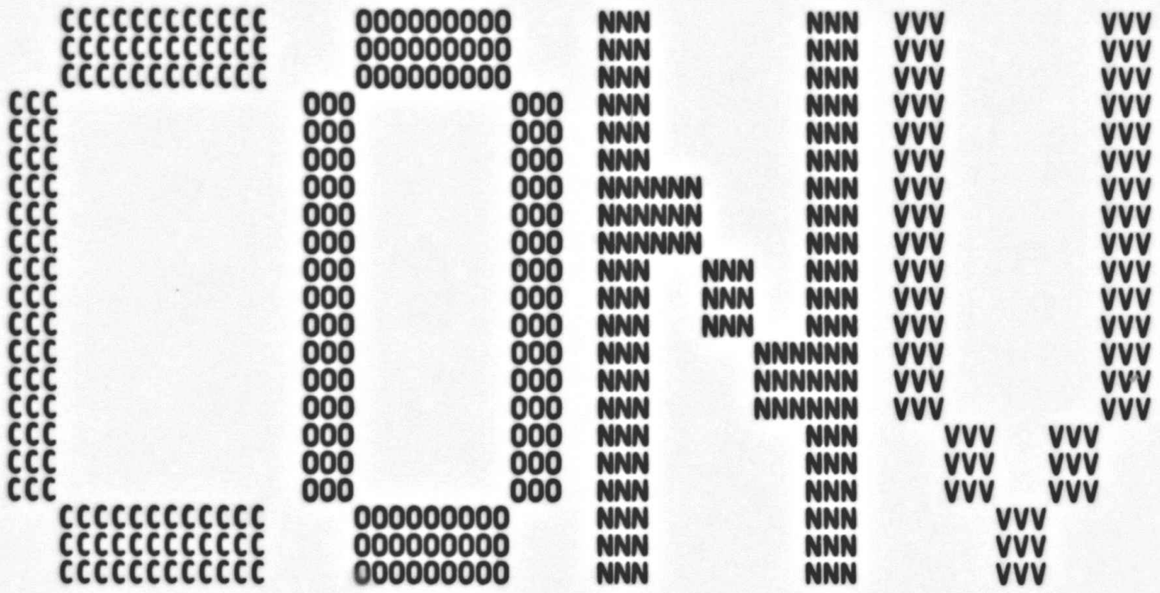
CCCCCCCCCCCC OOOOOOOOO NNN NNN VVV VVV
CCCCCCCCCCCC OOOOOOOOO NNN NNN VVV VVV
CCCCCCCCCCCC OOOOOOOOO NNN NNN VVV VVV
CCC OOO OOO NNN NNN VVV VVV
CCC OOO OOO NNN NNN VVV VVV
CCC OOO OOO NNN NNN VVV VVV
CCC OOO OOO NNNNNN NNN VVV VVV
CCC OOO OOO NNNNNN NNN VVV VVV
CCC OOO OOO NNNNNN NNN VVV VVV
CCC OOO OOO NNN NNN NNN VVV VVV
CCC OOO OOO NNN NNN NNN VVV VVV
CCC OOO OOO NNN NNN NNN VVV VVV
CCC OOO OOO NNN NNNNNN VVV VVV
CCC OOO OOO NNN NNNNNN VVV VVV
CCC OOO OOO NNN NNNNNN VVV VVV
CCC OOO OOO NNN NNN VVV VVV
CCC OOO OOO NNN NNN VVV VVV
CCC OOO OOO NNN NNN VVV VVV
CCCCCCCCCCCC OOOOOOOOO NNN NNN VVV
CCCCCCCCCCCC OOOOOOOOO NNN NNN VVV
CCCCCCCCCCCC OOOOOOOOO NNN NNN VVV

```
RRRRRRRR    EEEEEEEEEE    CCCCCCCC  LL          RRRRRRRR    MM      MM    SSSSSSSS    IIIIII      OOOOOO
RRRRRRRR    EEEEEEEEEE    CCCCCCCC  LL          RRRRRRRR    MM      MM    SSSSSSSS    IIIIII      OOOOOO
RR      RR  EE            CC        LL          RR      RR  MMMM  MMMM  SS              II      OO      OO
RR      RR  EE            CC        LL          RR      RR  MMMM  MMMM  SS              II      OO      OO
RR      RR  EE            CC        LL          RR      RR  MM  MM  MM  SS              II      OO      OO
RR      RR  EE            CC        LL          RR      RR  MM  MM  MM  SS              II      OO      OO
RRRRRRRR    EEEEEEEE      CC        LL          RRRRRRRR    MM      MM    SSSSSS        II      OO      OO
RRRRRRRR    EEEEEEEE      CC        LL          RRRRRRRR    MM      MM    SSSSSS        II      OO      OO
RR  RR      EE            CC        LL          RR  RR      MM      MM          SS    II      OO      OO
RR  RR      EE            CC        LL          RR  RR      MM      MM          SS    II      OO      OO
RR    RR    EE            CC        LL          RR    RR    MM      MM          SS    II      OO      OO
RR    RR    EE            CC        LL          RR    RR    MM      MM          SS    II      OO      OO    ....
RR      RR  EEEEEEEEEE    CCCCCCCC  LLLLLLLLLL  RR      RR  MM      MM    SSSSSSSS    IIIIII      OOOOOO      ....
RR      RR  EEEEEEEEEE    CCCCCCCC  LLLLLLLLLL  RR      RR  MM      MM    SSSSSSSS    IIIIII      OOOOOO      ....

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 %TITLE  'VAX-11 CONVERT/RECLAIM'
;  2          0002  0 MODULE  RECL$RMSIO      ( IDENT='V04-000',
;  3          0003  0                         OPTLEVEL=3
;  4          0004  0                         ) =
;  5          0005  0
;  6          0006  1 BEGIN
;  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 !**********************************************************************
```

```
;    31          0030  1 !++
;    32          0031  1 !
;    33          0032  1 ! Facility:     VAX-11 CONVERT/RECLAIM
;    34          0033  1 !
;    35          0034  1 ! Environment:
;    36          0035  1 !
;    37          0036  1 !               VAX/VMS Operating System
;    38          0037  1 !
;    39          0038  1 !
;    40          0039  1 ! Abstract:     CONVERT/RECLAIM facility RMS I/O routines
;    41          0040  1 !
;    42          0041  1 ! Contents:
;    43          0042  1 !               ALLOCATE_BUFFERS
;    44          0043  1 !               GET_LAST_VBN
;    45          0044  1 !               OPEN_FILE
;    46          0045  1 !               GET_NEXT_BUCKET
;    47          0046  1 !
;    48          0047  1 !--
;    49          0048  1
;    50          0049  1 !
;    51          0050  1 ! Author:       Keith B Thompson
;    52          0051  1 !               Peter Lieberwirth       Creation date:  August-1981
;    53          0052  1 !
;    54          0053  1 !
;    55          0054  1 ! Modified by:
;    56          0055  1 !
;    57          0056  1 !       V03-007 KBT0554         Keith B. Thompson       20-Jul-1983
;    58          0057  1 !               Init the outsum xab
;    59          0058  1 !
;    60          0059  1 !       V03-006 KBT0388         Keith B. Thompson       27-Oct-1982
;    61          0060  1 !               Add support for prologue 3 sidrs
;    62          0061  1 !
;    63          0062  1 !       V03-005 KBT0359         Keith B. Thompson       6-Oct-1982
;    64          0063  1 !               Use new merged ctx definitions
;    65          0064  1 !
;    66          0065  1 !       V03-004 KBT0352         Keith B. Thompson       5-Oct-1982
;    67          0066  1 !               Use new linkage definitions
;    68          0067  1 !
;    69          0068  1 !       V03-003 KBT0051         Keith Thompson          10-May-1982
;    70          0069  1 !               Remove the check to see if the vbn has been read already and
;    71          0070  1 !               fix the bucket address sample compare
;    72          0071  1 !
;    73          0072  1 !       V03-002 KBT0040         Keith Thompson          3-Apr-1982
;    74          0073  1 !               Add recl$gl_search_buffer and allocate it.  Also make
;    75          0074  1 !               sure you find the last data level bucket.
;    76          0075  1 !
;    77          0076  1 !       V03-001 KBT0009         Keith Thompson          16-Mar-1982
;    78          0077  1 !               Store LEVEL of bucket in RTX, correct first VBN at data
;    79          0078  1 !               level and make sure the last bucket at the index levels
;    80          0079  1 !               is found. Also correct bug in error processing and fix
;    81          0080  1 !               the way we wirte area descriptors.
;    82          0081  1 !
;    83          0082  1 !****
```

```
;    85          0083  1
;    86          0084  1 PSECT
;    87          0085  1         OWN     = _CONV$RECL_D  (PIC),
;    88          0086  1         GLOBAL  = _CONV$RECL_D  (PIC),
;    89          0087  1         PLIT    = _CONV$PLIT    (SHARE,PIC),
;    90          0088  1         CODE    = _CONV$RECL_S  (SHARE,PIC);
;    91          0089  1
;    92          0090  1 LIBRARY 'SYS$LIBRARY:LIB.L32';
;    93          0091  1 LIBRARY 'SRC$:CONVERT';
;    94          0092  1
;    95          0093  1 DEFINE_ERROR_CODES;
;    96          0094  1
;    97          0095  1 EXTERNAL ROUTINE
;    98          0096  1         RECL$$SWAP_BUFFERS      : RL$JSB_REG_9  NOVALUE,
;    99          0097  1         CONV$$READ_PROLOGUE     : CL$READ_PROLOGUE      NOVALUE,
;   100          0098  1         CONV$$RMS_ERROR         : NOVALUE,
;   101          0099  1         CONV$$RMS_OPEN_ERROR    : NOVALUE,
;   102          0100  1         CONV$$RMS_READ_ERROR    : NOVALUE,
;   103          0101  1         CONV$$GET_VM            : CL$GET_VM,
;   104          0102  1         CONV$$GET_TEMP_VM       : CL$GET_TEMP_VM;
;   105          0103  1
;   106          0104  1 FORWARD ROUTINE
;   107          0105  1         RECL$$GET_NEXT_BUCKET   : RL$JSB_REG_9  NOVALUE,
;   108          0106  1         GET_LAST_VBN            : RL$JSB_REG_9;
;   109          0107  1
;   110          0108  1 EXTERNAL
;   111          0109  1         CONV$AB_FLAGS           : BLOCK [ ,BYTE ],
;   112          0110  1         CONV$AB_OUT_FAB         : $FAB_DECL,
;   113          0111  1         CONV$AB_OUT_NAM         : $NAM_DECL,
;   114          0112  1         CONV$AB_OUT_RAB         : $RAB_DECL,
;   115          0113  1         CONV$AB_OUT_XABSUM      : $XABSUM_DECL,
;   116          0114  1
;   117          0115  1         RECL$GL_BUCKET_COUNT    : LONG,
;   118          0116  1         RECL$GL_DATA_COUNT      : LONG,
;   119          0117  1         RECL$GL_INDEX_COUNT     : LONG;
;   120          0118  1
;   121          0119  1 GLOBAL
;   122          0120  1         RECL$GL_SEARCH_BUFFER   : LONG,
;   123          0121  1         RECL$GL_WRITE_AREA      : LONG INITIAL ( 0 ),
;   124          0122  1         RECL$GL_WRITE_KEY       : LONG INITIAL ( 0 );
;   125          0123  1
```

```
;  127          0124  1 %SBTTL  'ALLOCATE_BUFFERS'
;  128          0125  1 GLOBAL ROUTINE  RECL$$ALLOCATE_BUFFERS : RL$JSB_REG_9 =
;  129          0126  1 !++
;  130          0127  1 !
;  131          0128  1 ! Functional Description:
;  132          0129  1 !
;  133          0130  1 !       Allocates context buffers, two bucket buffers for each level
;  134          0131  1 !       and reads in first virtical row of index and data buckets.
;  135          0132  1 !
;  136          0133  1 ! Calling Sequence:
;  137          0134  1 !
;  138          0135  1 !       RECL$$ALLOCATE_BUFFERS()
;  139          0136  1 !
;  140          0137  1 ! Input Parameters:
;  141          0138  1 !       none
;  142          0139  1 !
;  143          0140  1 ! Implicit Inputs:
;  144          0141  1 !       none
;  145          0142  1 !
;  146          0143  1 ! Output Parameters:
;  147          0144  1 !       none
;  148          0145  1 !
;  149          0146  1 ! Implicit Outputs:
;  150          0147  1 !       none
;  151          0148  1 !
;  152          0149  1 ! Routine Value:
;  153          0150  1 !
;  154          0151  1 !       SS$_NORMAL
;  155          0152  1 !
;  156          0153  1 ! Routines Called:
;  157          0154  1 !
;  158          0155  1 !       CONV$$GET_VM
;  159          0156  1 !       RECL$$GET_NEXT_BUCKET
;  160          0157  1 !       GET_LAST_VBN
;  161          0158  1 !       REC[$$SWAP_BUFFERS
;  162          0159  1 !
;  163          0160  1 ! Side Effects:
;  164          0161  1 !       none
;  165          0162  1 !
;  166          0163  1 !--
;  167          0164  1
;  168          0165  2     BEGIN
;  169          0166  2
;  170          0167  2     DEFINE_CTX;
;  171          0168  2     DEFINE_BUCKET;
;  172          0169  2     DEFINE_KEY_DESC;
;  173          0170  2
;  174          0171  2     LOCAL
;  175          0172  2         BYTES,
;  176          0173  2         CTX_SIZE,                      ! Size in bytes of the context buffer
;  177          0174  2         DAT_BUF_SIZE,                  ! Size in bytes of level 0 bucket buffer
;  178          0175  2         IDX_BUF_SIZE;                  ! Size in bytes of level >0 bucket buffer
;  179          0176  2
;  180          0177  2     ! Get the number of bytes for the context block, one for each level
;  181          0178  2     !
;  182          0179  2     CTX_SIZE = ( .KEY_DESC [ KEY$B_ROOTLEV ] + 1 ) * CTX$K_BLN;
;  183          0180  2
```

```
;  184          0181  2     ! Get the number of bytes for the data level buffers
;  185          0182  2     !
;  186          0183  2     DAT_BUF_SIZE = .KEY_DESC [ KEY$B_DATBKTSZ ] * BLOCK_SIZE;
;  187          0184  2
;  188          0185  2     ! Get the number of bytes for the index level bucket buffers
;  189          0186  2     !
;  190          0187  2     IDX_BUF_SIZE = .KEY_DESC [ KEY$B_IDXBKTSZ ] * BLOCK_SIZE;
;  191          0188  2
;  192          0189  2     ! Add em up NOTE: Two bucket buffers for each level and a search buffer
;  193          0190  2     !                 which is index bucket size
;  194          0191  2     !
;  195          0192  2     BYTES = .CTX_SIZE + ( 2 * ( .DAT_BUF_SIZE +
;  196          0193  2                         ( .IDX_BUF_SIZE * .KEY_DESC [ KEY$B_ROOTLEV ] ) ) ) +
;  197          0194  2                         .IDX_BUF_SIZE;
;  198          0195  2
;  199          0196  2     ! Allocate the virtual memory and point the context block to it
;  200          0197  2     !
;  201          0198  2     CTX = CONV$$GET_TEMP_VM( .BYTES );
;  202          0199  2
;  203          0200  2     ! Set the size, area number and first VBN of the data level bucket
;  204          0201  2     !
;  205          0202  2     CTX [ CTX$W_BUCKET_SIZE ]   = .DAT_BUF_SIZE;
;  206          0203  2     CTX [ CTX$B_AREA ]          = .KEY_DESC [ KEY$B_DANUM ];
;  207          0204  2     CTX [ CTX$L_FIRST_VBN ]     = .KEY_DESC [ KEY$L_LDVBN ];
;  208          0205  2
;  209          0206  2     ! Set up the pointers for the data level buffers
;  210          0207  2     !
;  211          0208  2     CTX [ CTX$L_CURRENT_BUFFER ] = .CTX + .CTX_SIZE;
;  212          0209  2     CTX [ CTX$L_PREVIOUS_BUFFER ] = .CTX [ CTX$L_CURRENT_BUFFER ] +
;  213          0210  2                                                  .DAT_BUF_SIZE;
;  214          0211  2
;  215          0212  2     ! Set up the pointers for the rest of the context blocks
;  216          0213  2     !
;  217          0214  2     BEGIN
;  218          0215  2
;  219          0216  2     LOCAL
;  220          0217  2         BUF_PTR;
;  221          0218  2
;  222          0219  2     ! Keep a pointer of where we are in the bucket buffers
;  223          0220  2     !
;  224          0221  2     BUF_PTR = .CTX [ CTX$L_PREVIOUS_BUFFER ] + .DAT_BUF_SIZE;
;  225          0222  2
;  226          0223  2     ! Loop once for each level
;  227          0224  2     !
;  228          0225  2     INCR I FROM 1 TO .KEY_DESC [ KEY$B_ROOTLEV ]
;  229          0226  2     DO
;  230          0227  4         BEGIN
;  231          0228  4
;  232          0229  4         ! Point to the next block
;  233          0230  4         !
;  234          0231  4         CTX = .CTX + CTX$K_BLN;
;  235          0232  4
;  236          0233  4         ! Fill in the buffer pointers etc.
;  237          0234  4         !
;  238          0235  4         CTX [ CTX$L_CURRENT_BUFFER ]    = .BUF_PTR;
;  239          0236  4         CTX [ CTX$L_PREVIOUS_BUFFER ]   = .BUF_PTR + .IDX_BUF_SIZE;
;  240          0237  4         CTX [ CTX$W_BUCKET_SIZE ]       = .IDX_BUF_SIZE;
```

```
;  241     0238  4          CTX [ CTX$B_LEVEL ]                = .I;
;  242     0239  4
;  243     0240  4          ! Set the area number of the block (Since there are two possible areas
;  244     0241  4          ! depending on the level in the index we must check )
;  245     0242  4          !
;  246     0243  4          IF .I EQLU 1
;  247     0244  4          THEN
;  248     0245  4              CTX [ CTX$B_AREA ] = .KEY_DESC [ KEY$B_LANUM ]
;  249     0246  4          ELSE
;  250     0247  4              CTX [ CTX$B_AREA ] = .KEY_DESC [ KEY$B_IANUM ];
;  251     0248  4
;  252     0249  5          BUF_PTR = .BUF_PTR + ( .IDX_BUF_SIZE * 2 )
;  253     0250  5
;  254     0251  3          END;
;  255     0252  3
;  256     0253  3      ! The last piece is the search buffer
;  257     0254  3      !
;  258     0255  3      RECL$GL_SEARCH_BUFFER = .BUF_PTR
;  259     0256  3
;  260     0257  2      END;
;  261     0258  2
;  262     0259  2      ! At the finish of the above loop the context block is pointing
;  263     0260  2      ! to the root block
;  264     0261  2      !
;  265     0262  3      BEGIN
;  266     0263  3
;  267     0264  3      LOCAL
;  268     0265  3          LAST_VBN;
;  269     0266  3
;  270     0267  3      ! To read in the first and last bucket in each level of the index we
;  271     0268  3      ! first read in the root then search downward, also set the
;  272     0269  3      ! first vbn pointer for the root
;  273     0270  3      !
;  274     0271  3      CTX [ CTX$L_NEXT_VBN ]      = .KEY_DESC [ KEY$L_ROOTVBN ];
;  275     0272  3      CTX [ CTX$L_FIRST_VBN ]     = .KEY_DESC [ KEY$L_ROOTVBN ];
;  276     0273  3      CTX [ CTX$L_PREVIOUS_VBN ]  = .KEY_DESC [ KEY$L_ROOTVBN ];
;  277     0274  3
;  278     0275  3      ! Read in root bucket
;  279     0276  3      !
;  280     0277  3      RECL$$GET_NEXT_BUCKET();
;  281     0278  3
;  282     0279  3      ! Get the last index record vbn
;  283     0280  3      !
;  284     0281  3      LAST_VBN = GET_LAST_VBN();
;  285     0282  3
;  286     0283  3      ! Loop once for each level    NOTE: This loop will also set the first_vbn
;  287     0284  3      !
;  288     0285  3      INCR I FROM 1 TO .KEY_DESC [ KEY$B_ROOTLEV ] BY 1
;  289     0286  3      DO
;  290     0287  4          BEGIN
;  291     0288  4
;  292     0289  4          ! Move down the levels
;  293     0290  4          !
;  294     0291  4          CTX = .CTX - CTX$K_BLN;
;  295     0292  4
;  296     0293  4          ! Use vbn found for the next get
;  297     0294  4          !
```

```
;  298          0295  4          CTX [ CTX$L_NEXT_VBN ]  = .LAST_VBN;
;  299          0296  4
;  300          0297  4          ! Read in the last bucket bucket
;  301          0298  4          !
;  302          0299  4          RECL$$GET_NEXT_BUCKET();
;  303          0300  4
;  304          0301  4          !+
;  305          0302  4          ! If this is an index bucket make sure this is the last one in the
;  306          0303  4          ! level, is not (rms had a update failure) get the last bucket.
;  307          0304  4          ! If it is a data bucket we want to go all the way to the end of the
;  308          0305  4          ! chain so we cah start at the beginning of the file
;  309          0306  4          !-
;  310          0307  4
;  311          0308  4          ! Check for last bucket in chain. NOTE: GET_NEXT_BUCKET sets BUCKET
;  312          0309  4          !                                       to the current buffer
;  313          0310  4          !
;  314          0311  4          WHILE NOT .BUCKET [ BKT$V_LASTBKT ]
;  315          0312  4          DO
;  316          0313  5              BEGIN
;  317          0314  5
;  318          0315  5              ! Swap the buffers
;  319          0316  5              !
;  320          0317  5              RECL$$SWAP_BUFFERS();
;  321          0318  5
;  322          0319  5              ! Get the next bucket in the chain
;  323          0320  5              !
;  324          0321  5              RECL$$GET_NEXT_BUCKET()
;  325          0322  5
;  326          0323  4              END;
;  327          0324  4
;  328          0325  4          ! If this is the index level get the last vbn pointer in the bucket
;  329          0326  4          !
;  330          0327  4          IF .CTX [ CTX$B_LEVEL ] GTRU 0
;  331          0328  4          THEN
;  332          0329  4              LAST_VBN = GET_LAST_VBN();
;  333          0330  4
;  334          0331  4          ! Swap the buffers
;  335          0332  4          !
;  336          0333  4          RECL$$SWAP_BUFFERS();
;  337          0334  4
;  338          0335  4          ! Get the next bucket in the chain which should be the first
;  339          0336  4          !
;  340          0337  4          RECL$$GET_NEXT_BUCKET();
;  341          0338  4
;  342          0339  4          ! If index level set the first index record vbn (because of duplicates
;  343          0340  4          ! etc. this is not always true for data level buckets)
;  344          0341  4          !
;  345          0342  4          IF .CTX [ CTX$B_LEVEL ] GTRU 0
;  346          0343  4          THEN
;  347          0344  4              CTX [ CTX$L_FIRST_VBN ] = .CTX [ CTX$L_CURRENT_VBN ]
;  348          0345  4
;  349          0346  4          END
;  350          0347  4
;  351          0348  2      END;
;  352          0349  2
;  353          0350  2      RETURN RECL$_SUCCESS
;  354          0351  2
```

```
;  355          0352  1     END;


                                        .TITLE  RECL$RMSIO VAX-11 CONVERT/RECLAIM
                                        .IDENT  \V04-000\

                                        .PSECT  _CONV$RECL_D,NOEXE,  PIC,2

                              00000 RECL$GL_SEARCH_BUFFER::
                                        .BLKB   4
                     00000000 00004 RECL$GL_WRITE_AREA::
                                        .LONG   0                                       ;
                     00000000 00008 RECL$GL_WRITE_KEY::
                                        .LONG   0                                       ;

                                        .EXTRN  CONVERT$_FACILITY
                                        .EXTRN  CONV$_FAO_MAX, CONV$_BADBLK
                                        .EXTRN  CONV$_BADLOGIC, CONV$_BADSORT
                                        .EXTRN  CONV$_CONFQUAL, CONV$_CREATEDSTM
                                        .EXTRN  CONV$_CREA_ERR, CONV$_DELPRI
                                        .EXTRN  CONV$_DUP, CONV$_EXTN_ERR
                                        .EXTRN  CONV$_FATALEXC, CONV$_FILLIM
                                        .EXTRN  CONV$_IDX_LIM, CONV$_ILL_KEY
                                        .EXTRN  CONV$_ILL_VALUE
                                        .EXTRN  CONV$_INP_FILES
                                        .EXTRN  CONV$_INSVIRMEM
                                        .EXTRN  CONV$_INVBKT, CONV$_KEY
                                        .EXTRN  CONV$_KEYREF, CONV$_LOADIDX
                                        .EXTRN  CONV$_NARG, CONV$_NI
                                        .EXTRN  CONV$_NOKEY, CONV$_NOTIDX
                                        .EXTRN  CONV$_NOTSEQ, CONV$_NOWILD
                                        .EXTRN  CONV$_ORDER, CONV$_OPENEXC
                                        .EXTRN  CONV$_OPENIN, CONV$_OPENOUT
                                        .EXTRN  CONV$_PAD, CONV$_PLV
                                        .EXTRN  CONV$_PROERR, CONV$_PROL_WRT
                                        .EXTRN  CONV$_READERR, CONV$_RSK
                                        .EXTRN  CONV$_RSZ, CONV$_RTL
                                        .EXTRN  CONV$_RTS, CONV$_SEQ
                                        .EXTRN  CONV$_UDF_BKS, CONV$_UDF_BLK
                                        .EXTRN  CONV$_VFC, CONV$_WRITEERR
                                        .EXTRN  RECL$$SWAP_BUFFERS
                                        .EXTRN  CONV$$READ_PROLOGUE
                                        .EXTRN  CONV$$RMS_ERROR
                                        .EXTRN  CONV$$RMS_OPEN_ERROR
                                        .EXTRN  CONV$$RMS_READ_ERROR
                                        .EXTRN  CONV$$GET_VM, CONV$$GET_TEMP_VM
                                        .EXTRN  CONV$AB_FLAGS, CONV$AB_OUT_FAB
                                        .EXTRN  CONV$AB_OUT_NAM
                                        .EXTRN  CONV$AB_OUT_RAB
                                        .EXTRN  CONV$AB_OUT_XABSUM
                                        .EXTRN  RECL$GL_BUCKET_COUNT
                                        .EXTRN  RECL$GL_DATA_COUNT
                                        .EXTRN  RECL$GL_INDEX_COUNT

                                        .PSECT  _CONV$RECL_S,NOWRT,  SHR,  PIC,2

                  1C  BB 00000 RECL$$ALLOCATE_BUFFERS::
```

```
                                                    PUSHR   #^M<R2,R3,R4>                           ; 0125
                   54         09   AB   9A 00002    MOVZBL  9(KEY_DESC), R4                         ; 0179
      52           54   0000005C   8F   C5 00006    MULL3   #92, R4, R2                             ;
                   52         5C   A2   9E 0000E    MOVAB   92(R2), CTX_SIZE                        ;
                   53         0B   AB   9A 00012    MOVZBL  11(KEY_DESC), DAT_BUF_SIZE              ; 0183
      53           53              09   78 00016    ASHL    #9, DAT_BUF_SIZE, DAT_BUF_SIZE          ;
                   51         0A   AB   9A 0001A    MOVZBL  10(KEY_DESC), IDX_BUF_SIZE              ; 0187
      51           51              09   78 0001E    ASHL    #9, IDX_BUF_SIZE, IDX_BUF_SIZE          ;
      50           51              54   C5 00022    MULL3   R4, IDX_BUF_SIZE, R0                    ; 0193
                   50              53   C0 00026    ADDL2   DAT_BUF_SIZE, R0                        ;
                   50            6240   3E 00029    MOVAW   (CTX_SIZE)[R0], R0                      ; 0192
                   50              51   C0 0002D    ADDL2   IDX_BUF_SIZE, BYTES                     ; 0194
                                   50   DD 00030    PUSHL   BYTES                                   ; 0198
                                0000G   30 00032    BSBW    CONV$$GET_TEMP_VM                       ;
                   5E              04   C0 00035    ADDL2   #4, SP                                  ;
                   5A              50   D0 00038    MOVL    R0, CTX                                 ;
             58    AA              53   B0 0003B    MOVW    DAT_BUF_SIZE, 88(CTX)                   ; 0202
             01    AA         08   AB   90 0003F    MOVB    8(KEY_DESC), 1(CTX)                     ; 0203
             24    AA         54   AB   D0 00044    MOVL    84(KEY_DESC), 36(CTX)                   ; 0204
  04   AA          5A              52   C1 00049    ADDL3   CTX_SIZE, CTX, 4(CTX)                   ; 0208
             40    AA      04 BA43      9E 0004E    MOVAB   @4(CTX)[DAT_BUF_SIZE], 64(CTX)          ; 0210
      52           53         40   AA   C1 00054    ADDL3   64(CTX), DAT_BUF_SIZE, BUF_PTR          ; 0221
      50           51              01   78 00059    ASHL    #1, IDX_BUF_SIZE, R0                    ; 0249
                                   53   D4 0005D    CLRL    I                                       ;
                                   29   11 0005F    BRB     4$                                      ;
                   5A         5C   AA   9E 00061 1$: MOVAB  92(R10), CTX                            ; 0231
             04    AA              52   D0 00065    MOVL    BUF_PTR, 4(CTX)                         ; 0235
  40   AA          52              51   C1 00069    ADDL3   IDX_BUF_SIZE, BUF_PTR, 64(CTX)          ; 0236
             58    AA              51   B0 0006E    MOVW    IDX_BUF_SIZE, 88(CTX)                   ; 0237
             02    AA              53   90 00072    MOVB    I, 2(CTX)                               ; 0238
                   01              53   D1 00076    CMPL    I, #1                                   ; 0243
                                   07   12 00079    BNEQ    2$                                      ;
             01    AA         07   AB   90 0007B    MOVB    7(KEY_DESC), 1(CTX)                     ; 0245
                                   05   11 00080    BRB     3$                                      ;
             01    AA         06   AB   90 00082 2$: MOVB   6(KEY_DESC), 1(CTX)                     ; 0247
                   52              50   C0 00087 3$: ADDL2  R0, BUF_PTR                             ; 0249
       D3          53              54   F3 0008A 4$: AOBLEQ R4, I, 1$                               ;
          0000'    CF              52   D0 0008E    MOVL    BUF_PTR, RECL$GL_SEARCH_BUFFER          ; 0255
             50    AA         0C   AB   D0 00093    MOVL    12(KEY_DESC), 80(CTX)                   ; 0271
             24    AA         0C   AB   D0 00098    MOVL    12(KEY_DESC), 36(CTX)                   ; 0272
             44    AA         0C   AB   D0 0009D    MOVL    12(KEY_DESC), 68(CTX)                   ; 0273
                                0000V   30 000A2    BSBW    RECL$$GET_NEXT_BUCKET                   ; 0277
                                0000V   30 000A5    BSBW    GET_LAST_VBN                            ; 0281
                   54              50   D0 000A8    MOVL    R0, LAST_VBN                            ;
                   53         09   AB   9A 000AB    MOVZBL  9(KEY_DESC), R3                         ; 0285
                                   52   D4 000AF    CLRL    I                                       ;
                                   2F   11 000B1    BRB     9$                                      ;
                   5A         A4   AA   9E 000B3 5$: MOVAB  -92(R10), CTX                           ; 0291
             50    AA              54   D0 000B7    MOVL    LAST_VBN, 80(CTX)                       ; 0295
                                0000V   30 000BB 6$: BSBW   RECL$$GET_NEXT_BUCKET                   ; 0299
                   05         0D   A9   E8 000BE    BLBS    13(BUCKET), 7$                          ; 0311
                                0000G   30 000C2    BSBW    RECL$$SWAP_BUFFERS                      ; 0317
                                   F4   11 000C5    BRB     6$                                      ; 0321
                              02   AA   95 000C7 7$: TSTB   2(CTX)                                  ; 0327
                                   06   13 000CA    BEQL    8$                                      ;
                                0000V   30 000CC    BSBW    GET_LAST_VBN                            ; 0329
                   54              50   D0 000CF    MOVL    R0, LAST_VBN                            ;
```

```
                        0000G 30 000D2 8$:      BSBW    RECL$$SWAP_BUFFERS                      ; 0333
                        0000V 30 000D5          BSBW    RECL$$GET_NEXT_BUCKET                   ; 0337
                    02    AA  95 000D8          TSTB    2(CTX)                                  ; 0342
                          05  13 000DB          BEQL    9$                                      ;
                 24     AA    08 AA  D0 000DD    MOVL    8(CTX), 36(CTX)                         ; 0344
              CD               52  53  F3 000E2 9$:      AOBLEQ  R3, I, 5$                       ; 0342
                               50  01  D0 000E6          MOVL    #1, R0                          ; 0350
                                   1C  BA 000E9          POPR    #^M<R2,R3,R4>                   ; 0352
                                       05 000EB          RSB                                     ;

; Routine Size:  236 bytes,    Routine Base:  _CONV$RECL_S + 0000
```

```
;   357          0353  1 %SBTTL  'GET_LAST_VBN'
;   358          0354  1 ROUTINE GET_LAST_VBN : RL$JSB_REG_9 =
;   359          0355  1 !++
;   360          0356  1 !
;   361          0357  1 ! Functional Description:
;   362          0358  1 !
;   363          0359  1 !       Returns the VBN pointer of the last index record in the bucket
;   364          0360  1 !
;   365          0361  1 ! Calling Sequence:
;   366          0362  1 !
;   367          0363  1 !       GET_LAST_VBN()
;   368          0364  1 !
;   369          0365  1 ! Input Parameters:
;   370          0366  1 !       none
;   371          0367  1 !
;   372          0368  1 ! Implicit Inputs:
;   373          0369  1 !       none
;   374          0370  1 !
;   375          0371  1 ! Output Parameters:
;   376          0372  1 !       none
;   377          0373  1 !
;   378          0374  1 ! Implicit Outputs:
;   379          0375  1 !       none
;   380          0376  1 !
;   381          0377  1 ! Routine Value:
;   382          0378  1 !       VBN of the last index record in the bucket
;   383          0379  1 !
;   384          0380  1 ! Routines Called:
;   385          0381  1 !       none
;   386          0382  1 !
;   387          0383  1 ! Side Effects:
;   388          0384  1 !       none
;   389          0385  1 !
;   390          0386  1 !--
;   391          0387  1
;   392          0388  2     BEGIN
;   393          0389  2
;   394          0390  2     DEFINE_CTX;
;   395          0391  2     DEFINE_BUCKET;
;   396          0392  2     DEFINE_KEY_DESC;
;   397          0393  2
;   398          0394  2     LOCAL
;   399          0395  2         VBN_FREE_SPACE,
;   400          0396  2         VBN_POINTER;
;   401          0397  2
;   402          0398  2     VBN_POINTER = .CTX [ CTX$W_BUCKET_SIZE ] - 4;
;   403          0399  2
;   404          0400  2     VBN_FREE_SPACE = .BUCKET[ .VBN_POINTER, 0, 16, 0 ];
;   405          0401  2
;   406          0402  2     VBN_POINTER = .VBN_FREE_SPACE + 1;
;   407          0403  2
;   408          0404  2     RETURN .BUCKET [    .VBN_POINTER,
;   409          0405  2                         0,
;   410          0406  2                         (( .BUCKET [ BKT$V_PTR_SZ ] + 2 ) * 8 ),
;   411          0407  2                         0 ]
;   412          0408  2
;   413          0409  1     END;
```

```
                                        52  DD 00000 GET_LAST_VBN:
                                                     PUSHL   R2                                                      ; 0354
                                51      58   AA  3C 00002         MOVZWL  88(CTX), VBN_POINTER                       ; 0398
                                51           03  C2 00006         SUBL2   #3, VBN_POINTER                            ;
                                          7149  9F 00009         PUSHAB  -(VBN_POINTER)[BUCKET]                     ; 0400
                                50           9E  3C 0000C         MOVZWL  @(SP)+, VBN_FREE_SPACE                     ;
                                51      01   A0  9E 0000F         MOVAB   1(R0), VBN_POINTER                         ; 0402
        50      0D   A9         02           03  EF 00013         EXTZV   #3, #2, 13(BUCKET), R0                     ; 0406
                                50           08  C4 00019         MULL2   #8, R0                                     ;
                                50           10  C0 0001C         ADDL2   #16, R0                                    ;
        52          6149        50           00  EF 0001F         EXTZV   #0, R0, (VBN_POINTER)[BUCKET], R2          ;
                                50           52  D0 00025         MOVL    R2, R0                                     ; 0404
                                             04  BA 00028         POPR    #^M<R2>                                    ; 0409
                                                 05 0002A         RSB                                                ;

; Routine Size:  43 bytes,    Routine Base:  _CONV$RECL_S + 00EC
```

```
;  415     0410  1 %SBTTL  'OPEN_FILE'
;  416     0411  1 GLOBAL ROUTINE  RECL$$OPEN_FILE ( FILE_NAME : REF BLOCK [ ,BYTE ] ) =
;  417     0412  1 !++
;  418     0413  1 !
;  419     0414  1 ! Functional Description:
;  420     0415  1 !
;  421     0416  1 !       Opens the input file described by the string descriptor FILE_NAME
;  422     0417  1 !
;  423     0418  1 ! Calling Sequence:
;  424     0419  1 !
;  425     0420  1 !       RECL$$OPEN_FILE( file_name )
;  426     0421  1 !
;  427     0422  1 ! Input Parameters:
;  428     0423  1 !
;  429     0424  1 !       file_name       - Address of a descriptor
;  430     0425  1 !
;  431     0426  1 ! Implicit Inputs:
;  432     0427  1 !
;  433     0428  1 !
;  434     0429  1 ! Output Parameters:
;  435     0430  1 !       none
;  436     0431  1 !
;  437     0432  1 ! Implicit Outputs:
;  438     0433  1 !       none
;  439     0434  1 !
;  440     0435  1 ! Routine Value:
;  441     0436  1 !
;  442     0437  1 !       RMS or CONVERT error code or ss$_normal
;  443     0438  1 !
;  444     0439  1 ! Routines Called:
;  445     0440  1 !
;  446     0441  1 !       CONV$$GET_VM
;  447     0442  1 !       $PARSE
;  448     0443  1 !       CONV$$RMS_OPEN_ERROR
;  449     0444  1 !       $SEARCH
;  450     0445  1 !       $OPEN
;  451     0446  1 !       $DISPLAY
;  452     0447  1 !       $CONNECT
;  453     0448  1 !       CONV$$GET_VM
;  454     0449  1 !       $READ
;  455     0450  1 !       CONV$$RMS_ERROR
;  456     0451  1 !
;  457     0452  1 ! Side Effects:
;  458     0453  1 !
;  459     0454  1 !       Opens the input file and reads in the prologue
;  460     0455  1 !
;  461     0456  1 !--
;  462     0457  1
;  463     0458  2     BEGIN
;  464     0459  2
;  465     0460  2     LOCAL
;  466     0461  2         VM_POINTER,
;  467     0462  2         BYTES;
;  468     0463  2
;  469     0464  2     ! Allocate some name block buffers
;  470     0465  2     !
;  471     0466  2     BYTES = ESA_BUF_SIZ + RSA_BUF_SIZ;
```

```
;  472          0467  2
;  473          0468  2      VM_POINTER = CONV$$GET_VM ( .BYTES );
;  474          0469  2
;  475          0470  2      ! Initialize the rms blocks
;  476          0471  2      !
;  477        P 0472  2      $NAM_INIT ( NAM = CONV$AB_OUT_NAM,
;  478        P 0473  2                  ESA = .VM_POINTER,
;  479        P 0474  2                  ESS = ESA_BUF_SIZ,
;  480        P 0475  2                  RSA = .VM_POINTER + ESA_BUF_SIZ,
;  481          0476  2                  RSS = RSA_BUF_SIZ );
;  482          0477  2
;  483        P 0478  2      $FAB_INIT ( FAB = CONV$AB_OUT_FAB,
;  484        P 0479  2                  CTX = CONV$_OPENIN,
;  485        P 0480  2                  FAC = <BRO,GET,PUT>,
;  486        P 0481  2                  FNA = .FILE_NAME [ DSC$A_POINTER ],
;  487        P 0482  2                  FNS = .FILE_NAME [ DSC$W_LENGTH ],
;  488        P 0483  2                  FOP = NAM,
;  489        P 0484  2                  NAM = CONV$AB_OUT_NAM,
;  490          0485  2                  XAB = CONV$AB_OUT_XABSUM );
;  491          0486  2
;  492        P 0487  2      $RAB_INIT ( RAB = CONV$AB_OUT_RAB,
;  493        P 0488  2                  FAB = CONV$AB_OUT_FAB,
;  494          0489  2                  ROP = BIO );
;  495          0490  2
;  496        P 0491  2      $XABSUM_INIT ( XAB = CONV$AB_OUT_XABSUM,
;  497          0492  2                     NXT = 0 );
;  498          0493  2
;  499          0494  2      ! Parse the file name
;  500          0495  2      !
;  501          0496  2      $PARSE( FAB=CONV$AB_OUT_FAB,ERR=CONV$$RMS_OPEN_ERROR );
;  502          0497  2
;  503          0498  2      ! We are not allowing wildcards
;  504          0499  2      !
;  505          0500  2      IF .CONV$AB_OUT_NAM [ NAM$V_WILDCARD ]
;  506          0501  2      THEN
;  507          0502  2          RETURN CONV$_NOWILD;
;  508          0503  2
;  509          0504  2      ! Search for the file
;  510          0505  2      !
;  511          0506  2      $SEARCH( FAB=CONV$AB_OUT_FAB,ERR=CONV$$RMS_OPEN_ERROR );
;  512          0507  2
;  513          0508  2      ! Open the file
;  514          0509  2      !
;  515          0510  2      $OPEN( FAB=CONV$AB_OUT_FAB,ERR=CONV$$RMS_OPEN_ERROR );
;  516          0511  2
;  517          0512  2      ! Get all good info about the file
;  518          0513  2      !
;  519          0514  2      $DISPLAY( FAB=CONV$AB_OUT_FAB );
;  520          0515  2
;  521          0516  2      ! If the file is not index then error
;  522          0517  2      !
;  523          0518  2      IF .CONV$AB_OUT_FAB [ FAB$B_ORG ] NEQU FAB$C_IDX
;  524          0519  2      THEN
;  525          0520  2          RETURN CONV$_NOTIDX;
;  526          0521  2
;  527          0522  2      ! Make sure it is the correct prologue version
;  528          0523  2      !
```

```
;  529     0524  2     IF .CONV$AB_OUT_XABSUM [ XAB$W_PVN ] LSS 3
;  530     0525  2     THEN
;  531     0526  2         RETURN CONV$_PLV;
;  532     0527  2
;  533     0528  2     ! Connect the stream
;  534     0529  2     !
;  535     0530  2     $CONNECT( RAB=CONV$AB_OUT_RAB,ERR=CONV$$RMS_OPEN_ERROR );
;  536     0531  2
;  537     0532  2     ! Say that the file is open
;  538     0533  2     !
;  539     0534  2     CONV$AB_FLAGS [ CONV$V_OUT ] = _SET;
;  540     0535  2
;  541     0536  2     ! Read the prologue
;  542     0537  2     !
;  543     0538  2     CONV$$READ_PROLOGUE();
;  544     0539  2
;  545     0540  2     RETURN RECL$_SUCCESS
;  546     0541  2
;  547     0542  1     END;

                                                    .EXTRN  SYS$PARSE, SYS$SEARCH
                                                    .EXTRN  SYS$OPEN, SYS$DISPLAY
                                                    .EXTRN  SYS$CONNECT

                                    0FFC 00000      .ENTRY  RECL$$OPEN_FILE, Save R2,R3,R4,R5,R6,R7,R8,-; 0411
                                                            R9,R10,R11
                                5B      0000G   CF  9E 00002    MOVAB   CONV$$RMS_OPEN_ERROR, R11
                                5A      0000G   CF  9E 00007    MOVAB   CONV$AB_OUT_XABSUM, R10
                                59      0000G   CF  9E 0000C    MOVAB   $RMS_PTR, R9
                                58      0000G   CF  9E 00011    MOVAB   $RMS_PTR, R8
                                57      0000G   CF  9E 00016    MOVAB   $RMS_PTR, R7
                                50        A0    8F  9A 0001B    MOVZBL  #160, BYTES                         0466
                                            50  DD  0001F      PUSHL   BYTES                               0468
                                        0000G   30  00021      BSBW    CONV$$GET_VM
                                5E              04  C0 00024  ADDL2   #4, SP
                                56              50  D0 00027  MOVL    R0, VM_POINTER
0060    8F      00              6E              00  2C 0002A  MOVC5   #0, (SP), #0, #96, $RMS_PTR          0476
                                                68      00051
                                68        6002  8F  B0 00032  MOVW    #24578, $RMS_PTR
                            02  A8          50  8F  90 00037  MOVB    #80, $RMS_PTR+2
                            04  A8          50  A6  9E 0003C  MOVAB   80(R6), $RMS_PTR+4
                            0A  A8          50  8F  90 00041  MOVB    #80, $RMS_PTR+10
                            0C  A8              56  D0 00046  MOVL    VM_POINTER, $RMS_PTR+12
0050    8F      00              6E              00  2C 0004A  MOVC5   #0, (SP), #0, #80, $RMS_PTR          0485
                                                67      00051
                                67        5003  8F  B0 00052  MOVW    #20483, $RMS_PTR
                            04  A7  01000000    8F  D0 00057  MOVL    #16777216, $RMS_PTR+4
                            16  A7          43  8F  90 0005F  MOVB    #67, $RMS_PTR+22
                            18  A7  00000000G   8F  D0 00064  MOVL    #CONV$_OPENIN, $RMS_PTR+24
                            1F  A7              02  90 0006C  MOVB    #2, $RMS_PTR+31
                            24  A7              6A  9E 00070  MOVAB   CONV$AB_OUT_XABSUM, $RMS_PTR+36
                            28  A7              68  9E 00074  MOVAB   CONV$AB_OUT_NAM, $RMS_PTR+40
                                50          04  AC  D0 00078  MOVL    FILE_NAME, R0
                            2C  A7          04  A0  D0 0007C  MOVL    4(R0), $RMS_PTR+44
                            34  A7              60  90 00081  MOVB    (R0), $RMS_PTR+52
```

```
0044  8F              00              6E              00  2C 00085          MOVC5   #0, (SP), #0, #68, $RMS_PTR          ; 0489
                                                      69  0008C
                                      69      4401    8F  B0 0008D          MOVW    #17409, $RMS_PTR
                                   04 A9      0800    8F  3C 00092          MOVZWL  #2048, $RMS_PTR+4
                                   3C A9              67  9E 00098          MOVAB   CONV$AB_OUT_FAB, $RMS_PTR+60
                0C              00              6E              00  2C 0009C          MOVC5   #0, (SP), #0, #12, $RMS_PTR          ; 0492
                                                      6A  000A1
                                      6A      0C16    8F  B0 000A2          MOVW    #3094, $RMS_PTR
                                              0880    8F  BB 000A7          PUSHR   #^M<R7,R11>                          ; 0496
                                00000000G     00              02  FB 000AB          CALLS   #2, SYS$PARSE
                                              08        35    A8  E9 000B2          BLBC    CONV$AB_OUT_NAM+53, 1$               ; 0500
                                              50  00000000G    8F  D0 000B6          MOVL    #CONV$_NOWILD, R0                    ; 0502
                                                              04  000BD          RET
                                              0880    8F  BB 000BE 1$:      PUSHR   #^M<R7,R11>                          ; 0506
                                00000000G     00              02  FB 000C2          CALLS   #2, SYS$SEARCH
                                              0880    8F  BB 000C9          PUSHR   #^M<R7,R11>                          ; 0510
                                00000000G     00              02  FB 000CD          CALLS   #2, SYS$OPEN
                                                              57  DD 000D4          PUSHL   R7                                   ; 0514
                                00000000G     00              01  FB 000D6          CALLS   #1, SYS$DISPLAY
                                              20        1D    A7  91 000DD          CMPB    CONV$AB_OUT_FAB+29, #32              ; 0518
                                                              08  13 000E1          BEQL    2$
                                              50  00000000G    8F  D0 000E3          MOVL    #CONV$_NOTIDX, R0                    ; 0520
                                                              04  000EA          RET
                                              03        0A    AA  B1 000EB 2$:      CMPW    CONV$AB_OUT_XABSUM+10, #3            ; 0524
                                                              08  1E 000EF          BGEQU   3$
                                              50  00000000G    8F  D0 000F1          MOVL    #CONV$_PLV, R0                       ; 0526
                                                              04  000F8          RET
                                                    0A00      8F  BB 000F9 3$:      PUSHR   #^M<R9,R11>                          ; 0530
                                00000000G     00              02  FB 000FD          CALLS   #2, SYS$CONNECT
                                    0000G     CF              02  88 00104          BISB2   #2, CONV$AB_FLAGS+2                  ; 0534
                                                        0000G 30  00109          BSBW    CONV$$READ_PROLOGUE                  ; 0538
                                              50              01  D0 0010C          MOVL    #1, R0                               ; 0540
                                                              04  0010F          RET                                          ; 0542

; Routine Size:  272 bytes,    Routine Base:  _CONV$RECL_S + 0117

;  548          0543  1
```

```
;  550          0544  1 %SBTTL  'GET_NEXT_BUCKET'
;  551          0545  1 GLOBAL ROUTINE  RECL$$GET_NEXT_BUCKET : RL$JSB_REG_9 NOVALUE =
;  552          0546  1 !++
;  553          0547  1 !
;  554          0548  1 ! Functional Description:
;  555          0549  1 !
;  556          0550  1 !       Gets the bucket in the horizontal chain
;  557          0551  1 !
;  558          0552  1 ! Calling Sequence:
;  559          0553  1 !
;  560          0554  1 !       RECL$$GET_NEXT_BUCKET()
;  561          0555  1 !
;  562          0556  1 ! Input Parameters:
;  563          0557  1 !       none
;  564          0558  1 !
;  565          0559  1 ! Implicit Inputs:
;  566          0560  1 !       none
;  567          0561  1 !
;  568          0562  1 ! Output Parameters:
;  569          0563  1 !       none
;  570          0564  1 !
;  571          0565  1 ! Implicit Outputs:
;  572          0566  1 !       none
;  573          0567  1 !
;  574          0568  1 ! Routine Value:
;  575          0569  1 !       none
;  576          0570  1 !
;  577          0571  1 ! Routines Called:
;  578          0572  1 !
;  579          0573  1 !       $READ
;  580          0574  1 !
;  581          0575  1 ! Side Effects:
;  582          0576  1 !       none
;  583          0577  1 !
;  584          0578  1 !--
;  585          0579  1
;  586          0580  2     BEGIN
;  587          0581  2
;  588          0582  2     DEFINE_CTX;
;  589          0583  2     DEFINE_BUCKET;
;  590          0584  2     DEFINE_KEY_DESC;
;  591          0585  2
;  592          0586  2     LOCAL
;  593          0587  2         LOW_ADDRESS_WORD        : WORD,
;  594          0588  2         LAST_BYTE;
;  595          0589  2
;  596          0590  2     ! Set the bucket pointer to the current buffer at this level
;  597          0591  2     !
;  598          0592  2     BUCKET = .CTX [ CTX$L_CURRENT_BUFFER ];
;  599          0593  2
;  600          0594  2     ! A simple check could save an IO (unfortunatly it causes problems)
;  601          0595  2     !
;  602          0596  2 !   IF ( NOT ( .CTX [ CTX$L_CURRENT_VBN ] EQLU .CTX [ CTX$L_NEXT_VBN ] ) )
;  603          0597  2 !   THEN
;  604          0598  3         BEGIN
;  605          0599  3
;  606          0600  3         ! Point RMS to the target bucket
```

```
;  607     0601  3          !
;  608     0602  3          CONV$AB_OUT_RAB [ RAB$L_UBF ] = .BUCKET;
;  609     0603  3          CONV$AB_OUT_RAB [ RAB$W_USZ ] = .CTX [ CTX$W_BUCKET_SIZE ];
;  610     0604  3          CONV$AB_OUT_RAB [ RAB$L_BKT ] = .CTX [ CTX$L_NEXT_VBN ];
;  611     0605  3
;  612     0606  3          ! If error signal a readerr
;  613     0607  3          !
;  614     0608  3          CONV$AB_OUT_RAB [ RAB$L_CTX ] = CONV$_READERR;
;  615     0609  3
;  616     0610  3          ! Get the bucket
;  617     0611  3          !
;  618     0612  3          $READ( RAB=CONV$AB_OUT_RAB,ERR=CONV$$RMS_READ_ERROR );
;  619     0613  3
;  620     0614  3          ! Check to see if the bucket is valid
;  621     0615  3          !
;  622     0616  3          ! Check the address sample and check bytes
;  623     0617  3          !
;  624     0618  3          LAST_BYTE = .CTX [ CTX$W_BUCKET_SIZE ] - 1;
;  625     0619  3
;  626     0620  3          ! Get the low word of the longword address
;  627     0621  3          !
;  628     0622  3          LOW_ADDRESS_WORD = .CTX [ CTX$L_NEXT_VBN ];
;  629     0623  3
;  630     0624  3          ! If there are any errors signal them and stop
;  631     0625  3          !
;  632     0626  4          IF ( .BUCKET [ BKT$W_ADRSAMPLE ] NEQU .LOW_ADDRESS_WORD )
;  633     0627  3              OR
;  634     0628  4             ( .BUCKET [ BKT$B_CHECKCHAR ] NEQU .BUCKET [ .LAST_BYTE,0,8,0 ] )
;  635     0629  3          THEN
;  636     0630  4              BEGIN
;  637     0631  4
;  638     0632  4              LOCAL
;  639     0633  4                  NAM_DESC        : DESC_BLK;
;  640     0634  4
;  641     0635  4              NAM_DESC [ DSC$W_LENGTH ] = .CONV$AB_OUT_NAM [ NAM$B_RSL ];
;  642     0636  4              NAM_DESC [ DSC$A_POINTER ] = .CONV$AB_OUT_NAM [ NAM$L_RSA ];
;  643     0637  4
;  644     0638  4              ! Signal a readerr with the file name and the vbn which broke
;  645     0639  4              !
;  646     0640  4              SIGNAL_STOP( CONV$_READERR,
;  647     0641  4                           1,
;  648     0642  4                           NAM_DESC,
;  649     0643  4                           CONV$_INVBKT,
;  650     0644  4                           1,
;  651     0645  4                           .CTX [ CTX$L_NEXT_VBN ] )
;  652     0646  4
;  653     0647  3              END;
;  654     0648  3
;  655     0649  3          ! Set the new buckets vbn
;  656     0650  3          !
;  657     0651  3          CTX [ CTX$L_CURRENT_VBN ] = .CTX [ CTX$L_NEXT_VBN ];
;  658     0652  3
;  659     0653  2          END;
;  660     0654  2
;  661     0655  2      ! Set the next bucket address
;  662     0656  2      !
;  663     0657  2      CTX [ CTX$L_NEXT_VBN ] = .BUCKET [ BKT$L_NXTBKT ];
```

```
;  664          0658  2
;  665          0659  2      RETURN
;  666          0660  2
;  667          0661  1      END;


                                                .EXTRN  SYS$READ

                        5E              08  C2 00000 RECL$$GET_NEXT_BUCKET::
                                                  SUBL2   #8, SP                                  ; 0545
                        59        04    AA  D0 00003      MOVL    4(CTX), BUCKET                  ; 0592
                0000G   CF              59  D0 00007      MOVL    BUCKET, CONV$AB_OUT_RAB+36      ; 0602
                0000G   CF        58    AA  B0 0000C      MOVW    88(CTX), CONV$AB_OUT_RAB+32     ; 0603
                0000G   CF        50    AA  D0 00012      MOVL    80(CTX), CONV$AB_OUT_RAB+56     ; 0604
                0000G   CF 00000000G    8F  D0 00018      MOVL    #CONV$_READERR, CONV$AB_OUT_RAB+24  ; 0608
                             0000G      CF  9F 00021      PUSHAB  CONV$$RMS_READ_ERROR            ; 0612
                             0000G      CF  9F 00025      PUSHAB  CONV$AB_OUT_RAB
        00000000G   00                   02  FB 00029      CALLS   #2, SYS$READ
                        51        58    AA  3C 00030      MOVZWL  88(CTX), LAST_BYTE              ; 0618
                                        51  D7 00034      DECL    LAST_BYTE
                        50        50    AA  B0 00036      MOVW    80(CTX), LOW_ADDRESS_WORD       ; 0622
                        50        02    A9  B1 0003A      CMPW    2(BUCKET), LOW_ADDRESS_WORD     ; 0626
                                        06  12 0003E      BNEQ    1$
                      6149              69  91 00040      CMPB    (BUCKET), (LAST_BYTE)[BUCKET]   ; 0628
                                        28  13 00044      BEQL    2$
                        6E      0000G   CF  9B 00046 1$:  MOVZBW  CONV$AB_OUT_NAM+3, NAM_DESC     ; 0635
                    04  AE      0000G   CF  D0 0004B      MOVL    CONV$AB_OUT_NAM+4, NAM_DESC+4   ; 0636
                                  50    AA  DD 00051      PUSHL   80(CTX)                         ; 0645
                                        01  DD 00054      PUSHL   #1                              ; 0640
                             00000000G  8F  DD 00056      PUSHL   #CONV$_INVBKT
                                   0C   AE  9F 0005C      PUSHAB  NAM_DESC
                                        01  DD 0005F      PUSHL   #1
                             00000000G  8F  DD 00061      PUSHL   #CONV$_READERR
        00000000G   00                   06  FB 00067      CALLS   #6, LIB$STOP
                    08  AA        50    AA  D0 0006E 2$:  MOVL    80(CTX), 8(CTX)                 ; 0651
                    50  AA        08    A9  D0 00073      MOVL    8(BUCKET), 80(CTX)              ; 0657
                        5E              08  C0 00078      ADDL2   #8, SP                          ; 0661
                                        05  0007B      RSB                                     ;

; Routine Size:  124 bytes,    Routine Base:  _CONV$RECL_S + 0227
```

```
;  669          0662  1 %SBTTL  'WRITE_BUCKET'
;  670          0663  1 GLOBAL ROUTINE  RECL$$WRITE_BUCKET ( BUCKET_BLOCK ) : RL$JSB_REG_9 NOVALUE =
;  671          0664  1 !++
;  672          0665  1 !
;  673          0666  1 ! Functional Description:
;  674          0667  1 !
;  675          0668  1 !       Writes the bucket pointed to by the first longword in bucket_block
;  676          0669  1 !       to the VBN given in the second longword
;  677          0670  1 !
;  678          0671  1 ! Calling Sequence:
;  679          0672  1 !
;  680          0673  1 !       RECL$$WRITE_BUCKET( bucket_block )
;  681          0674  1 !
;  682          0675  1 ! Input Parameters:
;  683          0676  1 !
;  684          0677  1 !       bucket_block    - Pointer to a two longword block which looks like:
;  685          0678  1 !
;  686          0679  1 !
;  687          0680  1 !                               Bucket_block :  ------------------
;  688          0681  1 !                                               | Buffer_address |
;  689          0682  1 !                                               |----------------|
;  690          0683  1 !                                               |   Target_vbn   |
;  691          0684  1 !                                               ------------------
;  692          0685  1 ! Implicit Inputs:
;  693          0686  1 !
;  694          0687  1 !       The size of the bucket to be writen is taken from the current
;  695          0688  1 !       context block
;  696          0689  1 !
;  697          0690  1 ! Output Parameters:
;  698          0691  1 !       none
;  699          0692  1 !
;  700          0693  1 ! Implicit Outputs:
;  701          0694  1 !       none
;  702          0695  1 !
;  703          0696  1 ! Routine Value:
;  704          0697  1 !       none
;  705          0698  1 !
;  706          0699  1 ! Routines Called:
;  707          0700  1 !
;  708          0701  1 !       $WRITE
;  709          0702  1 !
;  710          0703  1 ! Side Effects:
;  711          0704  1 !
;  712          0705  1 !       Bucket check bytes incremented.
;  713          0706  1 !
;  714          0707  1 !--
;  715          0708  1
;  716          0709  2     BEGIN
;  717          0710  2
;  718          0711  2     DEFINE_CTX;
;  719          0712  2     DEFINE_BUCKET;
;  720          0713  2     DEFINE_KEY_DESC;
;  721          0714  2
;  722          0715  2     LOCAL
;  723          0716  2         LAST_BYTE,
;  724          0717  2         OUT_BUCKET:     REF BLOCK [ ,BYTE ];
;  725          0718  2
```

```
;  726          0719  2     BIND
;  727          0720  2         BKT_BLK = BUCKET_BLOCK : REF VECTOR [ 2,LONG ];
;  728          0721  2
;  729          0722  2     ! Update the check bytes, first point to the buffer and then increment
;  730          0723  2     ! the first check character and then copy it to the last check character.
;  731          0724  2     !
;  732          0725  2     OUT_BUCKET = .BKT_BLK[ 0 ];
;  733          0726  2
;  734          0727  2     ! Point to the last byte (final check byte) in the buffer.
;  735          0728  2     !
;  736          0729  2     LAST_BYTE = .CTX [ CTX$W_BUCKET_SIZE ] - 1;
;  737          0730  2
;  738          0731  2     ! Actually update the check bytes.
;  739          0732  2     !
;  740          0733  2     OUT_BUCKET[ BKT$B_CHECKCHAR ] = .OUT_BUCKET[ BKT$B_CHECKCHAR ] + 1;
;  741          0734  2     OUT_BUCKET[ .LAST_BYTE, 0, 8, 0 ] = .OUT_BUCKET[ BKT$B_CHECKCHAR ];
;  742          0735  2
;  743          0736  2     ! Point RMS to the target bucket
;  744          0737  2     !
;  745          0738  2     CONV$AB_OUT_RAB [ RAB$L_RBF ] = .BKT_BLK [ 0 ];
;  746          0739  2     CONV$AB_OUT_RAB [ RAB$L_BKT ] = .BKT_BLK [ 1 ];
;  747          0740  2     CONV$AB_OUT_RAB [ RAB$W_RSZ ] = .CTX [ CTX$W_BUCKET_SIZE ];
;  748          0741  2
;  749          0742  2     ! If error signal a write error
;  750          0743  2     !
;  751          0744  2     CONV$AB_OUT_RAB [ RAB$L_CTX ] = CONV$_WRITEERR;
;  752          0745  2
;  753          0746  2     ! NOTE: rms_read_error also works for writes
;  754          0747  2     !
;  755          0748  2     $WRITE( RAB=CONV$AB_OUT_RAB,ERR=CONV$$RMS_READ_ERROR );
;  756          0749  2
;  757          0750  2     RETURN
;  758          0751  2
;  759          0752  1     END;


                                                .EXTRN  SYS$WRITE

                                    52  DD 00000 RECL$$WRITE_BUCKET::
                                                 PUSHL   R2                                                          ; 0663
                          51        08    AE  D0 00002                MOVL    BKT_BLK, R1                           ; 0725
                          50              61  D0 00006                MOVL    (R1), OUT_BUCKET                      ;
                          52        58    AA  3C 00009                MOVZWL  88(CTX), LAST_BYTE                    ; 0729
                                          60  96 0000D                INCB    (OUT_BUCKET)                          ; 0733
                        7240              60  90 0000F                MOVB    (OUT_BUCKET), -(LAST_BYTE)[OUT_BUCKET]  ; 0734
                0000G   CF                61  D0 00013                MOVL    (R1), CONV$AB_OUT_RAB+40              ; 0738
                0000G   CF        04      A1  D0 00018                MOVL    4(R1), CONV$AB_OUT_RAB+56             ; 0739
                0000G   CF        58      AA  B0 0001E                MOVW    88(CTX), CONV$AB_OUT_RAB+34           ; 0740
                0000G   CF 00000000G      8F  D0 00024                MOVL    #CONV$_WRITEERR, CONV$AB_OUT_RAB+24   ; 0744
                                  0000G   CF  9F 0002D                PUSHAB  CONV$$RMS_READ_ERROR                  ; 0748
                                  0000G   CF  9F 00031                PUSHAB  CONV$AB_OUT_RAB                       ;
          00000000G   00                    02  FB 00035             CALLS   #2, SYS$WRITE                         ;
                                          04  BA C003C                POPR    #^M<R2>                               ; 0752
                                              05 0003E                RSB                                           ;

; Routine Size:  63 bytes,    Routine Base:  _CONV$RECL_S + 02A3
```

```
;  760          0753  1
;  761          0754  0 END     ELUDOM


                                                        .EXTRN  LIB$STOP

;                               PSECT SUMMARY
;
;       Name                    Bytes                   Attributes
;
;  _CONV$RECL_D                   12  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  _CONV$RECL_S                  738  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(2)


;                           Library Statistics
;                                       --------  Symbols  --------      Pages       Processing
;       File                            Total     Loaded   Percent       Mapped      Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1         18619        112         0         1000        00:01.8
;  _$255$DUA28:[CONV.SRC]CONVERT.L32;1     165         25        15           17        00:00.2



;                                   COMMAND QUALIFIERS

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

; Size:          738 code + 12 data bytes
; Run Time:        00:20.7
; Elapsed Time:    01:12.2
; Lines/CPU Min:    2184
; Lexemes/CPU-Min: 27493
; Memory Used:  176 pages
; Compilation Complete
```

0066 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

RECLDCL
LIS

RECLREC
LIS

RECLCTRL
LIS

RECLRMSIO
LIS

CONVMSG
LIS

CONVMAIN
LIS

CONVSORT
LIS

CONVVEC
LIS

COPYMSG
REQ

COPYSPECS
LIS

COPY

COPY
MAP

VMSMAC
REQ

COPY
REQ

COPYCLI
LIS

COPYMAIN
LIS

COPYSEMAN
LIS