```
   CCCCCCCCCCCC     000000000     NNN         NNN  VVV         VVV
   CCCCCCCCCCCC     000000000     NNN         NNN  VVV         VVV
   CCCCCCCCCCCC     000000000     NNN         NNN  VVV         VVV
CCC              000         000  NNN         NNN  VVV         VVV
CCC              000         000  NNN         NNN  VVV         VVV
CCC              000         000  NNN         NNN  VVV         VVV
CCC              000         000  NNNNNN      NNN  VVV         VVV
CCC              000         000  NNNNNN      NNN  VVV         VVV
CCC              000         000  NNNNNN      NNN  VVV         VVV
CCC              000         000  NNN   NNN   NNN  VVV         VVV
CCC              000         000  NNN   NNN   NNN  VVV         VVV
CCC              000         000  NNN   NNN   NNN  VVV         VVV
CCC              000         000  NNN      NNNNNN  VVV         VVV
CCC              000         000  NNN      NNNNNN  VVV         VVV
CCC              000         000  NNN      NNNNNN  VVV         VVV
CCC              000         000  NNN         NNN     VVV   VVV
CCC              000         000  NNN         NNN     VVV   VVV
CCC              000         000  NNN         NNN     VVV   VVV
   CCCCCCCCCCCC     000000000     NNN         NNN        VVV
   CCCCCCCCCCCC     000000000     NNN         NNN        VVV
   CCCCCCCCCCCC     000000000     NNN         NNN        VVV
```

```
**FILE**ID**CONVFSTIO                                  D  9




  CCCCCCCC    000000    NN      NN  VV      VV  FFFFFFFFFF    SSSSSSSS  TTTTTTTTTT    IIIIII      000000
  CCCCCCCC    000000    NN      NN  VV      VV  FFFFFFFFFF    SSSSSSSS  TTTTTTTTTT    IIIIII      000000
CC          00      00  NN      NN  VV      VV  FF          SS              TT          II      00      00
CC          00      00  NN      NN  VV      VV  FF          SS              TT          II      00      00
CC          00      00  NNNN    NN  VV      VV  FF          SS              TT          II      00      00
CC          00      00  NNNN    NN  VV      VV  FF          SS              TT          II      00      00
CC          00      00  NN  NN  NN  VV      VV  FFFFFFFF      SSSSSS        TT          II      00      00
CC          00      00  NN  NN  NN  VV      VV  FFFFFFFF      SSSSSS        TT          II      00      00
CC          00      00  NN    NNNN  VV      VV  FF                  SS      TT          II      00      00
CC          00      00  NN    NNNN  VV      VV  FF                  SS      TT          II      00      00
CC          00      00  NN      NN    VV  VV    FF                  SS      TT          II      00      00    ....
CC          00      00  NN      NN    VV  VV    FF                  SS      TT          II      00      00    ....
  CCCCCCCC    000000    NN      NN      VV      FF          SSSSSSSS        TT        IIIIII      000000      ....
  CCCCCCCC    000000    NN      NN      VV      FF          SSSSSSSS        TT        IIIIII      000000      ....

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
```

```
%TITLE  'VAX-11 CONVERT'
MODULE  CONV$FSTIO      ( IDENT='V04-000',
                        OPTLEVEL=3
                        ) =

BEGIN

!****************************************************************************
!*                                                                          *
!*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
!*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
!*  ALL RIGHTS RESERVED.                                                    *
!*                                                                          *
!*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
!*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
!*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
!*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
!*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
!*  TRANSFERRED.                                                            *
!*                                                                          *
!*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
!*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
!*  CORPORATION.                                                            *
!*                                                                          *
!*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
!*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
!*                                                                          *
!*                                                                          *
!****************************************************************************
```

```
0030  1 !++
0031  1 !
0032  1 ! Facility:      VAX-11 CONVERT
0033  1 !
0034  1 ! Abstract:      CONVERT fast load I/O and bucket routines
0035  1 !
0036  1 ! Contents:
0037  1 !                WRITE_BUCKET
0038  1 !                SET_NXTBKT
0039  1 !                SWAP_BUCKET
0040  1 !                GET_BUCKET
0041  1 !                INIT_BUCKET
0042  1 !                EXTEND_AREA
0043  1 !                CONVERT_VBN_ID
0044  1 !
0045  1 ! Environment:
0046  1 !
0047  1 !                VAX/VMS Operating System
0048  1 !
0049  1 !--
0050  1
0051  1 !
0052  1 ! Author:        Keith B Thompson        Creation date:          August-1980
0053  1 !
0054  1 !
0055  1 ! Modified by:
0056  1 !
0057  1 !       V03-005 RAS0331         Ron Schaefer            31-Jul-1984
0058  1 !               Accumulate total area allocation.
0059  1 !
0060  1 !       V03-004 KBT0478         Keith B. Thompson       29-Jan-1983
0061  1 !               Make extend_area global
0062  1 !
0063  1 !       V03-003 KBT0385         Keith B. Thompson       27-Oct-1982
0064  1 !               Make changes to support prologue 3 sidrs
0065  1 !
0066  1 !       V03-002 KBT0350         Keith B. Thompson       4-Oct-1982
0067  1 !               Use new linkage definitions
0068  1 !
0069  1 !       V03-001 KBT0024         Keith Thompson          25-Mar-1982
0070  1 !               Change the linkage to get_bucket and remove some useless code
0071  1 !
0072  1 !****
```

```
1
1 PSECT
1         OWN     = _CONV$FAST_D  (PIC),
1         GLOBAL  = _CONV$FAST_D  (PIC),
1         PLIT    = _CONV$PLIT    (SHARE,PIC),
1         CODE    = _CONV$FAST_S  (SHARE,PIC);
1
1 LIBRARY 'SYS$LIBRARY:LIB.L32';
1 LIBRARY 'SRC$:CONVERT';
1
1 DEFINE_ERROR_CODES;
1
1 EXTERNAL ROUTINE
1         CONV$$GET_TEMP_VM          : CL$GET_TEMP_VM,
1         CONV$$RMS_ERROR            : NOVALUE,
1         CONV$$RMS_READ_ERROR       : NOVALUE;
1
1 FORWARD ROUTINE
1         SET_NXTBKT                 : CL$JSB_REG_9  NOVALUE,
1         SWAP_BUCKET                : CL$JSB_REG_9  NOVALUE,
1         CONV$$INIT_BUCKET          : CL$JSB_REG_9  NOVALUE,
1         CONV$$EXTEND_AREA          : CL$EXTEND_AREA        NOVALUE;
1
1 EXTERNAL
1         CONV$GL_RFA_BUFFER,
1
1         CONV$AB_OUT_FAB            : $FAB_DECL,
1         CONV$AB_OUT_RAB            : $RAB_DECL,
1
1         CONV$GL_CTX_BLOCK,
1
1         CONV$GL_EOF_VBN,
1         CONV$GB_PROC_V3            : BYTE,
1         CONV$AR_AREA_BLOCK         : REF BLOCKVECTOR [ ,AREA$C_BLN,BYTE ],
1         CONV$GW_AREA_SIZE          : WORD;
1
1 MACRO
1
1 !       These macros make the code look a little better
1 !
1         BKT$W_VBNFS     = .CONV$GW_VBN_FS_PTR,0,16,0%,  ! VBN Freespace Pointer in index level
1         BKT$W_VBNFSO    = .CONV$GW_VBN_FS_PTRO,0,16,0%, ! VBN Freespace Pointer at the data level
1         BKT$L_LCBPTR    = .CONV$GW_LCB_PTR,0,32,0%;     ! Last Contuation Bucket Pointer
1
1 EXTERNAL
1         CONV$GW_VBN_FS_PTR         : WORD,
1         CONV$GW_VBN_FS_PTRO          WORD,
1         CONV$GW_LCB_PTR            : WORD;
1
1 GLOBAL  CONV$GL_CONT_VBN           : LONG;
1
```

```
1 %SBTTL  'WRITE_BUCKET'
1 GLOBAL ROUTINE  CONV$$WRITE_BUCKET : CL$JSB_REG_9 NOVALUE =
1 !++
1 !
1 ! Functional Description:
1 !
1 !       Writes the current bucket into the output file
1 !
1 ! Calling Sequence:
1 !
1 !       CONV$$WRITE_BUCKET()
1 !
1 ! Input Parameters:
1 !       none
1 !
1 ! Implicit Inputs:
1 !       none
1 !
1 ! Output Parameters:
1 !       none
1 !
1 ! Implicit Outputs:
1 !       none
1 !
1 ! Routine Value:
1 !       none
1 !
1 ! Routines Called:
1 !
1 !       SET_NXTBKT
1 !       SWAP_BUCKET
1 !       $WRITE
1 !       CONV$$RMS_ERROR - By RMS as an AST
1 !
1 ! Side Effects:
1 !       none
1 !
1 !--
1
2     BEGIN
2
2     DEFINE_CTX;
2     DEFINE_BUCKET;
2     DEFINE_KEY_DESC;
2
2     ! Set the next bucket pointer to the bucket
2     !
2     SET_NXTBKT();
2
2     ! Point RMS to the bucket.  NOTE: This is where OUT_RAB is changed
2     !
2     CONV$AB_OUT_RAB [ RAB$L_RBF ] = .BUCKET;
2     CONV$AB_OUT_RAB [ RAB$W_RSZ ] = .CTX [ CTX$L_SIZ ];
2     CONV$AB_OUT_RAB [ RAB$L_BKT ] = .CTX [ CTX$L_CURRENT_VBN ];
2
2     ! If we are doing double buffering on this bucket
2     ! then swap pointers to the buckets and set for asyc. operation
```

```
;  184          0181   2        !
;  185          0182   2        IF .CTX [ CTX$V_DBF ]
;  186          0183   2        THEN
;  187          0184   3            BEGIN
;  188          0185   3
;  189          0186   3            ! Switch the buffer
;  190          0187   3            !
;  191          0188   3            SWAP_BUCKET();
;  192          0189   3
;  193          0190   3            ! Set the asynchronous bit
;  194          0191   3            !
;  195          0192   3            CONV$AB_OUT_RAB [ RAB$V_ASY ] = _SET
;  196          0193   3
;  197          0194   3            END
;  198          0195   2        ELSE
;  199          0196   2
;  200          0197   2            ! If not then we need a syncrous call
;  201          0198   2            !
;  202          0199   2            CONV$AB_OUT_RAB [ RAB$V_ASY ] = _CLEAR;
;  203          0200   2
;  204          0201   2        ! Wait on the last IO if necessary
;  205          0202   2        !
;  206          0203   2        $WAIT ( RAB=CONV$AB_OUT_RAB );
;  207          0204   2
;  208          0205   2        ! Write The Bucket
;  209          0206   2        !
;  210          0207   2        $WRITE ( RAB=CONV$AB_OUT_RAB,ERR=CONV$$RMS_ERROR );
;  211          0208   2
;  212          0209   2        RETURN
;  213          0210   2
;  214          0211   1        END;


                                                        .TITLE  CONV$FSTIO VAX-11 CONVERT
                                                        .IDENT  \V04-000\

                                                        .PSECT  _CONV$FAST_D,NOEXE,  PIC,2

                                          00000 CONV$GL_CONT_VBN::
                                                        .BLKB   4

                                                        .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  CONV$$GET_TEMP_VM
                                                        .EXTRN  CONV$$RMS_ERROR
                                                        .EXTRN  CONV$$RMS_READ_ERROR
                                                        .EXTRN  CONV$GL_RFA_BUFFER
                                                        .EXTRN  CONV$AB_OUT_FAB
                                                        .EXTRN  CONV$AB_OUT_RAB
                                                        .EXTRN  CONV$GL_CTX_BLOCK
                                                        .EXTRN  CONV$GL_EOF_VBN
                                                        .EXTRN  CONV$GB_PROL_V3
                                                        .EXTRN  CONV$AR_AREA_BLOCK
                                                        .EXTRN  CONV$GW_AREA_SIZE
                                                        .EXTRN  CONV$GW_VBN_FS_PTR
                                                        .EXTRN  CONV$GW_VBN_FS_PTR0
                                                        .EXTRN  CONV$GW_LCB_PTR
                                                        .EXTRN  SYS$WAIT, SYS$WRITE

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

                                    0000V 30 00000 CONV$$WRITE_BUCKET::
                                                        BSBW    SET_NXTBKT                                          ; 0171
                    0000G  CF            59  D0 00003   MOVL    BUCKET, CONV$AB_OUT_RAB+40                          ; 0175
                    0000G  CF        20  AA  B0 00008   MOVW    32(CTX), CONV$AB_OUT_RAB+34                         ; 0176
                    0000G  CF        08  AA  D0 0000E   MOVL    8(CTX), CONV$AB_OUT_RAB+56                          ; 0177
            0A             6A            03  E1 00014   BBC     #3, (CTX), 1$                                       ; 0182
                                      0000V  30 00018   BSBW    SWAP_BUCKET                                         ; 0188
                    0000G  CF            01  88 0001B   BISB2   #1, CONV$AB_OUT_RAB+4                               ; 0192
                                         05  11 00020   BRB     2$                                                  ;
                    0000G  CF            01  8A 00022 1$: BICB2 #1, CONV$AB_OUT_RAB+4                               ; 0199
                           0000G  CF     9F 00027 2$:   PUSHAB  CONV$AB_OUT_RAB                                     ; 0203
                00000000G  00            01  FB 0002B   CALLS   #1, SYS$WAIT                                        ;
                           0000G  CF     9F 00032       PUSHAB  CONV$$RMS_ERROR                                     ; 0207
                           0000G  CF     9F 00036       PUSHAB  CONV$AB_OUT_RAB                                     ;
                00000000G  00            02  FB 0003A   CALLS   #2, SYS$WRITE                                       ;
                                             05 00041   RSB                                                         ; 0211

; Routine Size:  66 bytes,    Routine Base:  _CONV$FAST_S + 0000


;  215          0212  1
```

```
%SBTTL  'SET_NXTBKT'
ROUTINE SET_NXTBKT : CL$JSB_REG_9 NOVALUE =
!++
!
! Functional Description:
!
!       Writes the next bucket vbn field in the current bucket
!
! Calling Sequence:
!
!       SET_NXTBKT()
!
! Input Parameters:
!       none
!
! Implicit Inputs:
!       none
!
! Output Parameters:
!       none
!
! Implicit Outputs:
!       none
!
! Routine Value:
!       none
!
! Routines Called:
!       none
!
! Side Effects:
!       none
!
!--

    BEGIN

    DEFINE_CTX;
    DEFINE_BUCKET;

    LOCAL       AREA;

    ! Get the area that the bucket is in
    !
    AREA = .CTX [ CTX$B_AREA ];

    ! If this is the last bucket in a horz. chain
    ! then write back pointers to the beginning of the chain
    ! else write next bucket pointers
    !
    IF .BUCKET [ BKT$V_LASTBKT ]
    THEN

        ! Next bucket pointer points to the first bucket in this chain
        !
        BUCKET [ BKT$L_NXTBKT ] = .CTX [ CTX$L_FIRST_VBN ]
    ELSE
```

```
;  274        0270  2
;  275        0271  2          ! First see if the next bucket will fit in the current extent. If it dosen't
;  276        0272  2          ! then set the next bkt. ptr to EOF else set it to next bucket
;  277        0273  2          !
;  278        0274  3          IF ( .CONV$AR_AREA_BLOCK [ .AREA,AREA$B_ARBKTSZ ]  GTR
;  279        0275  4                ( .CONV$AR_AREA_BLOCK [ .AREA,AREA$L_CNBLK ] -
;  280        0276  3                      .CONV$AR_AREA_BLOCK [ .AREA,AREA$L_USED ] ) )
;  281        0277  2          THEN
;  282        0278  2
;  283        0279  2              ! Next bucket pointer points to end of file where the next extend
;  284        0280  2              ! will come from. (in INIT_BUCKET)
;  285        0281  2              !
;  286        0282  2              BUCKET [ BKT$L_NXTBKT ] = .CONV$GL_EOF_VBN
;  287        0283  2          ELSE
;  288        0284  2
;  289        0285  2              ! Next bucket pointer points to the next bucket VBN in this area
;  290        0286  2              !
;  291        0287  2              BUCKET [ BKT$L_NXTBKT ] = .CONV$AR_AREA_BLOCK [ .AREA,AREA$L_NXTVBN ];
;  292        0288  2
;  293        0289  2      RETURN
;  294        0290  2
;  295        0291  1      END;



                                                52  DD 00000 SET_NXTBKT:
                                                                      PUSHL    R2                                       ; 0214
                                     50      01  AA  9A 00002         MOVZBL   1(CTX), AREA                             ; 0257
                                     07      0D  A9  E9 00006         BLBC     13(BUCKET), 1$                           ; 0263
                               08    A9      24  AA  D0 0000A         MOVL     36(CTX), 8(BUCKET)                       ; 0268
                                                 2E  11 0000F         BRB      3$                                       ;
                                     51   0000G  CF  D0 00011 1$:     MOVL     CONV$AR_AREA_BLOCK, R1                   ; 0274
                    50               50      06  78 00016             ASHL     #6, R0, R0                               ;
                                          10 A041  9F 0001A           PUSHAB   16(R0)[R1]                               ; 0276
                                          14 A041  9F 0001E           PUSHAB   20(R0)[R1]                               ;
                    52               9E      9E  C3 00022             SUBL3    @(SP)+, @(SP)+, R2                       ;
          52        03 A041          08      00  ED 00026             CMPZV    #0, #8, 3(R0)[R1], R2                    ; 0275
                                             08  15 0002D             BLEQ     2$                                       ;
                               08    A9   0000G  CF  D0 0002F         MOVL     CONV$GL_EOF_VBN, 8(BUCKET)               ; 0282
                                             08  11 00035             BRB      3$                                       ;
                                          18 A041  9F 00037 2$:       PUSHAB   24(R0)[R1]                               ; 0287
                               08    A9      9E  D0 0003B             MOVL     @(SP)+, 8(BUCKET)                        ;
                                             04  BA 0003F 3$:         POPR     #^M<R2>                                  ; 0291
                                                 05 00041             RSB                                               ;

; Routine Size:  66 bytes,    Routine Base:  _CONV$FAST_S + 0042
```

```
1 %SBTTL  'SWAP_BUCKET'
1 ROUTINE SWAP_BUCKET :  CL$JSB_REG_9 NOVALUE =
1 !++
1 !
1 ! Functional Description:
1 !
1 !       Swaps the current bucket with the secondary bucket for
1 !       double buffering
1 !
1 ! Calling Sequence:
1 !
1 !       SWAP_BUCKET()
1 !
1 ! Input Parameters:
1 !       none
1 !
1 ! Implicit Inputs:
1 !       none
1 !
1 ! Output Parameters:
1 !       none
1 !
1 ! Implicit Outputs:
1 !       none
1 !
1 ! Routine Value:
1 !       none
1 !
1 ! Routines Called:
1 !       none
1 !
1 ! Side Effects:
1 !       none
1 !
1 !--
1
2     BEGIN
2
2     DEFINE_CTX;
2     DEFINE_BUCKET;
2
2     IF .CTX [ CTX$V_DBX ]
2     THEN
3         BEGIN
3         BUCKET = .CTX [ CTX$L_PT0 ];
3         CTX [ CTX$L_CURRENT_BUFFER ] = .CTX [ CTX$L_PT0 ];
3         CTX [ CTX$L_END ] = .CTX [ CTX$L_EN0 ]
3         END
2     ELSE
3         BEGIN
3         BUCKET = .CTX [ CTX$L_PT1 ];
3         CTX [ CTX$L_CURRENT_BUFFER ] = .CTX [ CTX$L_PT1 ];
3         CTX [ CTX$L_END ] = .CTX [ CTX$L_EN1 ]
2         END;
2
2     CTX [ CTX$V_DBX ] = NOT .CTX [ CTX$V_DBX ];
2
```

```
;  354          0349  2      RETURN
;  355          0350  2
;  356          0351  1      END;



                      10           6A           04  E1 00000 SWAP_BUCKET:
                                                                  BBC      #4, (CTX), 1$                          ; 0333
                                   59      10   AA  D0 00004      MOVL     16(CTX), BUCKET                        ; 0336
                              04   AA      10   AA  D0 00008      MOVL     16(CTX), 4(CTX)                        ; 0337
                              0C   AA      14   AA  D0 0000D      MOVL     20(CTX), 12(CTX)                       ; 0338
                                                0E  11 00012      BRB      2$                                     ;
                                   59      18   AA  D0 00014 1$:  MOVL     24(CTX), BUCKET                        ; 0342
                              04   AA      18   AA  D0 00018      MOVL     24(CTX), 4(CTX)                        ; 0343
                              0C   AA      1C   AA  D0 0001D      MOVL     28(CTX), 12(CTX)                       ; 0344
                                   6A           10  8C 00022 2$:  XORB2    #16, (CTX)                             ; 0347
                                                    05 00025      RSB                                             ; 0351

; Routine Size:  38 bytes,    Routine Base:  _CONV$FAST_S + 0084
```

```
1 %SBTTL  'GET_BUCKET'
1 GLOBAL ROUTINE  CONV$$GET_BUCKET ( AREA ) :  CL$JSB_REG_9 NOVALUE =
1 !++
1 !
1 ! Functional Description:
1 !
1 !         Allocates and formats a pair of buckets in memory
1 !
1 ! Calling Sequence:
1 !
1 !         CONV$$GET_BUCKET( area )
1 !
1 ! Input Parameters:
1 !
1 !         area    - Area from witch the bucket is to be allocated
1 !
1 ! Implicit Inputs:
1 !         none
1 !
1 ! Output Parameters:
1 !         none
1 !
1 ! Implicit Outputs:
1 !         none
1 !
1 ! Routine Value:
1 !         none
1 !
1 ! Routines Called:
1 !
1 !         CONV$$GET_TEMP_VM
1 !         INIT_BUCKET
1 !
1 ! Side Effects:
1 !         none
1 !
1 !--
1
2     BEGIN
2
2     DEFINE_CTX;
2     DEFINE_BUCKET;
2     DEFINE_KEY_DESC;
2
2     LOCAL
2         BYTES;
2
2     ! Get the number of bytes per bucket for that area:
2     ! ( bytes = blocks * block_size * 2 ) NOTE: *2 for double buffering
2     !
2     BYTES = ( .CONV$AR_AREA_BLOCK [ .AREA,AREA$B_ARBKTSZ ] * BLOCK_SIZE * 2 );
2
2     ! Get the space.
2     !
2     BUCKET = CONV$$GET_TEMP_VM ( .BYTES );
2
2     ! For double buffering then hide the extra buffer for later
```

```
2       !
2       BYTES = .BYTES / 2;
2
2       ! Initialize some static values
2       !
2       CTX [ CTX$L_CURRENT_BUFFER ] = .BUCKET;
2       CTX [ CTX$L_SIZ ] = .BYTES;
2
2       ! CTX$L_END points at the first free byte BEFORE the check byte
2       !
2       CTX [ CTX$L_END ] = .BUCKET + .BYTES - 2;
2       CTX [ CTX$B_AREA ] = .AREA;
2
2       !++
2       !
2       ! Init. static fields in the bucket
2       !
2       !--
2
2       ! Level (all prologues)
2       !
2       BUCKET [ BKT$B_LEVEL ] = .CTX [ CTX$B_LEVEL ];
2
2       ! Prologue dependent fields and some pointers into the
2       ! bucket and some sizes
2       !
2       IF .CONV$GB_PROL_V3
2       THEN
3           BEGIN           ! Prologue 3
3
3           ! Bucket key of ref
3           !
3           BUCKET [ BKT$B_INDEXNO ] = .KEY_DESC [ KEY$B_KEYREF ];
3
3           ! For level 0 (data) buckets we can have a LCB pointer
3           !
3           IF .CTX [ CTX$B_LEVEL ] EQL 0
3           THEN
3
3               IF ( .KEY_DESC [ KEY$B_KEYREF ] EQL 0 ) AND
3                                                   .KEY_DESC [ KEY$V_DUPKEYS ]
3               THEN
4                   BEGIN
4
4                   ! Only primary data bucket have a LCB pointer
4                   !
4                   CONV$GW_LCB_PTR = .BYTES - 6;
4
4                   CTX [ CTX$W_FREE ] = .BYTES - BKT$C_OVERHDSZ - 6
4
4                   END
3               ELSE
3                   CTX [ CTX$W_FREE ] = .BYTES - BKT$C_OVERHDSZ - 2
3
3           ELSE
4               BEGIN
4
```

```
                ! Index buckets only have a VBN freespace pointer
                !
                CONV$GW_VBN_FS_PTR = .BYTES - 4;

                CTX [ CTX$W_FREE ] = .BYTES - BKT$C_OVERHDSZ - 4

                END
            END                 ! Prologue 3
        ELSE
            BEGIN               ! Prologue 1,2

            ! Bucket area number
            !
            BUCKET [ BKT$B_AREANO ] = .AREA;

            ! Highest record id
            !
            BUCKET [ BKT$B_LSTRECID ] = 255;

            ! The space avail. is bytes - overhead - check byte
            !
            CTX [ CTX$W_FREE ] = .BYTES - BKT$C_OVERHDSZ - 1

            END;                ! Prologue 1,2

        ! For double buffering set up the pointers to the buffers and init
        ! the extra buffer
        !
        ! Set up the various pointers
        !
        CTX [ CTX$L_PT0 ] = .BUCKET;
        CTX [ CTX$L_PT1 ] = .BUCKET + .BYTES;
        CTX [ CTX$L_EN0 ] = .CTX [ CTX$L_END ];
        CTX [ CTX$L_EN1 ] = .CTX [ CTX$L_END ] + .BYTES;

        ! Init the second buffer by coping the static overhead into it
        !
        CH$MOVE( BKT$K_OVERHDSZ + 1,.CTX [ CTX$L_PT0 ],.CTX [ CTX$L_PT1 ] );

        ! Tell everyone that we are doind double buffering and which bucket
        ! are pointing to
        !
        CTX [ CTX$V_DBF ] = _SET;
        CTX [ CTX$V_DBX ] = _CLEAR;         ! Clear = bucket 0, Set = bucket 1

        ! Initialize dynamic values and update area descriptor
        !
        CONV$$INIT_BUCKET();

        ! Set the pointer for this level
        !
        CTX [ CTX$L_FIRST_VBN ] = .CTX [ CTX$L_CURRENT_VBN ];

        ! Say that the bucket is ready and that will have the first record
        !
        CTX [ CTX$V_RDY ] = _SET;
        CTX [ CTX$V_FST ] = _SET;
```

```
;  529            0523  2
;  530            0524  2        RETURN
;  531            0525  2
;  532            0526  1        END;



                                          3C  BB 00000 CONV$$GET_BUCKET::
                                                                 PUSHR   #^M<R2,R3,R4,R5>                                    ; 0353
                      50        14   AE        06  78 00002      ASHL    #6, AREA, R0                                        ; 0402
                                     50  0000G CF  C0 00007      ADDL2   CONV$AR_AREA_BLOCK, R0                              ;
                                     51     03 A0  9A 0000C      MOVZBL  3(R0), BYTES                                        ;
                      51             51        0A  78 00010      ASHL    #10, BYTES, BYTES                                   ;
                                               51  DD 00014      PUSHL   BYTES                                               ; 0406
                                           0000G  30 00016       BSBW    CONV$$GET_TEMP_VM                                   ;
                                     5E        04  C0 00019      ADDL2   #4, SP                                              ;
                                     59        50  D0 0001C      MOVL    R0, BUCKET                                          ;
                                     51        02  C6 0001F      DIVL2   #2, BYTES                                           ; 0410
                                04   AA        59  D0 00022      MOVL    BUCKET, 4(CTX)                                      ; 0414
                                20   AA        51  D0 00026      MOVL    BYTES, 32(CTX)                                      ; 0415
                                0C   AA   FE A149  9E 0002A      MOVAB   -2(BYTES)[BUCKET], 12(CTX)                          ; 0419
                                01   AA     14 AE  90 00030      MOVB    AREA, 1(CTX)                                        ; 0420
                                0C   A9     02 AA  90 00035      MOVB    2(CTX), 12(BUCKET)                                  ; 0430
                                     50     F2 A1  9E 0003A      MOVAB   -14(R1), R0                                         ; 0457
                                     34  0000G CF  E9 0003E      BLBC    CONV$GB_PROL_V3, 3$                                 ; 0435
                                01   A9     15 AB  90 00043      MOVB    21(KEY_DESC), 1(BUCKET)                             ; 0441
                                            02 AA  95 00048      TSTB    2(CTX)                                              ; 0445
                                               1D  12 0004B      BNEQ    2$                                                  ;
                                            15 AB  95 0004D      TSTB    21(KEY_DESC)                                        ; 0448
                                               11  12 00050      BNEQ    1$                                                  ;
                                     0D     10 AP  E9 00052      BLBC    16(KEY_DESC), 1$                                    ; 0449
              0000G   CF             51        06  A3 00056      SUBW3   #6, BYTES, CONV$GW_LCB_PTR                          ; 0455
                 28   AA             50        06  A3 0005C      SUBW3   #6, R0, 40(CTX)                                     ; 0457
                                               22  11 00061      BRB     4$                                                  ;
                 28   AA             50        02  A3 00063 1$:  SUBW3   #2, R0, 40(CTX)                                     ; 0461
                                               1B  11 00068      BRB     4$                                                  ; 0448
              0000G   CF             51        04  A3 0006A 2$:  SUBW3   #4, BYTES, CONV$GW_VBN_FS_PTR                       ; 0468
                 28   AA             50        04  A3 00070      SUBW3   #4, R0, 40(CTX)                                     ; 0470
                                               0E  11 00075      BRB     4$                                                  ; 0445
                                01   A9     14 AE  90 00077 3$:  MOVB    AREA, 1(BUCKET)                                     ; 0479
                                07   A9        01  8E 0007C      MNEGB   #1, 7(BUCKET)                                       ; 0483
                 28   AA             50        01  A3 00080      SUBW3   #1, R0, 40(CTX)                                     ; 0487
                                10   AA        59  D0 00085 4$:  MOVL    BUCKET, 16(CTX)                                     ; 0496
                 18   AA             59        51  C1 00089      ADDL3   BYTES, BUCKET, 24(CTX)                              ; 0497
                                14   AA     0C AA  D0 0008E      MOVL    12(CTX), 20(CTX)                                    ; 0498
                                1C   AA   0C BA41  9E 00093      MOVAB   @12(CTX)[BYTES], 28(CTX)                            ; 0499
                 18   BA        10   BA        0F  28 00099      MOVC3   #15, @16(CTX), @24(CTX)                             ; 0503
                                     6A        08  88 0009F      BISB2   #8, (CTX)                                           ; 0508
                                     6A        10  8A 000A2      BICB2   #16, (CTX)                                          ; 0509
                                           0000V  30 000A5       BSBW    CONV$$INIT_BUCKET                                   ; 0513
                                24   AA     08 AA  D0 000A8      MOVL    8(CTX), 36(CTX)                                     ; 0517
                                     6A        05  88 000AD      BISB2   #5, (CTX)                                           ; 0522
                                               3C  BA 000B0      POPR    #^M<R2,R3,R4,R5>                                    ; 0526
                                                   05 000B2      RSB                                                         ;
```

```
; Routine Size:  179 bytes,    Routine Base:  _CONV$FAST_
```

```
1 %SBTTL  'INIT_BUCKET'
1 GLOBAL ROUTINE  CONV$$INIT_BUCKET :  CL$JSB_REG_9 NOVALUE =
1 !++
1 !
1 ! Functional Description:
1 !
1 !       Gets a new VBN for a bucket in the proper area and initializes
1 !       all of the dynamic fields in the bucket
1 !
1 ! Calling Sequence:
1 !
1 !       CONV$$INIT_BUCKET()
1 !
1 ! Input Parameters:
1 !       none
1 !
1 ! Implicit Inputs:
1 !       none
1 !
1 ! Output Parameters:
1 !       none
1 !
1 ! Implicit Outputs:
1 !       none
1 !
1 ! Routine Value:
1 !       none
1 !
1 ! Routines Called:
1 !
1 !       CONV$$EXTEND_AREA
1 !
1 ! Side Effects:
1 !
1 !       Could extend the allocation of the output file
1 !
1 !--
1
2    BEGIN
2
2    DEFINE_CTX;
2    DEFINE_BUCKET;
2    DEFINE_KEY_DESC;
2
2    LOCAL       AREA;
2
2    AREA = .CTX [ CTX$B_AREA ];
2
2    ! See if the bucket will fit in the current extent if it doesent extend the
2    ! file.
2    !
3    IF ( .CONV$AR_AREA_BLOCK [ .AREA,AREA$B_ARBKTSZ ] GTR
4        ( .CONV$AR_AREA_BLOCK [ .AREA,AREA$L_CNBLK ] -
3                                 .CONV$AR_AREA_BLOCK [ .AREA,AREA$L_USED ] ) )
2    THEN
2
2        ! Extend area
```

```
2                !
2                CONV$$EXTEND_AREA ( .AREA );
2
2           ! Set the VBN of the bucket and determine the size of the VBN pointers
2           !
3           BEGIN       ! VBN local
3
3           LOCAL       VBN;
3
3           ! Get the next VBN of this bucket
3           !
3           VBN = .CONV$AR_AREA_BLOCK [ .AREA,AREA$L_NXTVBN ];
3
3           CTX [ CTX$L_CURRENT_VBN ] = .VBN;
3
3           ! Determine the pointer size needed for this VBN
3           !
3           IF .VBN LSS 65536
3           THEN
3               CTX [ CTX$V_VBN ] = 0           ! 2 byte pointer
3           ELSE
3               IF .VBN LSS 1048576
3               THEN
3                   CTX [ CTX$V_VBN ] = 1       ! 3 byte
3               ELSE
3                   CTX [ CTX$V_VBN ] = 2       ! 4 byte
3
2           END;        ! VBN local
2
2           !++
2           !
2           ! Update the area descriptor to account for the new bucket
2           !
2           !--
2
2           ! Correct the pointers and counters in the prologue area descriptor
2           !
2           CONV$AR_AREA_BLOCK [ .AREA,AREA$L_USED ] =
2                                       .CONV$AR_AREA_BLOCK [ .AREA,AREA$L_USED ] +
2                                       .CONV$AR_AREA_BLOCK [ .AREA,AREA$B_ARBKTSZ ];
2
2           CONV$AR_AREA_BLOCK [ .AREA,AREA$L_NXTVBN ] =
2                                       .CONV$AR_AREA_BLOCK [ .AREA,AREA$L_NXTVBN ] +
2                                       .CONV$AR_AREA_BLOCK [ .AREA,AREA$B_ARBKTSZ ];
2
2           CONV$AR_AREA_BLOCK [ .AREA,AREA$L_TOTAL_ALLOC ] =
2                                       .CONV$AR_AREA_BLOCK [ .AREA,AREA$L_TOTAL_ALLOC ] +
2                                       .CONV$AR_AREA_BLOCK [ .AREA,AREA$B_ARBKTSZ ];
2
2           !++
2           !
2           ! Init. dynamic fields in bucket
2           !
2           !--
2
2           ! Bucket control byte (all prologues)
2           !
```

```
2       BUCKET [ BKT$B_BKTCB ] = _CLEAR;
2
2       ! The freespace always points just past the bucket overhead (all prologues)
2       !
2       BUCKET [ BKT$W_FREESPACE ] = BKT$C_OVERHDSZ;
2
2       ! Set address sample (all prologues)
2       !
2       BUCKET [ BKT$W_ADRSAMPLE ] = .CTX [ CTX$L_CURRENT_VBN ];
2
2       ! Prologue dependent fields
2       !
2       IF .CONV$GB_PROL_V3
2       THEN
3           BEGIN           ! Prologue 3
3
3           ! Index buckets have VBN freespace pointers
3           !
3           IF .CTX [ CTX$B_LEVEL ] NEQ 0
3           THEN
4               BEGIN
4
4               LOCAL CTX_M1 : REF BLOCK [ ,BYTE ];
4
4               ! The VBN of the bucket one level down determines size
4               ! of the VBN pointers in this bucket
4               !
4               CTX_M1 = .CTX - CTX$K_BLN;
4
4               BUCKET [ BKT$V_PTR_SZ ] = .CTX_M1 [ CTX$V_VBN ];
4
4               ! The vbn freespace points to the byte just before the pointer
4               !
4               BUCKET [ BKT$W_VBNFS ] = .CONV$GW_VBN_FS_PTR - 1
4
3               END;
3
3           ! Reset the next record ID
3           .
3           BUCKET [ BKT$W_NXTRECID ] = 1
3
3           END             ! Prologue 3
2       ELSE
3           BEGIN           ! Prologue 1,2
3
3           ! Reset the record ID
3           !
3           BUCKET [ BKT$B_NXTRECID ] = 1
3
2           END;            ! Prologue 1,2
2
2       ! Reset the avaiable space in the bucket
2       !
2       CTX [ CTX$W_SPC ] = .CTX [ CTX$W_FREE ];
2
2       ! Indicate that the bucket has not been used yet
2       !
```

```
;  705         0698  2     CTX [ CTX$W_USE ] = 0;
;  706         0699  2
;  707         0700  2     RETURN
;  708         0701  2
;  709         0702  1     END;



                                          0C   BB 00000 CONV$$INIT_BUCKET::
                                                                   PUSHR    #^M<R2,R3>                              ; 0528
                                 50      01   AA   9A 00002         MOVZBL   1(CTX), AREA                            ; 0573
                                 51   0000G   CF   D0 00006         MOVL     CONV$AR_AREA_BLOCK, R1                  ; 0578
                    52           50           06   78 0000B         ASHL     #6, AREA, R2                            ;
                                         10 A241   9F 0000F         PUSHAB   16(R2)[R1]                              ; 0580
                                         14 A241   9F 00013         PUSHAB   20(R2)[R1]                              ;
                    53           9E           9E   C3 00017         SUBL3    @(SP)+, @(SP)+, R3                      ;
     53       03 A241            08           00   ED 0001B         CMPZV    #0, #8, 3(R2)[R1], R3                   ; 0579
                                              08   15 00022         BLEQ     1$                                      ;
                                              50   DD 00024         PUSHL    AREA                                    ; 0585
                                           0000V   30 00026         BSBW     CONV$$EXTEND_AREA                       ;
                                 5E           04   C0 00029         ADDL2    #4, SP                                  ;
                                 50   0000G   CF   D0 0002C 1$:     MOVL     CONV$AR_AREA_BLOCK, R0                  ; 0595
                                         18 A240   9F 00031         PUSHAB   24(R2)[R0]                              ;
                                 51           9E   D0 00035         MOVL     @(SP)+, VBN                             ;
                          08     AA           51   D0 00038         MOVL     VBN, 8(CTX)                             ; 0597
                    00010000     8F           51   D1 0003C         CMPL     VBN, #65536                             ; 0601
                                              06   18 00043         BGEQ     2$                                      ;
                                 6A      60   8F   8A 00045         BICB2    #96, (CTX)                              ; 0603
                                              15   11 00049         BRB      4$                                      ;
                    00100000     8F           51   D1 0004B 2$:     CMPL     VBN, #1048576                           ; 0605
                                              07   18 00052         BGEQ     3$                                      ;
     6A             02           05           01   F0 00054         INSV     #1, #5, #2, (CTX)                       ; 0607
                                              05   11 00059         BRB      4$                                      ;
     6A             02           05           02   F0 0005B 3$:     INSV     #2, #5, #2, (CTX)                       ; 0609
                                 51      03 A240   9A 00060 4$:     MOVZBL   3(R2)[R0], R1                           ; 0623
                                         14 A240   9F 00065         PUSHAB   20(R2)[R0]                              ;
                                 9E           51   C0 00069         ADDL2    R1, @(SP)+                              ;
                                         18 A240   9F 0006C         PUSHAB   24(R2)[R0]                              ; 0627
                                 9E           51   C0 00070         ADDL2    R1, @(SP)+                              ;
                                         32 A240   9F 00073         PUSHAB   50(R2)[R0]                              ; 0631
                                 9E           51   C0 00077         ADDL2    R1, @(SP)+                              ;
                                         0D   A9   94 0007A         CLRB     13(BUCKET)                              ; 0641
                          04     A9           0E   B0 0007D         MOVW     #14, 4(BUCKET)                          ; 0645
                          02     A9      08   AA   B0 00081         MOVW     8(CTX), 2(BUCKET)                       ; 0649
                                 26   0000G   CF   E9 00086         BLBC     CONV$GB_PROL_V3, 6$                     ; 0653
                                         02   AA   95 0008B         TSTB     2(CTX)                                  ; 0659
                                              1B   13 0008E         BEQL     5$                                      ;
                                 50      A4   AA   9E 00090         MOVAB    -92(R10), CTX_M1                        ; 0668
     51             60           02           05   EF 00094         EXTZV    #5, #2, (CTX_M1), R1                    ; 0670
 0D  A9             02           03           51   F0 00099         INSV     R1, #3, #2, 13(BUCKET)                  ;
                                 50   0000G   CF   3C 0009F         MOVZWL   CONV$GW_VBN_FS_PTR, R0                  ; 0674
                                            6049   9F 000A4         PUSHAB   (R0)[BUCKET]                            ;
                    9E           50           01   A3 000A7         SUBW3    #1, R0, @(SP)+                          ;
                          06     A9           01   B0 000AB 5$:     MOVW     #1, 6(BUCKET)                           ; 0680
                                              04   11 000AF         BRB      7$                                      ;
```

```
                         06    A9          01  90 000B1 6$:       MOVB    #1, 6(BUCKET)                                 ; 0688
                         2A    AA     28   AA  3C 000B5 7$:       MOVZWL  40(CTX), 42(CTX)                              ; 0694
                                           0C  BA 000BA           POPR    #^M<R2,R3>                                    ; 0702
                                               05 000BC           RSB                                                   ;

; Routine Size:  189 bytes,    Routine Base:  _CONV$FAST_S + 015D
```

```
1 %SBTTL  'EXTEND_AREA'
1 GLOBAL ROUTINE  CONV$$EXTEND_AREA ( AREA ) :  CL$EXTEND_AREA NOVALUE =
1 !++
1 !
1 ! Functiona' Description:
1 !
1 !           Extens the disk allocation of a specified area
1 !
1 ! Calling Sequence:
1 !
1 !           EXTEND_AREA ( .area )
1 !
1 ! Input Parameters:
1 !
1 !           AREA -   Area to be extended
1 !
1 ! Implicit Inputs:
1 !           none
1 !
1 ! Output Parameters:
1 !           none
1 !
1 ! Implicit Outputs:
1 !           none
1 !
1 ! Routine Value:
1 !           none
1 !
1 ! Routines Called:
1 !
1 !           $EXTEND
1 !           CONV$$RMS_ERROR - By RMS as an AST
1 !
1 ! Side Effects:
1 !           none
1 !
1 !--
1
2     BEGIN
2
2     ! Set the allocation to the largest size. (To aviod bad parameters)
2     !
2     CONV$AB_OUT_FAB [ FAB$L_ALQ ] =
2               MAX( .CONV$AR_AREA_BLOCK [ .AREA,AREA$W_DEQ ],
2                                   .CONV$AR_AREA_BLOCK [ .AREA,AREA$B_ARBKTSZ ] );
2
2     ! Wait on the rab in case we have a asyinc operation going on
2     !
2     $WAIT( RAB=CONV$AB_OUT_RAB );
2
2     ! Stuff the error if we get one
2     !
2     CONV$AB_OUT_FAB [ FAB$L_CTX ] = CONV$_EXTN_ERR;
2
2     ! Do the extend
2     !
2     $EXTEND ( FAB=CONV$AB_OUT_FAB,ERR=CONV$$RMS_ERROR );
```

```
;  768       0760  2
;  769       0761  2      ! Reset some pointers in the prologue
;  770       0762  2      !
;  771       0763  2      CONV$AR_AREA_BLOCK [ .AREA,AREA$L_CVBN ] = .CONV$GL_EOF_VBN;
;  772       0764  2      CONV$AR_AREA_BLOCK [ .AREA,AREA$L_NXTVBN ] = .CONV$GL_EOF_VBN;
;  773       0765  2      CONV$AR_AREA_BLOCK [ .AREA,AREA$L_CNBLK ] = .CONV$AB_OUT_FAB [ FAB$L_ALQ ];
;  774       0766  2      CONV$AR_AREA_BLOCK [ .AREA,AREA$L_USED ] = 0;
;  775       0767  2      CONV$GL_EOF_VBN = .CONV$GL_EOF_VBN + .CONV$AB_OUT_FAB [ FAB$L_ALQ ];
;  776       0768  2
;  777       0769  2      RETURN
;  778       0770  2
;  779       0771  1      END;


                                                                       .EXTRN  SYS$EXTEND

                                                52  DD 00000 CONV$$EXTEND_AREA::
                                                                       PUSHR   R2                                                   ; 0704
                        52       08    AE             06  78 00002     ASHL    #6, AREA, R2                                         ; 0746
                        51             52     0000G   CF  C1 00007     ADDL3   CONV$AR_AREA_BLOCK, R2, R1                           ;
                        50             52     0000G   CF  C1 0000D     ADDL3   CONV$AR_AREA_BLOCK, R2, R0                           ; 0747
                                       51        24   A1  3C 00013     MOVZWL  36(R1), R1                                           ;
          51      03   A0              08             00  ED 00017     CMPZV   #0, #8, 3(R0), R1                                    ;
                                                      04  15 0001D     BLEQ    1$                                                   ;
                                       51        03   A0  9A 0001F     MOVZBL  3(R0), R1                                            ;
                             0000G     CF             51  D0 00023 1$: MOVL    R1, CONV$AB_OUT_FAB+16                               ; 0746
                                              0000G   CF  9F 00028     PUSHAB  CONV$AB_OUT_RAB                                      ; 0751
                         00000000G     00             01  FB 0002C     CALLS   #1, SYS$WAIT                                         ;
                             0000G     CF 00000000G   8F  D0 00033     MOVL    #CONV$_EXTN_ERR, CONV$AB_OUT_FAB+24                  ; 0755
                                              0000G   CF  9F 0003C     PUSHAB  CONV$$RMS_ERROR                                      ; 0759
                                              0000G   CF  9F 00040     PUSHAB  CONV$AB_OUT_FAB                                      ;
                         00000000G     00             02  FB 00044     CALLS   #2, SYS$EXTEND                                       ;
                                       50     0000G   CF  D0 0004B     MOVL    CONV$AR_AREA_BLOCK, R0                               ; 0763
                                               0C A240    9F 00050     PUSHAB  12(R2)[R0]                                           ;
                                       9E     0000G   CF  D0 00054     MOVL    CONV$GL_EOF_VBN, @(SP)+                              ;
                                               18 A240    9F 00059     PUSHAB  24(R2)[R0]                                           ; 0764
                                       9E     0000G   CF  D0 0005D     MOVL    CONV$GL_EOF_VBN, @(SP)+                              ;
                                               10 A240    9F 00062     PUSHAB  16(R2)[R0]                                           ; 0765
                                       9E     0000G   CF  D0 00066     MOVL    CONV$AB_OUT_FAB+16, @(SP)+                           ;
                                               14 A240    9F 0006B     PUSHAB  20(R2)[R0]                                           ; 0766
                                                      9E  D4 0006F     CLRL    @(SP)+                                               ;
                             0000G     CF     0000G   CF  C0 00071     ADDL2   CONV$AB_OUT_FAB+16, CONV$GL_EOF_VBN                  ; 0767
                                                      04  BA 00078     POPR    #^M<R2>                                              ; 0771
                                                      05 0007A         RSB                                                          ;

; Routine Size:  123 bytes,    Routine Base:  _CONV$FAST_S + 021A
```

```
1 %SBTTL  'CONVERT_VBN_ID'
1 GLOBAL ROUTINE  CONV$$CONVERT_VBN_ID     : CL$CONVERT_VBN_ID NOVALUE =
1 !++
1 !
1 ! Functional Description:
1 !
1 !       Converts the rfa created by the sort of the output file
1 !       into a VBN and ID to be used as an alturnate index pointer
1 !
1 ! Calling Sequence:
1 !
1 !       CONV$$CONVERT_VBN_ID()
1 !
1 ! Input Parameters:
1 !       none
1 !
1 ! Implicit Inputs:
1 !       none
1 !
1 ! Output Parameters:
1 !       none
1 !
1 ! Implicit Outputs:
1 !
1 !       SORT_VBN - R6   VBN of the primary data record for this key
1 !       SORT_ID  - R7   ID of the primary data record for this key
1 !
1 ! Routine Value:
1 !       none
1 !
1 ! Routines Called:
1 !       none
1 !
1 ! Side Effects:
1 !       none
1 !
1 !--
1
2    BEGIN
2
2    EXTERNAL REGISTER
2        SORT_VBN,
2        SORT_ID;
2
2    !   Get the VBN an offset returned by SORT by RFA
2    !
2    LOCAL       SORT_RFA : REF BLOCK [ ,BYTE ];
2
2    SORT_RFA = .CONV$GL_RFA_BUFFER;
2
2    SORT_VBN = .SORT_RFA [ 0,0,32,0 ];
2    SORT_ID = .SORT_RFA [ 4,0,16,0 ];
2
2    RETURN
2
1    END;
```

```
                                50      0000G  CF   D0 00000 CONV$$CONVERT_VBN_ID::
                                                                     MOVL    CONV$GL_RFA_BUFFER, SORT_RFA             ; 0820
                                56             60   D0 00005         MOVL    (SORT_RFA), SORT_VBN                     ; 0822
                                57       04    A0   3C 00008         MOVZWL  4(SORT_RFA), SORT_ID                     ; 0823
                                                    05 0000C         RSB                                              ; 0827

; Routine Size:  13 bytes,    Routine Base:  _CONV$FAST_S + 0295


;  837            0828  1
;  838            0829  0 END      ELUDOM




;                                   PSECT SUMMARY
;
;       Name                      Bytes                    Attributes
;
;  _CONV$FAST_D                       4  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  _CONV$FAST_S                     674  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        40         0      1000          00:01.9
;  _$255$DUA28:[CONV.SRC]CONVERT.L32;1             165        31        18        17          00:00.2




;                                   COMMAND QUALIFIERS

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

; Size:         674 code + 4 data bytes
; Run Time:        00:18.4
; Elapsed Time:    01:08.8
; Lines/CPU Min:     2710
; Lexemes/CPU-Min: 19556
; Memory Used:  136 pages
; Compilation Complete
```

0065 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

CONVDATA
LIS

CONVFSTLD
LIS

CONVFSTIO
LIS

CONVFSTRC
LIS

CONVFILES
LIS

CONVERROR
LIS

CONVFASTM
LIS

CONVDCL
LIS