr          Internal librarian error
;  1856     2658  2 !       lbr$_normal             Normal exit
;  1857     2659  2 !       lbr$_nohistory          This library does not have an update history
;  1858     2660  2 !       lbr$_emptyhist          The history is empty
;  1859     2661  2 !---
;  1860     2662  2 perform (validate_ctl (..control_index));       ! Validate the control index
;  1861     2663  3 BEGIN
;  1862     2664  3 BIND
;  1863     2665  3     header = .lbr$gl_control [lbr$l_hdrptr] : BBLOCK,   ! library header
;  1864     2666  3     luhblkrfa = header [lhd$b_begluhrfa] : BBLOCK,      ! rfa of the oldest LUH record
;  1865     2667  3     beg_offset = luhblkrfa [rfa$w_offset] : WORD,       ! offset to first record
;  1866     2668  3     beg_vbn = luhblkrfa [rfa$l_vbn];                    ! VBN of first record
;  1867     2669  3 LOCAL
;  1868     2670  3     blkadr : REF BBLOCK,          ! block address of cached block
;  1869     2671  3     cache_entry : REF BBLOCK,     ! cache entry locating luh block
;  1870     2672  3     numrecs : WORD,               ! number of history records in library history
;  1871     2673  3     offset,                       ! offset to current LUH record being copied
;  1872     2674  3     vbn,                          ! VBN of current LUH record being copied
;  1873     2675  3     status;
;  1874     2676  3
;  1875     2677  3 IF .header [lhd$w_maxluhrec] EQL 0             ! History not maintained for this library
;  1876     2678  3 THEN RETURN lbr$_nohistory;
;  1877     2679  3 IF .header [lhd$w_numluhrec] EQL 0             ! History is empty for this library
;  1878     2680  3 THEN RETURN lbr$_emptyhist;
;  1879     2681  3 !
;  1880     2682  3 !    For as many LUH records as are in the library history, locate next record,
;  1881     2683  3 !    copy it to buffer, and call action_routine with buffer descriptor.
;  1882     2684  3 !
;  1883     2685  3 vbn = .beg_vbn;                   ! vbn of first record
;  1884     2686  3 offset = .beg_offset;             ! Offset within block to first record
;  1885     2687  3 status = find_block (.vbn, blkadr, cache_entry);        ! cache the block
;  1886     2688  3 cache_entry [cache$v_data] = true;
```

```
;  1887          2689  3 numrecs = .header [lhd$w_numluhrec];                    ! Number of LUH records
;  1888          2690  3 INCR i FROM 1 TO .numrecs BY 1 DO       ! for each record in history
;  1889          2691  4     BEGIN
;  1890          2692  4     LOCAL
;  1891          2693  4         cpyrecadr,
;  1892          2694  4         dstadr,
;  1893          2695  4         luhrec : REF BBLOCK,
;  1894          2696  4         pass_desc : BBLOCK [dsc$c_s_bln],       ! Descriptor to pass to user routine
;  1895          2697  4         save_desc : BBLOCK [dsc$c_s_bln],       ! Descriptor to use to dealloc buffer (In case user diddles
;  1896          2698  4         reclen,
;  1897          2699  4         reclenlft;
;  1898          2700  4
;  1899          2701  4     luhrec = .blkadr + luh$c_data + .offset;    ! beginning address of first record
;  1900          2702  4     IF .luhrec [luh$w_rechdr] NEQ luh$c_rechdrmrk       ! history is corrupted if mark header not here
;  1901          2703  4     THEN RETURN lbr$_intrnlerr;
;  1902          2704  4     reclen = .luhrec [luh$w_reclen];
;  1903          2705  4     reclenlft = .reclen;
;  1904          2706  4     save_desc [dsc$w_length] = .reclen;
;  1905          2707  4     perform ( get_zmem (.reclen, save_desc [dsc$a_pointer]) ); ! get buffer to put record in
;  1906          2708  4     pass_desc = .save_desc;                                 ! Pass_desc is a copy of save_desc
;  1907          2709  4     pass_desc [dsc$a_pointer] = .save_desc [dsc$a_pointer];
;  1908          2710  4     !
;  1909          2711  4     !   now get record into buffer
;  1910          2712  4     !   Since record can span several blocks, copy as much of record as is in current block.
;  1911          2713  4     !   then follow link to next block.  Continue until entire record copied into buffer.
;  1912          2714  4     !   Then call user routine with a descriptor of the copy of the record.
;  1913          2715  4     !
;  1914          2716  4     cpyrecadr = .luhrec + luh$c_rechdrlen;
;  1915          2717  4     offset = .offset + luh$c_rechdrlen;
;  1916          2718  4     dstadr = .save_desc [dsc$a_pointer];
;  1917          2719  4     WHILE .reclenlft GTR 0 DO   ! While there is more left, keep copying it over
;  1918          2720  5         BEGIN
;  1919          2721  5         LOCAL
;  1920          2722  5             cpylen;
;  1921          2723  5         cpylen = MIN( .reclenlft, luh$c_datfldlen - .offset);
;  1922          2724  5         CH$MOVE (.cpylen, .cpyrecadr, .dstadr);
;  1923          2725  5         reclenlft = .reclenlft - .cpylen;
;  1924          2726  5         offset = .offset + .cpylen;
;  1925          2727  5         dstadr = .dstadr + .cpylen;
;  1926          2728  6         IF (.offset GTR (luh$c_datfldlen - luh$c_rechdrlen))                                              ;
;  1927          2729  5         THEN
;  1928          2730  6             BEGIN
;  1929          2731  6             vbn = .blkadr [luh$l_nxtluhblk];
;  1930          2732  6             offset = 0;
;  1931          2733  6             status = find_block (.vbn, blkadr, cache_entry);
;  1932          2734  6             cache_entry [cache$v_data] = true;
;  1933          2735  6             cpyrecadr = .blkadr + luh$c_data
;  1934          2736  5             END;
;  1935          2737  4         END;                                        ! while copying over record to buffer
;  1936          2738  4     perform ( (.action_routine) (pass_desc) );  ! Call user routine
;  1937          2739  4     perform (validate_ctl (..control_index));   ! Validate the control index
;  1938          2740  4     perform ( dealloc_mem ( .save_desc [dsc$w_length], .save_desc [dsc$a_pointer] ));   ! Return the buffer
;  1939          2741  3     END;                    ! INCRement thru the history list
;  1940          2742  3 RETURN lbr$_normal;
;  1941          2743  2 END;
;  1942          2744  1 END;    ! lbr$get_history
```

```
                                 0FFC 00000          .ENTRY  LBR$GET_HISTORY, Save R2,R3,R4,R5,R6,R7,R8,-;  2633
                                                             R9,R10,R11
                              5E   24  C2 00002      SUBL2   #36, SP
                              50 04   BC  D0 00005      MOVL    @CONTROL_INDEX, R0                           ;  2662
                                 0000G 30 00009      BSBW    VALIDATE_CTL
                              7F   50  E9 0000C      BLBC    STATUS, 5$
                              50 0000G CF  D0 0000F      MOVL    LBR$GL_CONTROL, R0                           ;  2665
                              53   0A  A0  D0 00014      MOVL    10(R0), R3                                   ;  2677
                                   7C  A3  B5 00018      TSTW    124(R3)
                                   08  12  12 0001B      BNEQ    1$
                              50 00000000G 8F  D0 0001D      MOVL    #LBR$_NOHISTORY, R0                      ;  2678
                                       04  00024      RET
                                   7E  A3  B5 00025 1$:  TSTW    126(R3)                                      ;  2679
                                       08  12 00028      BNEQ    2$
                              50 00000000G 8F  D0 0002A      MOVL    #LBR$_EMPTYHIST, R0                      ;  2680
                                       04  00031      RET
                           04 AE 0080  C3  D0 00032 2$:  MOVL    128(R3), VBN                                 ;  2685
                              58 0084  C3  3C 00038      MOVZWL  132(R3), OFFSET                              ;  2686
                              52   0C  AE  9E 0003D      MOVAB   CACHE_ENTRY, R2                              ;  2687
                              51   10  AE  9E 00041      MOVAB   BLKADR, R1
                              50   04  AE  D0 00045      MOVL    VBN, R0
                                 0000G 30 00049      BSBW    FIND_BLOCK
                           08 AE       50  D0 0004C      MOVL    R0, STATUS
                              50   0C  AE  D0 00050      MOVL    CACHE_ENTRY, R0                              ;  2688
                           0C A0       02  88 00054      BISB2   #2, 12(R0)
                              50   7E  A3  B0 00058      MOVW    126(R3), NUMRECS                             ;  2689
                              6E       50  3C 0005C      MOVZWL  NUMRECS, (SP)                                ;  2690
                                       5B  D4 0005F      CLRL    I                                            ;  2738
                                 00B8  31 00061      BRW     10$
                        56           58  10  AE  C1 00064 3$:  ADDL3   BLKADR, OFFSET, R6                    ;  2701
                              52   06  A6  9E 00069      MOVAB   6(R6), LUHREC
                        ABBA  8F       62  B1 0006D      CMPW    (LUHREC), #43962                             ;  2702
                                       08  13 00072      BEQL    4$
                              50 00000000G 8F  D0 00074      MOVL    #LBR$_INTRNLERR, R0                      ;  2703
                                       04  00078      RET
                              50   02  A2  3C 0007C 4$:  MOVZWL  2(LUHREC), RECLEN                           ;  2704
                              57       50  D0 00080      MOVL    RECLEN, RECLENLFT                            ;  2705
                           14 AE       50  B0 00083      MOVW    RECLEN, SAVE_DESC                            ;  2706
                              51   18  AE  9E 00087      MOVAB   SAVE_DESC+4, R1                              ;  2707
                                 0000G 30 0008B      BSBW    GET_ZMEM
                              7A       50  E9 0008E 5$:  BLBC    STATUS, 9$
                           1C AE   14  AE  7D 00091      MOVQ    SAVE_DESC, PASS_DESC                         ;  2708
                              56   04  A2  9E 00096      MOVAB   4(R2), CPYRECADR                             ;  2716
                              58       04  C0 0009A      ADDL2   #4, OFFSET                                   ;  2717
                              5A   18  AE  D0 0009D      MOVL    SAVE_DESC+4, DSTADR                          ;  2718
                                       57  D5 000A1 6$:  TSTL    RECLENLFT                                    ;  2719
                                       55  15 000A3      BLEQ    8$
                        51 000001FA 8F       58  C3 000A5      SUBL3   OFFSET, #506, R1                     ;  2723
                              50       57  D0 000AD      MOVL    RECLENLFT, R0
                              51       50  D1 000B0      CMPL    R0, R1
                                       03  15 000B3      BLEQ    7$
                              50       51  D0 000B5      MOVL    R1, R0
                              59       50  D0 000B8 7$:  MOVL    R0, CPYLEN
```

```
                6A              66              59  28 000BB          MOVC3   CPYLEN, (CPYRECADR), (DSTADR)          ; 2724
                                57              59  C2 000BF          SUBL2   CPYLEN, RECLENLFT                      ; 2725
                                58              59  C0 000C2          ADDL2   CPYLEN, OFFSET                         ; 2726
                                5A              59  C0 000C5          ADDL2   CPYLEN, DSTADR                         ; 2727
          000001F6              8F              58  D1 000C8          CMPL    OFFSET, #502                           ; 2728
                                                  D0 15 000CF          BLEQ    6$                                     ;
                        04      AE      10      BE  D0 000D1          MOVL    @BLKADR, VBN                           ; 2731
                                                58  D4 000D6          CLRL    OFFSET                                 ; 2732
                                52      0C      AE  9E 000D8          MOVAB   CACHE_ENTRY, R2                        ; 2733
                                51      10      AE  9E 000DC          MOVAB   BLKADR, R1                             ;
                                50      04      AE  D0 000E0          MOVL    VBN, R0                                ;
                                            0000G   30 000E4          BSBW    FIND_BLOCK                             ;
                        08      AE              50  D0 000E7          MOVL    R0, STATUS                             ;
                                50      0C      AE  D0 000EB          MOVL    CACHE_ENTRY, R0                        ; 2734
                        0C      A0              02  88 000EF          BISB2   #2, 12(R0)                             ;
                56      10      AE              06  C1 000F3          ADDL3   #6, BLKADR, CPYRECADR                  ; 2735
                                                A7  11 000F8          BRB     6$                                     ; 2719
                                1C      AE      9F  000FA 8$:         PUSHAB  PASS_DESC                              ; 2738
                        08      BC              01  FB 000FD          CALLS   #1, @ACTION_ROUTINE                    ;
                                25              50  E9 00101         BLBC    STATUS, 11$                            ;
                                50      04      BC  D0 00104         MOVL    @CONTROL_INDEX, R0                     ; 2739
                                            0000G   30 00108         BSBW    VALIDATE_CTL                           ;
                                1B              50  E9 0010B 9$:     BLBC    STATUS, 11$                            ;
                                51      18      AE  D0 0010E         MOVL    SAVE_DESC+4, R1                        ; 2740
                                50      14      AE  3C 00112         MOVZWL  SAVE_DESC, R0                          ;
                                            0000G   30 00116         BSBW    DEALLOC_MEM                            ;
                                0D              50  E9 00119         BLBC    STATUS, 11$                            ;
        FF42                    5B              01  6E F1 0011C 10$:  ACBL    (SP), #1, I, 3$                        ; 2690
                                50 00000000G    8F  D0 00122         MOVL    #LBR$_NORMAL, R0                       ; 2742
                                                    04 00129 11$:     RET                                            ; 2744

; Routine Size:  298 bytes,    Routine Base:  $CODE$ + 1136

; 1943          2745  1
; 1944          2746  1 END
; 1945          2747  0 ELUDOM                                  ! Of module


;                                PSECT SUMMARY
;
;      Name                     Bytes                          Attributes
;
;  $CODE$                         4704  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        44          0        581          00:01.0



;                                       COMMAND QUALIFIERS

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

; Size:          4632 code + 72 data bytes
; Run Time:         01:27.0
; Elapsed Time:     02:45.7
; Lines/CPU Min:     1893
; Lexemes/CPU-Min: 23242
; Memory Used:  256 pages
; Compilation Complete
```

GETHELP
LIS

INDEX
LIS

GETPUT
LIS

GETMEM
LIS