I 6

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

FILEID**CONVFSTLD

```
  CCCCCCCC    OOOOOO    NN      NN  VV      VV  FFFFFFFFFF    SSSSSSSS  TTTTTTTTTT  LL          DDDDDDDD
  CCCCCCCC    OOOOOO    NN      NN  VV      VV  FFFFFFFFFF    SSSSSSSS  TTTTTTTTTT  LL          DDDDDDDD
CC          OO      OO  NN      NN  VV      VV  FF          SS              TT      LL          DD      DD
CC          OO      OO  NN      NN  VV      VV  FF          SS              TT      LL          DD      DD
CC          OO      OO  NNNN    NN  VV      VV  FF          SS              TT      LL          DD      DD
CC          OO      OO  NNNN    NN  VV      VV  FF          SS              TT      LL          DD      DD
CC          OO      OO  NN  NN  NN  VV      VV  FFFFFFFF      SSSSSS        TT      LL          DD      DD
CC          OO      OO  NN  NN  NN  VV      VV  FFFFFFFF      SSSSSS        TT      LL          DD      DD
CC          OO      OO  NN    NNNN  VV      VV  FF                  SS      TT      LL          DD      DD
CC          OO      OO  NN    NNNN  VV      VV  FF                  SS      TT      LL          DD      DD
CC          OO      OO  NN      NN    VV  VV    FF                  SS      TT      LL          DD      DD
CC          OO      OO  NN      NN    VV  VV    FF                  SS      TT      LL          DD      DD    ....
  CCCCCCCC    OOOOOO    NN      NN      VV      FF          SSSSSSSS        TT      LLLLLLLLLL  DDDDDDDD      ....
  CCCCCCCC    OOOOOO    NN      NN      VV      FF          SSSSSSSS        TT      LLLLLLLLLL  DDDDDDDD      ....

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
```

```
0001  0 %TITLE  'VAX-11 CONVERT'
0002  0 MODULE  CONV$FSTLD        ( IDENT='V04-000',
0003  0                           OPTLEVEL=3
0004  0                           ) =
0005  0
0006  1 BEGIN
0007  1
0008  1 !****************************************************************************
0009  1 !*                                                                          *
0010  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0011  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0012  1 !*  ALL RIGHTS RESERVED.                                                    *
0013  1 !*                                                                          *
0014  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0015  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0016  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0017  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0018  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0019  1 !*  TRANSFERRED.                                                            *
0020  1 !*                                                                          *
0021  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0022  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0023  1 !*  CORPORATION.                                                            *
0024  1 !*                                                                          *
0025  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0026  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0027  1 !*                                                                          *
0028  1 !*                                                                          *
0029  1 !****************************************************************************
```

```
1 !++
1 !
1 ! Facility:      VAX-11 CONVERT
1 !
1 ! Abstract:      This module contains the high level calls for the fast load
1 !                process along with the declaratons for the data specifically
1 !                used by fast load
1 !
1 ! Contents:
1 !                FAST_LOAD
1 !                INIT_FAST_LOAD
1 !                LOAD_PRIMARY
1 !                LOAD_SECONDARY
1 !                LOAD_DATA_BUCKET
1 !                LOAD_INDEX_BUCKET
1 !                FINISH_INDEX
1 !                BACKUP_INDEX
1 !
1 ! Environment:
1 !
1 !                VAX/VMS Operating System
1 !
1 !--
1
1 !
1 ! Author:        Keith B Thompson         Creation date:  August-1980
1 !
1 !
1 ! Modified by:
1 !
1 !        V03-013 RAS0305          Ron Schaefer              7-May-1984
1 !                Fix check for maximum index level so that we report
1 !                an error rather than get an access violation if the
1 !                index level exceeds 31.
1 !
1 !        V03-012 JWT0177          Jim Teague               17-Apr-1984
1 !                CONVERT always tried to load a sidr bucket, even if
1 !                all records in the file had null keys for the
1 !                index, thereby corrupting the file.  Correct this
1 !                error by making sure at least one non-null key is
1 !                encountered for an index before allocating and
1 !                loading a SIDR bucket.
1 !
1 !        V03-011 JWT0143          Jim Teague               25-Nov-1983
1 !                CONVERT used to blindly add records until the fill
1 !                factor was exceeded.  Now, check to see if adding
1 !                a record will bring us closer to the fill factor.
1 !                If we're closer before the addition (even though
1 !                we may be short of the fill factor), then don't
1 !                add the record.
1 !
1 !        V03-010 KBT0476          Keith B. Thompson        29-Jan-1983
1 !                Add support for the ADD_KEY function
1 !
1 !        V03-009 KBT0459          Keith B. Thompson        10-Jan-1983
1 !                Fix a bug when loading p3 sidrs with no dups
1 !
```

```
1 !             V03-008 KBT0404         Keith B. Thompson       19-Nov-1982
1 !                     Fix some of the sidr code
1 !
1 !             V03-007 KBT0382         Keith B. Thompson       26-Oct-1982
1 !                     Add prologue 3 sidr support
1 !
1 !             V03-006 KBT0375         Keith B. Thompson       20-Oct-1982
1 !                     Check for keys out of order from split_data
1 !
1 !             V03-005 KBT0349         Keith B. Thompson       4-Oct-1982
1 !                     Use new linkage definitions
1 !
1 !             V03-004 KBT0050         Keith Thompson          10-May-1982
1 !                     Check for empty file before calling finish index
1 !
1 !             V03-003 KBT0047         Keith Thompson          14-Apr-1982
1 !                     Fix end condition problem with the index buckets
1 !
1 !             V03-002 KBT0022         Keith Thompson          24-Mar-1982
1 !                     Fix problem with last data bucket being continuation bucket
1 !                     and more duplicate problems.  Change some linkages.
1 !
1 !             V03-001 KBT0012         Keith Thompson          16-Mar-1982
1 !                     Fix some prologue 3 duplicate bugs in load_data_bucket
1 !                     and remove prologue 3 secondary key code
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$$GET_VM            : CL$GET_VM,
1               CONV$$GET_TEMP_VM       : CL$GET_TEMP_VM,
1               CONV$$FREE_TEMP_VM      : CL$FREE_TEMP_VM       NOVALUE,
1               CONV$$EXCEPTION,
1               CONV$$END_OF_FILE       : NOVALUE,
1               CONV$$SORT_SECONDARY    : CL$SORT_SECONDARY,
1               CONV$$GET_RECORD        : CL$GET_RECORD,
1               CONV$$CHECK_S_DUP       : CL$JSB_REG_9,
1               CONV$$CHECK_NULL        : CL$JSB_REG_9,
1               CONV$$SPLIT_DATA        : CL$JSB_REG_9,
1               CONV$$COMPRESS_KEY      : CL$JSB_REG_9  NOVALUE,
1               CONV$$COMPRESS_INDEX    : CL$JSB_REG_9  NOVALUE,
1               CONV$$MAKE_INDEX        : CL$JSB_REG_9  NOVALUE,
1               CONV$$WRITE_VBN         : CL$JSB_REG_9  NOVALUE,
1               CONV$$COPY_KEY          : CL$COPY_KEY   NOVALUE,
1               CONV$$WRITE_BUCKET      : CL$JSB_REG_9  NOVALUE,
1               CONV$$GET_BUCKET        : CL$JSB_REG_9  NOVALUE,
1               CONV$$INIT_BUCKET       : CL$JSB_REG_9  NOVALUE,
1               CONV$$CREATE_HIGH_KEY   : CL$JSB_REG_9  NOVALUE,
1               CONV$$WRITE_PROLOGUE    : NOVALUE,
1               CONV$$CONVERT_VBN_ID    : CL$CONVERT_VBN_ID     NOVALUE,
1               CONV$$SET_KEY_DESC      : CL$SET_KEY_DESC,
1               CONV$$GET_NEXT_KEY      : CL$GET_NEXT_KEY,
1               CONV$$WRITE_KEY_DESC    : CL$WRITE_KEY_DESC     NOVALUE;
1
1 FORWARD ROUTINE
1               CONV$$INIT_FAST_LOAD    : CL$INIT_FAST_LOAD     NOVALUE,
1               LOAD_PRIMARY            : CL$JSB_REG_9,
1               CONV$$LOAD_SECONDARY    : CL$LOAD_SECONDARY     NOVALUE,
1               LOAD_DATA_BUCKET        : CL$JSB_REG_8  NOVALUE,
1               LOAD_INDEX_BUCKET       : CL$JSB_REG_9  NOVALUE,
1               FINISH_INDEX            : CL$JSB_REG_9  NOVALUE,
1               BACKUP_INDEX            : CL$JSB_REG_9  NOVALUE;
1
1 EXTERNAL
1               CONV$GL_FILL            : LONG,
1
1               CONV$GW_OUT_REC_SIZ     : SIGNED WORD,          ! Output Rec. Size
1
1               CONV$GL_RECORD_COUNT,
1               CONV$GL_EXCEPT_COUNT,
1               CONV$GL_VALID_COUNT,
1
1               CONV$GW_MAX_REC_SIZ     : WORD,                 ! Aprox. size of record buffer
```

```
          CONV$GL_RFA_BUFFER,

          CONV$AB_IN_RAB            : $RAB_DECL,
          CONV$AB_OUT_XABSUM        : $XABSUM_DECL,
          CONV$AB_OUT_FAB           : $FAB_DECL,
          CONV$AB_OUT_RAB           : $RAB_DECL,
          CONV$AB_RFA_RAB           : $RAB_DECL,

          CONV$GL_EOF_VBN           : LONG,
          CONV$GB_PROL_V1           : BYTE,
          CONV$GB_PROL_V2           : BYTE,
          CONV$GB_PROL_V3           : BYTE,
          CONV$AR_PROLOGUE          : REF BLOCK [ ,BYTE ],
          CONV$AR_AREA_BLOCK        : REF BLOCKVECTOR [ ,AREA$C_BLN,BYTE ];

LITERAL
          FALSE = 0,
          TRUE  = 1;

MACRO
!         Some needed macros to define the data record for a bucket
!
          IRC$L_RRV_VBN    = 3,0,32,0%,      ! RRV VBN Pointer
          IRC$L_RRV_VBN3   = 5,0,32,0%,      ! RRV VBN Pointer (Prologue 3)
          IRC$W_VAR_SIZ    = 7,0,16,0%,      ! Var. Rec. Format Size field
          IRC$L_DUPCOUNT   = 2,0,32,0%,      ! Duplicate count field
          IRC$W_DUPSZ      = 6,0,16,0%,      ! Size field when dup. are allowed
          IRC$W_NODUPSZ    = 2,0,16,0%,      ! Size field when dup. are not allowed
          IRC$W_P3SZ       = 0,0,16,0%,      ! Size field for prologue 3 files

!         These macros make the code look a little better
!
          BKT$W_VBNFS      = .CONV$GW_VBN_FS_PTR,0,16,0%,  ! VBN Freespace Pointer in index level
          BKT$W_VBNFSO     = .CONV$GW_VBN_FS_PTRO,0,16,0%, ! VBN Freespace Pointer at the data level
          BKT$L_LCBPTR     = .CONV$GW_LCB_PTR,0,32,0%;     ! Last Contuation Bucket Pointer

!         Data Decl. for Fast Load routines
!
GLOBAL
          CONV$GL_RECORD_PTR        : LONG,          ! Pointer to record bffer

          CONV$GW_VBN_FS_PTR        : WORD,
          CONV$GW_VBN_FS_PTRO       : WORD,
          CONV$GW_LCB_PTR           : WORD,

          CONV$GL_CTX_BLOCK         : LONG,          ! Pointer to the contex block
          CONV$GL_DUP_BUF           : LONG;          ! Pointer to the Duplicate buffer

OWN
          CONTINUATION     : BYTE,           ! Continuation bucket
          DUPLICATE        : BYTE SIGNED,    ! Duplicate record

          SAVE_FREESPACE   : WORD,           ! Save pointer for backing up index
          SAVE_KEYFRESPC   : WORD,           !    ''
          SAVE_VBNFS       : WORD;           !    ''
```

```
1 %SBTTL  'FAST_LOAD'
1 GLOBAL ROUTINE  CONV$$FAST_LOAD : CL$JSB_REG_11 =
1 !++
1 !
1 ! Functional Description:
1 !
1 !       FAST_LOAD is the driving routine for the fast loading process.  It
1 !       will load the primary key then sort and load all secondary keys if
1 !       any.
1 !
1 ! Calling Sequence:
1 !
1 !       CONV$$FAST_LOAD()
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 !
1 !       RMS$_EOF or error code
1 !
1 ! Routines called:
1 !
1 !       CONV$$INIT_FAST_LOAD
1 !       LOAD_PRIMARY
1 !       CONV$$END_OF_FILE
1 !       CONV$$WRITE_PROLOGUE
1 !       CONV$$SET_KEY_DESC
1 !       CONV$$SORT_SECONDARY
1 !       CONV$$LOAD_SECONDARY
1 !       CONV$$WRITE_KEY_DESC
1 !
1 ! Side Effects:
1 !       none
1 !
1 !--
1
2     BEGIN
2
2     DEFINE_KEY_DESC;
2     DEFINE_CTX_GLOBAL;
2     DEFINE_BUCKET_GLOBAL;
2
2     ! Init the fast load process for all keys
2     !
2     CONV$$INIT_FAST_LOAD( 0 );
2
2     ! Load the primary data and index
```

```
2       !
2       RET_ON_ERROR( LOAD_PRIMARY() );
2
2       ! Write prologue
2       !
2       CONV$$WRITE_PROLOGUE();
2
2       ! Also write the key desc.
2       !
2       CONV$$WRITE_KEY_DESC();
2
2       ! Finish off the input file
2       !
2       CONV$$END_OF_FILE();
2
2       ! Free the space taken up by the loading
2       !
2       CONV$$FREE_TEMP_VM();
2
2       ! Process the secondary keys if there we records put into the
2       ! output file.
2       !
2       ! NOTE: This could cause secondary key indexes to be uninitialized.
2       ! At the moment RMS doesn't mind, if they ever do, something must be fixed.
2       !
2       IF .CONV$GL_VALID_COUNT GTRU 0
2       THEN
2
2           ! Loop for each key
2           !
2           WHILE CONV$$GET_NEXT_KEY()
2           DO
3               BEGIN
3
3               ! Set up the sort for the secondary key.  The sort is a INDEX sort.
3               ! This type of sort will produce a file of RFA's and keys of the
3               ! primary data level we just made.
3               !
3               RET_ON_ERROR( CONV$$SORT_SECONDARY() );
3
3               ! Now that the file is sorted get the data and load it in.
3               !
3               CONV$$LOAD_SECONDARY();
3
3               ! Write the prologue
3               !
3               CONV$$WRITE_PROLOGUE();
3
3               ! And the key descriptor
3               !
3               CONV$$WRITE_KEY_DESC();
3
3               ! Free the space taken up by the last key load
3               !
3               CONV$$FREE_TEMP_VM()
3
2               END;
```

```
;  343      0340  2
;  344      0341  2         RETURN RMS$_EOF
;  345      0342  2
;  346      0343  1         END;


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

                                                            .PSECT  _CONV$FAST_D,NOEXE,  PIC,2

                                                 00000 CONV$GL_RECORD_PTR::
                                                            .BLKB   4
                                                 00004 CONV$GW_VBN_FS_PTR::
                                                            .BLKB   2
                                                 00006 CONV$GW_VBN_FS_PTR0::
                                                            .BLKB   2
                                                 00008 CONV$GW_LCB_PTR::
                                                            .BLKB   2
                                                 0000A      .BLKB   2
                                                 0000C CONV$GL_CTX_BLOCK::
                                                            .BLKB   4
                                                 00010 CONV$GL_DUP_BUF::
                                                            .BLKB   4
                                                 00014 CONTINUATION:
                                                            .BLKB   1
                                                 00015 DUPLICATE:
                                                            .BLKB   1
                                                 00016 SAVE_FREESPACE:
                                                            .BLKB   2
                                                 00018 SAVE_KEYFRESPC:
                                                            .BLKB   2
                                                 0001A SAVE_VBNFS:
                                                            .BLKB   2

                                                            .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$$GET_VM, CONV$$GET_TEMP_VM
                                                      .EXTRN   CONV$$FREE_TEMP_VM
                                                      .EXTRN   CONV$$EXCEPTION
                                                      .EXTRN   CONV$$END_OF_FILE
                                                      .EXTRN   CONV$$SORT_SECONDARY
                                                      .EXTRN   CONV$$GET_RECORD
                                                      .EXTRN   CONV$$CHECK_S_DUP
                                                      .EXTRN   CONV$$CHECK_NULL
                                                      .EXTRN   CONV$$SPLIT_DATA
                                                      .EXTRN   CONV$$COMPRESS_KEY
                                                      .EXTRN   CONV$$COMPRESS_INDEX
                                                      .EXTRN   CONV$$MAKE_INDEX
                                                      .EXTRN   CONV$$WRITE_VBN
                                                      .EXTRN   CONV$$COPY_KEY, CONV$$WRITE_BUCKET
                                                      .EXTRN   CONV$$GET_BUCKET
                                                      .EXTRN   CONV$$INIT_BUCKET
                                                      .EXTRN   CONV$$CREATE_HIGH_KEY
                                                      .EXTRN   CONV$$WRITE_PROLOGUE
                                                      .EXTRN   CONV$$CONVERT_VBN_ID
                                                      .EXTRN   CONV$$SET_KEY_DESC
                                                      .EXTRN   CONV$$GET_NEXT_KEY
                                                      .EXTRN   CONV$$WRITE_KEY_DESC
                                                      .EXTRN   CONV$GL_FILL, CONV$GW_OUT_REC_SIZ
                                                      .EXTRN   CONV$GL_RECORD_COUNT
                                                      .EXTRN   CONV$GL_EXCEPT_COUNT
                                                      .EXTRN   CONV$GL_VALID_COUNT
                                                      .EXTRN   CONV$GW_MAX_REC_SIZ
                                                      .EXTRN   CONV$GL_RFA_BUFFER
                                                      .EXTRN   CONV$AB_IN_RAB, CONV$AB_OUT_XABSUM
                                                      .EXTRN   CONV$AB_OUT_FAB
                                                      .EXTRN   CONV$AB_OUT_RAB
                                                      .EXTRN   CONV$AB_RFA_RAB
                                                      .EXTRN   CONV$GL_EOF_VBN
                                                      .EXTRN   CONV$GB_PROL_V1
                                                      .EXTRN   CONV$GB_PROL_V2
                                                      .EXTRN   CONV$GB_PROL_V3
                                                      .EXTRN   CONV$AR_PROLOGUE
                                                      .EXTRN   CONV$AR_AREA_BLOCK

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

               7E              59  7D 00000 CONV$$FAST_LOAD::
                                                      MOVQ     R9, -(SP)                                          ; 0227
                               7E  D4 00003           CLRL     -(SP)                                              ; 0280
                            0000V  30 00005           BSBW     CONV$$INIT_FAST_LOAD                               ;
               5E              04  C0 00008           ADDL2    #4, SP                                             ;
                            0000V  30 0000B           BSBW     LOAD_PRIMARY                                       ; 0284
               39              50  E9 0000E           BLBC     STATUS, 3$                                         ;
      0000G    CF              00  FB 00011           CALLS    #0, CONV$$WRITE_PROLOGUE                           ; 0288
                            0000G  30 00016           BSBW     CONV$$WRITE_KEY_DESC                               ; 0292
      0000G    CF              00  FB 00019           CALLS    #0, CONV$$END_OF_FILE                              ; 0296
                            0000G  30 0001E           BSBW     CONV$$FREE_TEMP_VM                                 ; 0300
                     0000G     CF  D5 00021           TSTL     CONV$GL_VALID_COUNT                                ; 0308
                               1C  13 00025           BEQL     2$                                                 ;
```

```
                                 0000G 30 00027 1$:    BSBW    CONV$$GET_NEXT_KEY                        ; 0313
                       16           50 E9 0002A        BLBC    R0, 2$                                    ;
                                 0000G 30 0002D        BSBW    CONV$$SORT_SECONDARY                      ; 0321
                       17           50 E9 00030        BLBC    STATUS, 3$                                ;
                                 0000V 30 00033        BSBW    CONV$$LOAD_SECONDARY                      ; 0325
                 0000G CF           00 FB 00036        CALLS   #0, CONV$$WRITE_PROLOGUE                  ; 0329
                                 0000G 30 0003B        BSBW    CONV$$WRITE_KEY_DESC                      ; 0333
                                 0000G 30 0003E        BSBW    CONV$$FREE_TEMP_VM                        ; 0337
                                    E4 11 00041        BRB     1$                                        ;
                       50 0001827A  8F D0 00043 2$:    MOVL    #98938, R0                                ; 0341
                       59           8E 7D 0004A 3$:    MOVQ    (SP)+, R9                                 ; 0343
                                       05 0004D        RSB                                               ;

; Routine Size:  78 bytes,    Routine Base:  _CONV$FAST_S + 0000
```

```
1 %SBTTL  'INIT_FAST_LOAD'
1 GLOBAL ROUTINE  CONV$$INIT_FAST_LOAD ( MAX_KEY ) : CL$INIT_FAST_LOAD NOVALUE =
1 !++
1 !
1 ! Functional Description:
1 !
1 !       Initialize the fast load process.  Get memory for buffers and set up
1 !       pointers.  There are up to 3 pointers to record buffers at each level
1 !       RCP, RDP and LKB for level 0 and prologue 3 files.  The RCP, LKP and
1 !       RDP for all but level 0 pointers are set here.  The proper sizes are:
1 !
1 !                                  -------------------------------------------
1 !       LEVEL 0         RCP --->!         max_key + 13                     !
1 !                                  -------------------------------------------
1 !
1 !                                  -------------------------------------------
1 !       LEVEL 1+        RCP --->!      5                                   !
1 !                                  -------------------------------------------
1 !
1 !                                  -------------------------------------------
1 !                       RDP --->!         max_key + 2                      !
1 !                                  -------------------------------------------
1 !
1 !                                  -------------------------------------------
1 !       ALL LEVELS      LKP --->!         max_key                          !
1 !                                  -------------------------------------------
1 !
1 !       The RDP for level 0 is set in load_primary and load_secondary.
1 !
1 ! Calling Sequence:
1 !
1 !       INIT_FAST_LOAD();
1 !
1 ! Input Parameters:
1 !       none
1 !
1 ! Implicit Inputs:
1 !       none
1 !
1 ! Output Parameters:
1 !       none
1 !
1 ! Implicit Outputs:
1 !       none
1 !
1 ! Routine Value:
1 !       none
1 !
1 ! Routines Called:
1 !
1 !       CONV$$SET_KEY_DESC
1 !       CONV$$GET_NEXT_KEY
1 !       CONV$GET_VM
1 !
1 ! Side Effects:
1 !
1 !       Sets the end of file VBN pointer.  Allocates memory for record buffers.
```

```
1 !         Sets up the record data pointers, record control pointers and last key
1 !         pointers.
1 !
1 !--
1
2       BEGIN
2
2       DEFINE_CTX;
2       DEFINE_BUCKET;
2       DEFINE_KEY_DESC;
2
2       ! Since we are doing block IO we dont need the XABs anymore
2       ! If they are keep around area xabs (if any) will override the fab during
2       ! an extend a screw everything up.
2       !
2       CONV$AB_OUT_FAB [ FAB$L_XAB ] = 0;
2
2       ! Find the end of file VBN. In a new file it the one block past the last
2       ! allocated area (the last area may not be allocated therefore look at
2       ! one)
2       !
3       BEGIN                   ! HIGH_VBN local
3
3       LOCAL           HIGH_VBN;
3
3       HIGH_VBN = 0;
3
3       INCR AREA FROM 0 TO ( .CONV$AB_OUT_XABSUM [ XAB$B_NOA ] - 1 ) BY 1
3       DO
3
3           ! If the current extent starts at a higher VBN then the last one
3           ! us this one to find the end of file
3           !
3           IF .CONV$AR_AREA_BLOCK [ .AREA,AREA$L_CVBN ] GTR .HIGH_VBN
3           THEN
3
3               ! The end of file is this the start of this extent plus the number
3               ! of blocks in the extent
3               !
3               CONV$GL_EOF_VBN = .CONV$AR_AREA_BLOCK [ .AREA,AREA$L_CVBN ] +
3                                         .CONV$AR_AREA_BLOCK [ .AREA,AREA$L_CNBLK ];
3
2       END;                    ! HIGH_VBN local
2
2       ! Calculate the max space needed for index key buffers and init. the contex
2       ! block.  If it was not given.
2       !
2       IF .MAX_KEY EQLU 0
2       THEN
3           BEGIN
3
3           ! The max. size is the size of the logest key.
3           ! So we check each key desc.
3           !
3           CONV$$SET_KEY_DESC( 0 );
3
3           DO
```

```
3
3               IF .KEY_DESC [ KEY$B_KEYSZ ] GTR .MAX_KEY
3               THEN
3                   MAX_KEY = .KEY_DESC [ KEY$B_KEYSZ ]
3
3           UNTIL NOT CONV$$GET_NEXT_KEY()
2           END;
2
3       BEGIN
3
3       LOCAL       BYTES;
3
3       ! Figure the total number of bytes. (SEE ABOVE)
3       !
3       BYTES = ( .MAX_KEY * ( MAX_IDX_LVL + 1 ) ) + ( MAX_IDX_LVL * 7 ) + 13;
3
3       ! For Prologue 3 files we may need the last key buffers
3       !
3       IF .CONV$GB_PROL_V3
3       THEN
3           BYTES = .BYTES + ( .MAX_KEY * ( MAX_IDX_LVL - 1 ) );
3
3       ! Add the space for the contex block
3       !
3       BYTES = .BYTES + ( MAX_IDX_LVL * CTX$K_BLN );
3
3       ! Get the zero filled space
3       !
3       CONV$GL_CTX_BLOCK = CONV$$GET_VM ( .BYTES )
3
2       END;
2
2       ! Set all of the record control pointers and record data pointers for
2       ! level one (1) and above.
2       !
2       CTX = .CONV$GL_CTX_BLOCK;
2
2       CTX [ CTX$L_RCP ] = .CTX + ( MAX_IDX_LVL * CTX$K_BLN );
2
3       BEGIN       ! BUFFER_OFFSET local
3
3       LOCAL       BUFFER_OFFSET;
3
3       BUFFER_OFFSET = .CTX [ CTX$L_RCP ] + .MAX_KEY + 13;
3
3       INCR I FROM 1 TO ( MAX_IDX_LVL - 1 ) BY 1
3       DO
4           BEGIN
4           CTX = .CTX + CTX$K_BLN;
4           CTX [ CTX$B_LEVEL ] = .I;
4           CTX [ CTX$L_RCP ] = .BUFFER_OFFSET;
4           CTX [ CTX$L_RDP ] = .BUFFER_OFFSET + 5;
4           BUFFER_OFFSET = .BUFFER_OFFSET + .MAX_KEY + 7
3           END;
3
3       ! Set up the last key buffer for level 0
3       !
```

```
;  519        0515  3      CTX = .CONV$GL_CTX_BLOCK;
;  520        0516  3      CTX [ CTX$L_LKP ] = .BUFFER_OFFSET;
;  521        0517  3
;  522        0518  3      ! Set up the last key buffer if necessary for levels 1 and above
;  523        0519  3      !
;  524        0520  3      IF .CONV$GB_PROL_V3
;  525        0521  3      THEN
;  526        0522  3          INCR I FROM 1 TO ( MAX_IDX_LVL - 1 ) BY 1
;  527        0523  3          DO
;  528        0524  4              BEGIN
;  529        0525  4              CTX = .CTX + CTX$K_BLN;
;  530        0526  4              BUFFER_OFFSET = .BUFFER_OFFSET + .MAX_KEY;
;  531        0527  4              CTX [ CTX$L_LKP ] = .BUFFER_OFFSET
;  532        0528  4              END
;  533        0529  4
;  534        0530  2      END;          ! BUFFER_OFFSET local
;  535        0531  2
;  536        0532  2      RETURN
;  537        0533  2
;  538        0534  1      END;




                                             1C  BB 00000 CONV$$INIT_FAST_LOAD::
                                                                  PUSHR    #^M<R2,R3,R4>                              ; 0345
                                     0000G   CF  D4 00002         CLRL     CONV$AB_OUT_FAB+36                         ; 0416
                                             54  D4 00006         CLRL     HIGH_VBN                                   ; 0426
                              53     0000G   CF  9A 00008         MOVZBL   CONV$AB_OUT_XABSUM+8, R3                   ; 0428
                              50        01   CE  0000D            MNEGL    #1, AREA                                   ; 0434
                                             1D  11 00010         BRB      2$                                         :
                    52        50        06   78 00012 1$:         ASHL     #6, AREA, R2                               :
                    51        52     0000G   CF  C1 00016         ADDL3    CONV$AR_AREA_BLOCK, R2, R1                 :
                              54        0C   A1  D1 0001C         CMPL     12(R1), HIGH_VBN                           :
                                             0D  15 00020         BLEQ     2$                                         :
                              52     0000G   CF  C0 00022         ADDL2    CONV$AR_AREA_BLOCK, R2                     ; 0441
         0000G    CF     0C   A1        10   A2  C1 00027         ADDL3    16(R2), 12(R1), CONV$GL_EOF_VBN            :
                  DF          50             53  F2 0002F 2$:     AOBLSS   R3, AREA, 1$                               ; 0434
                                        10   AE  D5 00033         TSTL     MAX_KEY                                    ; 0448
                                             1C  12 00036         BNEQ     5$                                         :
                                             7E  D4 00038         CLRL     -(SP)                                      ; 0455
                                          0000G  30 0003A         BSBW     CONV$$SET_KEY_DESC                         :
                              5E             04  C0 0003D         ADDL2    #4, SP                                     :
  10     AE       14     AB   08             00  ED 00040 3$:     CMPZV    #0, #8, 20(KEY_DESC), MAX_KEY              ; 0459
                                             05  15 00047         BLEQ     4$                                         :
                       10     AE        14   AB  9A 00049         MOVZBL   20(KEY_DESC), MAX_KEY                      ; 0461
                                          0000G  30 0004E 4$:     BSBW     CONV$$GET_NEXT_KEY                         ; 0463
                              EC             50  E8 00051         BLBS     R0, 3$                                     :
                              52        10   AE  D0 00054 5$:     MOVL     MAX_KEY, R2                                ; 0472
                    51        52             21  C5 00058         MULL3    #33, R2, R1                                :
                              50      00ED   C1  9E 0005C         MOVAB    237(R1), BYTES                             :
                              07     0000G   CF  E9 00061         BLBC     CONV$GB_PROL_V3, 6$                        ; 0476
                    51        52             1F  C5 00066         MULL3    #31, R2, R1                                ; 0478
                              50             51  C0 0006A         ADDL2    R1, BYTES                                  :
                              50      0B80   C0  9E 0006D 6$:     MOVAB    2944(R0), BYTES                            ; 0482
                                             50  DD 00072         PUSHL    BYTES                                      ; 0486
```

```
                                    0000G  30 00074         BSBW    CONV$$GET_VM                             ;
                   5E                  04  C0 00077         ADDL2   #4, SP                                   ;
            0000'  CF                  50  D0 0007A         MOVL    R0, CONV$GL_CTX_BLOCK                    ;
                   5A        0000'     CF  D0 0007F         MOVL    CONV$GL_CTX_BLOCK, CTX                   ; 0493
              30   AA        0B80      CA  9E 00084         MOVAB   2944(R10), 48(CTX)                       ; 0495
      50           52          30      AA  C1 0008A         ADDL3   48(CTX), R2, R0                          ; 0501
                   50                  0D  C0 0008F         ADDL2   #13, BUFFER_OFFSET                       ;
                   51                  01  D0 00092         MOVL    #1, I                                    ; 0503
                   5A          5C      AA  9E 00095 7$:     MOVAB   92(R10), CTX                             ; 0506
              02   AA                  51  90 00099         MOVB    I, 2(CTX)                                ; 0507
              30   AA                  50  D0 0009D         MOVL    BUFFER_OFFSET, 48(CTX)                   ; 0508
              34   AA          05      A0  9E 000A1         MOVAB   5(R0), 52(CTX)                           ; 0509
                   50          07    A240  9E 000A6         MOVAB   7(R2)[BUFFER_OFFSET], BUFFER_OFFSET      ; 0510
      E6           51                  1F  F3 000AB         AOBLEQ  #31, I, 7$                               ;
                   5A        0000'     CF  D0 000AF         MOVL    CONV$GL_CTX_BLOCK, CTX                   ; 0515
              3C   AA                  50  D0 000B4         MOVL    BUFFER_OFFSET, 60(CTX)                   ; 0516
                   12        0000G     CF  E9 000B8         BLBC    CONV$GB_PROL_V3, 9$                      ; 0520
                   51                  01  D0 000BD         MOVL    #1, I                                    ; 0522
                   5A          5C      AA  9E 000C0 8$:     MOVAB   92(R10), CTX                             ; 0525
                   50                  52  C0 000C4         ADDL2   R2, BUFFER_OFFSET                        ; 0526
              3C   AA                  50  D0 000C7         MOVL    BUFFER_OFFSET, 60(CTX)                   ; 0527
      F1           51                  1F  F3 000CB         AOBLEQ  #31, I, 8$                               ;
                                       1C  BA 000CF 9$:     POPR    #^M<R2,R3,R4>                            ; 0534
                                       05     000D1         RSB                                              ;

; Routine Size:  210 bytes,    Routine Base:  _CONV$FAST_S + 004E
```

```
1 %SBTTL  'LOAD_PRIMARY'
1 ROUTINE LOAD_PRIMARY : CL$JSB_REG_9 =
1 !++
1 !
1 ! Functional Description:
1 !
1 !         Loads the primary key of a index sequential file.
1 !
1 ! Calling Sequence:
1 !
1 !         LOAD_PRIMARY()
1 !
1 ! Input Parameters:
1 !         none
1 !
1 ! Implicit Inputs:
1 !         none
1 !
1 ! Output Parameters:
1 !         none
1 !
1 ! Implicit Outputs:
1 !         none
1 !
1 ! Routine Value:
1 !
1 !         RMS$_EOF or error codes
1 !
1 ! Routine Called:
1 !
1 !         CONV$$SET_KEY_DESC
1 !         CONV$$GET_TEMP_VM
1 !         CONV$$GET_BUCKET
1 !         CONV$$GET_RECORD
1 !         CONV$$EXCEPTION
1 !         CONV$$SPLIT_DATA
1 !         LOAD_DATA_BUCKET
1 !         FINISH_INDEX
1 !
1 ! Side Effects:
1 !
1 !         Loads primary key
1 !
1 !--
1
2     BEGIN
2
2     LABEL
2         DUP_BLK;
2
2     DEFINE_CTX;
2     DEFINE_BUCKET;
2     DEFINE_KEY_DESC;
2
2     CTX = .CONV$GL_CTX_BLOCK;
2
2     ! Set key to the primary index
```

```
2       !
2       CONV$$SET_KEY_DESC( 0 );
2
2       ! Errors on the rab from now on are WRITEERRs
2       !
2       CONV$AB_OUT_RAB [ RAB$L_CTX ] = CONV$_WRITEERR;
2
2       ! For prologue 3 files we need an extra buffer for the data record
2       ! Else we let the REC_DATA_PTR point to the user buffer of the output rab
2       !
2       IF .CONV$GB_PROL_V3
2       THEN
3           BEGIN
3
3           LOCAL   BYTES;
3
3           ! The worst case is fully non compressed record with compression info
3           !
3           BYTES = .CONV$GW_MAX_REC_SIZ + 3;
3
3           ! Get the space for the data buffer
3           !
3           ! Record data pointer at level 0 will point to the new buffer
3           !
3           CTX [ CTX$L_RDP ] = CONV$$GET_TEMP_VM ( .BYTES )
3
3           END
2       ELSE
2
2           ! Record data pointer at level 0 points to Record Ptr
2           !
2           CTX [ CTX$L_RDP ] = .CONV$GL_RECORD_PTR;
2
2       ! Get the Buckets for the data area and at least the first level of the index
2       !
2       ! Get the bucket for level 0
2       !
2       CONV$$GET_BUCKET( .KEY_DESC [ KEY$B_DANUM ] );
2
2       KEY_DESC [ KEY$L_LDVBN ] = .CTX [ CTX$L_CURRENT_VBN ];
2
2       ! Get the bucket for level 1
2       !
2       CTX = .CTX + CTX$K_BLN;
2       CONV$$GET_BUCKET( .KEY_DESC [ KEY$B_LANUM ] );
2
2       CTX = .CONV$GL_CTX_BLOCK;
2
2       ! For the primary key the Data comes from GET_RECORD. NOTE: Don't use the
2       ! UBF of the input RAB since some record conversion may be done. Also note
2       ! the RBF pointer of the output RAB is destroyed after the first call to
2       ! WRITE_BUCKET but it is ok to use it now.
2       !
3       BEGIN
3
3       DEFINE_RECORD_CTRL_GLOBAL;
3
```

```
3      LOCAL
3          STATUS;
3
3      RECORD_CTRL = .CTX [ CTX$L_RCP ];
3
3      ! Main record processing loop.  The call to GET_RECORD does any record format
3      ! processing and exception handling before it returns.  The size of the record
3      ! is passed back by OUT_REC_SIZ.
3      !
4      WHILE ( STATUS = CONV$$GET_RECORD() )
3      DO
4          BEGIN               ! Main Loop
4
4  DUP_BLK:
5          BEGIN               ! DUP_BLK Primary duplicate block
5
5              ! If the record is shorter the minium record length of the primary key
5              ! cause an exception
5
5          IF .CONV$GW_OUT_REC_SIZ LSS .KEY_DESC [ KEY$W_MINRECSZ ]
5          THEN
6              BEGIN
6
6              LOCAL STATUS;
6
6              ! If it was not fatal continue else exit
6              !
6              IF STATUS = CONV$$EXCEPTION( CONV$_RSK )
6              THEN
6                  LEAVE DUP_BLK
6              ELSE
6                  RETURN .STATUS
5              END;
5
5          ! Seperate the key from the data record if necessary and do
5          ! data compression if necessary also check if this is a duplicate
5          ! or the key is out of order
5          !
5          DUPLICATE = CONV$$SPLIT_DATA();
5
5          ! If out of order, i.e. duplicate = -1, signal exception and continue
5          !
5          IF .DUPLICATE LSS 0
5          THEN
6              BEGIN
6
6              LOCAL       STATUS;
6
6              ! If not fatal exception then continue else bomb
6              !
6              IF STATUS = CONV$$EXCEPTION ( CONV$_SEQ )
6              THEN
6                  LEAVE DUP_BLK
6              ELSE
6                  RETURN .STATUS
6
5              END;
```

```
5
5          ! If we got a dup and we dont allow dups then cause an exception
5          !
6          IF .DUPLICATE AND ( NOT .KEY_DESC [ KEY$V_DUPKEYS ] )
5          THEN
6              BEGIN
6
6              LOCAL       STATUS;
6
6              ! If not fatal exception then continue else bomb
6              !
6              IF STATUS = CONV$$EXCEPTION ( CONV$_DUP )
6              THEN
6                  LEAVE DUP_BLK
6              ELSE
6                  RETURN .STATUS
6
5              END;
5
5          ! Set up the control byte for the record
5          !
5          RECORD_CTRL [ IRC$B_CONTROL ] = 2;
5
5          ! Set the size field int the record
5          !
5          IF .CONV$GB_PROL_V3
5          THEN
6              BEGIN
6
6              ! A small non compressed fixed length record has no size field
6              !
6              IF .KEY_DESC [ KEY$V_REC_COMPR ] OR
6                  .KEY_DESC [ KEY$V_KEY_COMPR ] OR
7                  ( .CONV$AB_OUT_FAB [ FAB$B_RFM ] EQLU FAB$C_VAR )
6              THEN
6                  RECORD_CTRL [ 9,0,16,0 ] = .CTX [ CTX$W_RCS ] +
6                                                      .CTX [ CTX$W_RDS ] - 11
6              END
5          ELSE
5
5              ! Set up the record size for var. length records
5              ! for prologue 1 and 2 files
5              !
5              IF .CONV$AB_OUT_FAB [ FAB$B_RFM ] EQLU FAB$C_VAR
5              THEN
5                  RECORD_CTRL [ IRC$W_VAR_SIZ ] = .CONV$GW_OUT_REC_SIZ;
5
5          ! If we are in a continuation bucket and the current record is NOT a
5          ! duplicate then write the current bucket out and start a new one
5          !
5          ! For optimumization do the continuation check first
5
5          IF .CONTINUATION THEN IF NOT .DUPLICATE
5          THEN
6              BEGIN
6
6              CONV$$WRITE_BUCKET();
```

```
;  768        0763  6
;  769        0764  6                   CONV$$INIT_BUCKET();
;  770        0765  6
;  771        0766  6                   ! Continuation no longer need be set.  (the next record will always
;  772        0767  6                   ! fix into the new bucket
;  773        0768  6                   !
;  774        0769  6                   CONTINUATION = _CLEAR
;  775        0770  6
;  776        0771  5                   END;
;  777        0772  5
;  778        0773  5               ! Load the record
;  779        0774  5               !
;  780        0775  5               LOAD_DATA_BUCKET()
;  781        0776  5
;  782        0777  5               END              ! DUP_BLK Primary duplicate block
;  783        0778  3               END;             ! Main loop
;  784        0779  3
;  785        0780  3           ! If we exited because of end of file and there are records in the file
;  786        0781  3           ! then finish off the index
;  787        0782  3           !
;  788        0783  3           IF ( .STATUS EQLU RMS$_EOF ) AND
;  789        0784  4                           ( .CONV$GL_RECORD_COUNT NEQU .CONV$GL_EXCEPT_COUNT )
;  790        0785  3           THEN
;  791        0786  3               FINISH_INDEX();
;  792        0787  3
;  793        0788  3           RETURN CONV$_SUCCESS
;  794        0789  3
;  795        0790  3           END
;  796        0791  1           END;



                           0104    8F   BB 00000 LOAD_PRIMARY:
                                                          PUSHR    #^M<R2,R8>                                  ; 0536
                 5A       0000'    CF   D0 00004          MOVL     CONV$GL_CTX_BLOCK, CTX                      ; 0589
                                   7E   D4 00009          CLRL     -(SP)                                       ; 0593
                             0000G 30 0000B               BSBW     CONV$$SET_KEY_DESC                          ;
                 5E                04   C0 0000E          ADDL2    #4, SP                                      ;
       0000G     CF 00000000G      8F   D0 00011          MOVL     #CONV$_WRITEERR, CONV$AB_OUT_RAB+24         ; 0597
                 16       0000G    CF   E9 0001A          BLBC     CONV$GB_PROL_V3, 1$                         ; 0602
                 50       0000G    CF   3C 0001F          MOVZWL   CONV$GW_MAX_REC_SIZ, BYTES                  ; 0610
                 50                03   C0 00024          ADDL2    #3, BYTES                                   ;
                                   50   DD 00027          PUSHL    BYTES                                       ; 0616
                             0000G 30 00029               BSBW     CONV$$GET_TEMP_VM                           ;
                 5E                04   C0 0002C          ADDL2    #4, SP                                      ;
         34      AA                50   D0 0002F          MOVL     R0, 52(CTX)                                 ;
                                   06   11 00033          BRB      2$                                          ;
         34      AA       0000'    CF   D0 00035 1$:      MOVL     CONV$GL_RECORD_PTR, 52(CTX)                 ; 0623
                 7E         08     AB   9A 0003B 2$:      MOVZBL   8(KEY_DESC), -(SP)                          ; 0629
                             0000G 30 0003F               BSBW     CONV$$GET_BUCKET                            ;
         54      AB         08     AA   D0 00042          MOVL     8(CTX), 84(KEY_DESC)                        ; 0631
                 5A         5C     AA   9E 00047          MOVAB    92(R10), CTX                                ; 0635
                 6E         07     AB   9A 0004B          MOVZBL   7(KEY_DESC), (SP)                           ; 0636
                             0000G 30 0004F               BSBW     CONV$$GET_BUCKET                            ;
                 5E                04   C0 00052          ADDL2    #4, SP                                      ;
```

```
                                   5A     0000'  CF  D0 00055          MOVL    CONV$GL_CTX_BLOCK, CTX                        ; 0638
                                   58        30  AA  D0 0005A          MOVL    48(CTX), RECORD_CTRL                          ; 0652
                                            0000G 30 0005E 3$:         BSBW    CONV$$GET_RECORD                              ; 0658
                                   52            50  D0 00061          MOVL    R0, STATUS                                    ;
                                   03            52  E8 00064          BLBS    STATUS, 4$                                    ;
                                               0091  31 00067          BRW     14$                                           ;
                                   50        16  AB  3C 0006A 4$:      MOVZWL  22(KEY_DESC), R0                              ; 0668
            50      0000G  CF                    10                    
                                                 00  EC 0006E          CMPV    #0, #16, CONV$GW_OUT_REC_SIZ, R0              ;
                                                 08  18 00075          BGEQ    5$                                            ;
                                        00000000G 8F  DD 00077          PUSHL   #CONV$_RSK                                    ; 0676
                                                 21  11 0007D          BRB     7$                                            ;
                                            0000G 30 0007F 5$:         BSBW    CONV$$SPLIT_DATA                              ; 0687
                          0000'  CF              50  90 00082          MOVB    R0, DUPLICATE                                 ;
                                                 08  18 00087          BGEQ    6$                                            ; 0691
                                        00000000G 8F  DD 00089          PUSHL   #CONV$_SEQ                                    ; 0699
                                                 0F  11 0008F          BRB     7$                                            ;
                                   14     0000'  CF  E9 00091 6$:      BLBC    DUPLICATE, 9$                                 ; 0709
                                   10        10  AB  E8 00096          BLBS    16(KEY_DESC), 9$                              ;
                                        00000000G 8F  DD 0009A          PUSHL   #CONV$_DUP                                    ; 0717
                          0000G  CF              01  FB 000A0 7$:      CALLS   #1, CONV$$EXCEPTION                           ;
                                 6B              50  E9 000A5          BLBC    STATUS, 16$                                   ;
                                                 B4  11 000A8 8$:      BRB     3$                                            ; 0719
                                 68              02  90 000AA 9$:      MOVB    #2, (RECORD_CTRL)                             ; 0727
                                 23       0000G  CF  E9 000AD          BLBC    CONV$GB_PROC_V3, 11$                          ; 0731
                                             10  AB  95 000B2          TSTB    16(KEY_DESC)                                  ; 0737
                                                 0C  19 000B5          BLSS    10$                                           ;
                   07        10  AB              06  E0 000B7          BBS     #6, 16(KEY_DESC), 10$                         ; 0738
                                 02       0000G  CF  91 000BC          CMPB    CONV$AB_OUT_FAB+31, #2                        ; 0739
                                                 1F  12 000C1          BNEQ    12$                                           ;
                                 50        38    AA  3C 000C3 10$:     MOVZWL  56(CTX), R0                                   ; 0742
                                 51        3A    AA  3C 000C7          MOVZWL  58(CTX), R1                                   ;
                                 50              51  C0 000CB          ADDL2   R1, R0                                        ;
            09    A8             50              0B  A3 000CE          SUBW3   #11, R0, 9(RECORD_CTRL)                       ;
                                                 0D  11 000D3          BRB     12$                                           ; 0733
                                 02       0000G  CF  91 000D5 11$:     CMPB    CONV$AB_OUT_FAB+31, #2                        ; 0749
                                                 06  12 000DA          BNEQ    12$                                           ;
                       07        A8       0000G  CF  B0 000DC          MOVW    CONV$GW_OUT_REC_SIZ, 7(RECORD_CTRL)           ; 0751
                                 0F       0000'  CF  E9 000E2 12$:     BLBC    CONTINUATION, 13$                             ; 0758
                                 0A       0000'  CF  E8 000E7          BLBS    DUPLICATE, 13$                                ;
                                            0000G 30 000EC             BSBW    CONV$$WRITE_BUCKET                            ; 0762
                                            0000G 30 000EF             BSBW    CONV$$INIT_BUCKET                             ; 0764
                                          0000'  CF  94 000F2          CLRB    CONTINUATION                                  ; 0769
                                            0000V 30 000F6 13$:        BSBW    LOAD_DATA_BUCKET                              ; 0775
                                                 AD  11 000F9          BRB     8$                                            ; 0660
                     0001827A    8F              52  D1 000FB 14$:     CMPL    STATUS, #98938                                ; 0783
                                                 0C  12 00102          BNEQ    15$                                           ;
                          0000G  CF       0000G  CF  D1 00104          CMPL    CONV$GL_RECORD_COUNT, CONV$GL_EXCEPT_COUNT    ; 0784
                                                 03  13 0010B          BEQL    15$                                           ;
                                            0000V 30 0010D             BSBW    FINISH_INDEX                                  ; 0786
                                 50              01  D0 00110 15$:     MOVL    #1, R0                                        ; 0788
                                         0104    8F  BA 00113 16$:     POPR    #^M<R2,R8>                                    ; 0791
                                                 05  00117             RSB                                                   ;

; Routine Size:  280 bytes,    Routine Base:  _CONV$FAST_S + 0120


;  797          0792  1
```

```
1 %SBTTL  'LOAD_SECONDARY'
1 GLOBAL ROUTINE  CONV$$LOAD_SECONDARY : CL$LOAD_SECONDARY NOVALUE =
1 !++
1 !
1 ! Functional Description:
1 !
1 !          Loads a secondary key of an index sequential file.  Which secondary
1 !          index depends on KEY_REF.  The secondary
1 !          data records are read from the RFA file created and opened by
1 !          SORT_SECONDARY.  NOTE:  The overall operation of LOAD_SECONDARY is
1 !          fundementally different then LOAD_PRIMARY.
1 !
1 ! Calling Sequence:
1 !
1 !          CONV$$LOAD_SECONDARY();
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 !
1 !          RMS$_EOF or error codes
1 !
1 ! Routines Called:
1 !
1 !          CONV$$GET_BUCKET
1 !          CONV$$GET_TEMP_VM
1 !          $GET
1 !          CONV$$CHECK_NULL
1 !          CONV$$CHECK_S_DUP
1 !          LOAD_DATA_BUCKET
1 !          CONV$$COPY_KEY
1 !          CONV$$WRITE_BUCKET
1 !          CONV$$INIT_BUCKET
1 !          CONV$$CONVERT_VBN_ID
1 !          FINISH_INDEX
1 !
1 ! Side Effects:
1 !
1 !          Loads secondary index defined by KEY_REF
1 !
1 !--
1
2     BEGIN
2
2     DEFINE_CTX;
2     DEFINE_BUCKET;
2     DEFINE_KEY_DESC;
```

```

LABEL
    NULL_BLK;

LOCAL
    DUP_COUNT,
    MAX_NUM_DUP;

! Init some values
!
CONTINUATION = _CLEAR;
DUPLICATE = _CLEAR;

! Errors on the rab from now on are WRITEERRs
!
CONV$AB_OUT_RAB [ RAB$L_CTX ] = CONV$_WRITEERR;

! Point to the first block
!
CTX = .CONV$GL_CTX_BLOCK;

! Get the Buckets for the secondary data area and at least the
! first level of the index
!
! Get the bucket for level 0
!
CONV$$GET_BUCKET( .KEY_DESC [ KEY$B_DANUM ] );

KEY_DESC [ KEY$L_LDVBN ] = .CTX [ CTX$L_CURRENT_VBN ];

! Get the bucket for level 1
!
CTX = .CTX + CTX$K_BLN;
CONV$$GET_BUCKET( .KEY_DESC [ KEY$B_LANUM ] );

CTX = .CONV$GL_CTX_BLOCK;

! Before we start we need to calcluate the size of the level 0 index record
! buffer.  This calculation is VERY important it must be very accurate!
!
! If we allow dup. keys the it becomes complicated
!
! Find out the max. number of duplicates that can fit in this bucket
!
IF .KEY_DESC [ KEY$V_DUPKEYS ]
THEN

    ! Sizes are different for prologue 3
    !
    IF .CONV$GB_PROL_V3
    THEN

        ! For compression it is also different
        !
        IF .KEY_DESC [ KEY$V_IDX_COMPR ]
        THEN

```

```
;  913        0907  2                     ! The space in the bucket minus the key size and the record
;  914        0908  2                     !       overhead (2+2) divided by the size of the SIDR record
;  915        0909  2                     !       pointer (7)
;  916        0910  2                     !
;  917        0911  3                     MAX_NUM_DUP = ( .CTX [ CTX$W_SPC ] -
;  918        0912  2                                         ( .KEY_DESC [ KEY$B_KEYSZ ] + 4 ) ) / 7
;  919        0913  2                 ELSE
;  920        0914  2
;  921        0915  2                     ! The space in the bucket minus the key size and the record
;  922        0916  2                     !       overhead (2) divided by the size of the SIDR record
;  923        0917  2                     !       pointer (7)
;  924        0918  2                     !
;  925        0919  3                     MAX_NUM_DUP = ( .CTX [ CTX$W_SPC ] -
;  926        0920  2                                         ( .KEY_DESC [ KEY$B_KEYSZ ] + 2 ) ) / 7
;  927        0921  2
;  928        0922  2             ELSE
;  929        0923  2
;  930        0924  2                 ! The space in the bucket minus the key size and the record
;  931        0925  2                 !           overhead (8) divided by the size of the SIDR record
;  932        0926  2                 !           pointer (6)
;  933        0927  2                 !
;  934        0928  3                 MAX_NUM_DUP = ( .CTX [ CTX$W_SPC ] -
;  935        0929  2                                         ( .KEY_DESC [ KEY$B_KEYSZ ] + 8 ) ) / 6
;  936        0930  2         ELSE
;  937        0931  2             MAX_NUM_DUP = 1;
;  938        0932  2
;  939        0933  3         BEGIN
;  940        0934  3
;  941        0935  3         LOCAL       BYTES;
;  942        0936  3
;  943        0937  3         ! The size of the level 0 buffer consist of:
;  944        0938  3         !
;  945        0939  3         !   Space for all RRVs (one for each dup) : Largest rrv - prologue 3, 7 bytes
;  946        0940  3         !   Overhead : Maximun overhead - prologue 1, 8 bytes
;  947        0941  3         !
;  948        0942  3         ! We also need a duplicate buffer for things which is the size of the key
;  949        0943  3         !
;  950        0944  3         BYTES = ( .MAX_NUM_DUP * 7 ) + 8 + .KEY_DESC [ KEY$B_KEYSZ ];
;  951        0945  3
;  952        0946  3         ! Allocate the memory for the buffer
;  953        0947  3         ! The level 0 data record pointers points to this buffer
;  954        0948  3         !
;  955        0949  3         CTX [ CTX$L_RDP ] = CONV$$GET_TEMP_VM ( .BYTES );
;  956        0950  3
;  957        0951  3         ! The duplicate buffer is just past that
;  958        0952  3         !
;  959        0953  3         CONV$GL_DUP_BUF = .CTX [ CTX$L_RDP ] + ( .MAX_NUM_DUP * 7 ) + 8
;  960        0954  3
;  961        0955  2         END;
;  962        0956  2
;  963        0957  2         ! For the secondary key the Data comes from $GET on the RFA RAB
;  964        0958  2         !
;  965        0959  3         BEGIN                   ! RECORD_CTRL local
;  966        0960  3
;  967        0961  3         DEFINE_RECORD_CTRL_GLOBAL;
;  968        0962  3
;  969        0963  3         LOCAL
```

```
3                  ALL_NULL,
3                  SKIP,
3                  STATUS;
3
3          SKIP = _CLEAR;
3
3          RECORD_CTRL = .CTX [ CTX$L_RCP ];
3
3          ALL_NULL = _SET;      ! Could be nothing but null keys, you know...
3
3          ! Main record processing loop. The size of the record is returned in
3          ! RFA_RAB [ RAB$W_RSZ ]
3          !
4          WHILE ( STATUS = $GET( RAB=CONV$AB_RFA_RAB ) )
3          DO
4              BEGIN              ! Main Loop
4
4  NULL_BLK:
5              BEGIN              ! NULL_BLK null key value block
5
5              LOCAL    DUP;
5
5              ! If the record is too short (does not contain the complete key)
5              ! then treat it as a null key
5              !
5              IF ( .CONV$AB_RFA_RAB [ RAB$W_RSZ ] - 6 ) LSSU .KEY_DESC [ KEY$B_KEYSZ ]
5              THEN
5                  LEAVE NULL_BLK;
5
5              ! If the file allows null keys check to see if this is one
5              !
5              IF .KEY_DESC [ KEY$V_NULKEYS ]
5              THEN
5
5                  ! If this is a null key then just ignore this record
5                  !
5                  IF CONV$$CHECK_NULL()
5                  THEN
5                      LEAVE NULL_BLK;
5
5              !
5              ! If we got a non-null key, then all_null can no longer be true
5              !
5              IF .ALL_NULL THEN ALL_NULL = _CLEAR;
5
5              ! Check to see if this is a duplicate.
5              !
5              DUP = CONV$$CHECK_S_DUP();
5
5              ! Process the key
5              !
5              IF .KEY_DESC [ KEY$V_DUPKEYS ]
5              THEN
6                  BEGIN
6
6                  ! If this was a dup
6                  !
```

```
IF .DUP
THEN
    BEGIN

    DUP_COUNT = .DUP_COUNT + 1;

    ! If we have exceeded the max number of dups per bucket then
    ! get rid of this bucket and start a new one
    !
    IF .DUP_COUNT GEQ .MAX_NUM_DUP
    THEN
        BEGIN

        LOAD_DATA_BUCKET();

        ! The record to go into the next bucket will be a duplicate
        !
        DUPLICATE = _SET;

        ! We are now in a continuation bucket
        !
        SKIP = _SET;

        ! Copy the key into the record (in a continuation bucket
        ! there is no dup count ie. the 4)
        !
        CONV$$COPY_KEY( 4 );

        ! Start counting dups again
        !
        DUP_COUNT = 0;

        ! Set the sidr array record size
        !
        CTX [ CTX$W_RDS ] = 0;

        ! Set some control fields. NOTE: COPY_KEY sets prologue 3
        ! record size field.
        !
        IF NOT .CONV$GB_PROL_V3
        THEN
            BEGIN

            ! A continuation record has no duplicate pointer
            !
            RECORD_CTRL [ IRC$B_CONTROL ] = IRC$M_NOPTRSZ;

            ! Prologue 1,2 size field includes a key
            !
            RECORD_CTRL [ IRC$W_NODUPSZ ] = .KEY_DESC [ KEY$B_KEYSZ ]

            END
        END
    END
ELSE
    BEGIN
```

```
! If this is the first non-dup then don't load anything else
! load the last record processed
!
IF NOT .CTX [ CTX$V_FST ]
THEN
    BEGIN

    LOAD_DATA_BUCKET();

    ! The next record will not be a duplicate record
    !
    DUPLICATE = _CLEAR;

    ! If we were in a continuatio bucket then dont make an index
    ! for it. Also write the bucket because we don't put anything
    ! in a bucket after a dup.
    !
    IF .SKIP
    THEN
        BEGIN
        SKIP = _CLEAR;

        CONV$$WRITE_BUCKET();
        CONV$$INIT_BUCKET();

        ! The next record will always fit into the new bucket
        ! so clearing the continuation flag is ok
        !
        CONTINUATION = _CLEAR

        END
    END;

! Copy the key into the record past the dup count field (ie the 8)
!
CONV$$COPY_KEY( 8 );

! Start counting the dups
!
DUP_COUNT = 0;

! Set the sidr array record size
!
CTX [ CTX$W_RDS ] = 0;

! Set some control fields. NOTE: COPY_KEY sets prologue 3
! record size field.
!
IF NOT .CONV$GB_PROL_V3
THEN
    BEGIN

    ! The size of the dup pointer (1=4bytes)
    !
    RECORD_CTRL [ IRC$B_CONTROL ] = 1;

    ! Zero the field (not implemented)
```

```
8                     !
8                     RECORD_CTRL [ IRC$L_DUPCOUNT ] = 0;
8
8                     ! Prologue 1,2 size field includes a key
8                     !
8                     RECORD_CTRL [ IRC$W_DUPSZ ] = .KEY_DESC [ KEY$B_KEYSZ ]
8
8                     END
8
6                 END;
6
6             ! Add to the size of the dup for this record.
6             !
7             ( IF .CONV$GB_PROL_V3
7             THEN
7
7                 ! A prologue 3 RRV is 7 bytes (1 control,2 ID,4 VBN)
7                 !
7                 RECORD_CTRL [ IRC$W_P3SZ ] = .RECORD_CTRL [ IRC$W_P3SZ ] + 7
7
7             ELSE
8                 BEGIN
8
8                 ! A prologue 1,2 RRV is 6 bytes (1 control,1 ID,4 VBN )
8                 !
8                 IF .RECORD_CTRL [ IRC$V_NOPTRSZ ]
8                 THEN
8                     RECORD_CTRL [ IRC$W_NODUPSZ ] =
8                                             .RECORD_CTRL [ IRC$W_NODUPSZ ] + 6
8                 ELSE
8                     RECORD_CTRL [ IRC$W_DUPSZ ] =
7                                             .RECORD_CTRL [ IRC$W_DUPSZ ] + 6
7                 END )
7
6             END
5         ELSE
6             BEGIN
6
6             ! If the keys are duplicate and we are not allowing dup. then error
6             !
6             IF .DUP
6             THEN
6                 SIGNAL_STOP( CONV$_LOADIDX,
6                                 1,
6                                 .KEY_DESC [ KEY$B_KEYREF ],
6                                 RMS$_DUP );
6
6             ! If this is the first record don't load anything else load the
6             ! last record
6             !
6             IF NOT .CTX [ CTX$V_FST ]
6             THEN
6                 LOAD_DATA_BUCKET();
6
6             ! Move the key value
6             !
6             CONV$$COPY_KEY( 4 );
```

```

          ! Set the sidr array record size
          !
          CTX [ CTX$W_RDS ] = 0;

          ! Set some control fields. NOTE: COPY_KEY sets prologue 3 record
          ! size field NOT counting the pointer so we must add it here
          !
          IF .CONV$GB_PROL_V3
          THEN
              RECORD_CTRL [ IRC$W_P3SZ ] = .RECORD_CTRL [ IRC$W_P3SZ ] + 7
          ELSE
              BEGIN

              ! Non dup records don't have a dup count
              !
              RECORD_CTRL [ IRC$B_CONTROL ] = IRC$M_NOPTRSZ;

              RECORD_CTRL [ IRC$W_NODUPSZ ] = .KEY_DESC [ KEY$B_KEYSZ ] + 6

              END

          END;

      ! Load the SIDR array pointer
      !
      BEGIN   ! SIDR local

      DEFINE_VBN_ID_GLOBAL;

      LOCAL   SIDR    : REF BLOCK [ ,BYTE ];

      ! Convert the VBN and the ID that SORT returns in the file
      !
      CONV$$CONVERT_VBN_ID();

      ! Move the record pointer right after the last one, if any
      !
      SIDR = .CTX [ CTX$L_RDP ] + .CTX [ CTX$W_RDS ];

      ! If prologue 3 the ID is bigger
      !
      IF .CONV$GB_PROL_V3
      THEN
          BEGIN

          ! Set the first_key flag if necessary
          !
          IF .DUP
          THEN
              SIDR [ 0,0,8,0 ] = 2                       ! Can't be first if dup
          ELSE
              SIDR [ 0,0,8,0 ] = 2 + IRC$M_FIRST_KEY;    ! Set flag and size

          SIDR [ 1,0,16,0 ] = .SORT_ID;
          SIDR [ 3,0,32,0 ] = .SORT_VBN;
          CTX [ CTX$W_RDS ] = .CTX [ CTX$W_RDS ] + 7
```

```
; 1255        1249  7                 END
; 1256        1250  6             ELSE
; 1257        1251  7                 BEGIN
; 1258        1252  7                 SIDR [ 0,0,8,0 ] = 2;
; 1259        1253  7                 SIDR [ 1,0,8,0 ] = .SORT_ID;
; 1260        1254  7                 SIDR [ 2,0,32,0 ] = .SORT_VBN;
; 1261        1255  7                 CTX [ CTX$W_RDS ] = .CTX [ CTX$W_RDS ] + 6
; 1262        1256  7                 END
; 1263        1257  7
; 1264        1258  5             END;                ! SIDR local
; 1265        1259  5
; 1266        1260  5             ! If we are here then we have processed at least one non null record
; 1267        1261  5             !
; 1268        1262  5             CTX [ CTX$V_FST ] = _CLEAR;
; 1269        1263  5
; 1270        1264  5             ! If this is a non dup key then copy the current record into
; 1271        1265  5             ! dup buffer
; 1272        1266  5             !
; 1273        1267  5             IF NOT .DUP
; 1274        1268  5             THEN
; 1275        1269  5                 CH$MOVE( .KEY_DESC [ KEY$B_KEYSZ ],
; 1276        1270  5                          .CONV$GL_RFA_BUFFER + 6,
; 1277        1271  5                          .CONV$GL_DUP_BUF )
; 1278        1272  5
; 1279        1273  5             END                 ! NULL_BLK null key value block
; 1280        1274  5
; 1281        1275  3             END;                ! Main loop
; 1282        1276  3
; 1283        1277  3         ! If we exited because of end of file AND we got at least 1
; 1284        1278  3         !  non-null key value, then finish off the index
; 1285        1279  3         !
; 1286        1280  3         IF .STATUS EQL RMS$_EOF AND NOT .ALL_NULL
; 1287        1281  3         THEN
; 1288        1282  4             BEGIN
; 1289        1283  4
; 1290        1284  4             ! There is a SIDR record left over at this point
; 1291        1285  4             ! We must load it in before we finish off the index
; 1292        1286  4             !
; 1293        1287  4             LOAD_DATA_BUCKET();
; 1294        1288  4
; 1295        1289  4             FINISH_INDEX()
; 1296        1290  4
; 1297        1291  3             END;
; 1298        1292  3
; 1299        1293  3         RETURN
; 1300        1294  3
; 1301        1295  3         END         ! RECORD_CTRL local
; 1302        1296  3
; 1303        1297  1         END;
; INFO#250                L1:1025
; Referenced LOCAL symbol DUP_COUNT is probably not initialized


                                                                  .EXTRN  SYS$GET

                                           01FC   8F  BB 00000 CONV$$LOAD_SECONDARY::
```

```
                                                                PUSHR   #^M<R2,R3,R4,R5,R6,R7,R8>                 ; 0794
                               5E               14 C2 00004      SUBL2   #20, SP
                                        0000'   CF B4 00007      CLRW    CONTINUATION                              ; 0860
                       0000G   CF   00000000G   8F D0 0000B      MOVL    #CONV$_WRITEERR, CONV$AB_OUT_RAB+24       ; 0865
                               5A       0000'   CF D0 00014      MOVL    CONV$GL_CTX_BLOCK, CTX                    ; 0869
                               7E          08   AB 9A 00019      MOVZBL  8(KEY_DESC), -(SP)                        ; 0876
                                             0000G 30 0001D      BSBW    CONV$$GET_BUCKET
                          54   AB          08   AA D0 00020      MOVL    8(CTX), 84(KEY_DESC)                      ; 0878
                               5A          5C   AA 9E 00025      MOVAB   92(R10), CTX                              ; 0882
                               6E          07   AB 9A 00029      MOVZBL  7(KEY_DESC), (SP)                         ; 0883
                                             0000G 30 0002D      BSBW    CONV$$GET_BUCKET
                               5E               04 C0 00030      ADDL2   #4, SP
                               5A       0000'   CF D0 00033      MOVL    CONV$GL_CTX_BLOCK, CTX                    ; 0885
                               42          10   AB E9 00038      BLBC    16(KEY_DESC), 4$                          ; 0894
                               29       0000G   CF E9 0003C      BLBC    CONV$GB_PROL_V3, 3$                       ; 0899
                 10       10   AB               03 E1 00041      BBC     #3, 16(KEY_DESC), 1$                      ; 0904
                               50          2A   AA 3C 00046      MOVZWL  42(CTX), R0                               ; 0912
                               51          14   AB 9A 0004A      MOVZBL  20(KEY_DESC), R1
                               50               51 C2 0004E      SUBL2   R1, R0
                               50               04 C2 00051      SUBL2   #4, R0                                    ; 0911
                                                0E 11 00054      BRB     2$                                        ; 0912
                               50          2A   AA 3C 00056 1$:  MOVZWL  42(CTX), R0                               ; 0920
                               51          14   AB 9A 0005A      MOVZBL  20(KEY_DESC), R1
                               50               51 C2 0005E      SUBL2   R1, R0
                               50               02 C2 00061      SUBL2   #2, R0                                    ; 0919
                 6E            50               07 C7 00064 2$:  DIVL3   #7, R0, MAX_NUM_DUP                       ; 0920
                                                17 11 00068      BRB     5$                                        ; 0904
                               50          2A   AA 3C 0006A 3$:  MOVZWL  42(CTX), R0                               ; 0929
                               51          14   AB 9A 0006E      MOVZBL  20(KEY_DESC), R1
                               50               51 C2 00072      SUBL2   R1, R0
                               50               08 C2 00075      SUBL2   #8, R0                                    ; 0928
                 6E            50               06 C7 00078      DIVL3   #6, R0, MAX_NUM_DUP                       ; 0929
                                                03 11 0007C      BRB     5$                                        ; 0899
                               6E               01 D0 0007E 4$:  MOVL    #1, MAX_NUM_DUP                           ; 0931
                 51            6E               07 C5 00081 5$:  MULL3   #7, MAX_NUM_DUP, R1                       ; 0944
                               50          14   AB 9A 00085      MOVZBL  20(KEY_DESC), R0
                               50          08 A041 9E 00089      MOVAB   8(R0)[R1], BYTES
                                                50 DD 0008E      PUSHL   BYTES                                     ; 0949
                                             0000G 30 00090      BSBW    CONV$$GET_TEMP_VM
                               5E               04 C0 00093      ADDL2   #4, SP
                          34   AA               50 D0 00096      MOVL    R0, 52(CTX)
                 50            51          34   AA C1 0009A      ADDL3   52(CTX), R1, R0                           ; 0953
                       0000'   CF          08   A0 9E 0009F      MOVAB   8(R0), CONV$GL_DUP_BUF
                               04          AE   D4 000A5         CLRL    SKIP                                      ; 0968
                               58          30   AA D0 000A8      MOVL    48(CTX), RECORD_CTRL                      ; 0970
                          08   AE               01 D0 000AC      MOVL    #1, ALL_NULL                              ; 0972
                                        0000G   CF 9F 000B0 6$:  PUSHAB  CONV$AB_RFA_RAB                           ; 0977
                   00000000G   00               01 FB 000B4      CALLS   #1, SYS$GET
                          10   AE               50 D0 000BB      MOVL    R0, STATUS
                               03          10   AE E8 000BF      BLBS    STATUS, 7$
                                              013E 31 000C3      BRW     25$
                               50       0000G   CF 3C 000C6 7$:  MOVZWL  CONV$AB_RFA_RAB+34, R0                    ; 0989
                               50               06 C2 000CB      SUBL2   #6, R0
    50       14  AB            08               00 ED 000CE      CMPZV   #0, #8, 20(KEY_DESC), R0
                                                DA 1A 000D4      BGTRU   6$
                 06       10   AB               02 E1 000D6      BBC     #2, 16(KEY_DESC), 8$                      ; 0995
                                             0000G 30 000DB      BSBW    CONV$$CHECK_NULL                          ; 1000
                          CF        50  E8 000DE       BLBS    R0, 6$                                  ;
                          03   08   AE  E9 000E1  8$:  BLBC    ALL_NULL, 9$                            ; 1007
                               08   AE  D4 000E5       CLRL    ALL_NULL
                                 0000G  30 000E8  9$:  BSBW    CONV$$CHECK_S_DUP                       ; 1011
                          52        50  D0 000EB       MOVL    R0, DUP
                          7D   10   AB  E9 000EE       BLBC    16(KEY_DESC), 13$                       ; 1015
                          32        52  E9 000F2       BLBC    DUP, 10$                                ; 1021
                               0C   AE  D6 000F5       INCL    DUP_COUNT                               ; 1025
                          6E   0C   AE  D1 000F8       CMPL    DUP_COUNT, MAX_NUM_DUP                  ; 1030
                                    62  19 000FC       BLSS    12$
                                 0000V  30 000FE       BSBW    LOAD_DATA_BUCKET                        ; 1034
                 0000'    CF        01  90 00101       MOVB    #1, DUPLICATE                           ; 1038
                    04    AE        01  D0 00106       MOVL    #1, SKIP                                ; 1042
                                    04  DD 0010A       PUSHL   #4                                      ; 1047
                                 0000G  30 0010C       BSBW    CONV$$COPY_KEY
                          5E        04  C0 0010F       ADDL2   #4, SP
                               0C   AE  D4 00112       CLRL    DUP_COUNT                               ; 1051
                               3A   AA  B4 00115       CLRW    58(CTX)                                 ; 1055
                          43 0000G  CF  E8 00118       BLBS    CONV$GB_PROL_V3, 12$                    ; 1060
                          68        10  90 0011D       MOVB    #16, (RECORD_CTRL)                      ; 1066
                    02    A8   14   AB  9B 00120       MOVZBW  20(KEY_DESC), 2(RECORD_CTRL)            ; 1070
                                    39  11 00125       BRB     12$                                     ; 1030
                          18        6A  E8 00127 10$:  BLBS    (CTX), 11$                              ; 1081
                                 0000V  30 0012A       BSBW    LOAD_DATA_BUCKET                        ; 1085
                               0000'  CF  94 0012D     CLRB    DUPLICATE                               ; 1089
                          0D   04   AE  E9 00131       BLBC    SKIP, 11$                               ; 1095
                               04   AE  D4 00135       CLRL    SKIP                                    ; 1098
                                 0000G  30 00138       BSBW    CONV$$WRITE_BUCKET                      ; 1100
                                 0000G  30 0013B       BSBW    CONV$$INIT_BUCKET                       ; 1101
                               0000'  CF  94 0013E     CLRB    CONTINUATION                            ; 1106
                                    08  DD 00142 11$:  PUSHL   #8                                      ; 1113
                                 0000G  30 00144       BSBW    CONV$$COPY_KEY
                          5E        04  C0 00147       ADDL2   #4, SP
                               0C   AE  D4 0014A       CLRL    DUP_COUNT                               ; 1117
                               3A   AA  B4 0014D       CLRW    58(CTX)                                 ; 1121
                          4C 0000G  CF  E8 00150       BLBS    CONV$GB_PROL_V3, 16$                    ; 1126
                          68        01  90 00155       MOVB    #1, (RECORD_CTRL)                       ; 1132
                               02   A8  D4 00158       CLRL    2(RECORD_CTRL)                          ; 1136
                    06    A8   14   AB  9B 0015B       MOVZBW  20(KEY_DESC), 6(RECORD_CTRL)            ; 1140
                          3C 0000G  CF  E8 00160 12$:  BLBS    CONV$GB_PROL_V3, 16$                    ; 1148
          45              68        04  E0 00165       BBS     #4, (RECORD_CTRL), 18$                  ; 1160
                    06    A8        06  A0 00169       ADDW2   #6, 6(RECORD_CTRL)                      ; 1166
                                    43  11 0016D       BRB     19$                                     ; 1148
                          19        52  E9 0016F 13$:  BLBC    DUP, 14$                                ; 1175
                             000184EC  8F  DD 00172    PUSHL   #99564                                  ; 1177
                          7E   15   AB  9A 00178       MOVZBL  21(KEY_DESC), -(SP)                     ; 1179
                                    01  DD 0017C       PUSHL   #1                                      ; 1177
                            00000000G  8F  DD 0017E    PUSHL   #CONV$_LOADIDX
             00000000G    00        04  FB 00184       CALLS   #4, LIB$STOP
                          03        6A  E8 0018B 14$:  BLBS    (CTX), 15$                              ; 1185
                                 0000V  30 0018E       BSBW    LOAD_DATA_BUCKET                        ; 1187
                                    04  DD 00191 15$:  PUSHL   #4                                      ; 1191
                                 0000G  30 00193       BSBW    CONV$$COPY_KEY
                          5E        04  C0 00196       ADDL2   #4, SP
                               3A   AA  B4 00199       CLRW    58(CTX)                                 ; 1195
                          05 0000G  CF  E9 0019C       BLBC    CONV$GB_PROL_V3, 17$                    ; 1200
                          68        07  A0 001A1 16$:  ADDW2   #7, (RECORD_CTRL)                       ; 1202
```

```
                                  0C  11 001A4        BRB     19$                                                   ;
                      68          10  90 001A6 17$:   MOVB    #16, (RECORD_CTRL)                                    ; 1208
                02    A8     14   AB  9B 001A9        MOVZBW  20(KEY_DESC), 2(RECORD_CTRL)                          ; 1210
                02    A8          06  A0 001AE 18$:   ADDW2   #6, 2(RECORD_CTRL)                                    ;
                               0000G  30 001B2 19$:   BSBW    CONV$$CONVERT_VBN_ID                                  ; 1226
                      50     3A   AA  3C 001B5        MOVZWL  58(CTX), SIDR                                         ; 1230
                      50     34   AA  C0 001B9        ADDL2   52(CTX), SIDR                                         ;
                      1A  0000G   CF  E9 001BD        BLBC    CONV$GB_PROL_V3, 22$                                  ; 1234
                      05          52  E9 001C2        BLBC    DUP, 20$                                              ; 1240
                      60          02  90 001C5        MOVB    #2, (SIDR)                                            ; 1242
                                  04  11 001C8        BRB     21$                                                   ;
                      60     82   8F  90 001CA 20$:   MOVB    #-126, (SIDR)                                         ; 1244
                01    A0          57  B0 001CE 21$:   MOVW    SORT_ID, 1(SIDR)                                      ; 1246
                03    A0          56  D0 001D2        MOVL    SORT_VBN, 3(SIDR)                                     ; 1247
                3A    AA          07  A0 001D6        ADDW2   #7, 58(CTX)                                           ; 1248
                                  0F  11 001DA        BRB     23$                                                   ;
                      60          02  90 001DC 22$:   MOVB    #2, (SIDR)                                            ; 1252
                01    A0          57  90 001DF        MOVB    SORT_ID, 1(SIDR)                                      ; 1253
                02    A0          56  D0 001E3        MOVL    SORT_VBN, 2(SIDR)                                     ; 1254
                3A    AA          06  A0 001E7        ADDW2   #6, 58(CTX)                                           ; 1255
                      6A          01  8A 001EB 23$:   BICB2   #1, (CTX)                                             ; 1262
                      10          52  E8 001EE        BLBS    DUP, 24$                                              ; 1267
                      51     14   AB  9A 001F1        MOVZBL  20(KEY_DESC), R1                                      ; 1269
                      50  0000G   CF  D0 001F5        MOVL    CONV$GL_RFA_BUFFER, R0                                ; 1270
     0000'    DF      06     A0   51  28 001FA        MOVC3   R1, 6(R0), @CONV$GL_DUP_BUF                           ; 1271
                                FEAC  31 00201 24$:   BRW     6$                                                    ; 0979
         0001827A     8F     10   AE  D1 00204 25$:   CMPL    STATUS, #98938                                        ; 1280
                                  0A  12 0020C        BNEQ    26$                                                   ;
                      06     08   AE  E8 0020E        BLBS    ALL_NULL, 26$                                         ;
                               0000V  30 00212        BSBW    LOAD_DATA_BUCKET                                      ; 1287
                               0000V  30 00215        BSBW    FINISH_INDEX                                          ; 1289
                      5E          14  C0 00218 26$:   ADDL2   #20, SP                                               ; 1297
                         01FC     8F  BA 0021B        POPR    #^M<R2,R3,R4,R5,R6,R7,R8>                             ;
                                  05  0021F           RSB                                                           ;

; Routine Size:  544 bytes,    Routine Base:  _CONV$FAST_S + 0238
```

```
1 %SBTTL  'LOAD_DATA_BUCKET'
1 ROUTINE LOAD_DATA_BUCKET : CL$JSB_REG_8 NOVALUE =
1 !++
1 !
1 ! Functional Description:
1 !
1 !       Loads a data bucket independent of key of reference in the
1 !       index.  On a call to LOAD_DATA_BUCKET a record is loaded into a bucket
1 !       and return.  If the record for some reason does not fit into the current
1 !       bucket an index is made for the bucket and the bucket is written to the
1 !       output file.  The written bucket is initialized and then loaded with
1 !       the original record.  The index for a bucket is made by calling
1 !       LOAD_INDEX_BUCKET.
1 !
1 ! Calling Sequence:
1 !
1 !       LOAD_DATA_BUCKET();
1 !
1 ! Input Parameters:
1 !       none
1 !
1 ! Implicit Inputs:
1 !
1 ! Output Parameters:
1 !       none
1 !
1 ! Implicit Outputs:
1 !       none
1 !
1 ! Routine Value:
1 !
1 !       SS$NORMAL or error codes
1 !
1 ! Routines Called:
1 !
1 !       CONV$$GET_BUCKET
1 !       LOAD_INDEX_BUCKET
1 !       CONV$$SAVE_BUCKET
1 !       CONV$$WRITE_BUCKET
1 !       CONV$$INIT_BUCKET
1 !       CONV$$RESTORE_BUCKET
1 !       CONV$$COMPRESS_KEY
1 !       CONV$$MAKE_INDEX
1 !       CONV$$WRITE_VBN
1 !
1 ! Side Effects:
1 !
1 !       Loads a record into a bucket.  Writes buckets and creates indexs
1 !       for lower level buckets
1 !
1 !--
1
2     BEGIN
2
2     DEFINE_CTX;
2     DEFINE_BUCKET;
2     DEFINE_KEY_DESC;
```

```
2       DEFINE_RECORD_CTRL;
2
2       ! Set the bucket pointer to the bucket at this level
2       !
2       BUCKET = .CTX [ CTX$L_CURRENT_BUFFER ];
2
2       ! Will the record fit into the bucket, if not then call this thing
2       ! with an index to the record.
2       !
2       ! A record will not fit into a bucket if:
2       !
2       !   For all files:
2       !
2       ! a) the combined record data size and record control size is greater then
2       !    the space availaible in the bucket.
2       !
2       ! b) the FILL switch is OFF and the space left in the bucket is less then
2       !    that allowed by bucket fill quanities
2       !
2       !   For prologue 1 & 2 files:
2       !
2       ! c) the record ID of the new record is 0 indicating that the bucket is
2       !    filled (as far as id are concerned)
2       !
4       IF ( ( ( .CTX [ CTX$W_RDS ] + .CTX [ CTX$W_RCS ] ) GTRU
4                                                         .CTX [ CTX$W_SPC ] )
3           OR
3
4           ( IF .CONV$GB_PROL_V3
4             THEN  0
4             ELSE  .BUCKET [ BKT$B_NXTRECID ] EQLU 0 )
4
3           OR
3
4           ( ( NOT .CONV$GL_FILL ) AND
5             ( LOCAL
5                   SPACE_USED_IF_RECORD_ADDED;
5               SPACE_USED_IF_RECORD_ADDED = .CTX[CTX$W_USE] + .CTX[CTX$W_RCS]
5                                                + .CTX[CTX$W_RDS];
5               IF .KEY_DESC[KEY$W_DATFILL] - .CTX[CTX$W_USE]                ! If the difference now
5                           LEQ                                                !       (must be signed)
5                   .SPACE_USED_IF_RECORD_ADDED - .KEY_DESC[KEY$W_DATFILL]    ! is less than it would
5               THEN                                                           ! be if the record were added,
5                   TRUE                                                       ! then don't add it
5               ELSE                                                           ! else
5                   FALSE                                                      !       go ahead and add it
3             ) ) )
3
2       THEN
3           BEGIN   ! Load index block
3
3           ! If for some reason we dont want to make an index entry for this
3           ! record then skip it.
3           !
3           IF NOT .CONTINUATION
3           THEN
4               BEGIN
```

```

                ! Increse the level number for the next index level
                !
                CTX = .CTX + CTX$K_BLN;

                ! Call to LOAD_INDEX_BUCKET to load the next level of the index
                !
                LOAD_INDEX_BUCKET();

                ! Return the level
                !
                CTX = .CTX - CTX$K_BLN;

                ! Restore the bucket pointer to the current level bucket since
                ! we should be looking at some other one.
                !
                BUCKET = .CTX [ CTX$L_CURRENT_BUFFER ]

                END;

            ! Write the bucket we filled
            !
            CONV$$WRITE_BUCKET();

            ! If this is a dup then the next bucket is a continuation bucket
            !
            IF .DUPLICATE
            THEN
                CONTINUATION = _SET
            ELSE
                CONTINUATION = _CLEAR;

            ! Initialize the bucket to use it again
            !
            CONV$$INIT_BUCKET()

            END;    ! Load index block

        BEGIN       ! BKT_*_PTR local

          ! Load the record into the bucket...
          ! First we must set up pointers to where the record will go in the bucket
          ! These are:
          !
          LOCAL
              BKT_CTRL_PTR,     ! Control information
              BKT_DATA_PTR;     ! Actual data record

          ! For Prologue 3 files...
          !
          IF .CONV$GB_PROL_V3
          THEN
              BEGIN

              ! If key compression is on do it
              !
              IF .KEY_DESC [ KEY$V_KEY_COMPR ]
```

```
4           THEN
4               CONV$$COMPRESS_KEY();
4
4           ! Key of ref. specific things
4           !
4           IF .KEY_DESC [ KEY$B_KEYREF ] EQL 0
4           THEN
4
4               ! The Primary key...
4               !
5               BEGIN
5
5               ! The record ID
5               !
5               RECORD_CTRL [ IRC$W_ID ] = .BUCKET [ BKT$W_NXTRECID ];
5
5               ! The RRV points to it's self ie. it's own ID and VBN
5               !
5               RECORD_CTRL [ IRC$W_RRV_ID ] = .BUCKET [ BKT$W_NXTRECID ];
5               RECORD_CTRL [ IRC$L_RRV_VBN3 ] = .CTX [ CTX$L_CURRENT_VBN ];
5
5               ! Update the record next record id in the bucket
5               !
5               BUCKET [ BKT$W_NXTRECID ] = .BUCKET [ BKT$W_NXTRECID ] + 1
5
5               END
4           END
3       ELSE
3
3           ! For prologue 1 and 2 files...
3           !
4           BEGIN
4
4           ! The record ID
4           !
4           RECORD_CTRL [ IRC$B_ID ] = .BUCKET [ BKT$B_NXTRECID ];
4
4           ! If this is the primary data level the set up the RRV
4           !
4           IF .KEY_DESC [ KEY$B_KEYREF ] EQL 0
4           THEN
5               BEGIN
5
5               ! The RRV points to itself ie. it's own ID and VBN
5               !
5               RECORD_CTRL [ IRC$B_RRV_ID ] = .BUCKET [ BKT$B_NXTRECID ];
5               RECORD_CTRL [ IRC$L_RRV_VBN ] = .CTX [ CTX$L_CURRENT_VBN ]
5
4               END;
4
4           ! Update the next record id in the bucket
4           !
4           BUCKET [ BKT$B_NXTRECID ] = .BUCKET [ BKT$B_NXTRECID ] + 1
4
3           END;
3
3       ! For all data levels the control bytes are put at the bucket
```

```
        ! freespace.  The data bytes are put directly after the control.
        !
        BKT_CTRL_PTR = .BUCKET [ BKT$W_FREESPACE ] + .BUCKET;
        BKT_DATA_PTR = .BKT_CTRL_PTR + .CTX [ CTX$W_RCS ];

        ! Update the bucket pointer (NOTE: Same update for all cases)
        !
        BUCKET [ BKT$W_FREESPACE ] = .BUCKET [ BKT$W_FREESPACE ] +
                                                .CTX [ CTX$W_RCS ] +
                                                .CTX [ CTX$W_RDS ];

        ! Load the record into the bucket...
        ! Move the control bytes into the bucket
        !
        CH$MOVE( .CTX [ CTX$W_RCS ],.CTX [ CTX$L_RCP ],.BKT_CTRL_PTR );

        ! Move the data bytes (or sidr array) into the bucket

        CH$MOVE( .CTX [ CTX$W_RDS ],.CTX [ CTX$L_RDP ],.BKT_DATA_PTR );

        END;            ! BKT_*_PTR local

        ! Update the amount of space left in the bucket and the amount used
        !
        BEGIN

        LOCAL
            SPACE_USED;

        SPACE_USED = .CTX [ CTX$W_RCS ] + .CTX [ CTX$W_RDS ];

        CTX [ CTX$W_SPC ] = .CTX [ CTX$W_SPC ] - .SPACE_USED;

        CTX [ CTX$W_USE ] = .CTX [ CTX$W_USE ] + .SPACE_USED;

        END;

        ! Make an index for the next level
        !
        CONV$$MAKE_INDEX();

        ! Set the index record control bytes and bucket pointer
        !
        CONV$$WRITE_VBN();

        RETURN

        END;



                      007C    8F  BB 00000 LOAD_DATA_BUCKET:
                                           POSHR   #^M<R2,R3,R4,R5,R6>                            ; 1299
              59        04    AA  D0 00004 MOVL    4(CTX), BUCKET                                  ; 1359
              50        3A    AA  3C 00008 MOVZWL  58(CTX), R0                                     ; 1379
```

```
                              51      38   AA  3C 0000C          MOVZWL  56(CTX), R1                              ;
                              50           51  C0 00010          ADDL2   R1, R0                                   ;
    50       2A      AA       10           00  ED 00013          CMPZV   #0, #16, 42(CTX), R0                     ; 1380
                                           38  1F 00019          BLSSU   2$                                       ;
                              05    0000G  CF  E8 0001B          BLBS    CONV$GB_PROL_V3, 1$                      ; 1383
                                      06   A9  95 00020          TSTB    6(BUCKET)                                ; 1385
                                           2E  13 00023          BEQL    2$                                       ;
                              53    0000G  CF  E8 00025 1$:      BLBS    CONV$GL_FILL, 6$                         ; 1389
                              50      2C   AA  3C 0002A          MOVZWL  44(CTX), R0                              ; 1392
                              51      38   AA  3C 0002E          MOVZWL  56(CTX), R1                              ;
                              50           51  C0 00032          ADDL2   R1, R0                                   ;
                              52      3A   AA  3C 00035          MOVZWL  58(CTX), R2                              ; 1393
                              50           52  C0 00039          ADDL2   R2, SPACE_USED_IF_RECORD_ADDED           ;
                              51      1A   AB  3C 0003C          MOVZWL  26(KEY_DESC), R1                         ; 1394
                              52      2C   AA  3C 00040          MOVZWL  44(CTX), R2                              ;
                              51           52  C2 00044          SUBL2   R2, R1                                   ;
                              52      1A   AB  3C 00047          MOVZWL  26(KEY_DESC), R2                         ; 1396
                              50           52  C2 0004B          SUBL2   R2, R0                                   ;
                              50           51  D1 0004E          CMPL    R1, R0                                   ;
                                           2A  14 00051          BGTR    6$                                       ;
                              0F    0000'  CF  E8 00053 2$:      BLBS    CONTINUATION, 3$                         ; 1409
                              5A      5C   AA  9E 00058          MOVAB   92(R10), CTX                             ; 1415
                                       0000V  30 0005C           BSBW    LOAD_INDEX_BUCKET                        ; 1419
                              5A      A4   AA  9E 0005F          MOVAB   -92(R10), CTX                            ; 1423
                              59      04   AA  D0 00063          MOVL    4(CTX), BUCKET                           ; 1428
                                       0000G  30 00067 3$:       BSBW    CONV$$WRITE_BUCKET                       ; 1434
                              07    0000'  CF  E9 0006A          BLBC    DUPLICATE, 4$                            ; 1438
                     0000'    CF           01  90 0006F          MOVB    #1, CONTINUATION                         ; 1440
                                           04  11 00074          BRB     5$                                       ;
                                    0000'  CF  94 00076 4$:      CLRB    CONTINUATION                             ; 1442
                                       0000G  30 0007A 5$:       BSBW    CONV$$INIT_BUCKET                        ; 1446
                              21    0000G  CF  E9 0007D 6$:      BLBC    CONV$GB_PROL_V3, 8$                      ; 1462
             03      10       AB           06  E1 00082          BBC     #6, 16(KEY_DESC), 7$                     ; 1468
                                       0000G  30 00087           BSBW    CONV$$COMPRESS_KEY                       ; 1470
                                      15   AB  95 0008A 7$:      TSTB    21(KEY_DESC)                             ; 1474
                                           2B  12 0008D          BNEQ    10$                                      ;
                     01       A8      06   A9  B0 0008F          MOVW    6(BUCKET), 1(RECORD_CTRL)                ; 1483
                     03       A8      06   A9  B0 00094          MOVW    6(BUCKET), 3(RECORD_CTRL)                ; 1487
                     05       A8      08   AA  D0 00099          MOVL    8(CTX), 5(RECORD_CTRL)                   ; 1488
                                      06   A9  B6 0009E          INCW    6(BUCKET)                                ; 1492
                                           17  11 000A1          BRB     10$                                      ; 1474
                     01       A8      06   A9  90 000A3 8$:      MOVB    6(BUCKET), 1(RECORD_CTRL)                ; 1504
                                      15   AB  95 000A8          TSTB    21(KEY_DESC)                             ; 1508
                                           0A  12 000AB          BNEQ    9$                                       ;
                     02       A8      06   A9  90 000AD          MOVB    6(BUCKET), 2(RECORD_CTRL)                ; 1514
                     03       A8      08   AA  D0 000B2          MOVL    8(CTX), 3(RECORD_CTRL)                   ; 1515
                                      06   A9  96 000B7 9$:      INCB    6(BUCKET)                                ; 1521
                              51      04   A9  3C 000BA 10$:     MOVZWL  4(BUCKET), BKT_CTRL_PTR                  ; 1528
                              51           59  C0 000BE          ADDL2   BUCKET, BKT_CTRL_PTR                     ;
                              56      38   AA  3C 000C1          MOVZWL  56(CTX), BKT_DATA_PTR                    ; 1529
                              56           51  C0 000C5          ADDL2   BKT_CTRL_PTR, BKT_DATA_PTR               ;
                              50      04   A9  3C 000C8          MOVZWL  4(BUCKET), R0                            ; 1534
                              52      38   AA  3C 000CC          MOVZWL  56(CTX), R2                              ;
                              50           52  C0 000D0          ADDL2   R2, R0                                   ;
             04      A9       50      3A   AA  A1 000D3          ADDW3   58(CTX), R0, 4(BUCKET)                   ; 1535
             61      30       BA      38   AA  28 000D9          MOVC3   56(CTX), @48(CTX), (BKT_CTRL_PTR)        ; 1540
             66      34       BA      3A   AA  28 000DF          MOVC3   58(CTX), @52(CTX), (BKT_DATA_PTR)        ; 1544
```

```
                               50      38    AA  3C 000E5          MOVZWL  56(CTX), SPACE_USED                               ; 1555
                               51      3A    AA  3C 000E9          MOVZWL  58(CTX), R1                                       ;
                               50            51  C0 000ED          ADDL2   R1, SPACE_USED                                    ;
                         2A    AA            50  A2 000F0          SUBW2   SPACE_USED, 42(CTX)                               ; 1557
                         2C    AA            50  A0 000F4          ADDW2   SPACE_USED, 44(CTX)                               ; 1559
                                          0000G  30 000F8          BSBW    CONV$$MAKE_INDEX                                  ; 1565
                                          0000G  30 000FB          BSBW    CONV$$WRITE_VBN                                   ; 1569
                                    007C     8F  BA 000FE          POPR    #^M<R2,R3,R4,R5,R6>                               ; 1573
                                                 05 00102          RSB                                                       ;

; Routine Size:  259 bytes,    Routine Base:  _CONV$FAST_S + 0458
```

```
%SBTTL  'LOAD_INDEX_BUCKET'
ROUTINE LOAD_INDEX_BUCKET : CL$JSB_REG_9 NOVALUE =
!++
!
! Functional Description:
!
!       Loads an index bucket independent level in the index.  On a
!       call to LOAD_INDEX_BUCKET a record is loaded into a bucket and
!       return.  If the record for some reason does not fit into the current
!       bucket an index is made for the bucket and the bucket is written to the
!       output file.  The written bucket is initialized and then loaded with
!       the original record.  The index for a bucket is made by calling
!       LOAD_INDEX_BUCKET recursivly. Each recursive call to LOAD_INDEX_BUCKET
!       is to moving up the index tree.  CTX keeps track to where you are in
!       the tree.  Most all variables are dependent on CTX so that the
!       context of each level is saved.
!
! Calling Sequence:
!
!       LOAD_INDEX_BUCKET()
!
! Input Parameters:
!       none
!
! Implicit Inputs:
!       none
!
! Output Parameters:
!       none
!
! Implicit Outputs:
!       none
!
! Routine Value:
!
!       SS$NORMAL or error codes
!
! Routines Called:
!
!       CONV$$GET_BUCKET
!       LOAD_INDEX_BUCKET       - Recursive call
!       CONV$$WRITE_BUCKET
!       CONV$$INIT_BUCKET
!       CONV$$COMPRESS_INDEX
!       CONV$$WRITE_VBN
!
! Side Effects:
!
!       Loads a record into a bucket.  Writes buckets and creates indexs
!       for lower level buckets
!
!--

    BEGIN

    DEFINE_CTX;
    DEFINE_BUCKET;
```

```
2       DEFINE_KEY_DESC;
2
2       ! Set the bucket pointer to the bucket at this level
2       !
2       BUCKET = .CTX [ CTX$L_CURRENT_BUFFER ];
2
2       ! See if we have reached the maximun level. (If we have this is the
2       ! biggest file in the world!)
2       !
2       IF .CTX [ CTX$B_LEVEL ] GEQU MAX_IDX_LVL - 1
2       THEN
2           SIGNAL_STOP( CONV$_IDX_LIM );
2
2       ! Will the record fit into the bucket, if not then call this thing
2       ! with an index to the record.
2       !
2       ! A record will not fit into a bucket if:
2       !
2       !   For all files:
2       !
2       ! a) the combined record data size and record control size is greater then
2       !    the space availaible in the bucket.
2       !
2       ! b) the FILL switch is OFF and the space left in the bucket is less then
2       !    that allowed by bucket fill quanities
2       !
2       !   For prologue 3 files:
2       !
2       ! c) the bucket below has a different size vbn then this bucket (this
2       !    is to keep the same size vbn index buckets)
2       !
4       IF ( ( ( .CTX [ CTX$W_RDS ] + .CTX [ CTX$W_RCS ] ) GTRU
4                                                          .CTX [ CTX$W_SPC ] )
3           OR
3
4           ( ( NOT .CONV$GL_FILL ) AND
5             ( LOCAL
5                   SPACE_USED_IF_RECORD_ADDED;
5               SPACE_USED_IF_RECORD_ADDED = .CTX[CTX$W_USE] + .CTX[CTX$W_RCS]
5                                                   + .CTX[CTX$W_RDS];
5               IF .KEY_DESC[KEY$W_IDXFILL] - .CTX[CTX$W_USE]                   ! If the difference now
5                           LEQ                                                 !       (must be signed)
5                   .SPACE_USED_IF_RECORD_ADDED - .KEY_DESC[KEY$W_IDXFILL]      ! is less than it would
5               THEN                                                            ! be if the record were added,
5                   TRUE                                                        ! then don't add it
5               ELSE                                                            ! else
5                   FALSE                                                       !       go ahead and add it
4             ) )
4
3           OR
3
4           ( IF .CONV$GB_PROL_V3
4             THEN
5                 ( LOCAL CTX_M1 : REF BLOCK [ ,BYTE ];
5                   CTX_M1 = .CTX - CTX$K_BLN;
5                   IF .BUCKET [ BKT$V_PTR_SZ ] NEQU .CTX_M1 [ CTX$V_VBN ]
5                   THEN 1
```

```
5                    ELSE 0
5                    )
3             ELSE 0 ) )
2       THEN
3           BEGIN    ! Load index block
3
3           ! Switch for the next index level
3           !
3           CTX = .CTX + CTX$K_BLN;
3
3           ! See if the bucket in at the next level is ready if not get it ready
3           !
3           IF NOT .CTX [ CTX$V_RDY ]
3           THEN
3
3               ! Get the space for the bucket
3               !
3               CONV$$GET_BUCKET( .KEY_DESC [ KEY$B_IANUM ] );
3
3           ! Recursive call to LOAD_INDEX_BUCKET to load the next level of the index
3           !
3           LOAD_INDEX_BUCKET();
3
3           ! Return the level
3           !
3           CTX = .CTX - CTX$K_BLN;
3
3           ! Restore the bucket pointer to the current level bucket since
3           ! we should be looking at some other one.
3           !
3           BUCKET = .CTX [ CTX$L_CURRENT_BUFFER ];
3
3           ! Write the bucket we filled
3           !
3           CONV$$WRITE_BUCKET();
3
3           ! Initialize the bucket to use it again
3           !
3           CONV$$INIT_BUCKET()
3
2           END;     ! Load index block
2
3       BEGIN        ! CTX_P1 local
3
3       LOCAL        CTX_P1  : REF BLOCK [ ,BYTE ];
3
3       CTX_P1 = .CTX + CTX$K_BLN;
3
3       ! An index record is made for levels 2 and above ( level 0 and 1 are
3       ! made by LOAD_PRIMARY and LOAD_SECONDARY depending on KEY_REF )
3       ! NOTE: Do this now because latter the key could get compressed
3       !
3       CH$MOVE( .CTX [ CTX$W_RDS ],.CTX [ CTX$L_RDP ],.CTX_P1 [ CTX$L_RDP ] );
3
3       ! Set the size of the data record
3       !
3       CTX_P1 [ CTX$W_RDS ] = .CTX [ CTX$W_RDS ];
```

```

	! Set the size of the control record
	!
	IF .CONV$GB_PROL_V3
	THEN
	    CTX_P1 [ CTX$W_RCS ] = .CTX [ CTX$V_VBN ] + 2
	ELSE
	    CTX_P1 [ CTX$W_RCS ] = .CTX [ CTX$V_VBN ] + 3

	END;		! CTX_P1 local

	BEGIN		! BKT_*_PTR local

	! Load the record into the bucket...
	! First we must set up pointers to where the record will go in the bucket
	! These are:
	!
	LOCAL
	    BKT_CTRL_PTR,	! Control information
	    BKT_DATA_PTR;	! Actual data record

	! The reason we split them up is because prologue 3 files put the two pieces
	! in two different places depending on bucket type (ie. INDEX, PRIMARY data
	! and SECONDARY data bucket.
	!
	! For Prologue 3 files...
	!
	IF .CONV$GB_PROL_V3
	THEN
	    BEGIN

	    ! Prologue 3 files...
	    !
	    IF .KEY_DESC [ KEY$V_IDX_COMPR ]
	    THEN
		CONV$$COMPRESS_INDEX();

	    ! If level 1 save the pointers so we can backup latter
	    !
	    IF .BUCKET [ BKT$B_LEVEL ] EQLU 1
	    THEN
		BEGIN
		SAVE_VBNFS = .BUCKET [ BKT$W_VBNFS ];
		SAVE_KEYFRESPC = .BUCKET [ BKT$W_KEYFRESPC ]
		END;

	    ! Update this pointer first since we go backwards with it
	    !
	    BUCKET [ BKT$W_VBNFS ] = .BUCKET [ BKT$W_VBNFS ] - .CTX [ CTX$W_RCS ];

	    ! For the index levels the control bytes are put at the bucket
	    ! vbn freespace.  The data bytes are put at the key free space.
	    !
	    BKT_CTRL_PTR = .BUCKET [ BKT$W_VBNFS ] + .BUCKET + 1;
	    BKT_DATA_PTR = .BUCKET [ BKT$W_KEYFRESPC ] + .BUCKET;

	    ! Update the rest of the bucket pointers
```

```
4                 !
4                 BUCKET [ BKT$W_KEYFRESPC ] = .BUCKET [ BKT$W_KEYFRESPC ] +
4                                                       .CTX [ CTX$W_RDS ]
4
4                 END
3             ELSE
3
3                 ! For prologue 1 and 2 files...
3                 !
4                 BEGIN
4
4                 ! If level 1 save the pointers so we can backup latter
4                 !
4                 IF .BUCKET [ BKT$B_LEVEL ] EQLU 1
4                 THEN
4                     SAVE_FREESPACE = .BUCKET [ BKT$W_FREESPACE ];
4
4                 ! Set some pointers...
4                 !
4                 ! For prologue 1 and 2 files the control bytes are put at the bucket
4                 ! freespace.  The data bytes are put directly after the control.
4                 !
4                 BKT_CTRL_PTR = .BUCKET [ BKT$W_FREESPACE ] + .BUCKET;
4                 BKT_DATA_PTR = .BKT_CTRL_PTR + .CTX [ CTX$W_RCS ];
4
4                 ! Update the bucket pointer (NOTE: Same update for all cases)
4                 !
4                 BUCKET [ BKT$W_FREESPACE ] = .BUCKET [ BKT$W_FREESPACE ] +
4                                                  .CTX [ CTX$W_RCS ] +
4                                                  .CTX [ CTX$W_RDS ];
4
3                 END;
3
3             ! Load the record into the bucket...
3             ! Move the control bytes into the bucket
3             !
3             CH$MOVE( .CTX [ CTX$W_RCS ],.CTX [ CTX$L_RCP ],.BKT_CTRL_PTR );
3
3             ! Move the data bytes into the bucket
3             !
3             CH$MOVE( .CTX [ CTX$W_RDS ],.CTX [ CTX$L_RDP ],.BKT_DATA_PTR );
3
2             END;          ! BKT_*_PTR local
2
2             ! Update the amount of space left in the bucket and the amount used
2             !
3             BEGIN
3
3             LOCAL
3                 SPACE_USED;
3
3             SPACE_USED = .CTX [ CTX$W_RCS ] + .CTX [ CTX$W_RDS ];
3
3             CTX [ CTX$W_SPC ] = .CTX [ CTX$W_SPC ] - .SPACE_USED;
3
3             CTX [ CTX$W_USE ] = .CTX [ CTX$W_USE ] + .SPACE_USED;
3
```

```
; 1867        1859  2      END;
; 1868        1860  2
; 1869        1861  2      ! Set the index record control bytes and bucket pointer
; 1870        1862  2      !
; 1871        1863  2      CONV$$WRITE_VBN();
; 1872        1864  2
; 1873        1865  2      RETURN
; 1874        1866  2
; 1875        1867  1      END;



                                     00FC    8F   BB 00000 LOAD_INDEX_BUCKET:
                                                                      PUSHR   #^M<R2,R3,R4,R5,R6,R7>               ; 1575
                           59          04    AA   D0 00004            MOVL    4(CTX), BUCKET                       ; 1635
                           1F          02    AA   91 00008            CMPB    2(CTX), #31                          ; 1640
                                             0D   1F 0000C            BLSSU   1$                                   ;
                                     00000000G    8F   DD 0000E       PUSHL   #CONV$_IDX_LIM                       ; 1642
                  00000000G   00             01   FB 00014            CALLS   #1, LIB$STOP                         ;
                           50          3A    AA   3C 0001B 1$:        MOVZWL  58(CTX), R0                          ; 1662
                           51          38    AA   3C 0001F            MOVZWL  56(CTX), R1                          ;
                           50                51   C0 00023            ADDL2   R1, R0                               ;
  50        2A    AA       10                00   ED 00026            CMPZV   #0, #16, 42(CTX), R0                 ; 1663
                                             44   1F 0002C            BLSSU   3$                                   ;
                           29        0000G   CF   E8 0002E            BLBS    CONV$GL_FILL, 2$                     ; 1666
                           50          2C    AA   3C 00033            MOVZWL  44(CTX), R0                          ; 1669
                           51          38    AA   3C 00037            MOVZWL  56(CTX), R1                          ;
                           50                51   C0 0003B            ADDL2   R1, R0                               ;
                           52          3A    AA   3C 0003E            MOVZWL  58(CTX), R2                          ; 1670
                           50                52   C0 00042            ADDL2   R2, SPACE_USED_IF_RECORD_ADDED       ;
                           51          18    AB   3C 00045            MOVZWL  24(KEY_DESC), R1                     ; 1671
                           52          2C    AA   3C 00049            MOVZWL  44(CTX), R2                          ;
                           51                52   C2 0004D            SUBL2   R2, R1                               ;
                           52          18    AB   3C 00050            MOVZWL  24(KEY_DESC), R2                     ; 1673
                           50                52   C2 00054            SUBL2   R2, R0                               ;
                           50                51   D1 00057            CMPL    R1, R0                               ;
                                             16   15 0005A            BLEQ    3$                                   ;
                           34        0000G   CF   E9 0005C 2$:        BLBC    CONV$GB_PROL_V3, 5$                  ; 1682
                           50          A4    AA   9E 00061            MOVAB   -92(R10), CTX_M1                     ; 1685
  51              60       02                05   EF 00065            EXTZV   #5, #2, (CTX_M1), R1                 ; 1686
  51        0D    A9       02                03   ED 0006A            CMPZV   #3, #2, 13(BUCKET), R1               ;
                                             23   13 00070            BEQL    5$                                   ;
                           5A          5C    AA   9E 00072 3$:        MOVAB   92(R10), CTX                         ; 1696
                  0A       6A                02   E0 00076            BBS     #2, (CTX), 4$                        ; 1700
                           7E          06    AB   9A 0007A            MOVZBL  6(KEY_DESC), -(SP)                   ; 1705
                                     0000G   30 0007E                 BSBW    CONV$$GET_BUCKET                     ;
                           5E                04   C0 00081            ADDL2   #4, SP                               ;
                                      FF79   30 00084 4$:             BSBW    LOAD_INDEX_BUCKET                    ; 1709
                           5A          A4    AA   9E 00087            MOVAB   -92(R10), CTX                        ; 1713
                           59          04    AA   D0 0008B            MOVL    4(CTX), BUCKET                       ; 1718
                                     0000G   30 0008F                 BSBW    CONV$$WRITE_BUCKET                   ; 1722
                                     0000G   30 00092                 BSBW    CONV$$INIT_BUCKET                    ; 1726
                           56          5C    AA   9E 00095 5$:        MOVAB   92(R10), CTX_P1                      ; 1734
            34    B6       34    BA    3A    AA   28 00099            MOVC3   58(CTX), @52(CTX), @52(CTX_P1)       ; 1740
                           3A    A6    3A    AA   B0 000A0            MOVW    58(CTX), 58(CTX_P1)                  ; 1744
```

```
                                    0C   0000G    CF E9 000A5         BLBC    CONV$GB_PROL_V3, 6$                          ; 1748
    50              6A              02            05 EF 000AA         EXTZV   #5, #2, (CTX), R0                            ; 1750
          38        A6              50            02 A1 000AF         ADDW3   #2, R0, 56(CTX_P1)                           ;
                                                  0A 11 000B4         BRB     7$                                           ;
    50              6A              02            05 EF 000B6  6$:    EXTZV   #5, #2, (CTX), R0                            ; 1752
          38        A6              50            03 A1 000BB         ADDW3   #3, R0, 56(CTX_P1)                           ;
                                    49   0000G    CF E9 000C0  7$:    BLBC    CONV$GB_PROL_V3, 10$                         ; 1772
                    03        10    AB            03 E1 000C5         BBC     #3, 16(KEY_DESC), 8$                         ; 1778
                                               0000G 30 000CA         BSBW    CONV$$COMPRESS_INDEX                         ; 1780
                                    01      0C    A9 91 000CD  8$:    CMPB    12(BUCKET), #1                               ; 1784
                                                  13 12 000D1         BNEQ    9$                                           ;
                                    50   0000'    CF 3C 000D3         MOVZWL  CONV$GW_VBN_FS_PTR, R0                       ; 1787
                                                6049 9F 000D8         PUSHAB  (R0)[BUCKET]                                 ;
                           0000'    CF            9E B0 000DB         MOVW    @(SP)+, SAVE_VBNFS                           ;
                           0000'    CF      04    A9 B0 000E0         MOVW    4(BUCKET), SAVE_KEYFRESPC                    ; 1788
                                    50   0000'    CF 3C 000E6  9$:    MOVZWL  CONV$GW_VBN_FS_PTR, R0                       ; 1793
                                    56      38    AA 3C 000EB         MOVZWL  56(CTX), R6                                  ;
                                                6049 9F 000EF         PUSHAB  (R0)[BUCKET]                                 ;
                                    9E            56 A2 000F2         SUBW2   R6, @(SP)+                                   ;
                                                6049 9F 000F5         PUSHAB  (R0)[BUCKET]                                 ; 1798
                                    50            9E 3C 000F8         MOVZWL  @(SP)+, R0                                   ;
                                    51      01  A940 9E 000FB         MOVAB   1(BUCKET)[R0], BKT_CTRL_PTR                  ;
                                    57      04    A9 3C 00100         MOVZWL  4(BUCKET), BKT_DATA_PTR                      ; 1799
                                    57            59 C0 00104         ADDL2   BUCKET, BKT_DATA_PTR                         ;
                              04    A9      3A    AA A0 00107         ADDW2   58(CTX), 4(BUCKET)                           ; 1804
                                                  25 11 0010C         BRB     12$                                          ; 1803
                                    01      0C    A9 91 0010E  10$:   CMPB    12(BUCKET), #1                               ; 1815
                                                  06 12 00112         BNEQ    11$                                          ;
                           0000'    CF      04    A9 B0 00114         MOVW    4(BUCKET), SAVE_FREESPACE                    ; 1817
                                    50      04    A9 3C 0011A  11$:   MOVZWL  4(BUCKET), R0                                ; 1824
                    51              50            59 C1 0011E         ADDL3   BUCKET, R0, BKT_CTRL_PTR                     ;
                                    56      38    AA 3C 00122         MOVZWL  56(CTX), R6                                  ; 1825
                    57              51            56 C1 00126         ADDL3   R6, BKT_CTRL_PTR, BKT_DATA_PTR               ;
                                    50            56 C0 0012A         ADDL2   R6, R0                                       ; 1829
          04        A9              50      3A    AA A1 0012D         ADDW3   58(CTX), R0, 4(BUCKET)                       ; 1831
                    61        30    BA            56 28 00133  12$:   MOVC3   R6, @48(CTX), (BKT_CTRL_PTR)                 ; 1838
                    67        34    BA      3A    AA 28 00138         MOVC3   58(CTX), @52(CTX), (BKT_DATA_PTR)            ; 1842
                                    50      3A    AA 3C 0013E         MOVZWL  58(CTX), SPACE_USED                          ; 1853
                                    50            56 C0 00142         ADDL2   R6, SPACE_USED                               ;
                              2A    AA            50 A2 00145         SUBW2   SPACE_USED, 42(CTX)                          ; 1855
                              2C    AA            50 A0 00149         ADDW2   SPACE_USED, 44(CTX)                          ; 1857
                                               0000G 30 0014D         BSBW    CONV$$WRITE_VBN                              ; 1863
                                          00FC    8F BA 00150         POPR    #^M<R2,R3,R4,R5,R6,R7>                       ; 1867
                                                     05 00154         RSB                                                  ;

; Routine Size:  341 bytes,    Routine Base:  _CONV$FAST_S + 055B
```

```
1 %SBTTL  'FINISH_INDEX'
1 ROUTINE FINISH_INDEX : CL$JSB_REG_9 NOVALUE =
1 !++
1 !
1 ! Functional Description:
1 !
1 !       Loads and writes the last buckets in an index.
1 !
1 ! Calling Sequence:
1 !
1 !       FINISH_INDEX()
1 !
1 ! Input Parameters:
1 !       none
1 !
1 ! Implicit Inputs:
1 !       none
1 !
1 ! Output Parameters:
1 !       none
1 !
1 ! Implicit Outputs:
1 !       none
1 !
1 ! Routine Value:
1 !
1 !       CONV$_SUCCESS or error codes
1 !
1 ! Routines Called:
1 !
1 !       CONV$$WRITE_BUCKET
1 !       BACKUP_INDEX
1 !       CONV$$CREATE_HIGH_KEY
1 !       LOAD_INDEX_BUCKET
1 !
1 ! Side Effects:
1 !
1 !       Loads and writes the last buckets in an index.  Deallocates memory used
1 !       for bucket buffers.
1 !
1 !--
1
2     BEGIN
2
2     DEFINE_CTX;
2     DEFINE_BUCKET;
2     DEFINE_KEY_DESC;
2
2
2     ! Finish off the data level bucket.  The reason why we do ths seperatly
2     ! is that there are no more records to go in this bucket.  In the index
2     ! levels there are.
2     !
2     CTX = .CONV$GL_CTX_BLOCK;
2     BUCKET = .CTX [ CTX$L_CURRENT_BUFFER ];
2     BUCKET [ BKT$V_LASTBKT ] = _SET;
2
```

```
    ! Write the data level bucket
    !
    CONV$$WRITE_BUCKET();

    ! If the last data bucket was a continuation bucket then backup one
    ! index record an put the high key there
    !
    IF .CONTINUATION
    THEN
        BACKUP_INDEX();

    ! Create the high key index record to finish things off
    !
    CONV$$CREATE_HIGH_KEY();

    ! Write the last index records into the buckets and then write the
    ! buckets out
    !
    ! Move up to level 1
    !
    CTX = .CTX + CTX$K_BLN;

    ! Loop until each level is processed
    !
    WHILE .CTX [ CTX$V_RDY ]
    DO
        BEGIN

        LOCAL   CTX_P1 : REF BLOCK [ ,BYTE ];

        ! This call to load bucket will finish off this level bucket and create
        ! the index to the next.
        !
        LOAD_INDEX_BUCKET();

        ! Before we write out the last bucket set some control info. in it
        !
        BUCKET = .CTX [ CTX$L_CURRENT_BUFFER ];
        BUCKET [ BKT$V_LASTBKT ] = _SET;

        CTX_P1 = .CTX + CTX$K_BLN;

        ! If there is no bucket above this one then this is the root
        !
        IF ( NOT .CTX_P1 [ CTX$V_RDY ] )
        THEN
            BEGIN
            BUCKET [ BKT$V_ROOTBKT ] = _SET;
            KEY_DESC [ KEY$B_ROOTLEV ] = .CTX [ CTX$B_LEVEL ];
            KEY_DESC [ KEY$L_ROOTVBN ] = .CTX [ CTX$L_CURRENT_VBN ];
            KEY_DESC [ KEY$V_INITIDX ] = _CLEAR
            END;

        ! Write the last bucket at this level
        !
        CONV$$WRITE_BUCKET();
```

```
; 1991          1982  3             ! Clear the bucket ready flag at this level
; 1992          1983  3             !
; 1993          1984  3             CTX [ CTX$V_RDY ] = _CLEAR;
; 1994          1985  3
; 1995          1986  3             ! Prepare to work on the bucket one level up
; 1996          1987  3             !
; 1997          1988  3             CTX = .CTX_P1
; 1998          1989  3
; 1999          1990  2             END;
; 2000          1991  2
; 2001          1992  2         ! Make sure the last IO has completed
; 2002          1993  2         !
; 2003          1994  2         $WAIT( RAB=CONV$AB_OUT_RAB );
; 2004          1995  2
; 2005          1996  2         ! Any more IO will be Syncronous
; 2006          1997  2         !
; 2007          1998  2         CONV$AB_OUT_RAB [ RAB$V_ASY ] = _CLEAR;
; 2008          1999  2
; 2009          2000  2         RETURN
; 2010          2001  2
; 2011          2002  1         END;


                                                                   .EXTRN  SYS$WAIT

                                        52  DD 00000 FINISH_INDEX:
                                                                   PUSHL   R2                                                  ; 1869
                     5A       0000'  CF  D0 00002                  MOVL    CONV$GL_CTX_BLOCK, CTX                              ; 1921
                     59          04  AA  D0 00007                  MOVL    4(CTX), BUCKET                                      ; 1922
              0D     A9              01  88 0000B                  BISB2   #1, 13(BUCKET)                                      ; 1923
                                  0000G  30 0000F                  BSBW    CONV$$WRITE_BUCKET                                  ; 1927
                     03       0000'  CF  E9 00012                  BLBC    CONTINUATION, 1$                                    ; 1932
                                  0000V  30 00017                  BSBW    BACKUP_INDEX                                        ; 1934
                                  0000G  30 0001A 1$:              BSBW    CONV$$CREATE_HIGH_KEY                               ; 1938
                     5A          5C  AA  9E 0001D                  MOVAB   92(R10), CTX                                        ; 1945
        30           6A              02  E1 00021 2$:              BBC     #2, (CTX), 4$                                       ; 1949
                                   FE83  30 00025                  BSBW    LOAD_INDEX_BUCKET                                   ; 1958
                     59          04  AA  D0 00028                  MOVL    4(CTX), BUCKET                                      ; 1962
              0D     A9              01  88 0002C                  BISB2   #1, 13(BUCKET)                                      ; 1963
                     52          5C  AA  9E 00030                  MOVAB   92(R10), CTX_P1                                     ; 1965
        12           62              02  E0 00034                  BBS     #2, (CTX_P1), 3$                                    ; 1969
              0D     A9              02  88 00038                  BISB2   #2, 13(BUCKET)                                      ; 1972
              09     AB          02  AA  90 0003C                  MOVB    2(CTX), 9(KEY_DESC)                                 ; 1973
              0C     AB          08  AA  D0 00041                  MOVL    8(CTX), 12(KEY_DESC)                                ; 1974
              10     AB              10  8A 00046                  BICB2   #16, 16(KEY_DESC)                                   ; 1975
                                  0000G  30 0004A 3$:              BSBW    CONV$$WRITE_BUCKET                                  ; 1980
                     6A              04  8A 0004D                  BICB2   #4, (CTX)                                           ; 1984
                     5A              52  D0 00050                  MOVL    CTX_P1, CTX                                         ; 1988
                                     CC  11 00053                  BRB     2$                                                  ;
                              0000G  CF  9F 00055 4$:              PUSHAB  CONV$AB_OUT_RAB                                     ; 1994
           00000000G  00             01  FB 00059                  CALLS   #1, SYS$WAIT                                        ;
               0000G  CF             01  8A 00060                  BICB2   #1, CONV$AB_OUT_RAB+4                               ; 1998
                                     04  BA 00065                  POPR    #^M<R2>                                             ; 2002
                                     05  00067                     RSB                                                         ;

; Routine Size:  104 bytes,    Routine Base:  _CONV$FAST_S + 06B0
```

```
%SBTTL  'BACKUP_INDEX'
ROUTINE BACKUP_INDEX : CL$JSB_REG_9 NOVALUE =
!++
!
! Functional Description:
!
! Calling Sequence:
!
!       BACKUP_INDEX()
!
! Input Parameters:
!       none
!
! Implicit Inputs:
!       none
!
! Output Parameters:
!       none
!
! Implicit Outputs:
!       none
!
! Routine Value:
!       none
!
! Routines Called:
!       none
!
! Side Effects:
!
!       Loads and writes the last buckets in an index.  Deallocates memory used
!       for bucket buffers.
!
!--

    BEGIN

    DEFINE_CTX;
    DEFINE_BUCKET;
    DEFINE_KEY_DESC;

    LOCAL
        VBN_SIZE,
        CTX_P1          : REF BLOCK [ ,BYTE ],
        RECORD_CTRL     : REF BLOCK [ ,BYTE ];

    CTX_P1 = .CTX + CTX$K_BLN;

    BUCKET = .CTX_P1 [ CTX$L_CURRENT_BUFFER ];

    ! If the last data bucket was a continuation bucket then we will be backing
    ! up index record which requires using the vbn in the last record.  We
    ! can fake out conv$$write_vbn (called in conv$$create_high_key) by stuffing
    ! the vbn in the ctx field.  This is ok since it it never referenced again.
    !
    ! Get the size of the vbn in the old record (in bits)
    !
```

```
; 2070          2060  2     VBN_SIZE = ( .CTX_P1 [ CTX$V_VBN ] + 2 ) * 8;
; 2071          2061  2
; 2072          2062  2     ! Backup the pointers in the bucket above and get the vbn in the record
; 2073          2063  2     !
; 2074          2064  2     IF .CONV$GB_PROL_V3
; 2075          2065  2     THEN
; 2076          2066  3         BEGIN
; 2077          2067  3
; 2078          2068  3         ! For prologue 3 the vbn is at where we are (they go backwards)
; 2079          2069  3         !
; 2080          2070  3         RECORD_CTRL = .BUCKET [ BKT$W_VBNFS ] + .BUCKET + 1;
; 2081          2071  3
; 2082          2072  3         CTX [ CTX$L_CURRENT_VBN ] = .RECORD_CTRL [ 0,0,.VBN_SIZE,0 ];
; 2083          2073  3
; 2084          2074  3         BUCKET [ BKT$W_VBNFS ] = .SAVE_VBNFS;
; 2085          2075  3         BUCKET [ BKT$W_KEYFRESPC ] = .SAVE_KEYFRESPC
; 2086          2076  3
; 2087          2077  3         END
; 2088          2078  2     ELSE
; 2089          2079  3         BEGIN
; 2090          2080  3
; 2091          2081  3         BUCKET [ BKT$W_FREESPACE ] = .SAVE_FREESPACE;
; 2092          2082  3
; 2093          2083  3         RECORD_CTRL = .BUCKET [ BKT$W_FREESPACE ] + .BUCKET;
; 2094          2084  3
; 2095          2085  3         CTX [ CTX$L_CURRENT_VBN ] = .RECORD_CTRL [ 1,0,.VBN_SIZE,0 ]
; 2096          2086  3
; 2097          2087  2         END;
; 2098          2088  2
; 2099          2089  2     RETURN
; 2100          2090  2
; 2101          2091  1     END;



                                          52  DD 00000 BACKUP_INDEX:
                                                                 PUSHL   R2                                        ; 2004
                            50       5C  AA  9E 00002            MOVAB   92(R10), CTX_P1                           ; 2049
                            59       04  A0  D0 00006            MOVL    4(CTX_P1), BUCKET                         ; 2051
       50          60       02           05  EF 0000A            EXTZV   #5, #2, (CTX_P1), R0                      ; 2060
                            50           08  C4 0000F            MULL2   #8, VBN_SIZE                              ;
                            50           10  C0 00012            ADDL2   #16, VBN_SIZE                             ;
                            26     0000G CF  E9 00015            BLBC    CONV$GB_PROL_V3, 1$                       ; 2064
                            52     0000' CF  3C 0001A            MOVZWL  CONV$GW_VBN_FS_PTR, R2                    ; 2070
                                    6249  9F 0001F            PUSHAB  (R2)[BUCKET]                              ;
                            51           9E  3C 00022            MOVZWL  @(SP)+, R1                                ;
                            51     01 A941  9E 00025            MOVAB   1(BUCKET)[R1], RECORD_CTRL                ;
  08   AA          61       50           00  EF 0002A            EXTZV   #0, VBN_SIZE, (RECORD_CTRL), 8(CTX)       ; 2072
                                    6249  9F 00030            PUSHAB  (R2)[BUCKET]                              ; 2074
                            9E     0000' CF  B0 00033            MOVW    SAVE_VBNFS, @(SP)+                        ;
                      04    A9     0000' CF  B0 00038            MOVW    SAVE_KEYFRESPC, 4(BUCKET)                 ; 2075
                            14           11  0003E               BRB     2$
                      04    A9     0000' CF  B0 00040 1$:        MOVW    SAVE_FREESPACE, 4(BUCKET)                 ; 2081
                            51       04  A9  3C 00046            MOVZWL  4(BUCKET), RECORD_CTRL                    ; 2083
                            51           59  C0 0004A            ADDL2   BUCKET, RECORD_CTRL                       ;
```

```
      08   AA        01   A1          50            00  EF 0004D        EXTZV    #0, VBN_SIZE, 1(RECORD_CTRL), 8(CTX)          ; 2085
                                                    04  BA 00054 2$:    POPR     #^M<R2>                                        ; 2091
                                                    05  00056           RSB                                                     ;

; Routine Size:  87 bytes,    Routine Base:  _CONV$FAST_S + 0718


; 2102          2092  1
; 2103          2093  0 END      ELUDOM



                                                                                   .EXTRN  LIB$STOP

;                                    PSECT SUMMARY
;
;       Name                         Bytes                        Attributes
;
;  _CONV$FAST_D                           28  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  _CONV$FAST_S                         1903  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       56        0        1000           00:01.8
;  _$255$DUA28:[CONV.SRC]CONVERT.L32;1             165       43       26          17           00:00.2



; Information:  1
; Warnings:     0
; Errors:       0



;                                  COMMAND QUALIFIERS

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

; Size:           1903 code + 28 data bytes
; Run Time:          00:43.9
; Elapsed Time:      02:19.3
; Lines/CPU Min:       2863
; Lexemes/CPU-Min: 16797
; Memory Used:  250 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