```
   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**CONVFSTRC

```
  CCCCCCCC    OOOOOO    NN      NN  VV      VV  FFFFFFFFFF    SSSSSSSS  TTTTTTTTTT  RRRRRRRR        CCCCCCCC
  CCCCCCCC    OOOOOO    NN      NN  VV      VV  FFFFFFFFFF    SSSSSSSS  TTTTTTTTTT  RRRRRRRR        CCCCCCCC
CC          OO      OO  NN      NN  VV      VV  FF          SS              TT      RR      RR  CC
CC          OO      OO  NN      NN  VV      VV  FF          SS              TT      RR      RR  CC
CC          OO      OO  NNNN    NN  VV      VV  FF          SS              TT      RR      RR  CC
CC          OO      OO  NNNN    NN  VV      VV  FF          SS              TT      RR      RR  CC
CC          OO      OO  NN  NN  NN  VV      VV  FFFFFFFF      SSSSSS        TT      RRRRRRRR    CC
CC          OO      OO  NN  NN  NN  VV      VV  FFFFFFFF      SSSSSS        TT      RRRRRRRR    CC
CC          OO      OO  NN    NNNN  VV      VV  FF                  SS      TT      RR  RR      CC
CC          OO      OO  NN    NNNN  VV      VV  FF                  SS      TT      RR  RR      CC
CC          OO      OO  NN      NN    VV  VV    FF                  SS      TT      RR    RR    CC               ....
CC          OO      OO  NN      NN    VV  VV    FF                  SS      TT      RR    RR    CC               ....
  CCCCCCCC    OOOOOO    NN      NN      VV      FF          SSSSSSSS        TT      RR      RR    CCCCCCCC       ....
  CCCCCCCC    OOOOOO    NN      NN      VV      FF          SSSSSSSS        TT      RR      RR    CCCCCCCC       ....

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
```

```
0 %TITLE  'VAX-11 CONVERT'
0 MODULE  CONV$FSTRC        ( IDENT='V04-000'
0                           ,OPTLEVEL=3
0                           ) =
0
1 BEGIN
1
1 !*****************************************************************************
1 !*                                                                           *
1 !*   COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*   DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*   ALL RIGHTS RESERVED.                                                    *
1 !*                                                                           *
1 !*   THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*   ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*   INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*   COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*   OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*   TRANSFERRED.                                                            *
1 !*                                                                           *
1 !*   THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*   AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*   CORPORATION.                                                            *
1 !*                                                                           *
1 !*   DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*   SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                           *
1 !*                                                                           *
1 !*****************************************************************************
```

```
1 !++
1 !
1 ! Facility:       VAX-11 CONVERT
1 !
1 ! Abstract:       CONVERT index record processing routines
1 !
1 ! Contents:
1 !                 CHECK_S_DUP
1 !                 CHECK_NULL
1 !                 SPLIT_DATA
1 !                 IN_SEGMENT
1 !                 COMPRESS_KEY
1 !                 COMPRESS_INDEX
1 !                 MAKE_INDEX
1 !                 WRITE_VBN
1 !                 COPY_KEY
1 !                 CREATE_HIGH_KEY
1 !
1 ! Environment:
1 !
1 !                 VAX/VMS Operating System
1 !
1 !
1 ! Author:         Keith B Thompson         Creation date:          August-1980
1 !
1 !
1 ! Modified by:
1 !
1 !         V03-011 JWT0144          Jim Teague               17-Nov-1983
1 !                 Fix compare-packed for null keys.
1 !
1 !         V03-010 KBT0553          Keith B. Thompson        25-Aug-1983
1 !                 Packed decimal high key fix
1 !
1 !         V03-009 KBT0480          Keith B. Thompson        31-Jan-1983
1 !                 Change rec_buf_ptr to rfa_buffer
1 !
1 !         V03-008 KBT0440          Keith B. Thompson        16-Dec-1982
1 !                 Add quadword key support
1 !
1 !         V03-007 KBT0403          Keith B. Thompson        19-Nov-1982
1 !                 Fix one byte bug in compress_key, change make_index and
1 !                 check_s_dup
1 !
1 !         V03-006 KBT0383          Keith B. Thompson        26-Oct-1982
1 !                 Add support for prologue 3 sidrs
1 !
1 !         V03-005 KBT0378          Keith B. Thompson        20-Oct-1982
1 !                 Check for keys out of order in split_data
1 !
1 !         V03-004 KBT0351          Keith B. Thompson        4-Oct-1982
1 !                 Use new linkage definitions
1 !
1 !         V03-003 KBT0038          Keith Thompson           3-Apr-1982
1 !                 Correct the vbn_size correction
1 !
1 !         V03-002 KBT0023          Keith Thompson           24-Mar-1982
```

```
0087   1 !                Correct calculation of vbn_size in write_vbn
0088   1 !
0089   1 !        V03-001 KBT0013         Keith Thompson           16-Mar-1982
0090   1 !                Fix the call to compress_data
0091   1 !****
```

```
PSECT
        OWN     = _CONV$FAST_D  (PIC),
        GLOBAL  = _CONV$FAST_D  (PIC),
        PLIT    = _CONV$PLIT    (SHARE,PIC),
        CODE    = _CONV$FAST_S  (SHARE,PIC);

LIBRARY 'SYS$LIBRARY:LIB.L32';
LIBRARY 'SRC$:CONVERT';

DEFINE_ERROR_CODES;

LINKAGE

        CL$COMPARE_KEYS = JSB : GLOBAL( CTX = 10,
                                        KEY_DESC = 11 ),

        CL$IN_SEGMENT   = JSB : GLOBAL( BYTE_OFS = 6,
                                        NEXT_SEG = 7,
                                        NEXT_LEN = 8,
                                        KEY_DESC = 11 )
                                        NOTUSED( 3 );

EXTERNAL ROUTINE
        CONV$$CVT_TO_ASC        : CL$CVT_TO_ASC,
        CONV$$COMPRESS_DATA     : CL$JSB_LINK;

FORWARD ROUTINE
        COMPARE_KEYS    : CL$COMPARE_KEYS,
        IN_SEGMENT      : CL$IN_SEGMENT;

EXTERNAL
        CONV$GW_OUT_REC_SIZ     : SIGNED WORD,          ! Output Rec. Size

        CONV$GL_RFA_BUFFER,

        CONV$AB_OUT_FAB         : $FAB_DECL,

        CONV$GB_PROL_V3         : BYTE,
!
        Data Decl. for Fast Load routines
!
        CONV$GL_RECORD_PTR,
        CONV$GL_DUP_BUF;

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
```

```
1 %SBTTL  'CHECK_S_DUP'
1 GLOBAL ROUTINE  CONV$$CHECK_S_DUP : CL$JSB_REG_9 =
1 !++
1 !
1 ! Functional Description:
1 !
1 !       Checks if the current secondary key is a duplicate of the last key
1 !
1 ! Calling Sequence:
1 !
1 !       CONV$$CHECK_S_DUP()
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 !       1 - Key was a duplicate
1 !       0 - Key was NOT a duplicate
1 !
1 ! Routines Called:
1 !
1 !       CONV$$CVT_TO_ASC
1 !
1 ! Side Effects:
1 !       none
1 !
1 !--
1
2     BEGIN
2
2     DEFINE_CTX;
2     DEFINE_KEY_DESC;
2
2     ! If this is the very first record then just return 0
2     ! else compaire the record with the last one
2     !
2     IF .CTX [ CTX$V_FST ]
2     THEN
2         RETURN 0
2     ELSE
2         RETURN CH$EQL ( .KEY_DESC [ KEY$B_KEYSZ ],            ! Key size
2                         .CONV$GL_DUP_BUF,                     ! Old key
2                         .KEY_DESC [ KEY$B_KEYSZ ],
2                         .CONV$GL_RFA_BUFFER + 6 )             ! New key
2
1     END;
```

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

                                                              .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$$CVT_TO_ASC
                                                              .EXTRN  CONV$$COMPRESS_DATA
                                                              .EXTRN  CONV$GW_OUT_REC_SIZ
                                                              .EXTRN  CONV$GL_RFA_BUFFER
                                                              .EXTRN  CONV$AB_OUT_FAB
                                                              .EXTRN  CONV$GB_PROL_V3
                                                              .EXTRN  CONV$GL_RECORD_PTR
                                                              .EXTRN  CONV$GL_DUP_BUF

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

                                          1C  BB 00000 CONV$$CHECK_S_DUP::
                                                              PUSHR   #^M<R2,R3,R4>                          ; 0147
                           04             6A  E9 00002        BLBC    (CTX), 1$                              ; 0192
                                          50  D4 00005        CLRL    R0                                     ; 0196
                                          1F  11 00007        BRB     3$                                     ;
                           52      14     AB  9A 00009 1$:    MOVZBL  20(KEY_DESC), R2                       ;
                           51      14     AB  9A 0000D        MOVZBL  20(KEY_DESC), R1                       ; 0198
                           50   0000G     CF  D0 00011        MOVL    CONV$GL_RFA_BUFFER, R0                 ; 0199
                                          54  D4 00016        CLRL    R4                                     ; 0196
        51         00   0000G   DF        52  2D 00018        CMPC5   R2, @CONV$GL_DUP_BUF, #0, R1, 6(R0)    ;
                                   06     A0     0001F                                                       ;
                                          02  12 00021        BNEQ    2$                                     ;
                                          54  D6 00023        INCL    R4                                     ;
                           50             54  D0 00025 2$:    MOVL    R4, R0                                 ;
                                          1C  BA 00028 3$:    POPR    #^M<R2,R3,R4>                          ; 0201
                                          05     0002A        RSB                                            ;
```

```
; Routine Size:  43 bytes,    Routine Base:  _CONV$FAST_S + 0000
```

```
1 %SBTTL  'CHECK_NULL'
1 GLOBAL ROUTINE  CONV$$CHECK_NULL : CL$JSB_REG_9 =
1 !++
1 !
1 ! Functional Description:
1 !
1 !       Checks is the current secondary key is a null key value
1 !
1 ! Calling Sequence:
1 !
1 !       CONV$$CHECK_NULL()
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 !       1 - Key was a null value
1 !       0 - Key was NOT a null value
1 !
1 ! Routines Called:
1 !       none
1 !
1 ! Side Effects:
1 !       none
1 !
1 !--
1
2     BEGIN
2
2     DEFINE_CTX;
2     DEFINE_BUCKET;
2     DEFINE_KEY_DESC;
2
2     BUILTIN
2         CMPP;
2
2     BIND
2         SEG_SIZE = .KEY_DESC + 44 : VECTOR [ ,BYTE ];
2
2     LOCAL
2         KEYPTR  : REF BLOCK [ ,BYTE ],
2         NUL;
2
2     ! The new key is 6 bytes into the input record
2     !
2     KEYPTR = .CONV$GL_RFA_BUFFER + 6;
2
```

```
! Check the record one segment at a time
!
INCR I FROM 0 TO ( .KEY_DESC [ KEY$B_SEGMENTS ] - 1 ) BY 1
DO

    ! If we dont have a null then dont bother looking any further
    !
    IF ( NUL = ( CASE .KEY_DESC [ KEY$B_DATATYPE ] FROM KEY$C_STRING TO KEY$C_MAX_DATA OF
            SET
            [ KEY$C_STRING ]  : CH$EQL( 1,KEY_DESC [ KEY$B_NULLCHAR ],
                                        .SEG_SIZE [ .I ],
                                        .KEYPTR,
                                        .KEY_DESC [ KEY$B_NULLCHAR ] );

            [ KEY$C_SGNWORD,KEY$C_UNSGNWORD ] : .KEYPTR [ 0,WORD_U ] EQLU 0;

            [ KEY$C_SGNLONG,KEY$C_UNSGNLONG ] : .KEYPTR [ 0,LONG_U ] EQLU 0;

            [ KEY$C_PACKED ] :

                            BEGIN

                            STACKLOCAL NULL;

                            NULL = 12;

                            ! The value returned by CMPP is backwards
                            !
                            IF CMPP( %REF(0), NULL,
                                    %REF (( .SEG_SIZE [ .I ] * 2 ) - 1),
                                    .KEYPTR ) EQLU 0
                            THEN
                                _SET
                            ELSE
                                _CLEAR

                            END;

            [ KEY$C_SGNQUAD,KEY$C_UNSGNQUAD ] :
                                ( .KEYPTR [ 0,LONG_U ] EQLU 0 ) AND
                                ( .KEYPTR [ 4,LONG_U ] EQLU 0 );

            TES ) )
    THEN
        KEYPTR = .KEYPTR + .SEG_SIZE [ .I ]
    ELSE
        EXITLOOP;

RETURN .NUL

END;
```

```
                                   01FC    8F   BB 00000 CONV$$CHECK_NULL::
                                                              PUSHR   #^M<R2,R3,R4,R5,R6,R7,R8>                    ; 0203
                           5E          04   C2 00004          SUBL2   #4, SP                                       :
        56        0000G    CF          06   C1 00007          ADDL3   #6, CONV$GL_RFA_BUFFER, KEYPTR               ; 0257
                           58    12    AB   9A 0000D          MOVZBL  18(KEY_DESC), R8                             ; 0261
                           54          01   CE 00011          MNEGL   #1, I                                        ; 0266
                                     008B   31 00014          BRW     15$                                          :
        07                 00    11    AB   8F 00017 1$:      CASEB   17(KEY_DESC), #0, #7                         :
 0030   0028             0028        0010      0001C 2$:      .WORD   3$-2$,-                                      :
 0063   0063             003D        0030      00024                  5$-2$,-                                      :
                                                                      5$-2$,-                                      :
                                                                      6$-2$,-                                      :
                                                                      6$-2$,-                                      :
                                                                      9$-2$,-                                      :
                                                                      11$-2$,-                                     :
                                                                      11$-2$                                       :
                           50    2C  AB44   9A 0002C 3$:      MOVZBL  44(KEY_DESC)[I], R0                          ; 0269
                                       55   D4 00031          CLRL    R5                                           ; 0271
  50      13      AB     13    AB      01   2D 00033          CMPC5   #1, 19(KEY_DESC), 19(KEY_DESC), R0, -        :
                                       66      0003A                  (KEYPTR)                                     :
                                       02   12 0003B          BNEQ    4$                                           :
                                       55   D6 0003D          INCL    R5                                           :
                           57          55   D0 0003F 4$:      MOVL    R5, NUL                                      ; 0268
                                       53   11 00042          BRB     14$                                          :
                                       50   D4 00044 5$:      CLRL    R0                                           ; 0273
                                       66   B5 00046          TSTW    (KEYPTR)                                     :
                                       08   13 00048          BEQL    7$                                           :
                                       08   11 0004A          BRB     8$                                           :
                                       50   D4 0004C 6$:      CLRL    R0                                           ; 0275
                                       66   D5 0004E          TSTL    (KEYPTR)                                     :
                                       02   12 00050          BNEQ    8$                                           :
                                       50   D6 00052 7$:      INCL    R0                                           :
                           57          50   D0 00054 8$:      MOVL    R0, NUL                                      :
                                       3E   11 00057          BRB     14$                                          :
                           6E          0C   D0 00059 9$:      MOVL    #12, NULL                                    ; 0283
                           50    2C  AB44   9A 0005C          MOVZBL  44(KEY_DESC)[I], R0                          ; 0288
                           50          02   C4 00061          MULL2   #2, R0                                       :
                                       50   D7 00064          DECL    R0                                           :
  66              50       6E          00   37 00066          CMPP4   #0, NULL, R0, (KEYPTR)                       ; 0289
                                       55   DC 0006B          MOVPSL  R5                                           :
  55              55       02          02   EF 0006D          EXTZV   #2, #2, R5, R5                               :
                                       55   D7 00072          DECL    R5                                           ; 0287
                                       05   12 00074          BNEQ    10$                                          ; 0289
                           57          01   D0 00076          MOVL    #1, NUL                                      ; 0287
                                       1C   11 00079          BRB     14$                                          :
                                       57   D4 0007B 10$:     CLRL    NUL                                          :
                                       18   11 0007D          BRB     14$                                          :
                                       51   D4 0007F 11$:     CLRL    R1                                           ; 0298
                                       66   D5 00081          TSTL    (KEYPTR)                                     :
                                       02   12 00083          BNEQ    12$                                          :
                                       51   D6 00085          INCL    R1                                           :
                                       50   D4 00087 12$:     CLRL    R0                                           ; 0299
                                 04    A6   D5 00089          TSTL    4(KEYPTR)                                    :
                                       02   12 0008C          BNEQ    13$                                          :
                                       50   D6 0008E          INCL    R0                                           :
                           57          51   D2 00090 13$:     MCOML   R1, NUL                                      :
                  57       50          57   CB 00093          BICL3   NUL, R0, NUL                                 :
```

```
                              11        57  E9 00097 14$:     BLBC    NUL, 17$                                     ; 0266
                              50   2C AB44  9A 0009A          MOVZBL  44(KEY_DESC)[I], R0                          ; 0303
                              56        50  C0 0009F          ADDL2   R0, KEYPTR                                   ;
                 02           54        58  F2 000A2 15$:     AOBLSS  R8, I, 16$                                   ; 0266
                                        03  11 000A6          BRB     17$                                          ;
                                      FF6C  31 000A8 16$:     BRW     1$                                           ;
                              50        57  D0 000AB 17$:     MOVL    NUL, R0                                      ; 0307
                              5E        04  C0 000AE          ADDL2   #4, SP                                       ; 0310
                                 01FC   8F  BA 000B1          POPR    #^M<R2,R3,R4,R5,R6,R7,R8>                    ;
                                            05 000B5          RSB                                                  ;

; Routine Size:  182 bytes,    Routine Base:  _CONV$FAST_S + 002B
```

```
%SBTTL  'SPLIT_DATA'
GLOBAL ROUTINE  CONV$$SPLIT_DATA : CL$JSB_REG_9 =
!++
!
! Functional Description:
!
!         Copies the data record returned by GET_RECORD into the primary
!         key data buffer, for prologue 3 files it also extracts the key
!         from the data and places it into the record control buffer
!         Also checks to see if the present record is a duplicate of the
!         last record
!
! Calling Sequence:
!
!         CONV$$SPLIT_DATA()
!
! Input Parameters:
!         none
!
! Implicit Inputs:
!         none
!
! Output Parameters:
!         none
!
! Implicit Outputs:
!         none
!
! Routine Value:
!
!          1 - Key was a duplicate
!          0 - Key was NOT a duplicate
!         -1 - Key was out of order
!
! Routines Called:
!
!         CONV$$CVT_TO_ASC
!         IN_SEGMENT
!         CONV$$COMPRESS_DATA
!
! Side Effects:
!         none
!
!--

    BEGIN

    DEFINE_CTX;
    DEFINE_BUCKET;
    DEFINE_KEY_DESC;

    BIND
        SEG_SIZE        = .KEY_DESC + 44 : VECTOR [ ,BYTE ],
        SEG_POSI        = .KEY_DESC + 28 : VECTOR [ ,WORD ];

    LOCAL
        DUP     : SIGNED;
```

```
; 2
; 2     ! For prologue 3 files we must extract the key and
; 2     ! data from the input record
; 2     !
; 2     IF .CONV$GB_PROL_V3
; 2     THEN
; 3         BEGIN               ! Prologue 3 processing
; 3
; 4         BEGIN               ! Extract the key
; 4
; 4         LOCAL
; 4             OFFSET,
; 4             REC_PTR;                    ! Pointer TO the data buffer
; 4
; 4         ! Find out where the extracted key will go...
; 4         ! If we are doing key compression the start the control buffer + 13
; 4         ! else it's only 11
; 4         !
; 4         IF .KEY_DESC [ KEY$V_KEY_COMPR ]
; 4         THEN
; 4             OFFSET = 13
; 4         ELSE
; 4
; 4             ! If there is no record compression and the file is fixed then
; 4             ! there is no size field
; 4             !
; 4             IF .KEY_DESC [ KEY$V_REC_COMPR ] OR
; 5                 ( .CONV$AB_OUT_FAB [ FAB$B_RFM ] EQL FAB$C_VAR )
; 4             THEN
; 4                 OFFSET = 11
; 4             ELSE
; 4                 OFFSET = 9;
; 4
; 4         ! Key goes here
; 4         !
; 4         REC_PTR = .CTX [ CTX$L_RCP ] + .OFFSET;
; 4
; 4         ! Move the key to the record buffer where it is suppose to be
; 4         !
; 4         INCR I FROM 0 TO ( .KEY_DESC [ KEY$B_SEGMENTS ] - 1 ) BY 1
; 4         DO
; 5             BEGIN
; 5
; 5             ! Move it
; 5             !
; 5             CH$MOVE( .SEG_SIZE [ .I ],                          ! Segment size
; 5                     .CONV$GL_RECORD_PTR + .SEG_POSI [ .I ],     ! Source
; 5                     .REC_PTR );                                 ! Destination
; 5
; 5             ! Increment destination
; 5             !
; 5             REC_PTR = .REC_PTR + .SEG_SIZE [ .I ]
; 5
; 4             END;
; 4
; 4         ! The size of the control record is the key size plus the overhead
; 4         !
```

```
CTX [ CTX$W_RCS ] = .KEY_DESC [ KEY$B_KEYSZ ] + .OFFSET;

! Check for a duplicate with the last key
!
DUP = COMPARE_KEYS( .KEY_DESC [ KEY$B_KEYSZ ],          ! Key size
                    .CTX [ CTX$L_RCP ] + .OFFSET,       ! New key
                    .CTX [ CTX$L_LKP ] )                ! Last key

END;            ! Extract the key

BEGIN           ! Process the data

LOCAL
    BUFFER_PTR;

BEGIN           ! Extract the data record

GLOBAL REGISTER
    BYTE_OFS = 6,       ! Offset into the data record
    NEXT_SEG = 7,       ! No. of char. to the next segment
    NEXT_LEN = 8;       ! Length of next segment

! Extract the data record
!
BYTE_OFS = 0;

! Find out where to put the data record
!
IF .KEY_DESC [ KEY$V_REC_COMPR ]
THEN
    BUFFER_PTR = .CTX [ CTX$L_RDP ] + 2
ELSE
    BUFFER_PTR = .CTX [ CTX$L_RDP ];

! Loop for the entire length of the record
!
WHILE .BYTE_OFS LSS .CONV$GW_OUT_REC_SIZ
DO

    ! If the this byte is in a key segment then ignore it and move
    ! past this segment
    !
    IF IN_SEGMENT()
    THEN
        BYTE_OFS = .NEXT_SEG + .NEXT_LEN
    ELSE

        BEGIN

        ! If the byte is not in a segment then move it into the buffer
        !
        CH$MOVE( .NEXT_SEG - .BYTE_OFS,
                 .CONV$GL_RECORD_PTR + .BYTE_OFS,
                 .BUFFER_PTR );

        BUFFER_PTR = .BUFFER_PTR + ( .NEXT_SEG - .BYTE_OFS );
        BYTE_OFS = .NEXT_SEG + .NEXT_LEN
```

```
                        END
                END;                    ! Extract the data record

                ! If there is record compression then do it
                !
                IF .KEY_DESC [ KEY$V_REC_COMPR ]
                THEN
                    BEGIN

                    REGISTER
                        NXT_FIELD_PTR   = 6,
                        CNT_FIELD_PTR   = 7,
                        TRU_FIELD_PTR   = 3;

                    NXT_FIELD_PTR = .CTX [ CTX$L_RDP ];
                    CNT_FIELD_PTR = .BUFFER_PTR;

                    CONV$$COMPRESS_DATA();

                    ! We can now say what the data size might be
                    !
                    CTX [ CTX$W_RDS ] = .TRU_FIELD_PTR - .CTX [ CTX$L_RDP ]

                    END
                ELSE

                    ! If there is no compression then the data size is simular to above
                    !
                    CTX [ CTX$W_RDS ] = .BUFFER_PTR - .CTX [ CTX$L_RDP ]

                END                     ! Process the data

                END                     ! Prologue 3 processing
            ELSE
                BEGIN                   ! Prologue 2 processing

                LOCAL       LKP_PTR;

                ! For non prologue 3 files the size is simply what was returned by GET_RECORD
                !
                CTX [ CTX$W_RDS ] = .CONV$GW_OUT_REC_SIZ;

                ! The control is a bit more complicated...
                !
                ! Other files depend on var or fixed format
                !
                IF .CONV$AB_OUT_FAB [ FAB$B_RFM ] EQL FAB$C_FIX
                THEN
                    CTX [ CTX$W_RCS ] = IRC$C_FIXOVHDSZ
                ELSE
                    CTX [ CTX$W_RCS ] = IRC$C_VAROVHDSZ;

                ! Check for duplicate key
                !
                ! Compaire the index part of the record into the buffer segment by
```

```
;  544      0539 3          ! segment. Since the last key value is in last key pointer
;  545      0540 3          ! we can use it to compaire with RECORD_PTR  (The current
;  546      0541 3          ! record).
;  547      0542 3          !
;  548      0543 3          LKP_PTR = .CTX [ CTX$L_LKP ];
;  549      0544 3
;  550      0545 3          ! Compaire each segment until a key-out-of-order is found
;  551      0546 3          !
;  552      0547 3          INCR I FROM 0 TO ( .KEY_DESC [ KEY$B_SEGMENTS ] - 1 ) BY 1
;  553      0548 3          DO
;  554      0549 4              BEGIN
;  555      0550 4
;  556      0551 4              DUP = COMPARE_KEYS( .SEG_SIZE [ .I ],
;  557      0552 4                                  .CONV$GL_RECORD_PTR + .SEG_POSI [ .I ],
;  558      0553 4                                  .LKP_PTR );
;  559      0554 4
;  560      0555 4              ! If key is out of order check no more
;  561      0556 4              !
;  562      0557 4              IF .DUP LSS 0
;  563      0558 4              THEN
;  564      0559 4                  EXITLOOP
;  565      0560 4              ELSE
;  566      0561 4                  LKP_PTR = .LKP_PTR + .SEG_SIZE [ .I ]
;  567      0562 4
;  568      0563 4              END
;  569      0564 4
;  570      0565 2          END;                ! Prologue 2 processing
;  571      0566 2
;  572      0567 2      RETURN .DUP
;  573      0568 2
;  574      0569 1      END;
; INFO#250              L1:0505
; Referenced REGISTER symbol TRU_FIELD_PTR is probably not initialized



                                 01FC   8F  BB 00000 CONV$$SPLIT_DATA::
                                                             PUSHR   #^M<R2,R3,R4,R5,R6,R7,R8>          ; 0312
                  5E              10  C2 00004               SUBL2   #16, SP
                  56       2C     AB  9E 00007               MOVAB   44(KEY_DESC), R6                   ; 0363
                  03    0000G     CF  E8 0000B               BLBS    CONV$GB_PROL_V3, 1$                ; 0372
                           00CE       31 00010               BRW     13$
         05     10        AB      06  E1 00013 1$:           BBC     #6, 16(KEY_DESC), 2$               ; 0386
                  58              0D  D0 00018               MOVL    #13, OFFSET                        ; 0388
                                  14  11 0001B               BRB     5$                                 ;
                          10      AB  95 0001D 2$:           TSTB    16(KEY_DESC)                       ; 0394
                                  07  19 00020               BLSS    3$
                  02    0000G     CF  91 00022               CMPB    CONV$AB_OUT_FAB+31, #2             ; 0395
                                  05  12 00027               BNEQ    4$                                 ;
                  58              0B  D0 00029 3$:           MOVL    #11, OFFSET                        ; 0397
                                  03  11 0002C               BRB     5$                                 ;
                  58              09  D0 0002E 4$:           MOVL    #9, OFFSET                         ; 0399
           08     AE       30   BA48  9E 00031 5$:           MOVAB   @48(CTX)[OFFSET], REC_PTR          ; 0403
           0C     AE       12     AB  9A 00037               MOVZBL  18(KEY_DESC), 12(SP)               ; 0407
                  57              01  CE 0003C               MNEGL   #1, I                              ; 0414
```

```
                                          1B  11 0003F       BRB      7$                                           :
                            51          6746  9A 00041  6$:  MOVZBL   (I)[R6], R1                                  : 0413
                            50       1C AB47  3C 00045       MOVZWL   28(KEY_DESC)[I], R0                          : 0414
                            50      0000G CF  C0 0004A       ADDL2    CONV$GL_RECORD_PTR, R0                       :
               08    BE     60            51  28 0004F       MOVC3    R1, (R0), @REC_PTR                           : 0415
                            50          6746  9A 00054       MOVZBL   (I)[R6], R0                                  : 0419
                      08    AE            50  C0 00058       ADDL2    R0, REC_PTR                                  :
                     E0     57         0C AE  F2 0005C  7$:  AOBLSS   12(SP), I, 6$                                :
                            50         14 AB  9A 00061       MOVZBL   20(KEY_DESC), R0                             : 0425
               38    AA     50            58  A1 00065       ADDW3    OFFSET, R0, 56(CTX)                          :
                                       3C AA  DD 0006A       PUSHL    60(CTX)                                      : 0431
                                     30 BA48  9F 000?D       PUSHAB   @48(CTX)[OFFSET]                             : 0430
                            7E         14 AB  9A 000??       MOVZBL   20(KEY_DESC), -(SP)                          : 0429
                                       0000V  30 00075       BSBW     COMPARE_KEYS                                 :
                            5E            0C  C0 00078       ADDL2    #12, SP                                      :
                      0C    AE            50  D0 0007B       MOVL     R0, DUP                                      :
                                          56  D4 0007F       CLRL     BYTE_OFS                                     : 0449
                            50         34 AA  9E 00081       MOVAB    52(CTX), R0                                  : 0455
                                       10 AB  95 00085       TSTB     16(KEY_DESC)                                 : 0453
                                          0C  18 00088       BGEQ     8$                                           :
                      04    AE            60  D0 0008A       MOVL     (R0), 4(SP)                                  : 0455
               08    AE   04    AE        02  C1 0008E       ADDL3    #2, 4(SP), BUFFER_PTR                        :
                                          09  11 00094       BRB      9$                                           :
                      04    AE            60  D0 00096  8$:  MOVL     (R0), 4(SP)                                  : 0457
                      08    AE         04 AE  D0 0009A       MOVL     4(SP), BUFFER_PTR                            :
    56         0000G  CF    10            00  EC 0009F  9$:  CMPV     #0, #16, CONV$GW_OUT_REC_SIZ, BYTE_OFS       : 0461
                                          1C  15 000A6       BLEQ     11$                                          :
                                       0000V  30 000A8       BSBW     IN_SEGMENT                                   : 0467
                            10            50  E8 000AB       BLBS     R0, 10$                                      :
                      6E    57            56  C3 000AE       SUBL3    BYTE_OFS, NEXT_SEG, (SP)                     : 0476
               08    BE     0000GDF46     6E  28 000B2       MOVC3    (SP), @CONV$GL_RECORD_PTR[BYTE_OFS], -       : 0478
                                                                      @BUFFER_PTR                                  :
                      08    AE            6E  C0 000BA       ADDL2    (SP), BUFFER_PTR                             : 0480
                     56     57            58  C1 000BE 10$:  ADDL3    NEXT_LEN, NEXT_SEG, BYTE_OFS                 : 0481
                                          DB  11 000C2       BRB      9$                                           : 0467
                                       10 AB  95 000C4 11$:  TSTB     16(KEY_DESC)                                 : 0489
                                          0F  18 000C7       BGEQ     12$                                          :
                            56         04 AE  7D 000C9       MOVQ     4(SP), NXT_FIELD_PTR                         : 0498
                                       0000G  30 000CD       BSBW     CONV$$COMPRESS_DATA                          : 0501
               3A    AA     53         04 AE  A3 000D0       SUBW3    4(SP), TRU_FIELD_PTR, 58(CTX)                : 0505
                                          54  11 000D6       BRB      18$                                          :
               3A    AA   08    AE     04 AE  A3 000D8 12$:  SUBW3    4(SP), BUFFER_PTR, 58(CTX)                   : 0512
                                          4B  11 000DF       BRB      18$                                          : 0489
                      3A    AA      0000G CF  B0 000E1 13$:  MOVW     CONV$GW_OUT_REC_SIZ, 58(CTX)                 : 0524
                            01      0000G CF  91 000E7       CMPB     CONV$AB_OUT_FAB+31, #1                       : 0530
                                          06  12 000EC       BNEQ     14$                                          :
                      38    AA            07  B0 000EE       MOVW     #7, 56(CTX)                                  : 0532
                                          04  11 000F2       BRB      15$                                          :
                      38    AA            09  B0 000F4 14$:  MOVW     #9, 56(CTX)                                  : 0534
                            53         3C AA  D0 000F8 15$:  MOVL     60(CTX), LKP_PTR                             : 0543
                            54         12 AB  9A 000FC       MOVZBL   18(KEY_DESC), R4                             : 0547
                            52            01  CE 00100       MNEGL    #1, I                                        :
                                          23  11 00103       BRB      17$                                          :
                                          53  DD 00105 16$:  PUSHL    LKP_PTR                                      : 0553
                            50       1C AB42  3C 00107       MOVZWL   28(KEY_DESC)[I], R0                          : 0552
                                   0000GDF40  9F 0010C       PUSHAB   @CONV$GL_RECORD_PTR[R0]                      :
                            7E          6246  9A 00111       MOVZBL   (I)[R6], -(SP)                               : 0551
```

```
                                 0000V  30 00115        BSBW    COMPARE_KEYS                              ;
                     5E             0C  C0 00118        ADDL2   #12, SP                                   ;
             0C      AE             50  D0 0011B        MOVL    R0, DUP                                   ;
                                    0B  19 0011F        BLSS    18$                                       ; 0557
                     50           6246  9A 00121        MOVZBL  (I)[R6], R0                               ; 0561
                     53             50  C0 00125        ADDL2   R0, LKP_PTR                               ;
        D9           52             54  F2 00128 17$:   AOBLSS  R4, I, 16$                                ; 0557
                     50      0C     AE  D0 0012C 18$:   MOVL    DUP, R0                                   ; 0567
                     5E             10  C0 00130        ADDL2   #16, SP                                   ; 0569
                           01FC     8F  BA 00133        POPR    #^M<R2,R3,R4,R5,R6,R7,R8>                 ;
                                    05     00137        RSB                                               ;

; Routine Size:  312 bytes,    Routine Base:  _CONV$FAST_S + 00E1
```

```
1 %SBTTL  'COMPARE_KEYS'
1 ROUTINE COMPARE_KEYS ( SIZE,NEW : REF BLOCK [ ,BYTE ],OLD : REF BLOCK [,BYTE ] )
1                                                           : CL$COMPARE_KEYS =
1 !++
1 !
1 ! Functional Description:
1 !
1 !
1 ! Calling Sequence:
1 !
1 !       COMPARE_KEYS
1 !
1 ! Input Parameters:
1 !       none
1 !
1 ! Implicit Inputs:
1 !       none
1 !
1 ! Output Parameters:
1 !       none
1 !
1 ! Implicit Outputs:
1 !       none
1 !
1 ! Routine Value:
1 !
1 !        1 - Key/segment is duplicate
1 !        0 - Key/segment is not duplicate
1 !       -1 - Key/segment is out of order
1 !
1 ! Routines Called:
1 !       none
1 !
1 ! Side Effects:
1 !       none
1 !
1 !--
1
2    BEGIN
2
2    DEFINE_CTX;
2    DEFINE_KEY_DESC;
2
2    BUILTIN
2       CMPP;
2
2    LOCAL
2       DUP     : SIGNED;
2
2    ! If this is the first record it can't be anything
2    !
2    IF .CTX [ CTX$V_FST ]
2    THEN
3       BEGIN
3
3       CTX [ CTX$V_FST ] = _CLEAR;
3
```

```
                RETURN 0

                END;

        DUP = ( CASE .KEY_DESC [ KEY$B_DATATYPE ] FROM KEY$C_STRING TO KEY$C_MAX_DATA OF
        SET
            [ KEY$C_STRING ]        : CH$COMPARE( .SIZE,.NEW,.SIZE,.OLD );

            [ KEY$C_SGNWORD ]       : IF .NEW [ 0,WORD_S ] EQL .OLD [ 0,WORD_S ]
                                      THEN
                                          0
                                      ELSE IF .NEW [ 0,WORD_S ] LSS .OLD [ 0,WORD_S ]
                                      THEN
                                          -1
                                      ELSE
                                          1;

            [ KEY$C_UNSGNWORD ]     : IF .NEW [ 0,WORD_U ] EQLU .OLD [ 0,WORD_U ]
                                      THEN
                                          0
                                      ELSE IF .NEW [ 0,WORD_U ] LSSU .OLD [ 0,WORD_U ]
                                      THEN
                                          -1
                                      ELSE
                                          1;

            [ KEY$C_SGNLONG ]       : IF .NEW [ 0,LONG_S ] EQL .OLD [ 0,LONG_S ]
                                      THEN
                                          0
                                      ELSE IF .NEW [ 0,LONG_S ] LSS .OLD [ 0,LONG_S ]
                                      THEN
                                          -1
                                      ELSE
                                          1;

            [ KEY$C_UNSGNLONG ]     : IF .NEW [ 0,LONG_U ] EQLU .OLD [ 0,LONG_U ]
                                      THEN
                                          0
                                      ELSE IF .NEW [ 0,LONG_U ] LSSU .OLD [ 0,LONG_U ]
                                      THEN
                                          -1
                                      ELSE
                                          1;

            [ KEY$C_PACKED ]        : BEGIN

                                      LOCAL NIBBLES;

                                      NIBBLES = ( .SIZE * 2 ) - 1;

                                      CMPP( NIBBLES,.NEW,NIBBLES,.OLD )

                                      END;

            [ KEY$C_SGNQUAD ]       :
                                      ! Check the high longword first
                                      !
```

```
3                               IF .NEW [ 4,LONG_S ] EQL .OLD [ 4,LONG_S ]
3                               THEN
3
3                                   ! If the high long word is eq then check the lower
3                                   !
3                                   IF .NEW [ 0,LONG_U ] EQL .OLD [ 0,LONG_U ]
3                                   THEN
3                                       0
3                                   ELSE IF .NEW [ 0,LONG_U ] LSS .OLD [ 0,LONG_U ]
3                                   THEN
3                                       -1
3                                   ELSE
3                                       1
3                               ELSE IF .NEW [ 4,LONG_S ] LSS .OLD [ 4,LONG_S ]
3                               THEN
3                                   -1
3                               ELSE
3                                   1;
3
3           [ KEY$C_UNSGNQUAD ]   : IF .NEW [ 4,LONG_U ] EQLU .OLD [ 4,LONG_U ]
3                               THEN
3                                   IF .NEW [ 0,LONG_U ] EQLU .OLD [ 0,LONG_U ]
3                                   THEN
3                                       0
3                                   ELSE IF .NEW [ 0,LONG_U ] LSSU .OLD [ 0,LONG_U ]
3                                   THEN
3                                       -1
3                                   ELSE
3                                       1
3                               ELSE IF .NEW [ 4,LONG_U ] LSSU .OLD [ 4,LONG_U ]
3                               THEN
3                                   -1
3                               ELSE
3                                   1;
3
2           TES );
2
2       ! Fix the meaning of the flag
2       !
2       IF .DUP EQL 0
2       THEN
2           DUP = 1
2       ELSE
2           IF .DUP EQL 1
2           THEN
2               DUP = 0;
2
2       ! Return the value
2       !
2       RETURN .DUP
2
1       END;
```

```
                                  00FC   8F   BB 00000 COMPARE_KEYS:
                                                             PUSHR   #^M<R2,R3,R4,R5,R6,R7>              ; 0571
                        06                6A   E9 00004            BLBC    (CTX), 1$                           ; 0621
                        6A                01   8A 00007            BICB2   #1, (CTX)                           ; 0625
                                        0098   31 0000A            BRW     21$                                 ; 0627
                        55          20    AE   D0 0000D 1$:        MOVL    NEW, R5                             ; 0633
                        54          24    AE   D0 00011            MOVL    OLD, R4
                07      00          11    AB   8F 00015            CASEB   17(KEY_DESC), #0, #7                ; 0631
 002D           0028    001F            0010      0001A 2$:        .WORD   3$-2$,-                             ;
 005C           0053    0039            0032      00022                    4$-2$,-                             ;
                                                                           7$-2$,-                             ;
                                                                           8$-2$,-                             ;
                                                                           9$-2$,-                             ;
                                                                           11$-2$,-                            ;
                                                                           12$-2$,-                            ;
                                                                           13$-2$                              ;
                        56                01   D0 0002A 3$:        MOVL    #1, R6                              ; 0633
                64      65          1C    AE   29 0002D            CMPC3   SIZE, (R5), (R4)                    ;
                                          5C   1A 00032            BGTRU   18$                                 ;
                        56                01   D9 00034            SBWC    #1, R6                              ;
                                          57   11 00037            BRB     18$                                 ;
                        64                65   B1 00039 4$:        CMPW    (R5), (R4)                          ; 0635
                                          44   13 0003C 5$:        BEQL    14$                                 ;
                                          4D   18 0003E 6$:        BGEQ    17$                                 ; 0638
                                          46   11 00040            BRB     16$                                 ; 0640
                        64                65   B1 00042 7$:        CMPW    (R5), (R4)                          ; 0644
                                          08   11 00045            BRB     10$                                 ;
                        64                65   D1 00047 8$:        CMPL    (R5), (R4)                          ; 0653
                                          F0   11 0004A            BRB     5$                                  ;
                        64                65   D1 0004C 9$:        CMPL    (R5), (R4)                          ; 0662
                                          31   13 0004F 10$:       BEQL    14$                                 ;
                                          33   11 00051            BRB     15$                                 ; 0665
                        50          1C    AE   D0 00053 11$:       MOVL    SIZE, R0                            ; 0675
                        50                02   C4 00057            MULL2   #2, NIBBLES                         ;
                                          50   D7 0005A            DECL    NIBBLES                             ;
                64      65                50   35 0005C            CMPP3   NIBBLES, (R5), (R4)                 ; 0677
                                          57   DC 00060            MOVPSL  R7                                  ;
   57           57      02                02   EF 00062            EXTZV   #2, #2, R7, R7                      ;
                56      01                57   C3 00067            SUBL3   R7, #1, DUP                         ;
                                          23   11 0006B            BRB     18$                                 ;
          04    A4          04            A5   D1 0006D 12$:       CMPL    4(R5), 4(R4)                        ; 0684
                                          D3   13 00072            BEQL    8$                                  ;
                                          C8   11 00074            BRB     6$                                  ; 0697
          04    A4          04            A5   D1 00076 13$:       CMPL    4(R5), 4(R4)                        ; 0703
                                          09   12 0007B            BNEQ    15$                                 ;
                        64                65   D1 0007D            CMPL    (R5), (R4)                          ; 0705
                                          04   12 00080            BNEQ    15$                                 ;
                                          56   D4 00082 14$:       CLRL    DUP                                 ;
                                          0A   11 00084            BRB     18$                                 ;
                                          05   1E 00086 15$:       BGEQU   17$                                 ; 0713
                        56                01   CE 00088 16$:       MNEGL   #1, DUP                             ; 0715
                                          03   11 0008B            BRB     18$                                 ;
                        56                01   D0 0008D 17$:       MOVL    #1, DUP                             ; 0713
                                          56   D5 00090 18$:       TSTL    DUP                                 ; 0723
                                          05   12 00092            BNEQ    19$                                 ;
                        56                01   D0 00094            MOVL    #1, DUP                             ; 0725
                                          07   11 00097            BRB     20$                                 ;
```

```
                01              56  D1 00099 19$:   CMPL    DUP, #1                           ; 0727
                                02  12 0009C        BNEQ    20$                               ;
                                56  D4 0009E        CLRL    DUP                               ; 0729
                50              56  D0 000A0 20$:   MOVL    DUP, R0                           ; 0733
                                02  11 000A3        BRB     22$                               ;
                                50  D4 000A5 21$:   CLRL    R0                                ; 0735
                       00FC     8F  BA 000A7 22$:   POPR    #^M<R2,R3,R4,R5,R6,R7>            ;
                                    05 000AB        RSB                                       ;

; Routine Size:  172 bytes,    Routine Base:  _CONV$FAST_S + 0219
```

```
1 %SBTTL  'IN_SEGMENT'
1 ROUTINE IN_SEGMENT : CL$IN_SEGMENT =
1 !++
1 !
1 ! Functional Description:
1 !
1 !         Determines if the current position in the data record is in a key
1 !         segment or not
1 !
1 ! Calling Sequence:
1 !
1 !         IN_SEGMENT()
1 !
1 ! Input Parameters:
1 !         none
1 !
1 ! Implicit Inputs:
1 !
1 !         BYTE_OFS - R6   Offset into the data record
1 !         NEXT_SEG - R7   Start of the next segment
1 !         NEXT_LEN - R8   Length of next segment
1 !
1 ! Output Parameters:
1 !         none
1 !
1 ! Implicit Outputs:
1 !         none
1 !
1 ! Routine Value:
1 !
1 !         1 - Pointer is in a key segment
1 !         0 - Pointer is NOT in a key segment
1 !
1 ! Routines Called:
1 !         none
1 !
1 ! Side Effects:
1 !         none
1 !
1 !--
1
2      BEGIN
2
2      DEFINE_KEY_DESC;
2
2      EXTERNAL REGISTER
2              BYTE_OFS = 6,         ! Offset into the data record
2              NEXT_SEG = 7,         ! Start of the next segment
2              NEXT_LEN = 8;         ! Length of next segment
2
2      LOCAL        SEGMENT : SIGNED;
2
2      NEXT_SEG = .CONV$GW_OUT_REC_SIZ;
2      NEXT_LEN = 0;
2
2      SEGMENT = .KEY_DESC [ KEY$B_SEGMENTS ] - 1;
2
```

```
2            ! Check all of the segments
2            !
2            WHILE .SEGMENT GEQ 0
2            DO
3                BEGIN
3
3                ! See if we  are past a segment if so see if we are in it else
3                ! check the next segment
3                !
3                IF .BYTE_OFS GEQ .KEY_DESC [ ( 28 + ( .SEGMENT*2 )),WORD_U ]
3                THEN
4                    BEGIN
4
4                    ! See if we are in the segment.  If so return with the pointers
4                    !
5                    IF .BYTE_OFS LSS ( .KEY_DESC [ ( 44 + .SEGMENT ),BYTE_U ] +
5                                     .KEY_DESC [ ( 28 + ( .SEGMENT*2 )),WORD_U ] )
4                    THEN
5                        BEGIN
5                        NEXT_SEG = .KEY_DESC [ ( 28 + ( .SEGMENT*2 )),WORD_U ];
5                        NEXT_LEN = .KEY_DESC [ ( 44 + .SEGMENT ),BYTE_U ];
5                        RETURN 1
5                        END
4                    END
3                ELSE
3
3                    ! If this segment is closer than the last one then change pointers
3                    !
3                    IF .NEXT_SEG GTR .KEY_DESC [ ( 28 + ( .SEGMENT*2 )),WORD_U ]
3                    THEN
4                        BEGIN
4                        NEXT_SEG = .KEY_DESC [ ( 28 + ( .SEGMENT*2 )),WORD_U ];
4                        NEXT_LEN = .KEY_DESC [ ( 44 + .SEGMENT ),BYTE_U ]
3                        END;
3
3
3                ! Check the next segment
3                !
3                SEGMENT = .SEGMENT - 1
2                END;
2
2            RETURN 0
2
1            END;
```

```
                        52  DD 00000 IN_SEGMENT:
                                             PUSHL   R2                                      : 0737
        57      0000G   CF  32 00002         CVTWL   CONV$GW_OUT_REC_SIZ, NEXT_SEG           : 0788
                        58  D4 00007         CLRL    NEXT_LEN                                : 0789
        50         12   AB  9A 00009         MOVZBL  18(KEY_DESC), SEGMENT                   : 0791
                        50  D7 0000D 1$:     DECL    SEGMENT                                 :
                        33  19 0000F         BLSS    3$                                      : 0795
        51       1C AB40    3C 00011         MOVZWL  28(KEY_DESC)[SEGMENT], R1               : 0802
```

```
      51          56  D1 00016        CMPL    BYTE_OFS, R1                          ;
                  1A  19 00019        BLSS    2$                                    ;
      52     2C A04B  9A 0001B        MOVZBL  44(SEGMENT)[KEY_DESC], R2             ; 0809
      52          51  C0 00020        ADDL2   R1, R2                                ;
      52          56  D1 00023        CMPL    BYTE_OFS, R2                          ; 0808
                  E5  18 00026        BGEQ    1$                                    ;
      57          51  D0 00028        MOVL    R1, NEXT_SEG                          ; 0812
      58     2C A04B  9A 0002B        MOVZBL  44(SEGMENT)[KEY_DESC], NEXT_LEN       ; 0813
      50          01  D0 00030        MOVL    #1, R0                                ; 0814
                  11  11 00033        BRB     4$                                    ;
      51          57  D1 00035 2$:    CMPL    NEXT_SEG, R1                          ; 0821
                  D3  15 00038        BLEQ    1$                                    ;
      57          51  D0 0003A        MOVL    R1, NEXT_SEG                          ; 0824
      58     2C A04B  9A 0003D        MOVZBL  44(SEGMENT)[KEY_DESC], NEXT_LEN       ; 0825
                  C9  11 00042        BRB     1$                                    ; 0831
                  50  D4 00044 3$:    CLRL    R0                                    ; 0834
                  04  BA 00046 4$:    POPR    #^M<R2>                               ; 0836
                      05 00048        RSB                                           ;

; Routine Size:  73 bytes,    Routine Base:  _CONV$FAST_S + 02C5
```

```
%SBTTL  'COMPRESS_KEY'
GLOBAL ROUTINE  CONV$$COMPRESS_KEY : CL$JSB_REG_9 NOVALUE =
!++
!
! Functional Description:
!
!       Does primary data level key compression for prologue 3 files
!
! Calling Sequence:
!
!       CONV$$COMPRESS_KEY()
!
! Input Parameters:
!       none
!
! Implicit Inputs:
!       none
!
! Output Parameters:
!       none
!
! Implicit Outputs:
!       none
!
! Routine Value:
!       none
!
! Side Effects:
!       none
!
!--

    BEGIN

    DEFINE_CTX;
    DEFINE_KEY_DESC;

    LOCAL
        KEY             : REF VECTOR [ ,BYTE ],
        LENGTH,
        COUNT,
        RECORD_CTRL     : REF BLOCK [ ,BYTE ];

    ! The key is in different place depending on ref
    !
    IF .KEY_DESC [ KEY$B_KEYREF ] EQL 0
    THEN
        KEY = .CTX [ CTX$L_RCP ] + 13
    ELSE
        KEY = .CTX [ CTX$L_RCP ] + 4;

    ! Start with the full length of the key
    !
    LENGTH = .KEY_DESC [ KEY$B_KEYSZ ];
    COUNT = 0;

    ! If this is not the first record in the bucket then compress off the front
```

```
!
IF .CTX [ CTX$W_USE ] NEQU 0
THEN
    BEGIN

    BIND
        LASTKEY      = .CTX [ CTX$L_LKP ] : VECTOR [ ,BYTE ];

    ! Compaire with the last key
    !
    INCR I FROM 0 TO ( .LENGTH - 1 ) BY 1
    DO

        ! If this character is the same as the character in the last key
        ! then it can be compressed off.
        !
        IF .LASTKEY [ .I ] EQLU .KEY [ .I ]
        THEN
            COUNT = .COUNT + 1
        ELSE
            EXITLOOP;

    ! The key could be a little bit shoter
    !
    LENGTH = .LENGTH - .COUNT;

    ! If we truncated off bytes we must move what is left of the key in a little
    !
    CH$MOVE ( .LENGTH,KEY [ .COUNT ],KEY [ 0 ] )

    END;

! If there are some characters left do rear end truncation
!
IF .LENGTH GTR 1
THEN

    ! Start looking from the back to compress off trailing dups
    !
    DECR I FROM ( .LENGTH - 1 ) TO 1 BY 1
    DO

        ! If this character is a dup cut it off by shorting the length
        !
        IF .KEY [ .I - 1 ] EQLU .KEY [ .I ]
        THEN
            LENGTH = .LENGTH - 1
        ELSE
            EXITLOOP;

! Point to the control fields in front of the key
!
RECORD_CTRL = .KEY - 4;

! Set the length field in the record
!
RECORD_CTRL [ 2,BYTE_U ] = .LENGTH;
```

```
;  959      0951  2
;  960      0952  2        ! Set the count field in the record
;  961      0953  2        !
;  962      0954  2        RECORD_CTRL [ 3,BYTE_U ] = .COUNT;
;  963      0955  2
;  964      0956  2        ! Set the record size field in the record
;  965      0957  2        !
;  966      0958  2        RECORD_CTRL [ 0,WORD_U ] = .RECORD_CTRL [ 0,WORD_U ] -
;  967      0959  2                                     ( .KEY_DESC [ KEY$B_KEYSZ ] - .LENGTH );
;  968      0960  2
;  969      0961  2        ! Set the new size of the control part of the record if needed
;  970      0962  2        !
;  971      0963  2        CTX [ CTX$W_RCS ] = .CTX [ CTX$W_RCS ] -
;  972      0964  2                                     ( .KEY_DESC [ KEY$B_KEYSZ ] - .LENGTH );
;  973      0965  2
;  974      0966  2        RETURN
;  975      0967  2
;  976      0968  1        END;


                       01FC   8F   BB 00000 CONV$$COMPRESS_KEY::
                                                    PUSHR   #^M<R2,R3,R4,R5,R6,R7,R8>           ; 0838
                         15   AB   95 00004         TSTB    21(KEY_DESC)                        ; 0882
                              07   12 00007         BNEQ    1$                                  ;
   56      30    AA           0D   C1 00009         ADDL3   #13, 48(CTX), KEY                   ; 0884
                              05   11 0000E         BRB     2$                                  ;
   56      30    AA           04   C1 00010 1$:     ADDL3   #4, 48(CTX), KEY                    ; 0886
                 57      14   AB   9A 00015 2$:     MOVZBL  20(KEY_DESC), LENGTH                ; 0890
                              58   D4 00019         CLRL    COUNT                               ; 0891
                         2C   AA   B5 0001B         TSTW    44(CTX)                             ; 0895
                              1B   13 0001E         BEQL    6$                                  ;
                 50           01   CE 00020         MNEGL   #1, I                               ; 0910
                              0A   11 00023         BRB     4$                                  ;
               6046      3C BA40   91 00025 3$:     CMPB    @60(CTX)[I], (I)[KEY]               ;
                              06   12 0002B         BNEQ    5$                                  ;
                              58   D6 0002D         INCL    COUNT                               ; 0912
   F2            50           57   F2 0002F 4$:     AOBLSS  LENGTH, I, 3$                       ; 0910
                 57           58   C2 00033 5$:     SUBL2   COUNT, LENGTH                       ; 0918
   66          6846           57   28 00036         MOVC3   LENGTH, (COUNT)[KEY], (KEY)         ; 0922
                 01           57   D1 0003B 6$:     CMPL    LENGTH, #1                          ; 0928
                              12   15 0003E         BLEQ    9$                                  ;
                 50           57   D0 00040         MOVL    LENGTH, I                           ; 0938
                              0A   11 00043         BRB     8$                                  ;
               6046      FF A046   91 00045 7$:     CMPB    -1(I)[KEY], (I)[KEY]                ;
                              05   12 0004B         BNEQ    9$                                  ;
                              57   D7 0004D         DECL    LENGTH                              ; 0940
                 F3           50   F5 0004F 8$:     SOBGTR  I, 7$                               ; 0938
                 50      FC   A6   9E 00052 9$:     MOVAB   -4(R6), RECORD_CTRL                 ; 0946
           02    A0           57   90 00056         MOVB    LENGTH, 2(RECORD_CTRL)              ; 0950
           03    A0           58   90 0005A         MOVB    COUNT, 3(RECORD_CTRL)               ; 0954
                 51      14   AB   9A 0005E         MOVZBL  20(KEY_DESC), R1                    ; 0959
                 57           51   C2 00062         SUBL2   R1, R7                              ;
                 60           57   A0 00065         ADDW2   R7, (RECORD_CTRL)                   ;
           38    AA           57   A0 00068         ADDW2   R7, 56(CTX)                         ; 0964
```

```
                  01FC   8F  BA 0006C        POPR    #^M<R2,R3,R4,R5,R6,R7,R8>          ; 0968
                             05 00070        RSB                                        ;

; Routine Size:  113 bytes,    Routine Base:  _CONV$FAST_S + 030E
```

```
1 %SBTTL  'COMPRESS_INDEX'
1 GLOBAL ROUTINE  CONV$$COMPRESS_INDEX : CL$JSB_REG_9 NOVALUE =
1 !++
1 !
1 ! Functional Description:
1 !
1 !          Does prologue 3 index compression on the index level record
1 !
1 ! Calling Sequence:
1 !
1 !          CONV$$COMPRESS_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 !          none
1 !
1 ! Side Effects:
1 !          none
1 !
1 !--
1
2     BEGIN
2
2     DEFINE_CTX;
2     DEFINE_BUCKET;
2     DEFINE_KEY_DESC;
2
2     LOCAL
2         LENGTH,
2         COUNT,
2         RECORD_CTRL       : REF BLOCK [ ,BYTE ];
2
2     BIND
2         KEY       = .CTX [ CTX$L_RCP ] + 2 : VECTOR [ ,BYTE ];
2
2     ! Init. counts
2     !
2     LENGTH = .KEY_DESC [ KEY$B_KEYSZ ];
2     COUNT = 0;
2
2     ! If this is not the first record in the bucket then do front compression
2     !
2     IF .CTX [ CTX$W_USE ] NEQU 0
2     THEN
3         BEGIN
3
```

```
	BIND
	    LASTKEY	= .CTX [ CTX$L_LKP ] : VECTOR [ ,BYTE ];

	! Compare this key with the last one
	!
	INCR I FROM 0 TO ( .LENGTH - 1 ) BY 1
	DO

	    ! Count the characters that are the same
	    !
	    IF .LASTKEY [ .I ] EQLU .KEY [ .I ]
	    THEN
		COUNT = .COUNT + 1
	    ELSE
		EXITLOOP;

	! The length of the key is a bit shorter now
	!
	LENGTH = .LENGTH - .COUNT

	END;

    ! Copy this key so that we have a last key to look at
    !
    CH$MOVE ( .KEY_DESC [ KEY$B_KEYSZ ],
			.CTX [ CTX$L_RDP ] + 2,
			.CTX [ CTX$L_LKP ] );

    ! If we shortened the key from the front move it
    !
    CH$MOVE ( .LENGTH,KEY [ .COUNT ],KEY [ 0 ] );

    ! If there are some characters left do rear end truncation
    !
    IF .LENGTH GTR 1
    THEN

	! Start looking from the back to compress off trailing dups
	!
	DECR I FROM ( .LENGTH - 1 ) TO 1 BY 1
	DO

	    ! If this character is a dup cut it off by shorting the length
	    !
	    IF .KEY [ .I - 1 ] EQLU .KEY [ .I ]
	    THEN
		LENGTH = .LENGTH - 1
	    ELSE
		EXITLOOP;

    ! Set up the fields in the record
    !
    RECORD_CTRL = .CTX [ CTX$L_RDP ];

    ! Set the record size field in the record
    !
    RECORD_CTRL [ 0,BYTE_U ] = .LENGTH;
```

```
2
2        ! Set the compression count
2        !
2        RECORD_CTRL [ 1,BYTE_U ] = .COUNT;
2
2        ! Set the new size of the data part of the record
2        !
2        CTX [ CTX$W_RDS ] = .LENGTH + 2;
2
2        RETURN
2
1        END;


                    01FC   8F    BB 00000 CONV$$COMPRESS_INDEX::
                                          PUSHR   #^M<R2,R3,R4,R5,R6,R7,R8>        ; 0970
             58      34    AA    D0 00004         MOVL    52(CTX), R8                      ; 1013
             57      02    A8    9E 00008         MOVAB   2(R8), R7                        ;
             56      14    AB    9A 0000C         MOVZBL  20(KEY_DESC), LENGTH             ; 1017
                           7E    D4 00010         CLRL    COUNT                            ; 1018
                     2C    AA    B5 00012         TSTW    44(CTX)                          ; 1022
                           16    13 00015         BEQL    4$                               ;
             50            01    CE 00017         MNEGL   #1, I                            ; 1031
                           0A    11 0001A         BRB     2$                               ;
           6047      3C  BA40    91 0001C 1$:     CMPB    @60(CTX)[I], (I)[R7]             ; 1036
                           06    12 00022         BNEQ    3$                               ;
                           6E    D6 00024         INCL    COUNT                            ; 1038
       F2    50            56    F2 00026 2$:     AOBLSS  LENGTH, I, 1$                    ; 1036
             56            6E    C2 0002A 3$:     SUBL2   COUNT, LENGTH                    ; 1044
             50      14    AB    9A 0002D 4$:     MOVZBL  20(KEY_DESC), R0                 ; 1050
  3C   BA    67            50    28 00031         MOVC3   R0, (R7), @60(CTX)               ; 1052
       67         00 BE47      56    28 00036     MOVC3   LENGTH, @COUNT[R7], (R7)         ; 1056
             01            56    D1 0003C         CMPL    LENGTH, #1                       ; 1060
                           12    15 0003F         BLEQ    7$                               ;
             50            56    D0 00041         MOVL    LENGTH, I                        ; 1065
                           0A    11 00044         BRB     6$                               ;
           6047      FF  A047    91 00046 5$:     CMPB    -1(I)[R7], (I)[R7]               ; 1070
                           05    12 0004C         BNEQ    7$                               ;
                           56    D7 0004E         DECL    LENGTH                           ; 1072
             F3            50    F5 00050 6$:     SOBGTR  I, 5$                            ; 1070
             50            58    D0 00053 7$:     MOVL    R8, RECORD_CTRL                  ; 1078
             60            56    90 00056         MOVB    LENGTH, (RECORD_CTRL)            ; 1082
        01   A0            8E    90 00059         MOVB    COUNT, 1(RECORD_CTRL)            ; 1086
  3A   AA    56            02    A1 0005D         ADDW3   #2, LENGTH, 58(CTX)              ; 1090
             5E            03    C0 00062         ADDL2   #3, SP                           ; 1094
                    01FC   8F    BA 00065         POPR    #^M<R2,R3,R4,R5,R6,R7,R8>        ;
                                 05 00069         RSB                                      ;

; Routine Size:  106 bytes,    Routine Base:  _CONV$FAST_S + 037F
```

```
1 %SBTTL  'MAKE_INDEX'
1 GLOBAL ROUTINE  CONV$$MAKE_INDEX : CL$JSB_REG_9 NOVALUE =
1 !++
1 !
1 ! Functional Description:
1 !
1 !       Makes an index record for the current record being processed
1 !       It also copys the key into the last key buffer
1 !
1 ! Calling Sequence:
1 !
1 !       CONV$$MAKE_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 !       none
1 !
1 ! Side Effects:
1 !       none
1 !
1 !--
1
2     BEGIN
2
2     DEFINE_CTX;
2     DEFINE_KEY_DESC;
2
2     LOCAL
2         OFFSET,
2         IDX_PTR,
2         CTX_P1   : REF BLOCK [ ,BYTE ];
2
2     ! We need the contex block from the level above
2     !
2     CTX_P1 = .CTX + CTX$K_BLN;
2
2     ! Account for control bytes if any
2     !
2     IF .KEY_DESC [ KEY$V_IDX_COMPR ]
2     THEN
2         OFFSET = 2
2     ELSE
2         OFFSET = 0;
2
2     ! Point to the destination
2     !
```

```
2       IDX_PTR = .CTX_P1 [ CTX$L_RDP ] + .OFFSET;
2
2       ! Copy the current key into the index buffer.
2       !
2       IF .KEY_DESC [ KEY$B_KEYREF ] EQLU 0
2       THEN
3           BEGIN
3
3           BIND
3               SEG_SIZE     = .KEY_DESC + 44 : VECTOR [ ,BYTE ],
3               SEG_POSI     = .KEY_DESC + 28 : VECTOR [ ,WORD ];
3
3           LOCAL       REC_PTR;
3
3           ! Copy the key into the index key buffer
3           !
3           INCR I FROM 0 TO ( .KEY_DESC [ KEY$B_SEGMENTS ] - 1 ) BY 1
3           DO
4               BEGIN
4
4               ! Point to the correct spot in the source string
4               !
4               REC_PTR = .CONV$GL_RECORD_PTR + .SEG_POSI [ .I ];
4
4               CH$MOVE( .SEG_SIZE [ .I ],.REC_PTR,.IDX_PTR );
4
4               ! Update the destination
4               !
4               IDX_PTR = .IDX_PTR + .SEG_SIZE [ .I ]
4
4               END
3           END
2       ELSE
2           CH$MOVE( .KEY_DESC [ KEY$B_KEYSZ ],
2                   .CONV$GL_DUP_BUF,
2                   .IDX_PTR );
2
2       ! Copy the key from the index record into last key buffer
2       !
2       CH$MOVE( .KEY_DESC [ KEY$B_KEYSZ ],
2               .CTX_P1 [ CTX$L_RDP ] + .OFFSET,
2               .CTX [ CTX$L_LKP ] );
2
2       ! Set the index record size
2       !
2       CTX_P1 [ CTX$W_RDS ] = .KEY_DESC [ KEY$B_KEYSZ ] + .OFFSET;
2
2       ! Set the index control size
2       !
2       IF .CONV$GB_PROL_V3
2       THEN
2
2           ! The record control size is simply the size of the
2           ! VBN pointer to the bucket
2           !
2           CTX_P1 [ CTX$W_RCS ] = .CTX [ CTX$V_VBN ] + 2
2
```

```
; 1219      1209  2       ELSE
; 1220      1210  2
; 1221      1211  2            ! The record control size is the size of the
; 1222      1212  2            ! VBN pointer to the bucket plus one byte of control info
; 1223      1213  2            !
; 1224      1214  2            CTX_P1 [ CTX$W_RCS ] = .CTX [ CTX$V_VBN ] + 3;
; 1225      1215  2
; 1226      1216  2       RETURN
; 1227      1217  2
; 1228      1218  1       END;


                                  03FC   8F   BB 00000 CONV$$MAKE_INDEX::
                                                              PUSHR    #^M<R2,R3,R4,R5,R6,R7,R8,R9>                     ; 1096
                          5E           0C   C2 00004          SUBL2    #12, SP                                          ;
                          56      5C   AA   9E 00007          MOVAB    92(R10), CTX_P1                                  ; 1140
             05      10   AB           03   E1 0000B          BBC      #3, 16(KEY_DESC), 1$                             ; 1144
                          59           02   D0 00010          MOVL     #2, OFFSET                                       ; 1146
                                       02   11 00013          BRB      2$                                               ;
                                       59   D4 00015 1$:      CLRL     OFFSET                                           ; 1148
             04           AE    34 B049   9E 00017 2$:      MOVAB    @52(CTX_P1)[OFFSET], 4(SP)                       ; 1152
                          58      04   AE   D0 0001D          MOVL     4(SP), IDX_PTR                                   ;
                                  15   AB   95 00021          TSTB     21(KEY_DESC)                                     ; 1156
                                       2D   12 00024          BNEQ     5$                                               ;
                          6E      12   AB   9A 00026          MOVZBL   18(KEY_DESC), (SP)                               ; 1168
                          57           01   CE 0002A          MNEGL    #1, I                                            ; 1174
                                       1E   11 0002D          BRB      4$                                               ;
                          50    1C AB47   3C 0002F 3$:      MOVZWL   28(KEY_DESC)[I], R0                              ;
                  08      AE  0000GDF40   9E 00034          MOVAB    @CONV$GL_RECORD_PTR[R0], REC_PTR                 ;
                          50    2C AB47   9A 0003B          MOVZBL   44(KEY_DESC)[I], R0                              ; 1176
             68      08   BE           50   28 00040          MOVC3    R0, @REC_PTR, (IDX_PTR)                          ;
                          50    2C AB47   9A 00045          MOVZBL   44(KEY_DESC)[I], R0                              ; 1180
                          58           50   C0 0004A          ADDL2    R0, IDX_PTR                                      ;
             DE           57           6E   F2 0004D 4$:      AOBLSS   (SP), I, 3$                                      ;
                                       0A   11 00051          BRB      6$                                               ; 1168
                          50      14   AB   9A 00053 5$:      MOVZBL   20(KEY_DESC), R0                                 ; 1185
             68   0000G   DF           50   28 00057          MOVC3    R0, @CONV$GL_DUP_BUF, (IDX_PTR)                  ; 1187
                          50      14   AB   9A 0005D 6$:      MOVZBL   20(KEY_DESC), R0                                 ; 1191
        3C   BA      04   BE           50   28 00061          MOVC3    R0, @4(SP), @60(CTX)                             ; 1193
                          50      14   AB   9A 00067          MOVZBL   20(KEY_DESC), R0                                 ; 1197
        3A   A6           50           59   A1 0006B          ADDW3    OFFSET, R0, 58(CTX_P1)                           ;
                          0C   0000G   CF   E9 00070          BLBC     CONV$GB_PROL_V3, 7$                              ; 1201
   50             6A      02           05   EF 00075          EXTZV    #5, #2, (CTX), R0                                ; 1207
        38   A6           50           02   A1 0007A          ADDW3    #2, R0, 56(CTX_P1)                               ;
                                       0A   11 0007F          BRB      8$                                               ;
   50             6A      02           05   EF 00081 7$:      EXTZV    #5, #2, (CTX), R0                                ; 1214
        38   A6           50           03   A1 00086          ADDW3    #3, R0, 56(CTX_P1)                               ;
                          5E           0C   C0 0008B 8$:      ADDL2    #12, SP                                          ; 1218
                                  03FC   8F   BA 0008E          POPR     #^M<R2,R3,R4,R5,R6,R7,R8,R9>                     ;
                                            05 00092          RSB                                                       ;

; Routine Size:  147 bytes,    Routine Base:  _CONV$FAST_S + 03E9
```

```
%SBTTL  'WRITE_VBN'
GLOBAL ROUTINE  CONV$$WRITE_VBN : CL$JSB_REG_9 NOVALUE =
!++
!
! Functional Description:
!
!       Writes the vbn pointer into the index record, also sets the control
!       bytes describing the vbn pointer size
!
! Calling Sequence:
!
!       CONV$$WRITE_VBN
!
! Input Parameters:
!       none
!
! Implicit Inputs:
!       none
!
! Output Parameters:
!       none
!
! Implicit Outputs:
!       none
!
! Routine Value:
!       none
!
! Side Effects:
!       none
!
!--

    BEGIN

    DEFINE_CTX;
    DEFINE_KEY_DESC;

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

    CTX_P1 = .CTX + CTX$K_BLN;

    RECORD_CTRL = .CTX_P1 [ CTX$L_RCP ];

    ! Get the size in bits of the vbn
    !
    VBN_SIZE = ( .CTX [ CTX$V_VBN ] + 2 ) * 8;

    ! The control bytes are different for prologue 3 files
    !
    IF .CONV$GB_PROL_V3
    THEN
        RECORD_CTRL [ 0,0,.VBN_SIZE,0 ] = .CTX [ CTX$L_CURRENT_VBN ]
    ELSE
```

```
; 1287          1276 3          BEGIN
; 1288          1277 3          RECORD_CTRL [ IRC$B_CONTROL ] = .CTX [ CTX$V_VBN ];
; 1289          1278 3          RECORD_CTRL [ 1,0,.VBN_SIZE,0 ] = .CTX [ CTX$L_CURRENT_VBN ]
; 1290          1279 2          END;
; 1291          1280 2
; 1292          1281 2       RETURN
; 1293          1282 2
; 1294          1283 1       END;



                                          52  DD 00000 CONV$$WRITE_VBN::
                                                                PUSHL   R2                                           ; 1220
                              50     5C   AA  9E 00002          MOVAB   92(R10), CTX_P1                              ; 1262
                              51     30   A0  D0 00006          MOVL    48(CTX_P1), RECORD_CTRL                      ; 1264
         52          6A       02          05  EF 0000A          EXTZV   #5, #2, (CTX), R2                            ; 1268
                     50       52          03  78 0000F          ASHL    #3, R2, VBN_SIZE                             ;
                              50          10  C0 00013          ADDL2   #16, VBN_SIZE                                ;
                              08   0000G  CF  E9 00016          BLBC    CONV$GB_PROL_V3, 1$                          ; 1272
         61          50       00     08   AA  F0 0001B          INSV    8(CTX), #0, VBN_SIZE, (RECORD_CTRL)          ; 1274
                                          0A  11 00021          BRB     2$                                           ;
                              61          52  90 00023 1$:      MOVB    R2, (RECORD_CTRL)                            ; 1277
     01  A1          50       00     08   AA  F0 00026          INSV    8(CTX), #0, VBN_SIZE, 1(RECORD_CTRL)         ; 1278
                                          04  BA 0002D 2$:      POPR    #^M<R2>                                      ; 1283
                                              05 0002F          RSB                                                  ;

; Routine Size:  48 bytes,    Routine Base:  _CONV$FAST_S + 047C
```

```
%SBTTL  'COPY_KEY'
GLOBAL ROUTINE  CONV$$COPY_KEY ( OFFSET ) : CL$COPY_KEY NOVALUE =
!++
!
! Functional Description:
!
!       Copies the data record for secondary index into the proper
!       data record buffer and sets RCS
!
! Calling Sequence:
!
!       CONV$$COPY_KEY()
!
! Input Parameters:
!       none
!
! Implicit Inputs:
!       none
!
! Output Parameters:
!       none
!
! Implicit Outputs:
!       none
!
! Routine Value:
!       none
!
! Side Effects:
!       none
!
!--

    BEGIN

    DEFINE_CTX;
    DEFINE_KEY_DESC;
    DEFINE_RECORD_CTRL;

    ! If this is a prologue 3 file then the offset is either 2 or 4
    ! depending on compression
    !
    IF .CONV$GB_PROL_V3
    THEN
        BEGIN

        IF .KEY_DESC [ KEY$V_KEY_COMPR ]
        THEN
            OFFSET = 4
        ELSE
            OFFSET = 2;

        ! Set the size of the record here sice it is known, for prologue 1
        ! it is set in load_secondary
        !
        RECORD_CTRL [ 0,WORD_U ] = .KEY_DESC [ KEY$B_KEYSZ ] + .OFFSET - 2

```

```
; 1353      1341  2            END;
; 1354      1342  2
; 1355      1343  2        ! Move the key to the record control buffer
; 1356      1344  2        !
; 1357      1345  2        CH$MOVE( .KEY_DESC [ KEY$B_KEYSZ ],
; 1358      1346  2                 .CONV$GL_RFA_BUFFER + 6,
; 1359      1347  2                 .CTX [ CTX$L_RCP ] + .OFFSET );
; 1360      1348  2
; 1361      1349  2        ! Set the size
; 1362      1350  2        !
; 1363      1351  2        CTX [ CTX$W_RCS ] = .KEY_DESC [ KEY$B_KEYSZ ] + .OFFSET;
; 1364      1352  2
; 1365      1353  2        RETURN
; 1366      1354  2
; 1367      1355  1        END;


                                       3C  BB 00000 CONV$$COPY_KEY::
                                                        PUSHR   #^M<R2,R3,R4,R5>                       ; 1285
                          1B    0000G  CF  E9 00002     BLBC    CONV$GB_PROL_V3, 3$                    ; 1326
             06      10   AB           06  E1 00007     BBC     #6, 16(KEY_DESC), 1$                   ; 1330
                     14   AE           04  D0 0000C     MOVL    #4, OFFSET                             ; 1332
                                       04  11 00010     BRB     2$                                     ;
                     14   AE           02  D0 00012 1$: MOVL    #2, OFFSET                             ; 1334
                          50     14    AB  9A 00016 2$: MOVZBL  20(KEY_DESC), R0                       ; 1339
                          50     14    AE  C0 0001A     ADDL2   OFFSET, R0                             ;
             68           50           02  A3 0001E     SUBW3   #2, R0, (RECORD_CTRL)                  ;
                          52     14    AB  9A 00022 3$: MOVZBL  20(KEY_DESC), R2                       ; 1345
                          50  0000G    CF  D0 00026     MOVL    CONV$GL_RFA_BUFFER, R0                 ; 1346
             51      30   AA     14    AE  C1 0002B     ADDL3   OFFSET, 48(CTX), R1                    ; 1347
             61      06   A0           52  28 00031     MOVC3   R2, 6(R0), (R1)                        ;
                          50     14    AB  9A 00036     MOVZBL  20(KEY_DESC), R0                       ; 1351
         38  AA           50     14    AE  A1 0003A     ADDW3   OFFSET, R0, 56(CTX)                    ;
                                       3C  BA 00040     POPR    #^M<R2,R3,R4,R5>                       ; 1355
                                       05  00042        RSB                                            ;

; Routine Size:  67 bytes,    Routine Base:  _CONV$FAST_S + 04AC
```

```
1 %SBTTL  'CREATE_HIGH_KEY'
1 GLOBAL ROUTINE  CONV$$CREATE_HIGH_KEY   : CL$JSB_REG_9 NOVALUE =
1 !++
1 !
1 ! Functional Description:
1 !
1 !       Creates a high key vaule index for the current key of reference
1 !       and data type
1 !
1 ! Calling Sequence:
1 !
1 !       CONV$$CREATE_HIGH_KEY()
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 ! Side Effects:
1 !       none
1 !
1 !--
1
2     BEGIN
2
2     DEFINE_CTX;
2     DEFINE_BUCKET;
2     DEFINE_KEY_DESC;
2
2     BUILTIN
2         CVTLP;
2
2     LOCAL
2         CTX_P1          : REF BLOCK [ ,BYTE ],
2         RECORD_DATA     : REF BLOCK [ ,BYTE ];
2
2     ! Set up the control info for the index record
2     !
2     CONV$$WRITE_VBN();
2
2     ! The record we are making is actually for the level above
2     !
2     CTX_P1 = .CTX + CTX$K_BLN;
2
2     ! Get the start where the key should go
2     !
2     IF .KEY_DESC [ KEY$V_IDX_COMPR ]
```

```
THEN
    RECORD_DATA = 2
ELSE
    RECORD_DATA = 0;

! We know what the size of the finished record is now
!
CTX_P1 [ CTX$W_RDS ] = .KEY_DESC [ KEY$B_KEYSZ ] + .RECORD_DATA;

! Actually point to the record
!
RECORD_DATA = .RECORD_DATA + .CTX_P1 [ CTX$L_RDP ];

! Loop for all of the segments
!
INCR I FROM 0 TO ( .KEY_DESC [ KEY$B_SEGMENTS ] - 1 ) BY 1
DO
    BEGIN

    ! Select the data type
    !
    CASE .KEY_DESC [ KEY$B_DATATYPE ] FROM KEY$C_STRING TO KEY$C_MAX_DATA OF
    SET
        [ KEY$C_STRING ]  : BEGIN

                            LOCAL HIGH_VALUE : BYTE;

                            HIGH_VALUE = %X'FF';
                            CH$FILL ( .HIGH_VALUE,
                                      .KEY_DESC [ ( 44 + .I ),BYTE_U ],
                                      .RECORD_DATA )
                            END;

        [ KEY$C_SGNWORD ]   : RECORD_DATA [ 0,WORD_S ] = %X'7FFF';

        [ KEY$C_UNSGNWORD ] : RECORD_DATA [ 0,WORD_U ] = %X'FFFF';

        [ KEY$C_SGNLONG ]   : RECORD_DATA [ 0,LONG_S ] = %X'7FFFFFFF';

        [ KEY$C_UNSGNLONG ] : RECORD_DATA [ 0,LONG_U ] = %X'FFFFFFFF';

        [ KEY$C_PACKED ]  : BEGIN

                            LOCAL   SIZE,
                                    HIGH_VALUE : BYTE;

                            SIZE = .KEY_DESC [ ( 44 + .I ),BYTE_U ] - 1;
                            HIGH_VALUE = %X'99';

                            CH$FILL ( .HIGH_VALUE,
                                      .SIZE,
                                      .RECORD_DATA );

                            RECORD_DATA [ .SIZE,BYTE_U ] = %X'9C'

                            END;
```

```
; 1483      1470 4              [ KEY$C_SGNQUAD ]   : BEGIN
; 1484      1471 4                                    RECORD_DATA [ 4,LONG_S ] = %X'7FFFFFFF';
; 1485      1472 4                                    RECORD_DATA [ 0,LONG_U ] = %X'FFFFFFFF'
; 1486      1473 3                                    END;
; 1487      1474 3
; 1488      1475 4              [ KEY$C_UNSGNQUAD ] : BEGIN
; 1489      1476 4                                    RECORD_DATA [ 4,LONG_U ] = %X'FFFFFFFF';
; 1490      1477 4                                    RECORD_DATA [ 0,LONG_U ] = %X'FFFFFFFF'
; 1491      1478 3                                    END;
; 1492      1479 3
; 1493      1480 3          TES;
; 1494      1481 3
; 1495      1482 3          ! Update the pointer (add the segment length)
; 1496      1483 3          !
; 1497      1484 3          RECORD_DATA = .RECORD_DATA + .KEY_DESC [ ( 44 + .I ),BYTE_U ]
; 1498      1485 3
; 1499      1486 2          END;
; 1500      1487 2
; 1501      1488 2      RETURN
; 1502      1489 2
; 1503      1490 1      END;



                                   01FC    8F   BB 00000 CONV$$CREATE_HIGH_KEY::
                                                                  PUSHR   #^M<R2,R3,R4,R5,R6,R7,R8>                     ; 1357
                            5E           04   C2 00004            SUBL2   #4, SP                                         ;
                                         84   10 00007            BSBB    CONV$$WRITE_VBN                                ; 1404
                            50     5C    AA   9E 00009            MOVAB   92(R10), CTX_P1                                ; 1408
              05       10   AB           03   E1 0000D            BBC     #3, 16(KEY_DESC), 1$                           ; 1412
                            56           02   D0 00012            MOVL    #2, RECORD_DATA                                ; 1414
                                         02   11 00015            BRB     2$                                             ;
                                         56   D4 00017 1$:        CLRL    RECORD_DATA                                    ; 1416
                            51     14    AB   9A 00019 2$:        MOVZBL  20(KEY_DESC), R1                               ; 1420
         3A     A0          51           56   A1 0001D            ADDW3   RECORD_DATA, R1, 58(CTX_P1)                    ;
                            56     34    A0   C0 00022            ADDL2   52(CTX_P1), RECORD_DATA                        ; 1424
                            6E     12    AB   9A 00026            MOVZBL  18(KEY_DESC), (SP)                             ; 1428
                            57           01   CE 0002A            MNEGL   #1, I                                          ; 1434
                                         6B   11 0002D            BRB     14$                                            ;
                07          00     11    AB   8F 0002F 3$:        CASEB   17(KEY_DESC), #0, #7                           ;
    002C        0027        0020       0010      00034 4$:        .WORD   5$-4$,=                                        ;
    0057        004D        0035       005B      0003C                    6$-4$,-                                        ;
                                                                          7$-4$,-                                        ;
                                                                          8$-4$,-                                        ;
                                                                          12$-4$,-                                       ;
                                                                          9$-4$,-                                        ;
                                                                          10$-4$,-                                       ;
                                                                          11$-4$                                         ;
                            51           01   8E 00044 5$:        MNEGB   #1, HIGH_VALUE                                 ; 1440
                            50     2C  A74B   9A 00047            MOVZBL  44(I)[KEY_DESC], R0                            ; 1442
    50          51          6E           00   2C 0004C            MOVC5   #0, (SP), HIGH_VALUE, R0, (RECORD_DATA)        ; 1443
                                         66      00051                                                                   ;
                                         3E   11 00052            BRB     13$                                            ; 1441
                            66   7FFF    8F   B0 00054 6$:        MOVW    #32767, (RECORD_DATA)                          ; 1446
                                         37   11 00059            BRB     13$                                            ;
```

```
                                 66            01  AE 0005B 7$:      MNEGW   #1, (RECORD_DATA)                              ; 1448
                                               32  11 0005E          BRB     13$                                            ;
                                 66 7FFFFFFF   8F  D0 00060 8$:      MOVL    #2147483647, (RECORD_DATA)                     ; 1450
                                               29  11 00067          BRB     13$                                            ;
                                 58     2C A74B 9A 00069 9$:      MOVZBL  44(I)[KEY_DESC], SIZE                          ; 1459
                                               58  D7 0006E          DECL    SIZE                                           ;
                                 50      99    8F  90 00070          MOVB    #-103, HIGH_VALUE                              ; 1460
        58          50           6E            00  2C 00074          MOVC5   #0, (SP), HIGH_VALUE, SIZE, (RECORD_DATA)      ; 1464
                                               66     00079                                                                 ;
                               6846      9C    8F  90 0007A          MOVB    #-100, (SIZE)[RECORD_DATA]                     ; 1466
                                               11  11 0007F          BRB     13$                                            ;
                           04    A6 7FFFFFFF   8F  D0 00081 10$:     MOVL    #2147483647, 4(RECORD_DATA)                    ; 1471
                                               04  11 00089          BRB     12$                                            ; 1472
                           04    A6            01  CE 0008B 11$:     MNEGL   #1, 4(RECORD_DATA)                             ; 1476
                                 66            01  CE 0008F 12$:     MNEGL   #1, (RECORD_DATA)                              ; 1477
                                 50     2C A74B 9A 00092 13$:     MOVZBL  44(I)[KEY_DESC], R0                            ; 1484
                                 56            50  C0 00097          ADDL2   R0, RECORD_DATA                                ;
                    91           57            6E  F2 0009A 14$:     AOBLSS  (SP), I, 3$                                    ;
                                 5E            04  C0 0009E          ADDL2   #4, SP                                         ; 1490
                                      01FC     8F  BA 000A1          POPR    #^M<R2,R3,R4,R5,R6,R7,R8>                      ;
                                               05     000A5          RSB                                                    ;

; Routine Size:  166 bytes,    Routine Base:  _CONV$FAST_S + 04EF


; 1504           1491  1
; 1505           1492  0 END      ELUDOM




;                                    PSECT SUMMARY
;
;       Name                        Bytes                          Attributes
;
;  _CONV$FAST_S                         1429  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        25         0        1000          00:01.9
;  _$255$DUA28:[CONV.SRC]CONVERT.L32;1       165        26        15          17          00:00.2



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

```
;                                  COMMAND QUALIFIERS

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

; Size:           1429 code + 0 data bytes
; Run Time:          00:34.3
; Elapsed Time:      01:46.2
; Lines/CPU Min:      2610
; Lexemes/CPU-Min: 17160
; Memory Used:  181 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

RECLDCL
LIS

RECLREC
LIS

RECLCTRL
LIS

RECLRMSIO
LIS

CONVMSG
LIS

CONVMAIN
LIS

CONVSORT
LIS

CONVVEC
LIS