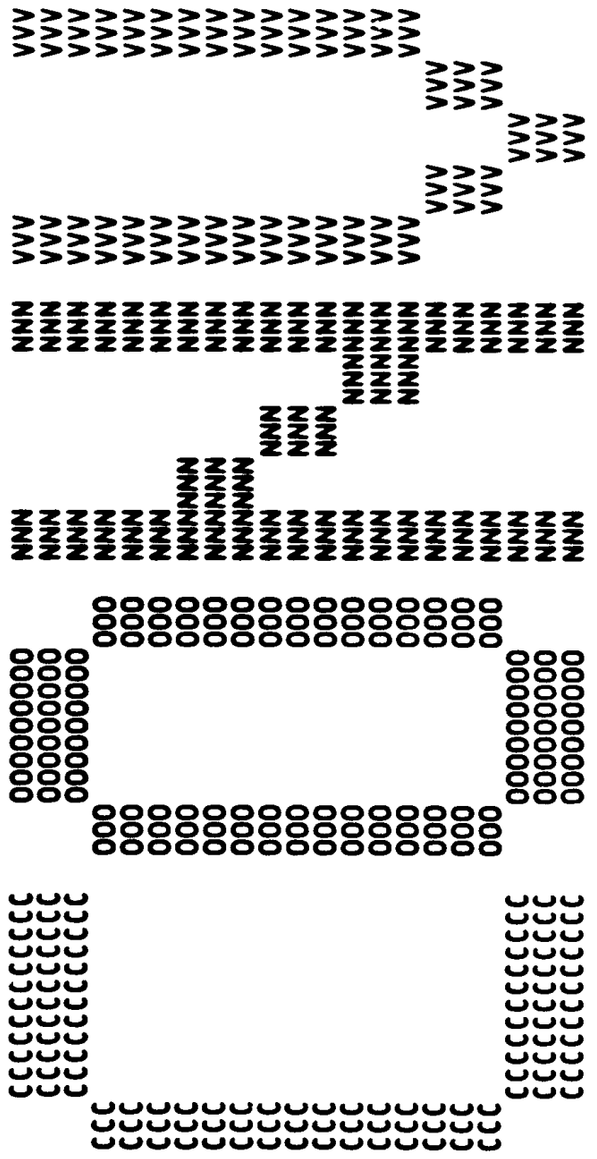
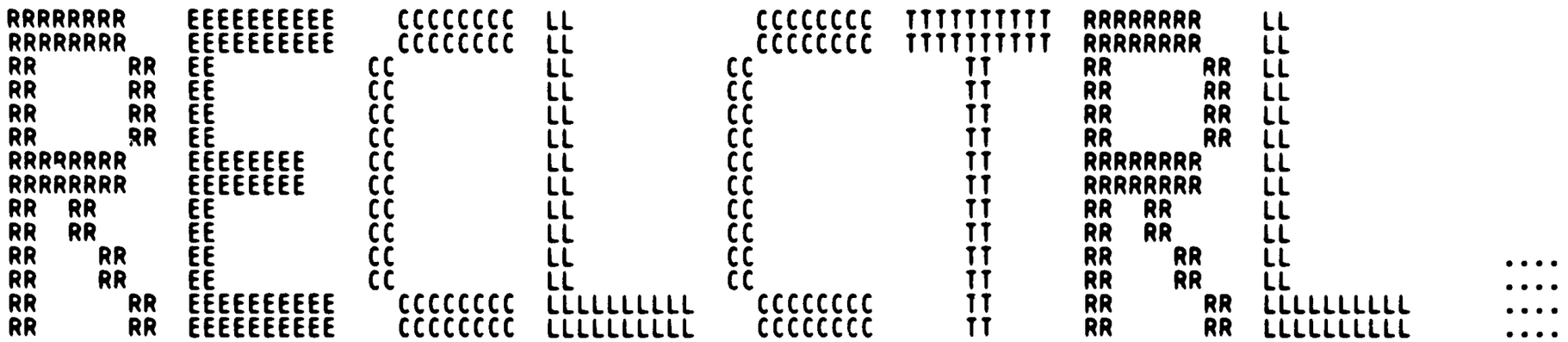


FILEID**RECLCTRL



```
;  1          0001  0 %TITLE  'VAX-11 CONVERT/RECLAIM'
;  2          0002  0 MODULE  RECL$CTRL       ( 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 !                VAX/VMS Operating System
;    37          0036  1 !
;    38          0037  1 ! Abstract:
;    39          0038  1 !
;    40          0039  1 !
;    41          0040  1 ! Contents:     SCAN_DATA_LEVEL
;    42          0041  1 !                UPDATE_INDEX
;    43          0042  1 !                REMOVE_BUCKET
;    44          0043  1 !                ZERO_BUCKET
;    45          0044  1 !                SWAP_BUFFERS
;    46          0045  1 !
;    47          0046  1 !--
;    48          0047  1
;    49          0048  1 !
;    50          0049  1 ! Author:       Keith B Thompson
;    51          0050  1 !                Peter Lieberwirth       Creation date:  September-1981
;    52          0051  1 !
;    53          0052  1 !
;    54          0053  1 ! Modified by:
;    55          0054  1 !
;    56          0055  1 !       V03-007 JWT0176         Jim Teague              13-Apr-1984
;    57          0056  1 !               Fix linkages to CONV$$WRITE_AREA_DESC and
;    58          0057  1 !               CONV$$WRITE_KEY_DESC.
;    59          0058  1 !
;    60          0059  1 !       V03-006 KBT0395         Keith B. Thompson       29-Oct-1982
;    61          0060  1 !               Add support for prologue 3 sidrs
;    62          0061  1 !
;    63          0062  1 !       V03-005 KBT0358         Keith B. Thompson       6-Oct-1982
;    64          0063  1 !               Use new merged ctx definitions
;    65          0064  1 !
;    66          0065  1 !       V03-004 KBT0353         Keith B. Thompson       5-Oct-1982
;    67          0066  1 !               Use new linkage definitions
;    68          0067  1 !
;    69          0068  1 !       V03-003 KBT0048         Keith Thompson          21-Apr-1982
;    70          0069  1 !               Do not reclaim the last index record in a bucket
;    71          0070  1 !
;    72          0071  1 !       V03-002 KBT0041         Keith Thompson          3-Apr-1982
;    73          0072  1 !               Add logic to swing index pointers if needed and fix index
;    74          0073  1 !               save bucket logic
;    75          0074  1 !
;    76          0075  1 !       V03-001 KBT0010         Keith Thompson          16-Mar-1982
;    77          0076  1 !               Fix a problem with end condition in update_index and
;    78          0077  1 !               add a few lines of comments.
;    79          0078  1 !
;    80          0079  1 !****
```

```
;    82          0080  1
;    83          0081  1 PSECT
;    84          0082  1         OWN     = _CONV$RECL_D  (PIC),
;    85          0083  1         GLOBAL  = _CONV$RECL_D  (PIC),
;    86          0084  1         PLIT    = _CONV$PLIT    (SHARE,PIC),
;    87          0085  1         CODE    = _CONV$RECL_S  (SHARE,PIC);
;    88          0086  1
;    89          0087  1 LIBRARY 'SYS$LIBRARY:LIB.L32';
;    90          0088  1 LIBRARY 'SRC$:CONVERT';
;    91          0089  1
;    92          0090  1 EXTERNAL ROUTINE
;    93          0091  1         RECL$$GET_NEXT_BUCKET   : RL$JSB_REG_9  NOVALUE,
;    94          0092  1         RECL$$BUCKET_EMPTY      : RL$JSB_REG_9,
;    95          0093  1         RECL$$GET_DOWN_POINTER  : RL$JSB_REG_8,
;    96          0094  1         RECL$$CHECK_LAST        : RL$JSB_REG_8,
;    97          0095  1         RECL$$COMPARE_POINTER   : RL$JSB_REG_8,
;    98          0096  1         RECL$$SWING_POINTER     : RL$JSB_REG_8  NOVALUE,
;    99          0097  1         RECL$$REMOVE_INDEX_RECORD: RL$JSB_REG_8,
;   100          0098  1         RECL$$WRITE_BUCKET      : RL$JSB_REG_9  NOVALUE,
;   101          0099  1         CONV$$WRITE_AREA_DESC   : CL$WRITE_AREA_DESC    NOVALUE,
;   102          0100  1         CONV$$WRITE_KEY_DESC    : CL$WRITE_KEY_DESC     NOVALUE;
;   103          0101  1
;   104          0102  1 FORWARD ROUTINE
;   105          0103  1         UPDATE_INDEX            : RL$JSB_REG_9,
;   106          0104  1         REMOVE_BUCKET           : RL$JSB_REG_9  NOVALUE,
;   107          0105  1         ZERO_BUCKET             : RL$JSB_REG_9  NOVALUE,
;   108          0106  1         RECL$$SWAP_BUFFERS      : RL$JSB_REG_9  NOVALUE;
;   109          0107  1
;   110          0108  1 EXTERNAL
;   111          0109  1         CONV$AR_AREA_BLOCK;
;   112          0110  1
```

```
;  114          0111  1 %SBTTL  'SCAN_DATA_LEVEL'
;  115          0112  1 GLOBAL ROUTINE  RECL$$SCAN_DATA_LEVEL : RL$JSB_REG_9 =
;  116          0113  1 !++
;  117          0114  1 !
;  118          0115  1 ! Functional Description:
;  119          0116  1 !
;  120          0117  1 !       This routine sequentially read along the data level buckets
;  121          0118  1 !       looking for an empty one.  If it finds one it trys to remove
;  122          0119  1 !       the index to it then trys to remove it.
;  123          0120  1 !
;  124          0121  1 ! Calling Sequence:
;  125          0122  1 !
;  126          0123  1 !       RECL$$SCAN_DATA_LEVEL()
;  127          0124  1 !
;  128          0125  1 ! Input Parameters:
;  129          0126  1 !       none
;  130          0127  1 !
;  131          0128  1 ! Implicit Inputs:
;  132          0129  1 !
;  133          0130  1 !       BUCKET
;  134          0131  1 !
;  135          0132  1 ! Output Parameters:
;  136          0133  1 !       none
;  137          0134  1 !
;  138          0135  1 ! Implicit Outputs:
;  139          0136  1 !       none
;  140          0137  1 !
;  141          0138  1 ! Routine Value:
;  142          0139  1 !
;  143          0140  1 !       normal
;  144          0141  1 !
;  145          0142  1 ! Routines Called:
;  146          0143  1 !
;  147          0144  1 !       BUCKET_EMPTY
;  148          0145  1 !       UPDATE_INDEX
;  149          0146  1 !       REMOVE_BUCKET
;  150          0147  1 !       SWAP_BUFFERS
;  151          0148  1 !       GET_NEXT_BUCKET
;  152          0149  1 !
;  153          0150  1 ! Side Effects:
;  154          0151  1 !       none
;  155          0152  1 !
;  156          0153  1 !--
;  157          0154  1
;  158          0155  2     BEGIN
;  159          0156  2
;  160          0157  2     DEFINE_CTX;
;  161          0158  2     DEFINE_BUCKET;
;  162          0159  2     DEFINE_KEY_DESC;
;  163          0160  2
;  164          0161  2     ! Loop untill the last bucket in chain if found.
;  165          0162  2     ! If this bucket is the last in the chain don't do it (it is to
;  166          0163  2     ! complaicated to reclaim this one bucket) instead go to the
;  167          0164  2     !
;  168          0165  3     WHILE ( NOT .BUCKET [ BKT$V_LASTBKT ] )
;  169          0166  2     DO
;  170          0167  3         BEGIN
```

```
;  171          0168  3
;  172          0169  3          ! If the bucket is empty the try to remove all traces of it
;  173          0170  3          !
;  174          0171  3          IF RECL$$BUCKET_EMPTY()
;  175          0172  3          THEN
;  176          0173  4              BEGIN
;  177          0174  4
;  178          0175  4              ! Remove the index record for this bucket
;  179          0176  4              !
;  180          0177  4              IF UPDATE_INDEX( .CTX [ CTX$L_CURRENT_VBN ] )
;  181          0178  4              THEN
;  182          0179  4
;  183          0180  4                  ! If the update was successful remove the bucket itself
;  184          0181  4                  !
;  185          0182  4                  REMOVE_BUCKET()
;  186          0183  4
;  187          0184  4              ELSE
;  188          0185  4
;  189          0186  4                  ! If index could not be update then swap the buffers in order
;  190          0187  4                  ! to save the previous bucket
;  191          0188  4                  !
;  192          0189  4                  RECL$$SWAP_BUFFERS()
;  193          0190  4
;  194          0191  4              END
;  195          0192  3          ELSE
;  196          0193  3
;  197          0194  3              ! If the bucket is not empty then swap the buffers in order to save
;  198          0195  3              ! the previous bucket
;  199          0196  3              !
;  200          0197  3              RECL$$SWAP_BUFFERS();
;  201          0198  3
;  202          0199  3              ! Get the next bucket
;  203          0200  3              !
;  204          0201  3              RECL$$GET_NEXT_BUCKET()
;  205          0202  3
;  206          0203  2              END;
;  207          0204  2
;  208          0205  2      RETURN RECL$_SUCCESS
;  209          0206  2
;  210          0207  1      END;
```

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

                                                .EXTRN  RECL$$GET_NEXT_BUCKET
                                                .EXTRN  RECL$$BUCKET_EMPTY
                                                .EXTRN  RECL$$GET_DOWN_POINTER
                                                .EXTRN  RECL$$CHECK_LAST
                                                .EXTRN  RECL$$COMPARE_POINTER
                                                .EXTRN  RECL$$SWING_POINTER
                                                .EXTRN  RECL$$REMOVE_INDEX_RECORD
                                                .EXTRN  RECL$$WRITE_BUCKET
                                                .EXTRN  CONV$$WRITE_AREA_DESC
                                                .EXTRN  CONV$$WRITE_KEY_DESC
                                                .EXTRN  CONV$AR_AREA_BLOCK
```

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

                                RECL$$SCAN_DATA_LEVEL::
                        1F  0D   A9  E8 00000           BLBS    13(BUCKET), 3$                  ; 0165
                            0000G 30 00004              BSBW    RECL$$BUCKET_EMPTY              ; 0171
                        11    50  E9 00007              BLBC    R0, 1$                          ;
                        08    AA  DD 0000A              PUSHL   8(CTX)                          ; 0177
                            0000V 30 0000D              BSBW    UPDATE_INDEX                    ;
                        5E      04  C0 00010            ADDL2   #4, SP                          ;
                        05      50  E9 00013            BLBC    R0, 1$                          ;
                            0000V 30 00016              BSBW    REMOVE_BUCKET                   ; 0182
                              03  11 00019              BRB     2$                              ;
                            0000V 30 0001B 1$:          BSBW    RECL$$SWAP_BUFFERS              ; 0197
                            0000G 30 0001E 2$:          BSBW    RECL$$GET_NEXT_BUCKET           ; 0201
                              DD  11 00021              BRB     RECL$$SCAN_DATA_LEVEL           ;
                        50      01  D0 00023 3$:        MOVL    #1, R0                          ; 0205
                                    05 00026            RSB                                     ; 0207

; Routine Size:  39 bytes,    Routine Base:  _CONV$RECL_S + 0000
```

```
;  212          0208  1 %SBTTL  'UPDATE_INDEX'
;  213          0209  1 ROUTINE UPDATE_INDEX ( VBN ) : RL$JSB_REG_9 =
;  214          0210  1 !++
;  215          0211  1 !
;  216          0212  1 ! Functional Description:
;  217          0213  1 !
;  218          0214  1 !       This routine updates the level above when a bucket on the lower level
;  219          0215  1 !       is deleted.  When called recursively, it updates the entire index.
;  220          0216  1 !
;  221          0217  1 ! Calling Sequence:
;  222          0218  1 !
;  223          0219  1 !       UPDATE_INDEX( VBN );
;  224          0220  1 !
;  225          0221  1 ! Input Parameters:
;  226          0222  1 !
;  227          0223  1 !       VBN     -       the VBN of the bucket being deleted on the lower level
;  228          0224  1 !
;  229          0225  1 ! Implicit Inputs:
;  230          0226  1 !
;  231          0227  1 !       BUCKET
;  232          0228  1 !       KEY_DESC
;  233          0229  1 !
;  234          0230  1 ! Output Parameters:
;  235          0231  1 !
;  236          0232  1 !       None.
;  237          0233  1 !
;  238          0234  1 ! Implicit Outputs:
;  239          0235  1 !
;  240          0236  1 !       None.
;  241          0237  1 !
;  242          0238  1 ! Routine Value:
;  243          0239  1 !
;  244          0240  1 !       SUCCESS or FAILURE
;  245          0241  1 !
;  246          0242  1 ! Routines Called:
;  247          0243  1 !
;  248          0244  1 !       GET_DOWN_POINTER
;  249          0245  1 !       COMPARE_POINTER
;  250          0246  1 !       SWING_POINTER
;  251          0247  1 !       REMOVE_INDEX_RECORD
;  252          0248  1 !       BUCKET_EMPTY
;  253          0249  1 !       UPDATE_INDEX
;  254          0250  1 !       REMOVE_BUCKET
;  255          0251  1 !       WRITE_BUCKET
;  256          0252  1 !       GET_NEXT_BUCKET
;  257          0253  1 !       SWAP_BUFFERS
;  258          0254  1 !
;  259          0255  1 ! Side Effects:
;  260          0256  1 !
;  261          0257  1 !       None.
;  262          0258  1 !
;  263          0259  1 !--
;  264          0260  1
;  265          0261  2     BEGIN
;  266          0262  2
;  267          0263  2     DEFINE_CTX;
;  268          0264  2     DEFINE_BUCKET;
```

```
;  269          0265  2     DEFINE_KEY_DESC;
;  270          0266  2     DEFINE_KEY_POINTER_GLOBAL;
;  271          0267  2
;  272          0268  2     LOCAL
;  273          0269  2         STATUS,
;  274          0270  2         NEXT_DATA_BUCKET;
;  275          0271  2
;  276          0272  2     ! Assume success
;  277          0273  2     !
;  278          0274  2     STATUS = RECL$_SUCCESS;
;  279          0275  2
;  280          0276  2     ! Return success if at level with root bucket
;  281          0277  2     !
;  282          0278  2     IF .BUCKET [ BKT$V_ROOTBKT ]
;  283          0279  2     THEN
;  284          0280  2         RETURN .STATUS;
;  285          0281  2
;  286          0282  2     ! Before we move up a level get the vbn of the next bucket (when this is
;  287          0283  2     ! the data level it will be important)
;  288          0284  2     !
;  289          0285  2     NEXT_DATA_BUCKET = .BUCKET [ BKT$L_NXTBKT ];
;  290          0286  2
;  291          0287  2     ! Point the context at the next higher level in the tree
;  292          0288  2     !
;  293          0289  2     CTX = .CTX + CTX$K_BLN;
;  294          0290  2
;  295          0291  2     ! Point to the new bucket
;  296          0292  2     !
;  297          0293  2     BUCKET = .CTX [ CTX$L_CURRENT_BUFFER ];
;  298          0294  2
;  299          0295  2     ! Save the position in the index so we can come back
;  300          0296  2     !
;  301          0297  2     CTX [ CTX$L_SAVE_VBN ] = .CTX [ CTX$L_PREVIOUS_VBN ];
;  302          0298  2
;  303          0299  2     ! Search all the buckets on the current level for a down pointer
;  304          0300  2     !
;  305          0301  2     DO
;  306          0302  3         BEGIN
;  307          0303  3
;  308          0304  3         ! Is down pointer in current bucket?
;  309          0305  3         !
;  310          0306  3         IF RECL$$GET_DOWN_POINTER( .VBN )
;  311          0307  3         THEN
;  312          0308  4             BEGIN
;  313          0309  4
;  314          0310  4             !++
;  315          0311  4             !
;  316          0312  4             ! Yes, we found the down pointer in the current bucket.
;  317          0313  4             ! Check to see if it is the last pointer in a bucket if so we
;  318          0314  4             ! can't reclaim it.
;  319          0315  4             ! If this is level 1 check to see if the next index pointer points
;  320          0316  4             ! to the next data bucket.  If it doesent we swing the current
;  321          0317  4             ! pointer to point to the next data bucket.  Otherwise we squish
;  322          0318  4             ! out the down pointer.  Then to see if squishing out the down
;  323          0319  4             ! pointer made the bucket reclaimable.  If it did, reclaim it after
;  324          0320  4             ! updating the index levels above this one.  If its not reclaimable
;  325          0321  4             ! just re-compress the index record following the deleted down
```

```
;  326          0322  4              ! pointer and write the bucket back.
;  327          0323  4              !
;  328          0324  4              !--
;  329          0325  4
;  330          0326  4              ! If this is the last index record in the bucket then don't reclaim it
;  331          0327  4              !
;  332          0328  4              IF RECL$$CHECK_LAST()
;  333          0329  4              THEN
;  334          0330  5                  BEGIN
;  335          0331  5                  STATUS = RECL$_FAILURE;
;  336          0332  5                  EXITLOOP
;  337          0333  4                  END;
;  338          0334  4
;  339          0335  4              IF .CTX [ CTX$B_LEVEL ] EQLU 1
;  340          0336  4              THEN
;  341          0337  4
;  342          0338  4                  ! Check to see if the next index pointer points to the
;  343          0339  4                  ! next data bucket
;  344          0340  4                  !
;  345          0341  4                  IF RECL$$COMPARE_POINTER( .NEXT_DATA_BUCKET )
;  346          0342  4                  THEN
;  347          0343  4
;  348          0344  4                      ! If it does, simply remove the current index record
;  349          0345  4                      !
;  350          0346  4                      RECL$$REMOVE_INDEX_RECORD()
;  351          0347  4
;  352          0348  4                  ELSE
;  353          0349  4
;  354          0350  4                      ! If it doesent, swing the current index record to point
;  355          0351  4                      ! to the next data bucket
;  356          0352  4                      !
;  357          0353  4                      RECL$$SWING_POINTER( .NEXT_DATA_BUCKET )
;  358          0354  4
;  359          0355  4              ELSE
;  360          0356  4
;  361          0357  4                  ! Squish out the index record in the current buffer
;  362          0358  4                  !
;  363          0359  4                  RECL$$REMOVE_INDEX_RECORD();
;  364          0360  4
;  365          0361  4              ! if this index bucket is empty then lets try to reclaim it!
;  366          0362  4              !
;  367          0363  4              IF RECL$$BUCKET_EMPTY()
;  368          0364  4              THEN
;  369          0365  5                  BEGIN
;  370          0366  5
;  371          0367  5                  ! If the index bucket is empty, try to update all the
;  372          0368  5                  ! index levels above.
;  373          0369  5                  ! If sucessful remove it.
;  374          0370  5                  !
;  375          0371  5                  IF STATUS = UPDATE_INDEX ( .CTX [ CTX$L_CURRENT_VBN ] )
;  376          0372  5                  THEN
;  377          0373  6                      BEGIN
;  378          0374  6
;  379          0375  6                      ! If the update was successful remove the bucket
;  380          0376  6                      !
;  381          0377  6                      REMOVE_BUCKET();
;  382          0378  6
```

```
;  383     0379  6                    ! Get the next bucket so we don't look at this one again
;  384     0380  6                    !
;  385     0381  6                    RECL$$GET_NEXT_BUCKET()
;  386     0382  6
;  387     0383  6                    END
;  388     0384  5                ELSE
;  389     0385  6                    BEGIN
;  390     0386  6
;  391     0387  6                    ! If the update failed then we must reread the buffer since
;  392     0388  6                    ! it was modified
;  393     0389  6                    !
;  394     0390  6                    CTX [ CTX$L_NEXT_VBN ] = .CTX [ CTX$L_SAVE_VBN ];
;  395     0391  6
;  396     0392  6                    ! Zero the current buffer vbn to force the read
;  397     0393  6                    !
;  398     0394  6                    CTX [ CTX$L_CURRENT_VBN ] = 0;
;  399     0395  6
;  400     0396  6                    ! Get the saved previous bucket
;  401     0397  6                    !
;  402     0398  6                    RECL$$GET_NEXT_BUCKET();
;  403     0399  6
;  404     0400  6                    EXITLOOP
;  405     0401  6
;  406     0402  6                    END
;  407     0403  6
;  408     0404  5                END
;  409     0405  4            ELSE
;  410     0406  5                BEGIN
;  411     0407  5
;  412     0408  5                ! bucket is not empty so just write the current
;  413     0409  5                ! buffer back, and return
;  414     0410  5                !
;  415     0411  5                RECL$$WRITE_BUCKET( CTX [ CTX$L_CURRENT_BUFFER ] );
;  416     0412  5
;  417     0413  5                EXITLOOP
;  418     0414  5
;  419     0415  5                END
;  420     0416  5
;  421     0417  4            END
;  422     0418  4
;  423     0419  3        ELSE
;  424     0420  3
;  425     0421  3            ! Down pointer is not in current buffer so read in the next bucket
;  426     0422  3            ! in the horizontal chain.
;  427     0423  3            !
;  428     0424  3            ! However, if this is already the last bucket in this level, we
;  429     0425  3            ! didn't find the down pointer, so return saying success, since
;  430     0426  3            ! if there's no down pointer we can certainly reclaim the bucket
;  431     0427  3            ! on the level below.
;  432     0428  3            !
;  433     0429  3            IF .BUCKET [ BKT$V_LASTBKT ]
;  434     0430  3            THEN
;  435     0431  4                BEGIN
;  436     0432  4
;  437     0433  4                ! If this bucket is the same as the save bucket then
;  438     0434  4                ! don't bother to reread it
;  439     0435  4                !
```

```
;  440          0436  4                IF .CTX [ CTX$L_CURRENT_VBN ] NEQU .CTX [ CTX$L_SAVE_VBN ]
;  441          0437  4                THEN
;  442          0438  5                    BEGIN
;  443          0439  5
;  444          0440  5                    ! Before we return go back to where we were
;  445          0441  5                    !
;  446          0442  5                    CTX [ CTX$L_NEXT_VBN ] = .CTX [ CTX$L_SAVE_VBN ];
;  447          0443  5
;  448          0444  5                    ! Get the saved previous bucket
;  449          0445  5                    !
;  450          0446  5                    RECL$$GET_NEXT_BUCKET()
;  451          0447  5
;  452          0448  4                    END;
;  453          0449  4
;  454          0450  4                ! Swap the suckers
;  455          0451  4                !
;  456          0452  4                RECL$$SWAP_BUFFERS();
;  457          0453  4
;  458          0454  4                ! Get the saved bucket
;  459          0455  4                !
;  460          0456  4                RECL$$GET_NEXT_BUCKET();
;  461          0457  4
;  462          0458  4                ! Return
;  463          0459  4                !
;  464          0460  4                EXITLOOP
;  465          0461  4
;  466          0462  4                END
;  467          0463  3            ELSE
;  468          0464  4                BEGIN
;  469          0465  4
;  470          0466  4                ! Its not the last bucket, so go read the next bucket
;  471          0467  4                !
;  472          0468  4                RECL$$SWAP_BUFFERS();
;  473          0469  4
;  474          0470  4                RECL$$GET_NEXT_BUCKET()
;  475          0471  4
;  476          0472  4                END
;  477          0473  4
;  478          0474  3            END
;  479          0475  3
;  480          0476  2        UNTIL RECL$_FOREVER;
;  481          0477  2
;  482          0478  2        ! We exit the loop on sucess so return the context back to where it
;  483          0479  2        ! was when we were called
;  484          0480  2        !
;  485          0481  2        CTX = .CTX - CTX$K_BLN;
;  486          0482  2
;  487          0483  2        BUCKET = .CTX [ CTX$L_CURRENT_BUFFER ];
;  488          0484  2
;  489          0485  2        RETURN .STATUS
;  490          0486  2
;  491          0487  1        END;
```

```
                                010C   8F   BB 00000 UPDATE_INDEX:
                                                     PUSHR   #^M<R2,R3,R8>                   ; 0209
                                  53      01   D0 00004         MOVL    #1, STATUS              ; 0274
                           03     0D   A9      01   E1 00007         BBC     #1, 13(BUCKET), 1$      ; 0278
                                       009F   31 0000C         BRW     14$
                                  52   08   A9   D0 0000F 1$:      MOVL    8(BUCKET), NEXT_DATA_BUCKET ; 0285
                                  5A   5C   AA   9E 00013         MOVAB   92(R10), CTX            ; 0289
                                  59   04   AA   D0 00017         MOVL    4(CTX), BUCKET          ; 0293
                                54   AA   44   AA   D0 0001B         MOVL    68(CTX), 84(CTX)        ; 0297
                                       10   AE   DD 00020 2$:      PUSHL   VBN                     ; 0306
                                     0000G  30 00023         BSBW    RECL$$GET_DOWN_POINTER
                                  5E      04   C0 00026         ADDL2   #4, SP
                                  56      50   E9 00029         BLBC    R0, 8$
                                     0000G  30 0002C         BSBW    RECL$$CHECK_LAST        ; 0328
                                  04      50   E9 0002F         BLBC    R0, 3$
                                          53   D4 00032         CLRL    STATUS                  ; 0331
                                          70   11 00034         BRB     13$                     ; 0330
                                  01   02   AA   91 00036 3$:      CMPB    2(CTX), #1              ; 0335
                                          15   12 0003A         BNEQ    4$
                                          52   DD 0003C         PUSHL   NEXT_DATA_BUCKET        ; 0341
                                     0000G  30 0003E         BSBW    RECL$$COMPARE_POINTER
                                  5E      04   C0 00041         ADDL2   #4, SP
                                  0A      50   E8 00044         BLBS    R0, 4$
                                          52   DD 00047         PUSHL   NEXT_DATA_BUCKET        ; 0353
                                     0000G  30 00049         BSBW    RECL$$SWING_POINTER
                                  5E      04   C0 0004C         ADDL2   #4, SP
                                          03   11 0004F         BRB     5$                      ; 0341
                                     0000G  30 00051 4$:      BSBW    RECL$$REMOVE_INDEX_RECORD ; 0359
                                     0000G  30 00054 5$:      BSBW    RECL$$BUCKET_EMPTY      ; 0363
                                  1D      50   E9 00057         BLBC    R0, 7$
                                       08   AA   DD 0005A         PUSHL   8(CTX)                  ; 0371
                                          A1   10 0005D         BSBB    UPDATE_INDEX
                                  5E      04   C0 0005F         ADDL2   #4, SP
                                  53      50   D0 00062         MOVL    R0, STATUS
                                  05      53   E9 00065         BLBC    STATUS, 6$
                                     0000V  30 00068         BSBW    REMOVE_BUCKET           ; 0377
                                          33   11 0006B         BRB     12$                     ; 0381
                           50     AA   54   AA   D0 0006D 6$:      MOVL    84(CTX), 80(CTX)        ; 0390
                                       08   AA   D4 00072         CLRL    8(CTX)                  ; 0394
                                          21   11 00075         BRB     10$                     ; 0398
                                       04   AA   9F 00077 7$:      PUSHAB  4(CTX)                  ; 0411
                                     0000G  30 0007A         BSBW    RECL$$WRITE_BUCKET
                                  5E      04   C0 0007D         ADDL2   #4, SP
                                          24   11 00080         BRB     13$                     ; 0406
                                  17      0D   A9   E9 00082 8$:      BLBC    13(BUCKET), 11$         ; 0429
                                54   AA      08   AA   D1 00086         CMPL    8(CTX), 84(CTX)         ; 0436
                                          08   13 0008B         BEQL    9$
                           50     AA   54   AA   D0 0008D         MOVL    84(CTX), 80(CTX)        ; 0442
                                     0000G  30 00092         BSBW    RECL$$GET_NEXT_BUCKET   ; 0446
                                     0000V  30 00095 9$:      BSBW    RECL$$SWAP_BUFFERS      ; 0452
                                     0000G  30 00098 10$:     BSBW    RECL$$GET_NEXT_BUCKET   ; 0456
                                          09   11 0009B         BRB     13$                     ; 0431
                                     0000V  30 0009D 11$:     BSBW    RECL$$SWAP_BUFFERS      ; 0468
                                     0000G  30 000A0 12$:     BSBW    RECL$$GET_NEXT_BUCKET   ; 0470
                                     FF7A   31 000A3         BRW     2$                      ; 0302
                                  5A   A4   AA   9E 000A6 13$:     MOVAB   -92(R10), CTX           ; 0481
                                  59   04   AA   D0 000AA         MOVL    4(CTX), BUCKET          ; 0483
```

```
                    50              53  D0 000AE 14$:   MOVL    STATUS, R0                      ; 0485
                            010C    8F  BA 000B1        POPR    #^M<R2,R3,R8>                   ; 0487
                                        05 000B5        RSB                                     ;

; Routine Size:  182 bytes,    Routine Base:  _CONV$RECL_S + 0027

;  492          0488  1
```

```
;  494          0489  1 %SBTTL  'REMOVE_BUCKET'
;  495          0490  1 ROUTINE REMOVE_BUCKET : RL$JSB_REG_9 NOVALUE =
;  496          0491  1 !++
;  497          0492  1 !
;  498          0493  1 ! Functional Description:
;  499          0494  1 !
;  500          0495  1 !       This routine takes the steps required to remove a bucket from the
;  501          0496  1 !       horizontal chain, write it to the AVAIL list, and update the key
;  502          0497  1 !       descriptor if necessary.
;  503          0498  1 !
;  504          0499  1 ! Calling Sequence:
;  505          0500  1 !
;  506          0501  1 !       REMOVE_BUCKET();
;  507          0502  1 !
;  508          0503  1 ! Input Parameters:
;  509          0504  1 !       none
;  510          0505  1 !
;  511          0506  1 ! Implicit Inputs:
;  512          0507  1 !
;  513          0508  1 !       CTX to point to current bucket, etc...
;  514          0509  1 !
;  515          0510  1 ! Output Parameters:
;  516          0511  1 !       none
;  517          0512  1 !
;  518          0513  1 ! Implicit Outputs:
;  519          0514  1 !
;  520          0515  1 !       The bucket is removed and written to the AVAIL list.  All pointers
;  521          0516  1 !       are updated.
;  522          0517  1 !
;  523          0518  1 ! Routine Value:
;  524          0519  1 !       none
;  525          0520  1 !
;  526          0521  1 ! Routines Called:
;  527          0522  1 !
;  528          0523  1 !       CONV$$WRITE_KEY_DESC
;  529          0524  1 !       RECL$$WRITE_BUCKET
;  530          0525  1 !       ZERO_BUCKET
;  531          0526  1 !       CONV$$WRITE_AREA_DESC
;  532          0527  1 !
;  533          0528  1 ! Side Effects:
;  534          0529  1 !
;  535          0530  1 !       RECL$GL_DATA_COUNT is incremented if we reclaim a data bucket.
;  536          0531  1 !       RECL$GL_INDEX_COUNT is incremented if we reclaim an index bucket.
;  537          0532  1 !
;  538          0533  1 !--
;  539          0534  1
;  540          0535  2     BEGIN
;  541          0536  2
;  542          0537  2     DEFINE_CTX;
;  543          0538  2     DEFINE_BUCKET;
;  544          0539  2     DEFINE_KEY_DESC;
;  545          0540  2
;  546          0541  2     EXTERNAL
;  547          0542  2         RECL$GL_DATA_COUNT,
;  548          0543  2         RECL$GL_INDEX_COUNT;
;   .9          0544  2
;  >50          0545  2     LOCAL
```

```
;  551          0546  2          AREA_DESC : REF BLOCK [ ,BYTE ];
;  552          0547  2
;  553          0548  2      ! The removal of a bucket is done in three steps, the order of which
;  554          0549  2      ! is of the utmost importance to the reliability of the utility. It
;  555          0550  2      ! is assumed that the index record for this bucket has been removed.
;  556          0551  2      !
;  557          0552  2      ! Step I
;  558          0553  2      !
;  559          0554  2      ! Update the previous bucket pointer to point to the next one in the chain
;  560          0555  2      !
;  561          0556  3      BEGIN
;  562          0557  3
;  563          0558  3      LOCAL PREVIOUS_BUCKET : REF BLOCK [ ,BYTE ];
;  564          0559  3
;  565          0560  3      PREVIOUS_BUCKET = .CTX [ CTX$L_PREVIOUS_BUFFER ];
;  566          0561  3
;  567          0562  3      ! Update the previous bucket in the chain
;  568          0563  3      !
;  569          0564  3      PREVIOUS_BUCKET [ BKT$L_NXTBKT ] = .CTX [ CTX$L_NEXT_VBN ];
;  570          0565  3
;  571          0566  3      RECL$$WRITE_BUCKET( CTX [ CTX$L_PREVIOUS_BUFFER ] )
;  572          0567  3
;  573          0568  2      END;
;  574          0569  2
;  575          0570  2      ! Step Ia
;  576          0571  2      !
;  577          0572  2      ! In the case that this is the first bucket in a chain then either do
;  578          0573  2      ! nothing or update the key descriptor, depending on the level.
;  579          0574  2      !
;  580          0575  2      ! Is this the first bucket in the chain
;  581          0576  2      !
;  582          0577  2      IF .CTX [ CTX$L_CURRENT_VBN ] EQLU .CTX [ CTX$L_FIRST_VBN ]
;  583          0578  2      THEN
;  584          0579  3          BEGIN
;  585          0580  3
;  586          0581  3          ! If this is the data level bucket then update the key descriptor
;  587          0582  3          ! else continue
;  588          0583  3          !
;  589          0584  3          IF .BUCKET [ BKT$B_LEVEL ] EQLU 0
;  590          0585  3          THEN
;  591          0586  4              BEGIN
;  592          0587  4
;  593          0588  4              KEY_DESC [ KEY$L_LDVBN ] = .CTX [ CTX$L_NEXT_VBN ];
;  594          0589  4
;  595          0590  4              CONV$$WRITE_KEY_DESC()
;  596          0591  4
;  597          0592  3              END;
;  598          0593  3
;  599          0594  3          ! The next vbn will now be the first in the chain
;  600          0595  3          !
;  601          0596  3          CTX [ CTX$L_FIRST_VBN ] = .CTX [ CTX$L_NEXT_VBN ]
;  602          0597  3
;  603          0598  2          END;
;  604          0599  2
;  605          0600  2      ! Step II
;  606          0601  2      !
;  607          0602  2      ! Update the current bucket to point to the first bucket in the area
```

```
;  608          0603  2      ! available list
;  609          0604  2      !
;  610          0605  2      ! To update the bucket we must use the area descriptor
;  611          0606  2      !
;  612          0607  2      AREA_DESC = .CONV$AR_AREA_BLOCK + ( .CTX [ CTX$B_AREA ] * AREA$K_BLN );
;  613          0608  2
;  614          0609  2      ! Point the bucket to the first avail. bucket
;  615          0610  2      !
;  616          0611  2      BUCKET [ BKT$L_NXTBKT ] = .AREA_DESC [ AREA$L_AVAIL ];
;  617          0612  2
;  618          0613  2      ! If first bucket on free list set the last bucket bit
;  619          0614  2      !
;  620          0615  2      IF .BUCKET [ BKT$L_NXTBKT ] EQLU 0
;  621          0616  2      THEN
;  622          0617  2          BUCKET [ BKT$V_LASTBKT ] = _SET;
;  623          0618  2
;  624          0619  2      ! Zero the data portion of the bucket
;  625          0620  2      !
;  626          0621  2      ZERO_BUCKET();
;  627          0622  2
;  628          0623  2      ! Write the bucket into the file
;  629          0624  2      !
;  630          0625  2      RECL$$WRITE_BUCKET( CTX [ CTX$L_CURRENT_BUFFER ] );
;  631          0626  2
;  632          0627  2      ! Count the reclaimed bucket.
;  633          0628  2      !
;  634          0629  2      IF .BUCKET[ BKT$B_LEVEL ] EQLU 0
;  635          0630  2      THEN
;  636          0631  2
;  637          0632  2          ! Its a data bucket we're reclaiming.
;  638          0633  2          !
;  639          0634  2          RECL$GL_DATA_COUNT = .RECL$GL_DATA_COUNT + 1
;  640          0635  2      ELSE
;  641          0636  2
;  642          0637  2          ! Its an index bucket we're reclaiming.
;  643          0638  2          !
;  644          0639  2          RECL$GL_INDEX_COUNT = .RECL$GL_INDEX_COUNT + 1;
;  645          0640  2
;  646          0641  2
;  647          0642  2      ! Step III
;  648          0643  2      !
;  649          0644  2      ! Update the area descriptor with the new bucket at the head of the
;  650          0645  2      ! availiable list
;  651          0646  2      !
;  652          0647  2      AREA_DESC [ AREA$L_AVAIL ] = .CTX [ CTX$L_CURRENT_VBN ];
;  653          0648  2
;  654          0649  2      CONV$$WRITE_AREA_DESC( .CTX [ CTX$B_AREA ] );
;  655          0650  2
;  656          0651  2      RETURN
;  657          0652  2
;  658          0653  1      END;


                                                                .EXTRN  RECL$GL_DATA_COUNT
                                                                .EXTRN  RECL$GL_INDEX_COUNT
```

```
                                                52  DD 00000 REMOVE_BUCKET:
                                                               PUSHL   R2                                              ; 0490
                                             50     40   AA  D0 00002                MOVL    64(CTX), PREVIOUS_BUCKET                ; 0560
                                          08 A0     50   AA  D0 00006                MOVL    80(CTX), 8(PREVIOUS_BUCKET)             ; 0564
                                             40   AA  9F 0000B                       PUSHAB  64(CTX)                                 ; 0566
                                                0000G 30 0000E                       BSBW    RECL$$WRITE_BUCKET
                                             5E     04   C0 00011                    ADDL2   #4, SP
                                          24 AA     08   AA  D1 00014                CMPL    8(CTX), 36(CTX)                         ; 0577
                                                    12   12 00019                    BNEQ    2$
                                                0C   A9  95 0001B                    TSTB    12(BUCKET)                              ; 0584
                                                    08   12 0001E                    BNEQ    1$
                                          54 AB     50   AA  D0 00020                MOVL    80(CTX), 84(KEY_DESC)                   ; 0588
                                                0000G 30 00025                       BSBW    CONV$$WRITE_KEY_DESC                    ; 0590
                                          24 AA     50   AA  D0 00028 1$:           MOVL    80(CTX), 36(CTX)                        ; 0596
                                             50     01   AA  9A 0002D 2$:           MOVZBL  1(CTX), R0                              ; 0607
                                50           50     06   78 00031                    ASHL    #6, R0, R0
                                52           50 0000G  CF  C1 00035                ADDL3   CONV$AR_AREA_BLOCK, R0, AREA_DESC
                                          08 A9     08   A2  D0 0003B                MOVL    8(AREA_DESC), 8(BUCKET)                 ; 0611
                                                    04   12 00040                    BNEQ    3$                                      ; 0615
                                          0D A9           01  88 00042             BISB2   #1, 13(BUCKET)                          ; 0617
                                                0000V 30 00046 3$:                   BSBW    ZERO_BUCKET                             ; 0621
                                                04   AA  9F 00049                   PUSHAB  4(CTX)                                  ; 0625
                                                0000G 30 0004C                       BSBW    RECL$$WRITE_BUCKET
                                             5E     04   C0 0004F                    ADDL2   #4, SP
                                                0C   A9  95 00052                    TSTB    12(BUCKET)                              ; 0629
                                                    06   12 00055                    BNEQ    4$
                                                0000G CF  D6 00057                   INCL    RECL$GL_DATA_COUNT                      ; 0634
                                                    04   11 0005B                    BRB     5$
                                                0000G CF  D6 0005D 4$:               INCL    RECL$GL_INDEX_COUNT                     ; 0639
                                          08 A2     08   AA  D0 00061 5$:           MOVL    8(CTX), 8(AREA_DESC)                    ; 0647
                                             51     01   AA  9A 00066                MOVZBL  1(CTX), R1                              ; 0649
                                                      0000G 30 0006A                 BSBW    CONV$$WRITE_AREA_DESC
                                                04   BA 0006D                       POPR    #^M<R2>                                 ; 0653
                                                           05 0006F                  RSB                                             ;

; Routine Size:  112 bytes,    Routine Base:  _CONV$RECL_S + 00DD

;  659          0654  1
```

```
;  661          0655  1 %SBTTL  'ZERO_BUCKET'
;  662          0656  1 ROUTINE ZERO_BUCKET : RL$JSB_REG_9 NOVALUE =
;  663          0657  1 !++
;  664          0658  1 !
;  665          0659  1 ! Functional Description:
;  666          0660  1 !
;  667          0661  1 !       Zeros out the data portion of a index bucket
;  668          0662  1 !
;  669          0663  1 ! Calling Sequence:
;  670          0664  1 !
;  671          0665  1 !       ZERO_BUCKET()
;  672          0666  1 !
;  673          0667  1 ! Input Parameters:
;  674          0668  1 !       none
;  675          0669  1 !
;  676          0670  1 ! Implicit Inputs:
;  677          0671  1 !       none
;  678          0672  1 !
;  679          0673  1 ! Output Parameters:
;  680          0674  1 !       none
;  681          0675  1 !
;  682          0676  1 ! Implicit Outputs:
;  683          0677  1 !       none
;  684          0678  1 !
;  685          0679  1 ! Routine Value:
;  686          0680  1 !       none
;  687          0681  1 !
;  688          0682  1 ! Routines Called:
;  689          0683  1 !       none
;  690          0684  1 !
;  691          0685  1 ! Side Effects:
;  692          0686  1 !       none
;  693          0687  1 !
;  694          0688  1 !--
;  695          0689  1
;  696          0690  2     BEGIN
;  697          0691  2
;  698          0692  2     DEFINE_CTX;
;  699          0693  2     DEFINE_BUCKET;
;  700          0694  2     DEFINE_KEY_DESC;
;  701          0695  2
;  702          0696  2     CH$FILL( 0,                                                 ! Fill with 0's
;  703          0697  2              .CTX [ CTX$W_BUCKET_SIZE ] - BKT$K_OVERHDSZ - 1,   ! This much
;  704          0698  2              .CTX [ CTX$L_CURRENT_BUFFER ] + BKT$K_OVERHDSZ );  ! Starting here
;  705          0699  2
;  706          0700  2     RETURN
;  707          0701  2
;  708          0702  1     END;


                                3C  BB 00000 ZERO_BUCKET:
                                             PUSHR   #^M<R2,R3,R4,R5>                ; 0656
                     51     58  AA  3C 00002         MOVZWL  88(CTX), R1             ; 0697
                     51         0F  C2 00006         SUBL2   #15, R1                 ;
```

```
                                50      04    AA  D0 00009         MOVL    4(CTX), R0                                      ; 0698
          51            00      6E            00  2C 0000D         MOVC5   #0, (SP), #0, R1, 14(R0)                        ;
                                          0E    A0     00012                                                            ;
                                                3C  BA 00014         POPR    #^M<R2,R3,R4,R5>                                ; 0702
                                                    05 00016         RSB                                                     ;

; Routine Size:  23 bytes,    Routine Base:  _CONV$RECL_S + 014D

;  709          0703  1
```

```
;  711          0704  1 %SBTTL  'SWAP_BUFFERS'
;  712          0705  1 GLOBAL ROUTINE  RECL$$SWAP_BUFFERS : RL$JSB_REG_9 NOVALUE =
;  713          0706  1 !++
;  714          0707  1 !
;  715          0708  1 ! Functional Description:
;  716          0709  1 !
;  717          0710  1 ! Calling Sequence:
;  718          0711  1 !
;  719          0712  1 ! Input Parameters:
;  720          0713  1 !       none
;  721          0714  1 !
;  722          0715  1 ! Implicit Inputs:
;  723          0716  1 !       none
;  724          0717  1 !
;  725          0718  1 ! Output Parameters:
;  726          0719  1 !       none
;  727          0720  1 !
;  728          0721  1 ! Implicit Outputs:
;  729          0722  1 !       none
;  730          0723  1 !
;  731          0724  1 ! Routine Value:
;  732          0725  1 !       none
;  733          0726  1 !
;  734          0727  1 ! Routines Called:
;  735          0728  1 !       none
;  736          0729  1 !
;  737          0730  1 ! Side Effects:
;  738          0731  1 !       none
;  739          0732  1 !
;  740          0733  1 !--
;  741          0734  1
;  742          0735  2     BEGIN
;  743          0736  2
;  744          0737  2     DEFINE_CTX;
;  745          0738  2     DEFINE_BUCKET;
;  746          0739  2     DEFINE_KEY_DESC;
;  747          0740  2
;  748          0741  2     LOCAL
;  749          0742  2         TEMP_BUF,
;  750          0743  2         TEMP_VBN;
;  751          0744  2
;  752          0745  2     ! Swap the current buffer with the previous buffer and change bucket
;  753          0746  2     !
;  754          0747  2     TEMP_BUF = .CTX [ CTX$L_PREVIOUS_BUFFER ];
;  755          0748  2     TEMP_VBN = .CTX [ CTX$L_PREVIOUS_VBN ];
;  756          0749  2
;  757          0750  2     CTX [ CTX$L_PREVIOUS_BUFFER ] = .CTX [ CTX$L_CURRENT_BUFFER ];
;  758          0751  2     CTX [ CTX$L_PREVIOUS_VBN ]    = .CTX [ CTX$L_CURRENT_VBN ];
;  759          0752  2
;  760          0753  2     CTX [ CTX$L_CURRENT_BUFFER ] = .TEMP_BUF;
;  761          0754  2     CTX [ CTX$L_CURRENT_VBN ]    = .TEMP_VBN;
;  762          0755  2
;  763          0756  2     BUCKET = .TEMP_BUF;
;  764          0757  2
;  765          0758  2     RETURN
;  766          0759  2
;  767          0760  1     END;
```

```
                                50      40   AA  7D 00000 RECL$$SWAP_BUFFERS::
                                                          MOVQ    64(CTX), TEMP_BUF                ; 0747
                            40  AA      04   AA  7D 00004          MOVQ    4(CTX), 64(CTX)          ; 0750
                            04  AA           50  7D 00009          MOVQ    TEMP_BUF, 4(CTX)         ; 0753
                                59           50  D0 00000D         MOVL    TEMP_BUF, BUCKET         ; 0756
                                                 05 00010          RSB                              ; 0760

; Routine Size:  17 bytes,    Routine Base:  _CONV$RECL_S + 0164

;  768          0761  1
;  769          0762  0 END     ELUDOM



;                                   PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  _CONV$RECL_S                    373  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         8         0        1000        00:01.8
;  _$255$DUA28:[CONV.SRC]CONVERT.L32;1      165        23        13          17        00:00.2



;                                   COMMAND QUALIFIERS

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

; Size:          373 code + 0 data bytes
; Run Time:        00:12.9
; Elapsed Time:    00:36.2
; Lines/CPU Min:    3538
; Lexemes/CPU-Min: 13500
; Memory Used:  103 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